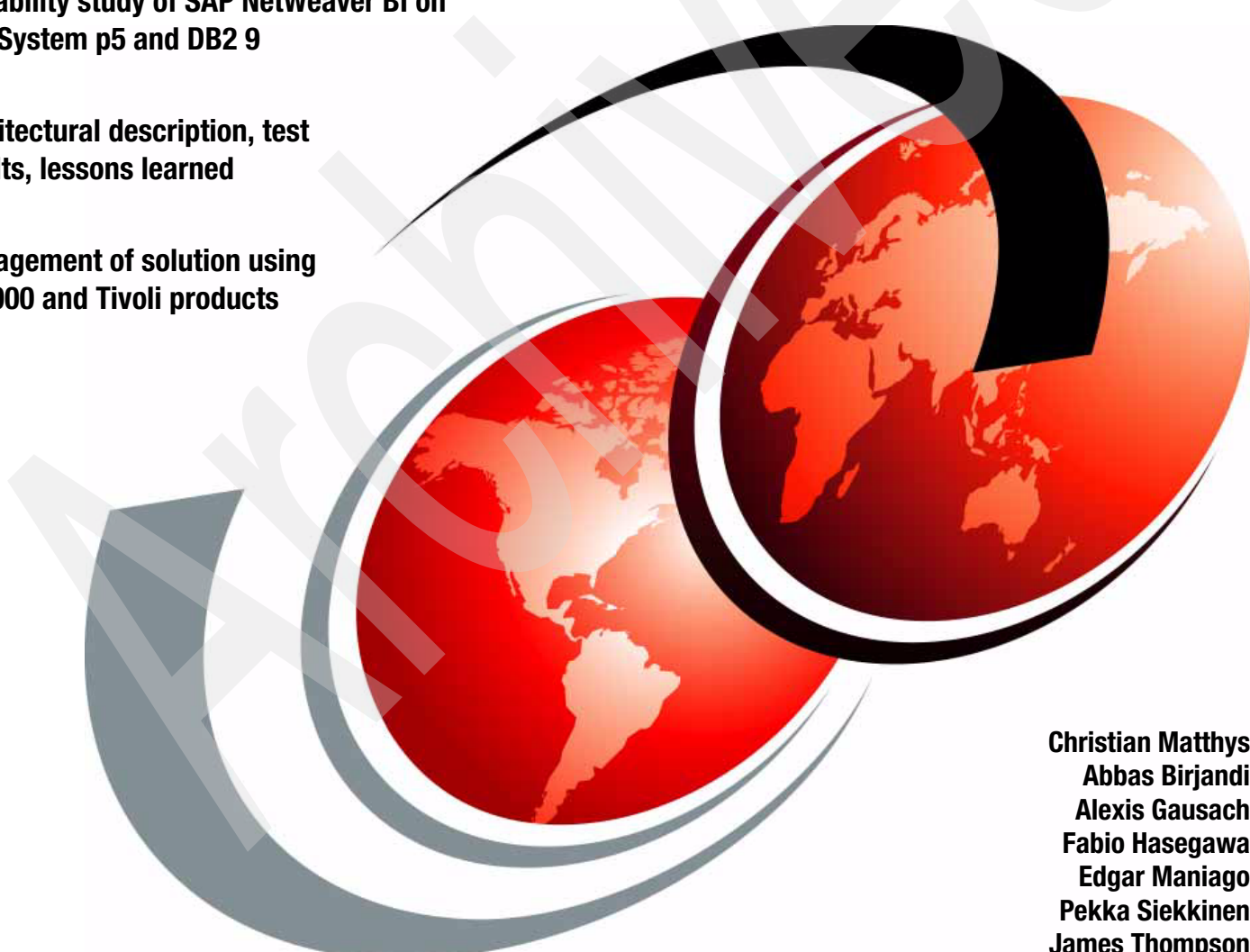


Infrastructure Solutions: Design, Manage, and Optimize a 60 TB SAP NetWeaver Business Intelligence Data Warehouse

Scalability study of SAP NetWeaver BI on
IBM System p5 and DB2 9

Architectural description, test
results, lessons learned

Management of solution using
DS8000 and Tivoli products



Christian Matthys
Abbas Birjandi
Alexis Gausach
Fabio Hasegawa
Edgar Maniago
Pekka Siekkinen
James Thompson

Redbooks



International Technical Support Organization

**Infrastructure Solutions: Design, Manage, and
Optimize a 60 TB SAP NetWeaver Business Intelligence
Data Warehouse**

Januray 2008

Archived

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

Archived

First Edition (Januray 2008)

This edition describes tests done in an AIX 5.3, DB2 9, and SAP NetWeaver BI 3.5 environment with IBM System p5 and using Tivoli Storage Management and Tivoli data Protection products for storage management.

© Copyright International Business Machines Corporation 2008. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

| | |
|--|------|
| Notices | vii |
| Trademarks | viii |
| Preface | ix |
| The team that wrote this book | x |
| Become a published author | xi |
| Comments welcome | xii |
| Chapter 1. Project overview: business objectives, architecture, infrastructure, and results | 1 |
| 1.1 The scope of the project | 2 |
| 1.1.1 The business context and the customer strategy | 2 |
| 1.1.2 Test objectives | 2 |
| 1.1.3 The required tests | 3 |
| 1.1.4 The infrastructure | 5 |
| 1.1.5 The performance tools | 9 |
| 1.2 The online KPIs | 13 |
| 1.2.1 The progressive tests | 13 |
| 1.2.2 Objectives | 16 |
| 1.2.3 The KPI-G results | 17 |
| 1.2.4 Trends | 37 |
| 1.3 The infrastructure KPIs | 42 |
| 1.4 Data compression | 44 |
| Chapter 2. The SAP NetWeaver BI perspective | 47 |
| 2.1 SAP NetWeaver BI overview | 48 |
| 2.1.1 The SAP NetWeaver BI information model | 48 |
| 2.1.2 InfoCubes and the extended star schema | 49 |
| 2.1.3 The data flow in SAP NetWeaver BI | 52 |
| 2.2 Our project environment | 53 |
| 2.2.1 SAP NetWeaver application servers overview | 54 |
| 2.2.2 The SAP data model | 55 |
| 2.2.3 The load process | 56 |
| 2.2.4 The aggregate rollup process | 62 |
| 2.2.5 The query process | 65 |
| 2.3 Options and parameters discussions | 72 |
| 2.3.1 Parameters affecting the load process | 72 |
| 2.3.2 Optimizing the load process | 76 |
| 2.3.3 Parameters affecting the aggregate rollup process | 81 |
| 2.3.4 Optimizing the aggregate rollup process | 82 |
| 2.3.5 Parameters affecting the query process | 88 |
| 2.3.6 Optimizing the query process | 89 |
| 2.3.7 Impact of query and aggregate processes running in parallel | 106 |
| 2.3.8 Impact of the load and the aggregate processes running together | 112 |
| 2.3.9 Impact of the load and the query processes running together | 116 |
| 2.4 Results from an SAP perspective | 119 |
| 2.4.1 Results summary | 119 |
| 2.4.2 Resources | 123 |
| 2.5 Lessons learned | 123 |

| | | |
|--|--|------------|
| 2.5.1 | The load process | 123 |
| 2.5.2 | The aggregate rollup process | 125 |
| 2.5.3 | The query process | 125 |
| 2.6 | DB2 9 features used by SAP | 126 |
| Chapter 3. The DB2 perspective | | 127 |
| 3.1 | Introducing DB2 | 128 |
| 3.1.1 | Instance | 128 |
| 3.1.2 | Database | 128 |
| 3.1.3 | Partitioned databases | 131 |
| 3.1.4 | DB2 9 | 135 |
| 3.1.5 | SAP and DB2 9 | 142 |
| 3.2 | The project environment | 149 |
| 3.2.1 | Database creation and data redistribution process | 149 |
| 3.2.2 | DB2 database data grow | 158 |
| 3.2.3 | DB2 migration | 159 |
| 3.2.4 | Database servers | 164 |
| 3.2.5 | Database partitions | 165 |
| 3.2.6 | Tablespaces and disk layout | 166 |
| 3.2.7 | Object distribution across partitions | 176 |
| 3.2.8 | Buffer pools | 183 |
| 3.2.9 | Monitoring tools | 185 |
| 3.3 | Options and parameters discussions | 193 |
| 3.3.1 | Registry and environment variables | 194 |
| 3.3.2 | Instance configuration | 198 |
| 3.3.3 | Database configuration | 203 |
| 3.4 | Results from a DB2 specialist's perspective | 206 |
| 3.4.1 | Stress test results - KPI-G. | 207 |
| 3.4.2 | Tests with DB2 data row compression | 228 |
| 3.4.3 | REORG and RUNSTATS comparison with and without compression | 235 |
| 3.4.4 | Stress test comparison with and without compression | 247 |
| 3.4.5 | Stress test comparison with concurrent I/O and file system cache | 265 |
| 3.4.6 | Multi Dimension Clustering tables test results | 271 |
| 3.4.7 | STMM and DPF (SAP NetWeaver BI) | 280 |
| Chapter 4. The IBM System p perspective | | 287 |
| 4.1 | Introducing IBM System p model p595 | 288 |
| 4.1.1 | The System p hardware | 288 |
| 4.1.2 | Simultaneous multi-threading | 288 |
| 4.1.3 | POWER Hypervisor | 289 |
| 4.1.4 | Virtualization features on IBM system p5 | 290 |
| 4.1.5 | Operating systems | 293 |
| 4.2 | The project environment | 294 |
| 4.2.1 | IBM System p595 servers and logical partitioning | 295 |
| 4.2.2 | Storage and file systems | 300 |
| 4.2.3 | Network | 305 |
| 4.3 | Settings and options | 310 |
| 4.3.1 | General options and recommendations | 310 |
| 4.3.2 | AIX kernel and network | 311 |
| 4.3.3 | About lsattr parameters | 312 |
| 4.4 | Lessons learned | 313 |
| 4.4.1 | Memory affinity and paging | 313 |
| 4.4.2 | Tcp_NodelayAck | 314 |

| | |
|--|------------|
| 4.4.3 Use of large pages | 315 |
| 4.4.4 Power5 virtualization | 316 |
| 4.4.5 JFS2 lock contention | 316 |
| 4.4.6 DB2 FCM | 316 |
| Chapter 5. The system storage perspective | 317 |
| 5.1 Introducing the system storage | 318 |
| 5.1.1 Disk - IBM System Storage DS8300 | 318 |
| 5.1.2 Library - IBM System Storage TS3500 Tape Library | 320 |
| 5.1.3 Tape - IBM System Storage TS1030 Tape Drive | 322 |
| 5.1.4 Storage Area Network | 322 |
| 5.2 The project environment | 324 |
| 5.2.1 Storage area network architecture | 325 |
| 5.2.2 Zoning | 326 |
| 5.2.3 Disk configuration | 330 |
| 5.2.4 Tape configuration | 337 |
| 5.3 Options and parameters discussion | 338 |
| 5.3.1 Disk options and parameters | 338 |
| 5.3.2 Tape options and parameters | 339 |
| 5.3.3 DB2 logs and tape devices | 339 |
| 5.4 Results | 341 |
| 5.4.1 Tape KPI results | 341 |
| Chapter 6. The Tivoli Storage Manager perspective | 347 |
| 6.1 The project environment | 348 |
| 6.1.1 The challenge | 348 |
| 6.1.2 Introducing the backup/restore solution | 348 |
| 6.1.3 Components and settings of the backup solution | 361 |
| 6.1.4 Backup and restore processes | 368 |
| 6.2 Options and parameters discussions | 371 |
| 6.2.1 Backup: parallel versus sequential | 371 |
| 6.2.2 Design consideration | 373 |
| 6.2.3 Multiple volume sets | 373 |
| 6.2.4 Multiple nodenames configuration | 374 |
| 6.3 Infrastructure tests | 375 |
| 6.3.1 KPI-3A: back up 60 TB in less than 12 hours | 375 |
| 6.3.2 KPI-1: FlashCopy restore and rollforward of 500 GB log files | 381 |
| 6.3.3 KPI-2: DB restore from tape with rollforward of 2 TB archived logs | 385 |
| Appendix A. The scripts used | 393 |
| For the restore | 394 |
| For the node 7 backup | 395 |
| The Tivoli Data Protection for mySAP configuration file | 396 |
| initEB8.fct | 399 |
| The Tivoli Data Protection ACS configuration file | 399 |
| The main backup script | 400 |
| For the node 7 restore | 404 |
| The script for the backup of node 0 | 404 |
| The Tivoli Storage Manager API configuration file | 408 |
| Abbreviations and acronyms | 409 |
| Related publications | 417 |
| IBM Redbooks | 417 |

| | |
|-------------------------------|------------|
| Online resources | 417 |
| How to get IBM Redbooks | 418 |
| Help from IBM | 418 |
| Index | 419 |

Archived

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.


This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo) ®
i5/OS®
pureXML™
xSeries®
z/OS®
AIX 5L™
AIX®
DB2 Universal Database™
DB2®
DS4000™

DS6000™
DS8000™
Enterprise Storage Server®
FlashCopy®
HACMP™
IBM®
Micro-Partitioning™
POWER™
POWER Hypervisor™
POWER5™

POWER5+™
Redbooks®
System p™
System p5™
System x™
System z™
System Storage™
Tivoli®
TotalStorage®
WebSphere®

The following terms are trademarks of other companies:

BAPI, ABAP, SAP NetWeaver, mySAP, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

Snapshot, and the Network Appliance logo are trademarks or registered trademarks of Network Appliance, Inc. in the U.S. and other countries.

InfiniBand, and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.

Java, JDBC, J2EE, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Excel, Microsoft, MS-DOS, Visual Basic, Windows NT, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Pentium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

In order to improve the performance and operational efficiency of businesses worldwide, a customer using SAP® wanted to establish a global business program to define and implement a standardized, group-wide business process architecture and associated master data for the parameterization of the group of software tools.

The expected growth of the number of users and the size of the database would be at a level never reached by other customers, however, so IBM® was asked to undertake the following tasks:

- ▶ Test the application to be sure that it can sustain such growth.
- ▶ Prove the manageability of the solution.
- ▶ Provide recommendations to optimize the infrastructure architecture.

This project illustrates the new *near real time* business intelligence (BI) context approached by customers who want the ability to rapidly analyze their business data to gain market shares. Data today comes from many diverse global sources and needs to be merged into an intelligent data warehouse.

This IBM Redbooks® publication describes the testing that was done in terms of performance and manageability in an SAP NetWeaver® BI and DB2® environment on IBM System p™ when scaling a client's solution to a data warehouse of 60 terabytes (TB). This book resulted from a joint cooperative effort that included the PSSC, the IBM/SAP International Competency Center, the DB2-SAP Center of Excellence, SAP AG, and a customer. The customer involved in this project is a worldwide company employing more than 250,000 employees with factories and logistics operations in almost every country in the world.

This project involved multiple technical skills and multiple products, as described here:

- ▶ Chapter 1, “Project overview: business objectives, architecture, infrastructure, and results” on page 1, summarizes the entire project, starting from the business needs through the description of the environment and options used, to the results achieved. This chapter can be viewed as an executive summary from an IT specialist perspective.
- ▶ Chapter 2, “The SAP NetWeaver BI perspective” on page 47; Chapter 3, “The DB2 perspective” on page 127; and Chapter 4, “The IBM System p perspective” on page 287, provide detailed views of the project from the perspectives of SAP specialists, DB2 specialists, and System p and AIX® specialists, respectively.
- ▶ Chapter 5, “The system storage perspective” on page 317, and Chapter 6, “The Tivoli Storage Manager perspective” on page 347, describe the storage environment and the manageability issues in such a large environment.
- ▶ Finally Appendix A, “The scripts used” on page 393, provides the scripts that we needed to develop for this project.

Notes: The tests and results provided in this book correspond to a specific application in a specific environment with specific criteria. Using the performance numbers to design another environment for another application with other criteria would be inappropriate.

This book describes one project involving a series of tests using multiple products. It does not detail all the parameters of every product. Only the parameters capable of influencing these specific tests are discussed.

For detailed explanations about each product, refer to the appropriate product documentation.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Christian Matthys is a Consulting IBM IT Specialist who has spent more than 25 years with IBM as a System Engineer working with large, mainframe-oriented customers in France. He spent three years as an ITSO Project Leader on assignment in Poughkeepsie, NY. In 2000, he joined the EMEA Design Center for On Demand Business in the PSSC, helping customer projects exploit leading edge technologies. Christian works as a Project Leader for the Poughkeepsie ITSO Center, leading and supporting residencies from his base in the PSSC.

Abbas Birjandi is an IBM certified professional with more than 35 years of experience in the computing industry. He holds a Ph.D. in Computer Science. He joined IBM France and subsequently PSSC in IBM Montpellier in 1994. He has held many positions, including computer research and faculty positions at various universities, systems architect, project leader, and management positions. For the last few years he has been working in projects related to IBM educational activities and his last position before joining the ITSO was as EMEA Project Manager for the Early Support Program.

Alexis Gausach is an IT Specialist working at the Products and Solutions Support Center in Montpellier, France, as a System Storage™ Benchmark Manager. He has worked at IBM France for 18 years, and his areas of expertise include DS8K, DS6K, DS4000™, and Tivoli® Storage Manager.

Fabio Hasegawa is a Consulting IT Specialist leading the DBA Distributed Center of Competency, IBM Global Business Services in Brazil. He has been mastering solutions that involve infrastructure architecture over distinct IBM products, such as IBM DB2, WebSphere® Application Server, WebSphere Message Broker, and WebSphere Information Integration. He has worked for over 10 years in IT. During this time, he worked on projects helping worldwide companies in distinct segments (telco, financial, government, and banking). Fabio's areas of specialty are business intelligence solutions, infrastructure sizing, performance management, and designing and improving high availability solutions focused on information management services.

Edgar Maniago has been with SAP since May 2005, when he received his Software Engineering degree from McMaster University in Canada. He is currently a member of the joint IBM/SAP Integration and Support Centre, where he tests, develops, and integrates new features of DB2 with SAP NetWeaver. Edgar assists SAP consultants and customers with activities such as troubleshooting and performance optimization through his SAP development support role as well. Prior to joining SAP, he worked for IBM on the DB2 Development Infrastructure team in the IBM Toronto Lab.

Pekka Siekkinen is a certified IT Specialist in IBM Finland. He has been with IBM for 19 years. He has been working with AIX since 1991. His areas of expertise include AIX, System p, storage systems, and high availability solutions.

James Thompson is a Performance Analyst for IBM Systems & Technology Group in Tucson, Arizona. He has worked at IBM for eight years, the first two years as a Level 2 Support Engineer for Tivoli Storage Manager, and for the past six years he has provided performance support for the development of IBM Tape and NAS products. He holds a bachelor's degree in Computer Science from Utah State University.

A very special thanks to the following people for their contributions to this book:

| | |
|-----------------------------|--|
| Thomas Becker | SAP AG, Germany |
| Francois Briant | PSSC, IBM France |
| Carol Davis | IBM SAP International Competency Centre, IBM Germany |
| Jean-Philippe Durney | PSSC, IBM France |
| Edmund Haefele | Technical sales support, IBM Germany |
| Philippe Jachimczyk | PSSC, IBM France |
| Michael Junges | SAP AG, Germany |
| Thomas Marien | PSSC, IBM France |
| Thomas Rech | IBM DB2 Center of Excellence, IBM Germany |
| Herve Sabrie | PSSC Complex Project Manager, IBM France |

We express our appreciation to the many people who contributed their time and skills to this project:

| | |
|-----------------------------|---|
| Mathew Accapadi | IBM AIX Laboratory, IBM US |
| Thomas Aiche | PSSC, IBM France |
| Franck Almarcha | PSSC, IBM France |
| Jean-Armand Broyelle | PSSC, IBM France |
| Guiyun G Cao | IBM DB2 laboratory, IBM US |
| Christian Casanova | PSSC, IBM France |
| Andreas Christian | IBM DB2 Center of Excellence, IBM Germany |
| Thierry Huche | PSSC, IBM France |
| Said Kemiche | PSSC, IBM France |
| Lee La Frese | IBM Storage Laboratory, IBM US |
| Steffen Mueller | SAP AG, Germany |
| Antoine Naudet | PSSC, IBM France |
| Stephan Navarro | PSSC, IBM France |
| Robert Nicolas | PSSC, IBM France |

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:
ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review book form found at:

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400



Project overview: business objectives, architecture, infrastructure, and results

This chapter is a technical summary of the project, explaining the business objectives, the infrastructure, and the tools used. It provides a summary of the results for all the requested tests in addition to the specific DB2 compression feature.

1.1 The scope of the project

This section provides a summary of the objectives and the infrastructure set up to achieve these objectives.

1.1.1 The business context and the customer strategy

In order to gain a competitive advantage, all industries look for new ways to maximize their information insight. *Near real time* business intelligence (BI) provides a new approach to companies that want the ability to rapidly analyze and act upon the hidden benefits of their business information. The phrase *near real time* pertains to the delay introduced, by automated data processing or network transmission, between the occurrence of an event and the use of the processed data. It usually implies that there are no significant delays (the distinction between near real time and real time can be somewhat nebulous and must be defined for the situation at hand). Making fast and accurate business decisions based on the business data has always been critical to success.

Data today comes from many diverse global sources. It needs to be merged into an intelligent data warehouse. The projected dimension of these business intelligence data warehouse systems, for the near future, far exceeds the boundaries of the current experience. The size of the database is not the most important factor in such an infrastructure architecture. The end users must get rapid and constant responses to their requests (reports and queries). They must be assured that the data they use is the latest available business data, to support the correct business decisions. They need a global system that is stable and always available, to support their daily business challenges and decisions.

The customer involved in this project is a worldwide company employing more than 250,000 employees with factories and logistics operations in almost every country in the world. Recently, this customer launched a very large SAP implementation project with the aim to unlock business potential by creating and adapting common global business processes while still providing to each market the ability to make their specific analysis and local decisions. *Think global, act local* is the new mantra for companies of all shapes and sizes that have global aspirations.

The customer business strategy is based on a repository of consolidated data that combines production data extracted from the day-to-day operations and sales and financial data representing the market trends. This repository is considered a strategic tool for informed business decisions and further business development. This repository can quickly become very large. From 7 TB of active data today, it is expected to achieve over 60 TB of active data in the next few years, with more than 200 TB of disk. The reliability, scalability, and manageability of the solution is then vital.

1.1.2 Test objectives

This customer asked its business partners to validate its approach. IBM proposed running an SAP NetWeaver BI stress test (called a proof of concept or POC) with the following three objectives:

- ▶ Prove that the complete SAP NetWeaver BI solution (infrastructure, database, and SAP application) is stable and manageable with significant levels of data volumes (up to five times the current volumes), application, administration, and infrastructure activity levels (from 0.8 transactions per second to 2.08 transactions per second). The number of users (up to five times the current user activity) is not really significant in this type of test.

- ▶ Cover all aspects of the solution including:
 - The infrastructure
 - The SAP NetWeaver BI administrative activities
 - The impact of the layered and scalable architecture on operational activities
 - The simulation of user activity, including online and batch activities
- ▶ Demonstrate that the infrastructure can be easily managed, thereby improving usability and satisfaction for users.

IBM European Products and Solutions Support Center (PSSC), located in Montpellier, France, was asked to set up this SAP NetWeaver BI stress test to address the customer expectations in terms of infrastructure scalability and manageability. This was a joint cooperative effort that included the IBM/SAP International Competency Center, the DB2-SAP Center of Excellence, SAP AG, multiple IBM labs, and the customer.

The test consisted of a two-stage plan to prove that SAP NetWeaver BI 3.5 can scale to manage up to 60 TB of data, and that the IBM infrastructure represented by IBM System p5™ servers, DB2, and IBM TotalStorage® products could manage the performance, throughput, and management requirements of such an SAP NetWeaver BI system.

1. The first stage of the project would test the system from 7 TB, the current customer environment, to 20 TB on the hardware landscape as defined by the customer, using one System p5-595 and a single DS8300. The findings of the first stage would provide the information necessary to design the hardware infrastructure necessary to scale to 60 TB. This first stage is described in *Infrastructure Solutions: Design, Manage, and Optimize a 20 TB SAP NetWeaver Business Intelligence Data Warehouse*, SG24-7289.
2. The second stage of the project would test the proposed new architecture to 60 TB, a level of data that the customer expects to reach in the next few years. This book describes this second stage.

Fourteen months were needed from the beginning of the first stage until the end of the second stage.

1.1.3 The required tests

The tests were divided into two categories: performance and administration. Two types of tests were required by the customer:

- ▶ *Online* tests, focusing on load, including high volume updates of business data, aggregate building, and online reporting with fixed response time requirements. The load is made of:
 - Data load, called the *upload* - The InfoCube load activity is generated by process chains provided by the customer.
 - Online query load, called the *query load* - The online load is generated via an external tool and the scripts provided by the customer simulate HTML online user queries.
 - The *aggregation* load - Aggregation is the BI process of reorganizing, filtering, and reducing data in the fact tables to create smaller aggregate cubes for quicker and more efficient access by specific queries.

The three functions have to be tested separately and then in a combined workload. The *combined* load is the test consisting of all three of the individual loads at the same time: query, upload (sometimes called data load), and aggregation.

- ▶ *Infrastructure* tests, focusing on the manageability, including backup/restore and disaster recovery scenarios, to make sure that the infrastructure is able to deliver the expected operational windows. They include the simulation of a daily backup to disk, a daily backup to tape, the recovery of a full day of work, and a disaster recovery scenario.

The initial data were provided by the customer. Sensitive data have been removed and data growth up to 20 TB and then to 60 TB was applied using SAP load and aggregate processes.

Table 1-1 defines the different requested key performance indicators (KPI).

Table 1-1 Key performance indicator tests

| Type of test | KPI name | KPI environment | | KPI objective |
|---------------------|--|---|-------------------------------------|---|
| Online KPIs | KPI-A | Simulation of 100 users and reproduction of the current data load with 7 TB and 20 TB using the current customer architecture | | |
| | KPI-D | Simulation of 300 users and three times the current load, with 20 TB and 60 TB | KPI-D0 | 20 TB, single DS8000™ and DB2 V8 |
| | | | KPI-D1 | 20 TB, multi DS8000 and DB2 V8 |
| | | | KPI-D2 | 20 TB, multi DS8000 and DB2 9 |
| | | | KPI-D3 | 60 TB, multi DS8000 and DB2 V8 |
| | | | KPI-D4 | 60 TB, multi DS8000 and DB2 9 |
| | | | KPI-D5 | 60 TB, multi DS8000, DB2 9 and using virtualization features |
| KPI-G | Simulation of 500 users and five times the current load, with 60 TB. | | 60 TB DB2 9, multi DS8000 and DB2 9 | |
| Infrastructure KPIs | KPI-1 | 60 TB data warehouse | | Flashback and simultaneous roll forward of 500 GB of data in less than 8 hours |
| | KPI-2 | 60 TB data warehouse | | Database restore from tape using Tivoli Storage Manager server and roll forward of 2 TB of logs in less than 24 hours |
| | KPI-3a | 60 TB data warehouse | | Tape backup of a flashcopy in less than 12 hours with no online workload |

The KPI-A used the current customer architecture. The tests are not part of this book. They are documented in *Infrastructure Solutions: Design, Manage, and Optimize a 20 TB SAP NetWeaver Business Intelligence Data Warehouse*, SG24-7289.

The KPI-D is made of multiple tests:

- ▶ KPI-D0 proposed a new architecture and is described in *Infrastructure Solutions: Design, Manage, and Optimize a 20 TB SAP NetWeaver Business Intelligence Data Warehouse*, SG24-7289.
- ▶ The KPI-D1 to KPI-D5 were intermediate tests to validate the effect of the changes in the environment.

The KPI-G is the customer's ultimate objective.

It is out of the scope of this book to cover the IBM High Availability Cluster Multiprocessing (HACMP™) solution and Logical Volume Manager (LVM) mirroring (even though both of these technologies are used by the customer), as well as the migration path to go from the existing architecture to the new proposed architecture.

1.1.4 The infrastructure

This section describes the architecture and the hardware and software components used for the tests.

The architectural foundations

The architecture was based on the results of the first stage and additional work among the customer, IBM, and the SAP teams.

- ▶ In stage 1, 33 DB nodes were assigned to five different System p logical partitions (LPAR). One System p595 and one DS8000 was used. It was decided to use the same architecture: 33 DB nodes, and all the different components (applications servers, storage agents, backup servers, and so on) are spread over five System p LPARS, using five System p servers and four disk sub-systems.
- ▶ The virtualization features of the System p servers support the very diverse mix of workload: online queries, data loading, aggregate roll-up, and data back-up. Using the shared processor pool functionality, the system is able to dynamically match the resource and the requirements despite the volatile production load mix, and the difficulty of predicting the growth pace.
- ▶ Manageability is critical for this architecture, again due to the very broad mix of workload having to be supported by the customer, and due to the variations of data and workload volumes, which are in this case difficult to predict. It is important to have some flexibility in the sizing of the different components (DB nodes, file systems, logical unit numbers, and so on) to be able to efficiently manage such large volumes of data and types of workloads.

The key components

The following system and key software components are used:

- ▶ SAP: SAP NetWeaver Business Intelligence Version 3.5 - No functional change is brought to SAP for this phase of test, though the lessons learned during the first phase are fully leveraged, including the way the different cubes are used for the different SAP workloads (query, updates, aggregate) and to generate the data volumes required for this phase of the test.
- ▶ AIX: AIX 5L™ Version 5.3 - The virtualization functions of System p are key new features added to the 60 TB architecture. These functions allow application servers, DB servers, and storage agents in separate LPARs on the same physical machine to share machine resources. The shared processor pool allows immediate and dynamic resource redistribution according to need and priority scheme, resulting in a better balance of system resources when addressing a broad mix of workload types.
- ▶ DB2 - The initial proposal was to use DB2 V8 and to study the DB2 9 impact in parallel, in particular for the compression, the back-up restore, and the optimization for SAP. Thirty-three DB nodes were distributed across five different servers (with the same LPAR design as for the phase 1). Four file systems per DB2 partition (data and temporary) are initially used. A DB2 multi-partition shared-nothing architecture is used, which means that each DB2 partition has allocated CPUs, RAM, and disks. This configuration offers a nearly linear scalable architecture with the number of partitions, increases DB2 parallelism and so the DB2 performance, and offers some flexibility through the ability to add or remove DB2 partitions, depending on needs and test results.

- ▶ Storage - One disk subsystem DS8300 Turbo is allocated per System p hosting the data (except for the DB2 partition 0). The data and the index of each DB partition is spread over eight arrays (shared-nothing architecture). The online production gets priority 1 and there are no dedicated arrays for the FlashCopy®. Twenty percent of extra capacity has been defined as necessary to achieve the manageability objectives.
- ▶ Tivoli Storage Manager and Tivoli Data Protection - Storage agents are distributed over the five systems, enabling the allocation of additional system resources, when needed, for backup and restore activities.

The physical infrastructure

Figure 1-1 shows, for the query process, a summary of the logical architecture, which is simple: a front end injects the workload to application servers that process the data. An architecture with multiple DB2 partitions is used. The DB2 partitions are grouped into multiple DB servers (p5 LPARS). At 60TB, the sheer data volume, for both production and backup, requires multiple DS8300 storage servers.

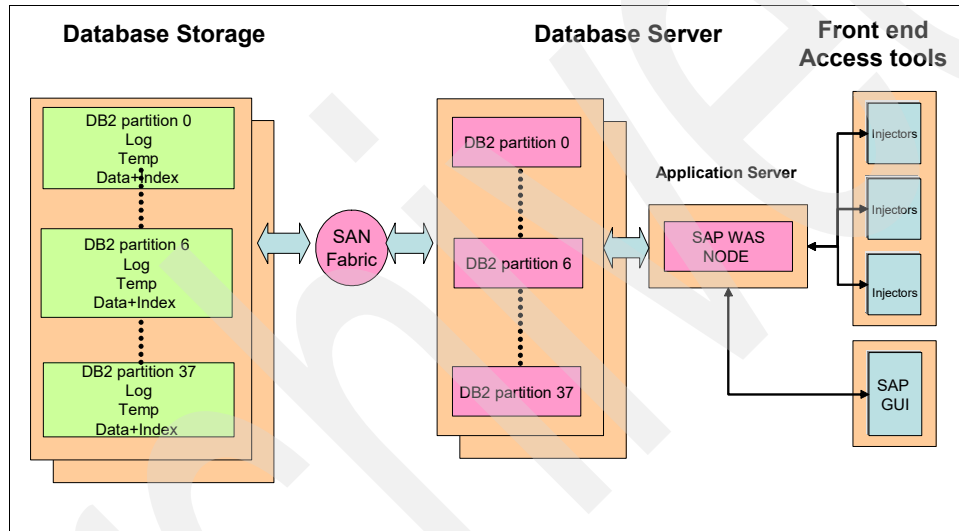


Figure 1-1 Logical architecture

Figure 1-2 shows the functional architecture.

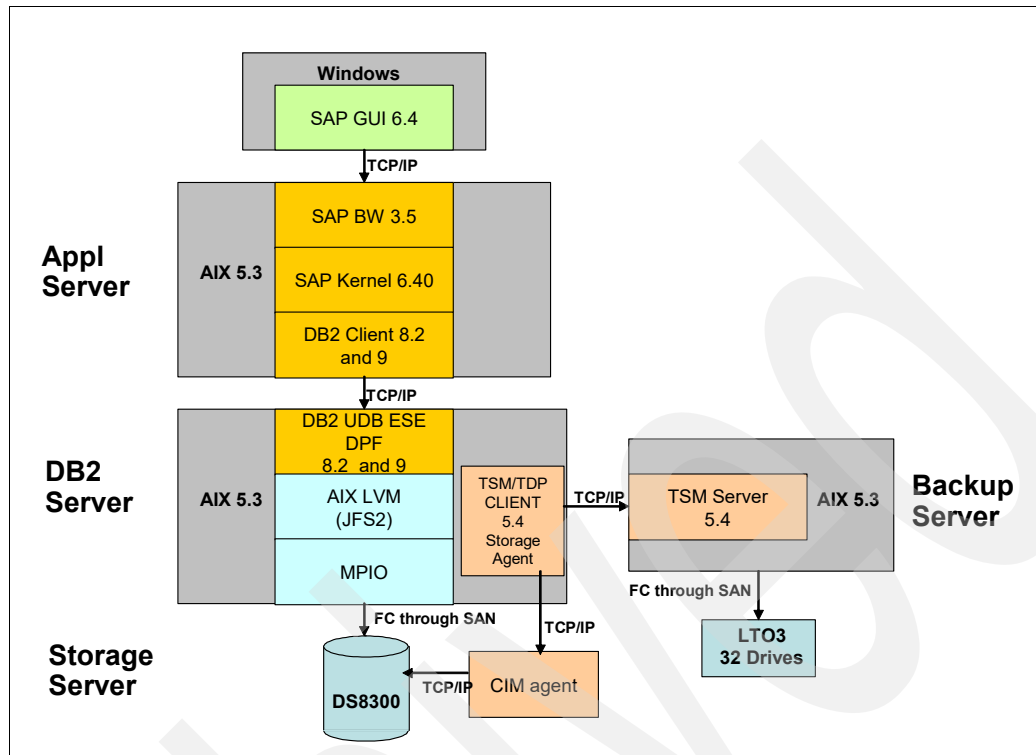


Figure 1-2 Functional architecture

The application servers, SAP components, receive the workload from external simulators hosted by BladeCenters with Windows®.

DB2 is the main component of the database servers. DB2 V8 and DB2 9 were used. The database servers are connected to the storage servers and a backup server whose job is to host the Tivoli components that drive the infrastructure KPIs.

The infrastructure was set up with the key components shown in Figure 1-3.

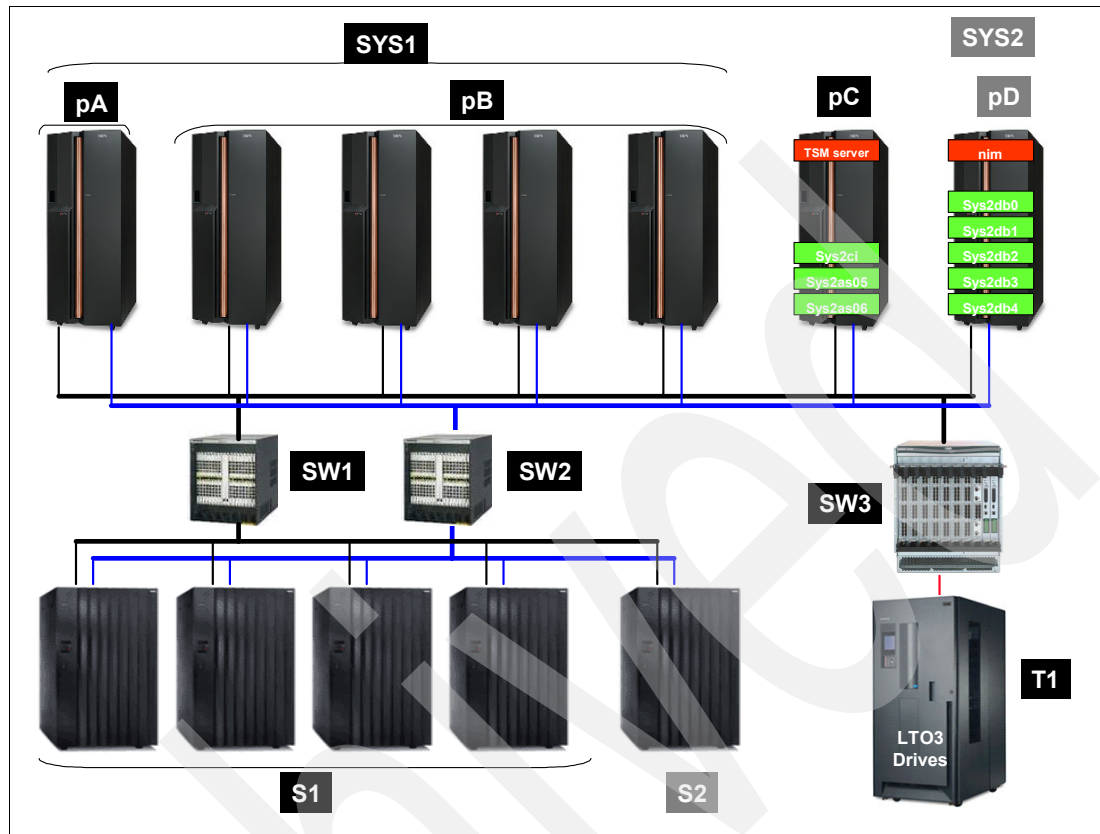


Figure 1-3 The physical infrastructure

The key components are:

- ▶ Six System p5 595, POWER5+™ 2.3 GHZ Turbo, 64 CPUs, and 512 GB RAM each.
 - Five for the DB2 database and application servers used in the system (called SYS1). Each System p5 hosts multiple partitions: to host the SAP application servers, to host the storage agent, and to host the 33 DB2 partitions.

Multiple application servers are used:

- SYS1CI is the SAP central instance used to manage the systems.
- SYS1BTC is an application server for batch processing, mainly for extractors and aggregate rollup processes (which are mainly batch processes).
- SYS1ONLn are application servers for the queries execution.
- SYS1AS0n are application servers to process the InfoCubes update process.
- SYS1STAn hosts the storage agent used by Tivoli Data Protection for Advanced Copy Services.

Each application server is usually hosted by one LPAR with one difference for one of the System p5's:

- One System p5 is configured with five LPARs:
 - One LPAR for the DB2 partition 0 and the SAP Central Instance (CI)
 - One LPAR for the queries execution (the online activities)
 - One LPAR for the aggregation activities (batch related)
 - One LPAR for the data load extractors (batch related)

- One LPAR for the storage agent

Initially, the intent was to give the DB2 partition 0 and the SAP CI the resource priority because these components manage sub-components for which the resources may be critical.

- Four other System p5's are configured, each with four LPARs:
 - One LPAR for DB2 with eight DB2 partitions; one system hosting the DB2 partitions 6 to 13, the second one hosting the DB2 partitions 14 to 21, the third one hosting the DB2 partitions 22 to 29, the fourth one hosting the DB2 partitions 30 to 37
 - One LPAR for the online activity (queries)
 - One LPAR for the batch activity (upload)
 - One LPAR for the storage agent
- One System p5 hosts the Tivoli Storage Manager server and its application servers.

Another System p5 was set up for the DB2 database, called SYS2, used for manageability, not for the tests.

- ▶ Four DS8300 Turbos with two expansion frames, with 640 disks each.
Another DS8000 was set up for the system SYS2 and was not used for the tests.
- ▶ One library 3584 and 68 LTO3 tape drives.
- ▶ Two McData SAN Directors ED140 - IBM SAN140M (2027-140) and one McData SAN Director ED10000 - IBM SAN256M (2027-256) and two McData 16 switches.

In total, 448 CPUs, 3–5 TB of memory, 240 TB disk capacity, and 370 Fibre Channel connections were configured with a 100 GB network.

1.1.5 The performance tools

Many reports were needed to make the calculation and to synchronize the results among the different systems. This section describes the tools used to report the performances.

Subsystems reports

For each subsystem specific tools were used.

- ▶ In the SAP environment, we needed:
 - To capture the aggregation logs to measure the start and stop times of the aggregation load using the *overview of background jobs* (SM37) SAP transaction, which allows to display the jobs in the archived data.
 - To capture the load profile to get the load throughput interval and the load ramp-up and down timeframes. We get this information using a tool provided by SAP, which extracts this data from monitoring tables. This tool allows tracking the loading volume in snapshots, extrapolated to the hourly throughput at the rate that was processed during that snapshot. With this information we build the little graphs that show the development of the loading rate over time.
 - To capture the aggregate statistics to get the aggregation end-to-end throughput using the *application statistics* (ST03) SAP transaction in the workload monitor, which allows us to analyze the application statistics (ASTAT) with every statistics record it creates.
 - To capture the query profile to get the query response times over a time interval using the ST03 SAP transaction.

- ▶ In the AIX environment, we used:
 - LPARMon¹ to get the real time system utilization. The graphical LPAR monitor for System p5 Servers (LPARMon) is a graphical logical partition (LPAR) monitoring tool that keeps an eye on the state of one or more LPARs on a System p5 server. The state information that can be monitored includes LPAR entitlement/tracking, simultaneous multithreading (SMT) state, and processor and memory use. LPARMon also includes gauges that display overall shared processor pool use and total memory use.
 - The NMON² tool to capture data. The NMON free tool is designed for AIX and Linux® performance specialists to use for monitoring and analyzing performance data, including: CPU utilization, memory use, kernel statistics and run queue information, disks I/O rates, transfers, and read/write ratios and many more statistics. The NMON tool can capture the data to a text file for later analysis and graphing for reports, and the output is in a spreadsheet format (.csv).

Data is captured at specific time intervals and written to a .csv file. CSV files are then analyzed with a spreadsheet and analyzed by a customized tool to provide, graphs as shown in Figure 1-4.

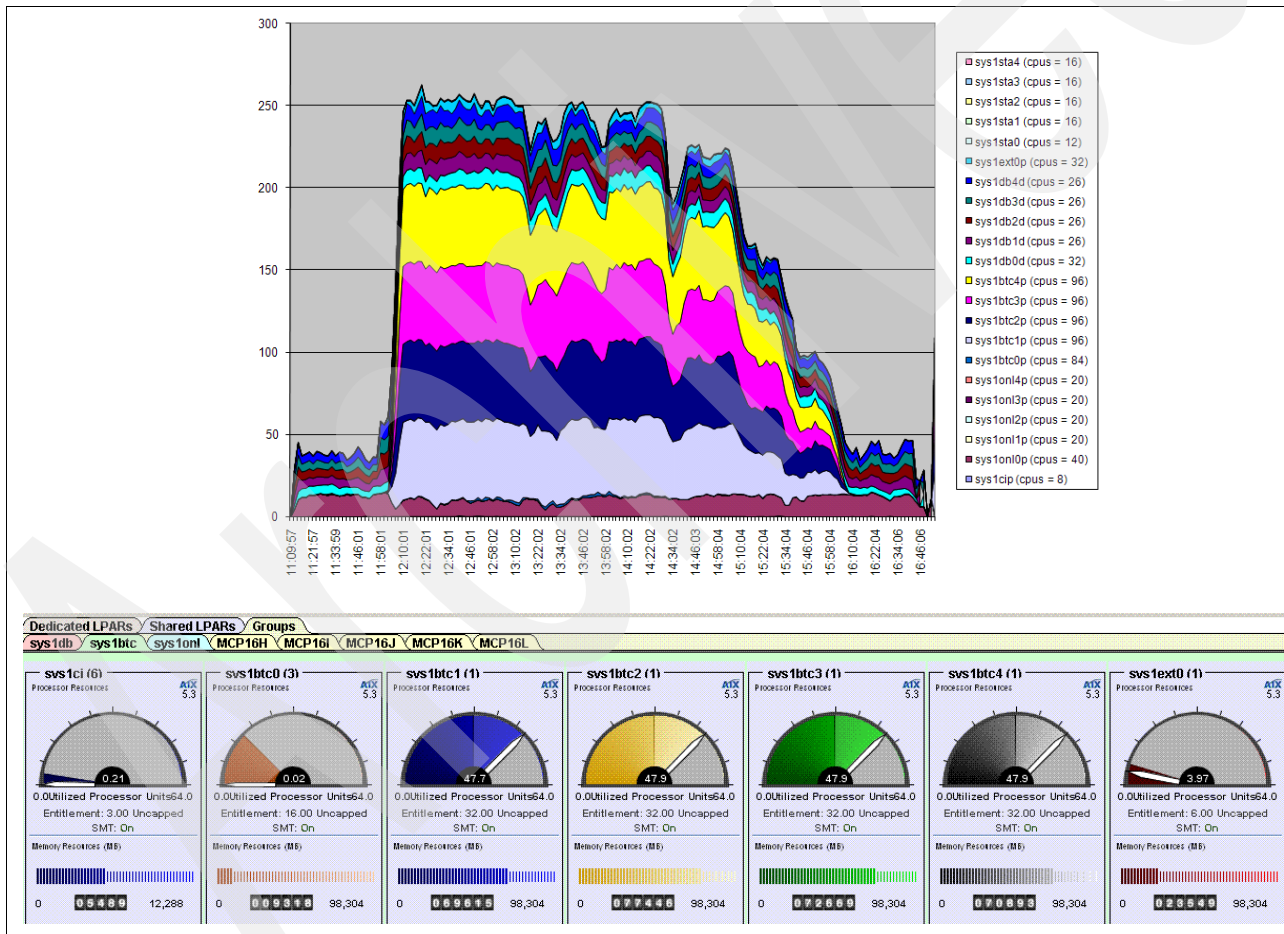


Figure 1-4 AIX reporting

¹ More information about the LPARmon tool is available at:
http://www.alpha-works.ibm.com/tech/lparmon?open&S_TACT=105AGX16&S_CMP=DWPA
² More information about the NMON tool is available at:
http://www.ibm.com/developerworks/aix/library/au-analyze_aix/index.html

- ▶ DB2 Performance Expert³ (DB2 PE) was used to capture DB2 data. DB2 PE is a workstation-based performance analysis and tuning tool that simplifies DB2 performance management. It provides the capability to monitor applications, system statistics, and system parameters in real-time and in historical mode. Data (cache utilization, SQL activity, locking, buffer pools, and so on) is captured at specific time intervals and stored in a DB2 database. DB2 PE provides online monitoring, short-term history monitoring, and long-term history monitoring. Figure 1-5 is an example of the capability of this tool.

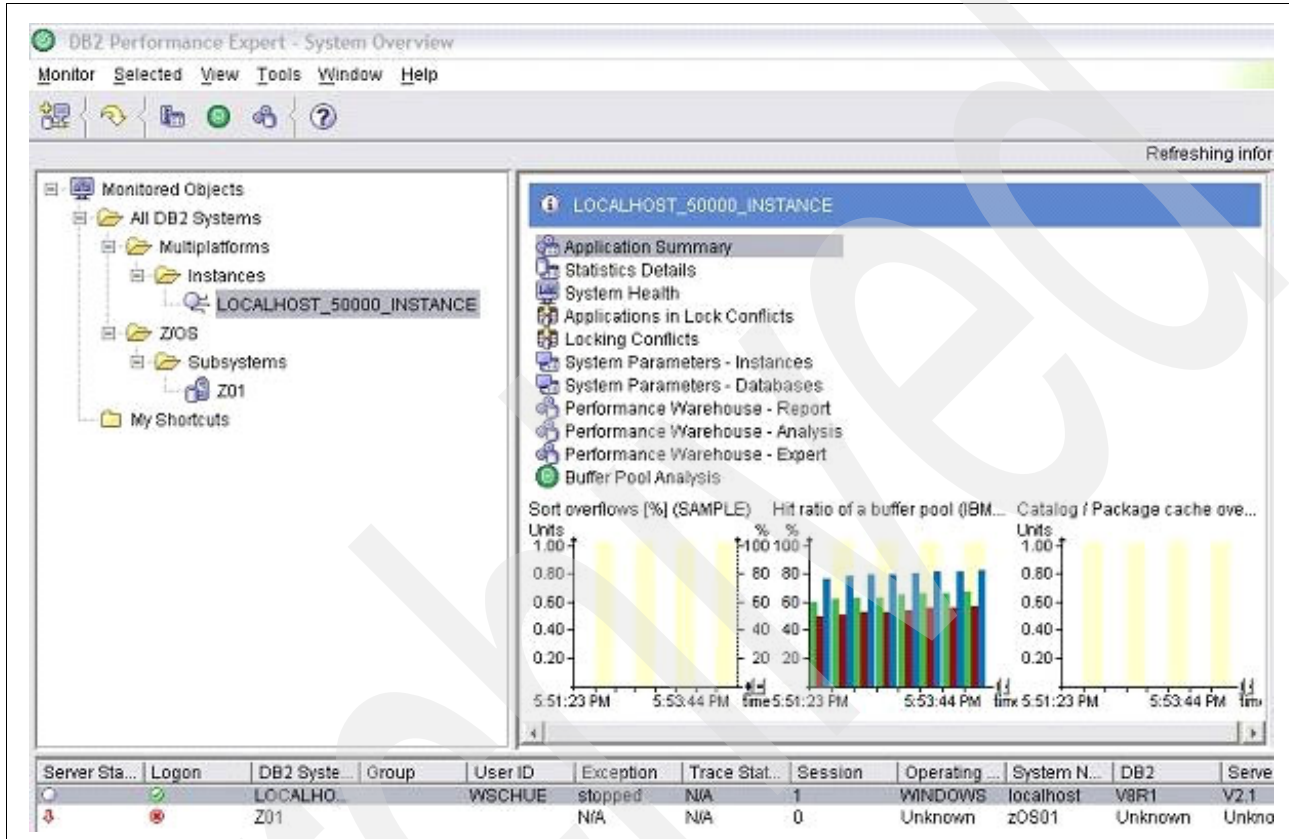


Figure 1-5 DB2 PE sample

³ More about DB2 PE is available at:
<http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0409schuetz/>

- ▶ Performance Data Collection Unit (PDCU) was used to capture storage data. PDCU is an internal IBM standalone Java™-based tool for performance data collection for the DS6000™ and the DS8000. PDCU collects performance metrics for offline analysis. Data (total I/O, read/write, cache, and son on) is captured at two-minute intervals and written to a .txt file. Text files are then analyzed with a spreadsheet. PDCU captures equivalent data to the IBM TotalStorage Productivity Center for disk product that is part of the IBM Total StorageProductivity Center Suite solution (IBM TPC)⁴. Figure 1-6 is an example of the capability of this tool.

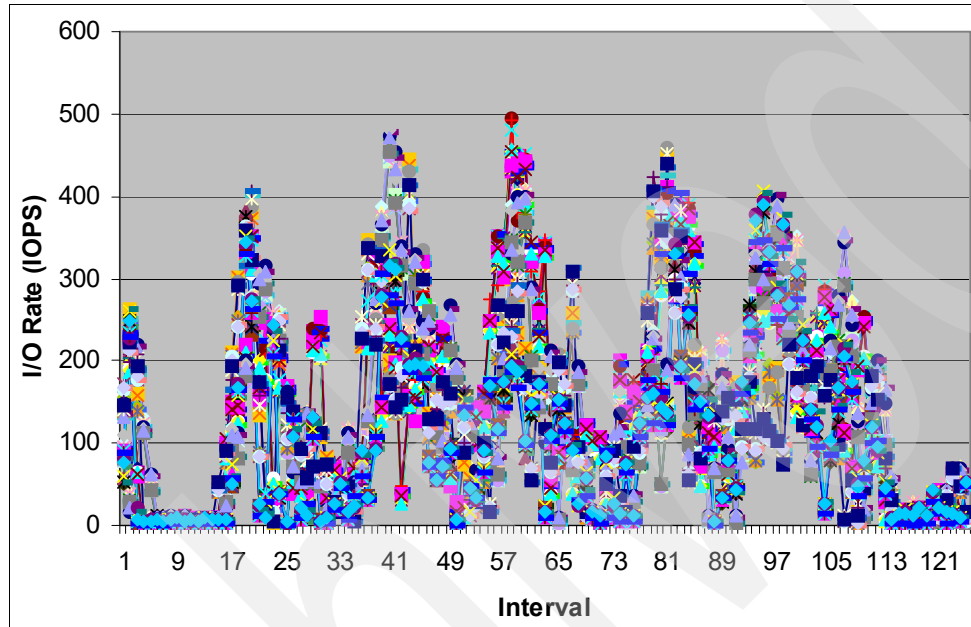


Figure 1-6 PDCU sample

Consolidation

All these subsystem tools were then combined to provide one single view.

⁴ More about the IBM TPC at ftp://ftp.software.ibm.com/common/ssi/rep_sp/n/TSD00757GBEN/TSD00757GBEN.PDF

We need to report the upload, the aggregate, and the query response time for a combined workload on the same graph and make the calculation while avoiding the warm-up and the wrap-up of the activities. The graph shown in Figure 1-7 is used to report the test activities.

- ▶ **A** shows the query response time every 1 to 5 minutes (depending on the tests) in real time.
- ▶ **B** shows the aggregation activity, which is averaged over the total run time.
- ▶ **C** shows the upload activity in real time.
- ▶ The window to report the KPI results, **D**, avoids the start of the different loads (**E**) and the end of the tests.
- ▶ **T** represents the time interval we keep for the calculation of the KPIs.

In this example, the test is said to be valid between 17h53 and 19h43, and the calculations, the average response time per transaction for the queries, and the number of records loaded or aggregated will be measured for 110 minutes.

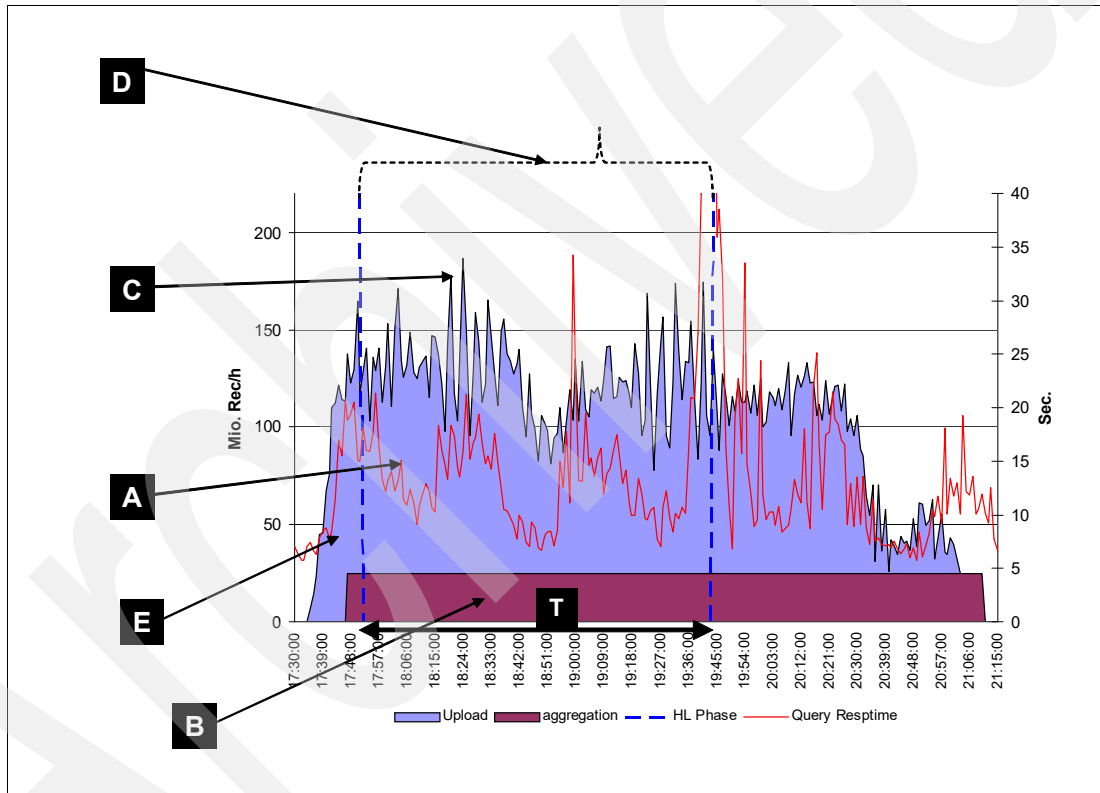


Figure 1-7 Consolidation of the reports

1.2 The online KPIs

This section provides the results of the requested KPIs. The details and discussions are then provided in all the other chapters of this book.

1.2.1 The progressive tests

In this project, the starting point was the current hardware and software infrastructure in production at a customer site. The hardware and software levels used by the customer

provided the baseline at the start of the project and determined the progress through the infrastructure migration. The full project, described in Figure 1-8, consisted of two phases. Phase 1 was built with an SAP NetWeaver BI data warehouse of 20 TB. Phase 2 was built with an SAP NetWeaver BI data warehouse of 60 TB. This book relates to phase 2 only.

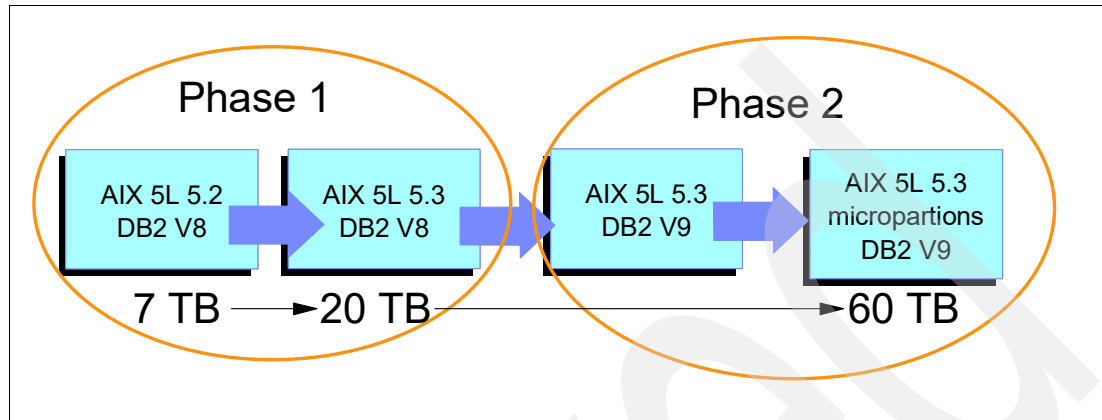


Figure 1-8 The phases of the full project

Going from the customer environment, a 7 TB data warehouse in a DB2 V8 and AIX 5L 5.2 environment, to the final proposed environment, a 60 TB data warehouse in a DB2 9 and AIX 5L 5.3, progressive steps were required, as shown in Figure 1-9 on page 15.

- ▶ Step 1 was the simulation of the current customer load: 100 concurrent users accessing a 7 TB data warehouse, with the current customer environment of AIX 5L 5.2 and DB2 V8. The initial implementation used 5+1 DB2 partitions with one System p 595:
 - Five data partitions to host the fact tables, the operational data stores, the aggregates, and the persistent staging area.
 - One DB2 partition to host the DB2 system catalog, the base tables, and the dimension tables.
 One System p595 (POWER5™, 1.9 GHz Turbo) was used.
- ▶ Step 2 was a simulation of three times the current customer load: 300 concurrent users accessing a 20 TB data warehouse. AIX 5L 5.3 with the hardware Simultaneous Multithreading (SMT) feature was used. The number of DB partitions was increased to 32+1. One hundred and twenty-eight CPUs in total were needed to achieve the objectives.
- ▶ Step 3 used the same tests as step 2, but using DB2 9 instead of DB2 V8.
- ▶ Step 4 used the same tests as step 3, but using the micro-partitioning feature of System p.
- ▶ Step 5 tests used all the features (DB2 9, AIX 5L 5.3 with SMT and micro-partitioning) for a load of 300 concurrent users accessing a 60 TB data warehouse.

- Step 6 was the final test, the objective requested by the customer: sustain a load of 500 concurrent users (five times the current workload) with a 60 TB data warehouse.

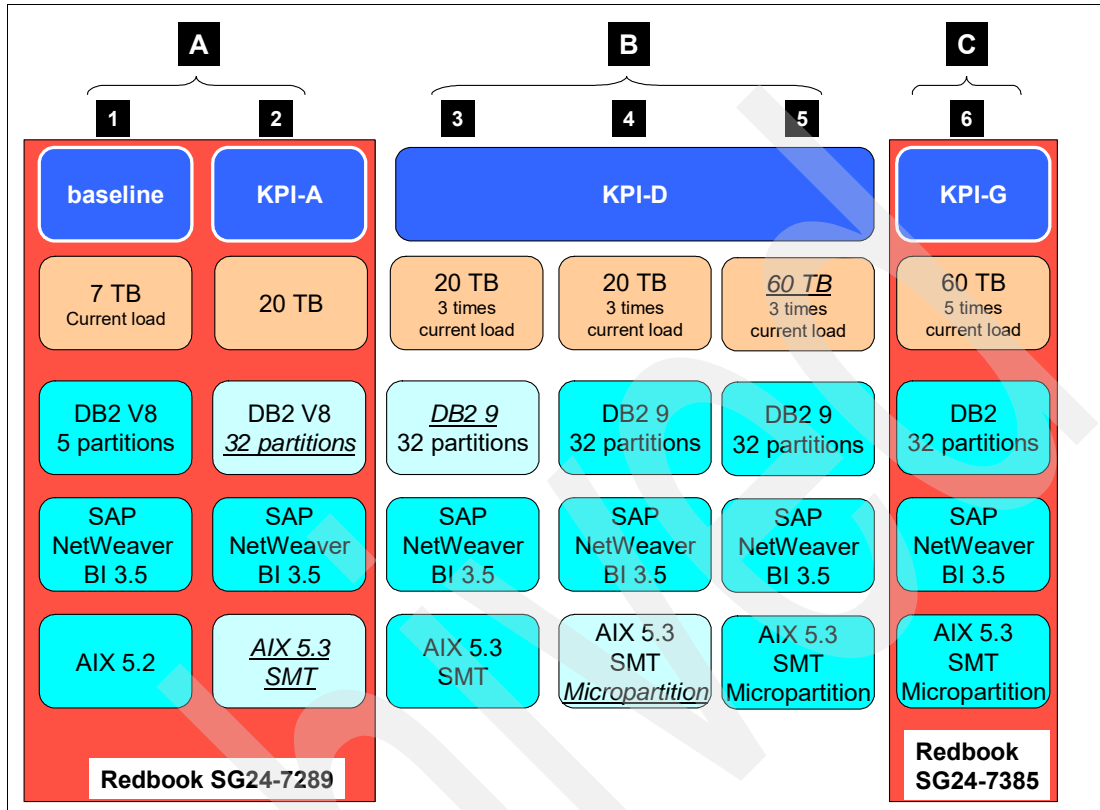


Figure 1-9 Progressive tests from baseline (7 TB data warehouse) to objective (60 TB data warehouse)

Steps 1 and 2 were the foundation for phase 1 and were documented in *Infrastructure Solutions: Design, Manage, and Optimize a 20 TB SAP NetWeaver Business Intelligence Data Warehouse*, SG24-7289. This book documents and discusses the options and some of the intermediate tests and the final test.

1.2.2 Objectives

The tests represent a combined load scenario (shown in Figure 1-10) in which data translation or cube loading (1), aggregate building (2), and online reporting (3), with fixed response time requirements, run simultaneously.

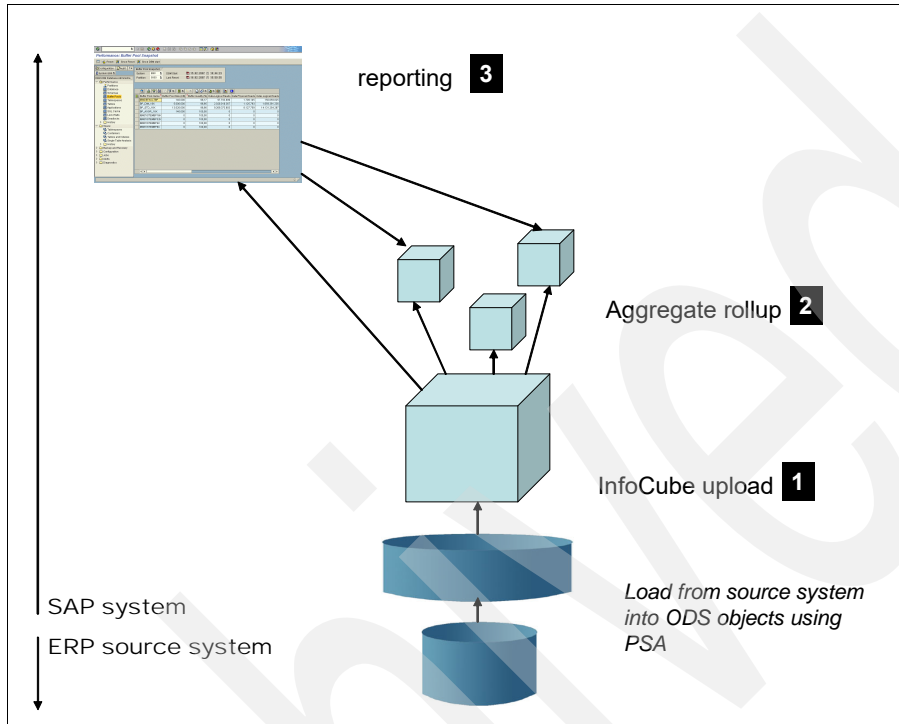


Figure 1-10 The combined load

The combined load test scenario consists of three different load types, each with a very different profile and a different KPI requirement. They represent the online report generation and the InfoCube maintenance necessary to bring new data online for reporting.

- ▶ One objective is to simulate user reporting. For this scenario we use online transactions. The *query load* consists of 10 different query types with variants that cause them to range over 50 different InfoCubes used for reporting. The queries are designed such that some of them use the OLAP cache in the application servers and 50% could not, some use aggregates (80%), and others (20%) go directly to the fact tables. This load is affected by the database growth: the number of rows in the fact tables increased from 20 million rows (in the KPI-A) to 200 million rows (in the KPI-G) for a number of transactions per second increasing from 0.8 to 2.08 with the same response time (20 seconds maximum). The query load is database focused and database sensitive. Competition for database resources is immediately translated into poor response times. The online users are simulated by queries initiated by an injector tool.
- ▶ InfoCube maintenance includes two activities. Both are initiated by SAP Job-Chains:
 - Data *load* (or upload of data)

The objective is to transform new data into the format defined for the reporting objects and to load the data into the target objects. For this scenario we use a pseudo batch, a batch driver spawning massive parallel dialog tasks. The batch extractor selects the data in data blocks from the source. It then initiates an asynchronous dialog task to take over the job of processing the block through the translation rules and updating the target InfoCubes.

This load allows for a wide variety of configuration options (level of parallelism, size of data packets, number of target cubes per extractor, and load balancing mechanism).

– Data *aggregation*

The objective is to aggregate the new data according to rules defined to improve the access efficiency of known queries. For this scenario we used a batch. The aggregation of the loaded data is primarily database intensive. There is not much configuration or tuning possibility for the aggregate load. The options available in this scenario were:

- The type of InfoCube: profitability analysis or sales statistics. This has an effect on the weight of the translation rules defined for the InfoCube type.
- The number of InfoCubes to use: This was based on the number required to get the KPI throughput. There was not that much in the way of fine tuning.
- The block size used: There was little guidance on this possibility available, so the initial setting was used.

Figure 1-11 shows the objective for each step of the tests.

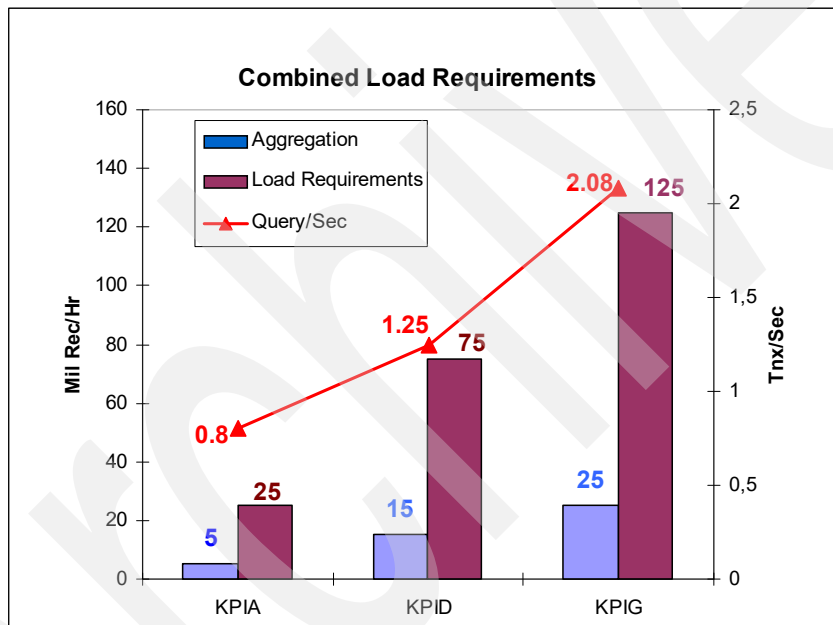


Figure 1-11 Online KPIs targets

1.2.3 The KPI-G results

Multiple tests were done with different configuration parameters to check for improvement and to demonstrate that the performance can be repeated and that the results can be achieved more than one time.

Table 1-2 summarizes the results for the most important cases. Four tests are described in this section, and we provide specific information for each of them when it is appropriate for the objectives. For some of them you can see that some of the objectives are not achieved.

Table 1-2 Four tests for KPI-G

| KPI-G | Load (millions records/hour) | Aggregation (millions records/hour) | Transactions/ sec. | Response time per transaction (sec.) |
|--------------|---|--|-------------------------------|---|
| Objectives | 125.00 | 25.00 | 2.08 | 20.00 |
| Test-1 | 115.53 | 22.69 | 1.96 | 11.82 |
| Test-2 | 125.05 | 25.03 | 2.13 | 16.09 |
| Test-3 | 132.14 | 24.84 | 2.08 | 18.00 |
| Test-4 | 132.92 | 24.01 | 2.05 | 16.32 |

Test-1

Table 1-3 describes the configuration parameters for test-1.

Table 1-3 KPI-G test-1 configuration parameters

| Parameters | Values |
|-------------------|---------------|
| Maxprocs | 4 x 32 |
| DIA_WP | 4 x 67 |
| vUsers | 29 |
| Extractors | 8 |
| Load cubes | 24 |
| Roll cubes | 8 |
| Aggregates | 10 |

Figure 1-12 shows the consolidation report for test-1. In this test none of the objectives are achieved, though they come close.

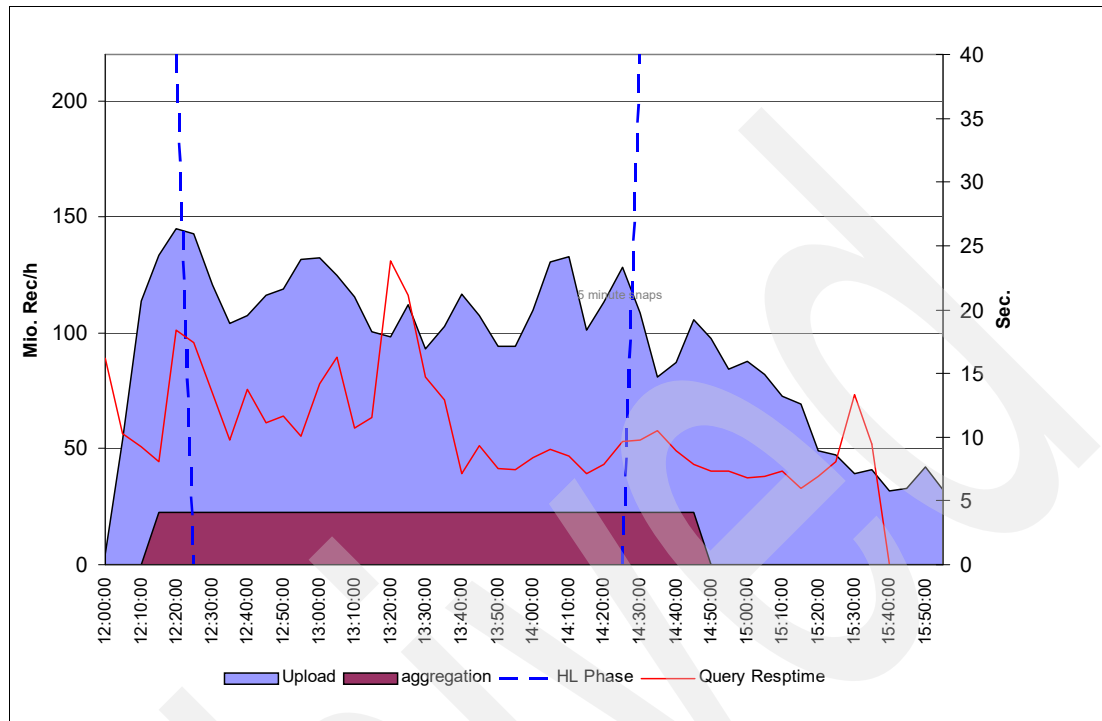


Figure 1-12 KPI-G test-1 report

Test-2

Table 1-4 describes the configuration parameters for test-2. Compared to test-1 parameters, Maxprocs, DIA_WP, and vUsers have been largely increased.

Table 1-4 KPI-G test-2 configuration parameters

| Parameters | Values |
|------------|---------|
| Maxprocs | 4 x 48 |
| DIA_WP | 4 x 100 |
| vUsers | 60 |
| Extractors | 8 |
| Load cubes | 24 |
| Roll cubes | 12 |
| Aggregates | 10 |

Figure 1-13 shows the consolidation report for the test-2. All the objectives are achieved.

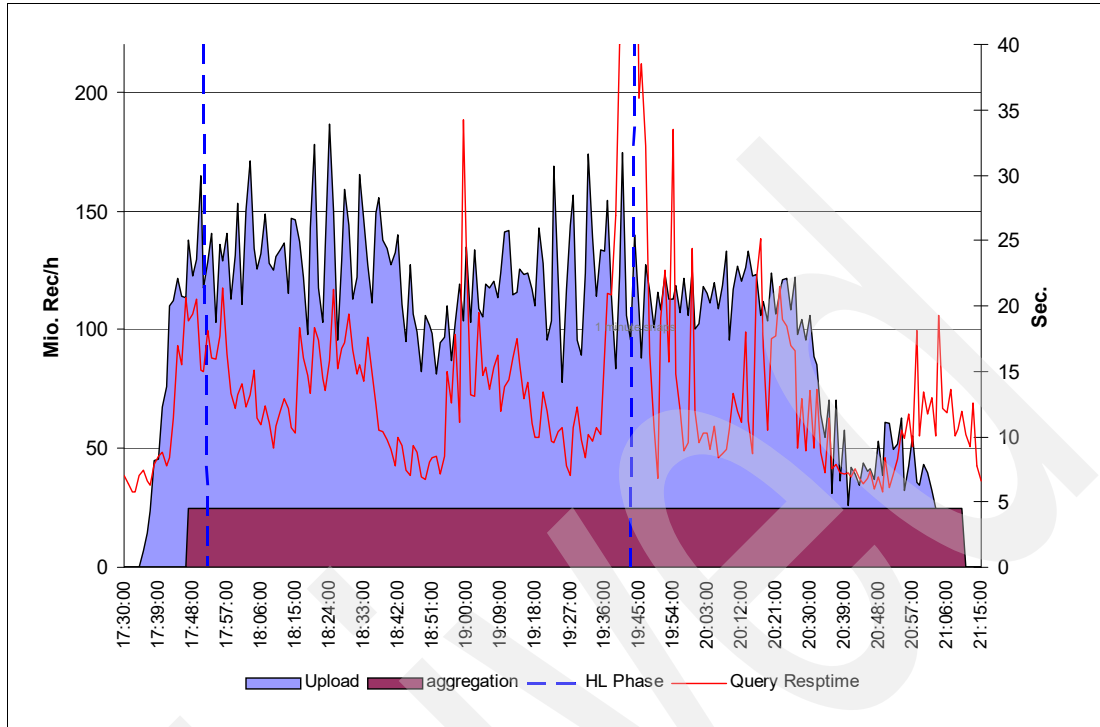


Figure 1-13 KPI-G test-2 report

Some remarks related to test-2:

- ▶ The total memory used by DB2 was about 1,250 GB:
 - 13.2 GB for all the DB2 instances' shared memory (0.4 GB per DB2 partition)
 - 1,103 GB for the main bufferpools (33.9 GB per DB2 data partition plus 18.2 GB for the DB2 partition 0)
 - 130.3 GB for the agent private memory (3.6 GB per DB2 data partition plus 15.1 GB for the DB2 partition 0)

- ▶ Figure 1-14 shows the System p5 CPU consumption. The legend to the right provides the number of CPs given for each LPAR. The graph summarizes the CPU consumption for all the LPARs.

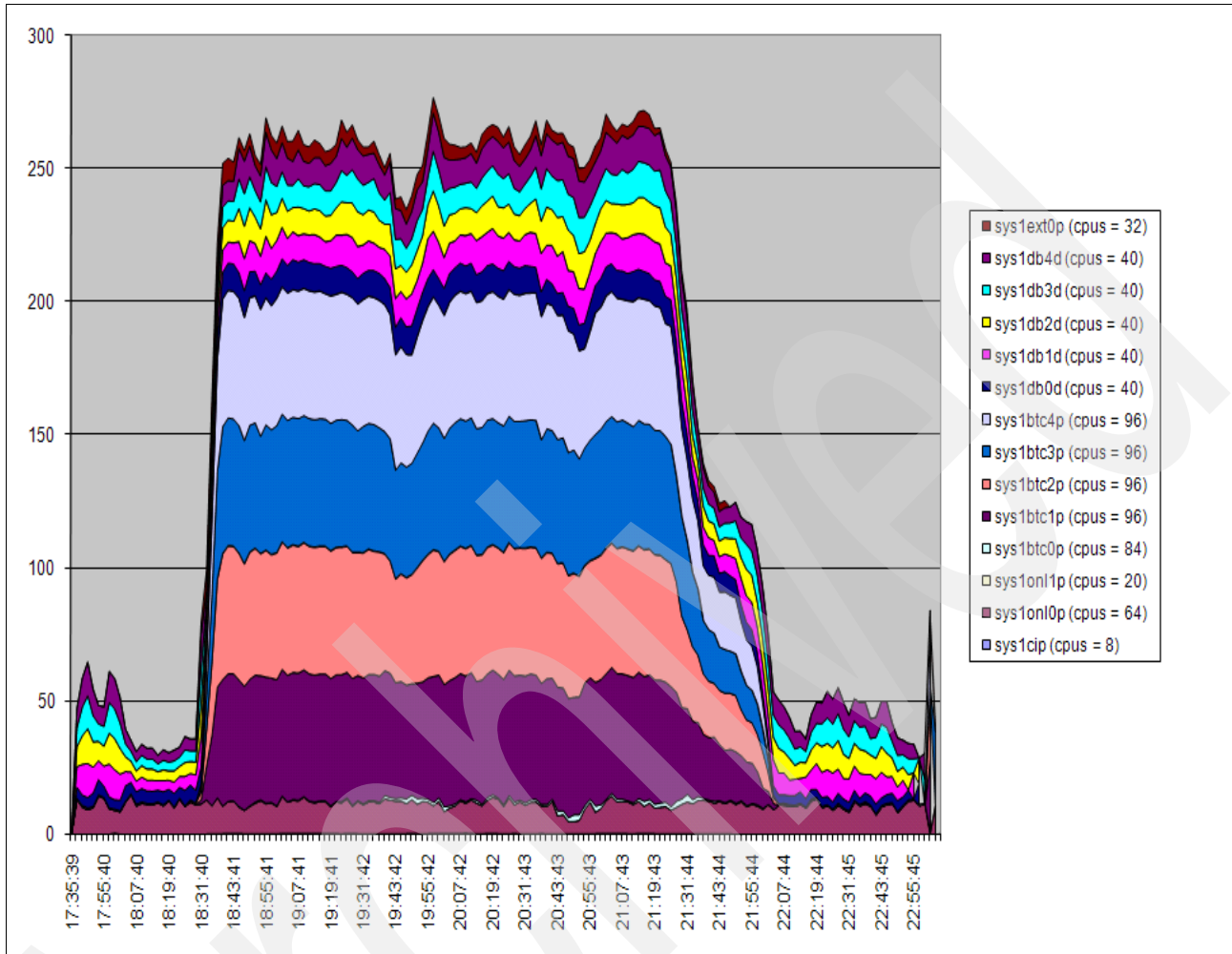


Figure 1-14 KPI-G test-2 total CPU consumption (all LPARs)

Table 1-5 provides more details about this CPU consumption. It provides the peak and the average CPU consumption per LPAR.

Table 1-5 KPI-G - test-2 - CPU max and average per LPAR

| LPAR | Average physical CPU | Maximum physical CPU | VP |
|-----------|----------------------|----------------------|-------|
| sys1cip | 0.21 | 0.52 | 4.00 |
| sys1onl0p | 10.92 | 14.73 | 32.00 |
| sys1btc0p | 0.60 | 2.90 | 42.00 |
| sys1btc1p | 27.88 | 48.00 | 48.00 |
| sys1btc2p | 27.85 | 48.01 | 48.00 |
| sys1btc3p | 27.42 | 47.99 | 48.00 |
| sys1btc4p | 27.76 | 47.95 | 48.00 |

| LPAR | Average physical CPU | Maximum physical CPU | VP |
|--------------|----------------------|----------------------|------------|
| sys1db0d | 7.81 | 11.37 | 20.00 |
| sys1db1d | 9.88 | 15.30 | 20.00 |
| sys1db2d | 9.59 | 15.13 | 20.00 |
| sys1db3d | 9.33 | 14.95 | 20.00 |
| sys1db4d | 9.59 | 14.75 | 20.00 |
| sys1ext0p | 2.98 | 8.51 | 16.00 |
| Total | 171.82 | 290.13 | 386 |

- ▶ More than 2 GB of physical memory is used for all the LPARs, as shown by Table 1-6.

Table 1-6 KPI-G - test-2 - AIX memory usage

| LPAR | Application memory average (MB) | Application memory maximum (MB) | Physical memory |
|--------------|---------------------------------|---------------------------------|-----------------|
| sys1cip | 8.9 | 9.0 | 12.0 |
| sys1onl0p | 20.0 | 25.9 | 90.0 |
| sys1btc0p | 13.6 | 22.4 | 96.0 |
| sys1btc1p | 39.9 | 81.3 | 96.0 |
| sys1btc2p | 40.2 | 74.0 | 96.0 |
| sys1btc3p | 40.6 | 80.2 | 96.0 |
| sys1btc4p | 40.1 | 79.5 | |
| sys1db0d | 42.3 | 43.3 | 64.0 |
| sys1db1d | 341.6 | 348.8 | 350.0 |
| sys1db2d | 334.3 | 339.3 | 350.0 |
| sys1db3d | 332.9 | 338.0 | 350.0 |
| sys1db4d | 333.9 | 339.8 | 350.0 |
| sys1ext0p | 19.5 | 28.4 | 96.0 |
| Total | 19.5 | 28.4 | 2,142.00 |

- The STR-D types of query⁵ are the main contributor to the query response time and the number of I/Os, as shown in Figure 1-15.

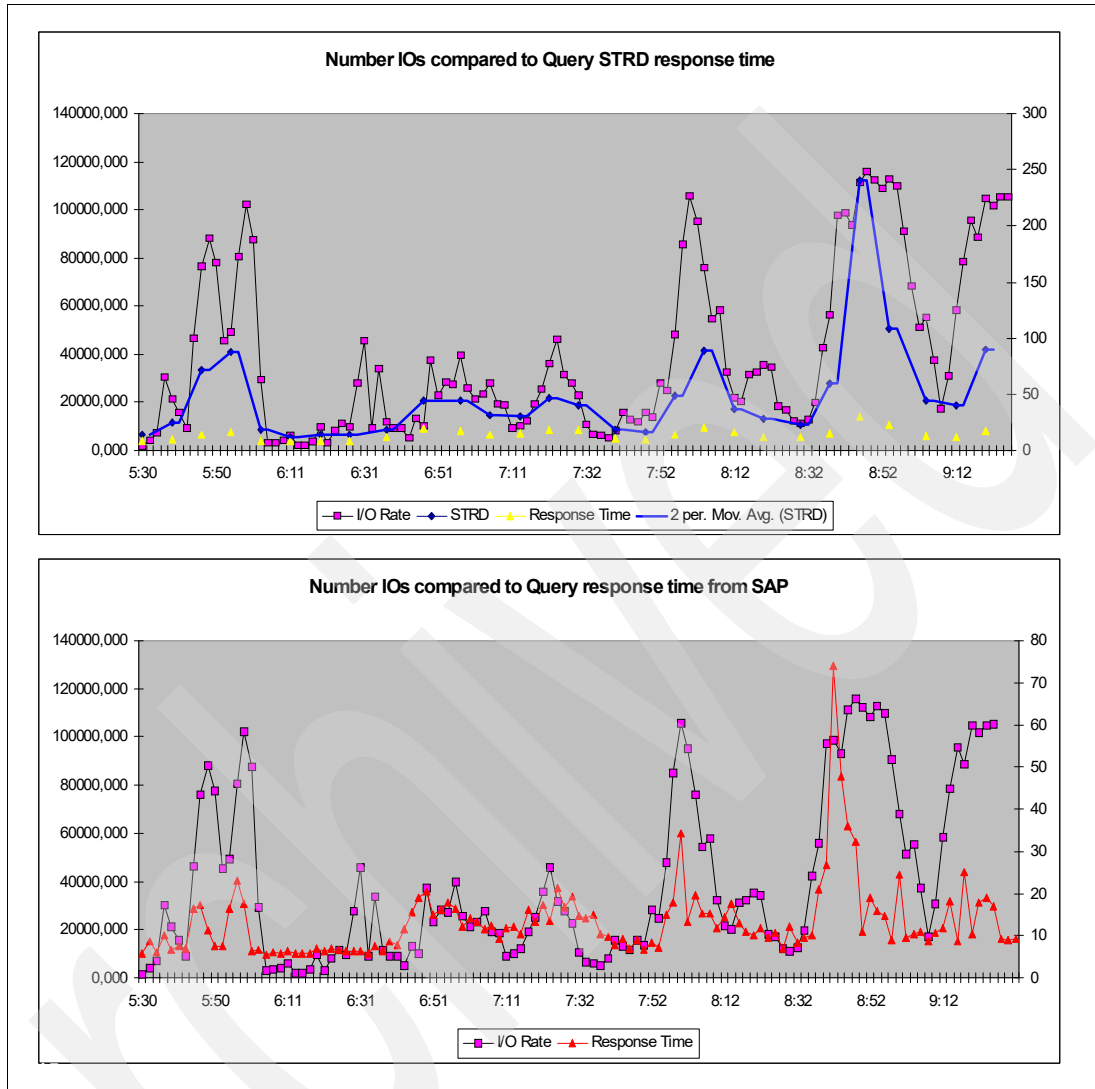


Figure 1-15 KPI-G test-2 I/O consideration

Test-3

Table 1-7 describes the configuration parameters for test-2. Compared to the two previous tests, the number of aggregates has been more than doubled, with 10 identical InfoCubes with 21 aggregates. The aggregation load profiles here are very different with the same rollup sequence. The number of Maxprocs is increased compared to test-1, a little bit below the value from test-2.

Table 1-7 KPI-G test-3 configuration parameters

| Parameters | Values |
|------------|---------|
| Maxprocs | 4 x 44 |
| DIA_WP | 4 x 100 |

⁵ The STR-D queries are queries that are not satisfied by any of the more efficient methods but must go directly to the large fact tables. Other queries can use the OLTP cache of SAP or have aggregates that answer to their needs. The STR-D are the ad hoc queries, as opposed to the standard queries where aggregates have been prepared.

| Parameters | Values |
|------------|--------|
| pUsers | 60 |
| Extractors | 8 |
| Load Cubes | 24 |
| Roll Cubes | 12 |
| Aggregates | 21 |

Figure 1-16 shows the consolidation report for test-3. Most of the objectives are achieved. The aggregation objective is very close, though not achieved.

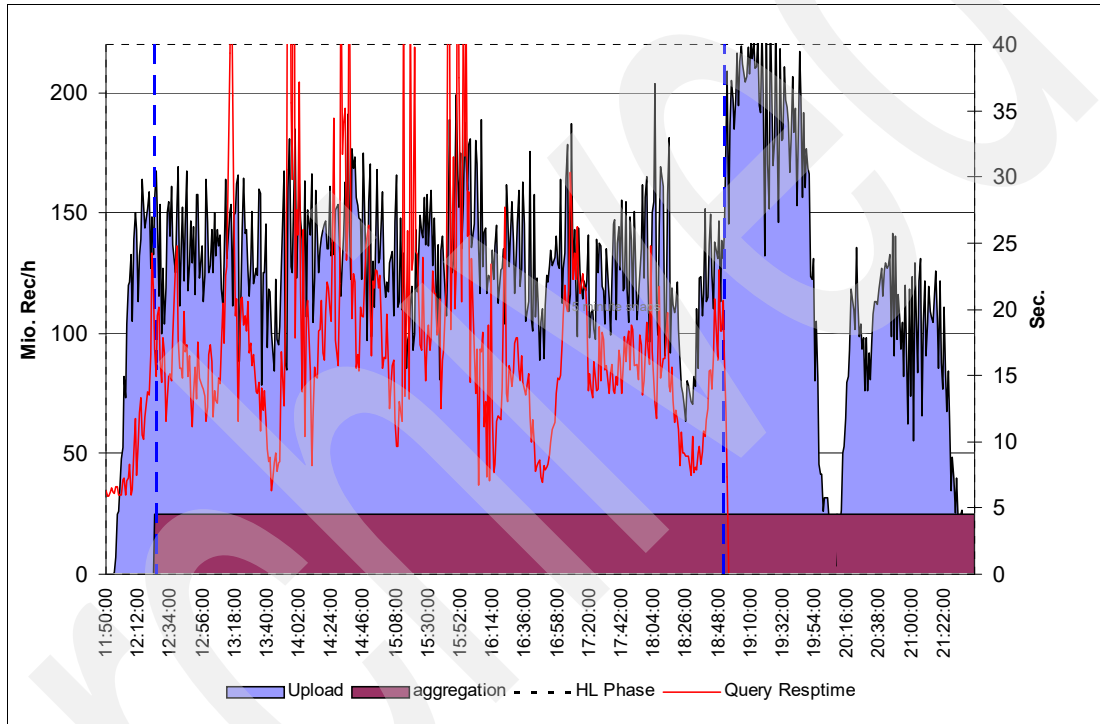


Figure 1-16 KPI-G test-3 report

Figure 1-17 shows the consolidated view of the aggregate load (10 cubes with 21 aggregates).

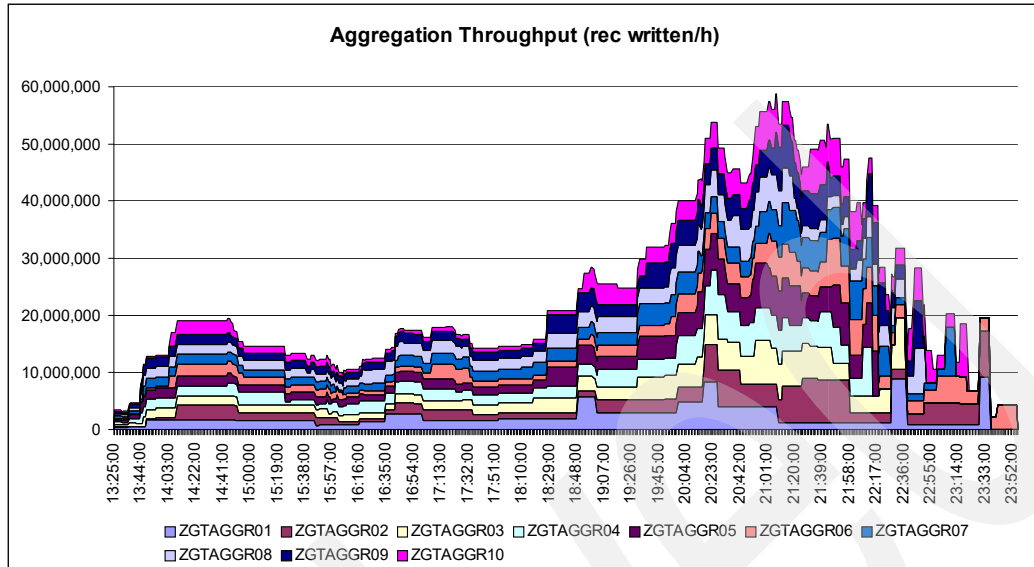


Figure 1-17 KPI-G test-3 aggregation throughput

Figure 1-18 shows the average response time per transaction.

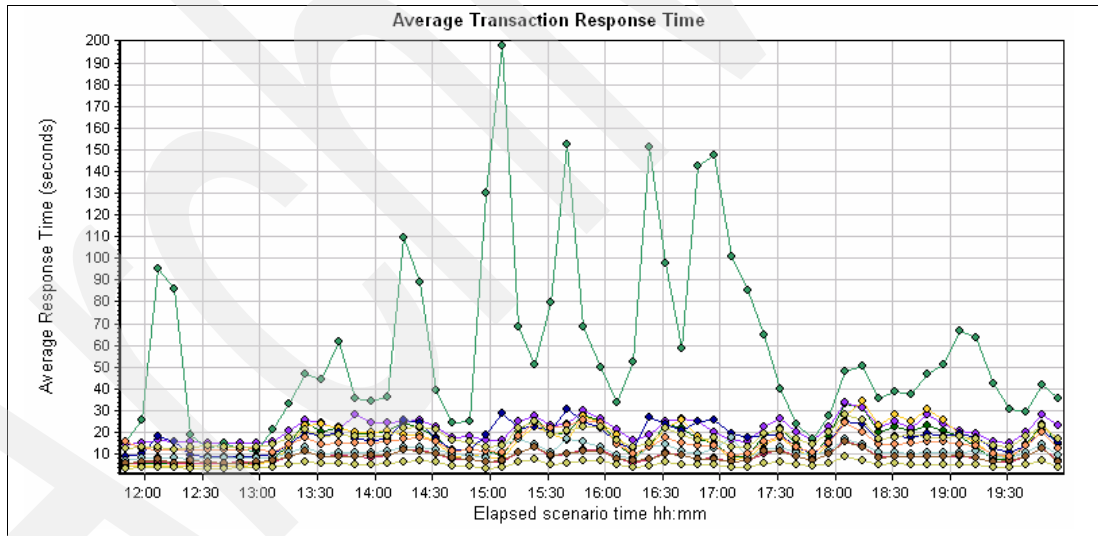


Figure 1-18 KPI-G - test-3 - 1uery load response time

Figure 1-19 shows the number of transactions per second.

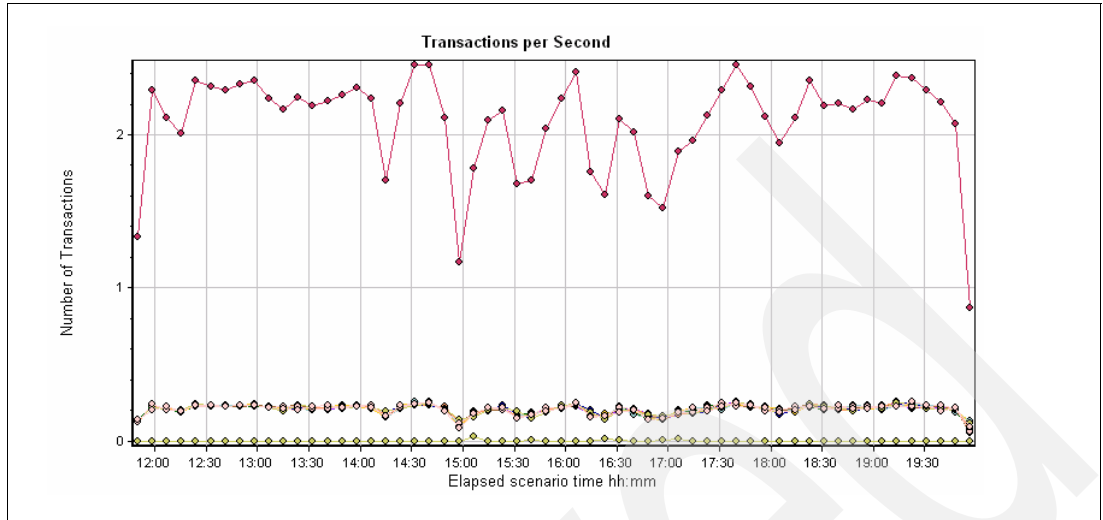


Figure 1-19 KPI-G - test-3 transactions per second

In this test we observe that:

- The physical reads almost disappear after queries are stopped. The buffer pool quality then increases, as shown in Figure 1-20.

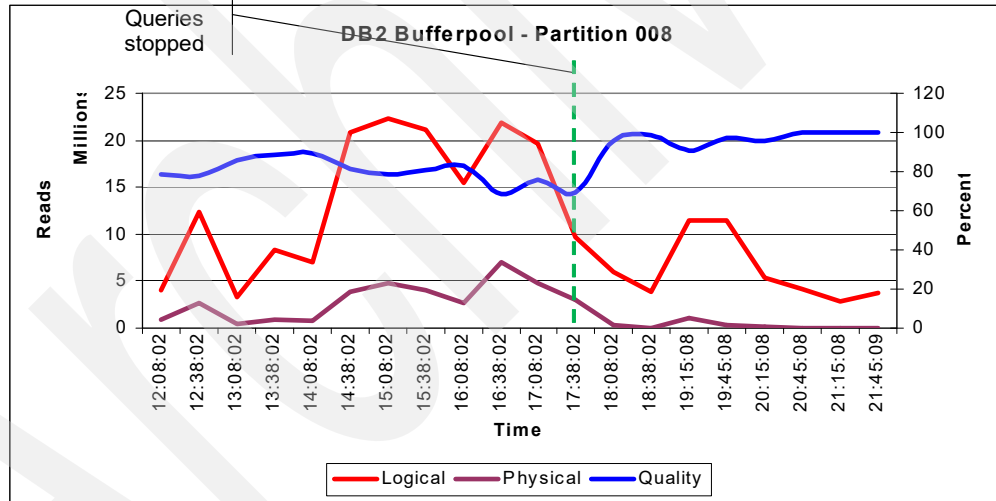


Figure 1-20 KPI-G - test-3 buffer pool quality

- ▶ The CPU consumption is a little bit more compared to test-2 numbers. Figure 1-21 shows the System p5 CPU consumption. The legend to the left provides the number of CPs given for each LPAR. The graph summarizes the CPU consumption for all the LPARs.

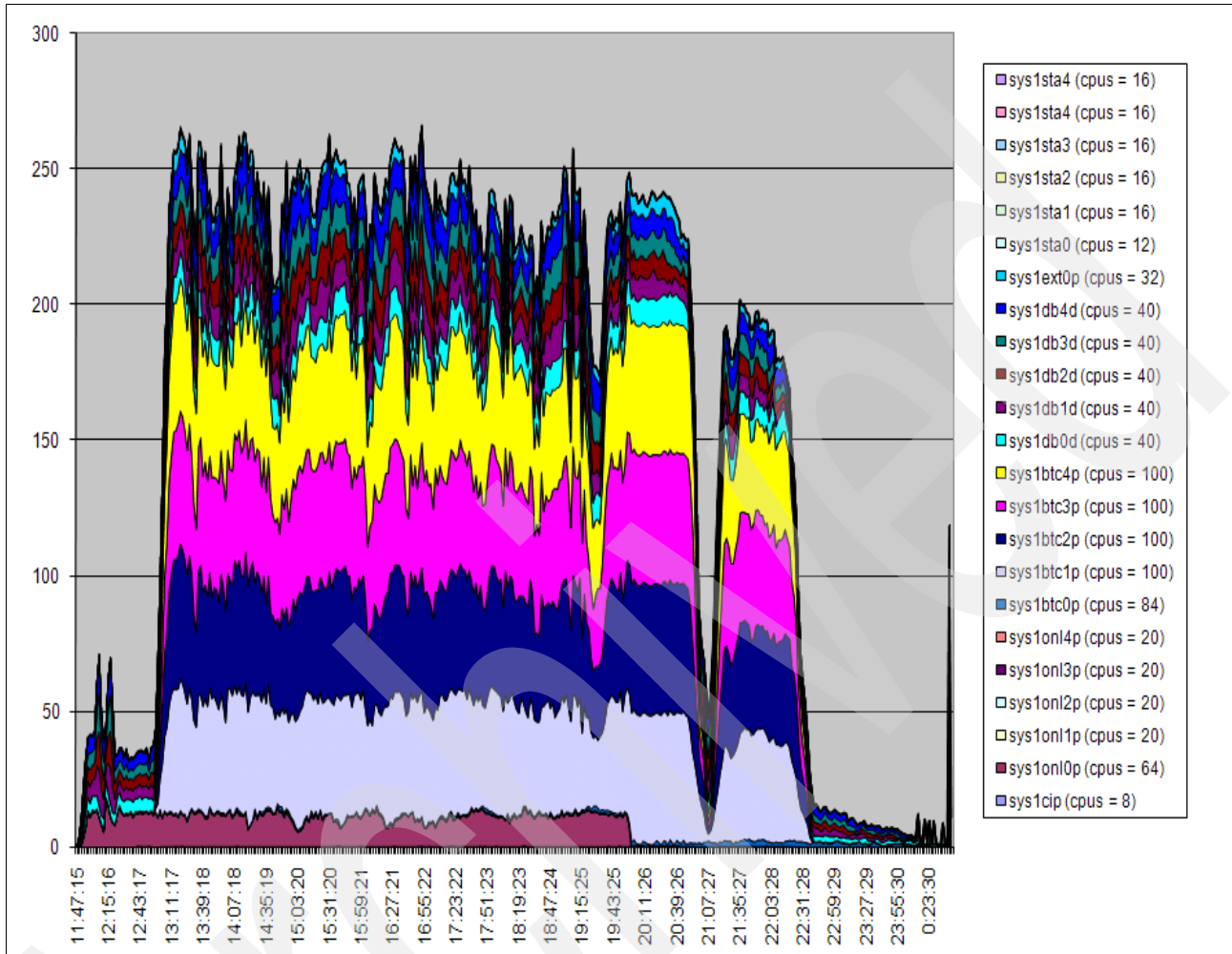


Figure 1-21 KPI-G - test-3 CPU consumption

Table 1-8 provides more details about this CPU consumption. It provides the peak and the average CPU consumption per LPAR.

Table 1-8 KPI-G - test-3 - CPU max and average per LPAR

| LPAR | Average physical CPU | Maximum physical CPU | VP |
|-----------|----------------------|----------------------|-------|
| sys1cip | 0.21 | 0.53 | 4.00 |
| sys1onl0p | 7.12 | 17.84 | 32.00 |
| sys1btc0p | 0.80 | 2.80 | 42.00 |
| sys1btc1p | 29.39 | 48.61 | 50.00 |
| sys1btc2p | 29.43 | 49.67 | 50.00 |
| sys1btc3p | 30.06 | 48.99 | 50.00 |
| sys1btc4p | 29.47 | 49.21 | 50.00 |

| LPAR | Average physical CPU | Maximum physical CPU | VP |
|--------------|----------------------|----------------------|------------|
| sys1db0d | 7.65 | 11.53 | 20.00 |
| sys1db1d | 8.44 | 16.27 | 20.00 |
| sys1db2d | 8.23 | 16.21 | 20.00 |
| sys1db3d | 8.11 | 16.21 | 20.00 |
| sys1db4d | 8.25 | 16.27 | 20.00 |
| sys1ext0p | 2.93 | 7.90 | 16.00 |
| Total | 170.08 | 302.05 | 394 |

- More than 2 GB of physical memory is used for all the LPARs, as shown by Table 1-9. This about the same as in the previous tests.

Table 1-9 KPI-G - test-3 - AIX memory usage

| LPAR | Application memory average (MB) | Application memory maximum (MB) | Physical memory |
|--------------|---------------------------------|---------------------------------|-----------------|
| sys1cip | 11.5 | 11.9 | 12.0 |
| sys1onl0p | 23.7 | 32.6 | 90.0 |
| sys1btc0p | 19.6 | 24.8 | 96.0 |
| sys1btc1p | 46.6 | 84.1 | 96.0 |
| sys1btc2p | 47.4 | 82.4 | 96.0 |
| sys1btc3p | 51.9 | 88.9 | 96.0 |
| sys1btc4p | 51.9 | 85.1 | 96.0 |
| sys1db0d | 47.7 | 49.8 | 64.0 |
| sys1db1d | 344.2 | 349.3 | 350.0 |
| sys1db2d | 339.4 | 344.8 | 350.0 |
| sys1db3d | 337.7 | 342.1 | 350.0 |
| sys1db4d | 339.9 | 344.5 | 350.0 |
| sys1ext0p | 24.1 | 31.5 | 96.0 |
| Total | | 1,871.6 | 2,142.0 |

- The total I/O rate is low with a maximum of 30,000 I/O per second, as shown in Figure 1-22.

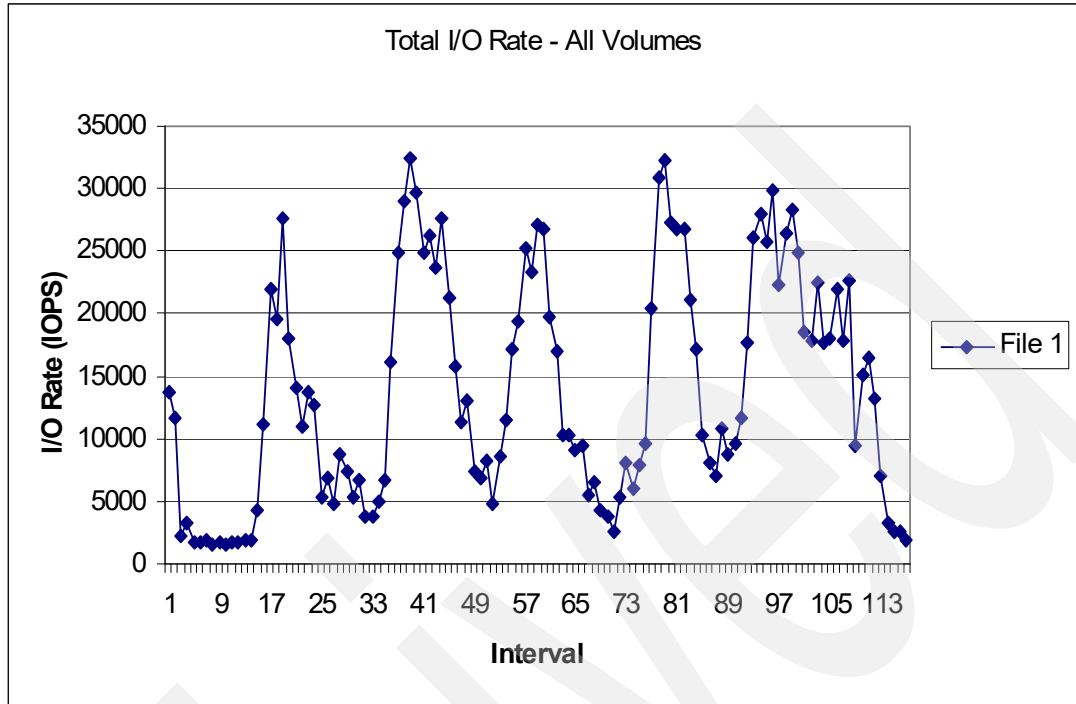


Figure 1-22 KPI-G - test-3 - total I/O rate

The response time per I/O is good, as shown in Figure 1-23.

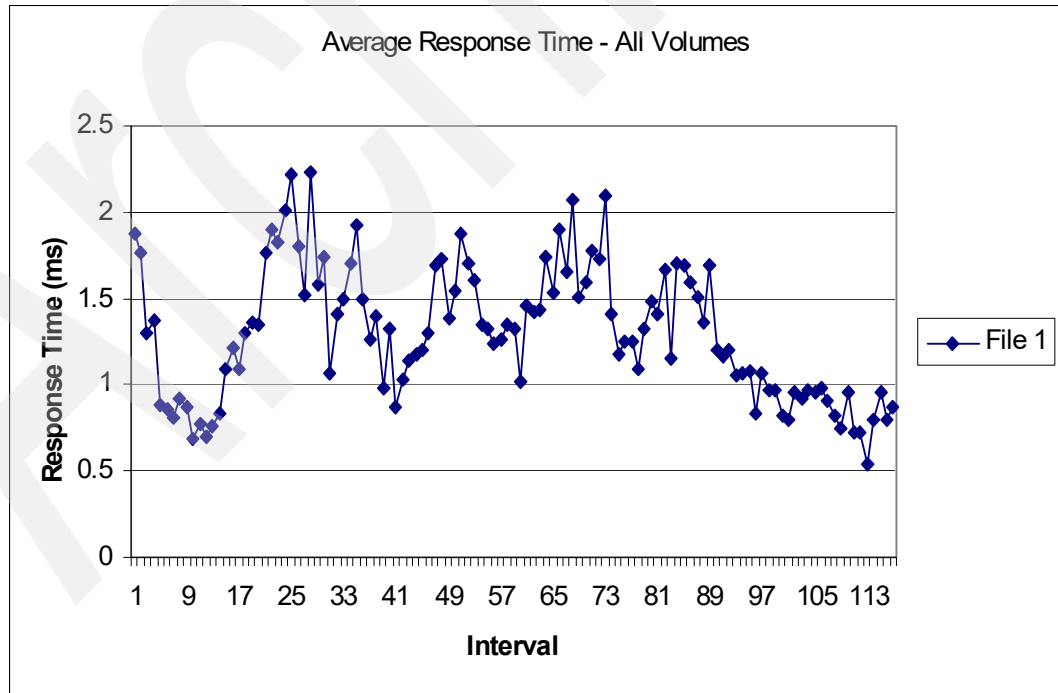


Figure 1-23 KPI-G - test-3 - average I/O response time

The Fibre Channel utilization, around 25% max, is low.

The average response time (from disk to DS8000 cache) can be considered normal, as shown in Figure 1-24.

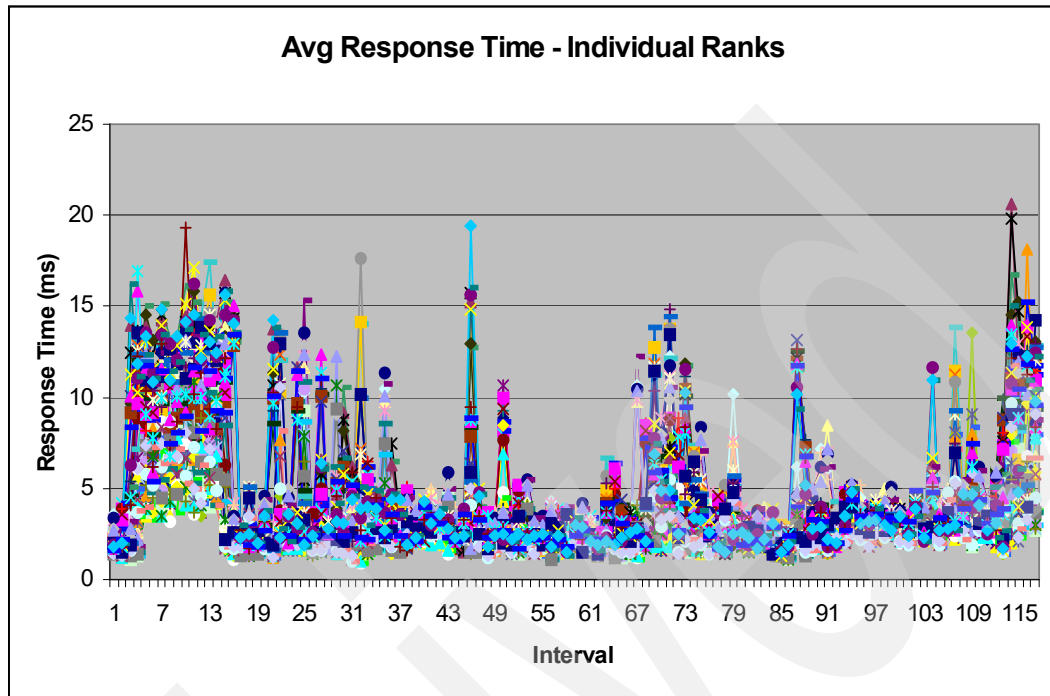


Figure 1-24 KPI-G - test-3 - average response time

There is low bandwidth utilization of the RAID array. The maximum is 25 MBps (~30%), as shown in Figure 1-25.

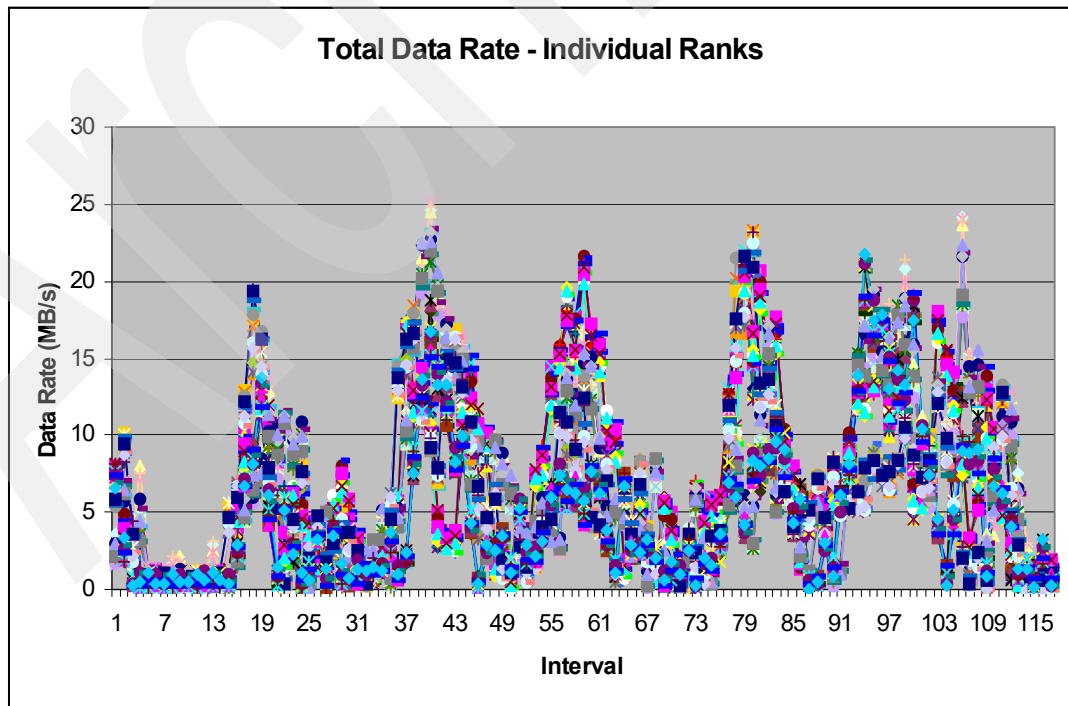


Figure 1-25 KPI-G - test-3 - total I/O rate

We observe that the maximum number of I/O per second is low.

Test-4

This test was requested to demonstrate that the results from test-3 can be repeated. Table 1-10 lists the configuration parameters for test-4.

Table 1-10 Test-1 configuration

| Parameters | Values |
|------------|---------|
| Maxprocs | 4 x 44 |
| DIA_WP | 4 x 100 |
| pUsers | 60 |
| Extractors | 8 |
| Load cubes | 24 |
| Roll cubes | 12 |
| Aggregates | 21 |

Figure 1-26 shows the consolidation report for test-4. Most of the objectives are achieved: the aggregation objective is very close, though not achieved. The number of transactions per second is very close, though not achieved. The response time per transaction is pretty good.

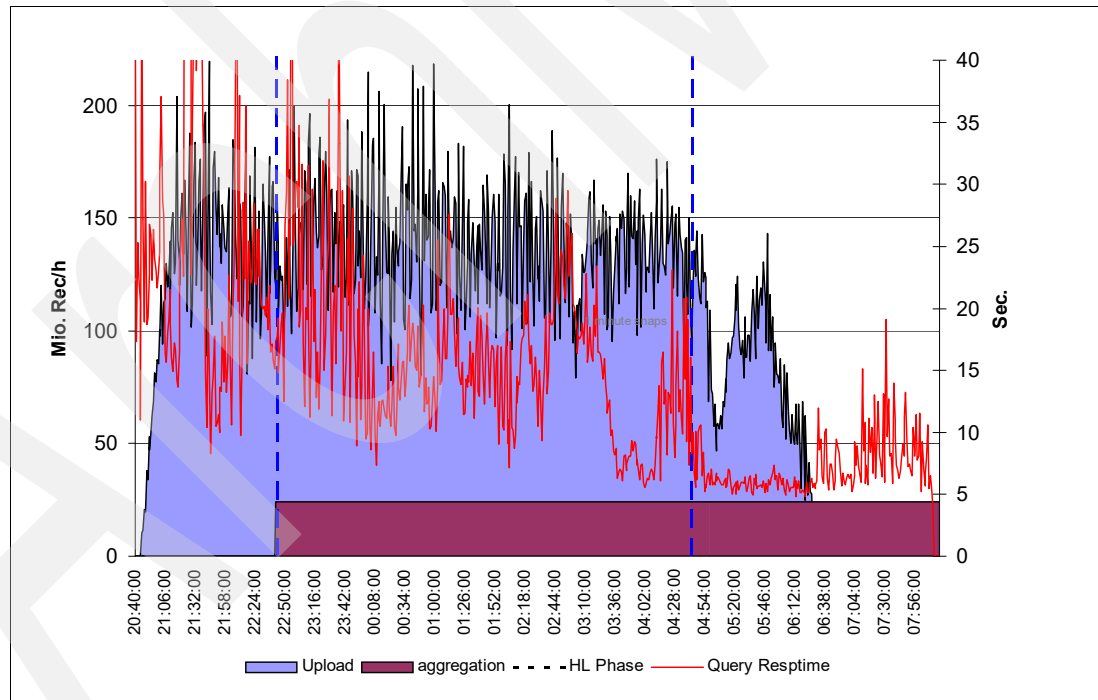


Figure 1-26 KPI-G - test-4 - summary report

Figure 1-27 shows the impact on response time caused by STR-D types of queries. In the chart, the average response time includes the response times for all queries, whereas the selective response times show the average response times of all queries except STR-D. This chart is provided to demonstrate the significant impact of the STR-D type of queries.

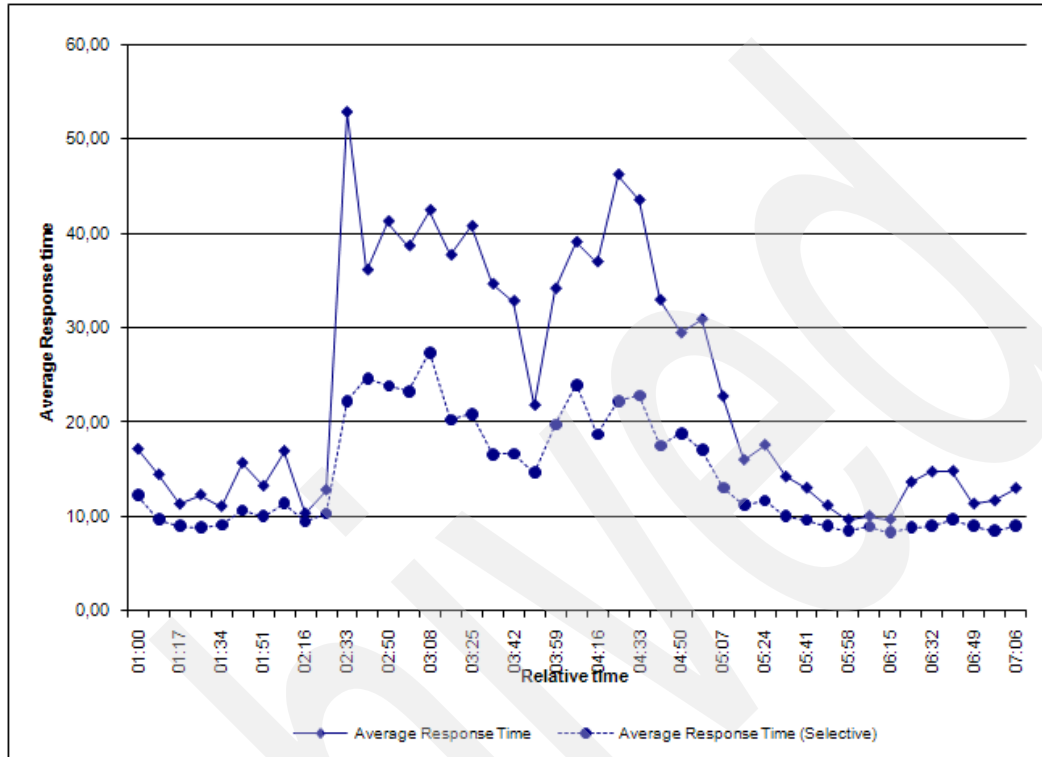


Figure 1-27 fKPI-G - test-4 - SAP STR-D response time

Figure 1-28 shows the average query response times of each of the query types.

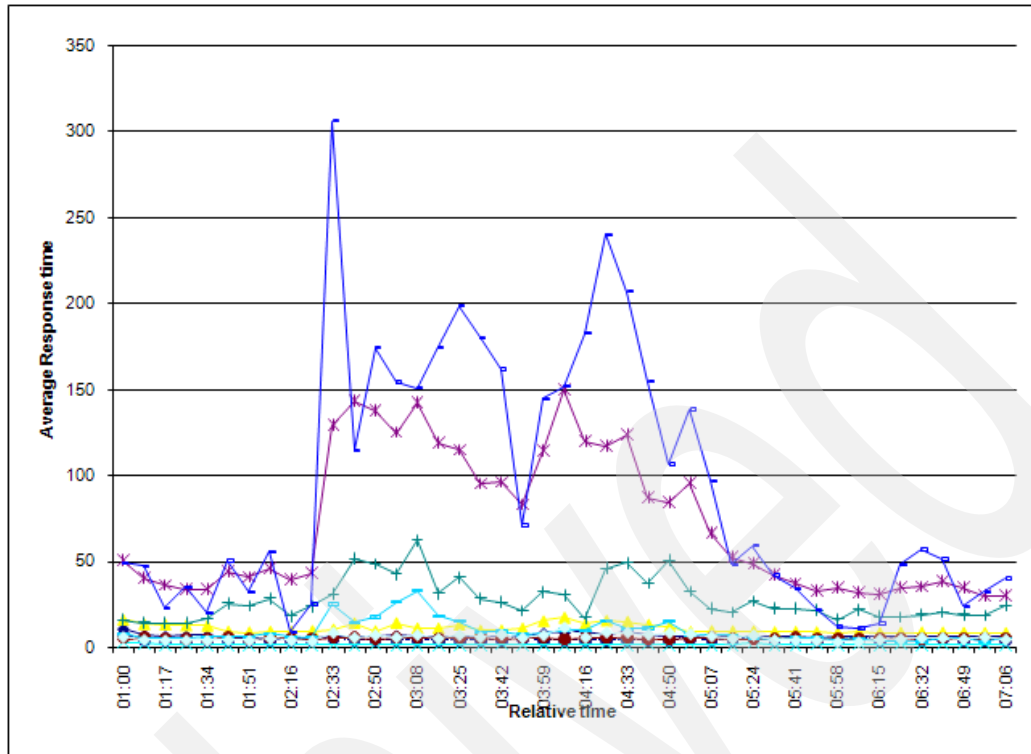


Figure 1-28 KPI-G - Test4 - Selective response time

On average, if the number of STR_D queries is larger than 20, then the response time criteria is not met.

To further analyze the impact of the STR_D type of queries on the overall average query response time, a small script was written to monitor the actual number of STR_D queries running in parallel. Figure 1-29 shows in orange the graph corresponding to the number of STR_D queries running in parallel (note that the graph does not have a scale for representing the actual number of STR_D queries in parallel). We find that the trend of the orange STR_D graph relates very well to the response time graph and therefore conclude that STR_D query types *are* the most important influence on query response times. From the number of STR_D queries running in parallel, we can also conclude that if this number is more than 20, we will most likely not achieve the query response time targets.

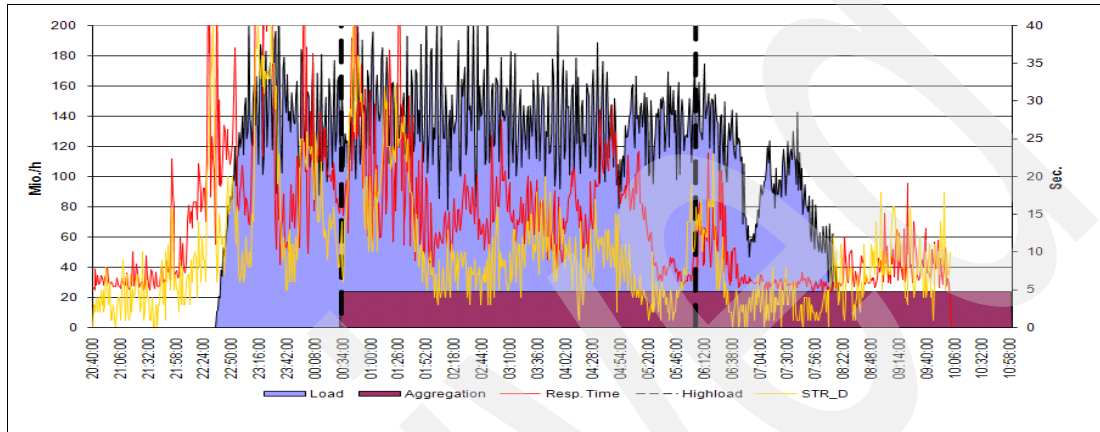


Figure 1-29 STR_D queries running in parallel

Figure 1-30 provides a close-up view of the striking relationship between the number of STR_D queries and the average query response time.

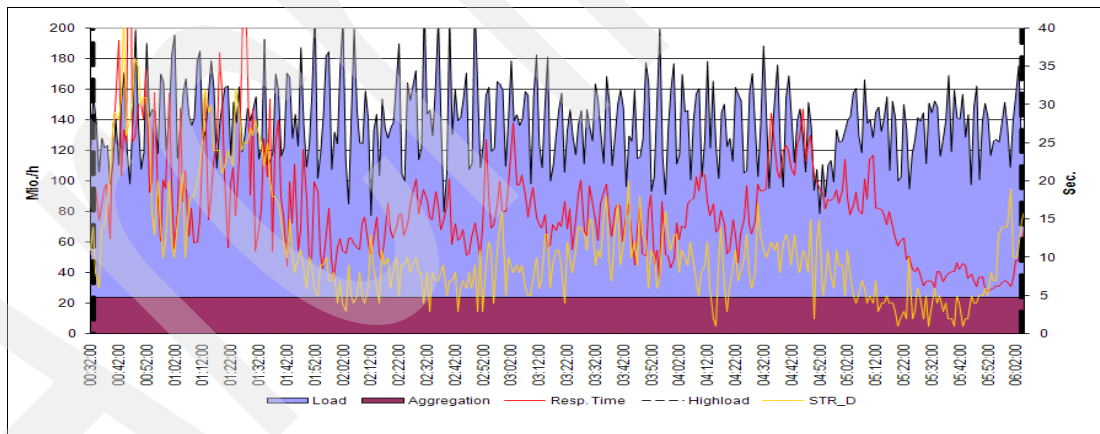


Figure 1-30 Relationship between the number of STR_D queries and the average query response time

We can observe that:

- ▶ The CPU consumption is a little bit more compared to the test-2 numbers. Figure 1-31 shows the System p5 CPU consumption. The legend of the left provides the number of CPs given for each LPAR. The graph summarizes the CPU consumption for all the LPARs.

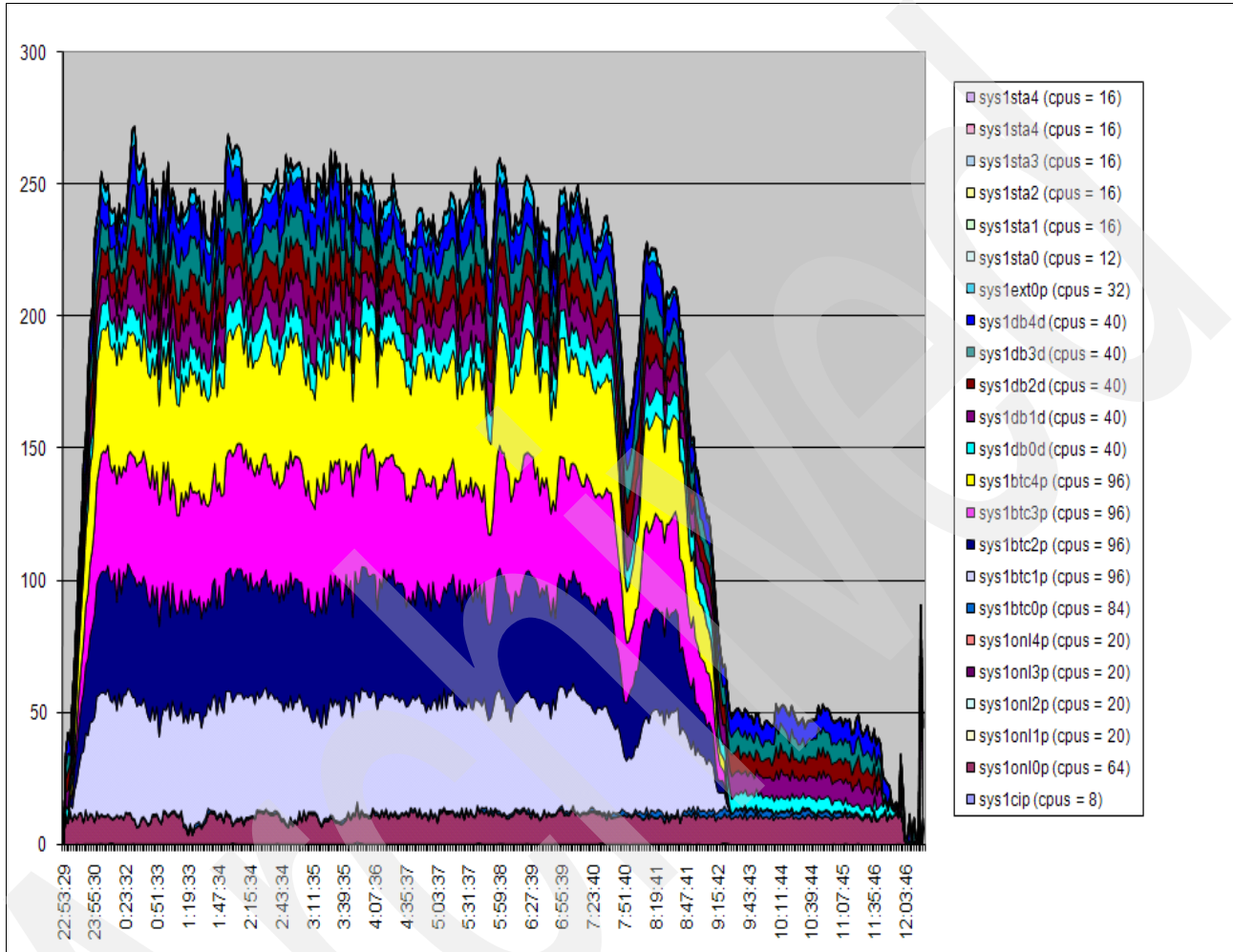


Figure 1-31 KPI-G - test-4 - CPU consumption

Table 1-11 provides more details about this CPU consumption. It provides the peak and the average CPU consumption per LPAR.

Table 1-11 KPI-G - test-4 - CPU max and average per LPAR

| LPAR | Average physical CPU | Maximum physical CPU | VP |
|-----------|----------------------|----------------------|-------|
| sys1cip | 0.22 | 0.97 | 4.0 |
| sys1onl0p | 10.08 | 14.15 | 32.00 |
| sys1btc0p | 0.80 | 3.12 | 42.00 |
| sys1btc1p | 29.78 | 47.38 | 48.00 |
| sys1btc2p | 30.19 | 47.60 | 48.00 |

| LPAR | Average physical CPU | Maximum physical CPU | VP |
|--------------|----------------------|----------------------|------------|
| sys1btc3p | 29.97 | 47.41 | 48.00 |
| sys1btc4p | 30.65 | 47.54 | 48.00 |
| sys1db0d | 8.56 | 11.69 | 20.00 |
| sys1db1d | 10.31 | 15.79 | 20.00 |
| sys1db2d | 10.18 | 15.66 | 20.00 |
| sys1db3d | 10.01 | 16.03 | 20.00 |
| sys1db4d | 10.17 | 15.55 | 20.00 |
| sys1ext0p | 2.95 | 7.52 | 16.00 |
| Total | 183.86 | 290.39 | 386 |

- More than 2 GB of physical memory is used for all the LPARs, as shown by Table 1-12. This about exactly the same as in the previous test.

Table 1-12 KPI-G - test-3 - AIX memory usage

| LPAR | Application memory average (MB) | Application memory maximum (MB) | Physical memory |
|--------------|---------------------------------|---------------------------------|-----------------|
| sys1cip | 9.3 | 9.8 | 12.0 |
| sys1onl0p | 23.4 | 30.3 | 90.0 |
| sys1btc0p | 19.3 | 24.6 | 96.0 |
| sys1btc1p | 50.0 | 87.1 | 96.0 |
| sys1btc2p | 51.3 | 88.5 | 96.0 |
| sys1btc3p | 50.7 | 86.9 | 96.0 |
| sys1btc4p | 51.5 | 88.1 | 96.0 |
| sys1db0d | 43.6 | 44.9 | 64.0 |
| sys1db1d | 339.9 | 343.4 | 350.0 |
| sys1db2d | 340.1 | 344.1 | 350.0 |
| sys1db3d | 340.0 | 345.1 | 350.0 |
| sys1db4d | 337.2 | 341.6 | 350.0 |
| sys1ext0p | 24.1 | 29.9 | 96.0 |
| Total | | 1,864.2 | 2,142.0 |

- Figure 1-32 and Figure 1-33 show the total I/O rate in the interval. They present the I/O rate across all volumes during the entire KPI. The chart correlates nicely with the number of STR_D queries running in parallel.

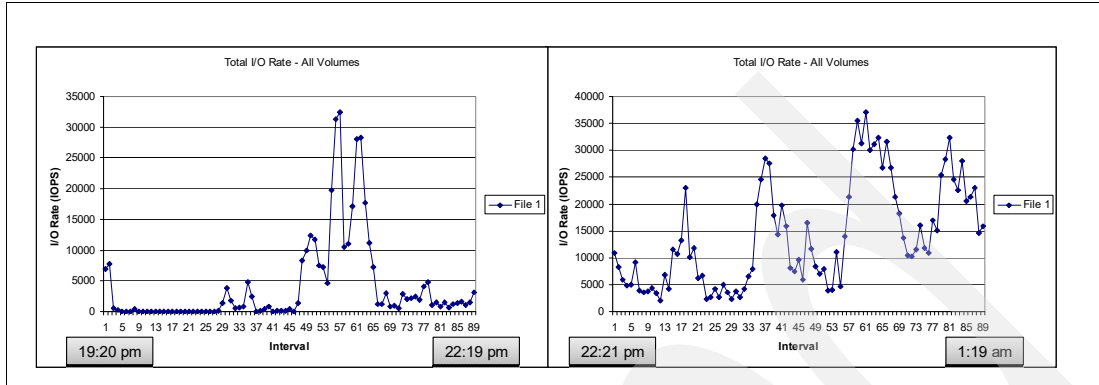


Figure 1-32 KPI-G - test-4 - total I/O rate

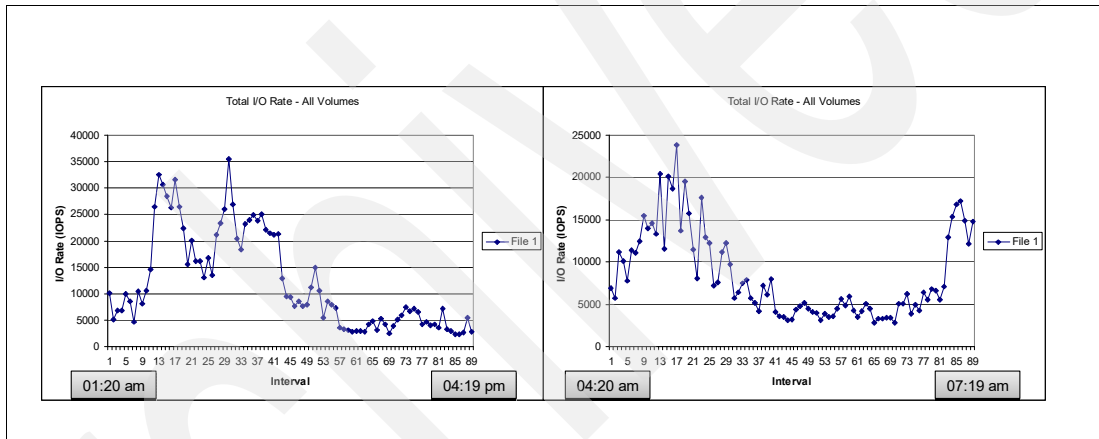


Figure 1-33 KPI-G - test-4 - total I/O rate

1.2.4 Trends

This section focuses on the trends recognized over the lifetime of the test that play major roles in predicting resource requirements and scalability of the solution.

Load profiling

The objective of phase 1 was to study the load behavior, both in resource consumption and I/O requirements, in order to size the final configuration. This was done primarily for CPU and I/O requirements.

Figure 1-34 shows the number of I/Os per type of load.

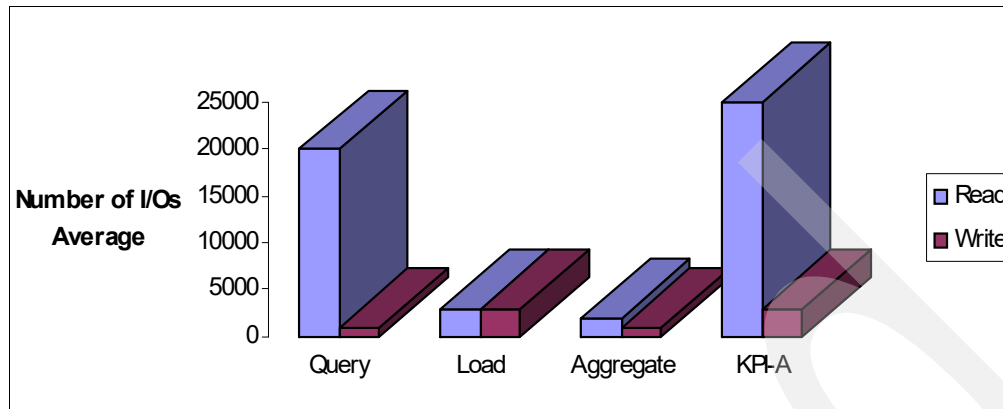


Figure 1-34 Load profiling for I/O capacity

Figure 1-35 shows the load profiling done for the concurrent online load in distribution ratios of DB versus the application servers. Query and aggregation have similar general resource requirements.

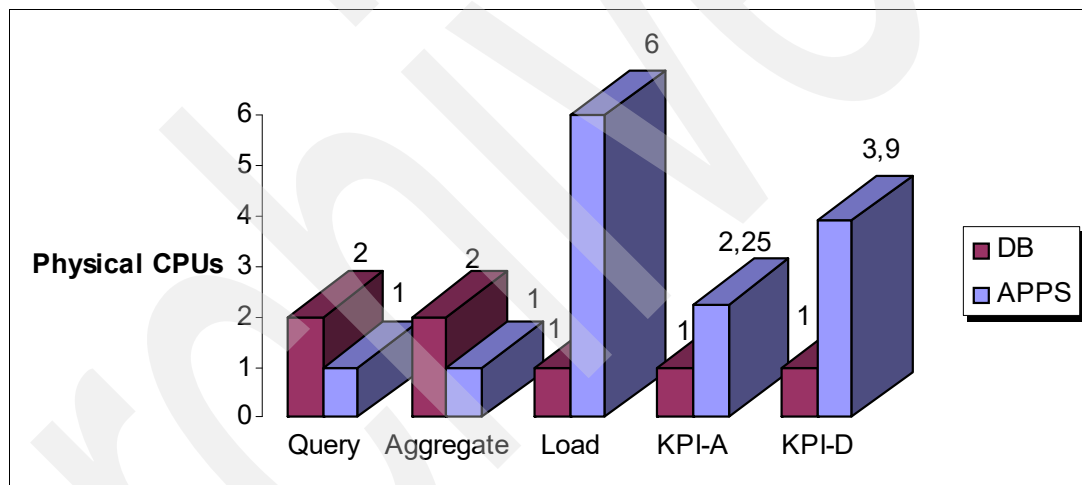


Figure 1-35 Load profiling for resource consumption

The data loading is primarily application server load. Application servers:

- ▶ Are to a high proportion independent of each others' load.
- ▶ Can be scaled out. Additional application servers can be added.

The trend shows that the App:DB ratio increases as the load scales up.

- ▶ In KP-ID (three times the baseline), the combined ratio was 3.9 APPS:DB.
- ▶ in KPI-G (five times the baseline), shown in Figure 1-36, the application server requirement in relation to the DB requirement has dropped to 2.72:1. The additional DB load is coming from the scale-up of the aggregation.

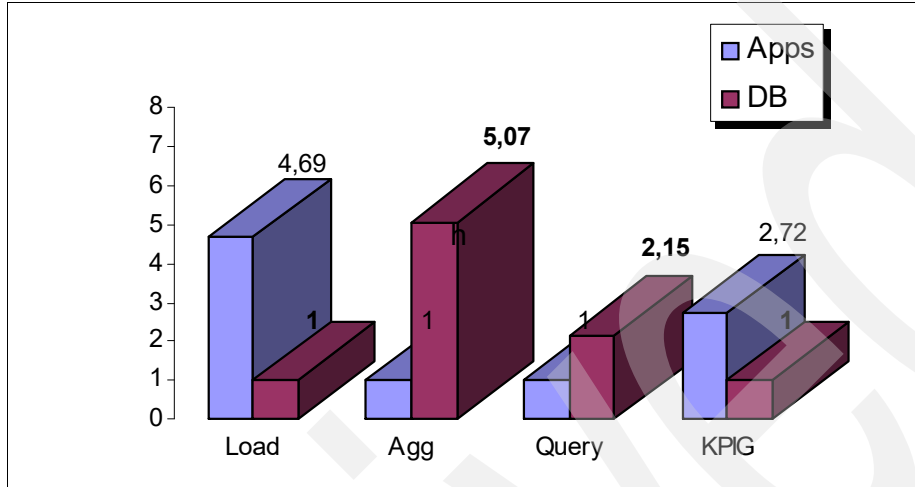


Figure 1-36 Load profiles of KPI-G

Trends for aggregation requirements

Figure 1-37 and Table 1-13 show the behavior of aggregation across the project evolution stages in the form of the CPU requirements on the DB servers.

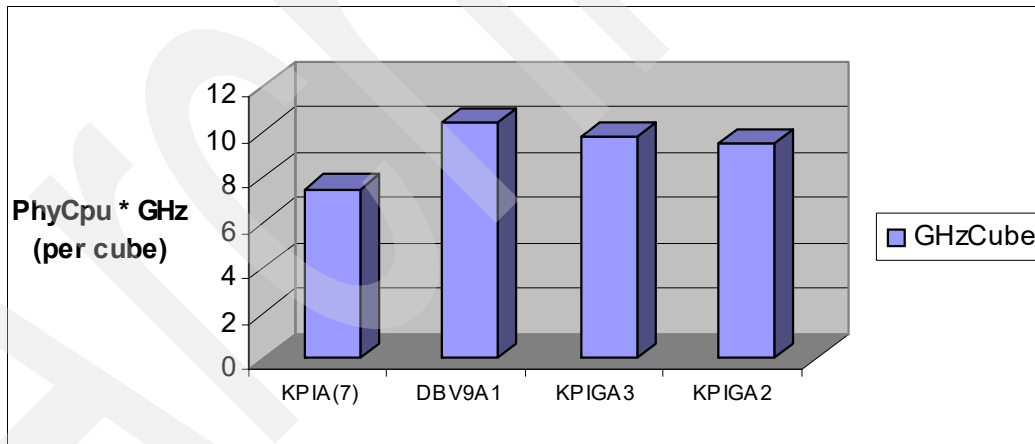


Figure 1-37 Aggregation scaling

Table 1-13 Aggregation scaling

| Run ID | GHz/cube | Agg | Config | Comments |
|---------|----------|-----|--------|----------------------|
| KPI-A | 7.492294 | 10 | 7 TB | 2 cubes type A |
| DB9A1 | 10.43115 | 10 | 20 TB | 8 cubes mixed typesc |
| KPI-GA3 | 9.782196 | 21 | 60 TB | 8 cubes type A |
| KPI-GA2 | 9.494389 | 10 | 60 TB | 8 cubes type A |

The CPU capacity used in the unit tests is translated into GHz to cover the speed bump along the way. The total GHz used is divided by the number of cubes running concurrently to show the capacity requirement for cube. A cube represents a batch job running in the app-servers. Each batch job is handling a number of aggregates, depending on the cube design. In the 7-20 TBs, there were two different cube structures for aggregation: A and B. They represent the *sales and distribution* and *profit analysis* cube designs. In the 60 TB, only the type A cube was built for aggregation tests. Therefore, the statistics cover two cube designs and two cube layouts. Some cubes have 10 aggregate and others have 21 aggregates.

The tests ranged over all the available cube configurations and combinations. The results on a per-cube basis are relatively stable in CPU requirements. The DB for run DBV9A1 (KPI-D at 20 TBs) was slightly memory bound and therefore does show higher CPU overhead.

The difference between KPI-A 7 TBs and KPI-G spans the entire range of changes in the infrastructure, including the move into virtualization and the additional contention of eight cubes. This gives a rudimentary sizing picture — the expectation of CPU requirement for x concurrent cubes.

Scaling and resource consumption

Figure 1-38 shows the positive scaling development over the test life cycle. Using KPI-A as the baseline, extrapolation are done for three times and five times the load. The extrapolation is done in GHz. The reason is the infrastructure upgrade between KPI-A and KPI-D (1.9 Ghz to 2.3 Ghz).

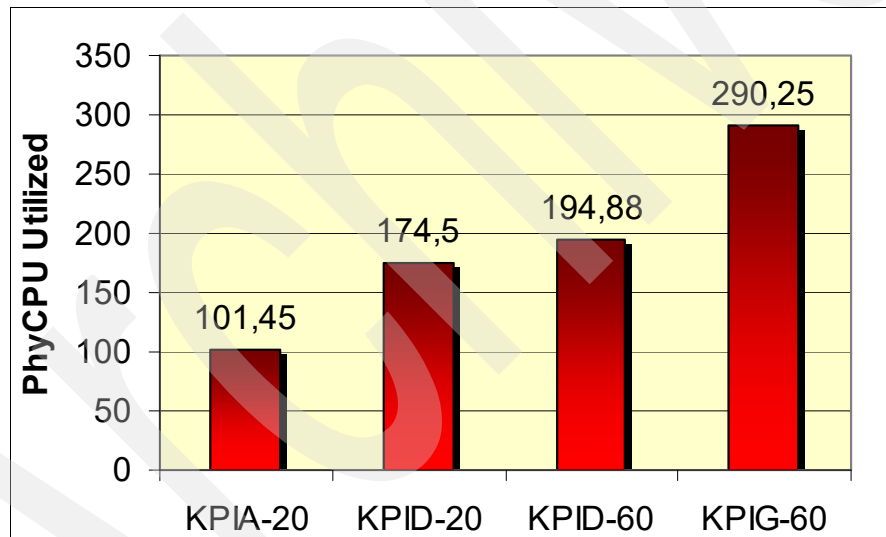


Figure 1-38 Total physical CPUs consumed

Figure 1-39 shows the actual resource consumption (not using KPI-A as the baseline) and the expectation of the consumption. The actual consumption is significantly less than predicted because of some infrastructure improvements (statviews, DB2 V9) and tuning during the tests.

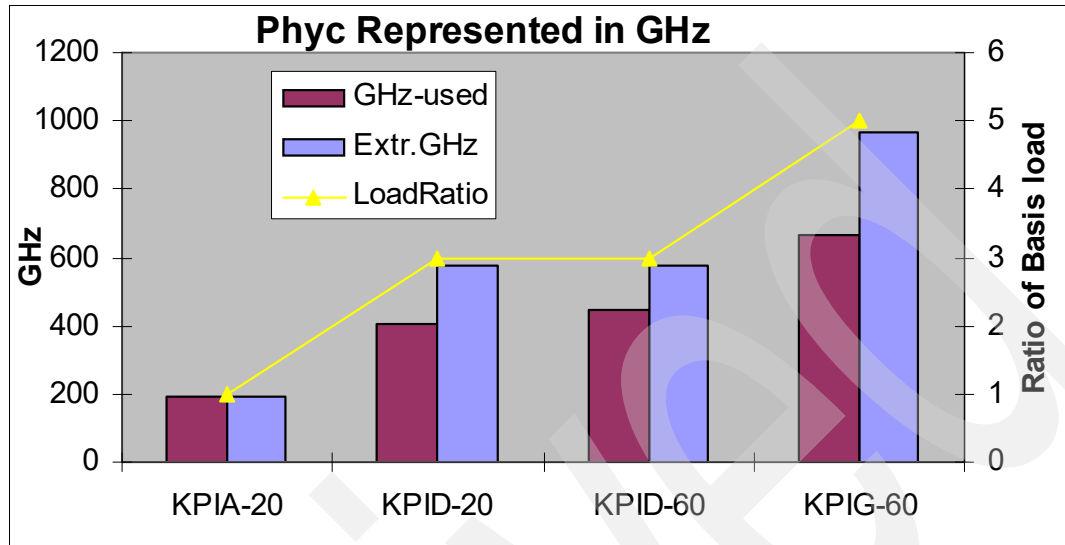


Figure 1-39 Predicted resource consumption versus actual utilization

The DB is the focal point of an SAP implementation. For the solution to scale, it is vital that the DB scales. Figure 1-40 shows the CPU utilization distribution between the application servers and the DB. We expect the application server load to increase far in excess of the DB load due to the CPU-intensive cube-loading scenario. What is interesting here is the increase in DB requirements for the same load at 20 TB and then at 60 TB.

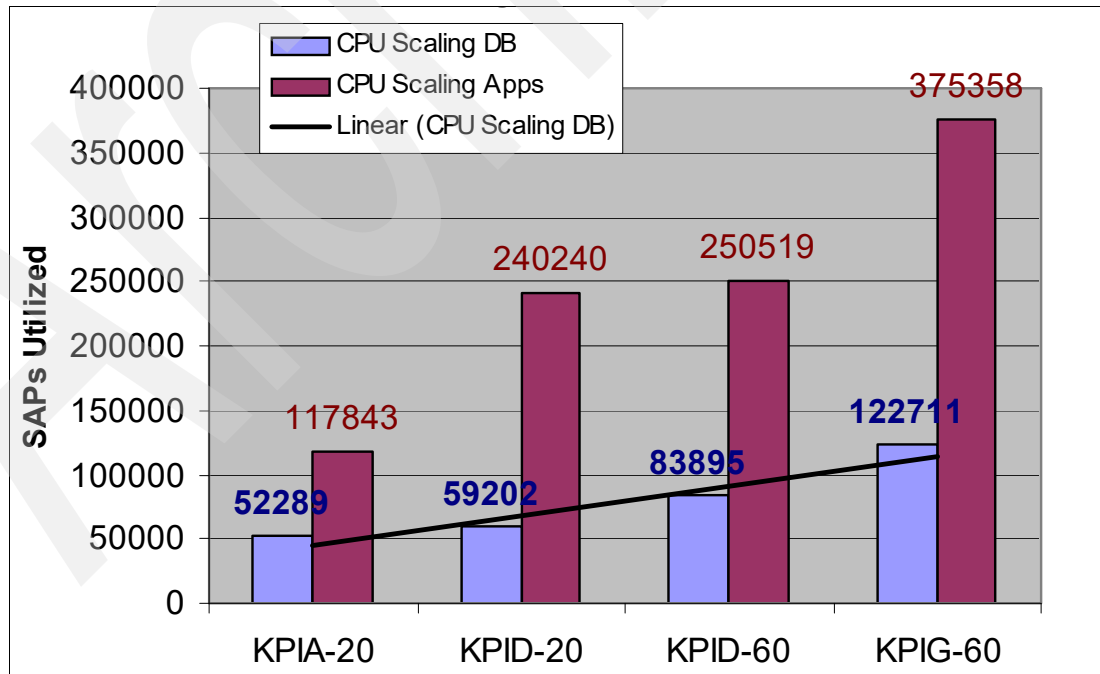


Figure 1-40 DB scaling

This will be the result of the ab hoc queries that hit 40–48 million rows at 20 TB, and 200 million rows in the fact tables at 60 TB. Utilization is shown in physical CPU utilized and converted into SAPs⁶.

Solution scalability

Figure 1-41 shows the scalability of the load in relation to the infrastructure requirements needed to support the load. KPI-A represents the baseline in this graph, and all values are in scaling relation to KP-IA. Units are translated to GHz.

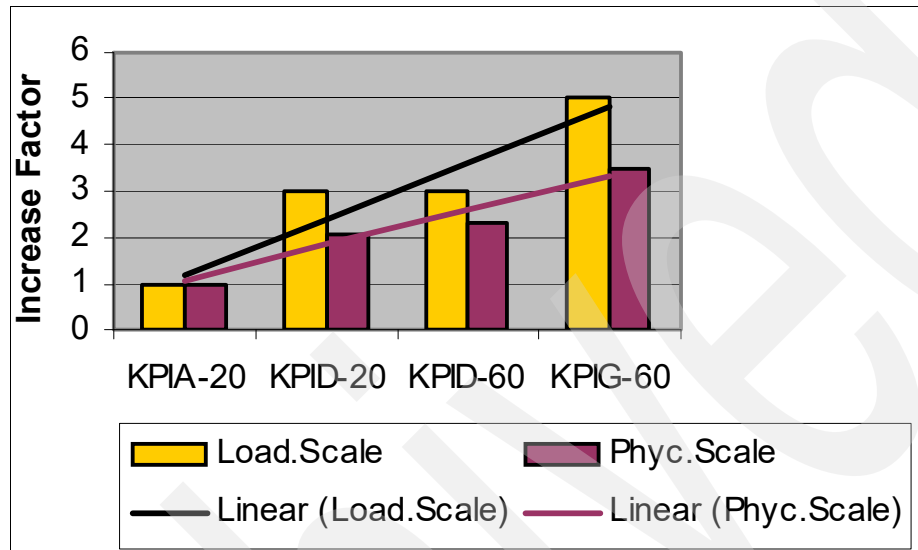


Figure 1-41 Infrastructure scaling

The trend is very positive for solution scalability. The use of virtualization is likely to improve this scalability in the high-end. In this overview, KPI-G is running on virtualization, but it is not utilizing the full physical footprint. Virtualization shows the most benefits when there is contention for CPU. In this case it is normal to see an ever-increasing throughput on the same physical hardware

1.3 The infrastructure KPIs

Table 1-14 shows that all the infrastructure KPIs were easily achieved. They are detailed in Chapter 6, “The Tivoli Storage Manager perspective” on page 347.

Table 1-14 Infrastructure KPIs results

| Reference | Description | Objective | Achievement |
|-----------|--|-----------|-------------|
| KPI-3a | Backup to tape. | < 12h | 6h10mn |
| KPI-1 | Restore database and roll forward 500 GB of logs. | < 8h | 2h40mn |
| KPI-2 | Database restore from tapes and roll forward of 2 TBs of logs. | <24h | 10h00mn |

⁶ The SAP Application Performance Standard (SAPS) is a hardware-independent unit that describes the performance of a system configuration in the SAP environment. It is derived from the Sales and Distribution (SD) Benchmark. For more information go to <http://www.sap.com/solutions/benchmark/measuring/index.epx>.

Table 1-15, Table 1-16, and Table 1-17 on page 44 show the timeframes for each of the tests.

► Backup to tape: KPI-3a

Table 1-15 KPI 3A - process duration

| Time | Task |
|-------|---|
| 00h | Start TDP on all storage agents. (One process for each LPAR hosting DB partitions) TDP checks configuration and source-target volume relationships. |
| 00h17 | TDP invokes FlashCopy for DB volumes. Set database to „write suspend“ for all DB partitions. Start FlashCopy for all LUNs via the CIM Agent. Set database to „write resume“ for all DB partitions. (Some 10 seconds „offline“ time) |
| 01h06 | Mount FC LUNs on storage agents done. Acquire disk devices and volume groups, and mount file systems. Run db2inidb as standby. Start backup of partition 0 (4 sessions). |
| 01h27 | Backup of partition 0 finished. Start all backups for remaining partitions in parallel. Each individual backup uses one tape: (32 + 5) backups * 1 session = 37 tapes used. |
| 06h01 | Backup done for all partitions done. |
| 06h10 | Cleanup on storage agents done. Unmount file systems, export volume groups, remove disk devices. |

► Restore from tape: KPI-1

Table 1-16 KPI-1 - process duration

| timeline | Task |
|----------|--|
| 00 | Invoke TDP processes on all DB servers: <ul style="list-style-type: none"> ► One process started on each server hosting DB partitions. ► TDP checks available (FC and tape) backups, configuration on the DB servers, checks source-target volumes relationship, file systems, and so on. |
| 15 min. | User entries for selecting the backup done. |

| timeline | Task |
|----------|---|
| 31 min. | Final confirmation for invoking the FlashCopy restore: <ul style="list-style-type: none"> ▶ Unmount/remove all file systems, export volume groups. ▶ Start FlashCopy restore for all volumes using CIM Agent. ▶ Acquire disk devices and volume groups for all FC volumes, mount file systems. ▶ Mount of all FC file systems on backup server. |
| 1h 10 | Reactivate database <ul style="list-style-type: none"> ▶ Run „db2inidb as mirror“. ▶ Manual „Retrieve of Logs“ is skipped due to direct archiving. |
| 1h 20 | TDP processing finished. |
| 1h20 | Roll forward recovery of about 680 GB LOGs started. Direct archiving retrieves LOGs directly from Tivoli Storage Manager during the ROLLFORWARD. |
| 2h40 | Roll forward finished. |

- ▶ Roll forward of logs: KPI-2

Table 1-17 KPI-2 - process duration

| Time | Task |
|--|-------------------------------------|
| 22:30 | Tape restore started |
| 05:10 | Tape restore finished |
| Run time for the tape restore: 6 hours, 40 minutes | |
| 05:15 | Roll forward started |
| 07:53 | Roll forward recovery phase stopped |
| 08:01 | Roll forward stopped |
| 08:07 | SAP started |
| Run time for the roll forward: 2 hours, 52 minutes | |

1.4 Data compression

Several tests demonstrated the benefit of the new DB2 9 storage optimization feature. The tests are detailed in 3.4.2, “Tests with DB2 data row compression” on page 228.

Important: The tests in our specific environment have shown a real benefit of the DB2 storage optimization feature in the context of SAP NetWeaver BI. Based on the test definition and setup, the results seem very realistic and may be transferred to real live implementations to a large extent. However, in other implementations, some details of resource usage may significantly change with different characteristics of the workload, infrastructure, and options parameters.

As expected, the main benefit of using the DB2 storage optimization feature is in the area of storage savings: 48% fewer pages are used in the database, and they reduce the number of I/O operations. The counterpart could be seen in the CPU usage: the CPU usage (on the database servers) increased from 48% to 65%. However, the performances were better: the query performance increased by 23%, the aggregate throughput by 15%, and the bufferpool quality increased to 93% for the large database objects (which is an excellent result for OLAP databases).

- ▶ We observed that overall compression ratios of about 40–50% can be achieved. Compression rates go up to 85% for single tables. Indexes are not compressed and so the overall compression result will be influenced by the number and size of indexes used in the SAP NetWeaver BI system.
- ▶ From the I/O perspective, we observed that the number of I/O per second decreased by about 68%.
- ▶ While the query response time is fluctuating without compression, the query run time is much more stable when compression is enabled, because of a better bufferpool quality. The throughput for the aggregate build increased by 15%, while the average response time decreased by 23%. The average number of transactions per second remains almost the same.
- ▶ The number of reads from disk decreased. The maximum is reduced by 65% and the average is reduced by 72%. The decrease in I/O, both KB and I/O operations per second, is based on the reduced amount of data on storage, but also on the improved bufferpool quality. DB2 effectively reads less data from disk to retrieve the same number of records and avoids some I/O operations by serving requests with logical reads instead of physical reads.
- ▶ The network traffic with compression enabled was slightly higher, which probably is related to the higher throughput of the whole system. The average reads went up to 2%, while the KB written increased by 5.5%.
- ▶ The average CPU increased from 30.70% to 53%, which is a significant increase. However, wait time decreased from 23.90% to 5.70%, and also system time decreased from 16.50% to 12.30%, which is also related to the less frequent I/O calls in the system. The increase of user time is not only related to the additional CPU cycles needed for compression, but is also a result of the higher throughput in aggregate build and better response time for the query workload.

There is no overall significant increase for the CPU usage of the SAP application server. As the DB2 Storage optimization feature is transparent to the application, the expected result is no change in the CPU usage of the SAP application server. The change of average CPU usage for the three application servers from 17.70 to 19.20 is related to the increased performance of the system.

- ▶ There was almost no difference in the usage of the memory.

Archived

The SAP NetWeaver BI perspective

This chapter provides an overview of the project from an SAP technical specialist view. In this chapter:

- ▶ “SAP NetWeaver BI overview” on page 48 introduces SAP NetWeaver BI and provides an overview of its structure and operations.
- ▶ “Our project environment” on page 53 describes the processes involved with running and monitoring the different project scenarios.
- ▶ “Options and parameters discussions” on page 72 analyzes the online workload, with optimization and tuning options.
- ▶ “Results from an SAP perspective” on page 119 details the results of the stress test from an SAP perspective.
- ▶ “Lessons learned” on page 123 provides a summary of the knowledge gained from this project.

2.1 SAP NetWeaver BI overview

SAP NetWeaver BI is a data warehousing solution, a business intelligence platform, and a suite of business intelligence tools developed by SAP AG. SAP NetWeaver BI provides companies with the ability to report, analyze, and interpret their business data. It allows them to make sound decisions to maintain their competitive edge, improve performance, and build stronger relationships with their customers.

As a data warehouse solution, SAP NetWeaver BI integrates data that is scattered across the organization. Data can be collected from other productive SAP applications, as well as from external non-SAP sources. The information is transformed, consolidated, and prepared for analysis.

As a business intelligence platform, SAP NetWeaver BI provides the infrastructure technologies for analysis, such as an Online Analytical Processing (OLAP) processor, metadata repository, business planning, and simulation. The OLAP processor is used to prepare large amounts of data for multidimensional analysis. With the HTML-based metadata repository, information regarding objects, their properties, and their relationship to other objects can be accessed from a central location. The business planning and simulation applications allow companies to create and develop their own planning applications or use planning applications delivered with SAP NetWeaver BI.

The business intelligence platform also provides special analytical services, such as the Analysis Processor Designer (APD). With the APD, customers can create their own data-mining processes or use SAP data-mining methods to determine patterns, connections, and correlations to generate new information.

As a business intelligence suite, SAP NetWeaver BI provides tools through the Business Explorer (BEx). These tools provide flexible analysis and reporting methods necessary for decision-making support within the business. BEx provides functions for querying, analyzing, and evaluating data through various levels of details and views. Information can be reported through different mediums, such as Web applications, spreadsheets, e-mail, or through the SAP enterprise portal.

2.1.1 The SAP NetWeaver BI information model

The SAP NetWeaver BI informational model is the basic structural element in the SAP NetWeaver BI architecture, which is shown in Figure 2-1.

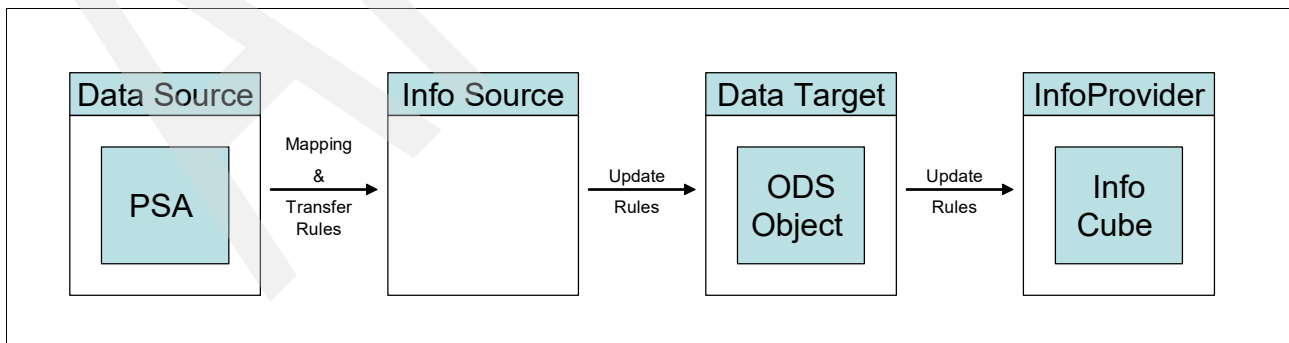


Figure 2-1 Elements in the SAP NetWeaver BI information model

The key elements include the following:

- ▶ *Data sources*: They contain the metadata description of the source system and include data that logically belong together. Data is stored in a flat data structure, and used to extract data from a source system and to transfer data into SAP NetWeaver BI.
- ▶ *Persistent Staging Area (PSA)*: This is the initial storage area for extracted data from the source system. The extracted data is stored in a collection of tables and remains unchanged from the source system (that is, it has not been transformed).
- ▶ *InfoObjects*: These are the lowest-level business evaluation objects on which the SAP NetWeaver BI information model is based. They structure information needed to create data targets, and describe business process and informational requirements. InfoObjects are divided into key figures and characteristics. Key figures hold the numeric information, such as revenue, quantity, and amount. They are the data part of the data target and therefore provide the value reported on in the query. Characteristics are used to describe objects dealt with in the business process, such as customer or product. They specify the classification of the data set and therefore provide the sorting keys of the query. For example, characteristics can specify a unit, such as currency, used in conjunction with key figures for amount and quantity. In the InfoCube (described later this section), characteristics are stored in the dimensions, which are linked to the key figures in the fact table, thus determining the granularity of the key figures.
- ▶ *InfoSource*: These group together InfoObjects that logically belong together from a business point of view. The fields of the data source are mapped and assigned to the corresponding InfoObjects through transfer rules. InfoSources describe the relationship between the extracted data contained in the data source to the InfoObjects. The transfer rules cleanse the data and make it analyzable.
- ▶ *Data Target*: These are objects into which data is loaded from the InfoSource. Data targets include ODS objects and InfoCubes.
- ▶ *Update Rules*: These specify how the data (InfoObject) is updated to the data target.
- ▶ *Operational Data Store (ODS)*: The ODS is a storage location for cleansed, transformed, and consolidated data from one or more InfoSources. Data is stored in transparent, flat database tables. ODS objects can be updated into InfoCubes or other ODS objects. These tables can be used in queries for analysis, however, they are typically used to integrate data from different sources and update InfoCubes.
- ▶ *InfoCubes*: InfoCubes are multidimensional data structures consisting of several relational tables. InfoCubes are used to answer complex business questions, as they are powerful analytical tools. Through multidimensional analysis, insight and understanding not possible with two-dimensional analysis can be achieved.
- ▶ *InfoProvider*: These are objects that can be used in reporting. InfoProviders include objects that physically store information, such as InfoCubes and ODS, or objects that are logical views, such as InfoSets, RemoteCubes, and MultiProviders.
- ▶ *InfoSets*: These are a type of InfoProvider. They are a BI-specific view of the data. They describe data sources that are usually joins on ODS objects.
- ▶ *MultiProvider*: A MultiProvider combines data from several InfoProviders. They do not contain actual data like InfoCubes, but rather virtual data.

2.1.2 InfoCubes and the extended star schema

Traditional relational databases are based on tables that provide a two-dimensional analysis on the rows and columns. For example, an analyst would like to view the sales for a certain car per region. This is possible through a two-dimensional model of sales by region. However, more complex analysis is usually required. For example, the analyst would now like

to compare the sales of a certain car with manual or automatic transmission for different regions for the third quarter. The analyst would also like to compare these results with the results over the past several years. In order to do so through traditional databases, the user would have to know the underlying structure of the tables and complete several SQL joins. InfoCubes make multidimensional analysis easily possible.

Each of the three cubes in Figure 2-2 illustrates multidimensional analysis along three dimensions — region, product, and time.

- ▶ The first cube illustrates analysis of region A along the dimensions of product and time.
- ▶ The second cube depicts the analysis of product Y with respect to the region and time dimensions.
- ▶ The third cube demonstrates the Q2 analysis of all the products and regions.

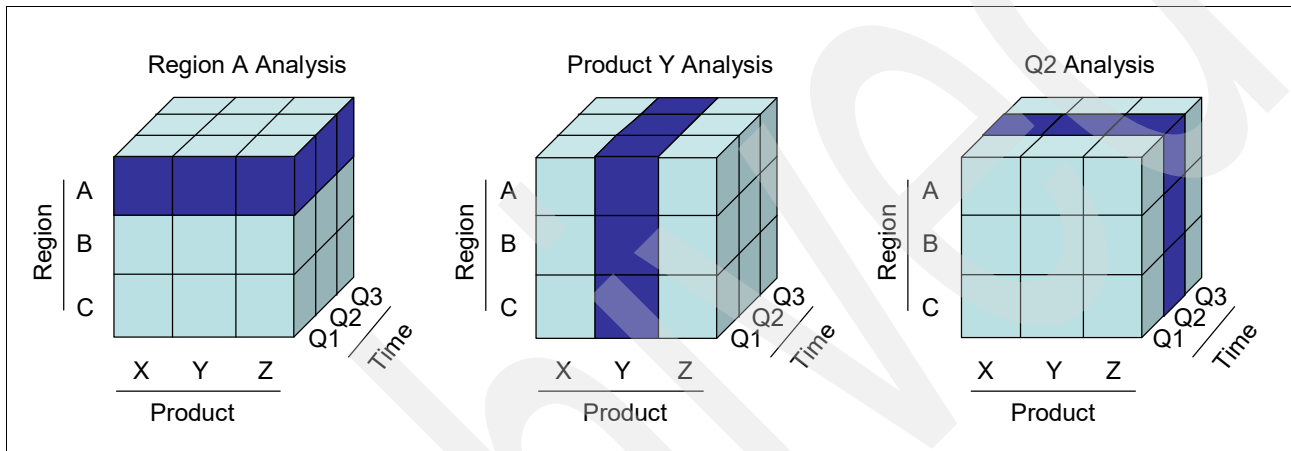


Figure 2-2 Multidimensional analysis

In order to have multidimensional analysis through relational database tables, SAP NetWeaver BI utilizes the *extended star schema*. The extended star schema is a technique that organizes data in terms of data facts and business dimensions. With the extended star schema, there is a central *fact table* that is surrounded by several *dimension tables*. Fact tables contain the key figures of the InfoObject, while the dimension tables store the characteristics needed for evaluating and reporting on the key figures.

As seen in Figure 2-3, the dimension tables (Dx) are independent of each other. The fact table (F in the figure) links the dimensions and key figures through the use of dimension identifiers (DIMID). Every InfoCube has three default dimensions: data package (DP), time (DT), and unit (DU). An InfoCube can have a maximum of 16 dimensions. Master data about attributes, hierarchies, and text are stored on separate tables. The dimension tables link to the master data through surrogate identifiers (SID). The master data can be shared between InfoCubes and thus stored only once, reducing redundancy of data.

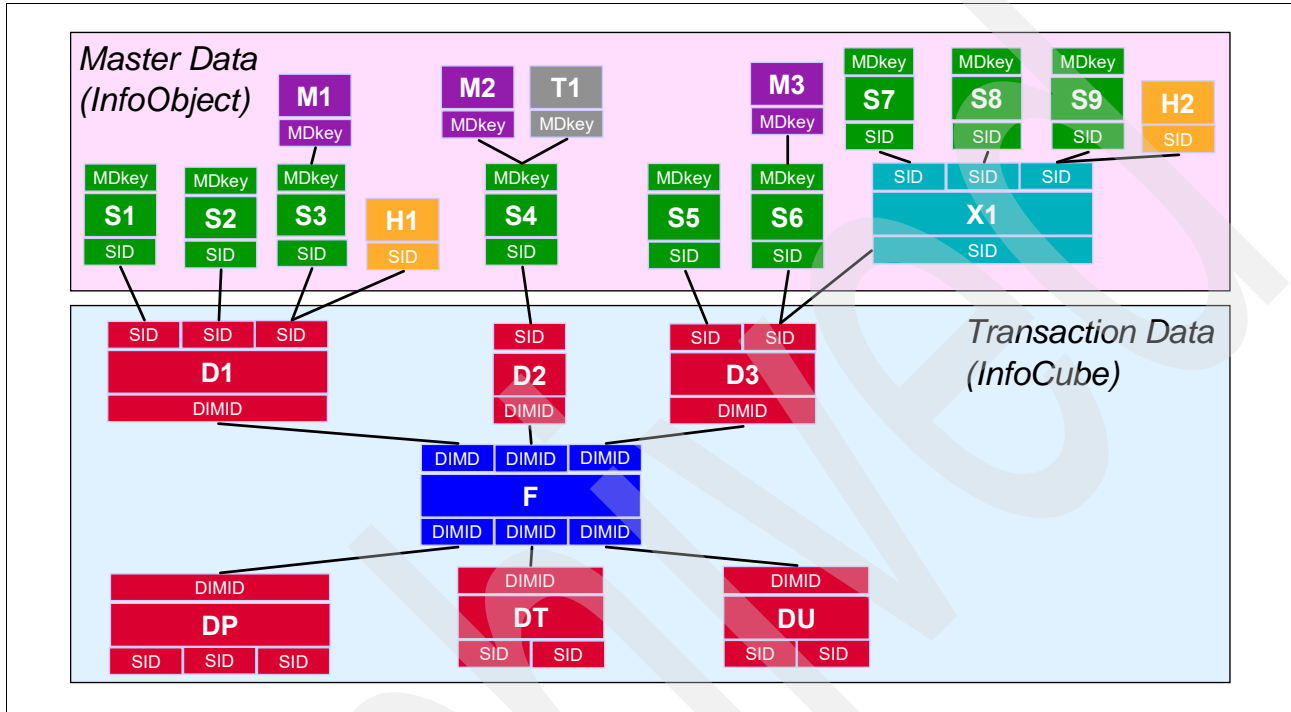


Figure 2-3 SAP NetWeaver BI extended star schema

2.1.3 The data flow in SAP NetWeaver BI

The data flow is shown at a high level in Figure 2-1 on page 48 for a general understanding. Figure 2-4 shows a more complete view of the possible data flow in SAP NetWeaver BI.

Data from the source system is normally loaded into the PSA first. From the PSA, data is then loaded into the ODS objects and then into the InfoCubes. The loading of data is described as the extraction, transformation, and loading process. After data has been loaded, the information is accessed for analysis through queries against the InfoCubes or from the ODS objects.

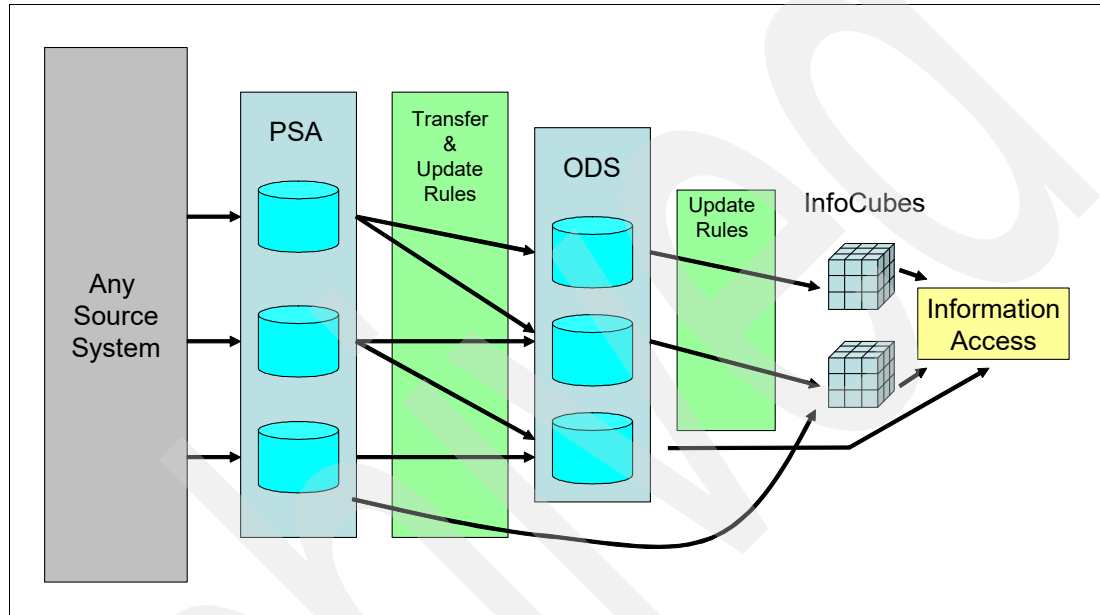


Figure 2-4 The data flow into SAP NetWeaver BI

Extraction, transformation, and loading

Extraction, transformation, and loading (ETL) is the process that involves taking data from a source system, cleansing, preparing, and storing the data.

Extraction involves acquiring the data. Data can be extracted from several heterogeneous systems, such as:

- ▶ SAP systems
- ▶ SAP NetWeaver BI systems
- ▶ Flat files
- ▶ XML files
- ▶ Other SAP supported databases

Since data comes from a variety of sources, it must be transformed. The source data must be cleansed, which means removing unwanted, incorrect, or inconsistent data. The source data may be reformatted to other data types, as well as mapping to InfoObjects. The transformation process is done through transfer and update rules.

Loading involves storing the data. As mentioned before, data is initially stored in the PSA, where it undergoes transformation before it is stored in the ODS or InfoCubes.

SAP NetWeaver BI comes with a suite of ETL tools. ETL tasks can be linked together easily through process chains. These process chains can be scheduled to run as either a time-based or event-based process.

Information access

Once data has been loaded into the various data targets, they can be accessed for analysis. Figure 2-5 shows the flow of information access of SAP NetWeaver BI. Users access the information through the SAP NetWeaver BI comprehensive analysis tool suite Business Explorer. BEx allows users to define queries on the different InfoProviders and InfoObjects.

Analysis on multi-dimensions is facilitated through the OLAP processor, which lies between the user (BEx) and the data warehouse. The OLAP processor is optimized to handle the analysis of very large data sets. It splits the request from BEx into several database queries. The system then looks for the best InfoProvider and generates the underlying SQL statements for the database system. It consolidates the results and sends them back to the BEx tools.

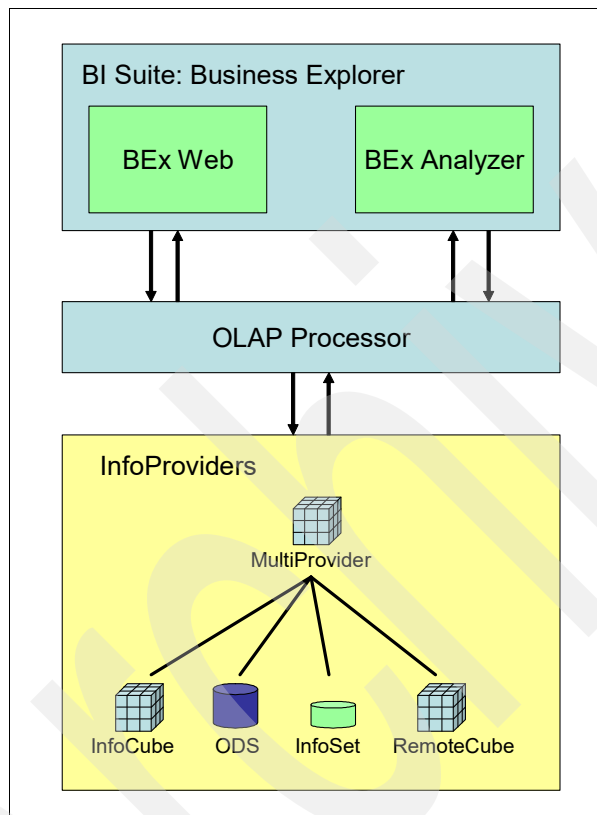


Figure 2-5 Information access

2.2 Our project environment

Three SAP NetWeaver BI processes were chosen for this proof of concept. Each of the processes reflects the tasks performed in a productive environment. The following processes were benchmarked:

- ▶ The *load* process: Data is loaded from an ODS to target InfoCubes and subjected to expensive transformation rules implemented at the start routines. Million of records loaded per hour (Mio rec/hr) was used as the main unit of measurement for throughput.
- ▶ The *aggregate* rollup process: Aggregates are updated with new data from the underlying InfoCubes (aggregates are introduced in 2.2.4, “The aggregate rollup process” on page 62). Million of records per hour (Mio rec/hr) was used as the main unit of measurement for throughput.

- ▶ The *query* process: the SAP NetWeaver BI system is subjected to users running a variety of queries, accessing different InfoProviders. Transactions per second (tx/sec) and response time in seconds were used as the main units of measurement.

The load and rollup processes are representative of actions performed during ETL, while the query process benchmarks the information access.

2.2.1 SAP NetWeaver application servers overview

Several application servers were used to complete the three different SAP NetWeaver BI processes. Figure 2-6 displays an overview of the servers.

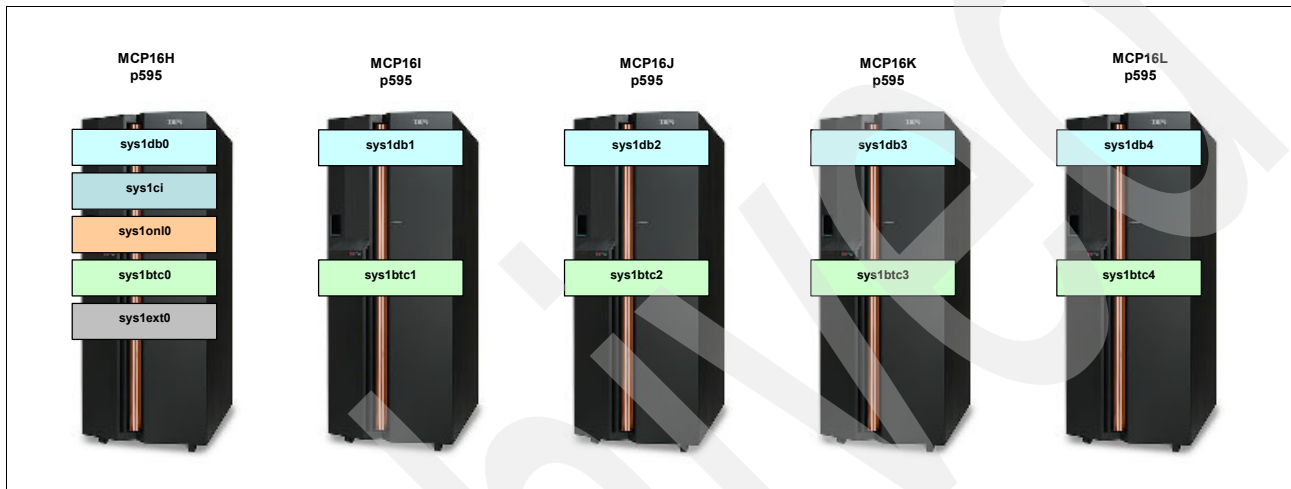


Figure 2-6 SAP Netweaver Application Servers

Multiple servers were defined. The database servers consisted of sys1db0, sys1db1, sys1db2, sys1db3, and sys1db4. The SAP central instance was sys1ci. The load process used servers sys1ext0, sys1btc1, sys1btc2, sys1btc3, and sys1btc4. The rollup process used server sys1btc0 and the query process used sys1onl0. The servers that were used varied depending on the tests.

Each of the application servers was running SAP NetWeaver 04 BI (3.5) on SP 16 with SAP NetWeaver BI 3.5 SP 13. Table 2-1 shows the setup of the application servers in terms of number of dialog and background work processes.

Table 2-1 Application server setup

| Application server | Number of dialog work processes | Number of background work processes |
|--------------------|---------------------------------|-------------------------------------|
| sys1ci | 60 | 25 |
| sys1onl0 | 96 | 25 |
| sys1ext0 | 32 | 32 |
| sys1btc0 | 32 | 18 |
| sys1btc1 | 100 | 18 |
| sys1btc2 | 100 | 18 |
| sys1btc3 | 100 | 18 |

| Application server | Number of dialog work processes | Number of background work processes |
|--------------------|---------------------------------|-------------------------------------|
| sys1btc4 | 100 | 18 |

2.2.2 The SAP data model

The customer provided us with two different InfoCubes and ODS structures:

- ▶ Sales Statistics Mkt B (InfoCube technical name YGTFC301)
- ▶ Profitability Analysis Controlling Mkt B (InfoCube technical name YGTFC101)

These cubes have been copied and used for the load, aggregate, and query/reporting processes.

For maintenance reasons, all objects have been distributed across four logical LPARS or InfoAreas, which are not directly related to LPARS on the hardware or database. Each LPAR contains 35 InfoCubes and two ODS sources. Figure 2-7 displays the distribution across the different InfoAreas/LPARS.

All the InfoCubes contain approximately 200 million rows. Fifty InfoCubes are used for the query scenario and 50 more InfoCubes contribute to the data volume. InfoCubes 100 to 140 (labeled IC101-IC140 in Figure 2-7) are used for the data load scenario and aggregate scenario.

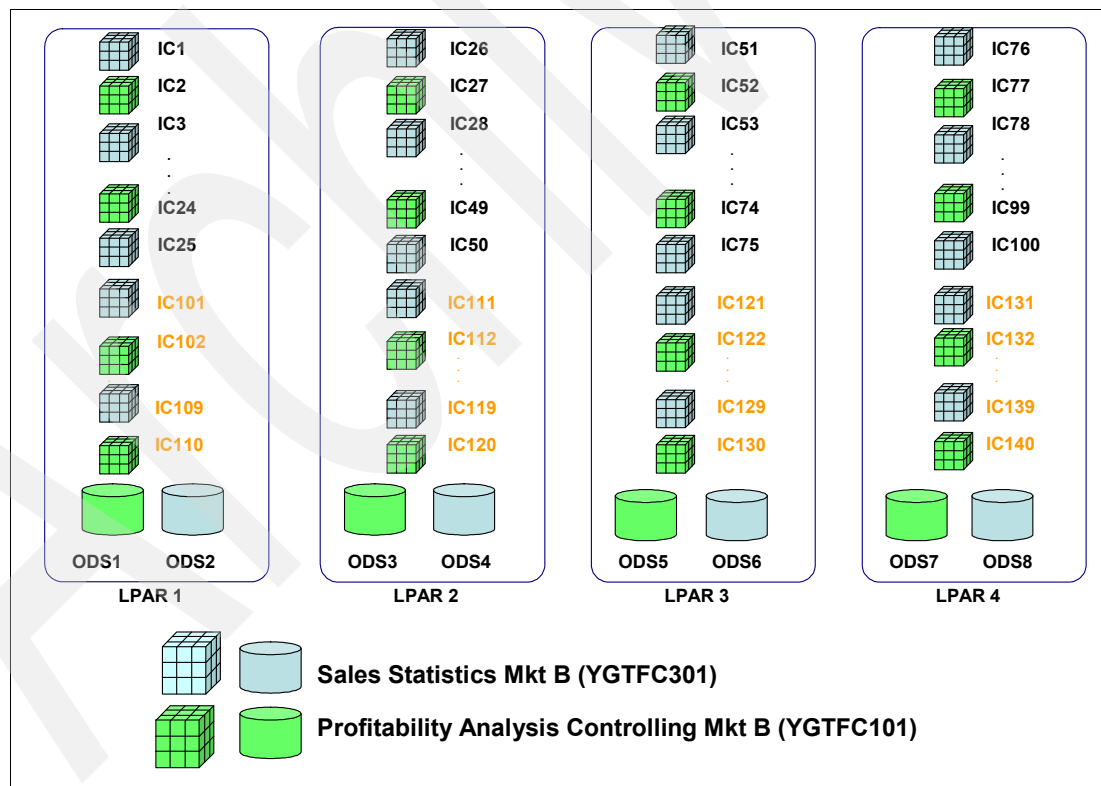


Figure 2-7 InfoCube distribution

2.2.3 The load process

The load process reads data from one or more ODS sources and inserts the data into one or more InfoCubes. The data being transferred is subjected to update rules before being uploaded into the InfoCubes, as shown in Figure 2-8. In this example we show a 1:2 pipe, a single ODS source, and two target InfoCubes.

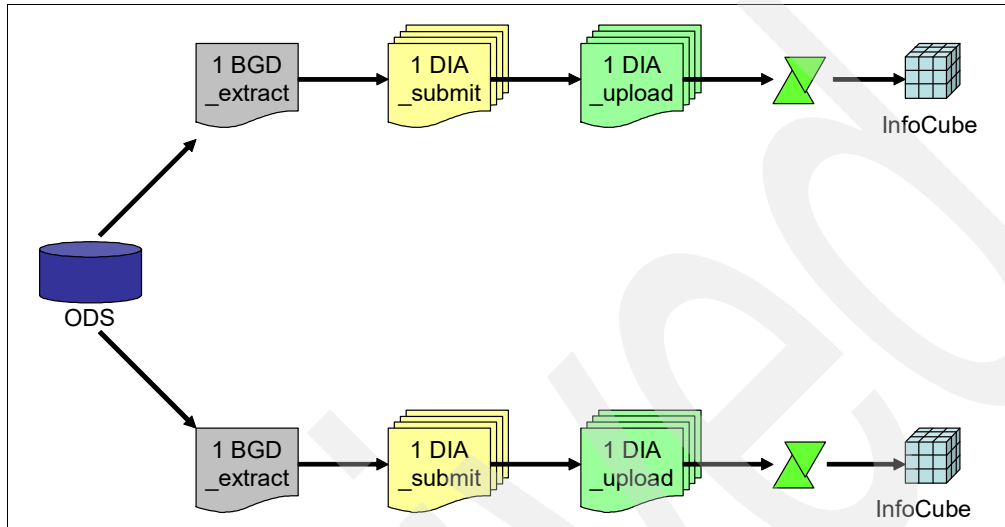


Figure 2-8 The upload process steps

The process flow of the load is defined in a process chain (transaction RSPC). The process chain for the example is shown in Figure 2-9. Once the process chain has been scheduled, a BGD_trigger process is started. This process represents the start block in the process chain and triggers the sub-chains.

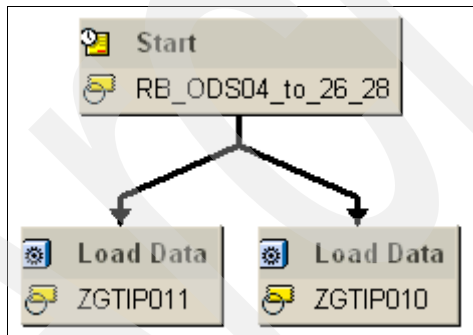


Figure 2-9 Example process chain

This process runs against an instance defined when the process chain is scheduled. In the example, the instance used was sys1ext0. Transaction SM37 provides the information shown in Figure 2-10.

| Job | Ln | Job CreatedBy | Status | Start date | Start time | Duration(sec.) | Delay (sec.) | TargetServr |
|--------------------|----|---------------|----------|------------|------------|----------------|--------------|---------------|
| BI_PROCESS_TRIGGER | | EDGAR | Finished | 12.03.2007 | 18:30:04 | 8 | 4 | sys1ext0p_EB8 |
| *Summary | | | | | | 8 | 4 | |

Figure 2-10 Load process: example step 1

Subsequently, the BGD_trigger process starts two BGD_loading processes, which represent the two sub-chains. Once the two BGD_loading processes have been started, the trigger process changes to the finished state. The BGD_loading processes run against the same instance as the trigger process. Transaction SM37 provides the information shown in Figure 2-11

| Job | Ln | Job CreatedBy | Status | Start date | Start time | Delay (sec.) |
|---|----|---------------|----------|------------|------------|--------------|
| <input type="checkbox"/> BI_PROCESS_LOADING | | EDGAR | Finished | 12.03.2007 | 18:31:10 | 6 |
| <input type="checkbox"/> BI_PROCESS_LOADING | | EDGAR | Finished | 12.03.2007 | 18:31:10 | 6 |
| *Summary | | | | | | 12 |

Figure 2-11 Load process: example step 2

Each BGD_loading process starts a BGD_extract process that runs for the remainder of the load process. Once the BGD_extract processes are started, the BGD_loading process changes its state to finished. The BGD_extract process runs against the server defined in the transaction SBIW settings (ROIDOCPRMS or ROOSPRMS tables). SM37 provides the information shown in Figure 2-12

| Job | Ln | Job CreatedBy | Status | Start date | Start time | Delay (sec.) |
|---|----|---------------|----------|------------|------------|--------------|
| <input type="checkbox"/> BIREQU_453N4LTEPD9L7K6SATF75V7UJ | | RFC-BIWEB8 | Finished | 12.03.2007 | 18:36:14 | 0 |
| <input type="checkbox"/> BIREQU_453P3BZR75LM07MB0LI42U00R | | RFC-BIWEB8 | Finished | 12.03.2007 | 18:33:41 | 0 |
| *Summary | | | | | | 0 |

Figure 2-12 Load process: example step 3

The BGD_extract process's main role is to mine data from the source ODS object and to submit it to a DIA_submit process. The DIA_submit processes have very short run times. The DIA_submit processes run under a single user, RFC_BIWEB8S. This user remains logged on until the end of the upload process.

The DIA_submit process starts another DIA_update process via RFC and transfers the package data. The new DIA_update process runs on one of the instances provided in the logon group defined in transaction SMLG. In our example, the logon group used is EB8PRBW102. DIA_update processes apply transformation rules and update a single data package for all target InfoCubes in sequence. The DIA_update processes run under a single user: RFC_BIWEB8. Transaction SM66 shows these users in action.

Once a request has completed and been uploaded to an InfoCube, the aggregates will be rolled up if automatic rollup has been switched on.

InfoCubes and InfoPackages used for loading

For our test, 30 target InfoCubes were created specifically for the loading process. Each of the InfoCubes had 107 key figures, 9 defined time characteristics, 71 navigation attributes, and 81 characteristics spread over the 16 dimensions (3 default, 13 defined).

The InfoPackages for the load process were created from four ODS sources, two ODS sources from each LPAR/InfoArea.

Application servers

The load process used five application servers:

- ▶ Sys1ext0 was used for the extraction of the data from the ODS, that is, the BGD_trigger, BGD_loading, BGD_extract, and DIA_submit processes.
- ▶ Sys1btc1, sys1btc2, sys1btc3, and sys1btc4 were used for the loading of the data, that is, the DIA_update process.

Load balancing

To distribute the workload across the application servers, a round-robin load balancing technique was used. To enable round-robin load balancing, a logon group RFC_BW was created with transaction SMLG, as seen in Figure 2-13. All the instances that will be used for the loading were added to the group.

| Logon Group | Instance | Status |
|-------------|------------------|--------|
| ODS_ACT | sys1ext0p_EB8_00 | Green |
| RFC-OTHER | sys1btc0p_EB8_00 | Green |
| RFC-OTHER | sys1btc0p_EB8_00 | Green |
| RFC-OTHER | sys1cip_EB8_00 | Green |
| RFC-OTHER | sys1on10p_EB8_00 | Green |
| RFC_BW | sys1btc1p_EB8_01 | Green |
| RFC_BW | sys1btc2p_EB8_02 | Green |
| RFC_BW | sys1btc3p_EB8_03 | Green |
| RFC_BW | sys1btc4p_EB8_04 | Green |
| RFC_QUERY | sys1on10p_EB8_00 | Green |
| RFC_QUERY | sys1on11p_EB8_01 | Green |
| RFC_QUERY | sys1on12p_EB8_02 | Green |
| RFC_QUERY | sys1on13p_EB8_03 | Green |
| RFC_QUERY | sys1on14p_EB8_04 | Green |

Figure 2-13 SMLG: maintain logon groups

Once the instances have been added to the logon group RFC_BW, the logon group has to be enabled for round-robin in table RZLLICLASS through transaction SE16, as seen in Figure 2-14.

| CLASSNUM | CLASSNAME | GROUPTYPE | TIMERERD | LOGRERD | FAVTYPE |
|----------|-----------|-----------|----------|---------|---------|
| 01 | RFC_BW | | 120 | 200 | R |
| 02 | RFC_QUERY | | 120 | 200 | B |
| 03 | RFC-OTHER | | 001 | 001 | B |

Figure 2-14 SE16: table RZLLICLASS

Column FAVTYPE contains the load distribution method.

- ▶ B is for best load distribution method.
- ▶ R is for round-robin.

Monitoring the load process

To monitor the load process, a combination of existing tools and tables were used, as well as new tools.

ST03

To view the load information, run transaction ST03 and switch to expert mode. From the BW System Load tree menu, select **Last Minute's Load**, as seen in Figure 2-15.

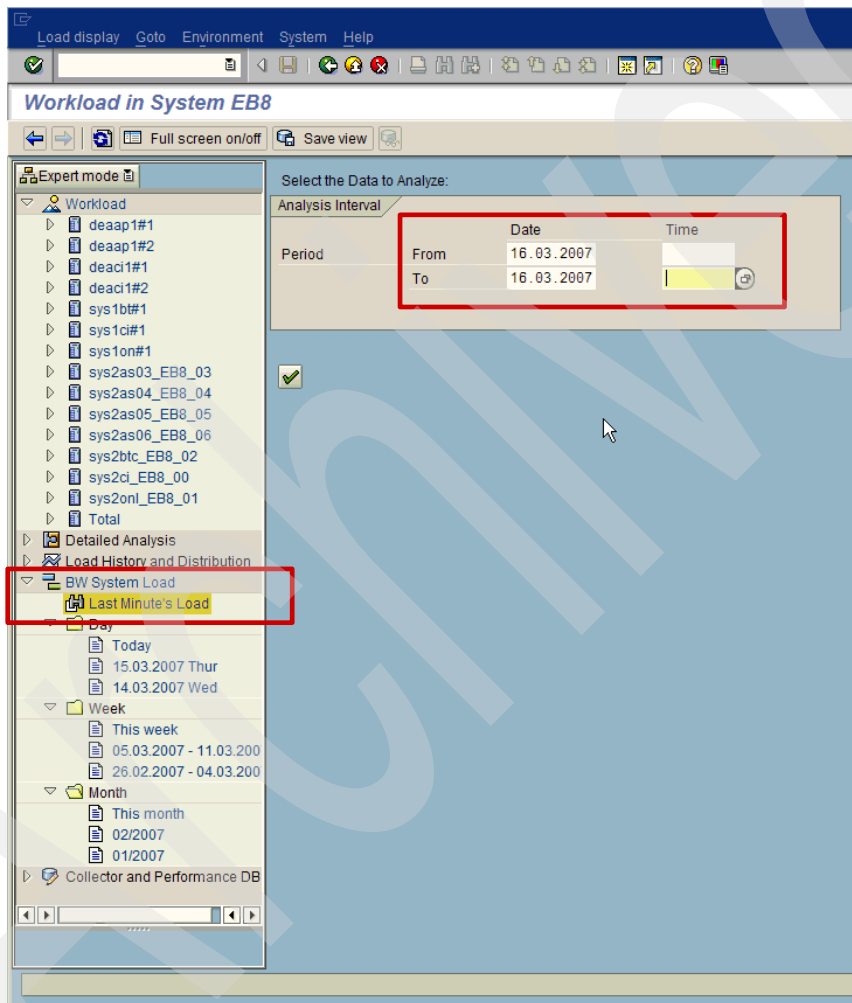


Figure 2-15 ST03: BW system load monitoring

Specify a time frame larger than the actual load job. That means a start time before the load started and an end time after the load stopped. Select **Load Data** from the tree. 'Figure 2-16 provides an example of the resulting information.

From ST03 Load Data, there are different views and aggregation levels available. The data can be sorted to view, for example, the following:

- ▶ Which InfoSources loaded the most data
- ▶ If there are any long-running load processes
- ▶ Which phase of the process took the longest time
- ▶ If parallel loading was used

The screenshot shows the 'Upload - TARGET OBJECTS: Runtimes' view in SAP ST03. The table displays performance metrics for a total load and 15 individual ZGTLOAD processes. The columns include: Target Obj, # InfoSrc, # Requests, Total time, % Source, % Transfer, % PSA, % Update, % TargObj, # Written, t(Source), t(Transf.), t(PSA), t(Update), and t(Target).

| Target Obj | # InfoSrc | # Requests | Total time | % Source | % Transfer | % PSA | % Update | % TargObj | # Written | t(Source) | t(Transf.) | t(PSA) | t(Update) | t(Target) |
|------------|-----------|------------|------------|----------|------------|-------|----------|-----------|---------------|-----------|------------|--------|--------------|-------------|
| TOTAL | 4 | 40 | 248.083,1 | 32,80 | 39,90 | 0,00 | 0,00 | 0,00 | 1.241.484.840 | 81.361,2 | 98.974,6 | 0,0 | 25.948.082,5 | 3.036.439,3 |
| ZGTLOAD19 | 1 | 10 | 68.468,7 | 30,57 | 35,54 | 0,00 | 364,99 | 203,25 | 54.783.580 | 20.932,7 | 24.331,4 | 0,0 | 249.905,5 | 139.161,5 |
| ZGTLOAD20 | 1 | 10 | 68.468,7 | 30,57 | 35,54 | 0,00 | 925,11 | 199,32 | 54.783.580 | 20.932,7 | 24.331,4 | 0,0 | 633.408,1 | 136.474,4 |
| ZGTLOAD21 | 1 | 10 | 68.468,7 | 30,57 | 35,54 | 0,00 | 0,00 | 0,00 | 54.783.580 | 20.932,7 | 24.331,4 | 0,0 | 1.014.036,6 | 143.964,6 |
| ZGTLOAD22 | 1 | 10 | 68.468,7 | 30,57 | 35,54 | 0,00 | 0,00 | 0,00 | 54.783.580 | 20.932,7 | 24.331,4 | 0,0 | 1.402.646,2 | 139.834,5 |
| ZGTLOAD23 | 1 | 10 | 68.468,7 | 30,57 | 35,54 | 0,00 | 0,00 | 0,00 | 54.783.580 | 20.932,7 | 24.331,4 | 0,0 | 1.786.399,6 | 138.326,0 |
| ZGTLOAD24 | 1 | 10 | 68.468,7 | 30,57 | 35,54 | 0,00 | 0,00 | 0,00 | 54.783.580 | 20.932,7 | 24.331,4 | 0,0 | 2.168.840,9 | 137.163,0 |
| ZGTLOAD06 | 1 | 10 | 67.711,9 | 29,57 | 35,56 | 0,00 | 0,00 | 0,00 | 54.783.580 | 20.022,6 | 24.081,5 | 0,0 | 2.158.443,5 | 136.848,2 |
| ZGTLOAD05 | 1 | 10 | 67.711,9 | 29,57 | 35,56 | 0,00 | 0,00 | 0,00 | 54.783.580 | 20.022,6 | 24.081,5 | 0,0 | 1.772.852,3 | 141.493,9 |
| ZGTLOAD04 | 1 | 10 | 67.711,9 | 29,57 | 35,56 | 0,00 | 0,00 | 0,00 | 54.783.580 | 20.022,6 | 24.081,5 | 0,0 | 1.392.181,1 | 136.439,6 |
| ZGTLOAD03 | 1 | 10 | 67.711,9 | 29,57 | 35,56 | 0,00 | 0,00 | 0,00 | 54.783.580 | 20.022,6 | 24.081,5 | 0,0 | 1.012.373,9 | 135.712,2 |
| ZGTLOAD02 | 1 | 10 | 67.711,9 | 29,57 | 35,56 | 0,00 | 932,47 | 202,96 | 54.783.580 | 20.022,6 | 24.081,5 | 0,0 | 631.394,1 | 137.425,8 |
| ZGTLOAD01 | 1 | 10 | 67.711,9 | 29,57 | 35,56 | 0,00 | 367,72 | 205,02 | 54.783.580 | 20.022,6 | 24.081,5 | 0,0 | 248.991,3 | 138.824,6 |
| ZGTLOAD28 | 1 | 10 | 56.071,1 | 36,95 | 45,50 | 0,00 | 349,69 | 200,51 | 48.673.490 | 20.715,7 | 25.509,8 | 0,0 | 196.073,9 | 112.426,4 |
| ZGTLOAD33 | 1 | 10 | 56.071,1 | 36,95 | 45,50 | 0,00 | 0,00 | 0,00 | 48.673.490 | 20.715,7 | 25.509,8 | 0,0 | 1.724.197,3 | 113.778,0 |
| ZGTLOAD32 | 1 | 10 | 56.071,1 | 36,95 | 45,50 | 0,00 | 0,00 | 0,00 | 48.673.490 | 20.715,7 | 25.509,8 | 0,0 | 1.422.663,5 | 110.620,5 |
| ZGTLOAD31 | 1 | 10 | 56.071,1 | 36,95 | 45,50 | 0,00 | 0,00 | 0,00 | 48.673.490 | 20.715,7 | 25.509,8 | 0,0 | 1.116.802,8 | 115.027,6 |
| ZGTLOAD30 | 1 | 10 | 56.071,1 | 36,95 | 45,50 | 0,00 | 0,00 | 0,00 | 48.673.490 | 20.715,7 | 25.509,8 | 0,0 | 808.856,2 | 117.172,4 |
| ZGTLOAD29 | 1 | 10 | 56.071,1 | 36,95 | 45,50 | 0,00 | 889,89 | 213,66 | 48.673.490 | 20.715,7 | 25.509,8 | 0,0 | 498.974,1 | 119.803,9 |
| ZGTLOAD10 | 1 | 10 | 55.831,3 | 35,27 | 44,87 | 0,00 | 349,11 | 196,81 | 48.673.490 | 19.690,1 | 25.051,9 | 0,0 | 194.911,8 | 109.881,7 |
| ZGTLOAD11 | 1 | 10 | 55.831,3 | 35,27 | 44,87 | 0,00 | 886,95 | 199,15 | 48.673.490 | 19.690,1 | 25.051,9 | 0,0 | 495.197,6 | 111.186,7 |
| ZGTLOAD12 | 1 | 10 | 55.831,3 | 35,27 | 44,87 | 0,00 | 0,00 | 0,00 | 48.673.490 | 19.690,1 | 25.051,9 | 0,0 | 796.050,7 | 110.625,5 |
| ZGTLOAD13 | 1 | 10 | 55.831,3 | 35,27 | 44,87 | 0,00 | 0,00 | 0,00 | 48.673.490 | 19.690,1 | 25.051,9 | 0,0 | 1.097.161,7 | 121.093,2 |
| ZGTLOAD14 | 1 | 10 | 55.831,3 | 35,27 | 44,87 | 0,00 | 0,00 | 0,00 | 48.673.490 | 19.690,1 | 25.051,9 | 0,0 | 1.408.374,7 | 118.479,6 |
| ZGTLOAD15 | 1 | 10 | 55.831,3 | 35,27 | 44,87 | 0,00 | 0,00 | 0,00 | 48.673.490 | 19.690,1 | 25.051,9 | 0,0 | 1.717.345,0 | 114.675,5 |

Figure 2-16 ST03: load data

RSDDSTATWHM

Information for ST03 comes from the database table RSDDSTATWHM. To view this data, run transaction SE16. In the selection window, fill in the InfoCube name. For example, from the information returned the following can be determined:

- ▶ The number of records uploaded
- ▶ The source ODS
- ▶ The start time of the load process
- ▶ The overall duration

RSMONMESS

Table RSMONMESS contains information for each request identification and how many records were uploaded at a given time stamp. A program (actually named Z60TB_UPLOAD_REALTIME_MONITOR2, called Zmonitor in the following) was written to analyze the data from this table. The Zmonitor program will take start date, start time, end date, end time, and interval period as parameters. It will then read the RSMONMESS table to

determine how many records were uploaded during that time and display the information per interval. As an example, see Figure 2-17.

Z60TB_UPLOAD_REALTIME_MONITOR2 1

UPLOAD REALTIME MONITORING & ANALYSIS

Start date / time: 16.03.2007 / 11:00:00
 End date / time: 16.03.2007 / 13:40:00
 Time now: 16.03.2007 / 16:30:06

| Timestamp | # steps | # records | Throughput [mio recs/h] |
|---------------------|---------|------------|-------------------------|
| 16.03.2007 11:00:00 | 0 | 0 | 0,00 |
| 16.03.2007 11:05:00 | 0 | 0 | 0,00 |
| 16.03.2007 11:10:00 | 0 | 0 | 0,00 |
| 16.03.2007 11:15:00 | 0 | 0 | 0,00 |
| 16.03.2007 11:20:00 | 0 | 0 | 0,00 |
| 16.03.2007 11:25:00 | 0 | 0 | 0,00 |
| 16.03.2007 11:30:00 | 0 | 0 | 0,00 |
| 16.03.2007 11:35:00 | 70 | 1.731.148 | 20,78 |
| 16.03.2007 11:40:00 | 253 | 6.305.790 | 75,70 |
| 16.03.2007 11:45:00 | 391 | 9.628.954 | 115,59 |
| 16.03.2007 11:50:00 | 454 | 10.582.155 | 127,04 |
| 16.03.2007 11:55:00 | 519 | 11.740.427 | 140,94 |
| 16.03.2007 12:00:00 | 503 | 11.422.612 | 137,13 |
| 16.03.2007 12:05:00 | 478 | 11.019.315 | 132,28 |
| 16.03.2007 12:10:00 | 409 | 9.969.428 | 119,68 |

Figure 2-17 The Zmonitor program to monitor the load process

The Zmonitor program would also calculate the throughput and the number of parallel processes being used at a given time. See Figure 2-18 as an example.

Z60TB_UPLOAD_REALTIME_MONITOR2 1

| TOTAL | 10.128 | 248.296.968 | 93,11 |
|-------|--------|-------------|-------|
|-------|--------|-------------|-------|

First insert: 16.03.2007 / 11:35:03
 Last insert: 16.03.2007 / 13:31:18

RAMP-UP INFORMATION

| Request number | Ramp-Up finish | pkgs proc |
|--------------------------------|----------------|-----------|
| REQU_4556FP5E55NGYDE34LYFI2FSR | 11:46:19 | 000044 |
| REQU_4556FP5E55NID0FKIKA2GTLKR | 11:49:00 | 000044 |
| REQU_45558VYHMCR9I66E5QDMXKWHN | 11:48:06 | 000044 |
| REQU_4556FK02SY2M8DHZR90MIFZAZ | 11:49:30 | 000044 |
| REQU_4556FP5E55NIWH6INPKX4RXIZ | 11:59:48 | 000044 |
| REQU_4556FP5E55NIWG33N51HWAU0R | 12:01:18 | 000044 |
| REQU_4556F02ZDC6HZS9EVKZ2KVI3 | 11:59:18 | 000036 |
| REQU_4556F02ZDC6ER0V519PYBBZAZ | 12:02:49 | 000034 |

Figure 2-18 The Zmonitor program to monitor the load process

Statistics for ST03, RSDDSTATWHM and RSMONMESS are only collected if the statistic collection has been activated manually for the InfoProviders. To do so, run transaction RSA1, select **Tools** from the menu bar, followed by **BW Statistics for InfoProvider**. From the resulting window, select the InfoProviders for which statistics should be collected.

High-load phase

For the stress test, only a certain period was measured for throughput and not end-to-end. There are several key phases of the load process: the extract, the ramp-up, the loading, and the ramp-down phases. The throughput was measured during the *loading* phase only.

- ▶ The extract phase is when data is read from the ODS. During this phase, there is activity on the database servers as the BGD_extract process is reading data from the ODS tables. However, on the application servers, there is no activity.
- ▶ The ramp-up phase is when the DIA_submit processes are handing data packages to the DIA_update processes. During this phase, the number of DIA_update processes increases steadily until it reaches the MAXPROCS setting.
- ▶ The third phase is where the actual loading occurs. During this phase, the transformation rules are applied to the records in the data packages. The processing of the transformation rules is CPU intensive. The transformed data is then submitted to the database.
- ▶ The last phase is the ramp-down phase. The number of DIA_update processes steadily decreases as the last of the data packages are processed and added to the database. There is a steady decrease of CPU during this phase.

Therefore, there is no throughput during the extract phase. The throughput steadily increases during the ramp-up phase, steady during the high-load phase, and decreases during the ramp-down phase. We decided to only measure the throughput during the third phase, the high-load phase, as it is the most constant and CPU-intensive phase.

2.2.4 The aggregate rollup process

Aggregates are extremely useful for reporting. The use of aggregates decreases the query response time and reduces the overall system load. For example, if a report is run at the end of each week to determine the number of units sold for a particular week, then there are two main approaches to this:

- ▶ All relevant data for that week is read by a query from an InfoCube and distilled to show the relevant data.
- ▶ An aggregate is called upon to very quickly and efficiently provide the same data.

A well-defined aggregate makes a query much more efficient by basing it on *aggregated information* as opposed to *data fetched* from InfoCubes. However, after loading new data into InfoCubes, the new data must be aggregated, causing the aggregates to be updated. Traditionally, aggregates are rolled up as part of a nighttime batch process.

Figure 2-19 shows an example of the process flow of the aggregate load.

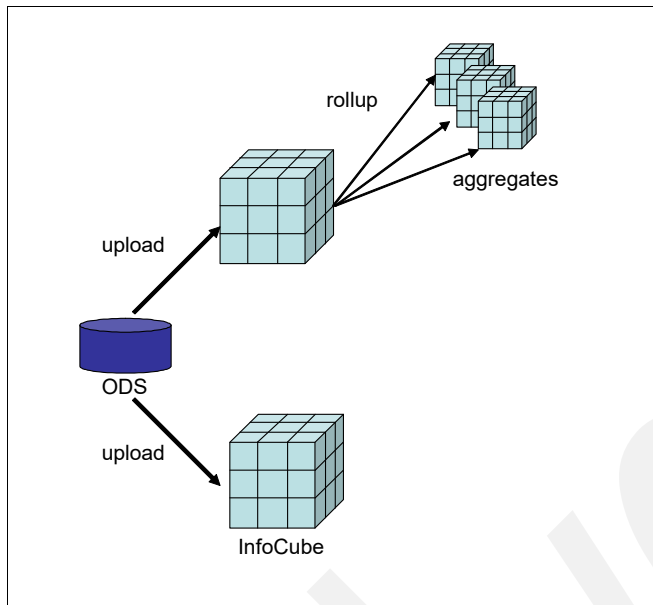


Figure 2-19 The aggregation process steps

Like the load process, the process flow of the aggregation load is defined in a process chain (transaction RSPC). Each InfoCube would only require a single BGD_roll process, as the aggregate fill was done sequentially.

InfoCubes and aggregates

For the test, two sets of InfoCubes were used for the aggregate rollup. One set consisted of 16 InfoCubes that each had 10 aggregates. The second set had 10 InfoCubes with 21 aggregates. Two sets were used for different tests. Instead of rolling up existing data with new data, the aggregates were rebuilt.

After every test, the data in the aggregates was dropped and the next test would refill it. We used two different sets to control the duration of the benchmark tests, as the second set takes longer to roll up more records.

Aggregate tree

Some of the aggregates that were created are dependent on other aggregates, creating a hierarchy or aggregate tree. Since there are dependencies, some aggregates cannot be rolled up before the dependent aggregate, thus the rollup job was sequential.

For example, Figure 2-20 shows an aggregate tree for an InfoCube with 10 aggregates. In this case, aggregate 103512 could not be rolled up before 103512.

| Aggregate Tree | Ac... | Proposed ... | Valuation | Records | Record... |
|------------------------------------|--------------------------|--------------|------------|---------|-----------|
| InfoCube | | | | | |
| 103508 : GC: Orga., Chan., Cust. g | <input type="checkbox"/> | | +... 80409 | 68 | |
| 103512 : GC: Orga., Channel. | <input type="checkbox"/> | | ----- 0 | 0 | |
| 103509 : GC: Orga., Chan., Prod. g | <input type="checkbox"/> | | +... 85477 | 64 | |
| 103511 : GC: Orga., Chan., Produc | <input type="checkbox"/> | | ----- 0 | 0 | |
| 103507 : GC: 0035 CT B0 all | <input type="checkbox"/> | | ----- 0 | 0 | |
| 103516 : GC: RIG CTB0 | <input type="checkbox"/> | | ----- 0 | 0 | |
| 103504 : GC: 0035 CT 10 all | <input type="checkbox"/> | | ----- 0 | 0 | |
| 103506 : GC: 0035 CT 10 RU3 | <input type="checkbox"/> | | ----- 0 | 0 | |
| 103514 : GC: RIG CT10 | <input type="checkbox"/> | | ----- 0 | 0 | |
| 103505 : GC: 0035 CT 10 RU1 | <input type="checkbox"/> | | ----- 0 | 0 | |

Figure 2-20 Aggregate tree with 10 aggregates

Likewise, Figure 2-21 shows an aggregate tree for an InfoCube with 21 aggregates. In this case, aggregate 103066 has to be rolled up before aggregate 103084.

For a single InfoCube, the aggregate rollup was sequential. Unfortunately, the parallelization of the rollup for a single infocube is not available with SAP NetWeaver BI 3.5.

| Aggregate Tree | Ac... | Proposed ... | Valuation | Records | Records S... | Usa... |
|----------------|--------------------------|--------------|-----------|---------|--------------|--------|
| InfoCube | | | | | | |
| 103066 : 10 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103084 : 8 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103067 : 11 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103069 : 13 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103070 : 14 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103065 : 1 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103072 : 16 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103073 : 17 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103074 : 18 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103077 : 20 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103082 : 6 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103078 : 21 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103079 : 3 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103080 : 4 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103081 : 5 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103071 : 15 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103083 : 7 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103085 : 9 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103068 : 12 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |
| 103076 : 2 | <input type="checkbox"/> | | ----- 0 | 0 | 0 | 0 |

Figure 2-21 Aggregate tree with 21 aggregates

Application servers

Only one application server, sys1btc0, was used to perform the aggregate fill. Since the job was sequential, only one BGD process is needed for each InfoCube. Thus, using more than one application server was not necessary.

Monitoring the aggregate rollup process

ST03 and SM37 were used to monitor the aggregate process.

ST03

To view the aggregate information, run transaction ST03 and switch to expert mode. Similar to monitoring the load process, expand the **BW System Load** menu tree and select **Last Minute's Load**. Select a time frame that encompasses the aggregate load, which is a start time before the job started and an end time after the job has ended. See Figure 2-15 on page 59 as an example.

Select **Aggregates** from the Analysis View menu tree. Figure 2-22 provides an example of the aggregated results at an InfoCube level.

| InfoCube | No. Aggr. | Total time | % Rebuild | % Rollup | % Delta | Rebuild | Rebuild | Ø Rebuild | Records(R) | Rollup | Ø Rollup | Records(R) | Delta Time | Delta | Ø Delta | Records(D) |
|-----------|-----------|------------|-----------|----------|---------|-----------|---------|-----------|-------------|--------|----------|------------|------------|-------|---------|------------|
| TOTAL | 210 | 379.255,2 | 100,00 | 0,00 | 0,00 | 379.255,2 | 210 | 1.806,0 | 252.612.400 | 0,0 | 0 | 0,0 | 0 | 0,0 | 0 | 0 |
| ZGTAGGR02 | 21 | 41.806,1 | 100,00 | 0,00 | 0,00 | 41.806,1 | 21 | 1.990,8 | 25.261.240 | 0,0 | 0 | 0,0 | 0 | 0,0 | 0 | 0 |
| ZGTAGGR07 | 21 | 38.519,8 | 100,00 | 0,00 | 0,00 | 38.519,8 | 21 | 1.834,3 | 25.261.240 | 0,0 | 0 | 0,0 | 0 | 0,0 | 0 | 0 |
| ZGTAGGR03 | 21 | 38.124,5 | 100,00 | 0,00 | 0,00 | 38.124,5 | 21 | 1.815,5 | 25.261.240 | 0,0 | 0 | 0,0 | 0 | 0,0 | 0 | 0 |
| ZGTAGGR10 | 21 | 37.691,6 | 100,00 | 0,00 | 0,00 | 37.691,6 | 21 | 1.794,8 | 25.261.240 | 0,0 | 0 | 0,0 | 0 | 0,0 | 0 | 0 |
| ZGTAGGR09 | 21 | 37.690,3 | 100,00 | 0,00 | 0,00 | 37.690,3 | 21 | 1.794,8 | 25.261.240 | 0,0 | 0 | 0,0 | 0 | 0,0 | 0 | 0 |
| ZGTAGGR08 | 21 | 37.578,0 | 100,00 | 0,00 | 0,00 | 37.578,0 | 21 | 1.789,4 | 25.261.240 | 0,0 | 0 | 0,0 | 0 | 0,0 | 0 | 0 |
| ZGTAGGR06 | 21 | 37.533,9 | 100,00 | 0,00 | 0,00 | 37.533,9 | 21 | 1.787,3 | 25.261.240 | 0,0 | 0 | 0,0 | 0 | 0,0 | 0 | 0 |
| ZGTAGGR05 | 21 | 37.161,0 | 100,00 | 0,00 | 0,00 | 37.161,0 | 21 | 1.769,6 | 25.261.240 | 0,0 | 0 | 0,0 | 0 | 0,0 | 0 | 0 |
| ZGTAGGR01 | 21 | 36.871,5 | 100,00 | 0,00 | 0,00 | 36.871,5 | 21 | 1.755,8 | 25.261.240 | 0,0 | 0 | 0,0 | 0 | 0,0 | 0 | 0 |
| ZGTAGGR04 | 21 | 36.278,5 | 100,00 | 0,00 | 0,00 | 36.278,5 | 21 | 1.727,5 | 25.261.240 | 0,0 | 0 | 0,0 | 0 | 0,0 | 0 | 0 |

Figure 2-22 ST03: aggregate analysis view

From this data we can determine which InfoCubes have been rolled-up, the time elapsed, and the number of records involved. In order to get a more detailed view, and preferably at the aggregate level, we can change the level of aggregation. As a result, each InfoCube is displayed along with the aggregates.

Similar to monitoring the load, statistics for ST03 are only collected if statistic collection has been activated manually for the InfoProviders. To do so, run transaction RSA1, select **Tools** from the menu bar, followed by **BW Statistics for InfoProvider**. From the resulting window, select the InfoProviders for which statistics should be collected.

SM37

The job logs of each of the aggregate fills can be viewed through transaction SM37. The job log contains the information regarding each aggregate, the sequential sequence in which they were rolled up, and the duration.

2.2.5 The query process

Once data has been loaded into the data warehouse, it needs to be accessed. Analyzing the data from InfoProviders is the main focus of SAP NetWeaver BI. The query process used for the test simulates user reporting.

Injectors were used to execute queries against SAP NetWeaver BI to simulate a query load. These injectors have preconfigured templates of queries that were suitable for the test. The

queries were selected from a load that currently runs against the customer’s productive systems. A high-level logical overview of the infrastructure is shown in Figure 2-23

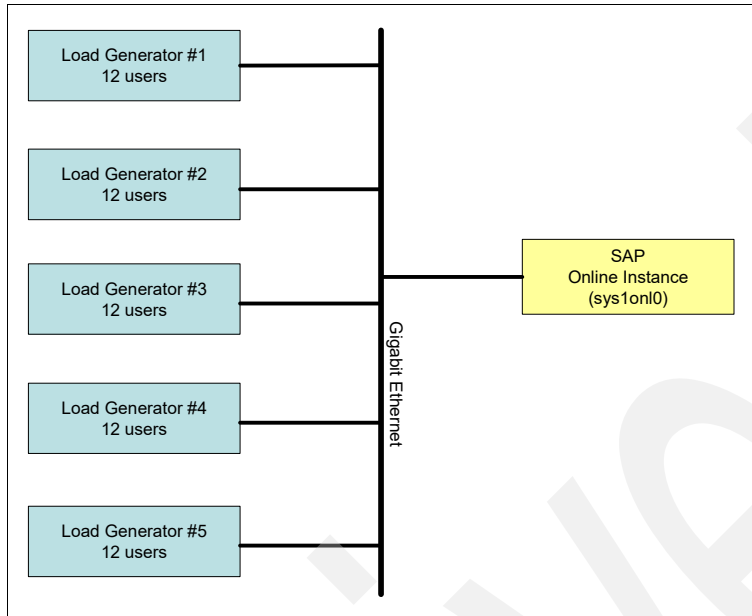


Figure 2-23 Infrastructure for query injection

Five injectors were used. Each injector simulate 12 users, each executing the 10 queries in a loop. One user executes the 10 queries sequentially over and over again. The queries were run with different parameters each time, thus varying the selection of data.

The following physical hardware configuration was used:

- ▶ Injector controllers: IBM Thinkcentre workstation, Pentium® IV 3.4 GHz, RAM 2 GB HD 70 GB
- ▶ Injectors: 5 x IBM xSeries® x330, dual Pentium III 1GHz, RAM 1 GB, HD 20 GB

The following software was configured. All operating systems of load generators and controllers run the latest service packs and patches (available at test time):

- ▶ For the five generators: Windows 2000 Advanced server + SP4 + latest patches.
- ▶ For the controller: Windows XP + SP2 + latest patches.
- ▶ The same tool is used for both the generators and the controller.

Only one application server, sys1onl0, was used for the query process.

Overview of the queries

There were 10 queries defined to access two different multi-providers. Each injector ran the queries against two different multi-providers. Table 2-2 provides an overview of the 10 queries.

Table 2-2 Query overview

| Query number | Query | Multi-provider | Cache mode | Aggregate or fact table |
|--------------|-----------|----------------|------------|-------------------------|
| 1 | Z60_CCS_A | ZGTFCMP01 | 1 | AGGREGATE |

| Query number | Query | Multi-provider | Cache mode | Aggregate or fact table |
|--------------|-----------|----------------|------------|-------------------------|
| 2 | Z60_PLR_A | ZGTFCMP01 | NO | AGGREGATE |
| 3 | Z60_YTD_A | ZGTFCMP02 | 1 | AGGREGATE |
| 4 | Z60_STR_C | ZGTFCMP02 | NO | AGGREGATE |
| 5 | Z60_RIG_A | ZGTFCMP01 | 1 | AGGREGATE |
| 6 | Z60_YTD_B | ZGTFCMP02 | NO | AGGREGATE |
| 7 | Z60_STR_A | ZGTFCMP02 | 1 | AGGREGATE |
| 8 | Z60_STR_B | ZGTFCMP02 | NO | FACT TABLE |
| 9 | Z60_CCS_B | ZGTFCMP01 | 4 | AGGREGATE |
| 10 | Z60_STR_D | ZGTFCMP02 | NO | FACT TABLE |

The cache mode determines whether and in what ways the query results and navigational states calculated by the OLAP processor as highly compressed data are to be saved in a cache. You can set the respective mode in customizing as the standard value for an InfoProvider and in the query monitor for a query. Caching is a means of improving query performance. The different modes are discussed in “OLAP cache” on page 70.

MultiProviders

Although Table 2-2 on page 66 only shows two MultiProviders (ZGTFCMP01 and ZGTFCMP02), there were in fact three additional sets of MultiProviders. MultiProviders sets ZGTFCMP03 and ZGTFCMP04, ZGTFCMP05 and ZGTFCMP06, and ZGTFCMP07 and ZGTFCMP08 were also available.

The queries defined for those MultiProviders are the same as in Table 2-2 on page 66, except that they used different input data. When viewing the information within SAP, queries that had “_1” appended to the query name ran against MultiProviders ZGTFCMP03 and ZGTFCMP04. Queries that had a “_2” appended to the query name ran against MultiProviders ZGTFCMP05 and ZGTFCMP06. Likewise for “_3”, queries ran against ZGTFCMP07 and ZGTFCMP08. See Figure 2-24 on page 68 for an example.

Reporting - InfoCubes: 0 Times / Navigation Step (s)

Queries for InfoCube ZGTFCMP03

| InfoCube | Name of Query | Cache | No. of Nav. | Total time | Ø Total | MED: Total | OLAP Time | Ø OLAP | DB Time | Ø DB | Frontend | Ø Fronten | Selected | Select. / Transf. | Cells | Cells / Record |
|-----------|---------------|-------|-------------|------------|---------|------------|-----------|--------|----------|------|----------|-----------|-------------|-------------------|-------|----------------|
| ZGTFCMP03 | TOTAL | ALL | 9.594 | 72.684,8 | 7,6 | 5,8 | 10.188,0 | 1,1 | 18.344,9 | 1,9 | 41.727,0 | 4,3 | 108.661.000 | 1.261,0 | 0 | 0 |
| ZGTFCMP03 | Z60_PLR_A_1 | ALL | 2.403 | 30.551,2 | 12,7 | 12,4 | 4.650,5 | 1,9 | 16.249,1 | 6,8 | 9.219,4 | 3,8 | 84.495.927 | 1.300,4 | 0 | 0 |
| ZGTFCMP03 | Z60_CCS_A_1 | ALL | 2.403 | 17.312,5 | 7,2 | 6,8 | 2.029,1 | 0,8 | 1.985,8 | 0,8 | 12.389,9 | 5,2 | 23.776.860 | 8.989,4 | 0 | 0 |
| ZGTFCMP03 | Z60_CCS_B_1 | ALL | 2.393 | 17.015,8 | 7,1 | 6,4 | 2.889,8 | 1,2 | 80,7 | 0,0 | 13.334,6 | 5,6 | 388.213 | 20,9 | 0 | 0 |
| ZGTFCMP03 | Z60_RIG_A_1 | ALL | 2.395 | 7.805,3 | 3,3 | 3,0 | 618,6 | 0,3 | 29,2 | 0,0 | 6.783,1 | 2,8 | 0 | 669,1 | 0 | 0 |

Figure 2-24 Query name and MultiProvider example

InfoCubes

Each of the MultiProviders has six or seven InfoCubes defined for access. Table 2-3 provides an overview of the MultiProvider, its InfoCubes, and its query name suffix (_1, _2, or _3).

Table 2-3 MultiProvider and InfoCubes

| Query name suffix | MultiProvider | InfoCubes | MultiProvider | InfoCubes |
|-------------------|---------------|--|---------------|--|
| | ZGTFCMP01 | ZGTFC014 ZGTFC016 ZGTFC018 ZGTFC020 ZGTFC022 ZGTFC024 | ZGTFCMP02 | ZGTFC013 ZGTFC015 ZGTFC017 ZGTFC019 ZGTFC021 ZGTFC023 ZGTFC025 |
| _1 | ZGTFCMP03 | ZGTFC039 ZGTFC041 ZGTFC043 ZGTFC045 ZGTFC047 ZGTFC049 | ZGTFCMP04 | ZGTFC040 ZGTFC042 ZGTFC044 ZGTFC046 ZGTFC048 ZGTFC050 |
| _2 | ZGTFCMP05 | ZGTFC064 ZGTFC066 ZGTFC068 ZGTFC070 ZGTFC072 ZGTFC074 | ZGTFCMP06 | ZGTFC063 ZGTFC065 ZGTFC067 ZGTFC069 ZGTFC071 ZGTFC073 ZGTFC075 |
| _3 | ZGTFCMP07 | ZGTFC089 ZGTFC091 ZGTFC093 ZGTFC095 ZGTFC097 ZGTFC099 | ZGTFCMP08 | ZGTFC090 ZGTFC092 ZGTFC094 ZGTFC096 ZGTFC098 ZGTFC100 |

InfoCubes can be assigned to MultiProviders through transaction RSA1. See Figure 2-25 for an example.

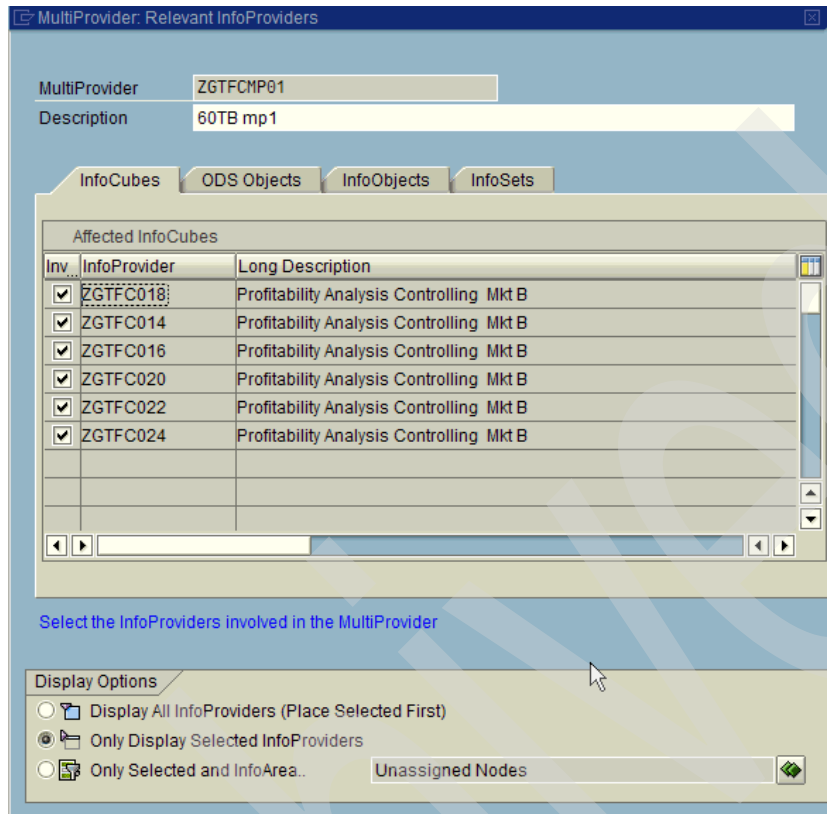


Figure 2-25 MultiProvider and relevant InfoProviders

Records scanned and returned

Each of the queries has a different profile. they each access and return a different number of records. In practice, a majority of the queries scans a relatively small number of records and returns an even smaller number of records. Some *ad hoc* queries, however, have the potential of scanning a large number of records, which can have an impact on system performance and overall response time. To simulate query weight, the following definitions apply:

- ▶ Eighty percent of the queries scan between 1,000–9,000 records and return less than 1,000 records. Queries 1, 2, 3, 4, 5, 6, 7, and 8 are affected.
- ▶ Twenty percent of the queries scan between 20,000–900,000 records and return between 1,000–10,000 records. Queries 9 and 10 are affected.

As mentioned before, each time the injector runs the query, it uses different selection parameters, thus varying the workload.

Level of aggregation

The use of aggregates in queries often helps to improve the response time and reduce the overall system load as less records are being read. However, sometimes queries are not based on aggregated data and are executed against the FactTables of the InfoCubes. To simulate different levels of aggregation, the following definitions apply:

- ▶ Forty percent of the queries use aggregates containing 0–1.2 million records. Queries 1, 2, 3, and 4 are affected.
- ▶ Twenty percent of the queries use aggregates containing 3–6 million records. Queries 5 and 6 are affected.
- ▶ Twenty percent of the queries use aggregates containing 7–9 million records. Queries 7 and 9 are affected.
- ▶ Twenty percent of the queries do not use aggregates and access the fact table. Queries 8 and 10 are affected.

OLAP cache

Using the *OLAP cache* is a method to improve query performance. If the information a query needs is stored in the cache, then the query does not have to read the information from the database, thus saving response time.

There are several different cache modes available:

- ▶ Mode 0: The cache is inactive.
- ▶ Mode 1: main memory cache without swapping
The cache data is stored in the main memory. When the cache memory has been exhausted, a least recently used (LRU) algorithm is invoked to remove excess data. If the removed data is accessed again, the query has to read from the InfoProvider again.
- ▶ Mode 2: main memory cache with swapping.
The cache data is stored in main memory. When the cache memory has been exhausted, the excess data is removed through a LRU algorithm. The excess data is written to a background store, such as a file or a cluster-table. When the removed data needs to be accessed again, it is loaded from the background store.
- ▶ Mode 3: cluster/flat file cache per application server.
The cache data is stored persistently as cluster-tables or a file in a directory attainable by the application server. When saving the data in the database, it does put a strain on the system. However, it is still faster to access the cache data through the table than by having to re-read the data from the InfoProvider.
- ▶ Mode 4: cross-application server cluster/flat file cache.
The cache data is stored persistently on a cross-application server cluster table or in a file on a network file system accessible by the application server. There is no displacement of data and no restriction on memory size.

Since caching helps improve query response time, the following restrictions were requested from the customer:

- ▶ Fifty percent of queries hit the OLAP cache (parameter defined).
- ▶ No cache for queries 2, 4, 6, 8, and 10 (OLAP cache is off).
- ▶ OLAP cache mode 1 for queries 1, 3, 5, and 7.
- ▶ OLAP cache mode 4 for query 9.

Monitoring the query process

The main SAP tool used to monitor the query process was through transaction ST03. Like monitoring the load and aggregate rollup, switch to expert mode. From the BW System Load menu tree, select **Last Minute's Load** and enter a suitable time frame. From the Analysis View menu tree, select **Query Runtimes**. Information about a MultiProvider level is shown in Figure 2-26.

| InfoCube | No. Queries | Cache | No. of Nav. | Total time | Ø Tot | MED: Tot | OLAP Time | Ø OLAP | DB Time | Ø ... | Frontend | Ø Fronten | Selected | Select. / Transf. |
|--------------|-------------|-------|-------------|-------------|-------|----------|-----------|--------|-----------|-------|-----------|-----------|---------------|-------------------|
| TOTAL | 40 | ALL | 95.562 | 1.312.509,0 | 13,7 | 8,0 | 59.433,7 | 0,6 | 659.945,1 | 6,9 | 563.678,0 | 5,9 | 4.614.773.317 | 319,4 |
| ZGTFCMP01 | 4 | ALL | 9.568 | 71.592,2 | 7,5 | 5,8 | 10.221,6 | 1,1 | 17.657,6 | 1,8 | 41.295,9 | 4,3 | 70.152.786 | 991,2 |
| ZGTFCMP02 | 6 | ALL | 14.374 | 225.918,4 | 15,7 | 9,3 | 4.759,1 | 0,3 | 117.060,7 | 8,1 | 99.139,9 | 6,9 | 1.014.160.829 | 275,1 |
| ZGTFCMP03 | 4 | ALL | 9.594 | 72.684,8 | 7,6 | 5,8 | 10.188,0 | 1,1 | 18.344,9 | 1,9 | 41.727,0 | 4,3 | 108.661.000 | 1.261,0 |
| ZGTFCMP04 | 6 | ALL | 14.390 | 277.631,1 | 19,3 | 10,6 | 4.813,7 | 0,3 | 168.141,3 | 11,7 | 99.709,5 | 6,9 | 1.381.538.630 | 378,0 |
| ZGTFCMP05 | 4 | ALL | 9.536 | 67.278,2 | 7,1 | 5,7 | 9.968,0 | 1,0 | 13.564,6 | 1,4 | 41.324,3 | 4,3 | 79.859.205 | 987,0 |
| ZGTFCMP06 | 6 | ALL | 14.292 | 237.311,7 | 16,6 | 9,0 | 4.654,8 | 0,3 | 129.474,7 | 9,1 | 98.249,5 | 6,9 | 411.088.182 | 122,8 |
| ZGTFCMP07 | 4 | ALL | 9.517 | 67.925,3 | 7,1 | 5,7 | 10.173,3 | 1,1 | 13.728,3 | 1,4 | 41.622,8 | 4,4 | 79.692.439 | 1.222,2 |
| ZGTFCMP08 | 6 | ALL | 14.291 | 292.167,3 | 20,4 | 10,8 | 4.655,3 | 0,3 | 181.973,1 | 12,7 | 100.609,2 | 7,0 | 1.469.620.246 | 424,8 |

Figure 2-26 ST03 query runtimes: MultiProvider level

From here we can see that the average response time for all the queries during the selected time frame was 13.7 seconds.

It is possible to change the level of detail displayed. For example, we can view each of the queries executed against the different MultiProviders, as shown in Figure 2-27. From here we can see each of the queries and the different durations spent accessing the database or OLAP, the number of records scanned, and the number of records transferred.

| InfoCube | Name of Query | Cache | No. of Nav. | Total time | Ø Total | MED: Total | OLAP Time | Ø OLAP | DB Time | Ø DB | Frontend | Ø Fronten | Selected | Select. / Transf. | Cells | Cells |
|-----------|---------------|-------|-------------|------------|---------|------------|-----------|--------|----------|------|----------|-----------|------------|-------------------|-------|-------|
| ZGTFCMP05 | TOTAL | ALL | 9.536 | 67.278,2 | 7,1 | 5,7 | 9.968,0 | 1,0 | 13.564,6 | 1,4 | 41.324,3 | 4,3 | 79.859.205 | 987,0 | 0 | 0 |
| ZGTFCMP05 | Z60_PLR_A_2 | ALL | 2.387 | 25.181,6 | 10,5 | 10,1 | 4.581,1 | 1,9 | 11.101,1 | 4,7 | 9.061,4 | 3,8 | 44.388.336 | 800,8 | 0 | 0 |
| ZGTFCMP05 | Z60_CCS_A_2 | ALL | 2.389 | 17.276,9 | 7,2 | 6,9 | 2.078,8 | 0,9 | 1.876,4 | 0,8 | 12.419,3 | 5,2 | 14.456.117 | 6.321,0 | 0 | 0 |
| ZGTFCMP05 | Z60_CCS_B_2 | ALL | 2.380 | 17.256,9 | 7,3 | 6,3 | 2.704,9 | 1,1 | 548,5 | 0,2 | 13.287,7 | 5,6 | 21.014.752 | 905,9 | 0 | 0 |
| ZGTFCMP05 | Z60_RIG_A_2 | ALL | 2.380 | 7.562,8 | 3,2 | 2,9 | 603,3 | 0,3 | 38,7 | 0,0 | 6.555,9 | 2,8 | 0 | 9.002,7 | 0 | 0 |

Figure 2-27 ST03 query runtimes: query level

To determine the throughput (navigations/second) for the selected time frame, from the Analysis View menu tree, click **Time Profiles**. From this view, we export the information to a spreadsheet, as seen in Figure 2-28. The average throughput is calculated by taking the average of the column Steps/sec. This metric was used for measuring the performance of the query process.

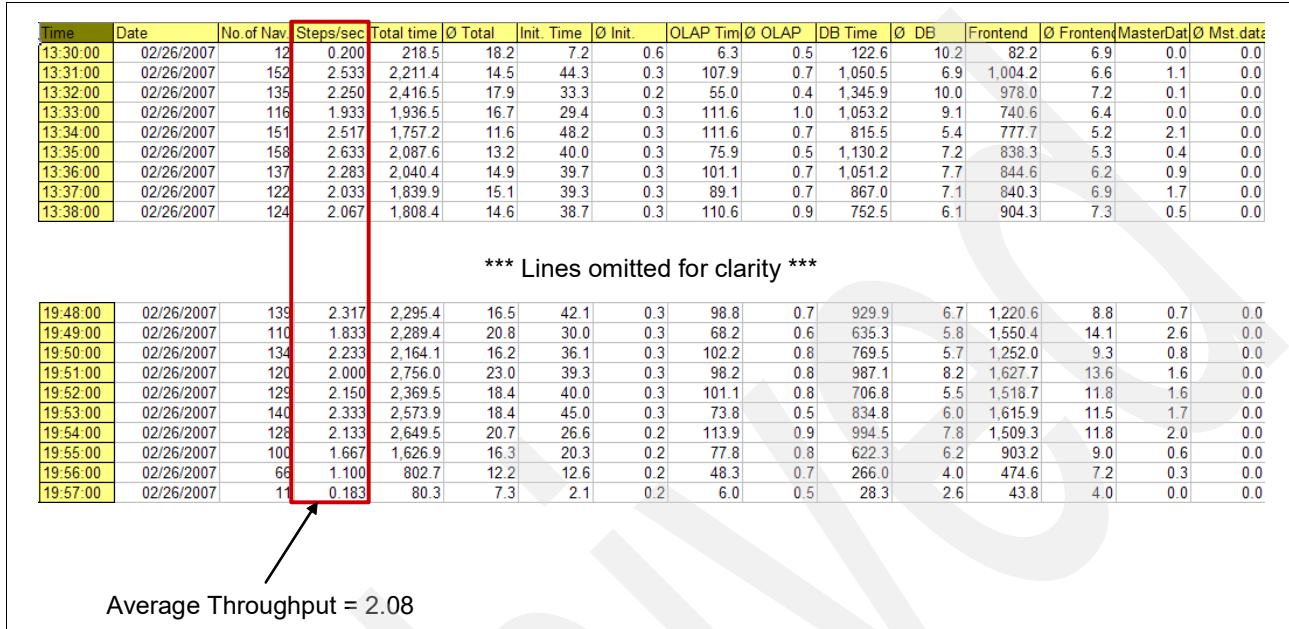


Figure 2-28 ST03 time profile: calculating throughput

2.3 Options and parameters discussions

The parameters described in the subsequent sections have an impact on the performance of the different processes of the tests in our environment.

2.3.1 Parameters affecting the load process

Two types of parameters affect the load process: parameters related to SAP and parameters related to the instance.

SAP parameters

The following parameters affect the performance of the load process:

- ▶ Maximum size of the data package (MAXSIZE)

When data is transferred in SAP NetWeaver BI, the individual data records are transferred through data packages. This parameter controls the size of the data package. The default setting for this parameter is 10,000 KB. In other words, data will be transferred to the target through data packages of 10,000 KB in size. Changing this parameter also affects the memory footprint of the application server, as a larger package consumes more memory.

- ▶ Maximum number of rows in the data package

MAXLINES controls the maximum amount of records contained within each data package. With a large data package size, this parameter affects the memory footprint of the

application server. The more rows contained within the data package, the larger the memory it will consume.

- ▶ Frequency (STATFRQU)

The value of frequency determines how many data IDocs an Info IDoc describes. A default value of 1 means that an IDoc will follow every data IDoc. That is, the lower the frequency value, the more often an IDoc is sent. The IDocs are used for monitoring purposes, as they contain information as to whether the data was successfully uploaded.

- ▶ Maximum number of parallel processes for the data transfer (MAXPROCS)

This parameter determines how many parallel processes are created for each extractor process. This setting is affected by the number of dialog processes available for asynchronous work on the application server. Therefore, the configuration of this parameter depends heavily on the environment and the setup of the application servers.

To maintain the above parameters, run transaction SBIW and select **Maintain Control Parameters for Data Transfer**, as displayed in Figure 2-29.

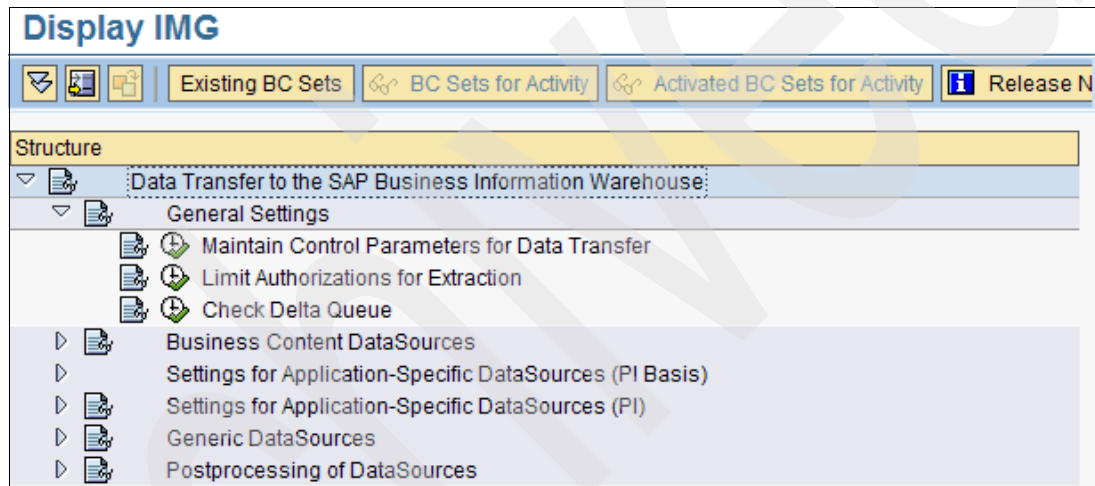


Figure 2-29 SBIW transaction

A window, as shown in Figure 2-30, is then displayed. Locate the source system and change the necessary values.

- ▶ The second column contains the value for the maximum size of the data package.
- ▶ The third column controls the value for the maximum number of rows in the data package.
- ▶ The fourth column changes the setting for the frequency of IDocs.
- ▶ The fifth column determines the maximum number of parallel processes used by each extractor.
- ▶ The last column is used to specify which application server executes the background extractor process.

| Src.system | Max. (...) | Max. lines | Fr. | Max. | Tgt System |
|------------|------------|------------|-----|------|------------|
| A18PPBW102 | 20000 | 100000 | 10 | 6 | |
| A19PPSE102 | 20000 | 60000 | 10 | 3 | |
| AB8PRBW102 | 20000 | 100000 | 10 | 6 | |
| AB9PRSE102 | 20000 | 60000 | 10 | 3 | |
| CB0PRBW102 | 20000 | 60000 | 10 | 3 | |
| E18PPBW102 | 20000 | 100000 | 10 | 6 | |
| E19PPSE102 | 20000 | 60000 | 10 | 3 | |
| EB8PRBW102 | 20000 | 80000 | 10 | 44 | sys1ext0p |
| EB9PRSE102 | 20000 | 60000 | 10 | 3 | |
| FQ8QABW102 | 20000 | 100000 | 10 | 6 | |
| FQ9QASE102 | 20000 | 60000 | 10 | 3 | |
| FR8RTBW102 | 20000 | 100000 | 10 | 6 | |
| FR9RTSE102 | 20000 | 60000 | 10 | 3 | |
| GD8DVBW102 | 20000 | 100000 | 10 | 6 | |
| GD9DVSM102 | 20000 | 60000 | 10 | 3 | |
| GD8DVBW102 | 20000 | 60000 | 10 | 3 | |
| GE8QABW102 | 20000 | 60000 | 10 | 3 | |
| GE8QABW102 | 20000 | 100000 | 10 | 6 | |
| GEBRTBW102 | 20000 | 60000 | 10 | 3 | |
| GELQASM102 | 20000 | 60000 | 10 | 3 | |
| GF8IKBW102 | 20000 | 100000 | 10 | 6 | |

Figure 2-30 SBIW: maintain control parameters for data transfer

Changes made through transaction SBIW affect the entire landscape. However, these settings can be changed through the table ROOSPRMS, which only affects the data source. Any changes made to the table ROOSPRMS override the settings made through the transaction SBIW.

To change the settings for each data source, run transaction SE16 and enter table ROOSPRMS. On the selection window enter the data source. Figure 2-31 shows the settings for data sources named 8ZGTFnnn.

| OLTPSOURCE | RLOGSYS | SRLGSYS | UPDMODE | MAXSIZE | STATFRQU | MAXPROCS | MAXDPAKS |
|------------------------------------|------------|------------|---------|---------|----------|----------|----------|
| <input type="checkbox"/> 8ZGTF0015 | EB8PRBW102 | EB8PRBW102 | F | 080000 | 10 | 99 | |
| <input type="checkbox"/> 8ZGTF0001 | EB8PRBW102 | EB8PRBW102 | F | 080000 | 10 | 64 | |
| <input type="checkbox"/> 8ZGTF0002 | EB8PRBW102 | EB8PRBW102 | F | 080000 | 10 | 44 | |
| <input type="checkbox"/> 8ZGTF0003 | EB8PRBW102 | EB8PRBW102 | F | 080000 | 10 | 44 | |
| <input type="checkbox"/> 8ZGTF0004 | EB8PRBW102 | EB8PRBW102 | F | 060000 | 10 | 64 | |
| <input type="checkbox"/> 8ZGTF0005 | EB8PRBW102 | EB8PRBW102 | F | 080000 | 10 | 64 | |
| <input type="checkbox"/> 8ZGTF0006 | EB8PRBW102 | EB8PRBW102 | F | 080000 | 10 | 44 | |
| <input type="checkbox"/> 8ZGTF0007 | EB8PRBW102 | EB8PRBW102 | F | 080000 | 10 | 44 | |
| <input type="checkbox"/> 8ZGTF0008 | EB8PRBW102 | EB8PRBW102 | F | 080000 | 10 | 64 | |
| <input type="checkbox"/> 8ZGTF0011 | EB8PRBW102 | EB8PRBW102 | F | 065000 | 10 | 20 | |
| <input type="checkbox"/> 8ZGTF0012 | EB8PRBW102 | EB8PRBW102 | F | 080000 | 10 | 99 | |
| <input type="checkbox"/> 8ZGTF0013 | EB8PRBW102 | EB8PRBW102 | F | 065000 | 10 | 20 | |
| <input type="checkbox"/> 8ZGTF0014 | EB8PRBW102 | EB8PRBW102 | F | 080000 | 10 | 99 | |
| <input type="checkbox"/> 8ZGTF0015 | EB8PRBW102 | EB8PRBW102 | F | 065000 | 10 | 20 | |
| <input type="checkbox"/> 8ZGTF0016 | EB8PRBW102 | EB8PRBW102 | F | 065000 | 10 | 20 | |
| <input type="checkbox"/> 8ZGTF0017 | EB8PRBW102 | EB8PRBW102 | F | 065000 | 10 | 20 | |
| <input type="checkbox"/> 8ZGTF0018 | EB8PRBW102 | EB8PRBW102 | F | 065000 | 10 | 20 | |
| <input type="checkbox"/> 8ZGTF00Y2 | EB8PRBW102 | EB8PRBW102 | F | 020000 | 10 | 06 | |

Figure 2-31 SE16: table ROOSPRMS

Select the data source and change the necessary values. The column OLTPSOURCE contains the data source. The column MAXSIZE controls the maximum size of the data package in KB. The column STATFRQU controls the frequency of IDocs. The column MAXPROCS determines the maximum number of parallel processes for the data transfer.

Instance parameters

The following instance parameters affect the performance of the load process:

- ▶ rdisp/wp_no_dia

This parameter controls the number of dialog work processes. It directly affects the maximum number of parallel processes used for data transfer. The loading of data is conducted through an RFC request by the extractor and runs on the different application servers set up for the round-robin workload balancing. If there are too few dialog work processes available on the application server, the requests are put into a queue, thus limiting the throughput of the load process.

- ▶ rdisp/wp_no_btc

This parameter controls the number of work processes used for batch processing. Each extractor uses one background process to read the source ODS. For the loading process, we need at least one background process for each extractor.

- ▶ rdisp/rfc_min_wait_dia_wp

This parameter controls the number of dialog work process that are kept free for other users. This parameter prevents parallel RFCs from using all the available work processes. For example, if the maximum number of parallel processes is 40, the extractor will try to use up all 40 work processes on the remote application server. If this parameter is set to 5, 35 dialog process will be used by the RFC request, 5 will be kept free, and 5 RFC requests will be placed in a queue.

- ▶ `rdisp/rfc_max_login`
This parameter defines the maximum number of logins allowed on the application server. Once this number is exceeded, any further RFC login is rejected.
- ▶ `rdisp/rfc_max_own_login`
This parameter defines the maximum number of logins on the application server. It differs from the `rdisp/rfc_max_login`, as it restricts the logins for one user. This value is a percentage.
- ▶ `rdisp/rfc_max_own_used_up`
This parameter defines the maximum number of dialog work processes and RFCs that a user may use simultaneously. This value is a percentage of the total number of configured dialog work processes.

The above parameters are maintained through transaction RZ10. Any changes to the RFC quotes take effect after an application server restart. The RFC quotas can dynamically be changed through the report RSARFCLD. However, changes made through the report are lost after the application server is restarted.

2.3.2 Optimizing the load process

In order to optimize the load process, it is not only necessary to understand the parameters involved, but also the relationships between objects and processes:

- ▶ ODS to InfoCube
The relationship between ODS to InfoCube is defined through InfoPackages. InfoPackages can have once source and many targets.
- ▶ InfoPackages to extractors
For every InfoPackage a single batch extract process will be started. Therefore, the ratio is always 1:1.
- ▶ CPUs to DIA_update process
Once a DIA_update process has started, it consumes an entire CPU until it has completed the processing of the update rules and submitted the records to the database. As a result, this process is CPU intensive.

Two preliminary tests were conducted to determine the settings for the load process that will be used for KPI-G. The first set of tests involved the parallelization of the number of target InfoCubes, the data package size, and the MAXPROCS setting. The second set of tests involved the change of the number of extractor processes for each of the ODS sources.

Settings the number of InfoCubes

Several tests were executed to determine the optimal setting of the parameters. For the tests, the parameters MAXPROCS and MAXSIZE and the number of target InfoCubes were changed to determine the best throughput.

Table 2-4 has a brief summary of the tests.

Table 2-4 Preliminary load tests

| Test number | # of extractors | # of InfoCubes | MAXSIZE | MAXPROCS | # of DIA WP used | Throughput (rec/hr) |
|-------------|-----------------|----------------|---------|----------|------------------|---------------------|
| 1 | 1 | 1 | 160,000 | 64 | 15 | 4,066,977 |
| 2 | 1 | 2 | 160,000 | 64 | 26 | 7,508,086 |

| Test number | # of extractors | # of InfoCubes | MAXSIZE | MAXPROCS | # of DIA WP used | Throughput (rec/hr) |
|-------------|-----------------|----------------|---------|----------|-------------------------|---------------------|
| 3 | 1 | 4 | 160,000 | 64 | 48 | 13,062,090 |
| 4 | 1 | 5 | 160,000 | 64 | 61 | 15,425,640 |
| 5 | 1 | 7 | 160,000 | 64 | 63 | 17,399,421 |
| 6 | 1 | 4 | 80,000 | 64 | 51 | 14,686,996 |
| 7 | 1 | 7 | 80,000 | 63 | 63 | 18,548,141 |
| 8 | 1 | 7 | 80,000 | 73 | 72 | 19,642,708 |
| 9 | 1 | 7 | 80,000 | 88 | 88 | 21,345,015 |
| 10 | 1 | 4 | 320,000 | 64 | Negative impact on load | |
| 11 | 1 | 7 | 40,000 | 64 | SAP locking problems | |

- ▶ From Table 2-4 on page 76, the first set of tests (tests 1 to 5) involves the change of the number of target InfoCubes for each InfoPackage or ODS source. Test 5, with 7 target InfoCubes and 64 MAXPROCS, yielded the highest throughput at 17.4 Mio rec/hr. Therefore, we conclude that the greater the number of target InfoCubes, the more dialog workplaces in parallel are used, thus giving a greater throughput.

With more target InfoCubes, the extractor only reads once from the database and writes several times to the target InfoCubes. In Figure 2-32 the read once is shown, as some values for the read/hour are zero, but the data is written four times to four separate InfoCubes.

| Target Obj | # InfoSrc. | # Request | Read/hour | Written/h | Read /hour | Written/ h | # Read | # Update | # Written |
|------------|------------|-----------|-----------|-----------|------------|------------|-------------|-------------|-------------|
| TOTAL | 5 | 10 | 0 | 0 | 16.062.036 | 1.678.750 | 102.527.414 | 134.443.379 | 134.443.379 |
| ZGTFC032 | 1 | 1 | 5.714.035 | 3.259.669 | 21.007.648 | 1.818.140 | 22.697.293 | 12.948.060 | 12.948.060 |
| ZGTFC062 | 1 | 1 | 5.807.049 | 3.032.627 | 20.401.231 | 1.998.238 | 22.697.293 | 11.853.255 | 11.853.255 |
| ZGTFC029 | 1 | 2 | 5.144.439 | 2.468.138 | 13.548.156 | 1.864.910 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC033 | 1 | 2 | 0 | 0 | 13.548.156 | 1.520.156 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC035 | 1 | 2 | 0 | 0 | 13.548.156 | 1.479.478 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC037 | 1 | 2 | 0 | 0 | 13.548.156 | 1.869.721 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC052 | 1 | 2 | 5.205.626 | 2.497.494 | 13.613.174 | 1.751.285 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC054 | 1 | 2 | 0 | 0 | 13.613.174 | 1.578.463 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC056 | 1 | 2 | 0 | 0 | 13.613.174 | 1.478.747 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC058 | 1 | 2 | 0 | 0 | 13.613.174 | 1.748.876 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC077 | 1 | 2 | 5.184.062 | 2.487.148 | 13.688.812 | 1.681.121 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC079 | 1 | 2 | 0 | 0 | 13.688.812 | 1.581.518 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC083 | 1 | 2 | 0 | 0 | 13.688.812 | 1.639.471 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC085 | 1 | 2 | 0 | 0 | 13.688.812 | 1.605.623 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC002 | 1 | 2 | 5.240.978 | 2.514.455 | 13.689.854 | 1.654.821 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC004 | 1 | 2 | 0 | 0 | 13.689.854 | 1.553.045 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC008 | 1 | 2 | 0 | 0 | 13.689.854 | 1.634.047 | 14.283.207 | 6.852.629 | 6.852.629 |
| ZGTFC010 | 1 | 2 | 0 | 0 | 13.689.854 | 1.654.338 | 14.283.207 | 6.852.629 | 6.852.629 |

Reads Once (per extractor) Writes Many (per target) Conclusion: reduces the load on the extractor and the database

Figure 2-32 Read once write many

- ▶ From Table 2-4 on page 76, the second set of tests (tests 6 to 9) also involves the change of the number of target InfoCubes per ODS source. However, the maximum size of the data package has been reduced to 80,000 KB. Similar to test 5, seven target InfoCubes yielded a higher throughput. We can also see that the more dialog workplaces are used in parallel, the higher the throughput.

Comparing test 5 and test 7, they both have the same target InfoCubes and MAXPROCS setting, however, test 7 uses a MAXSIZE value of 80,000, while test 5 uses a 160,000 KB data package. Test 7 provided a higher throughput, with approximately 1.1 Mio rec/hr more.

- ▶ Test 10 used a larger maximum data package setting, which resulted in a negative impact on the load, as more memory was used, which caused paging on the system.
- ▶ Test 11 used a smaller package size of 40,000 KB. As a result, less dialog processes were used in parallel, as the process would return to the extractor with a relatively short processing time. During this test, we ran into SAP locking problems with table RSREQDONE as we were running SAP NetWeaver BI support pack 13. This problem is fixed on a later SP level.

During these preliminary tests, we had to be mindful of the RFC quota settings on the application servers. We had to ensure that the number of dialog processes configured on the application servers is greater than or equal to the MAXPROCS setting. If there were too few dialog process available, the RFC requests would start to queue until a dialog process was available. Each of the requests requires a gateway session from the initiating gateway to the target gateway, and each session requires a significant amount of memory on the target application server. As a result, there was memory over-commitment causing paging, an overrun dispatcher queue, and dispatcher errors on the target application servers. These errors on the application server would cause the initiating gateway to crash.

Therefore, the optimal settings determined from the tests are multiple InfoCubes with a maximum data package size of 80,000 KB and a MAXPROCS setting equal to available dialog workprocesses.

Settings the number of extractors

Subsequent tests were done to further optimize the load process by changing the amount of extractor processes per ODS source. Table 2-5 provides a summary of the tests.

Table 2-5 Extractor load test

| Test number | # of ODS | # of Extractors per ODS | # of InfoCubes per ODS | MAXPROCS | Throughput (rec/hr) |
|-------------|----------|-------------------------|------------------------|---------------------|---------------------|
| 1 | 2 | 1 | 4 (total = 8) | 58 (total = 116) | 29,782,400 |
| 2 | 2 | 2 | 4 (total = 8) | 29 (total = 116) | 40,990,400 |
| 3 | 4 | 1 | 4 (total = 16) | 29 (total = 116) | 39,840,000 |

- ▶ Test 1 involved two ODS sources, one extractor for each ODS source with four target InfoCubes. The MAXPROCS settings was 58. Figure 2-33 depicts the process for test 1.

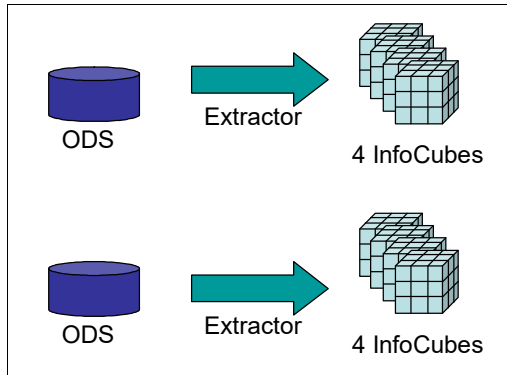


Figure 2-33 Extractor test 1

- ▶ Test 2 involved two ODS sources, with two extractors for each ODS, thus giving a total of four extractors. Unlike test 1, the number of MAXPROCS has been reduced to 29, so the total maximum parallel processes would be the same, that is, 116 dialog work processes. Like test 1, the same amount of target cubes were used. Figure 2-34 shows the configuration.

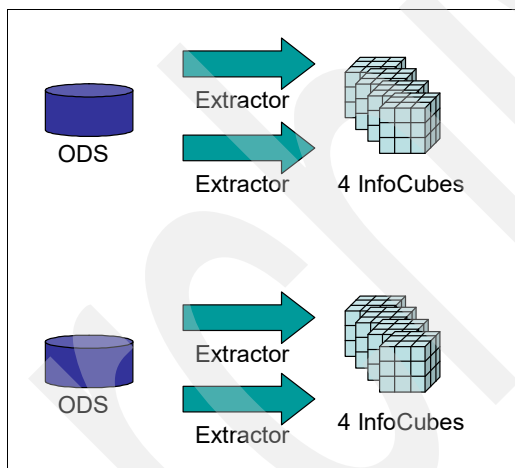


Figure 2-34 Extractor test 2

- ▶ Test 3 involved four ODS sources, each with one extractor, but each extractor had four target cubes giving a total of 16 cubes. Figure 2-35 shows the configuration. Test 3 is very similar to test 1, as each ODS source has only one extractor. Similar to test 2, the maximum number of dialog work processes for each extractor is 29. Therefore, the total maximum number of dialog processes in parallel is 116, like test 1 and test 2.

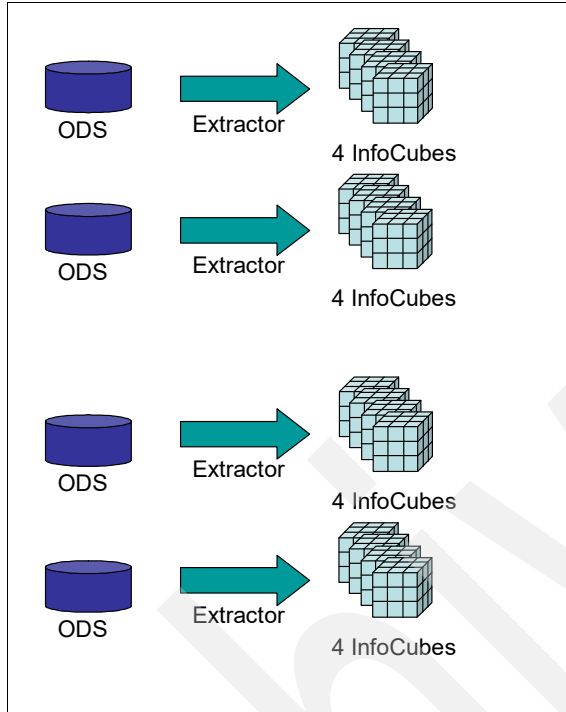


Figure 2-35 Extractor test 3

As we can see in Table 2-5 on page 78, test 2 had the best throughput. Test 2 used the same amount of parallel processes as the other tests. The number of CPUs used was also collected. Table 2-6 shows the CPU usage and calculates the throughput per CPU.

Table 2-6 Extractor test: throughput per CPU

| Test number | Total number of dialog WP | CPU used | Throughput (rec/hr) | Throughput per CPU |
|-------------|---------------------------|----------|---------------------|--------------------|
| 1 | 116 | 65.6 | 29,782,400 | 454,000 |
| 2 | 116 | 74.8 | 40,990,400 | 548,000 |
| 3 | 116 | 96 | 39,840,000 | 415,000 |

We can see that test 3 had a higher throughput than test 1, however, more CPUs were used and the throughput per CPU was less. Test 2 used fewer CPUs than test 3 and yielded a higher throughput, thus giving a better throughput per CPU. Therefore, multiple extractors per ODS gives a better throughput with less cost.

Summary: After running the preliminary load and extractor tests, the results are optimized when:

- ▶ We use multiple extractors per ODS.
- ▶ We use multiple InfoCubes per extractor.
- ▶ MAXSIZE is set up at 80,000 KB.
- ▶ MAXPROCS is set equal to the number of dialog work processes available.

2.3.3 Parameters affecting the aggregate rollup process

Two types of parameters affect the rollup process: parameters related to SAP and parameters related to the instance.

SAP parameters

The block size parameter affects the performance of the aggregate rollup process.

This parameter affects how much data is read from the source during the initial filling of an aggregate. If the amount of data is large, the system reads the data in blocks instead of reading it all at once. This parameter is related to the temporary tablespace size in the database. If the value is too small, there will be more read processes, and this might increase the run time. If the value is too large, the temporary tablespace on the database might overflow.

To change the BLOCK SIZE parameter, run transaction SPRO. Next, expand **SAP Netweaver** from the menu tree, then expand **General BW Settings**, as shown in Figure 2-36.

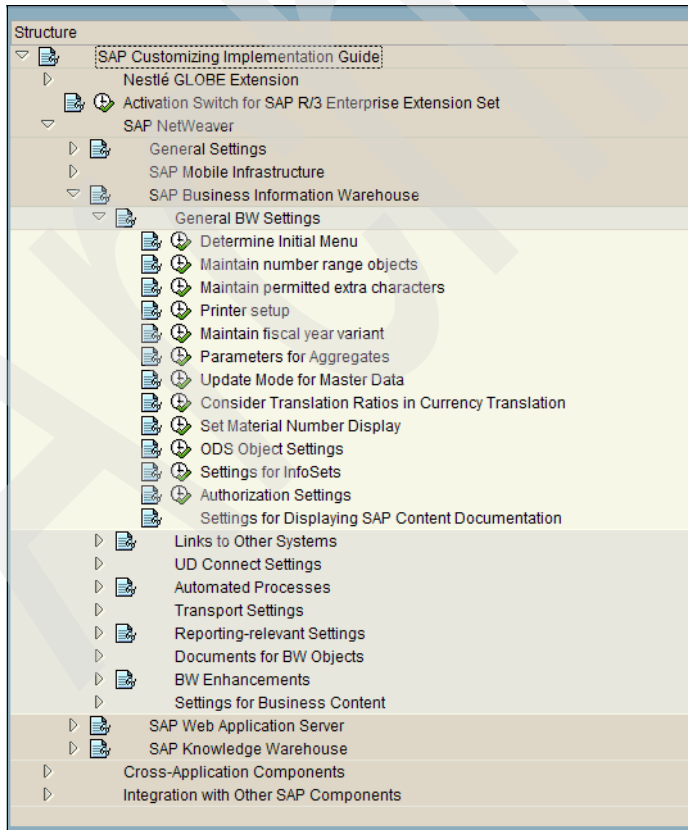


Figure 2-36 SPRO: general SAP settings

Next click **Parameters for Aggregates**, and the window shown in Figure 2-37 is displayed.

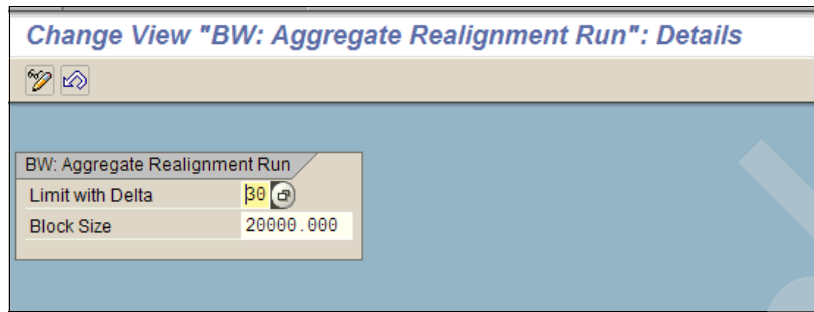


Figure 2-37 SPRO: parameters for aggregates

Instance parameters

The `rdisp/wp_no_btc` parameter affects the aggregate rollup process. This parameter controls the number of work processes used for batch processing. For the initial fill of the aggregates, only one work process is used for each InfoCube. For example, if we have 10 InfoCubes each with 21 aggregates, only 10 work processes will be used to fill the 21 aggregates.

2.3.4 Optimizing the aggregate rollup process

As previously mentioned, the initial fill of the aggregates was processed sequentially. As a result, the CPU speed was directly related to the performance of the rollup process. With SAP NetWeaver BI, parallel processing of the aggregates is not possible. However, with SAP NetWeaver BI 7.0, parallel execution of aggregates is feasible. To overcome this, we ran several initial fillings of aggregates in parallel. Processes and aggregates belonging to different InfoCubes were executed at the same time.

Optimization with SAP NetWeaver BI 7.0

With SAP NetWeaver BI 7.0, transaction RSBATCH (BI Background Management) (shown in Figure 2-38), the following aggregate-related processes can be configured to execute in parallel:

- ▶ AGGRFILL: initial fill of the aggregate.
- ▶ ATTRIBCHAN: attribute change run.
- ▶ CHECKAGGR: Check aggregate during roll up.
- ▶ CONDAGGR: Compress aggregates.
- ▶ ROLLUP: Roll up

The total number of parallel processes can be changed from the current default of 3.

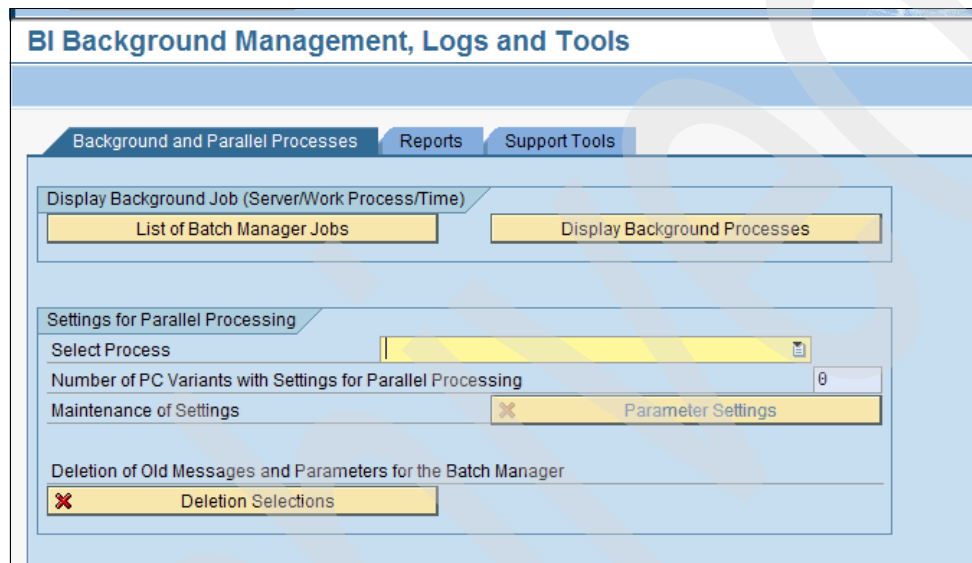


Figure 2-38 RSBATCH transaction

The complete list of processes that can be parallelized is shown in Figure 2-39.

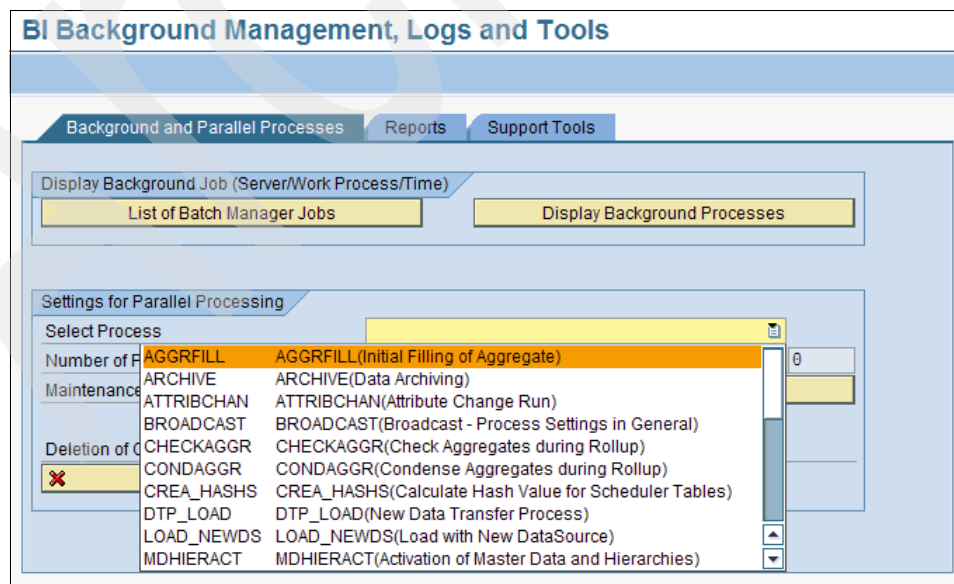


Figure 2-39 RSBATCH: list of processes to parallelize

Optimization with SAP NetWeaver BI 3.5

Since the stress test was executed using SAP NetWeaver BI 3.5, to increase the throughput, several initial fillings of the aggregates were executed in parallel. This means that aggregates belonging to the same InfoCube were filled with one process. In parallel, aggregates belonging to different InfoCubes were also filled.

We conducted a unit test, filling the aggregates of 12 different InfoCubes in parallel. Each of the InfoCubes contained 10 aggregates. Figure 2-40 shows the throughput. The total time to fill 120 aggregates was approximately 1 hour and 29 minutes.

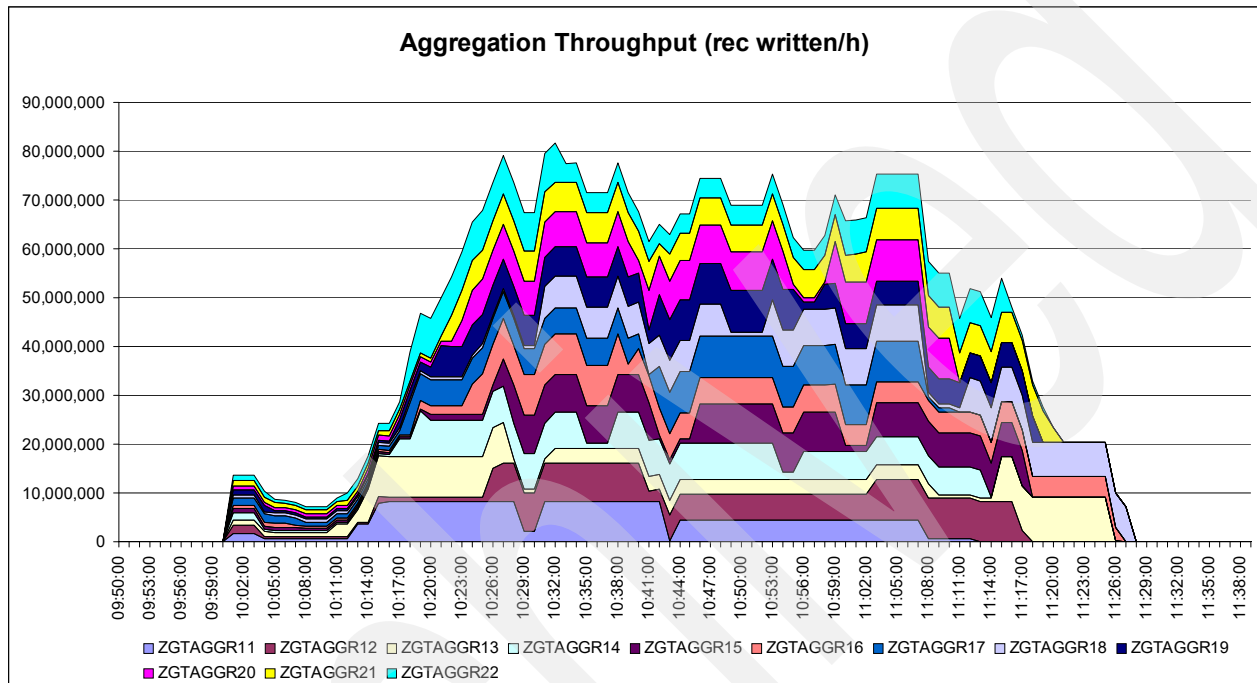


Figure 2-40 Aggregate throughput

On the graph, each colored block represents the aggregates belonging to one InfoCube. The throughput, end-to-end of the run, was approximately 51.3 Mio rec/hr. The highest peak was 81.6 Mio rec/hr. For one InfoCube, the average throughput for the 10 aggregates, sequentially, is approximately 4.3 Mio rec/hr with a standard deviation of 0.29 Mio rec/hr.

Only one application server, sys1btc0, was used for the initial fill. Looking at its CPU usage, the initial fill of the aggregates is not CPU intensive. There was an average CPU usage of 10.2%, as seen in Figure 2-41.

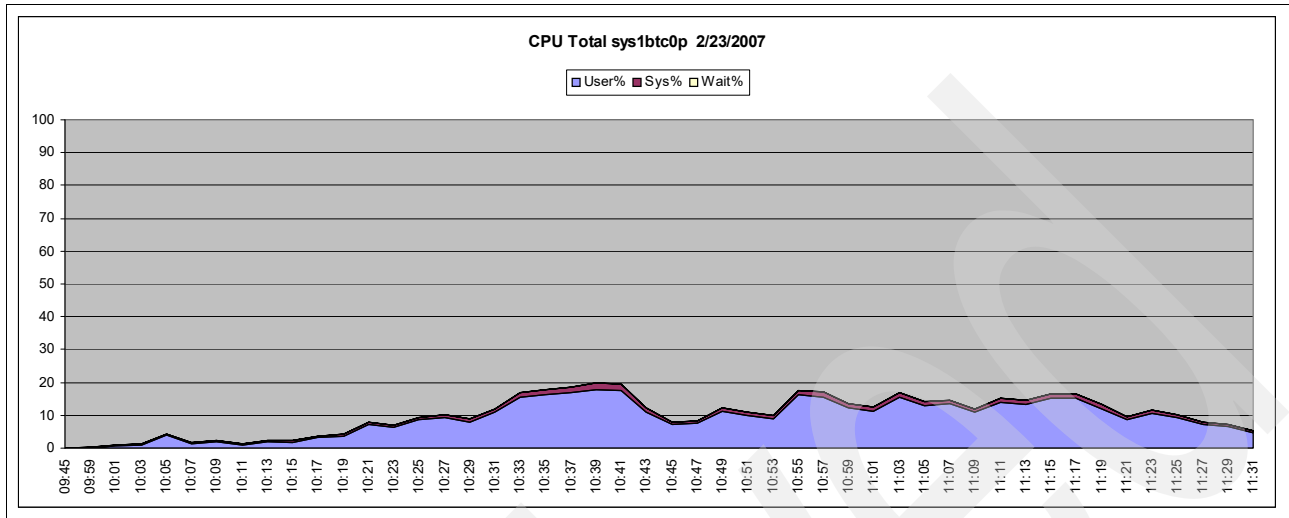


Figure 2-41 CPU usage: sys1btc0

However, on the database LPARs, excluding the catalog node (sys1db0), the average CPU utilization across the four servers was approximately 40%. Figure 2-42 and Figure 2-43 on page 87 show the CPU utilization for the database servers (excluding sys1db0). The CPU usage on the database servers is attributed to the database servers executing the summation. The data is summed on the database before the information is returned to the application server.

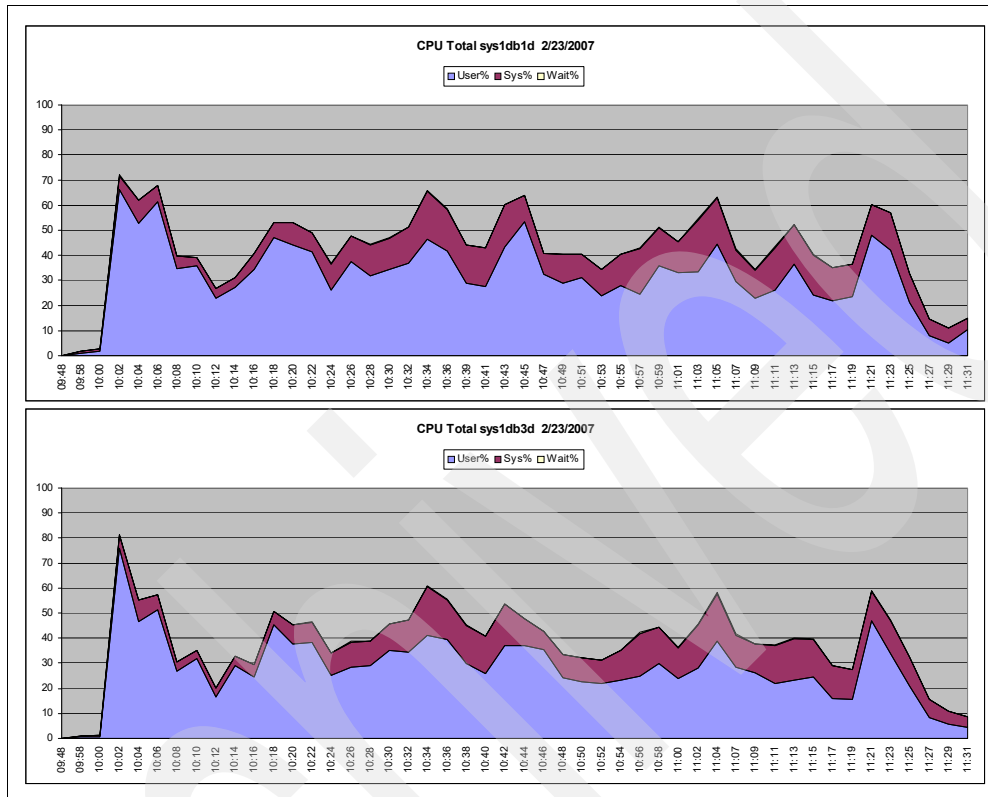


Figure 2-42 CPU usage part 1: sys1db1 and sy1db2

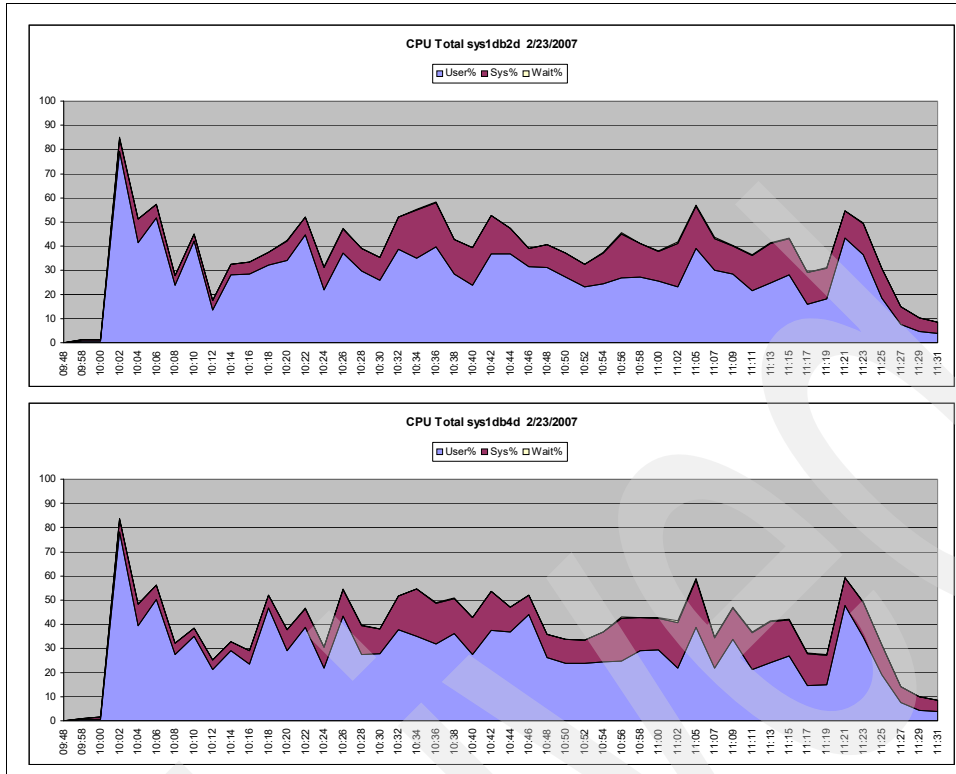


Figure 2-43 CPU usage part 2: sys1db3 and sys1db4

On average, 4.1 CPUs were used by each database LPAR, as seen in Figure 2-44, Figure 2-45 on page 88, and Figure 2-46 on page 88.

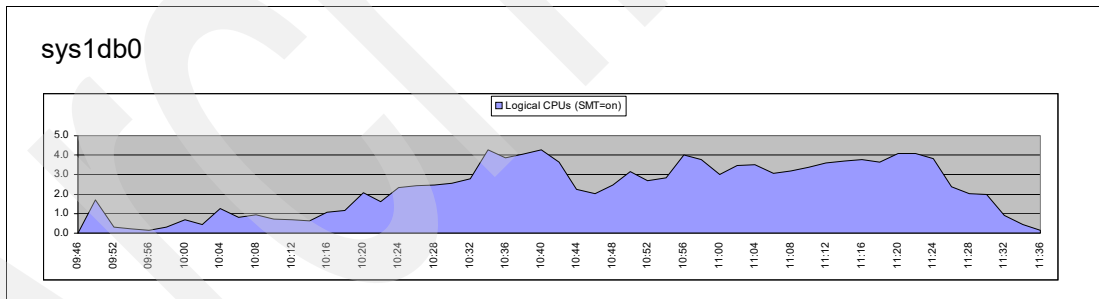


Figure 2-44 Logical CPU usage part 1: sys1db0

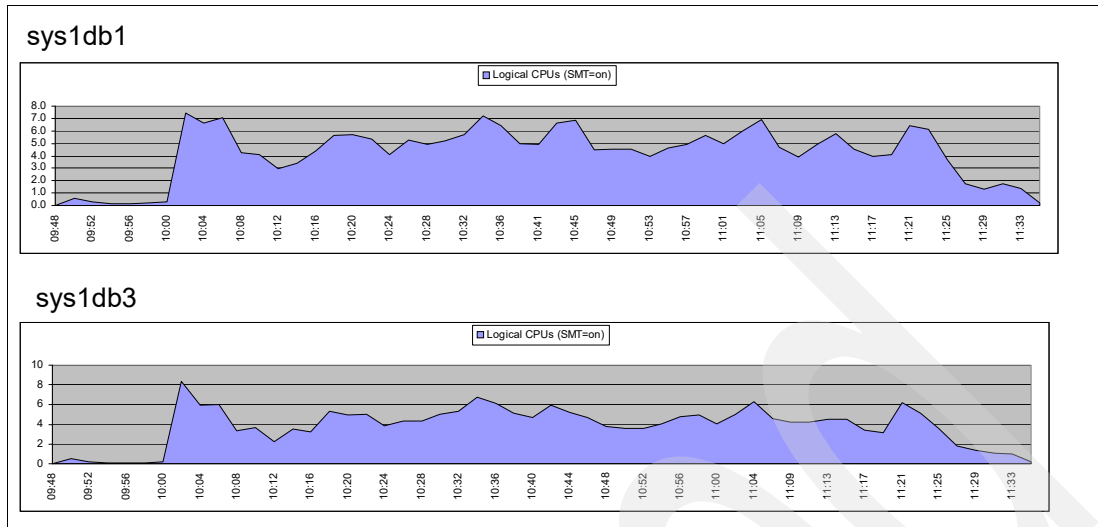


Figure 2-45 Logical CPU usage part 2: sys1db1 and sys1db3

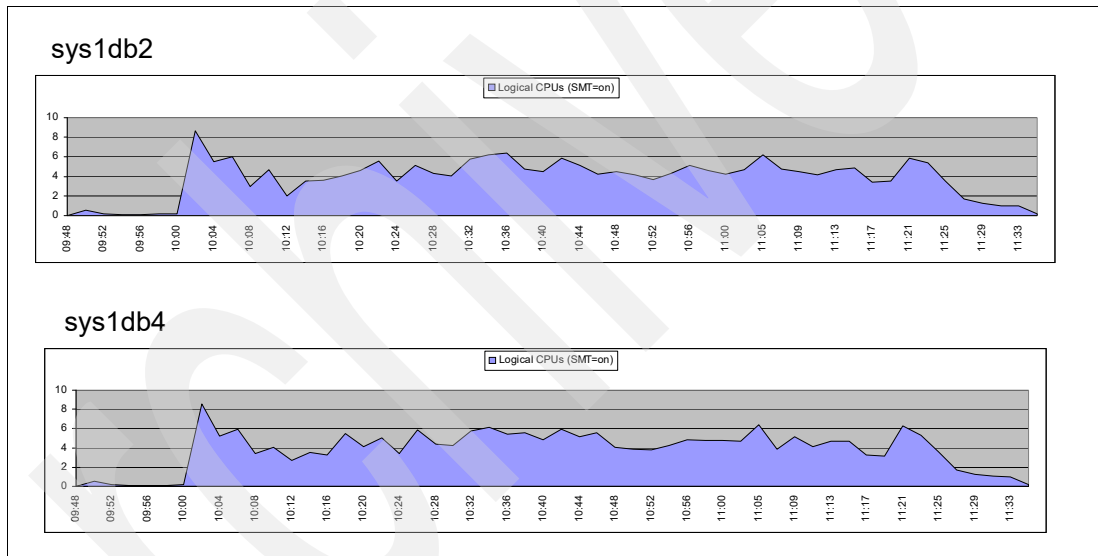


Figure 2-46 Logical CPU usage part 3: sys1db2 and sys1db4

Summary: To optimize the aggregation throughput, the initial fill of different InfoCubes' aggregates were executed in parallel. However, we must be mindful of the database CPU usage, as the summation is executed on the database.

2.3.5 Parameters affecting the query process

Most parameters that affect the query process are database related and require database tuning. For the KPI-G test, there were restrictions defined by the customer to emulate the query workload they experience currently on their system. Several of the restrictions directly affect the performance of the queries, specifically:

- ▶ Fifty percent of the queries can utilize the OLAP cache.
- ▶ Eighty percent of the queries can use aggregates.

These restrictions affect the query process:

- ▶ **OLAP cache:** The OLAP cache is similar to the database bufferpool. The result set of queries and the navigational states used by the OLAP processor are stored in the cache. For example, a user executes a query and the result set is stored in the cache. If another user executes the same or a similar query, the results are retrieved from the cache. It is quicker to retrieve from the cache instead of reading from the InfoProvider again.
- ▶ **Aggregates:** Aggregates are useful to query performance, as they contain a summarized version of the data stored in the InfoCubes. Less rows are scanned and returned when using an aggregate instead of an InfoCube.

2.3.6 Optimizing the query process

Since the only tuning allowed for the queries was through database tuning of the highly selective ad hoc queries, the next section describes the unit test of the queries. The remaining sections are some general recommendations for overall query tuning from an SAP perspective, which may not have been implemented for this stress test.

The query unit test

We ran and monitored the queries for approximately three hours, using the setup described in 2.2.5, “The query process” on page 65. Figure 2-47 shows the average response time and the average throughput (transactions per second).

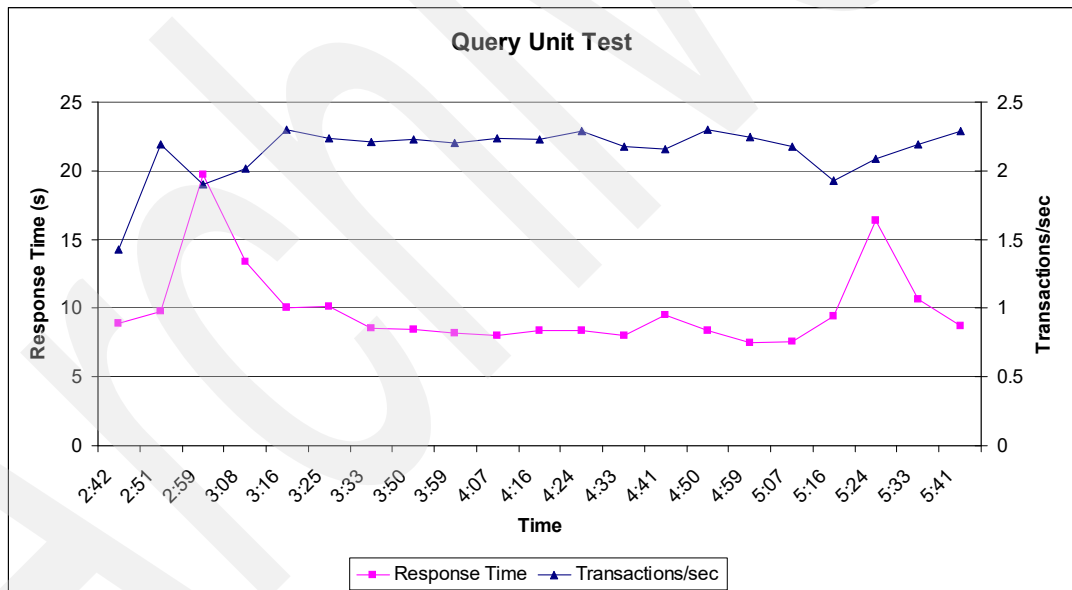


Figure 2-47 Query unit test

For the entire run, the average response time was 9.9 seconds, with an average of 2.14 transactions/sec. We can see that at certain times the response time increases to just under 20 seconds at 2:59. Of course, as the response time increases, the transaction/sec decreases to 1.9, as the average response time is longer.

The CPU usage on the application server sys1onl0 was relatively high. On average, 11 CPUs were used with a utilization of approximately 94.7%. Figure 2-48 on page 90 displays the CPU usage and the number of physical CPUs used for the application server sys1onl0. Perhaps we could spread the query workload across several application servers by using a

Web dispatcher to distribute the queries. We could also restrict the queries and users through logon groups, thus certain types of queries only execute on certain application servers.

Comparing the graphs between the query response time and the CPU usage, we can see that when the response times spike, the application server's CPUs are not fully utilized. This could indicate that the query is accessing the database and the application server is waiting for the query results.

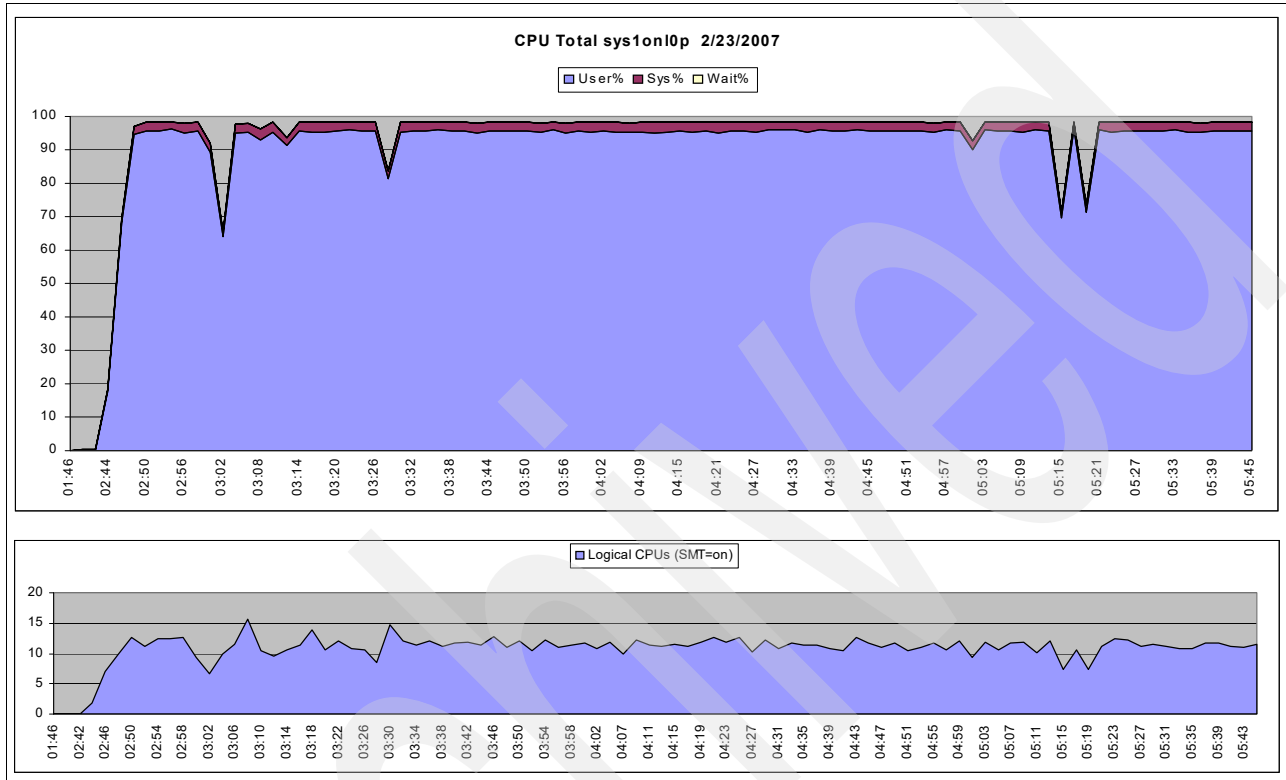


Figure 2-48 Query unit test: CPU usage sys1onl0

The database servers, specifically sys1db1, sys1db2, sys1db3, and sys1db4 had very similar CPU utilization. Figure 2-49, Figure 2-50 on page 92, and Figure 2-51 on page 93 display the CPU usage for all the database LPARs. We can see that when the utilization increases beyond 70%, the response time of the queries also increase, as the query is spending more time reading from the database.

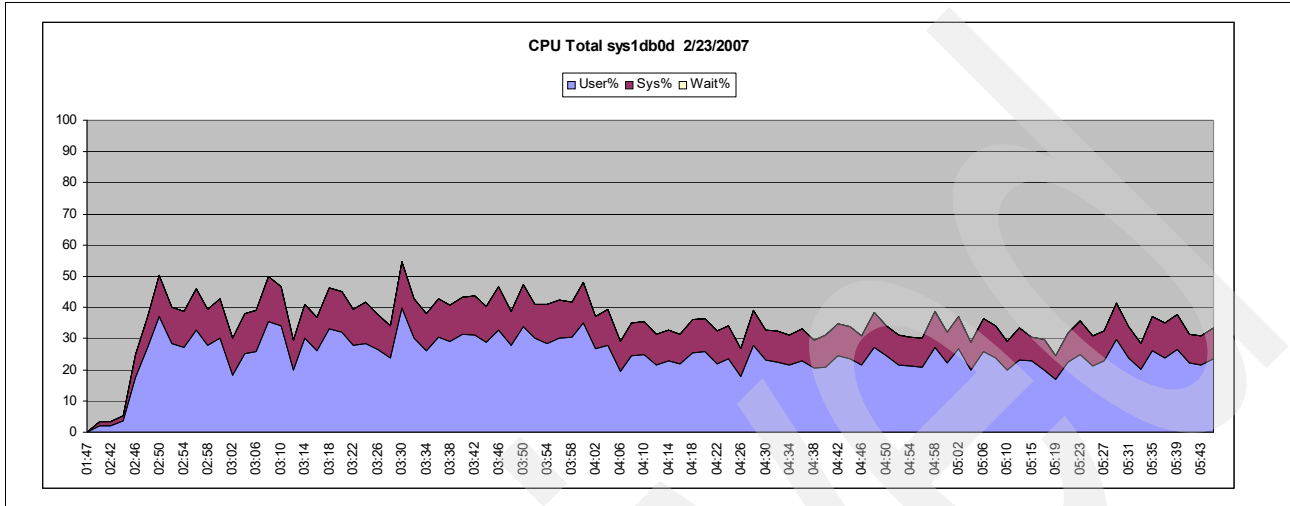


Figure 2-49 Query unit test - database LPAR CPU usage part 1 sys1db0

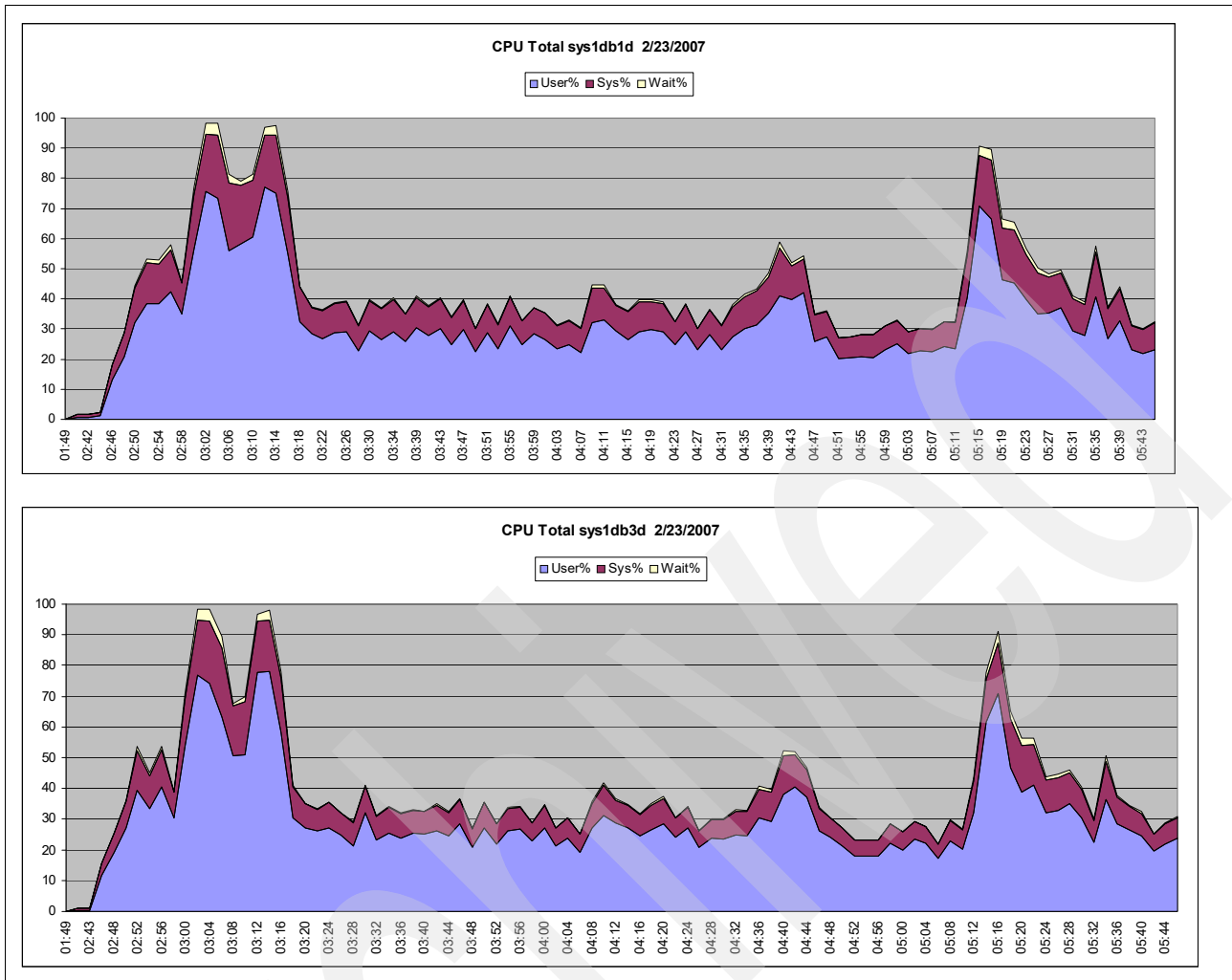


Figure 2-50 Query unit test - database LPAR CPU usage part 2 sys1db1 and sys1db3

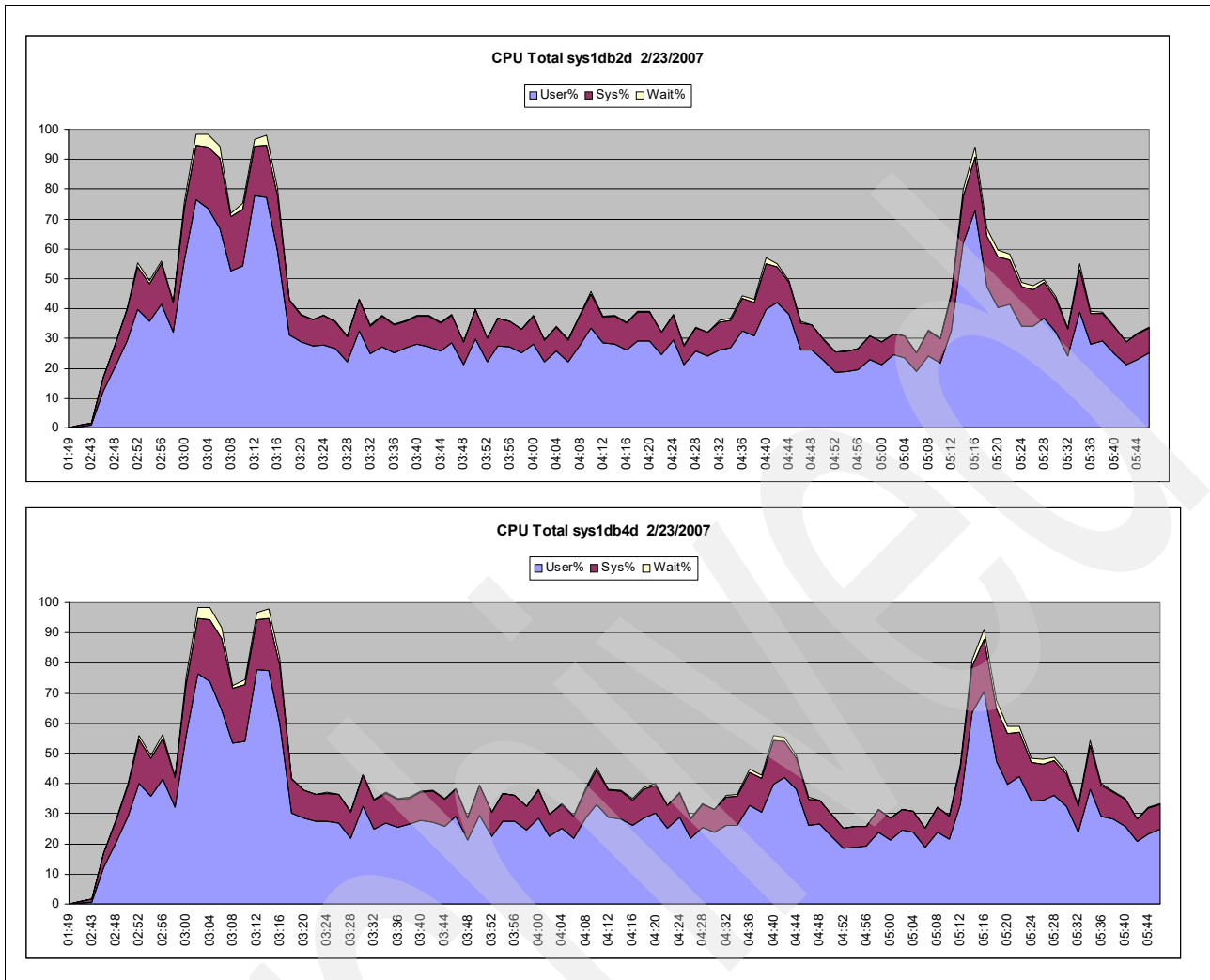


Figure 2-51 Query unit test - database LPAR CPU usage part 3 sys1db2 and sys1db4

Figure 2-52 and Figure 2-53 show the number of CPUs each database LPAR is using. On average, the CPU usage for all the database LPARs was approximately 4.5 CPUs. We can see that during the increases in CPU utilization, the total number of CPUs each LPAR is using increases significantly to 13 CPUs.

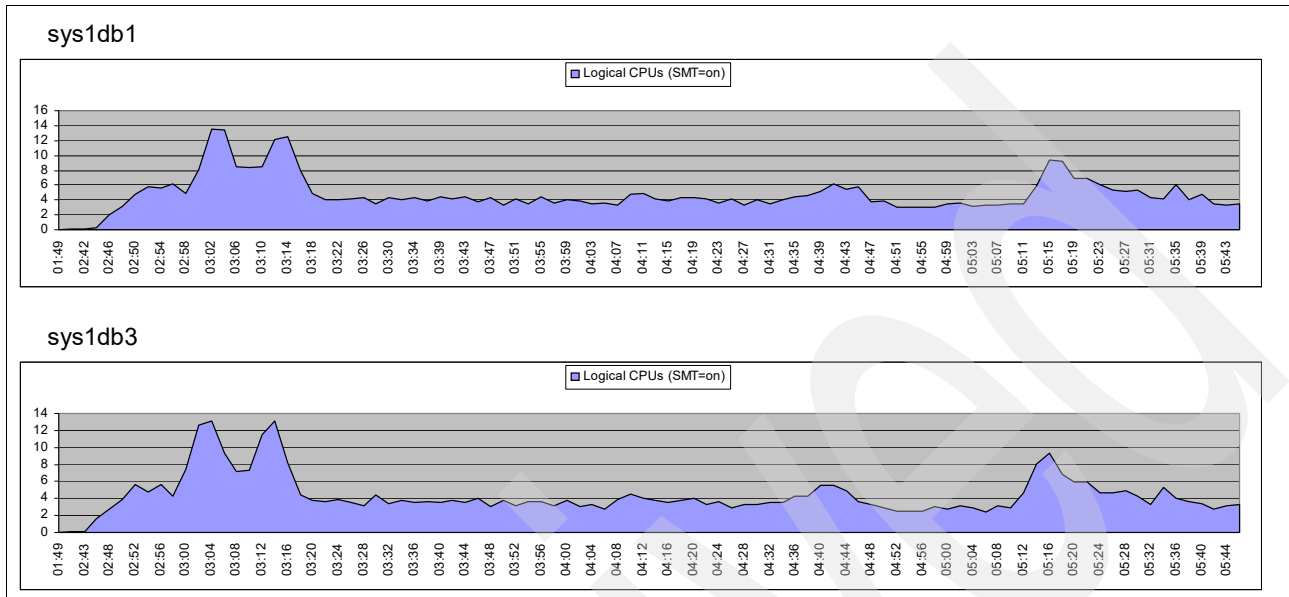


Figure 2-52 Number of CPUs for each database LPAR part 1: sys1db1 and sys1db3

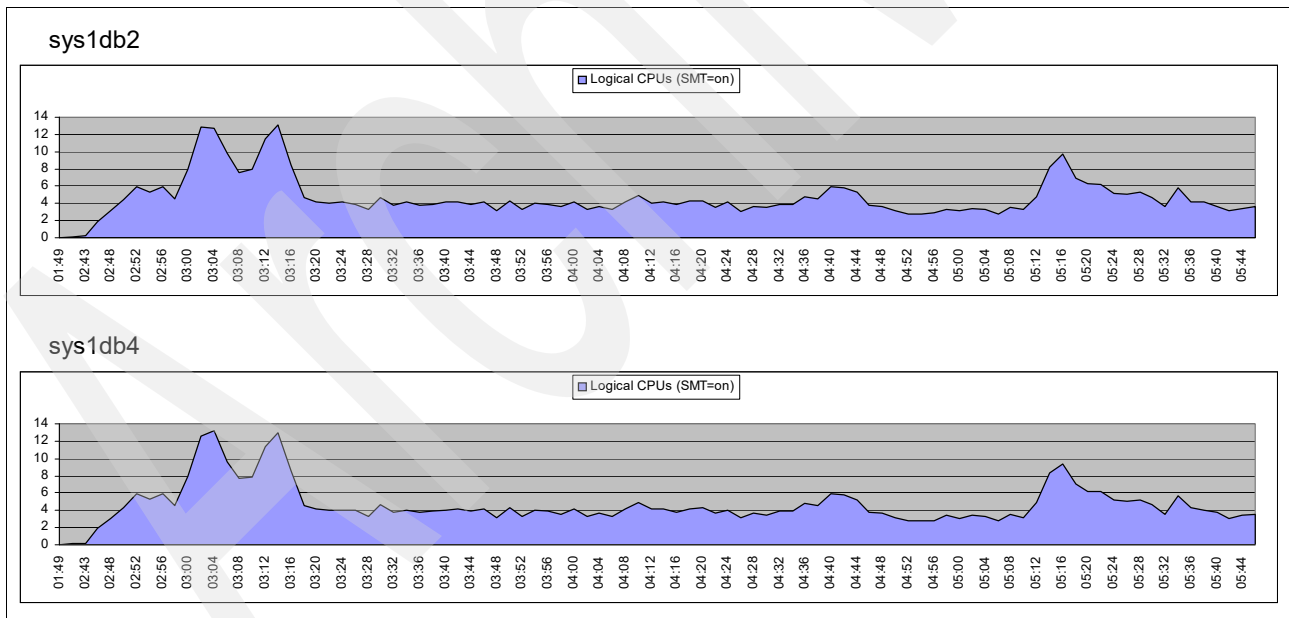


Figure 2-53 Number of CPUs for each database LPAR part 2: sys1db2 and sys1db4

Figure 2-54, Figure 2-55 on page 96, and Figure 2-56 on page 97 display the I/O and CPU utilization. We can see that the spikes in I/O translate to longer response times with the queries. Therefore, during these spikes in response time, there is a lot of reading done on the database.

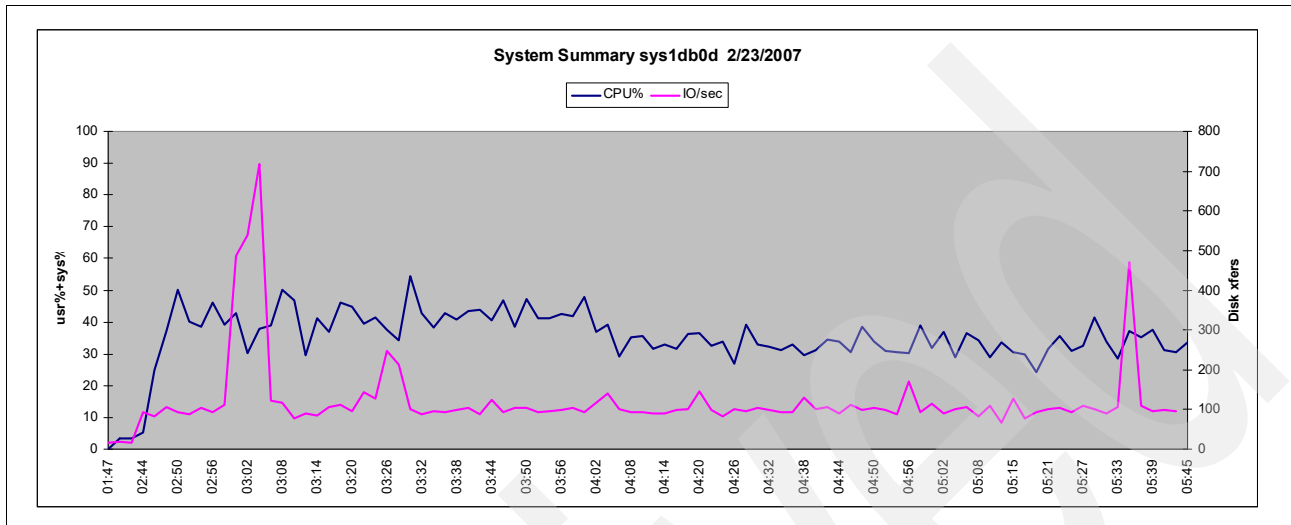


Figure 2-54 Query unit test - I/O and CPU utilization part 1: sys1db0

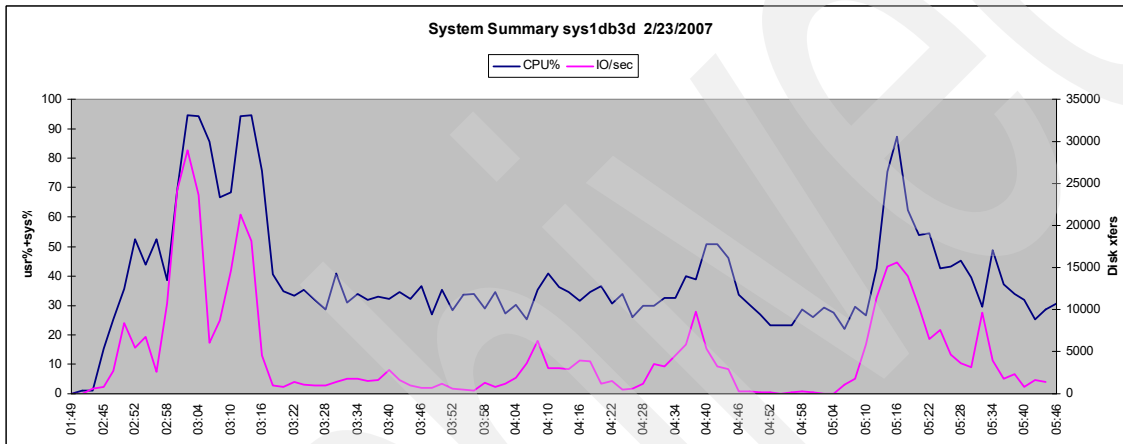
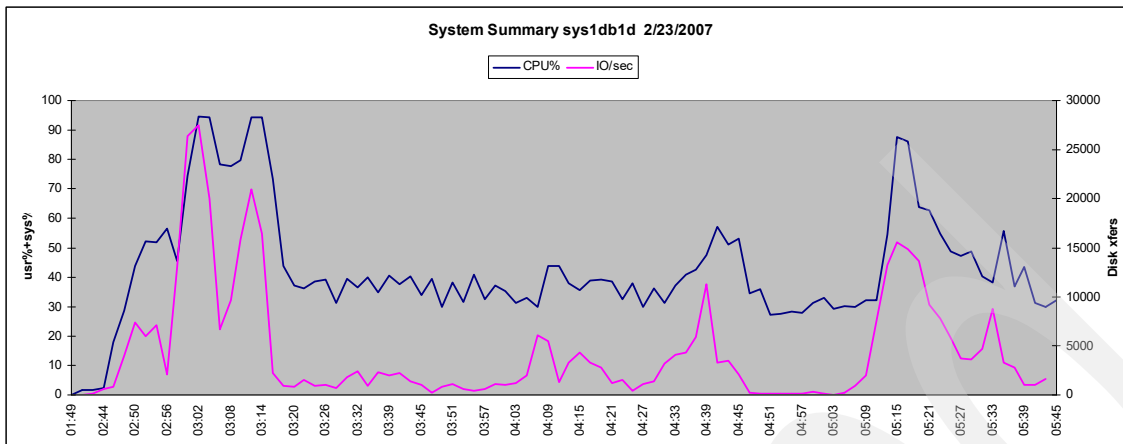


Figure 2-55 Query unit test - I/O and CPU utilization part 2: sys1db1 and sys1db3

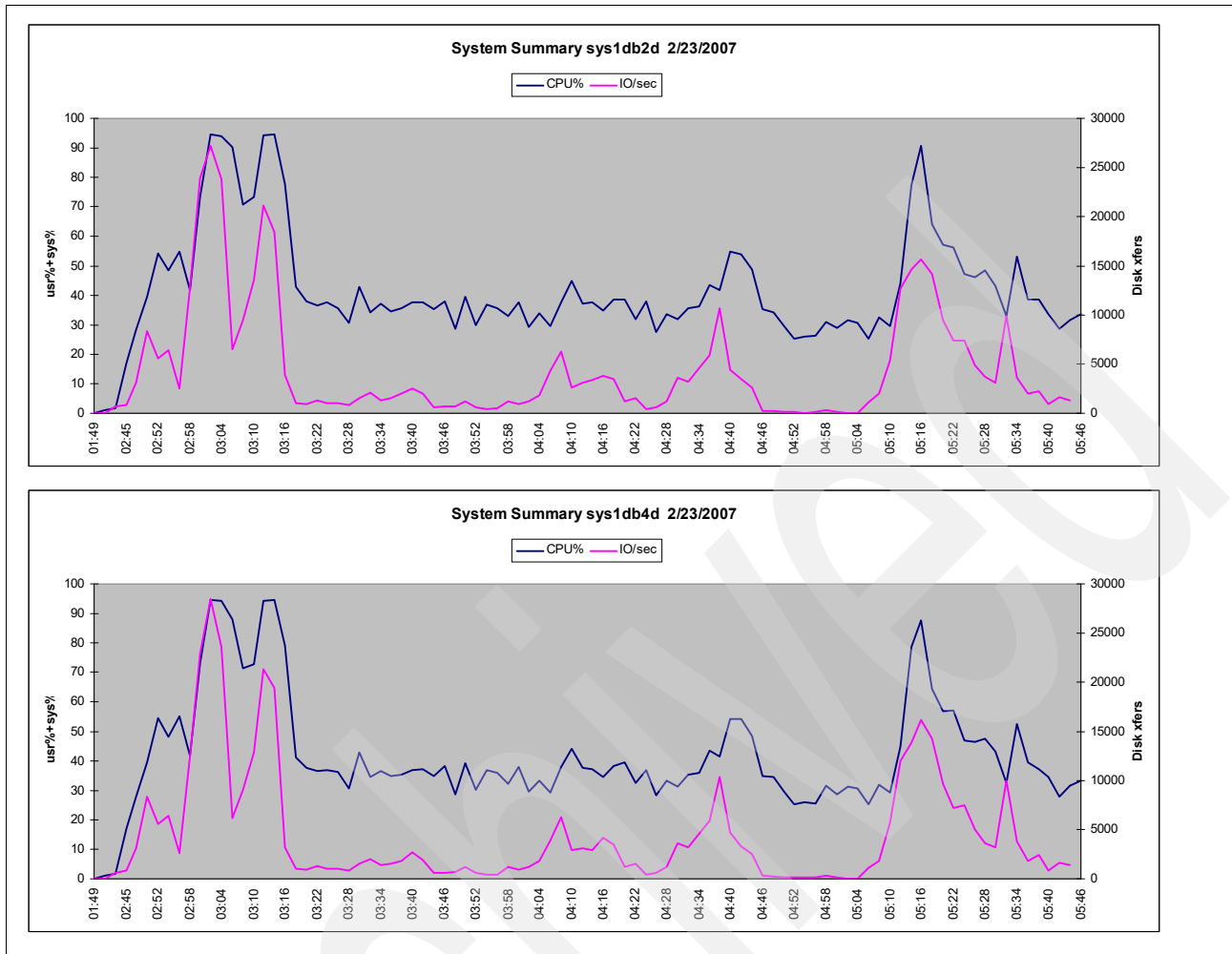


Figure 2-56 Query unit test - I/O and CPU utilization part 3: sys1db2 and sys1db4

Looking at the query profiles found in Table 2-2 on page 66, we see that two queries, in particular query Z60_STR_D, read from the fact tables of the InfoCubes instead of from an aggregate. Reading from an InfoCube instead of a fact table generally scans more rows than aggregates. Query Z60_STR_D not only reads from the fact table, but it does not utilize the OLAP cache, and it reads and returns more records than the others.

In Figure 2-57 we graphed the response times of all the queries with the average response time in yellow. Query Z60_STR_D is in red and plotted against the secondary y-axis on the right. We can see here that the average response time is greatly influenced by the response time of query Z60_STR_D, as it has the worst and most erratic response times.

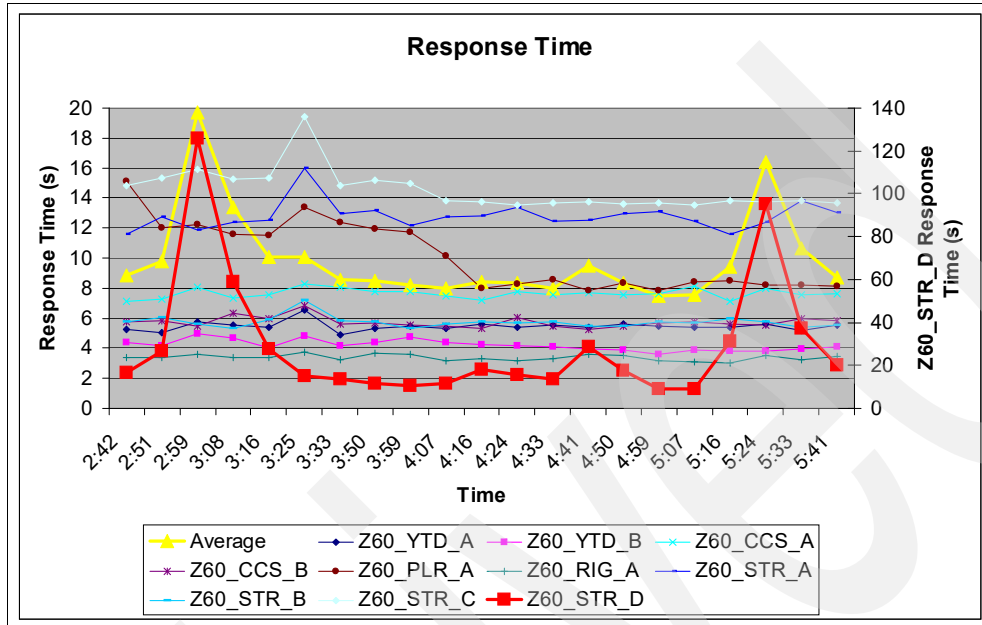


Figure 2-57 Average response time of all the queries

Since the STR_D query has the longest response time, we also looked at the number of STR_D queries running in parallel. Figure 2-58 displays a graph that shows the response time and the number of STR_D queries in parallel during the KPIG stress test.

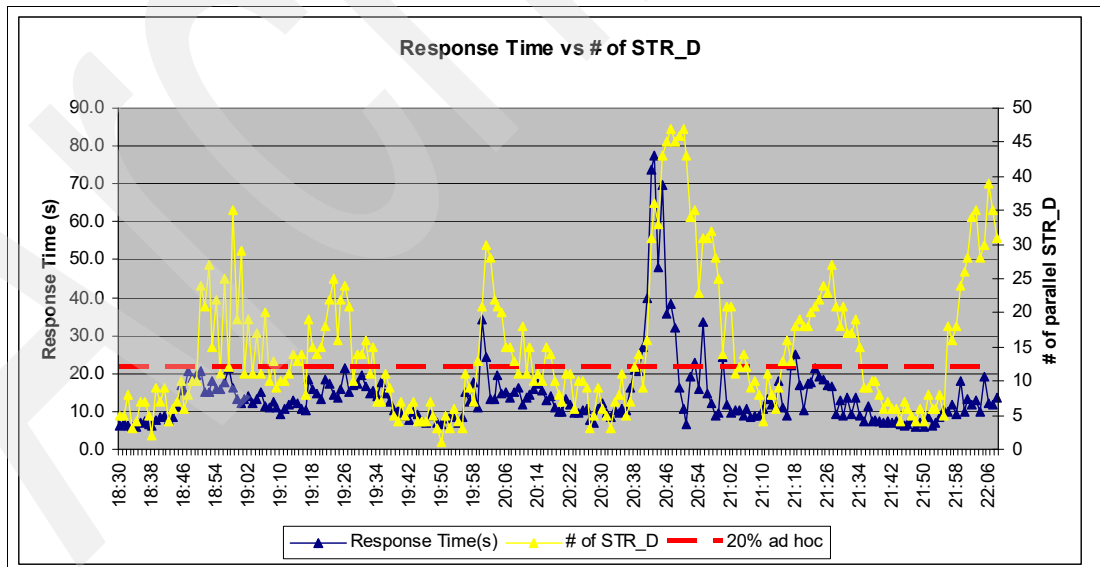


Figure 2-58 Response time versus number of STR_D

From the graph, we can see that as the number of STR_D queries running in parallel increases, the response time increases as well.

We also looked at the effect of the number of STR_D queries in parallel to the number of transactions/sec, which is displayed in Figure 2-59.

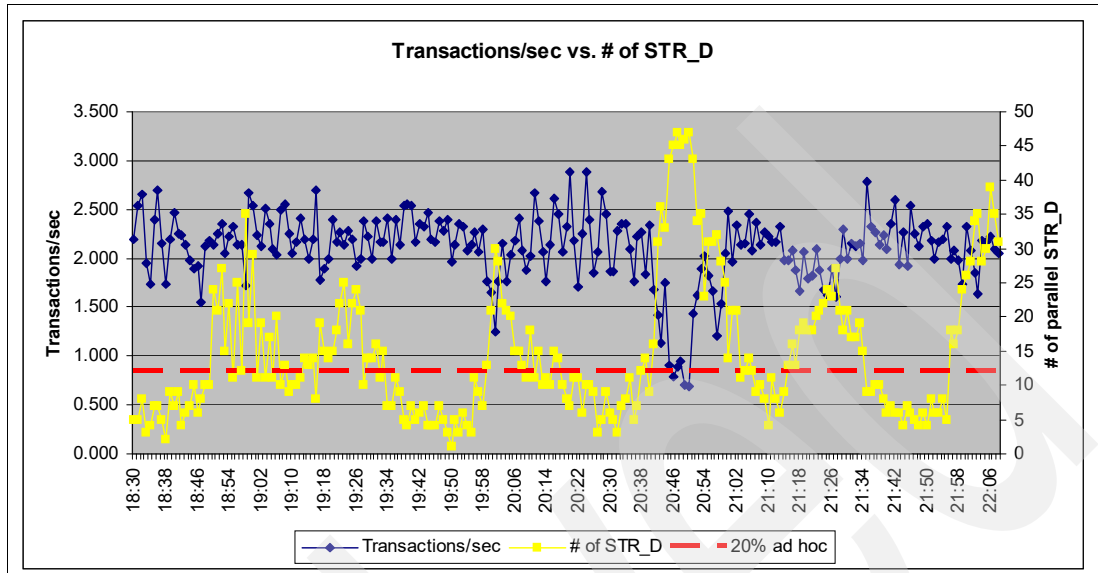


Figure 2-59 Transactions/sec versus the number of STR_D

Here we can see that as the number of STR_D queries running in parallel increases, the number of transactions/sec decreases.

Since the STR_D query represents ad hoc queries, or queries from power users, the customer required that 20% of the query workload consist of these queries. From our tests, we see that at the times that the number of STR_D queries in parallel makes up more than 40% of the workload, our response time increases above 20 seconds.

Therefore, from the unit test we can observe the following:

- ▶ CPU usage on the application server is relatively high.
 - We might want to consider spreading the query workload across several application servers to help alleviate the workload on some servers.
- ▶ Reading from the database is expensive in terms of throughput.
 - The number of STR_D queries in parallel greatly affects our average throughput and response times. Our response time increases to above 20 seconds when 40% of the workload is ad hoc queries.
 - If we could use an aggregate instead of reading from the InfoCube to reduce the amount of records read, we could improve our response time.

The OLAP cache

During the query process, the benefits of the OLAP cache were evident. In general, queries utilizing the OLAP cache had better response times and throughput (transactions/sec) compared to queries not using the OLAP cache. For example, Figure 2-60 shows two ST03 outputs. The top output displays the information for the MultiProviders that used the OLAP cache, while the bottom output is without the OLAP cache.

The figure consists of two screenshots of SAP ST03 reports. The top screenshot is titled "Reporting - InfoCubes: Ø Times / Navigation Step (s) / OLAP Cache Only" and the bottom is "Reporting - InfoCubes: Ø Times / Navigation Step (s) / No OLAP Cache". Red boxes highlight the "Ø Total" and "DB Time" columns in both reports. Arrows point from these boxes to labels "Response Time" and "Time spent in the database" respectively.

| InfoCube | No. Queries | Cache | No. of Nav. | Total time | Ø Total | MED: Tot. | OLAP Time | Ø OLAP | DB Time | Ø ... | Frontend | Ø Fronten |
|-----------|-------------|-------|-------------|------------|---------|-----------|-----------|--------|---------|-------|-----------|-----------|
| TOTAL | 20 | X | 36.825 | 225.797,5 | 6,1 | 5,2 | 25.716,8 | 0,7 | 0,0 | 0,0 | 190.455,4 | 5,2 |
| ZGTFCMP01 | 3 | X | 5.993 | 32.349,9 | 5,4 | 4,5 | 4.599,2 | 0,8 | 0,0 | 0,0 | 26.283,6 | 4,4 |
| ZGTFCMP02 | 2 | X | 3.141 | 23.331,6 | 7,4 | 6,5 | 1.844,4 | 0,6 | 0,0 | 0,0 | 20.587,8 | 6,6 |
| ZGTFCMP03 | 3 | X | 6.009 | 32.483,5 | 5,4 | 4,5 | 4.542,2 | 0,8 | 0,0 | 0,0 | 26.466,4 | 4,4 |
| ZGTFCMP04 | 2 | X | 3.332 | 25.141,6 | 7,5 | 6,5 | 1.946,2 | 0,6 | 0,0 | 0,0 | 22.229,5 | 6,7 |
| ZGTFCMP05 | 3 | X | 5.997 | 32.284,6 | 5,4 | 4,4 | 4.365,9 | 0,7 | 0,0 | 0,0 | 26.413,4 | 4,4 |
| ZGTFCMP06 | 2 | X | 3.128 | 22.971,3 | 7,3 | 6,4 | 1.867,5 | 0,6 | 0,0 | 0,0 | 20.197,2 | 6,5 |
| ZGTFCMP07 | 3 | X | 6.018 | 33.042,2 | 5,5 | 4,6 | 4.744,4 | 0,8 | 0,0 | 0,0 | 26.813,6 | 4,5 |
| ZGTFCMP08 | 2 | X | 3.207 | 24.192,8 | 7,5 | 6,5 | 1.807,0 | 0,6 | 0,0 | 0,0 | 21.463,8 | 6,7 |

| InfoCube | No. Queries | Cache | No. of Nav. | Total time | Ø Tot. | MED: Tot. | OLAP Time | Ø OLAP | DB Time | Ø ... | Frontend | Ø Fronten | Selected | Select / Transf. |
|-----------|-------------|-------|-------------|-------------|--------|-----------|-----------|--------|-----------|-------|-----------|-----------|---------------|------------------|
| TOTAL | 40 | | 58.737 | 1.086.711,5 | 18,5 | 10,9 | 33.716,9 | 0,6 | 659.945,1 | 11,2 | 373.222,7 | 6,4 | 4.614.773.317 | 319,4 |
| ZGTFCMP01 | 4 | | 3.575 | 39.242,2 | 11,0 | 9,0 | 5.622,4 | 1,8 | 17.657,6 | 4,9 | 15.012,3 | 4,2 | 70.152.786 | 991,2 |
| ZGTFCMP02 | 6 | | 11.233 | 202.586,8 | 18,0 | 10,6 | 2.914,6 | 0,3 | 117.060,7 | 10,4 | 78.552,1 | 7,0 | 1.014.160.829 | 275,1 |
| ZGTFCMP03 | 4 | | 3.585 | 40.201,3 | 11,2 | 10,3 | 5.645,8 | 1,6 | 18.344,9 | 5,1 | 15.260,6 | 4,3 | 108.661.000 | 1.261,0 |
| ZGTFCMP04 | 6 | | 11.058 | 252.489,5 | 22,8 | 12,9 | 2.867,5 | 0,3 | 168.141,3 | 15,2 | 77.480,0 | 7,0 | 1.381.538.630 | 378,0 |
| ZGTFCMP05 | 4 | | 3.539 | 34.993,6 | 9,9 | 9,3 | 5.602,1 | 1,8 | 13.564,6 | 3,8 | 14.911,0 | 4,2 | 79.859.205 | 987,0 |
| ZGTFCMP06 | 6 | | 11.164 | 214.340,4 | 19,2 | 10,2 | 2.787,3 | 0,2 | 129.474,7 | 11,6 | 78.052,2 | 7,0 | 411.088.182 | 122,8 |
| ZGTFCMP07 | 4 | | 3.499 | 34.883,1 | 10,0 | 9,6 | 5.429,0 | 1,6 | 13.728,3 | 3,9 | 14.809,2 | 4,2 | 79.692.439 | 1.222,2 |
| ZGTFCMP08 | 6 | | 11.084 | 267.974,5 | 24,2 | 13,1 | 2.848,3 | 0,3 | 181.973,1 | 16,4 | 79.145,4 | 7,1 | 1.469.620.246 | 424,8 |

Figure 2-60 OLAP cache versus no OLAP cache

As you can see in Figure 2-60, the MultiProviders that used the OLAP cache have faster response times compared to the MultiProviders that did not use the OLAP cache. For example, MultiProvider ZGTFCMP01 had three queries that used the OLAP cache, executing a total of 5993 navigations with a response time of 5.4 seconds. ZGTFCMP01 also had four queries that did not use the OLAP cache, executing 3,575 navigations with a response time of 11 seconds. The queries that used that OLAP cache did not have to read from the database again. This is why we see 0 seconds for the DB Time column. As a result of not reading the information from the database again, the response time was less.

Therefore, if we can limit the amount of times of re-reading the data from the database, we can improve the response time.

OLAP cache modes

It is not possible to avoid reading from the database. To lessen the amount of times necessary to read the database there are different cache modes available. As mentioned in “OLAP cache” on page 70, the following OLAP cache modes can be used:

- ▶ Main memory cache without swapping
- ▶ Main memory cache with swapping
- ▶ Cluster/flat file cache per application server
- ▶ Cluster/flat file cache across application servers

Each of the different cache modes has different benefits and costs. Knowing when to use one over the other depends on the query and infrastructure. For example, when a *without swapping* mode is used, displaced cache data is removed from the cache. If a *with swapping* mode is used, the displaced cache data can be reloaded into the cache from the background store, without having to re-read the data from the database again. Generally, the without swapping mode is used when the system suffers from poor I/O.

If there are high amounts of varying queries or large result sets are returned, it may be better to use a cluster/flat file cache as the memory restrictions from the application server are alleviated.

If certain queries are restricted to certain application servers, it is beneficial to use main memory caching, as the cached data does not change frequently. It also may be beneficial to use a cluster/flat file cache per application servers. However, for queries that execute across several application servers, a cross-application server cluster/flat file cache may be more beneficial.

Figure 2-61 briefly summarizes these considerations and recommendations. A single plus sign (+) indicates a suggested cache mode usage for the scenario/behavior. Double plus signs (++) indicate that the recommendation is suggested over the single plus sign recommendation. Three plus signs (+++) indicate that the recommendation is suggested over the double plus sign recommendation.

| | Inactive Cache | Main Memory without swapping | Main Memory with swapping | Cluster/Flat File per App Server | Cross-App server Cluster/Flat File |
|---|----------------|------------------------------|---------------------------|----------------------------------|------------------------------------|
| Frequent changes in data | + | - | - | - | - |
| Large result set | - | - | + | ++ | +++ |
| High amount of different queries | - | - | ++ | +++ | +++ |
| High amount of active users and query navigations | --- | +++ | +++ | +++ | +++ |
| Many ad hoc queries | ++ | --- | -- | - | - |
| High load on InfoProvider database tables | --- | +++ | +++ | +++ | +++ |
| Logon groups for queries | --- | +++ | +++ | +++ | + |
| Poor I/O | --- | +++ | ++ | + | + |

Figure 2-61 OLAP cache modes

Monitoring the OLAP cache

The OLAP cache can be monitored through transaction RSRCACHE. Using this transaction, a window (Figure 2-62) is displayed. It shows the local and global cache parameters and settings. The local OLAP cache size is part of the application buffer, so it is also important to use the Buffer Monitor and Buffer Overview push buttons available with transaction RSRCACHE.

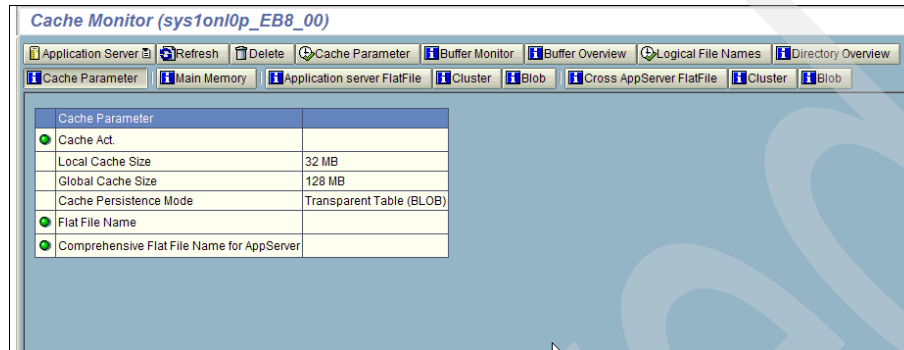


Figure 2-62 RSRCACHE: OLAP cache monitoring

From the Buffer Monitor window (Figure 2-63) we can check whether there is enough cache memory available. The shared memory rows in the Buffer Overview window area are relevant to the OLAP cache. From here we can see various types of information such as the maximum size of the cache, the number of free bytes, and the efficiency (hit ratio).

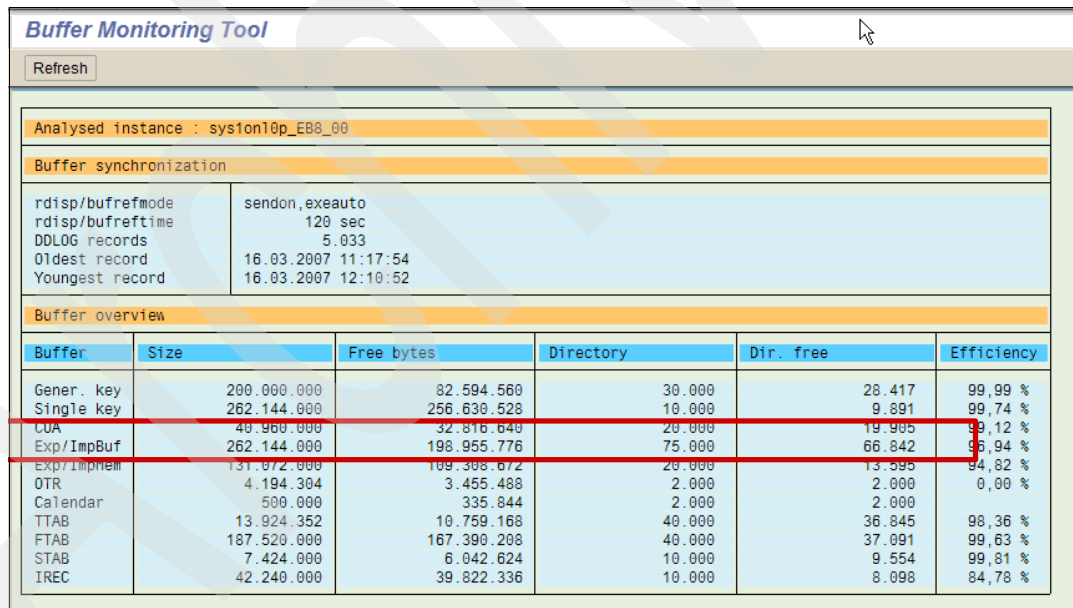


Figure 2-63 RSRCACHE: buffer monitoring

From the Buffer Overview window, which is the same as transaction ST02, displayed in Figure 2-64, the shared memory rows pertain to the OLAP cache. We can also see the efficiency, that is, how often the data is found in the buffer, and the number of swaps.

| Tune Summary (sys1on10p_EB8_00) | | | | | | | | | |
|--|-----------------|------------------------------|-----------------|--------------|----------------------|------------------------|--------------|-------|-------------------|
| Current parameters | | | | | Detail analysis menu | | | | |
| System: sys1on10p_EB8_00 | | Tune summary | | | | | | | |
| Date & time of snapshot: 16.03.2007 12:11:24 | | Startup: 15.03.2007 17:56:32 | | | | | | | |
| Buffer | Hitratio [%] | Allocated [kB] | Free space [kB] | Free [%] | Dir. size Entries | Free directory Entries | Free [%] | Swaps | Database accesses |
| Nametable (NTAB) | | | | | | | | | |
| Table definition | 98,36 | 13.598 | 10.506 | 92,11 | 40.000 | 36.843 | 92,11 | 0 | 60.536 |
| Field description | 99,63 | 183.125 | 163.458 | 90,81 | 40.000 | 37.089 | 92,72 | 0 | 12.253 |
| Short NTAB | 99,81 | 7.250 | 5.901 | 98,35 | 10.000 | 9.554 | 95,54 | 0 | 446 |
| Initial records | 84,76 | 41.250 | 38.889 | 97,22 | 10.000 | 8.096 | 80,96 | 0 | 11.204 |
| Program | | | | | | | | | |
| CUA | 99,09 | 40.000 | 31.994 | 97,52 | 20.000 | 19.902 | 99,51 | 0 | 99 |
| Screen | 99,60 | 39.063 | 37.632 | 96,91 | 10.000 | 9.943 | 99,43 | 0 | 61 |
| Calendar | 100,00 | 488 | 328 | 100,00 | 2.000 | 2.000 | 100,00 | 0 | 0 |
| OTR | 100,00 | 4.096 | 3.375 | 100,00 | 2.000 | 2.000 | 100,00 | 0 | 0 |
| Tables | | | | | | | | | |
| Generic key | 100,00 | 195.313 | 80.659 | 43,36 | 30.000 | 28.416 | 94,72 | 0 | 2.901 |
| Single record | 99,74 | 256.000 | 250.616 | 99,11 | 10.000 | 9.891 | 98,91 | 0 | 13.143 |
| Export/import | 96,94 | 256.000 | 194.291 | 84,83 | 75.000 | 66.836 | 89,11 | 0 | 0 |
| Exp./Imp. SHM | 94,82 | 128.000 | 106.732 | 88,35 | 20.000 | 13.589 | 67,95 | 0 | 0 |
| SAP memory | | | | | | | | | |
| | Current use [%] | Max. use [kB] | In memory [kB] | On disk [kB] | SAP cursor cache | | Hitratio [%] | | |
| Roll area | 0,23 | 2.411 | 4.032 | 128.000 | IDs | | 99,76 | | |
| Paging area | 0,08 | 826 | 1.704 | 128.000 | Statements | | 99,00 | | |
| Extended Memory | 6,23 | 4.169.728 | 9.039.872 | 60977.792 | | | | | |
| Heap Memory | | 0 | 0 | | | | | | |
| Call statistics | | | | | | | | | |
| 364 tables buffered | Hitratio [%] | ABAP/4 Requests | Processor Fails | Total calls | Database AvgTime[ms] | Rows affected | | | |
| Select single | 98,83 | 11.004.421 | 698.123 | 418.714 | 0,000 | 10.306.298 | | | |
| Select | 86,42 | 15.833.463 | 0 | 2.155.401 | 0,000 | 46.237.250 | | | |
| Insert | | 10.136 | 800 | 10.136 | 0,000 | 10.406 | | | |
| Update | | 60 | 0 | 60 | 0,000 | 60 | | | |
| Delete | | 18.608 | 9.302 | 18.608 | 0,000 | 10.382 | | | |
| Total | 92,63 | 26.866.688 | 708.225 | 2.602.919 | | 56.564.396 | | | |

Figure 2-64 RSRCACHE: buffer overview

For more information regarding the tuning of the OLAP cache visit:

<https://www.sdn.sap.com/irj/sdn/howtoguides>

Navigate to the Business Intelligence section.

Warming up the OLAP cache

A strategy to help with response time and the efficiency of the cache is to *warm up* the cache. This involves running key queries against the new data before the first user queries begin to access the data. As a result, some of the new data is loaded into the OLAP cache and increases the chances that there will be an efficient *hit* in the OLAP cache.

Figure 2-65 shows some of the average response times of queries for different runs. We can see that the first couple of queries have higher response times, as the query has to read from the InfoProvider, as it is not in the OLAP cache. However, as time continues, the probability of the information being in the OLAP cache increases. We can see that the average response times after the OLAP cache has been warmed up decreases.

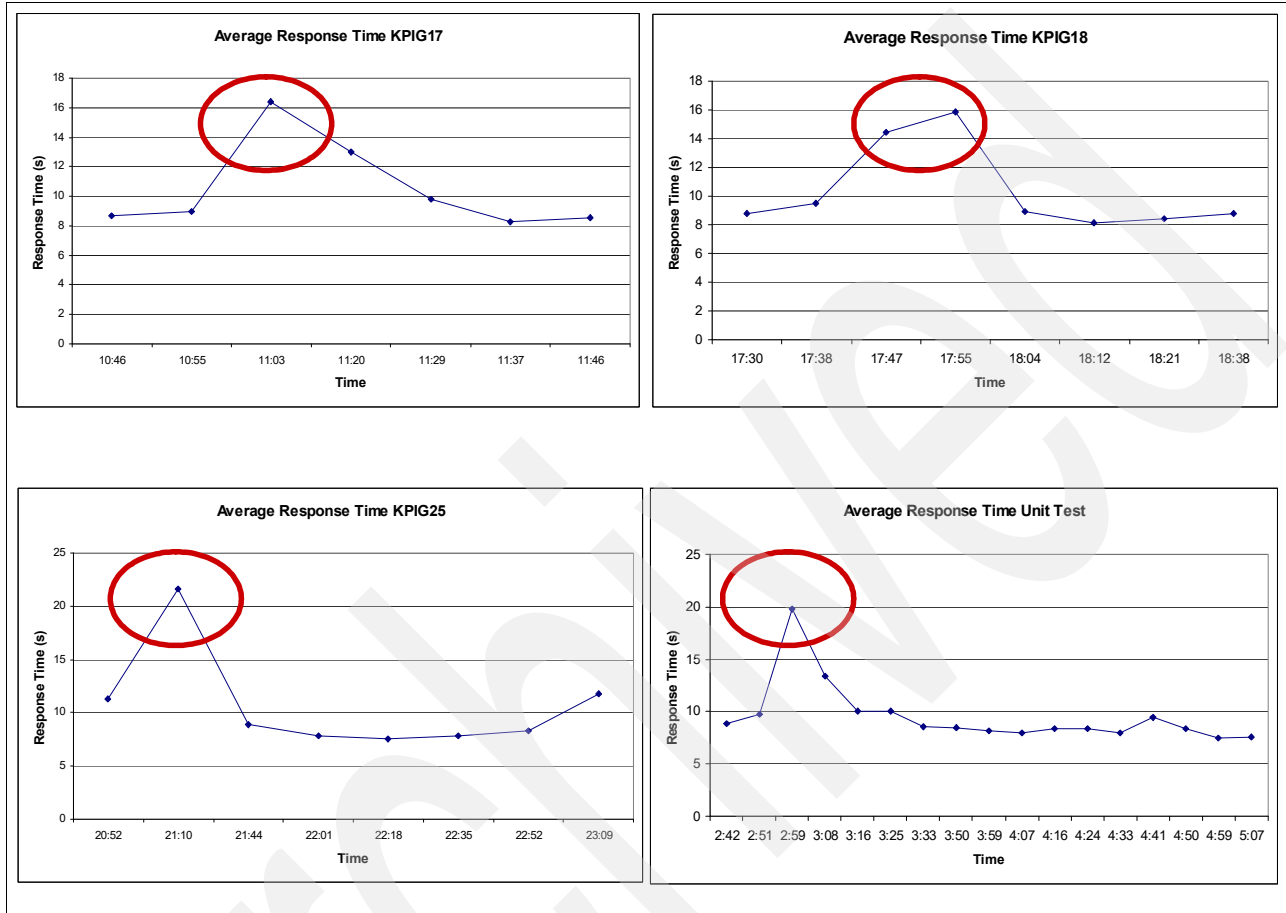


Figure 2-65 Warming up the OLAP cache

Note: If you would like to read more of information regarding the performance tuning of the OLAP cache, you can visit <http://service.sap.com/bi> and navigate to **Media Library** → **Performance**.

Aggregates

Aggregates can improve query performance, as the volume of data read from an aggregate is usually less than reading from the InfoCube. We can analyze the ST03 data from the query workload during the stress tests and we can find potential aggregates that could possibly improve the query performance. The general rule for creating and finding reasonable aggregates is as follows:

- ▶ The total time spent in the database comprises a large portion of the total time (greater than 30%).
- ▶ The ratio of selected rows to transferred rows is greater than 10.

Keeping the above rules in mind, we looked at the ST03 data, shown in Figure 2-66, for one of the MultiProviders. Here we see that the time spent in the database is approximately 47.7% of the total time, and the ratio of selected to transferred records is approximately 800. An aggregate could increase the performance of this query.

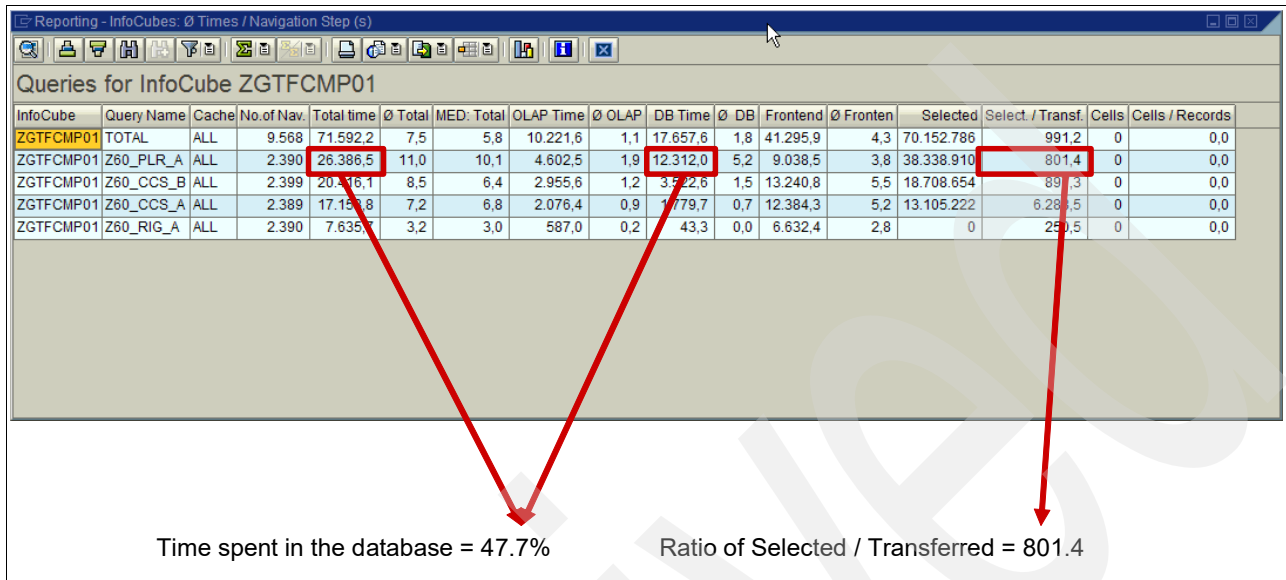


Figure 2-66 Potential aggregate - case 2

Looking at the ST03 data for another query (shown in Figure 2-67) we can see that the ratio of selected to transferred records is approximately 800. However, the ratio of time spent in the database is only 17%. A new aggregate for this query might not improve the query performance.

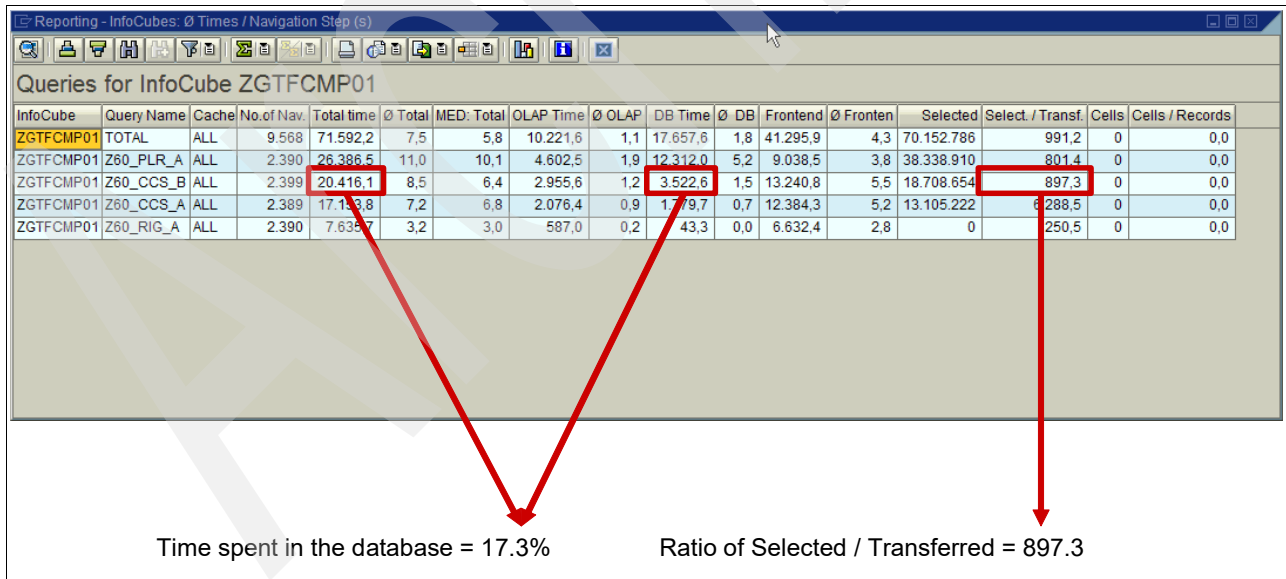


Figure 2-67 Not a potential aggregate - case 2

Looking at another query shown in Figure 2-68, an aggregate would definitely help the performance of the query. Here the query spends 79% of the time in the database. It selects a lot of records and transfers very little data, as the ratio is 190.

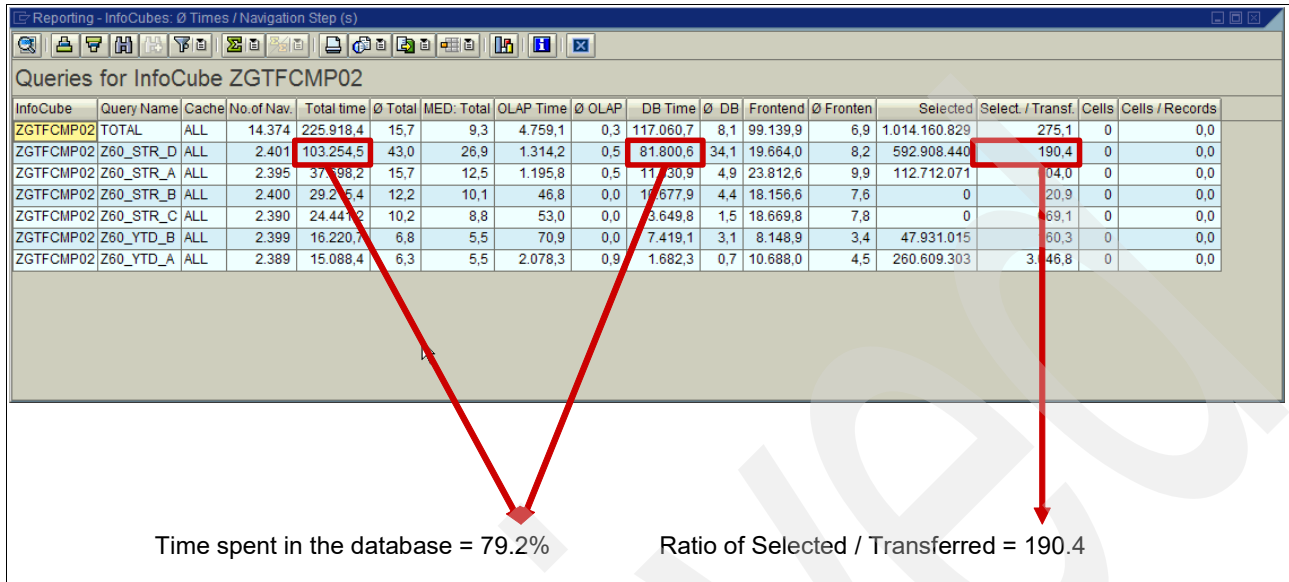


Figure 2-68 Potential aggregate case 3

As we can see, there may be benefits to creating new aggregates for the queries analyzed. However, there is a trade-off. The more aggregates we have, the more maintenance is required. This impacts the load process of roll-up and change-runs. Therefore, you have to find the right balance.

Summary: From the unit test and various monitoring techniques the following recommendations can be done to optimize the query process:

- ▶ Utilize the OLAP cache.

Queries that use the OLAP cache generally have faster response times than those that have to re-read the information from the InfoProvider.
- ▶ Warm up the OLAP cache.

With a warm OLAP cache, the probability that the data requested is in the cache is higher.
- ▶ Choose the correct cache mode for the infrastructure.

For example, if some user groups and queries are restricted to a specific application server, it may be beneficial to use a cache mode such as main memory with swapping or cluster/flat file.
- ▶ Use aggregates.

Aggregates lessen the load on the database as less records are read. However, there is a trade-off. The more aggregates that there are, the more maintenance is needed during the load process due to rollup and change runs.

2.3.7 Impact of query and aggregate processes running in parallel

In preparation for the combined load stress test, we executed the query and aggregate processes in parallel. We used the same setup as the query unit test described in “The query

unit test” on page 89. For the initial fill of the aggregates, only 10 InfoCubes with 10 aggregates were used.

From the unit test of the aggregates (“Optimization with SAP NetWeaver BI 3.5” on page 84) we found that the average throughput was 4.3 Mio rec/hr with a standard deviation of 0.29 Mio rec/hr. With 10 InfoCubes, we estimate that the throughput would be approximately 43 Mio rec/hr plus or minus 2.9 Mio rec/hr if the query process does not affect the processes significantly.

We started the aggregate process first, and after 45 minutes, we started the query process. Figure 2-69 displays the throughput of the initial fill of the aggregates along with the response time of the queries.

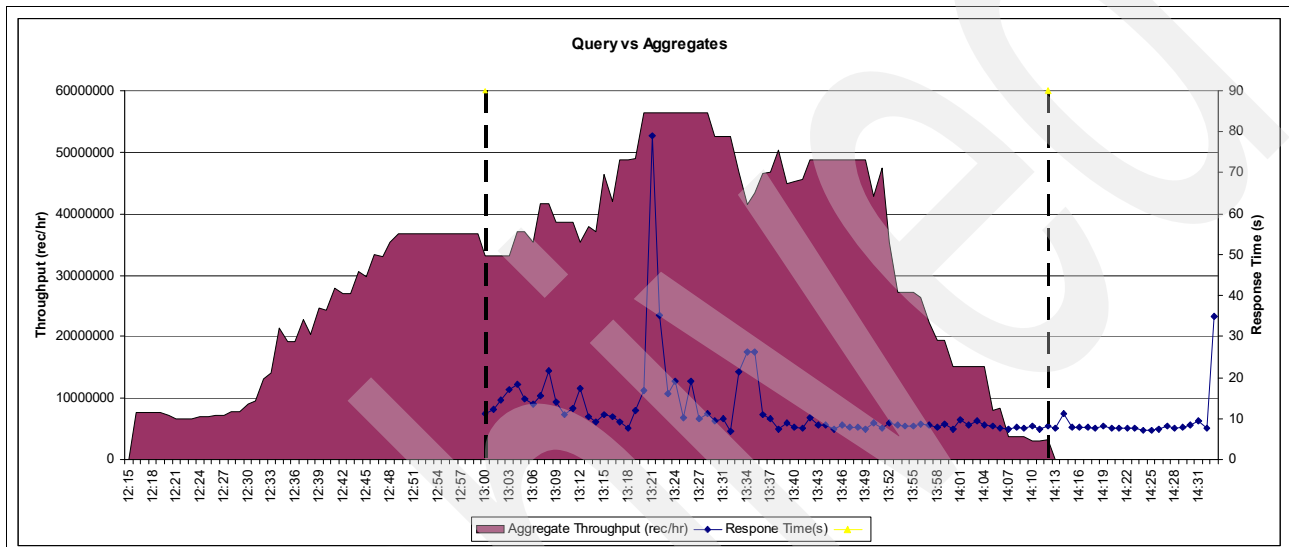


Figure 2-69 Query and aggregate throughput

The throughput was calculated during the time frame of when both processes were running, from 13:00 to 14:12. The black dotted lines represent the high load phase.

During the high load phase, the queries achieved an average of 2.03 transactions/sec with an average response time of 12.3 seconds, shown in Figure 2-70.

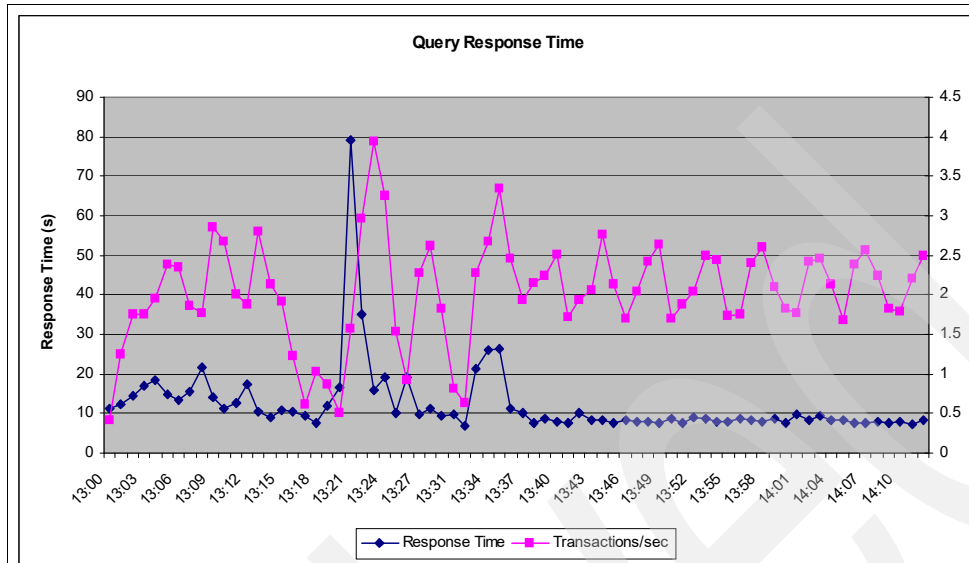


Figure 2-70 Query throughput during high-load phase

During the high load phase, the throughput of the aggregates was 36.9 Mio rec/hr, as displayed in Figure 2-71.

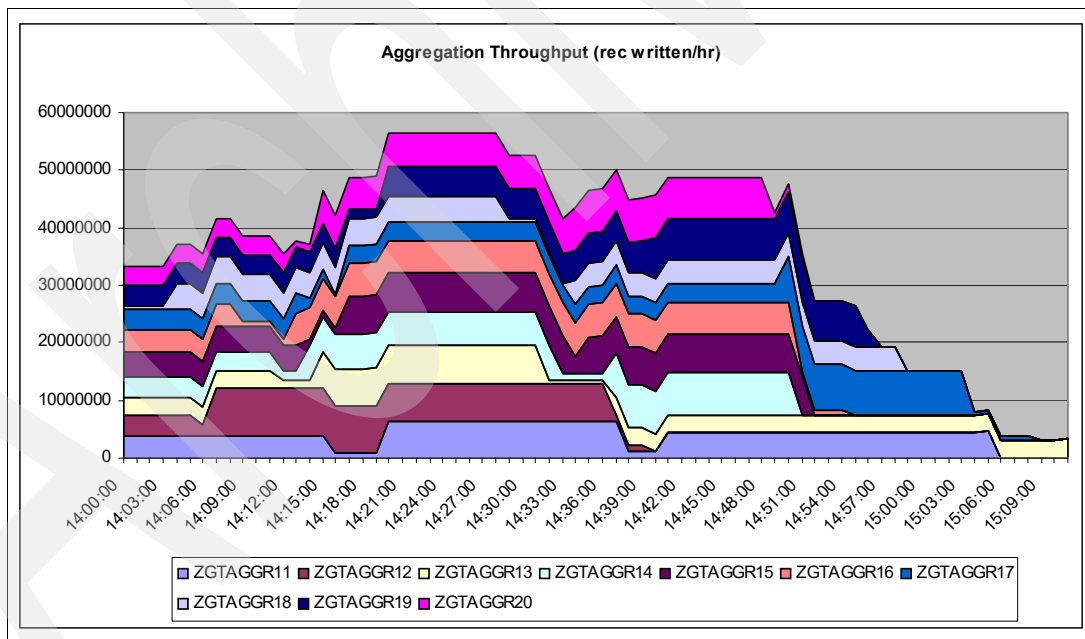


Figure 2-71 Aggregate throughput during high load phase

Before the test, we estimated that the aggregates would achieve slightly less than 42.7 Mio rec/hr. However, the queries had impacted the throughput of the aggregates significantly. The average throughput during the high load phase was 36.9 Mio rec/hr. The throughput was reduced by 5.8 Mio rec/hr. The total time to fill 100 aggregates was approximately 1 hour and 56 minutes. Compared to the unit test, the time increased by 27 minutes.

From the response times of the queries, there was a moment when the response time jumped to 78 seconds. Looking at the CPU usage of the database LPARs, displayed in Figure 2-72 and Figure 2-73 on page 110, we see that the high spikes in response time correspond to the times when there are spikes in I/O.

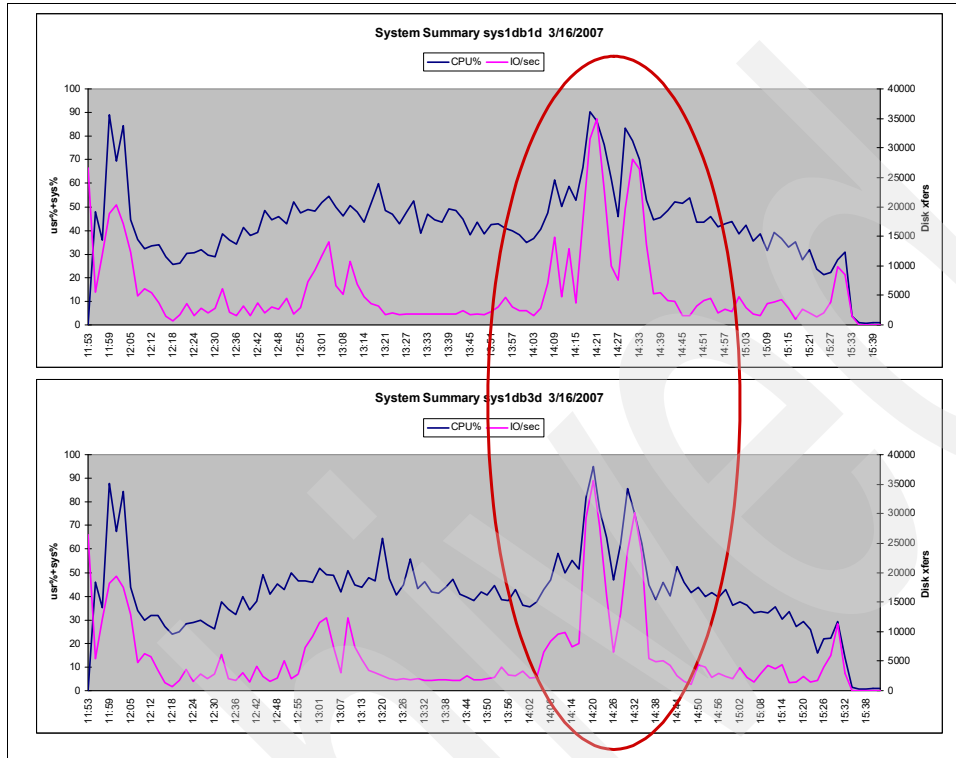


Figure 2-72 CPU and IO part 1- sysdb1 and sysdb3

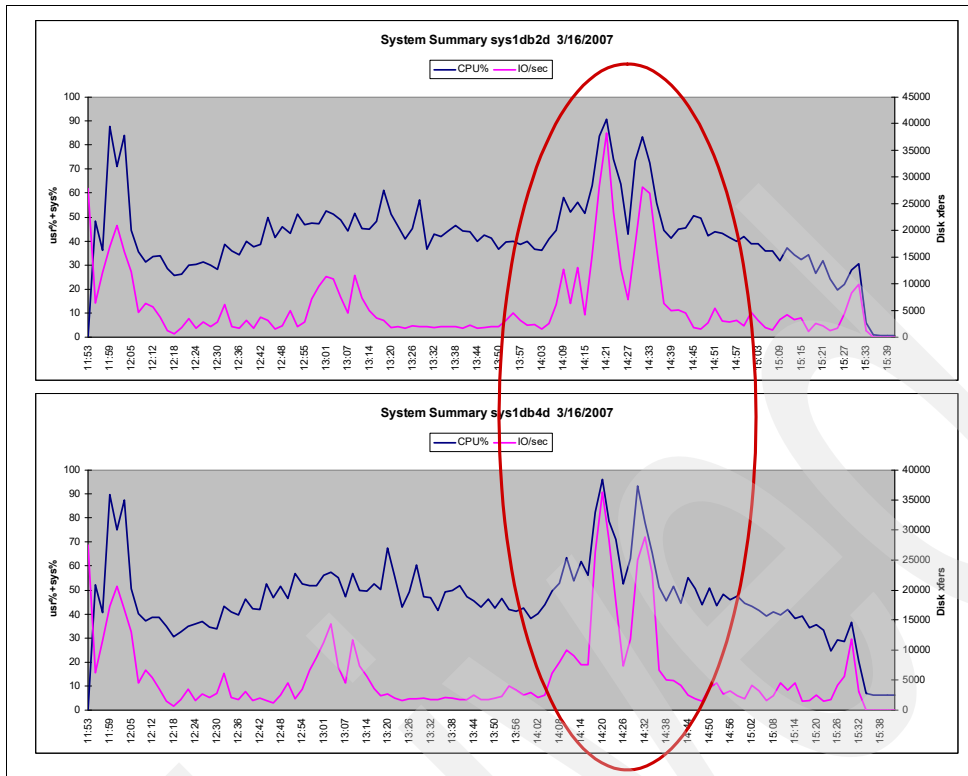


Figure 2-73 CPU and I/O part 2: sys2db and sys4db

The overall average CPU used during the high load phase across all the database LPARs was approximately 8.5 CPUs, nearly double compared to the query unit test and aggregate unit test. Figure 2-74, Figure 2-75 on page 111, and Figure 2-76 on page 111 display the number of CPUs used.

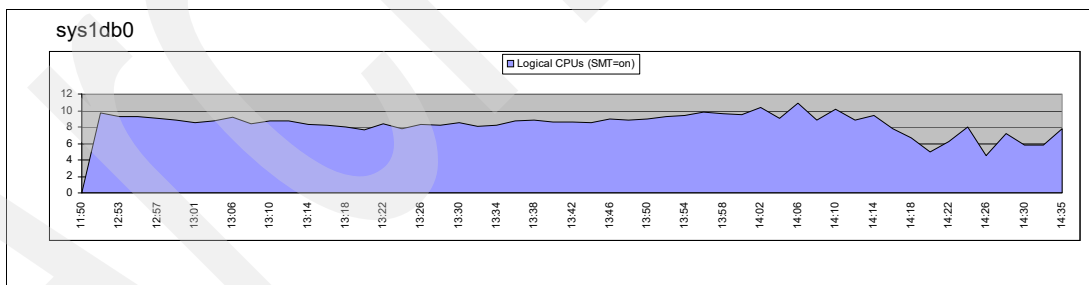


Figure 2-74 CPU usage of the database LPARs part 1 - sys1db0

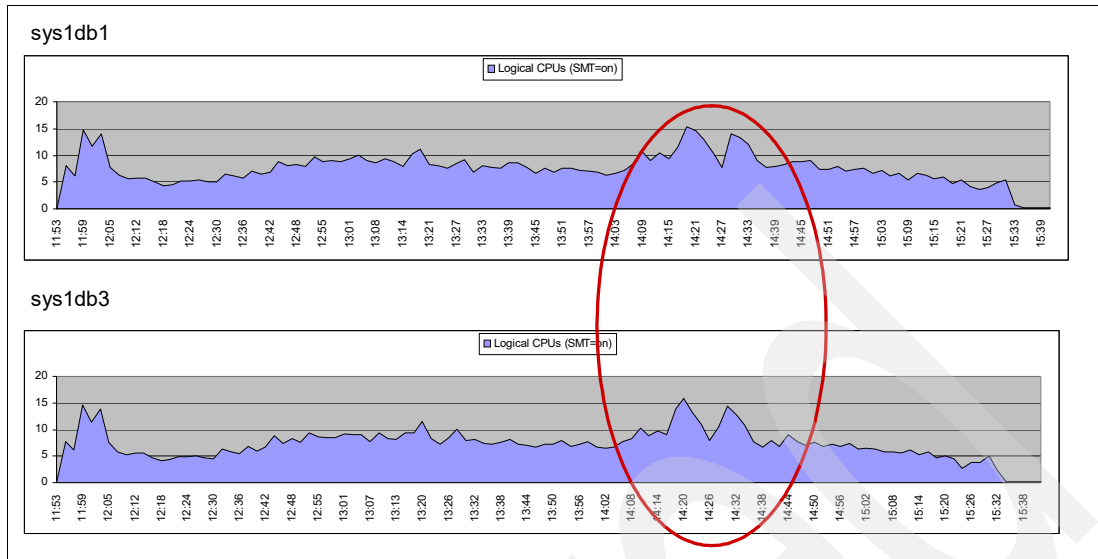


Figure 2-75 CPU usage of the database LPARs part 2 - sys1db1 and sys1db3

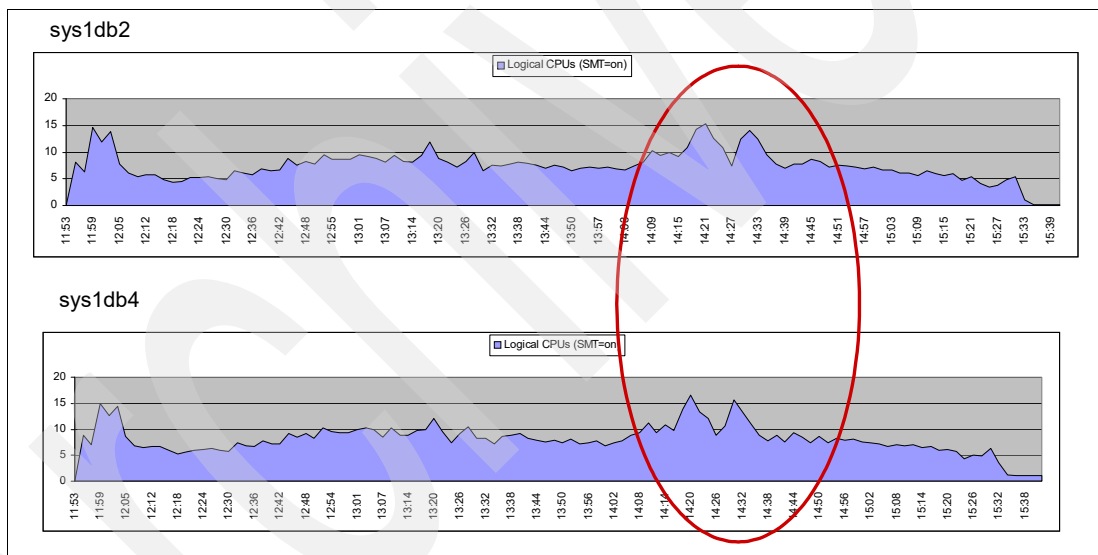


Figure 2-76 CPU usage of the database LPARs part 3 - sys1db2 and sys1db4

Summary: The query process affects the aggregate process. The throughput was reduced by approximately 5.8 Mio rec/hr and the length of the run increased by 27 minutes. The total number of CPU used on the database LPARs increased by 89%.

The aggregate process does not affect the query process significantly. The response time increased by 24%, from 9.9 seconds during the unit test to 12.3 seconds. The transactions/sec also decreased by 5.1% from 2.14 transactions/sec during the unit test to 2.03 transactions/sec

2.3.8 Impact of the load and the aggregate processes running together

In preparation for the combined stress test, we executed the load process and the aggregate fill process together. The purpose of this test was to see how the combined processes behave and affect each other.

The settings for the initial fill of the aggregates were the same as the settings in 2.3.7, “Impact of query and aggregate processes running in parallel” on page 106, that is, 10 InfoCubes with 10 aggregates each.

Using the settings determined from the preliminary tests, we ran used the settings and configuration for the load process described in Table 2-7 and Table 2-77.

Table 2-7 Load test settings

| # of ODS | # of extractors per ODS | # of InfoCubes per ODS | MAXPROC | MAXSIZE |
|----------|-------------------------|------------------------|---------|---------|
| 4 | 2 | 6 | 44 | 80,000 |

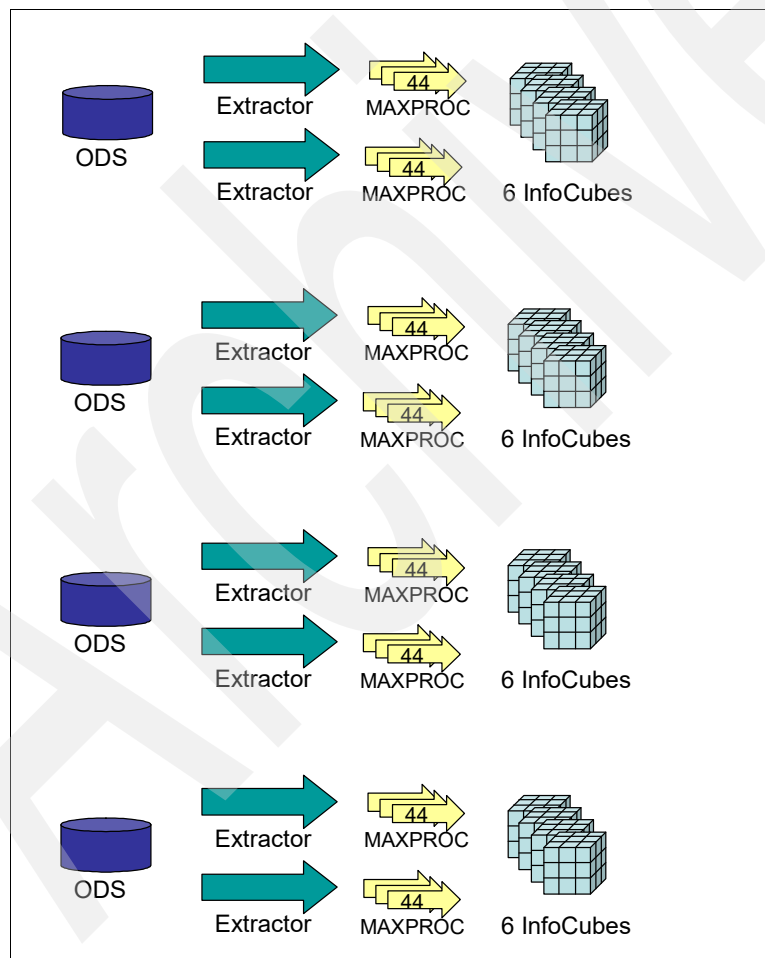


Figure 2-77 Load test settings

With a MAXPROC setting of 44, this translates to each extractor running up to 44 parallel processes, for a total of 352 dialog processes that could be used. Therefore, each application

server should have at least 88 dialog processes available for the load. Table 2-8 describes the settings used for the application servers.

Table 2-8 Application servers used for loading

| Application Server | Use | # of DIA wp | # of BTC wp |
|--------------------|----------------------------------|-------------|-------------|
| sys1ext0 | Extraction | 32 | 32 |
| sys1btc1 | Transformation rules and loading | 100 | 18 |
| sys1btc2 | Transformation rules and loading | 100 | 18 |
| sys1btc3 | Transformation rules and loading | 100 | 18 |
| sys1btc4 | Transformation rules loading | 100 | 18 |

For the application servers used to perform the transformation rules and loading, the number of dialog workprocess was set to 100, ensuring that each extractor can use up to 44 dialog processes in parallel.

The throughput should be calculated during the high-load phase (“High-load phase” on page 62). Therefore, we started the load process first, and once it has finished ramp-up, we started the aggregate process.

Figure 2-78 displays the throughput of the load and aggregate processes. The high-load phase is marked off with the vertical black dotted lines.

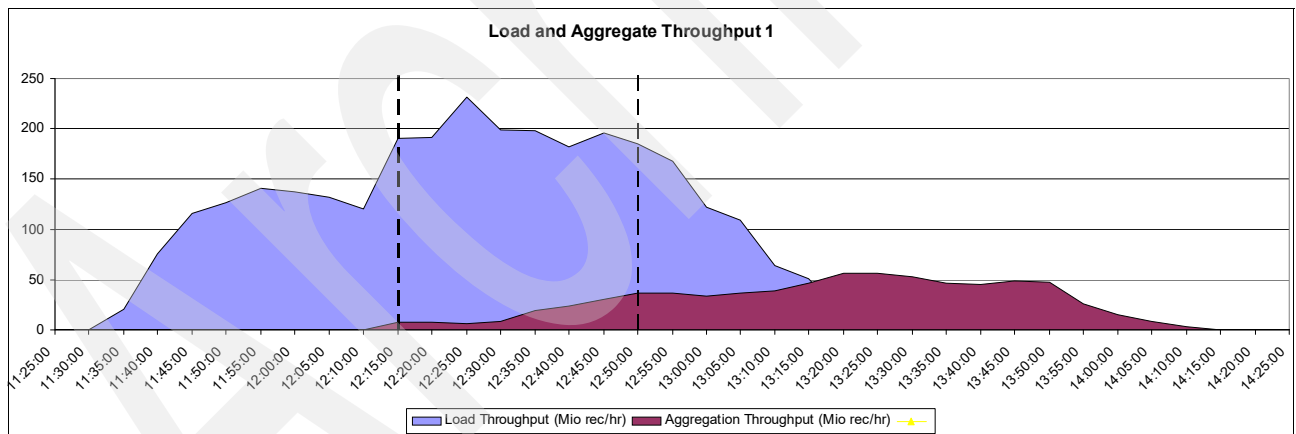


Figure 2-78 Load and aggregate throughput 2

The average throughput of the load during the high-load phase was 196.78 Mio rec/hr, while the throughput of the initial fill of the aggregates was 17.61Mio rec/hr.

However, looking at the aggregate throughput, it looks like the aggregate process is ramping up, and the throughput is relatively low at the start of the processes and increases steadily. Therefore, we decided to run another test, with the aggregates ramping up first and then starting the load process.

Figure 2-79 displays the second load and aggregate stress test. The high-load phase is marked off with the vertical black dotted lines.

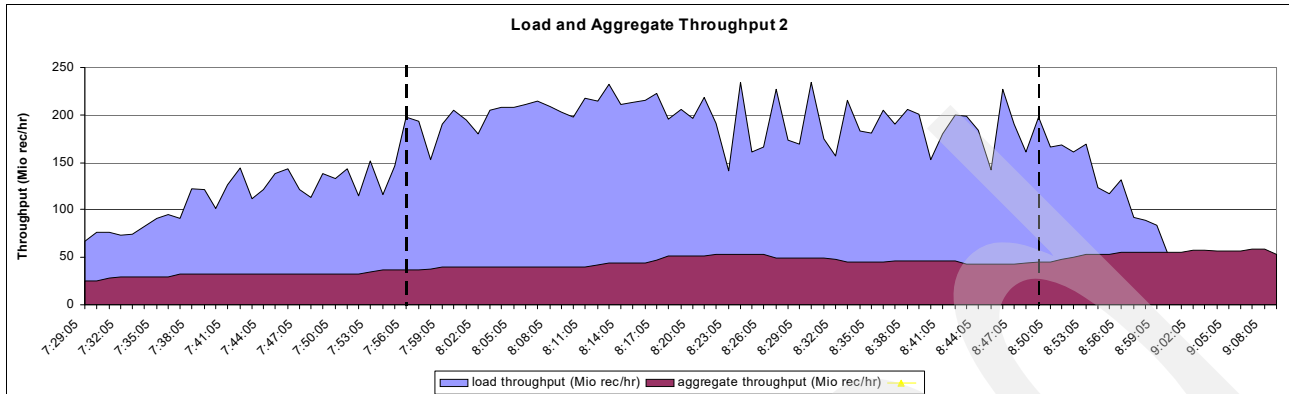


Figure 2-79 Load and aggregate throughput 2

The average throughput of the load during the high-load phase was 195.63 Mio rec/hr, while the throughput of the initial fill of the aggregates was 44.83 Mio rec/hr.

The CPU usage of the application servers involved with the load process (sys1ext0, sys1btc1, sys1btc2, sys1btc3, and sys1btc4) was high. With a MAXPROCS setting of 44, each extractor spawns 44 processes. Since we had eight extractors in parallel, there would be a total of 352 processes spread across the application servers. With four application servers (sys1btc1, sys1btc2, sys1btc3, and sys1btc4), each application server should use 88 processes. With SMT, each application server should only use 44 processes. In fact, this was not the case, as the application servers would use close to 48 processes, as shown in Figure 2-80, Figure 2-81 on page 115, Figure 2-82 on page 115, and Figure 2-83 on page 116.

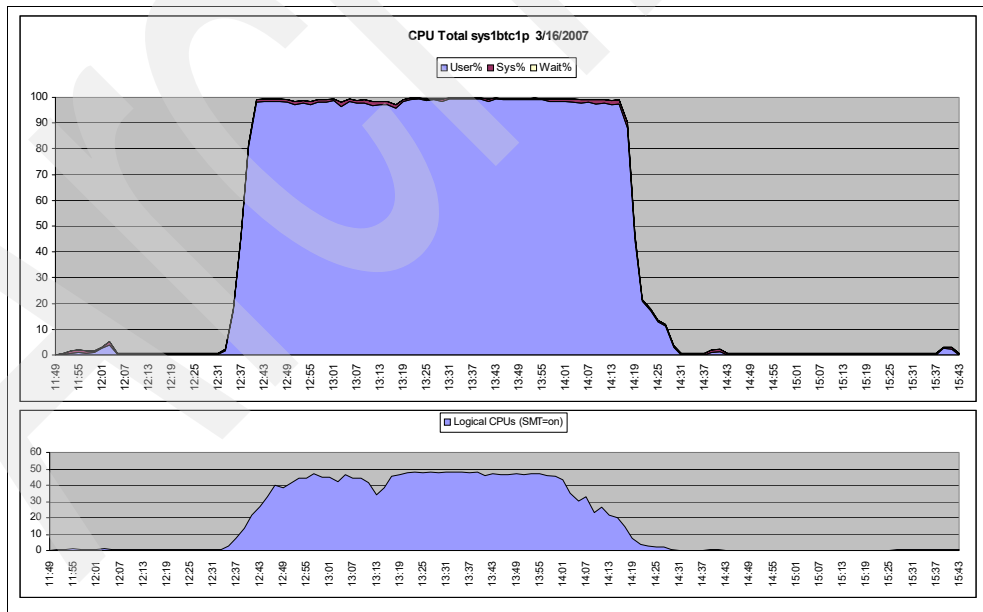


Figure 2-80 CPU usage of loading application servers - part 1

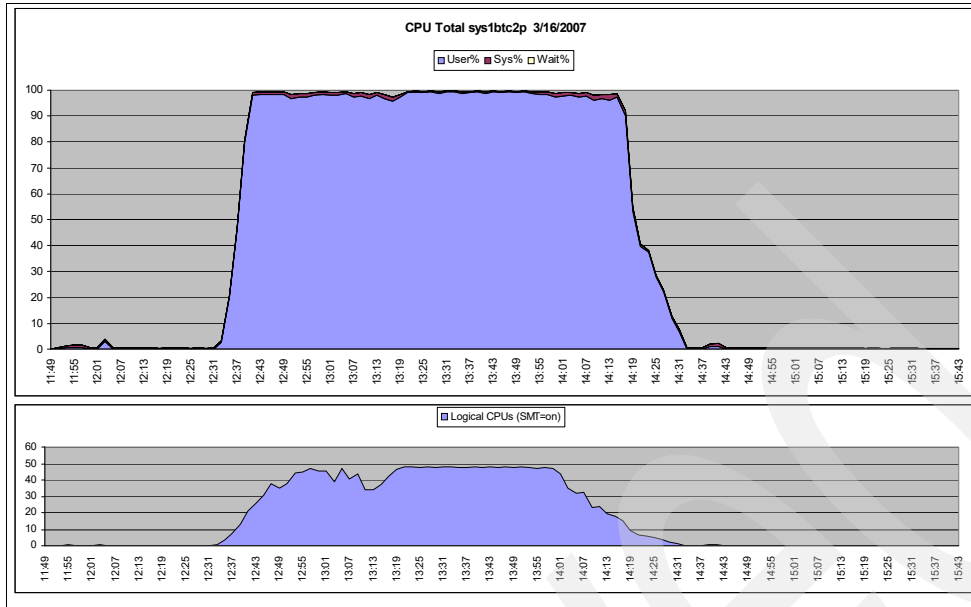


Figure 2-81 CPU usage of loading application servers - part 2

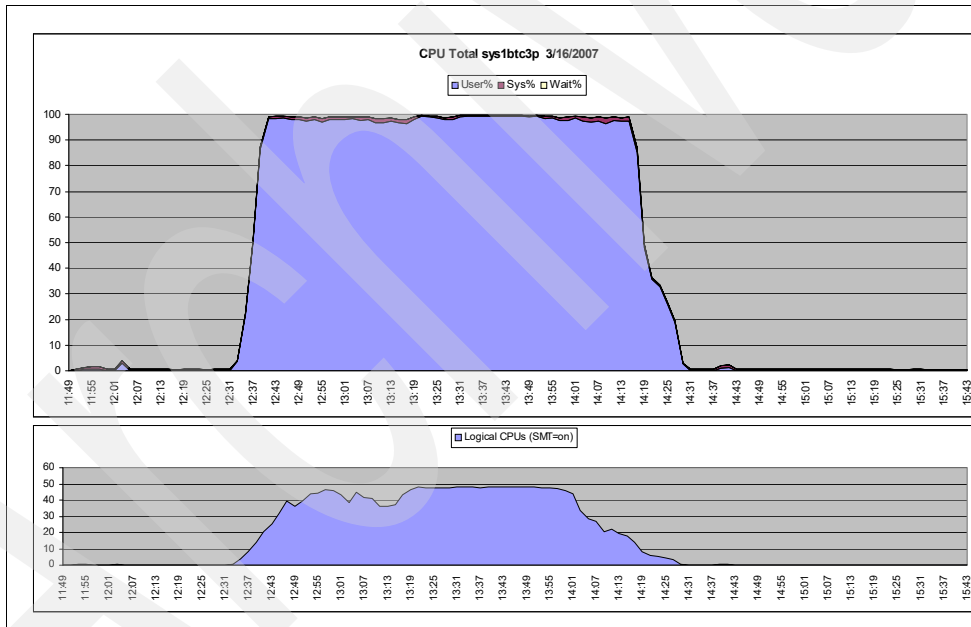


Figure 2-82 CPU usage of loading application servers - part 3

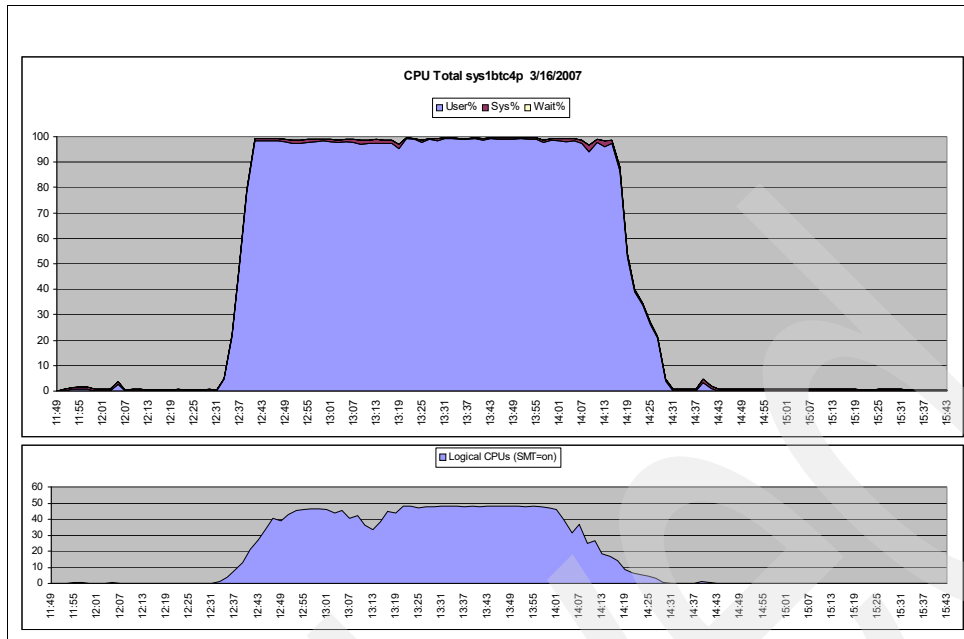


Figure 2-83 CPU usage of loading application servers - part 4

Therefore, with a throughput of 195.63 Mio rec/hr and approximately each application server using 48 CPUs, the throughput per CPU is 1.02 Mio rec/hr/CPU.

From the unit test of the aggregates, we achieved an average of 4.3 Mio rec/hr with a standard deviation of 0.29. Therefore, with 10 InfoCubes we estimated 43 Mio rec/hr plus or minus 2.9 Mio rec/hr. Comparing the results of the load and aggregate stress test, we see that we were very close to our estimate as we achieved 44.83 Mio rec/hr.

Summary: The impact of the load process on the aggregates is insignificant.

2.3.9 Impact of the load and the query processes running together

To prepare the combined stress test, we executed the load process along with the query process to determine the affect of the two processes running in parallel.

The load process settings used were the same as the settings in 2.3.8, “Impact of the load and the aggregate processes running together” on page 112. Likewise, the query process settings were the same in 2.3.7, “Impact of query and aggregate processes running in parallel” on page 106 settings.

We started the query process first, followed by the load process. The high load phase starts at 12:00 when ramp-up finishes. Figure 2-84 displays the load throughput and the query response time.

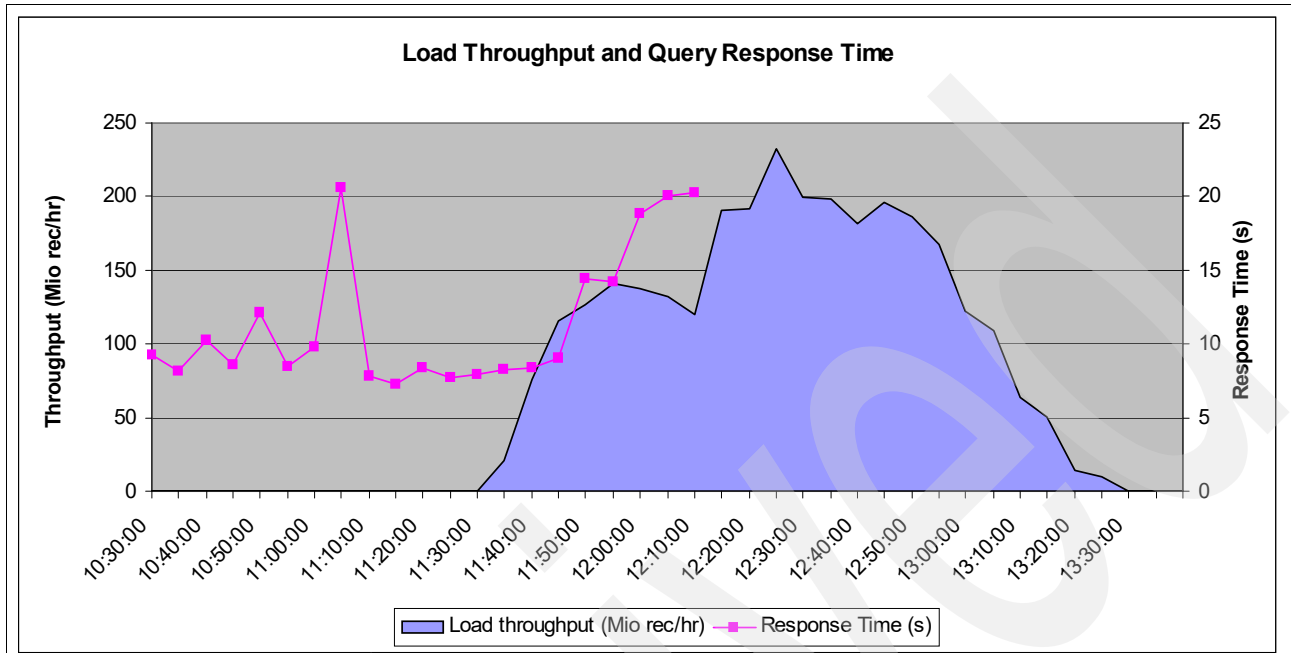


Figure 2-84 Load throughput and query response time

The average load throughput from when ramp-up finishes (12:00) until the time when the queries stop (12:15) was 129.7 Mio rec/hr. Interestingly, once the queries stop, the average throughput from then until ramp-down starts (12:50) is 196.78 Mio rec/hr. That is a 51.7% increase in throughput without queries. Therefore, the query process largely affects the load process.

Figure 2-85 displays the response time and transactions/sec for the query process. The average response time of the queries was calculated over several periods to determine the effects of the loading process. Before the loading starts, the OLAP cache needs to warm up, as outlined in the chart. After the OLAP cache has warmed up, the average response time is 8.6 seconds. Once the loading has started, but before it has finished ramping-up (high load phase), the average response time was 10.7 seconds. Finally, once the load has reached the high-load phase, the average response time increases to 19.6 seconds.

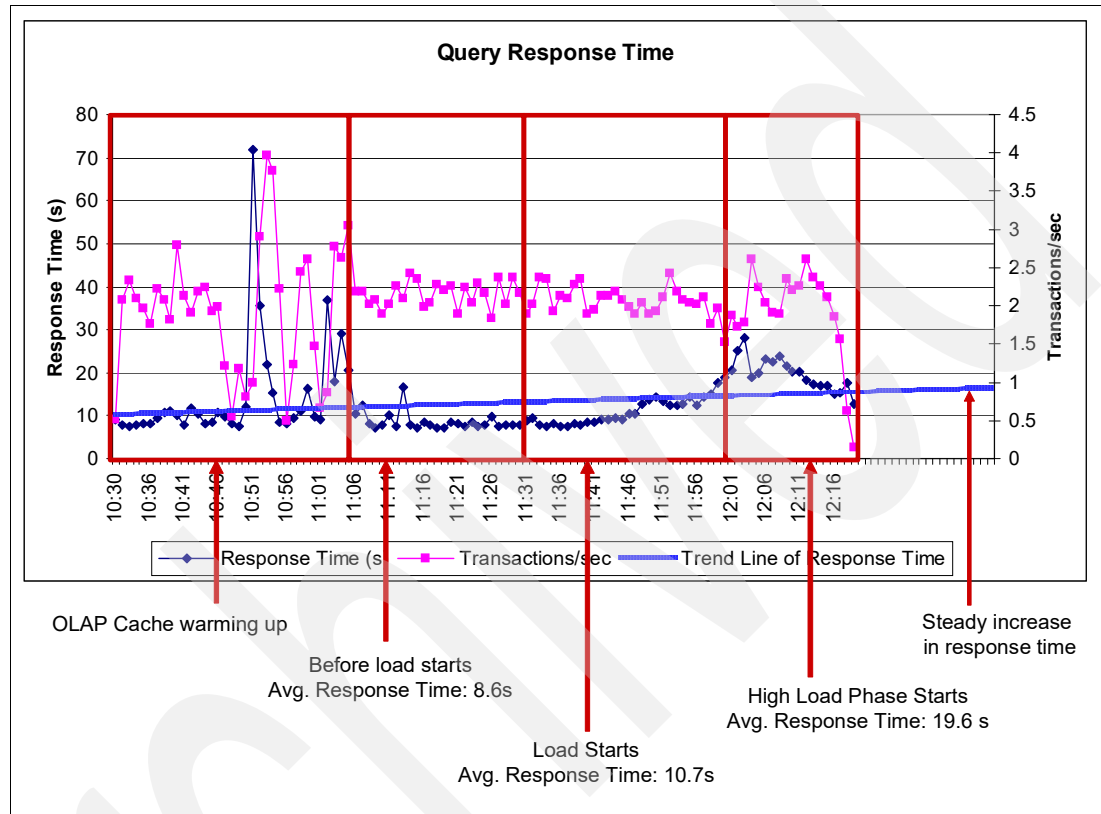


Figure 2-85 Query response time and transactions/sec

Comparing the average response time from before the load starts (8.6s) to the high-load phase (19.6s) shows an increase of 128%. Clearly, the loading process affects the query process significantly.

A trend line of the response time was included in the graph in Figure 2-85 as well. This trend line can be used as a forecast for when we run the combined stress test (load, aggregate, and query processes). The trend line shows a steady increase in the query response time as the load process starts. The trend line was extended several time periods to help with the forecast of the combined stress test. Therefore, we predict that the average query response time will stay under 20 seconds for the combined stress.

2.4 Results from an SAP perspective

Attention: Keeping in mind that customer SAP NetWeaver BI installations vary in many aspects, the KPI-G figures achieved in this test can be taken as performance indicators that can be transferred to other installations to a limited extent if the complexity of the tested scenarios is taken into account. Unlike the SAP Application Performance Standard (SAPS) values achieved by benchmarking SD-Applications in ERP, the throughput values obtained in this test cannot be considered standardized figures since they are specific for the implemented scenarios.

The KPI-G represents a combination of online and batch activities:

► Query throughput

The workload is similar to the customer environment. Eighty percent of the queries can use aggregates, while 20% are highly selective and access the InfoCubes' fact tables, and 50% of the queries can use the OLAP cache.

The objective is to achieve a throughput of 2.08 transactions per second with a response time under 20 seconds.

► InfoCube load

The objective is to write 125 million records per hour to any number of target InfoCubes.

► Aggregate rollup (initial fill)

The target is a throughput of 25 million records per hour.

2.4.1 Results summary

We ran the stress test with the settings learned in the previous section, 2.3, "Options and parameters discussions" on page 72.

The load process used four ODS sources, each with six target InfoCubes, and each ODS source used two extractors. MAXPROCS was set to 48 and MAXSIZE was 80,000 KB.

To parallelize the aggregate processes, we ran the initial fill of 10 InfoCubes' aggregates in parallel. Each InfoCube had 10 aggregates that were filled sequentially.

The query process was run as specified by the customer. That is, the workload consists of 80% of the queries that use aggregates. The remaining 20% access the fact table, simulating ad hoc queries and power users. Fifty percent of the queries were allowed to use the OLAP cache.

We start the query process first, allowing the OLAP cache to warm up. Approximately 1 hour later, the load process starts. The ramp-up takes approximately 30 minutes, after which the aggregate process should start.

Figure 2-86 shows the total throughput and response time of the stress test from the start of the queries until the end.

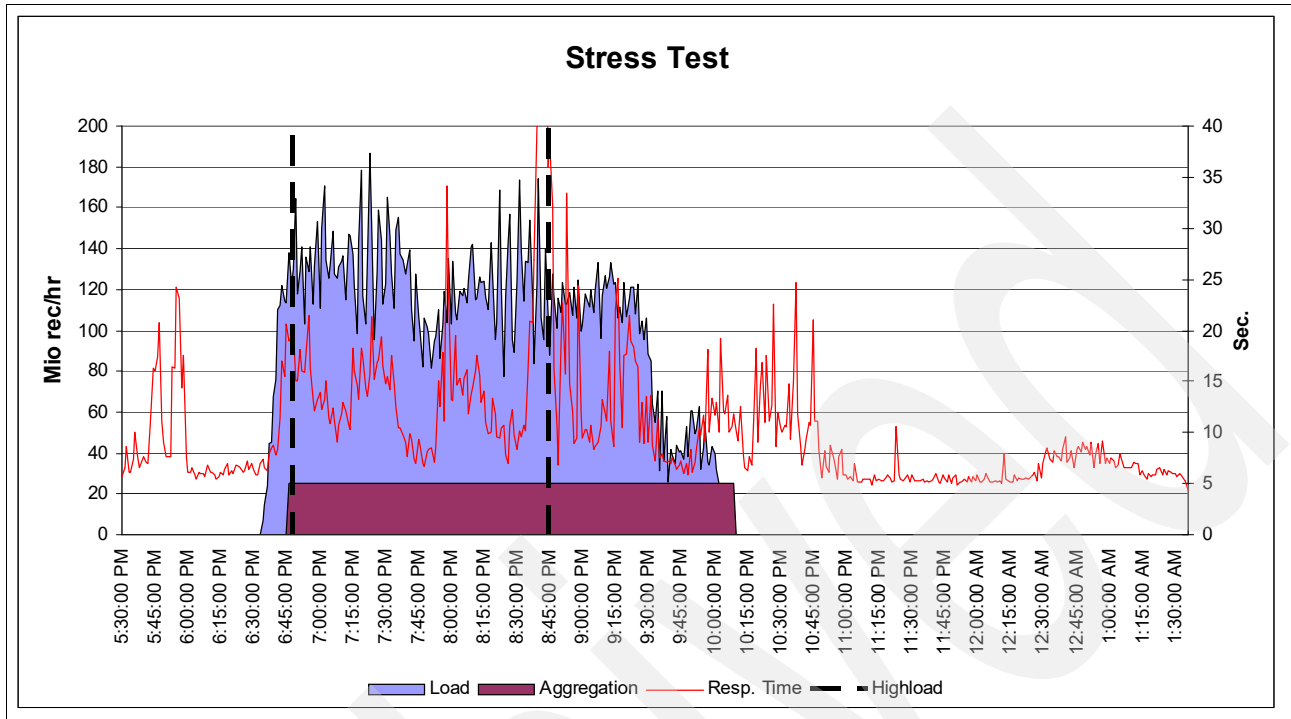


Figure 2-86 Stress test results

During the high load phase, the load achieved a throughput of 125.05 Mio rec/hr. The aggregates were filled with a throughput of 25.03 Mio rec/hr. The queries averaged 2.13 transactions/sec with an average response time of 16.09 seconds. Figure 2-87 shows the results during the high load phase only and is summarized in Table 2-9.

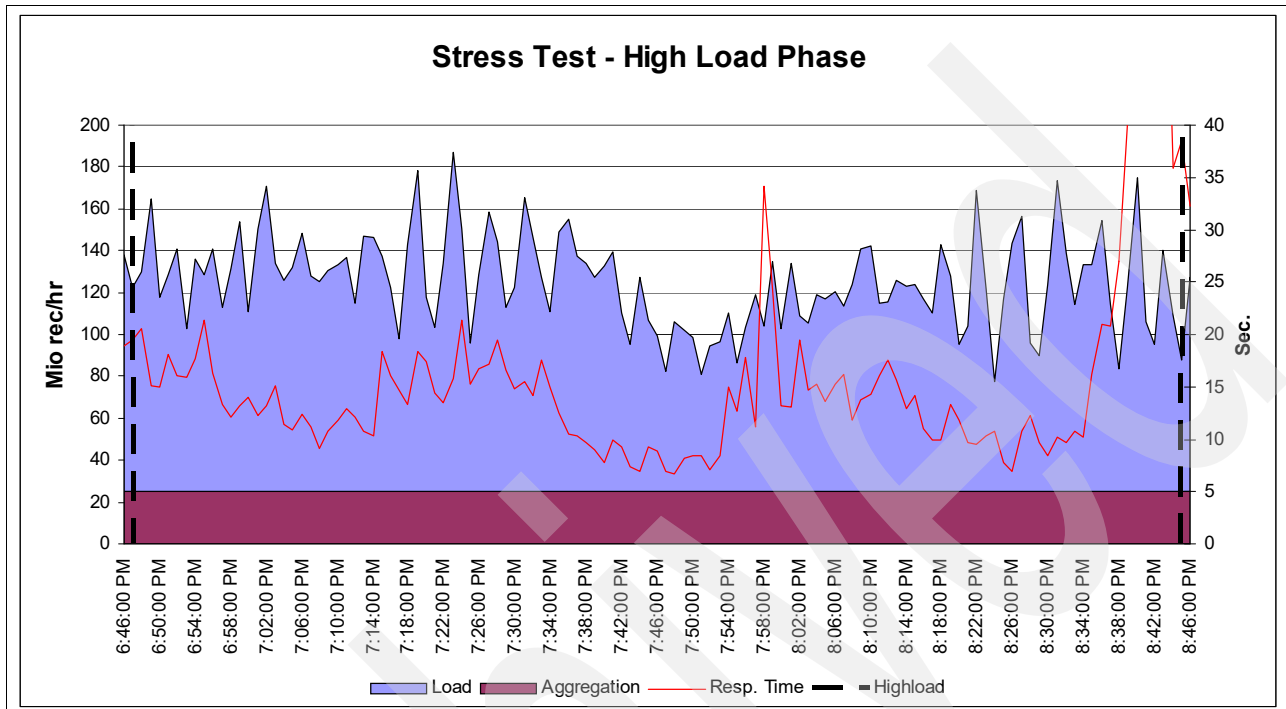


Figure 2-87 Stress test - high load phase

Table 2-9 KPI-G 60 TB results

| Test | Required | Achieved 60 TB |
|--------------------------|--------------------|-----------------------|
| Query | | |
| ▶ Transaction/sec | 2.08 | 2.13 |
| ▶ Average response times | < 20 sec | 16.09 |
| InfoCube load | 125 million rec/hr | 125.05 million rec/hr |
| Aggregate fill | 25 million rec/hr | 25.03 million rec/hr |

All three targets have been achieved.

- ▶ Table 2-10 provides more details regarding the results of the query scenario.
- ▶ Table 2-11 on page 122 provides more details regarding the aggregation scenario, its setup, and the resulting throughput.
- ▶ Table 2-12 on page 122 provides more details regarding the load scenario.

Table 2-10 Query scenario results summary

| Parameters | Result |
|---------------------------------|--------|
| High load start | 18:46 |
| High load end | 20:46 |
| Elapsed time of high load phase | 02:00 |

| Parameters | Result |
|--|--------|
| Total # of navigations during high load | 15,479 |
| % of workload STR_D queries during high load | 46% |
| Average steps/sec during high load | 2.13 |
| Average response time during high load | 16.09 |

During the high load phase, 46% of the workload was STR_D queries. Therefore, 46% of the workload was ad hoc, database-intensive queries and we still successfully achieved the results.

Table 2-11 Aggregate scenario results summary

| Parameters | Result |
|---|------------|
| Number of InfoCubes | 10 |
| Number of aggregates per InfoCube | 10 |
| Average throughput (rec written per hour) | 25,028,683 |

With 10 InfoCubes, a high query workload, we achieved the results set by the customer successfully.

Table 2-12 Load scenario results summary

| Parameters | Result |
|---|--------------------------|
| Job start time | 18:31 |
| Job end time | 22:05 |
| High load start | 18:46 |
| High load end | 20:46 |
| Elapsed time of high load phase | 02:00 |
| # of source ODS | 4 |
| # of extractors per ODS | 2 |
| # of InfoCubes per extractor | 6 |
| MAXSIZE | 80,000 KB |
| MAXPROCS | 48 |
| Total # of records inserted | 375,890,808 |
| Average throughput during high load phase | 125.05 million rec/hr |

We increased the MAXPROCS setting to 48. Although we achieved a high throughput, we have more room for improvement. Looking at Table 2-13 on page 123, we see that the application servers used for uploading (sys1btc1, sys1btc2, sys1btc3, and sys1btc4) were using, sometimes, all the CPUs (48). From the logs of the run, we saw that not all extractors were using 48 dialog processes. On average, each extractor used 44 dialog processes. Therefore, some dialog processes were waiting for CPUs. If we lower MAXPROCS to 44 for future runs, we should achieve a higher throughput, as all 44 can run without waiting for CPU.

2.4.2 Resources

Table 2-13 also provides the CPU usage of all the SAP application servers.

Table 2-13 CPU usage summary

| Application server | Physical CPU average | Physical CPU max | Virtual processors | Entitlement capacity |
|--------------------|----------------------|------------------|--------------------|----------------------|
| sys1ci | 0.21 | 0.52 | 4.00 | 8.00 |
| sys1onl0 | 10.92 | 14.73 | 32.00 | 64.00 |
| sys1ext0 | 2.98 | 8.51 | 16.00 | 32.00 |
| sys1btc0 | 0.60 | 2.90 | 42.00 | 84.00 |
| sys1btc1 | 27.88 | 48.00 | 48.00 | 96.00 |
| sys1btc2 | 27.85 | 48.01 | 48.00 | 96.00 |
| sys1btc3 | 27.42 | 47.99 | 48.00 | 96.00 |
| sys1btc4 | 27.76 | 47.95 | 48.00 | 96.00 |

Table 2-14 provides the memory usage of the SAP application servers.

Table 2-14 Memory usage summary

| Application servers | Average (MB) | Max (MB) | Physical memory (MB) |
|---------------------|--------------|-----------|----------------------|
| sys1ci | 9079.67 | 9215 | 12288.00 |
| sys1onl0 | 20500.84 | 26494.40 | 92160.00 |
| sys1ext0 | 19949.61 | 29047.80 | 98304.00 |
| sys1btc0 | 13951.46 | 22939.50 | 98304.00 |
| sys1btc1 | 40900.53 | 82291.10 | 98304.00 |
| sys1btc2 | 41119.20 | 75759.30 | 98304.00 |
| sys1btc3 | 41573.62 | 821440.20 | 98304.00 |
| sys1btc4 | 41049.58 | 81381.50 | 98304.00 |

2.5 Lessons learned

This section covers the experience gained and lessons learned when tuning the performance of the various workloads. Each subsequent section contains focuses on the different workload scenario and the recommendations and behavior observed during the tests.

2.5.1 The load process

The following observations and recommendations can be done regarding the load process:

- Number of extractors

Recommendation If possible, use multiple extractors. For example, two InfoPackages selecting different records, but loading to the same InfoCubes.

Background We ran two tests. One test had two ODS source, each with two extractors, therefore four extractors in total. The second test had four ODS sources, each with one extractor, therefore four extractors in total. The test with two extractors per ODS had a higher throughput and used less CPU than the test with one extractor per ODS.

With multiple extractors, each extractor spawns up to the MAXPROCS setting, therefore increasing the parallelization of the load process.

- ▶ Number of target InfoCubes

Recommendation If possible, have several target InfoCubes per InfoPackage.

Background As we increased the number of target InfoCubes per InfoPackage, we saw the throughput increase as well. Using several target InfoCubes, the extractor only needs to read once from the database and writes to several InfoCubes. Therefore, it lightens the load on the database, as less reading is necessary to write to several target InfoCubes.

- ▶ MAXSIZE setting

Recommendation This value varies depending on the infrastructure. For our environment, a MAXSIZE setting of 80,000 KB was most beneficial. For other environments, testing needs to be done to determine the optimal value.

Background We ran several tests, varying the MAXSIZE parameter. Comparing the throughput between 160,000 KB and 80,000 KB, we achieved a higher throughput with the 80,000 KB setting. Using a smaller MAXSIZE was not beneficial for the system either. The turn-around time for the data packages was too quick and the number of parallel work processes was never maximized. A very large setting of MAXSIZE would require more memory and cause our application servers to start paging.

- ▶ MAXPROCS setting

Recommendation This value depends on the infrastructure. For our environment, a MAXPROCS setting of 44 would give us the best throughput.

Background This setting allows each extractor to spawn dialog work processes up to the MAXPROCS value. With a setting of 44, this translates directly to the number of CPUs we had available for dialog processing. Each extractor would spawn 44 dialog work processes. We used a total of eight extractors, therefore a total of 352 dialog work processes would be running in parallel. We used four application servers, so each application server would use 88 dialog processes for the load scenario. Since we enabled SMT, each application server would use at least 44 CPUs. However, with a MAXPROCS setting of 44, we saw the CPU usage go as high as 48. Therefore, this setting is dependent on the environment.

During the tests, we also had to ensure that there were enough dialog processes available for the load scenario. That is, each application server had to be configured to allow over 88 dialog processes over RFC.

- ▶ Effects of the aggregate rollup and query processes

The initial fill of the aggregates had very little affect on the load process. With both scenarios running, we were still able to achieve a throughput of approximately 195 Mio rec/hr for the load and 44 Mio rec/hr for the aggregate fill.

The query scenario had a large impact on the load scenario. The load throughput decreased by 34.5% when running both in parallel.

When running the stress test, the load throughput decreased by 36.5%, that is, 125 Mio rec/hr as opposed to 197 Mio. rec/hr.

2.5.2 The aggregate rollup process

The following observations and recommendations can be done regarding the rollup process.

- ▶ Optimizing the rollup process

- Recommendation

Parallelize the aggregate rollup. With SAP NetWeaver BI 3.5, each of the aggregates that belongs to the same InfoCube is filled sequentially. Therefore, to parallelize this process, fill the aggregates belonging to different InfoCubes in parallel.

- Background

Since the InfoCubes' aggregates are filled sequentially, we ran several InfoCubes in parallel. That is, the process chain would start the filling of 10 InfoCubes' aggregates at the same time.

- Future recommendations

Switch to SAP BI 7.0. With the new version, aggregate-related maintenance jobs can be done in parallel. For example, unlike SAP NetWeaver BI 3.5, the initial filling of an aggregate can use several processes in parallel.

- ▶ Effects of the load and query processes

The load process has very little affect on the aggregate initial fill. The query process, however, has a significant affect. From the unit test of the aggregates, we were achieved an average throughput of 4.3 Mio rec/hr per InfoCube. When running just the query and aggregate scenarios, the aggregate throughput decreased by 14%.

2.5.3 The query process

The following recommendations and observations can be done regarding the query process:

- ▶ OLAP cache

Recommendation

Enable the OLAP cache. Using the OLAP cache reduces the workload on the database and improves query response time.

Background

From running the query scenarios, we see that the queries that use the OLAP cache have significantly quicker response times, as it does not have to read from the database again.

- ▶ OLAP cache modes

Recommendation

Different cache modes are recommended for different environments. See Figure 2-61 on page 101 for general recommendations.

Background

With different cache modes, there are advantages of using one mode over another, for example, with and without swapping cache modes. With swapping cache mode allows displaced data to be swapped into the cache. Without swapping cache mode, once data

is displaced, has to be re-read from the database again. For similar queries that are frequently accessed, with swapping is beneficial. However, for many ad hoc queries in the workload, the benefits of with swapping cache mode may not be realized.

- ▶ **Aggregates**

Recommendation Use aggregates to improve the response time of queries.

Background As aggregates contain highly summarized data compared to the InfoCubes, load on the database when reading from an aggregate is reduced. Fewer rows have to be scanned, thus improving the response time of the query.

However, having too many aggregates can also be detrimental. With more aggregates, more maintenance is needed. Having more aggregates means that when new data is loaded to the InfoCube, more data has to be rolled up to more aggregates. The maintenance cycle is longer with more rollup and change run processes.

- ▶ **Effects of the load and aggregation processes**

Both the load and aggregate rollup processes are affected by running the query process in parallel. Also, the query process is affected by both the load process and the aggregate process. When running just the load and query scenarios, the load throughput decreased by 34.5% and the query response time increased by 43%. When running the query and aggregate scenarios, the aggregate throughput decreased by 14% and the query response time increased by 24%.

2.6 DB2 9 features used by SAP

SAP uses many of the new features in DB2 9, such as:

- ▶ New deployment features
- ▶ Row compression
- ▶ Large RIDs
- ▶ Recovery enhancements

Although many of the features were used within the project, some features were not required, as they did not fit within the scope of this project. More information regarding these new features can be found in SAP Note 930487 DB6: Using DB2 9 with SAP Software, available at:

<https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/webcontent/uuid/e02ce76b-588c-2910-7dab-faa1e94ad012>

The features are detailed in 3.1.5, “SAP and DB2 9” on page 142.

The DB2 perspective

This chapter describes the database environment and the tests executed to achieve pre-established measurements and new features behavior.

In our tests, DB2 shared nothing architecture was demonstrated to be highly scalable from all perspectives, both the pure online load that database management systems must handle and infrastructure daily maintenance activities like backups, reorganization of data, and keeping index statistics up-to-date.

We executed a set of tests to stress the database design or infrastructure, therefore measuring specific results and analyzing, with the goal of providing you with a pattern and behavior of a small instance from the whole set of variables. Performance analysis is an empirical science, and by this foundation, each instance is unique. This does not mean that no pattern can be highlighted or extracted, since we use an empirical science to measure a logical science.

In this chapter we directed all our efforts to extract as much information possible from the database design to the stress test results so that you can make your own judgements based on facts. This is particularly important, since you must use your experience and knowledge together with our conclusions and facts to drive you.

3.1 Introducing DB2

We expect you to have a basic understanding of database concepts. This section highlights the DB2 architecture from a macro perspective. IBM DB2 UDB is composed of several objects. It relies on relational database concepts.

The following sections describe the IBM DB2 UDB concepts needed to understand our test scenarios. Some of the information provided here may vary in other scenarios, due to operating system particularities.

3.1.1 Instance

In IBM DB2 UDB, the *instance* is the highest entity in the structure. It is responsible for setting up communication parameters, Fast Communication Manager (FCM) memory parameters, authentication methods, authorization, and so on. The highest roles in DB2 UDB are configured at the instance level (for example, SYSADM, SYSCTRL, and SYSMANT).

On UNIX®, an instance must always be assigned to an operating system user, normally called the *instance owner*. This user and its primary group are automatically added on the SYSADM group by default. On UNIX and Linux systems, when you start the DB2 UDB processes, they are owned by the instance owner user identification (*user ID*), creating a pool of resources required by the databases.

The IBM DB2 UDB structure is also known as *shared-nothing*. This means that the data portion managed by one of the instances is not visible and cannot be accessed by any other instance in the multiple instance structure. This structure relies on a control instance, which is responsible for coordinating requests throughout the remaining instances.

Read 3.1.3, “Partitioned databases” on page 131, for further information about database partitioning.

3.1.2 Database

All the authorization levels that are related to the database level are granted or revoked as Data Definition Language (DDL) in the database. When a database is created, three basic tablespaces are created and four hidden system buffer pools are also created, one for each page size: 4 KB, 16 KB, 32 KB, and 64 KB. These buffer pools are very small and are only used when a tablespace with a specific page size does not match with any explicitly created buffer pool or any error or deferred memory allocation of the explicit buffer pool allocation.

At this database level, several performance parameters directly interfere with response time and optimization. A database can be configured to use *circular log* or *linear log*. When using circular log, logs are overwritten as needed and no online backup is allowed. When using linear log, a user exit method should also be configured to archive inactive logs.

In DB2 UDB 8.2.2 DB2 UDB Version 8 fix pack 9, a new clause was added to the database creation command that relates to *automatic storage*. This clause allows you to manage storage and space without interventions.

It is a best practice to create the database with the automatic storage feature enabled because no conversion or enablement of this feature on an existing database is supported at the time of writing. This feature is implicitly set to on.

A database is composed of several objects that may relate each other in a referential way. The main database objects are:

- ▶ Tablespaces
- ▶ Buffer pools
- ▶ Tables and indexes

Tablespaces

A tablespace is a logical aggregation of data *containers* or data *files*. These data containers or data files are allocated and assigned to a specific tablespace, which is responsible for balancing the data across containers by the use of tablespace maps and tablespace extent maps. These maps define stripes, which are a continuous blocks of extends spanning throughout containers.

Three basic tablespaces are created by default:

- ▶ The SYSCATSPACE tablespace holds system catalog information like definition of indexes, tables, buffer pools, and so on. These tablespaces and tables are unique for each database, meaning that there is no way to share information between databases without the use of federation or replication techniques.
- ▶ The TEMPSPACE tablespace is a system temporary tablespace used whenever persistent storage is needed, as in a table reorganization.
- ▶ The USERSPACE tablespace is a user tablespace that can be used to store user table data.

Although not created by default, another tablespace type that may also be required is the user temporary tablespace. This tablespace is required for applications that require the use of user temporary tables.

Tablespaces can be created with different page sizes: 4 KB, 16 KB, 32 KB, and 64 KB.

- ▶ When the page size is 4 KB, the row length can be up to 4,005 bytes.
- ▶ When the page size is 8 KB, the row length can be up to 8,101 bytes.
- ▶ When the page size is 16 KB, the row length can be up to 16 ,93 bytes.
- ▶ When the page size is 32 KB, the row length can be up to 32,677 bytes.

Having a larger page size facilitates a reduction in the number of levels in any index. If you are working with online transaction processing (OLTP) applications, which perform random row reads and writes, a smaller page size is better, because it wastes less buffer space with undesired rows. If you are working with decision support system (DSS) applications, which access large numbers of consecutive rows at a time, then using a larger page size is better, because it reduces the number of I/O requests required to read a specific number of rows.

An exception, however, is when the row size is smaller than the page size divided by 255. In such a case, there is wasted space on each page (there is a maximum of only 255 rows per page). To reduce this wasted space, a smaller page size may be more appropriate.

Buffer pools

A buffer pool is a an amount of memory used to cache table and index data pages as they are being read from disk or being modified. The buffer pool improves database system performance by allowing data to be accessed from memory instead of from disk. Because memory access is much faster than disk access, the less often the database manager needs to read from or write to a disk, the better is the performance. Because most data manipulation takes place in buffer pools, configuring buffer pools is the most important tuning area. Only large objects and long field data are not manipulated in a buffer pool.

When an application accesses a row of a table for the first time, the database manager places the page containing that row in the buffer pool. The next time any application requests data, the database manager looks for it in the buffer pool. If the requested data is in the buffer pool, it can be retrieved without disk access, resulting in faster performance.

Memory is allocated for the buffer pool when a database is activated or when the first application connects to the database. Buffer pools can also be created, dropped, and resized while the database manager is running.

If you use the IMMEDIATE keyword of the ALTER BUFFERPOOL statement to increase the size of the buffer pool, the memory is allocated as soon as you enter the command, if the memory is available. If the memory is not available, the change occurs when all applications are disconnected and the database is reactivated. If you decrease the size of the buffer pool, the memory is de-allocated at the committal time. When all applications are disconnected, the buffer pool memory is de-allocated.

To reduce the necessity of increasing the size of the DBHEAP database configuration parameter when the buffer pool sizes increase, nearly all buffer pool memory, which includes page descriptors, buffer pool descriptors, and the hash tables, comes out of the database shared memory set and is sized automatically.

To ensure that an appropriate buffer pool is available in all circumstances, DB2 creates small buffer pools, one with each page size: 4 KB, 8 KB, 16 KB, and 32 KB. The size of each buffer pool is 16 pages. These buffer pools are hidden from the user. They are not present in the system catalog or in the buffer pool system files. You cannot use or alter them directly, but DB2 uses these buffer pools in the following circumstances:

- ▶ When a buffer pool of the required page size is inactive because insufficient memory was available (for example, after a CREATE BUFFERPOOL statement was executed with the IMMEDIATE keyword). A message is written to the administration notification log. If necessary, tablespaces are re-mapped to a hidden buffer pool. Performance might be drastically negatively impacted.
- ▶ When the ordinary buffer pools cannot be brought up during a database connect. This problem is likely to have a serious cause, such as an out-of-memory condition. Although DB2 will be fully functional because of the hidden buffer pools, the performances can be largely deteriorated. A message is written to the administration notification log and you should address this problem immediately.

Pages remain in the buffer pool until the database is shut down, or until the space occupied by a page is required for another page. The following criteria determine which page is removed to bring in another page:

- ▶ How recently the page was referenced.
- ▶ The probability that the page will be referenced again by the last agent that looked at it.
- ▶ The type of data on the page.
- ▶ Whether the page was changed in memory but not written out to disk (changed pages are always written to disk before being overwritten).

Tables and indexes

A table consists of data logically arranged in columns and rows. All database and table data is assigned to tablespaces. Table data is accessed using the Structured Query Language (SQL). When creating a table, you can decide to store all related objects (for example, indexes and large object data) in the same tablespace, or keep them in separate tablespaces.

A referential constraint may be defined in such a way that either the parent table or the dependent table is a part of a table hierarchy. In such a case, the effect of the referential constraint is as follows:

► Effects of INSERT, UPDATE, and DELETE statements

If a referential constraint exists, in which PT indicates a *parent table* and DT indicates a *dependent table*, the constraint ensures that for each row of DT (or any of its sub-tables) that has a non-null foreign key, a row exists in PT (or one of its subtables) with a matching parent key. This rule is enforced against any action that affects a row of PT or DT, regardless of how that action is initiated.

► Effects of DROP TABLE statements

- For referential constraints in which the dropped table is the parent table or dependent table, the constraint is dropped.
- For referential constraints in which a supertable of the dropped table is the parent table, the rows of the dropped table are considered to be deleted from the supertable. The referential constraint is checked and its delete rule is invoked for each of the deleted rows.
- For referential constraints in which a supertable of the dropped table is the dependent table, the constraint is not checked. Deletion of a row from a dependent table cannot result in a violation of a referential constraint.

When any table is created, the definer of the table is granted CONTROL privilege. When a sub-table is created, the SELECT privilege that each user or group has on the immediate supertable is automatically granted on the subtable with the table definer as the grantor.

3.1.3 Partitioned databases

A partitioned database environment is a database that is made up of two or more database *partitions*. In this type of database, data is hashed for storage. A database partition consists of its own data, indexes, configuration files, and transaction logs. A database partition is sometimes called a *node* or a *database node*.

Tables can be located in one or more database partitions. When a table is distributed across multiple partitions, some of its rows are stored in one partition, and other rows are stored in other partitions. Data retrieval and update requests are decomposed automatically into sub-requests, and executed in parallel among the applicable database partitions. The fact that databases are split across database partitions is transparent to users.

Database administrators can choose how to distribute their data by declaring *distribution keys*. They can determine which and how many database partitions the table data can be spread across by selecting the database partition group and the tablespace in which the data should be stored. In addition, a *distribution map* specifies the mapping of distribution key values to database partitions. This makes it possible for flexible workload parallelization across a partitioned database for large tables, while allowing smaller tables to be stored on one or a small number of database partitions if this is the choice of the application designer. Each local database partition may have local indexes to provide high performance local data access.

Distribution key considerations

A distribution key is a column (or a group of columns) that is used to determine the database partition in which a particular row of data is stored. A distribution key is defined on a table using the CREATE TABLE statement. If a distribution key is not defined for a table in a tablespace that is divided across more than one database partition in a database partition group, one distribution key is created by default from the first column of the primary key. If no

primary key is specified, the default distribution key is the first non-long field column defined on that table (a long field column includes all long data types and all large object (LOB) data types). If you are creating a table in a tablespace associated with a single-partition database partition group, and you want to have a distribution key, you must define the distribution key explicitly. One is not created by default.

The process of choosing columns of the table to be the distribution key is very important, since it will be responsible for keeping data balanced across partitions. In our scalability test, partitioning keys are already chosen by SAP.

DB2 9 also supports *range partitioning*, which is based on the partitioning of the table based on a range of values. This is very useful when you need to spread data across tablespaces or even partitions, by a specific range. More details about range partitioning can be found in “Range partitioning” on page 139.

Backing up partitioned database

To execute an offline backup in a partitioned database environment, you should back up the catalog partition separately from all other database partitions. For example, you can back up the catalog partition first, then back up the other database partitions. This is necessary because the backup operation may require an exclusive database connection on the catalog partition, during which the other database partitions cannot connect. If you are performing an online backup, all database partitions (including the catalog partition) can be backed up simultaneously or in any order.

Users and groups

All partitions must run under the same user and group, whether running on the same physical server or not. If your database goes across different physical servers, the instance owner user and any related group or user used for security roles like SYSADMIN, SYSMANT, or entitled authorized users must be created with the same names and IDs as specified in the concentrator server.

The home directory should be shared across the remaining participating servers. In UNIX servers, you should share the instance’s home directory by using an NFS export. It is important to share all directories used for instance control purposes as well.

Configuration considerations

Partitions must be configured in `db2nodes.cfg`, as depicted in Example 3-1.

Example 3-1 db2nodes.cfg with five physical servers and 38 partitions

```
0 sys2db0p 0
1 sys2db0p 1
2 sys2db0p 2
3 sys2db0p 3
4 sys2db0p 4
5 sys2db0p 5
6 sys2db1p 0
7 sys2db1p 1
8 sys2db1p 2
9 sys2db1p 3
10 sys2db1p 4
11 sys2db1p 5
12 sys2db1p 6
13 sys2db1p 7
14 sys2db2p 0
15 sys2db2p 1
```

```

16 sys2db2p 2
17 sys2db2p 3
18 sys2db2p 4
19 sys2db2p 5
20 sys2db2p 6
21 sys2db2p 7
22 sys2db3p 0
23 sys2db3p 1
24 sys2db3p 2
25 sys2db3p 3
26 sys2db3p 4
27 sys2db3p 5
28 sys2db3p 6
29 sys2db3p 7
30 sys2db4p 0
31 sys2db4p 1
32 sys2db4p 2
33 sys2db4p 3
34 sys2db4p 4
35 sys2db4p 5
36 sys2db4p 6
37 sys2db4p 7

```

This configuration is for a partitioned database structure with 38 partitions with logical and physical partitioning. The number of partitions and server names is depicted in Table 3-1.

Parse db2nodes.cfg, starting from the first field at the left. The second field is for server names, which in our case are sys2db0,sys2db1,sys2db2, sys2db3, and sys2db4. The last field relates to the partition communication port and must match service port entries in the services on each physical server, as depicted in Example 3-2.

Table 3-1 Host names and partitions

| Host name | Number of partitions |
|-----------|----------------------|
| sys2db0 | 6 |
| sys2db1 | 8 |
| sys2db2 | 8 |
| sys2db3 | 8 |
| sys2db4 | 8 |

The ports must be reserved in the /etc/services file, as shown in Example 3-2, to enable the FCM communication between the instances. By default, DB2 reserves the first four ports starting at 60,000 when you create an instance.

Example 3-2 Ports used for FCM and allocated in the /etc/services file

```

DB2_db2eb8      5934/tcp
DB2_db2eb8_1    5935/tcp
DB2_db2eb8_2    5936/tcp
DB2_db2eb8_3    5937/tcp
DB2_db2eb8_4    5938/tcp
DB2_db2eb8_5    5939/tcp
DB2_db2eb8_6    5940/tcp

```

| | |
|----------------|----------|
| DB2_db2eb8_7 | 5941/tcp |
| DB2_db2eb8_8 | 5942/tcp |
| DB2_db2eb8_9 | 5943/tcp |
| DB2_db2eb8_10 | 5944/tcp |
| DB2_db2eb8_11 | 5945/tcp |
| DB2_db2eb8_12 | 5946/tcp |
| DB2_db2eb8_13 | 5947/tcp |
| DB2_db2eb8_14 | 5948/tcp |
| DB2_db2eb8_15 | 5949/tcp |
| DB2_db2eb8_16 | 5950/tcp |
| DB2_db2eb8_17 | 5951/tcp |
| DB2_db2eb8_18 | 5952/tcp |
| DB2_db2eb8_19 | 5953/tcp |
| DB2_db2eb8_20 | 5954/tcp |
| DB2_db2eb8_21 | 5955/tcp |
| DB2_db2eb8_22 | 5956/tcp |
| DB2_db2eb8_23 | 5957/tcp |
| DB2_db2eb8_24 | 5958/tcp |
| DB2_db2eb8_25 | 5959/tcp |
| DB2_db2eb8_26 | 5960/tcp |
| DB2_db2eb8_27 | 5961/tcp |
| DB2_db2eb8_28 | 5962/tcp |
| DB2_db2eb8_29 | 5963/tcp |
| DB2_db2eb8_30 | 5964/tcp |
| DB2_db2eb8_31 | 5965/tcp |
| DB2_db2eb8_32 | 5966/tcp |
| DB2_db2eb8_33 | 5967/tcp |
| DB2_db2eb8_34 | 5968/tcp |
| DB2_db2eb8_35 | 5969/tcp |
| DB2_db2eb8_36 | 5970/tcp |
| DB2_db2eb8_37 | 5971/tcp |
| DB2_db2eb8_END | 5972/tcp |

It is very important to understand the difference between these ports and the *communication port* configured in the SVCENAME instance parameter.

In our scenario, with the IBM DB2 version and level, remote shell was used to enable the communication between the servers, but ssh trust based on keys would also be an option. Prior to IBM DB2 V8.2 fix pack 3, the only method supported was by the use of remote shell (rsh) (in the case of ssh adoption, the generation of ssh keys and the addition of the public key into a file called `authorized_keys`). The ssh key is generated by the command `ssh-keygen` and the detailed process is also described in the IBM DB2 InfoCenter.

When configuring DPF on multiple distinct servers or LPARs, one additional service must also be considered. Since any relational database management system relies on transactions and generation of logs, the time stamps must be as accurate as possible. To achieve that, a time synchronization service must be configured and normally requests the use of network time protocol (NTP). In our scenario, this was accomplished by the use of the XNTPD daemon. To check whether you have XNTPD running you can use the `!ssrc` command. In the Peer field of the `!ssrc -!s xntpd` command you can check what is the time server configured. This may also be achieved by checking the server parameter in the `/etc/ntp.conf` file.

3.1.4 DB2 9

DB2 9 jumps into a new management database system layer, as an hybrid database system, capable of storing relational and XML data, integrating this information on the fly. It creates a new line of database servers and introduces new features. We briefly describe the main features and further look at the enhancements that directly affect our tests. The major enhancements in DB2 9 include:

- ▶ Self-tuning memory for reducing storage
- ▶ Row compression
- ▶ Autonomic computing for easier management
- ▶ Label access security (LBAC) for extended data protection
- ▶ New table partitioning improving flexibility and scalability
- ▶ pureXML™
- ▶ Optimizations for SAP

Self-tuning memory manager

Workload in database servers varies accordingly to several factors, which can change dynamically and can be tightly related to the environment itself, the application nature, a specific period of time, and so on. DB2 9 introduces a new feature that allows easier memory tuning, keeping track dynamically of system behavior, and changing memory usage and utilization on the fly.

Self-tuning memory manager (STMM) relies on embedded intelligence to check and adapt specific database memory areas as needed. For example, if a buffer pool needs more memory and there is some memory available (locking or sort memory), it can dynamically move this memory to the buffer pool in order to achieve the best performance.

When you enable the STMM feature inside DB2 9, it automatically distributes the memory usage of:

- ▶ Buffer pools
- ▶ Package cache
- ▶ Locking memory
- ▶ Sort memory
- ▶ Total database shared memory

The enablement is fairly simple through a database variable called `self_tuning_mem`. This variable is by default enabled for non-partitioned databases. You can also choose to set each memory pool to automatic.

For further information about STMM, refer to 3.4.7, “STMM and DPF (SAP NetWeaver BI)” on page 280.

Row compression

DB2 9 adds a new feature to compress row data based on an algorithm and a static compression dictionary. Duplicated data are replaced by numeric keys and a dictionary is built based on this mapping. This dictionary is cached (it requests about 100 KB of memory). It is stored along with the table data rows in the data object portions of the table.

Although data row compression’s main role is to save storage, it can help to improve performance since buffer pool fetched data is compressed, allowing more data to be fetched from the buffer pool, reducing I/O utilization. This feature is very sensitive to the environment. Factors like storage access speed, large tables, or specific application characteristics may influence the results of compression. Of course, there is an associated cost in extra CPU to uncompress and compress data. In our environment we executed some tests with data row

compression. For further information and results go to 3.4.4, “Stress test comparison with and without compression” on page 247.

Savings in storage and size are propagated to any other activity that is sensitive of table size. For example, after compressing a table, the reorganization may be faster due to the reduced volume of I/O. The statistics update for a compressed table may face some degradation since it is based on a full table scan. On the other hand, for indexes and detailed index statistics, no overhead or impact should be seen. It is important to highlight that for detailed index statistics, a higher CPU usage is expected due to the update of CLUSTERFACTOR and PAGE_FETCH_PAIRS statistics.

Statistics considerations

PAGE_FETCH_PAIRS statistics contain pairs of numbers that model the number of I/Os required to read data pages into buffer pools of various sizes together with CLUSTERFACTOR information, and they can be used by the optimizer, if available, when the value of the CLUSTERRATIO column is -1.

The DETAILED statistics provide concise information about the number of physical I/Os required, accessing the data pages of a table if a complete index scan is performed under different buffer sizes. As RUNSTATS scans the pages of the index, it models the different buffer sizes, and gathers estimates of how often a page fault occurs. For example, if only one buffer page is available, new pages referenced by the index will result in a page fault. In the worse case, each row might reference a different page, resulting in, at most, the same number of I/Os as rows in the indexed table. At the other extreme, when the buffer is big enough to hold the entire table, all table pages are read once. As a result, the number of physical I/Os is a uniform, non-increasing function of the buffer size.

The statistical information also provides finer estimates of the degree of clustering of the table rows to the index order. The less the table rows are clustered in relation to the index, the more I/Os are required to access table rows through the index. The optimizer considers both the buffer size and the degree of clustering when it estimates the cost of accessing a table through an index.

You should collect DETAILED index statistics when queries reference columns that are not included in the index. In addition, DETAILED index statistics should be used in the following circumstances:

- ▶ The table has multiple un-clustered indexes with varying degrees of clustering.
- ▶ The degree of clustering is non-uniform among the key values.
- ▶ The values in the index are updated non-uniformly.

We executed reorganization and statistics update on both compressed and uncompressed tables, and some figures can be found in 3.4.3, “REORG and RUNSTATS comparison with and without compression” on page 235.

Autonomic computing

DB2 Version 8 already had some new features related to autonomic computing. The main idea of autonomic computing is to create a system that is intelligent enough to overcome errors by dynamically changing parameters to minimize impacts. The idea behind DB2 autonomic computing is to improve database service by allowing database administrators to focus on what is more valuable, reducing costs with simplicity of use.

DB2 autonomic features can help businesses, from small companies to large companies. One example is the feature added on DB2 V8.2.2 and on the base level of DB2 9: the auto resize option for DMS tablespaces. This can be valuable for tablespaces in small, medium, or large databases. In our scenario we used this feature.

Some precautions must be taken when you enable the auto resize feature:

- ▶ Ensure that you create the database with auto resize on. In DB2 9, it is by default enabled.
- ▶ Chose the increment size to match storage recommendations.
- ▶ Be sure that maxsize is properly configured to not run out of space in the file system or partition.
- ▶ Evaluate space and maxsize if the file system or partition is shared by more than one tablespace on which you plan to turn auto resize on.

Note: At the time this book was written, and for our tests, there is no support available for turning this feature on databases that were created without the auto resize feature at creation time. Talk to your IBM representative for the latest information.

Automatic runstats can improve database performance and reduce database administrator workload. By enabling this feature, DB2 evaluates statistics for tables and indexes and, if a statistics update must be executed, it best accommodates the task based on a defined time frame. This task can run on a maintenance window that you provide according to your environment and business needs. For example, if you can afford having an online maintenance window every day from 01:00 a.m. to 04:00 a.m. and an offline maintenance window on Sunday from 01:00 a.m. to 02:00 p.m., DB2 will execute the tasks during this interval. You can also automate reorgs and backups to be executed this way.

The benefits of new autonomic features go beyond the scope of this book. For further information about them refer to the DB2 9 InfoCenter:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp>

Label access security

Label access security enables great flexibility when controlling access to database tables, improving granularity and security control.

Label-based access control (LBAC) greatly increases the control that you have over anyone who can access your data. LBAC lets you decide exactly who has write access and who has read access to individual rows and individual columns.

The LBAC capability is very configurable and can be tailored to match your particular security environment. All LBAC configuration is performed by a security administrator, which is a user who has been granted the SECADM authority by the system administrator.

Attention: SECADM privileges the holder the ability to configure many things related to security of the database, and also to transfer ownership of database objects. For instance, all objects that are part of the label-based access control (LBAC) feature can be created, dropped, granted, or revoked by an user that holds SECADM authority. SECADM-specific abilities cannot be exercised by any other authority, not even SYSADM.

A security administrator configures the LBAC system by creating security policies and defining which security label components will be created. A *security policy* describes the criteria that will be used to decide who has access to what data. Only one security policy can be used to protect one table, but different tables can be protected by different security policies.

A *security label* component must be created with the following characteristics, according to your environment and business needs:

- ▶ **Tree:** if your security labels need to have a hierarchical structure based on nodes with roots and leaves, like a data flow of a company department or hierarchical employees positions, where you can have one or more entries below each node.
- ▶ **Array:** if your security labels have a sequential hierarchy, like a chain where sequential authorization levels may apply.
- ▶ **Set:** If your security labels does not depend of an hierarchy, as a standalone label for each entry, in this case order does not matter.

After creating the security label component and the security policy, a security administrator creates objects, the security labels, that are part of that policy. Exactly what makes up a security label is determined by the security label component and security policy and can be configured to represent the criteria or policy that you use to decide who should have access to particular data items. If you decide, for instance, that you want to look at a person's position in the company and what projects they are part of to decide what data they should see, then you can configure your security labels so that each label can include that information. LBAC is flexible enough to let you set up anything from very complicated criteria to a very simple system where each label represents either a high or a low level of trust.

Once created, a security label can be associated with individual columns and rows in a table to protect the data. Data, which is protected by a security label, is called *protected data*. A security administrator allows users access to protected data by granting them security labels. When a user tries to access protected data, that user's security label is compared to the security label protecting the data. The protecting label will block some security labels and not block others.

A user is allowed to hold security labels for multiple security policies at once. For any given security policy, however, a user can hold at most one label for read access and one label for write access.

A security administrator can also grant exemptions to users. An exemption allows you to access protected data that your security labels might otherwise prevent you from accessing. Together your security labels and exemptions are called your LBAC credentials.

If you try to access a protected column that your LBAC credentials do not allow you to access, then the access will fail and you will get an error message.

If you try to read protected rows that your LBAC credentials do not allow you to read, then DB2 acts as though those rows do not exist. Those rows cannot be selected as part of any SQL statement that you run, including SELECT, UPDATE, or DELETE. Even the aggregate functions ignore rows that your LBAC credentials do not allow you to read. The COUNT(*) function, for example, returns a count of only the rows that you have read access to.

LBAC and views

You can define a view on a protected table the same in which way you can define one on a non-protected table. When such a view is accessed, the LBAC protection on the underlying table is enforced. The LBAC credentials used are those of the session authorization ID. Two users accessing the same view might see different rows, depending on their LBAC credentials.

LBAC cannot be used to protect the following types of tables:

- ▶ A materialized query table (MQT)
- ▶ A table that a materialized query table (MQT) depends on
- ▶ A staging table
- ▶ A table that a staging table depends on
- ▶ A typed table
- ▶ LBAC protection cannot be applied to a nickname.

For further information about LBAC, read the DB2 9 InfoCenter:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/c0021485.htm>

Range partitioning

Table partitioning is a data organization schema in which table data is divided across multiple storage objects called data partitions or *ranges* according to values in one or more table columns. Each data partition is stored separately. These storage objects can be in different tablespaces, in the same tablespace, or in a combination of both.

This feature should not be confused with database partitioning, where the data is split over partitions, therefore having a different result. Although these are different approaches, they can be used together.

Storage objects behave like individual tables, making it easy to accomplish fast roll-in by incorporating an existing table into a partitioned table using the ALTER TABLE using the ATTACH clause. Likewise, easy roll-out is accomplished with the ALTER TABLE using the DETACH clause. Query processing can also take advantage of the separation of the data to avoid scanning irrelevant data, resulting in better query performance for many data warehouse style queries.

Table data is partitioned as specified by the PARTITION BY clause of the CREATE TABLE statement. The columns used in this definition are referred to as the table partitioning key columns.

This organization schema can be used in isolation or in combination with other organization schemes. By combining the DISTRIBUTE BY and PARTITION BY clauses of the CREATE TABLE statement, data can be spread across database partitions spanning multiple tablespaces, enabling partitioning in two different layers. The DB2 organization schemes include the following parameters:

- ▶ DISTRIBUTE BY HASH, for the data partitioning feature
- ▶ PARTITION BY RANGE, for the range partitioning feature (DB2 9 only)
- ▶ ORGANIZE BY DIMENSIONS, for MDC clustering and partitioning

The benefits of table partitioning in DB2 9 include:

- ▶ Improved manageability for large tables: DB2 9 allows the various data partitions to be administered independently. For example, you can choose to back up and restore individual data partitions instead of entire tables. This lets you break down time-consuming maintenance operations into a series of smaller operations.
- ▶ Increased query performance through partition elimination: The DB2 optimizer is data partition aware. Therefore, during a query execution, only the relevant data partitions are scanned. Eliminating the need to scan data partitions that are not impacted by the query can result in improved performance.
- ▶ Fast online data roll-in/roll-out: DB2 9 allows data partitions to be easily added or removed from the table without having to take the database offline. This ability can be particularly

useful in a data warehouse environment where you often need to load or delete data to run decision-support queries. For example, a typical insurance data warehouse may have three years of claims history. As each month is loaded and rolled-in into the warehouse, the oldest month can be archived and removed (rolled-out) from the active table. This method of rolling out data partitions is also more efficient, as it does not need to log delete operations, which would be the case when deleting specific data ranges.

- ▶ Better optimization of storage costs: Table partitioning in DB2 9 lets you integrate better with hierarchical storage models. By only using your fastest and most expensive storage hardware for only the most active data partitions, DB2 9 allows you to optimize your overall storage costs and improve performance. If most of your queries only run against the last three months of data, you have the option to assign slower and less expensive storage hardware to older data.
- ▶ Larger table capacity: Without partitioning, there are limits on the maximum amount of data a storage object, and hence a table, can hold. However, by dividing the contents of the table into multiple storage objects or data partitions, each capable of supporting as much data as in a non-partitioned table, you can effectively create databases that are virtually unlimited in size.tablespace
- ▶ Greater index placement flexibility: DB2 9 allows indexes for partitioned tables to be stored in their own storage objects (tablespaces), as opposed to being in the same storage object as the non-partitioned table. This index placement flexibility is particularly useful for performing faster index operations (such as drop index, online index create, and index reorganization), managing table growth, and reducing I/O contention, providing more efficient concurrent access to the index data for the table.

pureXML

One of the newest features added in DB2 9 is pureXML, which allows you to have one data store for hybrid relational-hierarchical data transparently to the user. This improves DB2 capability to deliver a resilient and scalable data source used in service-oriented architecture (SOA) solutions.

In contrast to XML Extender, the native XML data store enables well-formed XML documents to be stored in their hierarchical form within columns of a table. XML columns are defined with the XML data type. By storing XML data in XML columns, the data is kept in its native hierarchical form, rather than stored as text or mapped to a different data model.

Because the native XML data store is fully integrated into the DB2 database system, the stored XML data can be accessed and managed by leveraging DB2 functionality.

The storage of XML data in its native hierarchical form enables efficient search and retrieval of XML. XQuery, SQL, or a combination of both can be used to query XML data. SQL functions that return XML data or take XML arguments (referred to as SQL/XML functions) also enable XML data to be constructed or published from values retrieved from the database.

XML data is inherently hierarchical, and can be naturally represented in a tree form using nodes having parent, child, and sibling relationships. With DB2 9, collections of XML documents are captured in tables that contain one or more columns based on a new XML data type. Unlike XML data types offered by some of the other RDBMs (that under the covers transform XML data and store using relational constructs), DB2 offers an XML data type that stores parsed XML documents and fragments with node-level granularity preserving the hierarchical structures of the original XML data. By not force fitting XML data into relational data types and not incurring the associated overhead, the pure XML data type in DB2 provides efficient access to XML data, or to specific portions of it.

XQuery

XQuery is a generalized language for querying XML data. The DB2 database system allows XQuery to be invoked directly within SQL. Because the XML data is stored in DB2 tables and views, functions are provided that extract the XML data from specified tables and views by naming the table or the view directly, or by specifying an SQL query. XQuery supports various expressions for processing XML data and for constructing new XML objects, such as elements and attributes. The programming interface to XQuery provides facilities similar to those of SQL to execute queries and retrieve results.

SQL statements and SQL/XML functions

Many SQL statements support the new XML data type. This enables you to perform many common database operations with XML data, such as creating tables with XML columns, adding XML columns to existing tables, creating indexes over XML columns, creating triggers on tables with XML columns, and inserting, updating, or deleting XML documents. The set of SQL/XML functions, expressions, and specifications supported by the DB2 database system has been enhanced to take full advantage of the new XML data type.

XQuery can be invoked from within an SQL query. In this case, the SQL query can pass data to XQuery in the form of bound variables.

Import and export utilities

The import and export utilities have been updated to support the native XML data type. These utilities treat XML data like LOB data: both types of data are stored outside the actual table. Application development support for importing and exporting XML data is also provided by updated db2Import and db2Export APIs. These updated utilities permit data movement of XML documents stored in XML columns that is similar to the data movement support for relational data.

Explain and Visual Explain

The Explain facility and the Visual Explain GUI tool have been updated to support SQL enhancements for querying XML data and to support XQuery expressions. These updates to the Explain facility and to the Visual Explain GUI tool allow you to see quickly how DB2 evaluates query statements against XML data.

Indexes over XML data

Indexing support is available for data stored in XML columns. The use of indexes over XML data can improve the efficiency of queries issued against XML documents. Similar to a relational index, an index over XML data indexes a column. They differ, however, in that a relational index indexes an entire column, while an index over XML data indexes part of a column. You indicate which parts of an XML column are indexed by specifying an XML pattern, which is a limited XPath expression.

Optimizer

The optimizer has been updated to support the evaluation of SQL, XQuery, and SQL/XML functions that embed XQuery against XML and relational data. The optimizer exploits statistics gathered over XML data, as well as data from indexes over XML data, to produce efficient query execution plans.

3.1.5 SAP and DB2 9

SAP uses many of the new features in DB2 9, such as:

- ▶ New deployment features
- ▶ Row compression
- ▶ Large RIDs
- ▶ Recovery enhancements

Although many of the features were used within the project, some features were not required as they did not fit within the scope of this project. More information regarding these new features can be found in SAP Note 930487 DB6: Using DB2 9 with SAP Software.

New deployment features

With DB2 9, the following deployment and installation features are available:

- ▶ Multiple installs.
- ▶ One image for new installations, migrations, and fix packs.
- ▶ New installations are embedded with SAP.
- ▶ IBM DB2 CLI Driver.

These features allow for a more simplified installation and maintenance of the SAP landscape.

Multiple installs

Since Version 8, DB2 was installed on the host server and all instances used the same version. Any updates through DB2 fix packs always affected all the instances. The only exception was DB2 Alternate fix packs. However, these were not supported by SAP.

With DB2 9, it is now possible to have multiple installations. DB2 9 is no longer restricted to the default directory, and can now be installed in any directory. This allows users to install multiple versions as required on a single host.

Each installation no longer affects the other instances. Instances can switch to different versions by issuing **db2iupdt** from the desired version. This feature allows users to test different versions of DB2 on the same machine before moving it to production.

On UNIX, this feature applies to all SAP releases released with DB2 9, starting with SAP Basis 3.11.

On Windows, it is only possible to use multiple installations using Kernel 7.10.

One image

Previously, there was a distinction and differences between images for a new installation (full install) and an update (fix pack). Now with DB2 9, only one image is necessary. One image is used for both new installations or fix pack updates.

On UNIX, for a new installation, use the tool `db2setup`. To update an existing version to the latest fix pack, use the tool `installFixPak`.

On Windows, for a new installation and to update an existing version to the latest fix pack, the tool `db2setup` is used for both.

Embedded with SAP

As of SAP Basis 7.10, new installations through `SAPInst` will install local copies of the DB2 binaries for each instance in the home directory of the `db2<sid>` user. Therefore, the installation processes is simplified, as `SAPInst` installs DB2 for the user.

Also, since each SAP installation has a separate and independent DB2 installation, this facilitates an easier update procedure, as each instance can be updated with DB2 fix packs separately.

IBM DB2 CLI Driver

Previously, each application server required a local DB2 client and instance to be installed on the server. This is no longer the case with the new IBM DB2 CLI Driver. This driver can be installed in any directory and used by any application server. For example, we can install the CLI Driver to an NFS mounted directory like /usr/sap/<SID>/SYS/global/db6. Any application server just needs to mount that directory, and upon restarting the application server, it copies the drivers to a local directory. The application server then uses the local copy of the drivers to connect to the database.

Therefore, only one installation of the drivers is necessary, and all application servers can use it. As a result, application servers no longer need to have a local installation DB2 client or instance. The application servers now only need one user, <sid>adm. The database users, db2<sid> and sap<sid>, only need to exist on the database server. The database-related groups, db<sid>adm, db<sid>ctl, and db<sid>mnt, only need to exist on the database server and are no longer needed on the application server.

Since the NFS mounted drivers are copied to a local directory upon restarting the application server, this allows rolling upgrades of all the applications servers.

This feature is only available on Kernel 7.10.

Row compression

Row compression is actively used by SAP Business Intelligence. As of SAP BI 7.10, it is automatically activated for InfoCubes.

With SAP BI 7.10, the compression rates can be checked through the SAP DBA Cockpit (transaction ST04), as seen Figure 3-1.

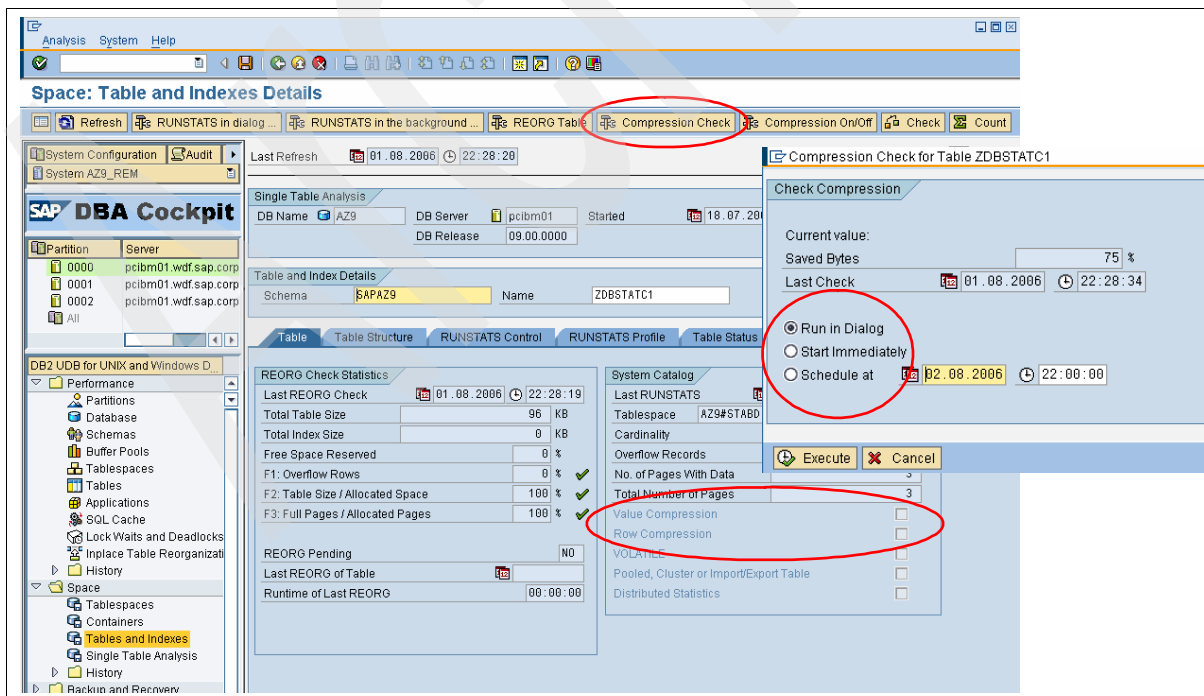


Figure 3-1 DBA cockpit: check compression

Compression can also be enabled through the SAP DBA Cockpit as well, as seen in Figure 3-2.

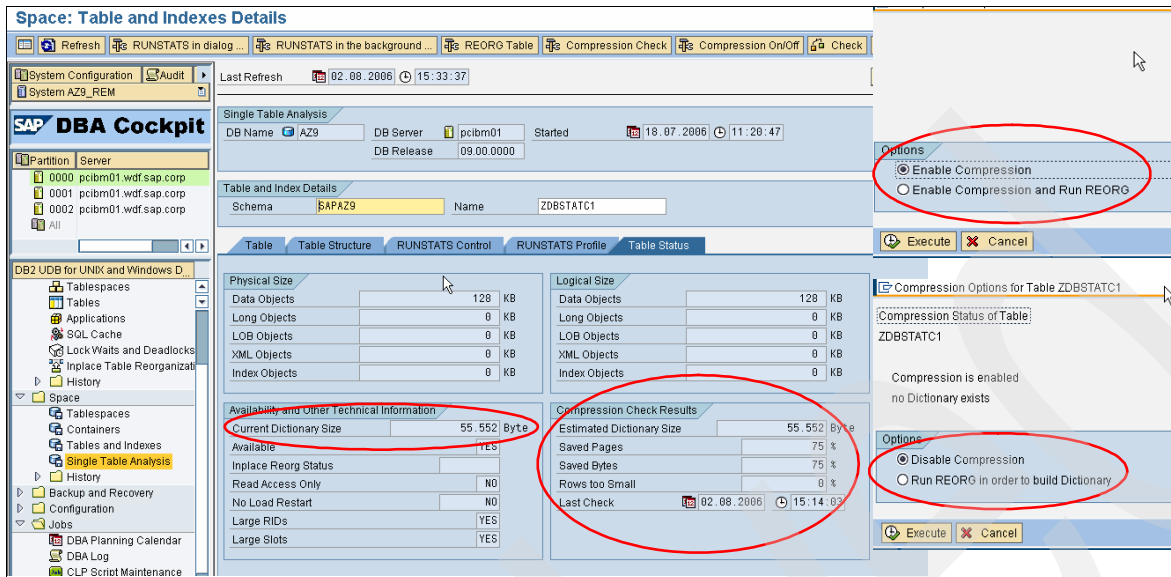


Figure 3-2 DBA Cockpit: enabling compression

Large RIDs

With previous versions of DB2, there were size restrictions on the tablespaces. With DB2 9 these restrictions no longer exist.

Addressing of pages within a tablespace is done with a row identifier (RID) that contains the page number and slot number. Previous versions used a row identifier of 4 bytes. This introduced different tablespace size limitations, depending on the page size, as outlined in Table 3-2.

Table 3-2 Tablespace limitations with 4-byte RID

| Page size | Maximum tablespace size |
|-----------|-------------------------|
| 4 KB | 64 GB |
| 8 KB | 128 GB |
| 16 KB | 256 GB |
| 32 KB | 512 GB |

With DB2 9, a 6-byte RID is now used and the maximum size of the tablespaces has increased, as outlined in Table 3-3.

Table 3-3 Tablespace limitations with 6-byte RID

| Page size | Maximum tablespace size |
|-----------|-------------------------|
| 4 KB | 2 TB |
| 8 KB | 4 TB |
| 16 KB | 8 TB |
| 32 KB | 16 TB |

Large RIDs are enabled by default with new DMS tables in DB2 9. The minimum requirement to enable large RIDs for an existing table is to:

1. Convert the data tablespace to LARGE through an **alter tablespace** command.
2. Rebuild the indexes for the desired table to enable the large RID.

These steps are shown in Example 3-3.

Example 3-3 Enable large RIDs on one table

```
$ db2 alter tablespace <tablespace> convert to large
$ db2 reorg indexes all for table <table name>
```

The **alter tablespace** command is fully logged and supports ROLLBACK and RESTORE/ROLLFORWARD. When a tablespace is converted to LARGE, a SQL warning is returned (SQL1237W) and ADM6104W is written to the notification log.

Although the **alter tablespace** command is quick, it acquires locks and scans some tables:

- ▶ SYSCAT.TABLESPACES
An X lock on row of the tablespace and updates it.
- ▶ SYSCAT.DATAPARTITIONS
Scanned for data and long data in the tablespace.
- ▶ SYSCAT.INDEXES
scanned for indexes in the tablespace.
- ▶ SYSCAT.TABLES
Z lock for every table whose definition indicates that the tablespace can be used. This effectively takes the tablespace offline until the transaction completes.

At the COMMIT, the changes are written to the SPCS file and changes are made to *in memory* structures, such as the BPS pool table and DMS table control blocks. Of course, the locks are then released.

A table does not support the large RIDs until all indexes of the table have been enabled. To check whether a table supports large RIDs, the command in Example 3-4 can be issued.

Example 3-4 Checking whether tables support large RIDs

```
select tabname, tabschema, dbpartitionnum from table (ADMIN_GET_TAB_INFO(“”,“”))
as tabinfo where large_rids = ‘P’
```

Executing an offline REORG of the table also enables large SLOTS. A table does not support more than 255 rows per page until an offline REORG has been used to reorganize the table. Reorganizing the table automatically reorganizes the indexes, thereby allowing large SLOTS and RIDs.

Table 3-4 shows the minimum record length and maximum number of records per data page regarding large SLOTS.

Table 3-4 Benefits of large SLOTS

| Page size | Reg TBSP min. rec. length | Reg TBSP max records | Large TBSP min. rec. length | Large TBSP max records |
|-----------|---------------------------|----------------------|-----------------------------|------------------------|
| 4 KB | 14 | 251 | 12 | 287 |
| 8 KB | 30 | 253 | 12 | 580 |
| 16 KB | 62 | 254 | 12 | 1165 |
| 32 KB | 127 | 253 | 12 | 2335 |

To check whether a table will benefit from large SLOTS, the command in Example 3-5 can be executed.

Example 3-5 Checking the benefits of large SLOTS

```
select tabschema, tabname, avgrowsize from syscat.tables
```

If the average row size is less than the minimum record length, there are possible storage benefits of enabling large SLOTS.

To check whether a table is using large SLOTS, a command similar to Example 3-6 can be executed.

Example 3-6 Checking for large SLOTS

```
select tabname, tabschema, dbpartitionnum from table (ADMIN_GET_TAB_INFO('','','')) as tabinfo where large_slots = 'P'
```

With the DBA Cockpit (transaction ST02), tables can be converted and checked if they are enabled for large RIDs and SLOTS, as seen in Figure 3-3.

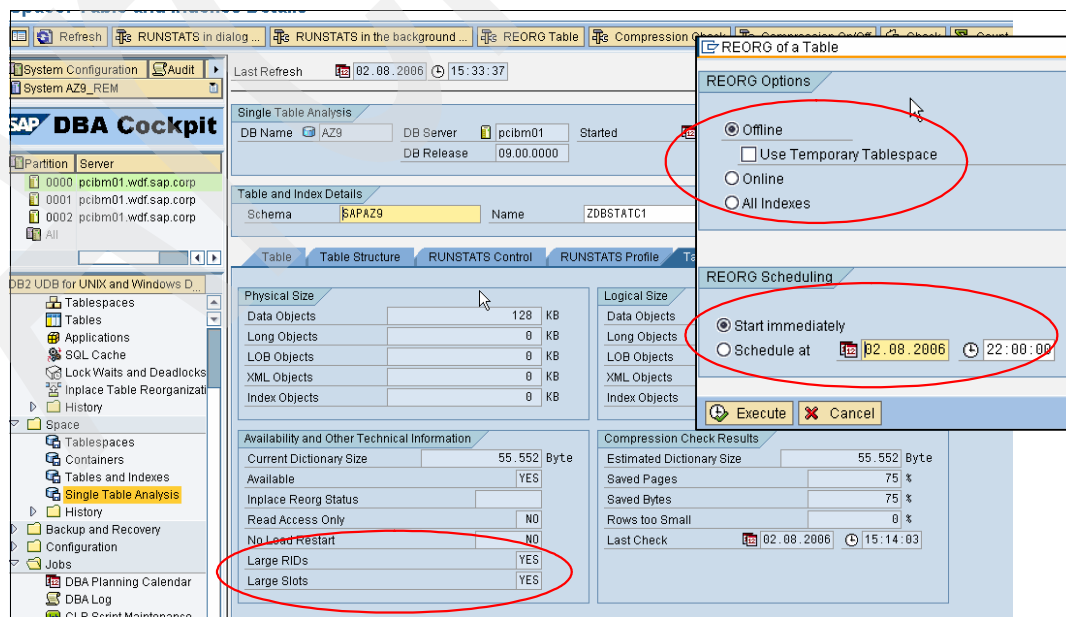


Figure 3-3 DBA Cockpit: enable large RIDs and SLOTS

Recovery enhancements

DB2 9 has improved and added the following enhancements for recovery:

- ▶ Resumable recover
- ▶ Redirected restore script from backup images
- ▶ Support for flash copy or file system backups

Resumable recover

In earlier versions of DB2 UDB, if a **recover** command is interrupted during the roll forward phase, the subsequent **recover** command always does a fresh database restore.

With DB2 9, this is no longer the case. If the **recover** command is in the roll-forward phase and is interrupted, and when the recover command is re-issued, it no longer starts with database restore, but tries to continue where it left off.

On the subsequent retry of the **recover** command, the point-in-time for the target can also be changed, provided that roll forward has not passed the new target time.

If we do not want the **recover** command to continue where it previously left off, we can issue the **recover** command with a restart option. This starts the recover process with a database restore.

Redirected restore script from backup images

In earlier versions, a redirected restore script can only be generated if the source database is available. As a result, SAP provided a tool that generates the redirected restore script.

With DB2 9, we can create a redirected restore script from any existing DB2 backup image, as shown in Example 3-7.

Example 3-7 Redirected restore script generation

```
restore db edgar from /home/db2sid/backups taken at 20051209091000 redirect
generate script edgar_node0000.clp
```

An easily modifiable CLP script will be generated that contains the **restore** command and all available options and the tablespace definitions, as seen in Example 3-8.

Example 3-8 Sample redirected restore script

```
RESTORE DATABASE EDGAR
--USER '<username>'
--USING '<password>'
FROM '/home/db2sid/backups'
TAKEN AT 20051209091000
-- DBPATH ON '<target-directory>'
INTO EDGAR
-- NEWLOGPATH '/home/db2sid/EDGAR/log_dir/NODE0000/SQL00001/SQLLOGDIR/'
-- WITH <num-buff> BUFFERS
-- BUFFER <buffer-size>
-- REPLACE HISTORY FILE
-- REPLACE EXISTING
REDIRECT
-- PARALLELISM <n>
-- WITHOUT ROLLING FORWARD
-- WITHOUT PROMPTING
;
```

```

-- *****
-- ** Tablespace name                = SYSCATSPACE
-- ** Tablespace ID                  = 0
-- ** Tablespace Type                = System managed space
-- ** Tablespace Content Type        = Any data
-- ** Tablespace Page size (bytes)   = 4096
-- ** Tablespace Extent size (pages) = 32
-- ** Using automatic storage         = No
-- ** Total number of pages          = 5572
-- *****
SET TABLESPACE CONTAINERS FOR 0
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
PATH 'SQLT0000.0'
);
-- *****
-- ** Tablespace name                = TEMPSPACE1
-- ** Tablespace ID                  = 1
-- ** Tablespace Type                = System managed space
-- ** Tablespace Content Type        = System Temporary data
-- ** Tablespace Page size (bytes)   = 4096
-- ** Tablespace Extent size (pages) = 32
-- ** Using automatic storage         = No
-- ** Total number of pages          = 0
-- *****
SET TABLESPACE CONTAINERS FOR 1
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
PATH 'SQLT0001.0'
);
-- *****
-- ** Tablespace name                = DMS
-- ** Tablespace ID                  = 2
-- ** Tablespace Type                = Database managed space
-- ** Tablespace Content Type        = Any data
-- ** Tablespace Page size (bytes)   = 4096
-- ** Tablespace Extent size (pages) = 32
-- ** Using automatic storage         = No
-- ** Auto-resize enabled             = No
-- ** Total number of pages          = 2000
-- ** Number of usable pages         = 1960
-- ** High water mark (pages)        = 96
-- *****
SET TABLESPACE CONTAINERS FOR 2
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
FILE '/tmp/dms1'                    1000
, FILE '/tmp/dms2'                  1000
);

RESTORE DATABASE EDGAR CONTINUE;

```

Since all the available options are commented out in the script, it is very easy for us to change the parameters and options.

Support for FlashCopy or file system backups

In order to do a FlashCopy, a split mirror, or a file system backup, it is necessary to know which files and directories are needed. Missing files lead to incomplete database copies that cannot be restored to a point of consistency.

DB2 9 makes it easy for us to know which files and directories are needed. A simple SQL command (shown in Example 3-9) gives a list of all the required data.

Example 3-9 ADMIN_LIST_DB_PATHS

```
select dbpartitionnum, type, path from table (admin_list_db_paths()) as files
```

3.2 The project environment

In this section we cover the details and design for database servers used for tests. We had two fully operational environments independent of each other, one with the full 60 TB database named SYS1 and one with a 16 TB database used for small tests named SYS2. Since most tests were executed on SYS1 in other sections of this book, you may not see any description of SYS2. However, in our case, from the DB2 perspective, some meaningful tests on this environment were executed as well.

Most configuration parameters for instances and partitions are shared between both environments, but they differ in size and design. Let us discuss this in detail.

It is important though to understand how we reached the currently design. We started with a 8 TB customer database image taken in DB2 Version 8, restored it on SYS2, and executed the data redistribution described latter in this section to move large objects to the 32 DB2 partitions design.

A process to inflate tables and grow the database was executed, reaching 16 TB in data size. After this point a backup was taken and restored in SYS1. Another inflating and grow process was executed to expand the 16 TB of data to 60 TB.

After all these processes, DB2 Version 8 was migrated to DB2 9. The steps and some statistics about the inflating process are better described later in this chapter.

3.2.1 Database creation and data redistribution process

In this section we describe the process used to increase the number of database partitions spread across different LPARs. The redistribution was executed using DB2 Version 8.

Step 1 - restoring the initial image in our environment

The initial image was restored by using the DB2 regular restore process. A redirect restore was executed to redirect containers that were being used by the customer to containers available in our servers. Figure 3-4 illustrates the situation that existed at that stage.

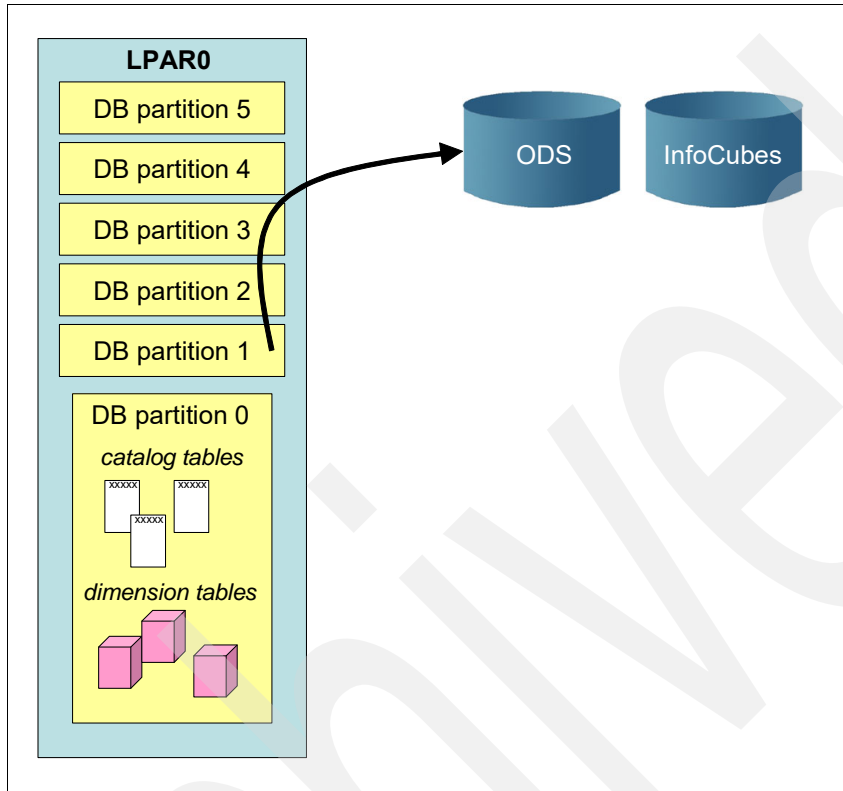


Figure 3-4 First database layout: as it is for the customer

This step simply creates a replica of the initial database into our environment with the same node group and design previously used by the customer, with 5+1 DB2 partitions. This database was restored in approximately 50 hours for an 8 TB database.

Step 2 - adding new LPARs and creating 32 new DB partitions

In our scenarios, four database-dedicated LPARs were added. For each LPAR, eight additional DB partitions were created, which resulted in the infrastructure shown in Figure 3-5.

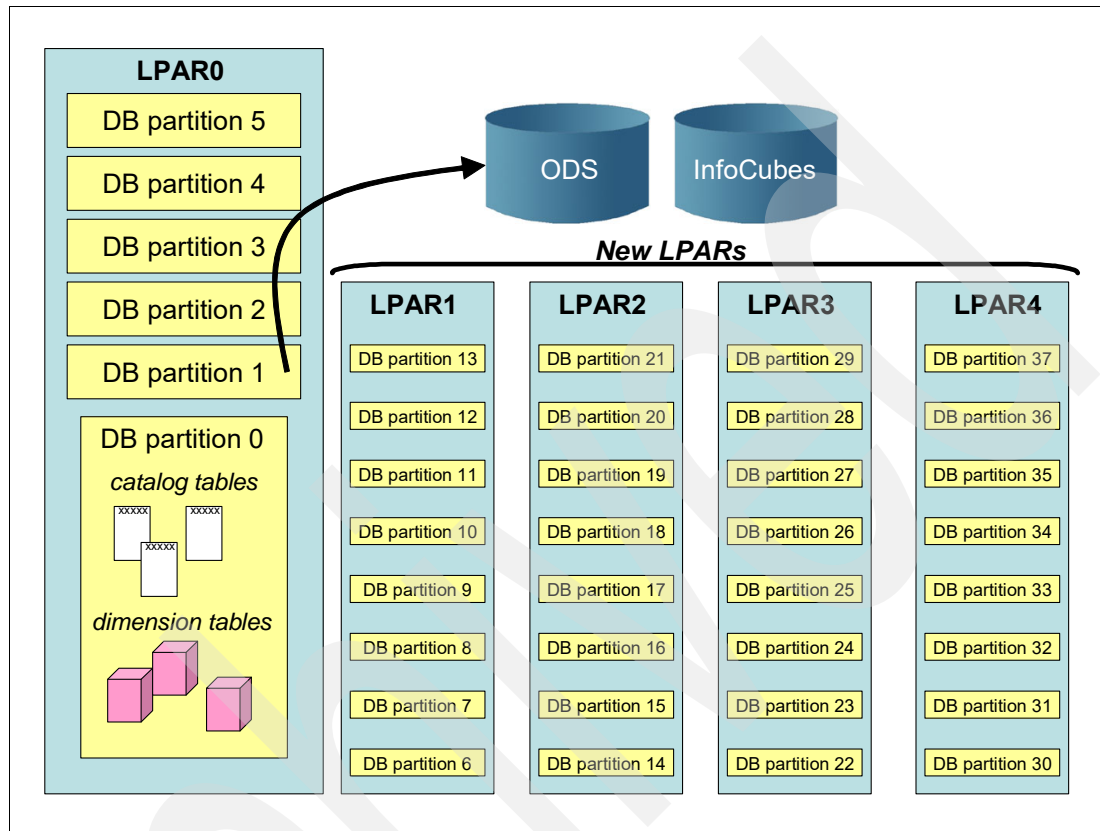


Figure 3-5 DB2 partitions and LPARs redistribution

The redistribution process presented a number of challenges, which we evaluated in order to select the most suitable method. The primary challenges were the time it would take, the amount of database logs created, the total number of tables to move, and the number of large tables. We evaluated a number of methods to perform this redistribution:

- ▶ The DB2 **redistribute** command

This is a standard DB2 command. We did not use this method because it would take too long to complete and would also generate a massive amount of logs (it uses a select/insert method to perform the move).
- ▶ The db6conv SAP ABAP™ utility

This is an SAP utility for table movement. It can be configured to use the DB2 Call Level Interface (CLI) load so that it does not log the inserts. The utility is best suited to move single tables. However, it can be configured to use a list of tables as input. We did not use this method because of the number of tables to be moved.
- ▶ The db2move native DB2 tool

This DB2 tool is suitable to move a large number of tables using the export and import load utilities of DB2. We did not use this method because it does not work with the SAP data dictionary and it uses the load-from-cursor, which means that the data must pass through the coordinator node, namely the DB2 partition 0.

► The SAP r3load utility tool

This is a tool designed to export and import data into new or already existing databases. This tool can work with some of the SAP data dictionary logic and information using SAP data classes to move large numbers of tables.

This utility was used to move the most tables. It also provides a compressing ratio for storing files created during the export.

► The IBM DB2 High Performance Unload (DB2 HPU) utility

This is a high-speed DB2 utility for unloading and loading tables.

We used this method to move the largest tables in the system (approximately 50 tables) that represented the heaviest amount of data (approximately 60% of the data in the system).

Step 3 - executing the data redistribution

After adding new LPARs, and in order to overcome future scaling challenges, data redistribution from the original database to the new DB2 partitions was required. We used the DB2 HPU utility (which was originally created for z/OS® environments but is becoming more common in System p and System x™ environments) and the SAP r3load.

As shown in table Table 3-5, the major amount of *data* was moved using DB2 HPU and the majority of the *tables* were moved using r3load. We chose this approach because r3load is SAP data-classes-aware, which helped the overall process, and the use of DB2 HPU high-performance for the largest tables was also very positive, reducing the time needed to redistribute data.

Table 3-5 Volume of data redistributed

| Number of tables | Amount of data (TB) | Approach used |
|------------------|---------------------|---------------|
| 48 | 4.5 | DB2 HPU |
| 41 716 | 3.5 | SAP r3load |

IBM DB2 High Performance Unload (DB2 HPU)

Using DB2 HPU in our environment was crucial for the node redistribution because it has the capability and the sensitivity to unload data based on the hashing key of the new partitioned table. This feature improved the load of this data since it would be already split, based on the new table hashing key. The hashing key is tightly related with the partitioning key. In our scenario, the partitioning key did not change, so we only had a newer hashing key based on the new partitioning schema. We used DB2 HPU Version 2.2.4 for our tests.

Note: Because it is hashing key-aware, DB2 HPU is very useful for SAP NetWeaver BI/DB2 partition redistribution, and for situations in which a massive amount of data needs to be redistributed.

DB2 HPU can extract tables from full, offline backups, including backups of system-managed space (SMS) or database-managed space (DMS) tablespaces or backups of the database. For backups of tablespaces where a suitable backup of the DB2 catalog exists, DB2 HPU can extract the tables from the backup even if DB2 is down.

Version 2.2 also offers new repartitioning capabilities for data redistribution on the same system or for data migration to a new system. High-speed data unloads and splitting can now be done in a single operation, thus eliminating the use of unnecessary disk space.

In our case, we avoided the issue of unnecessary disk space by using pipes. We exported the table to a pipe and imported it back to the new table and partition.

A script was created to execute the data unload and load phase. The DB2 HPU export/import script is provided in Example 3-10. The script is split in different parts:

1. Lines 1 to 3 represent the header and for loop in all tables that were chosen.
2. Lines 5 to 13 provide the function to gather tablespace information, index space information, and generated Data Definition Language (DDL) creation statements for the table, its views, and its indexes.
3. Lines 14 to 22 provide the function to rename the real SAP table name to a temporary name. If the table has dependent views, the views are dropped, the table is renamed, and the views are recreated. It also generates indexes renaming clauses.
4. Lines 23 to 24 provide the function to execute the renaming statements and to create the table with the real SAP name on new DB2 partitions, which will be used as a target table.
5. Lines 25 to 41 provide the function to generate the DB2 HPU control file.
6. Lines 42 to 52 provide the function to create pipe and to start data unload and load processes for each new DB2 partition.
7. Lines 53 to 59 provide the function to remove temporary unused files and close the loop.

Example 3-10 DB2 HPU script used to export and import data

```

1  #!/bin/ksh
2  for table in `cat tables.list`
3      do
4          echo "starting process for table " $table " -- " `date` >>
redistribute.log
5          # get tablespace and indexspace corresponding to table
6          ts=`grep $table redistribute.list | awk '{print $2}'`
7          is=`echo $ts | sed '          {s/ODSD/ODSI/
8              t done
9              s/D/I/
10             :done
11             }' `
12         # generate ddl for table
13         db2look -d eb8 -z SAPR3 -e -o redistribute1.ddl -xd -tw $table >
/dev/null 2>>redistribute.err
14         # generate RENAME statements in tom1 file
15         rm rename view
16         sed -n -f tom1.sed redistribute1.ddl
17         # modify generated ddl to add rename statements from previous step and
create table in target tablespace
18         sed '/CONNECT TO/ {
19             r view
20             r rename
21         }
22         s/IN "." INDEX IN "."/IN "$ts" INDEX IN "$is"/'
redistribute1.ddl > redistribute2.ddl
23         # run ddl to rename source table and create target table
24         db2 +c -stvf redistribute2.ddl 1>> redistribute.log 2>>redistribute.err
25         # customize HPU control file
26         sed '/@@@/ {
27             {
28                 s#@#@SOURCE#@#$table'#

```

```

29             s/BIC/TOM/
30         }
31         s###OUTPUT###'$table'#
32         s###TARGET###'$table'#
33     } ' tomunload.template > tomunload.ct1
34 # customize load control file
35 sed '/###/ {
36     {
37         s###OUTPUT###'$table'#
38         s#/BIC/##
39     }
40     s###TARGET###'$table'#
41 } ' tomload.template > tomload.ct1
42 # prepare pipes for loads
43 for node in 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36 37
44     do
45         mkfifo /tmp/pipes$table.del.0$node
46     done
47 # run load
48 db2 -tvf tomload.ct1 1>> redistribute.log 2>>redistribute.err &
49 # now run HPU
50 db2hpu -f tomunload.ct1 -m redistribute.msg 1>> redistribute.log
2>>redistribute.err
51 # wait 2 1/5 minutes for Load to finish building index
52 #sleep 150
53 for node in 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36 37
54     do
55         rm /tmp/pipes$table.del.0$node
56     done
57
58     echo "finished with process for table " $table " -- " `date` >>
redistribute.log
59     done

```

The DB2 HPU control file used for load and unload is shown in Example 3-11. In the part shown here, terms like @ @ @ <word> @ @ @ are meant to be replaced in the data unload and load script by the real SAP table names.

In the unload control file, the clause TARGET TABLE ensures that data unload is executed using the hashing key of the new table created on the new DB2 partitions.

Example 3-11 DB2 HPU control file

```

Unload control file:
1 GLOBAL CONNECT TO EB8
2 QUIESCE NO LOCK NO
3 ;
4 UNLOAD TABLESPACE DB2 NO
5 SELECT * FROM SAPR3."###SOURCE###" ;
6 OUTPUT ("/tmp/pipes###OUTPUT###.del" )
7 OPTIONS DOUBLE DELIM ON
8 FORMAT DEL
9 TARGET TABLE (SAPR3."###TARGET###" )
10 ;

```

Load control file:

```
1 LOAD FROM @@@OUTPUT@@.del
2     OF DEL
3     SAVECOUNT 100000
4     MESSAGES @@@OUTPUT@@.msg
5     REPLACE
6         INTO SAPR3."@@@TARGET@@@"
7     NONRECOVERABLE
8     ALLOW NO ACCESS
9     LOCK WITH FORCE
10    PARTITIONED DB CONFIG
11        PART_FILE_LOCATION /tmp/pipes/BIC
12        MODE LOAD_ONLY
13    ;
```

The DB2 HPU basic configuration is shown in Example 3-12

Example 3-12 DB2 HPU basic configuration

```
# HPU default configuration
bufsize=4194304
db2dbdft=EB8
db2instance=db2eb8
maxunloads=1
nbcpu=32
maxsockets=32
insthomes=db2eb8:/db2/db2eb8
instusers=db2eb8:db2eb8
doubledelim=off
portnumber=54002
db2version=V8
```

The process, shown in Figure 3-6, can be summarized as follows:

- ▶ Phase 1 is the initial phase: All SAP BW tables were spread over five nodes in a single LPAR.
- ▶ Phase 2: This step checks and drop-dependent views.
- ▶ Phase 3: The SAP NetWeaver BI table and indexes get renamed to temporary table and indexes name.
- ▶ Phase 4: The empty SAP BW table is created using the new DB2 partition group, which is spread over eight nodes in four distinct LPARs.
- ▶ Phase 5: The data unload and load phase occurs (shown as a horizontal red arrow in the phase 5 section of the figure).

After all these phase completed, a few tests were executed to validate the new tables, and then the temporary tables were dropped.

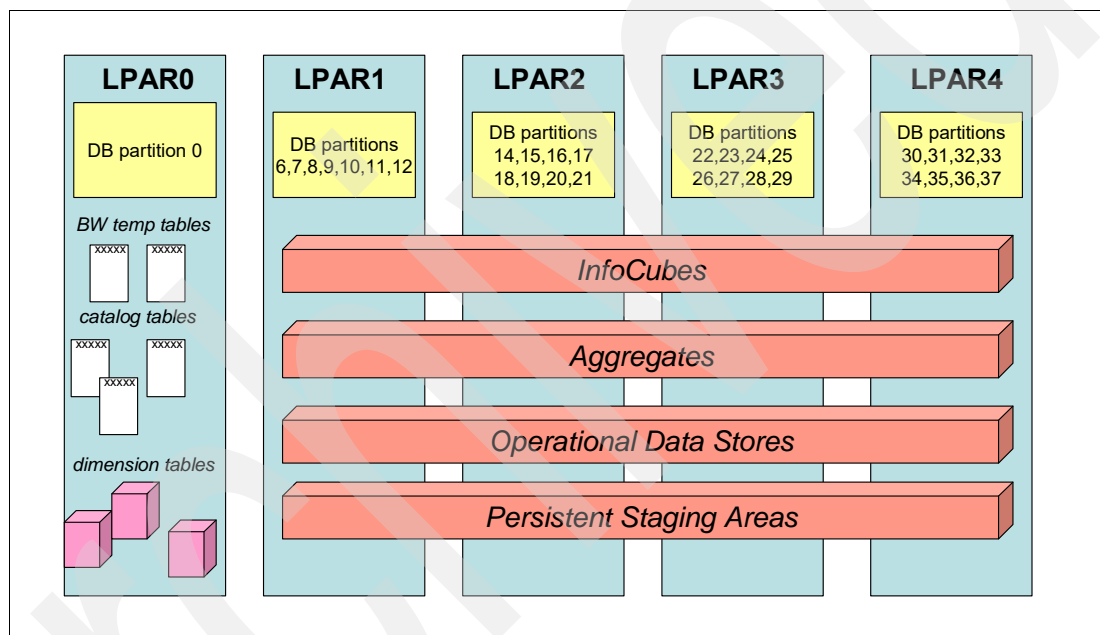


Figure 3-6 DB2 partitions redistribution with DB2 HPU

This structure offers straightforward benefits:

- ▶ Improved fall-back strategy

In our environment we did not have to deal with production constraints. However, by using this strategy, if something goes wrong during the load phase, a fallback is extremely fast since the table can be renamed to the original name.
- ▶ Reduced overload

Because the entire database did not have to be stopped, only processes that were dependent on the table being unloaded and loaded were affected. Of course, the redistribution process itself is an overload and runs in parallel, but the system itself suffered a minor impact, from the usage point of view.

The following actions, not used for complexity and time constraints in our case, may have improved the full process:

- ▶ Postponing index creation to the end of the load phase:
This considerably improves the load time, but adds another step (index creation) at the end of the phase.
- ▶ Parallelizing the number of tables being unloaded and loaded
In our environment, one table at a time was processed. Parallelizing would have considerably improved the redistribution. However, this would have add complexity to the scripts and have had some processing impact on the DB2 coordinator node.

Also note the following considerations:

- ▶ The db2hpu process is very CPU intensive. Consider and plan the execution of this activity to best fit in your own environment.
- ▶ Although we faced a few minor issues in using the tool, some of the issues were related to not having the latest level of AIX in our environment.

SAP r3load

The SAP r3load tool is provided by the SAPinst installation tool. It is a tool that helps you to export and import data based on SAP data classes. In our case, the tables that were not redistributed by using DB2 HPU were exported and imported using SAP r3load. Overall, it took 25 hours to export and import 3.5 TB of data.

Redistributed scenario

At the end these steps, you will have the redistributed environment illustrated in Figure 3-7. InfoCubes fact tables, aggregates, ODSs, and PSAs were spread over the four new LPARs and LPAR0 dedicated to the DB2 coordinator partition (partition 0) and to store dimension, SAP NetWeaver BI temporary tables, and catalog tables.

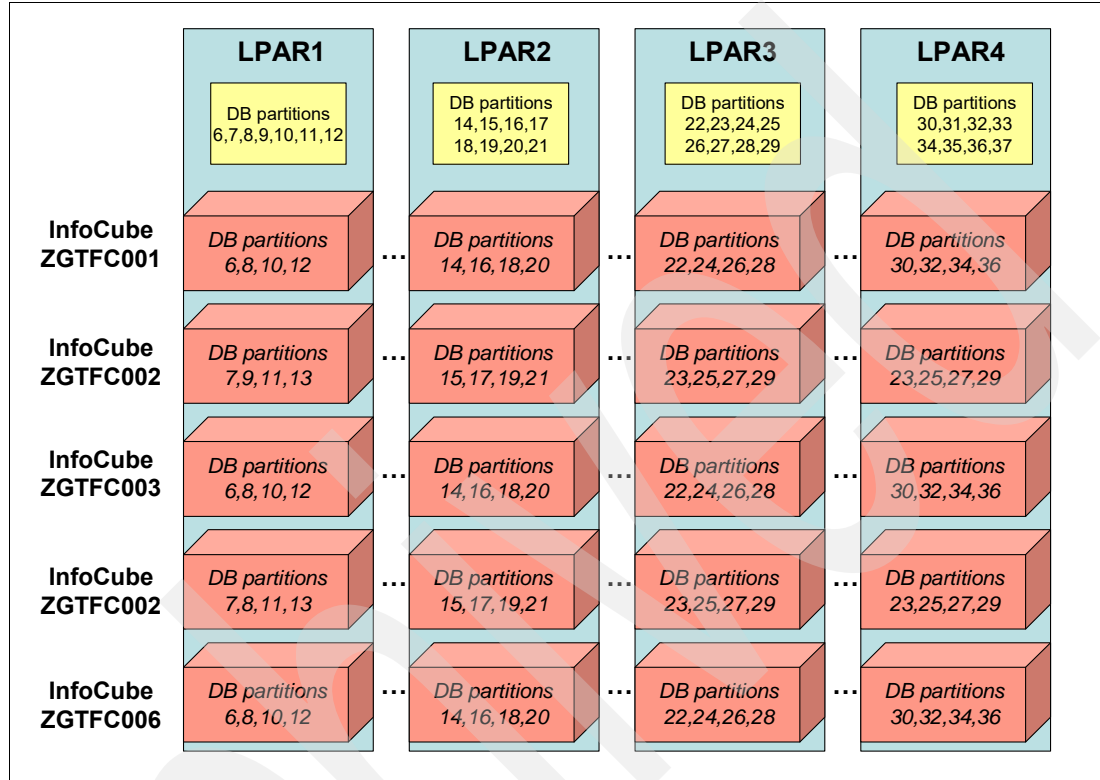


Figure 3-7 SAP BW objects and DB2 partitions

The approach adopted in our redistribution scenario improved system availability. The period that required SAP NetWeaver BI to be stopped occurred during the r3load redistribution phase, because the massive amount of data was redistributed before by using the DB2 HPU utility.

3.2.2 DB2 database data grow

That were two data grow phases in our environment:

1. The first one was executed to expand data from 8 TB to 16 TB (in SYS2).
2. The second grow phase was executed to expand from 16 TB to 60 TB (in the SYS1 system).

The data grow procedure was executed with SAP NetWeaver BI transactions to grow tables in size and creation of new objects. These new objects were critical to allow the KPI tests because they were used for the online stress test, named KPI_G. For further information about KPIs descriptions and results go to Chapter 1, "Project overview: business objectives, architecture, infrastructure, and results" on page 1 2.

Some information about the data grow process from 16 TB to 60 TB is shown in Table 3-6.

Table 3-6 New objects added

| Object type | Number of objects added | Specific use | Number of 16 KB pages |
|-------------|-------------------------|-----------------|-----------------------|
| InfoCubes | 36 | Upload tests | 0 |
| InfoCubes | 32 | Aggregate tests | 9983176 |
| ODS | 30 | N/A | 168405548 |
| PSA | 13 | N/A | 114778560 |

The depicted objects created expanded the database in approximately 4.6 TB. The remaining data grow was done by expanding existing tables in size using a script created by SAP. Most of the data grow consisted of expanding existing InfoCubes fact tables, but some operational data store tables and persistent storage area tables were also used.

3.2.3 DB2 migration

DB2 Version 8 was originally installed in our environment. Then we migrated to DB2 9 for both environments, SYS1 and SYS2.

The migration procedures were designed based on the SAP migration guide *Migration to Version 9 of IBM DB2 for Linux, UNIX, and Windows*, available at SAP Service Marketplace:

<http://www.service.sap.com>

Important: We strongly recommend that you read and define your migration plan based on the SAP guideline.

The migration of components in our environment to DB2 9 required different tasks:

- ▶ Migrating DB2 database servers
- ▶ Migrating DB2 clients

Because the client migration is straightforward and widely documented in the DB2 InfoCenter, we only describe the steps and statistics of the database server migration.

Migrating the DB2 servers

That are three main steps to execute for the migration:

- ▶ The instance migration
- ▶ The DB2 Administration Server (DAS) migration (optional)
- ▶ The database migration

Instance migration

The `db2imigr` command does the following actions:

- ▶ It migrates an existing instance to a new DB2 9 instance.
- ▶ It migrates instance profile registry variables. The global profile registry variables set by the user are not migrated.
- ▶ It migrates the database manager configuration (dbm cfg) file.
- ▶ It sets the `jdk_path` database manager (dbm cfg) parameter appropriately.
- ▶ It copies over other configuration files.

DB2 Administration Server migration

As part of the overall migration process to DB2 9, you can migrate your DB2 Administration Server (DAS) to keep your existing DAS configuration. Otherwise, you can drop your existing DAS and create a new DAS in DB2 9. You only need a DAS running on DB2 9 to use the Control Center for remote administration of DB2 9 instances, job management, and job scheduling.

Database migration

When migration is called explicitly using the `MIGRATE DATABASE` command, or implicitly using the `RESTORE DATABASE` command from a DB2 UDB Version 8 backup, the following database entities are converted during the database migration:

- ▶ Database configuration file
- ▶ Log file header
- ▶ Catalog tables
- ▶ Buffer pool files
- ▶ Table and Index root page
- ▶ History file

Note: Before migrating to DB2 9, we recommend that you test the migration in a test system prior production. It is extremely important to execute a database backup and create a fallback plan, which may differ between customers' environments, criticality, and business rules.

We observed that:

- ▶ The migration tasks and DAS were executed without any impact because it is a fast procedure that changes profiles, shared library links, and files to point to DB2 9.
- ▶ The migration of tables and indexes root page does not have the same impact on a non-partitioned database compared to a partitioned database. This is because indexes and tables would have a root page per partition, and its tablespace is spread. This can potentially increase the execution time-frame and the resource usage.
- ▶ As SAP NetWeaver BI has a high number of tables in which different DB2 partitions can be spread over. This number can grow exponentially.
- ▶ Depending on the number of objects, the catalog migration can also be a task that may consume considerable resources and time.
- ▶ The database migration of a partitioned environment has a particular migration sequence. All tasks that we describe are executed for each partition. However, in the concentrator and catalog partition, this procedure is executed first, fully migrating it at the very beginning. Afterwards, remaining DB2 partitions are migrated in parallel.

The DB2 partition migration sequence starts with the migration for the log file header, database configuration, and buffer pool files. The second step is the table and index root page migration. The log of this task execution in db2diag.log is shown in Example 3-13. The number in bold shows the file ID (FID) of the table, which can be found in SYSIBM.SYSTABLES. The last migration step is the history file migration.

Example 3-13 Message of an index and table root page migration in db2diag.log

```

2007-02-07-12.17.47.311328+060 I7222558A410      LEVEL: Info
PID      : 905386                TID   : 1                PROC  : db2agent (EB8) 0
INSTANCE: db2eb8                NODE  : 000              DB    : EB8
APPHDL   : 0-51                 APPID : *NO.db2eb8.070207111734
AUTHID   : DB2EB8
FUNCTION: DB2 UDB, data management, sqlDMigrateDAT, probe:675
MESSAGE  : Migrate DAT object ID
DATA #1  : unsigned integer, 2 bytes
1

```

Note: By the time that this book was being written there was a system problem on the migration that repeats the index root page migration. This will be fixed in fix pack 3 of DB2 9. Check with your IBM representative for the latest information.

The following tasks are executed:

- ▶ All tasks are executed in the DB2 partition 0 first and then replicated to the remaining DB2 partitions (as part of the catalog migration). We started the migration at 12:17:34, as shown in Example 3-14.

Example 3-14 Engine migration start

```

2007-02-07-12.17.34.601733+060 I7218095A259      LEVEL: Warning
PID      : 1732660              TID   : 1                PROC  : db2bp
INSTANCE: db2eb8                NODE  : 000
FUNCTION: DB2 UDB, base sys utilities, sqlmngdb, probe:10
MESSAGE  : Begin engn migration

```

- ▶ Afterwards, log file header, database configuration, and buffer pool files were migrated, but they are not shown in db2diag.log.
- ▶ The next step is the migration of table and index root pages for all tablespaces in the DB2 partition 0, as shown in Example 3-15.

Example 3-15 Table and index root page migration start and end messages

```

2007-02-07-12.17.47.305975+060 I7221663A479      LEVEL: Info
PID      : 905386                TID   : 1                PROC  : db2agent (EB8) 0
INSTANCE: db2eb8                NODE  : 000              DB    : EB8
APPHDL   : 0-51                 APPID : *NO.db2eb8.070207111734
AUTHID   : DB2EB8
FUNCTION: DB2 UDB, data management, sqlDMigrateDMSTablespace, probe:1307
MESSAGE  : Starting Migration of DMS tablespace ID
DATA #1  : String, 11 bytes
SYSCATSPACE
DATA #2  : unsigned integer, 2 bytes
0
<...> skip to fit in layout
2007-02-07-12.19.10.788462+060 I19732752A385      LEVEL: Info
PID      : 905386                TID   : 1                PROC  : db2agent (EB8) 0

```

```
INSTANCE: db2eb8          NODE : 000          DB : EB8
APPHDL  : 0-51           APPID: *NO.db2eb8.070207111734
AUTHID   : DB2EB8
FUNCTION: DB2 UDB, data management, sqlMigrateTablespaces, probe:1733
MESSAGE  : Finished physical objects
```

- ▶ The next step is the catalog tables migration. This step is only executed in the catalog and concentrator partition, the DB2 partition 0, as shown Example 3-16.
 - The first message is the catalog migration start.
 - The second set of messages shows the start and end of a specific table migration (in this case the SYSIBM.SYSCODEPROPERTIES).
 - And at the very end, the message states that the catalog migration has finished.

Example 3-16 Migration of one specific catalog table and start and end of catalog migration messages

```
2007-02-07-12.19.17.996733+060 I32244613A412    LEVEL: Warning
PID      : 905386          TID   : 1          PROC  : db2agent (EB8) 0
INSTANCE: db2eb8         NODE  : 000        DB    : EB8
APPHDL   : 0-51          APPID: *NO.db2eb8.070207111734
AUTHID   : DB2EB8
FUNCTION: DB2 UDB, catalog migration, sqlrM_catalog_migrate, probe:230
MESSAGE  : Start to migrate UDB DB2 Release: 0xa00 database
```

<...> skip to fit in layout

```
2007-02-07-12.24.03.665362+060 I32411796A413    LEVEL: Warning
PID      : 905386          TID   : 1          PROC  : db2agent (EB8) 0
INSTANCE: db2eb8         NODE  : 000        DB    : EB8
APPHDL   : 0-51          APPID: *NO.db2eb8.070207111734
AUTHID   : DB2EB8
FUNCTION: DB2 UDB, catalog migration, sqlrM_catalog_migrate, probe:275
MESSAGE  : Begin migrating catalog table: SYSCODEPROPERTIES
```

```
2007-02-07-12.24.03.701023+060 I32412210A412    LEVEL: Warning
PID      : 905386          TID   : 1          PROC  : db2agent (EB8) 0
INSTANCE: db2eb8         NODE  : 000        DB    : EB8
APPHDL   : 0-51          APPID: *NO.db2eb8.070207111734
AUTHID   : DB2EB8
FUNCTION: DB2 UDB, catalog migration, sqlrM_catalog_migrate, probe:310
MESSAGE  : End migrating catalog table: SYSCODEPROPERTIES
```

<...> skip to fit in layout

```
2007-02-07-12.24.04.027919+060 I32412623A418    LEVEL: Warning
PID      : 905386          TID   : 1          PROC  : db2agent (EB8) 0
INSTANCE: db2eb8         NODE  : 000        DB    : EB8
APPHDL   : 0-51          APPID: *NO.db2eb8.070207111734
AUTHID   : DB2EB8
FUNCTION: DB2 UDB, catalog migration, sqlrM_catalog_migrate, probe:640
MESSAGE  : Catalog Migration invocation is complete with rc = 0.
```

- The history file migration is the last step executed in DB2 partition 0. The stamp message is shown in Example 3-17. In our case, because we used the latest DB2 Version 8 fix pack before migration, it did not need to migrate it, since it was current.

Example 3-17 History file migration

```

2007-02-07-13.01.03.285366+060 I32428085A406      LEVEL: Warning
PID      : 905386                TID   : 1          PROC  : db2agent (EB8) 0
INSTANCE: db2eb8                NODE  : 000        DB    : EB8
APPHDL   : 0-51                 APPID : *NO.db2eb8.070207111734
AUTHID   : DB2EB8
FUNCTION: DB2 UDB, database utilities, sqlumigr, probe:205
MESSAGE : History file is already current. No migration required.

```

- After the DB2 partition 0 is migrated, the migration for all the remaining partitions is started in parallel.
- The last message stamped in db2diag.log after all DB2 partitions have been migrated is shown in Example 3-18.

Example 3-18 Engine migration end

```

2007-02-07-13.19.54.835948+060 I132865856A317  LEVEL: Warning
PID      : 1732660              TID   : 1          PROC  : db2bp
INSTANCE: db2eb8                NODE  : 000
APPID    : *NO.db2eb8.070207111734
FUNCTION: DB2 UDB, base sys utilities, sqlmngdb, probe:20
MESSAGE : End engn migration: with rc = 0, sqlcode = 0

```

Table 3-7 summarizes the migration steps and durations in our environment.

Table 3-7 Summary table for migration steps and durations

| Migration step | Node | Duration (hh:mm:ss) |
|---|------|-----------------------|
| Log file header, database configuration, and buffer pool files. | 0 | 00:00:13 |
| Table and index root page for all tablespaces. | 0 | 00:01:29 |
| Catalog tables migration. | 0 | 00:04:48 |
| History file migration. | 0 | 00:00:00 ^a |
| Full migration of the remaining DB2 partitions. | 1-37 | 00:18:19 |
| Total migration execution time. | 0-37 | 00:24:49 |

a. No migration required (because the file was current due of the fix pack level of the DB2 we used).

3.2.4 Database servers

For our tests, five database servers for SYS1 and five database servers for SYS2 were created. These servers were hosted in AIX LPARs. The difference between the SYS1 and SYS2 environments are the LPARs layout and the amount of resources dedicated to each LPAR.

- ▶ For SYS1 we created each AIX LPAR in a specific System p595, physically spreading the DB2 LPAR partitions, as shown in Figure 3-8.

In SYS1, the first DB2 LPAR partition, called sys1db0, is used by the DB2 partition 0, and the remaining AIX LPARs by the DB2 partitions 6 to 37. The dimension tables and the catalog tables are held in the DB2 partition 0 and all larger objects like PSA, ODSs, and InfoCubes are distributed in DB2 partitions 6 to 37.

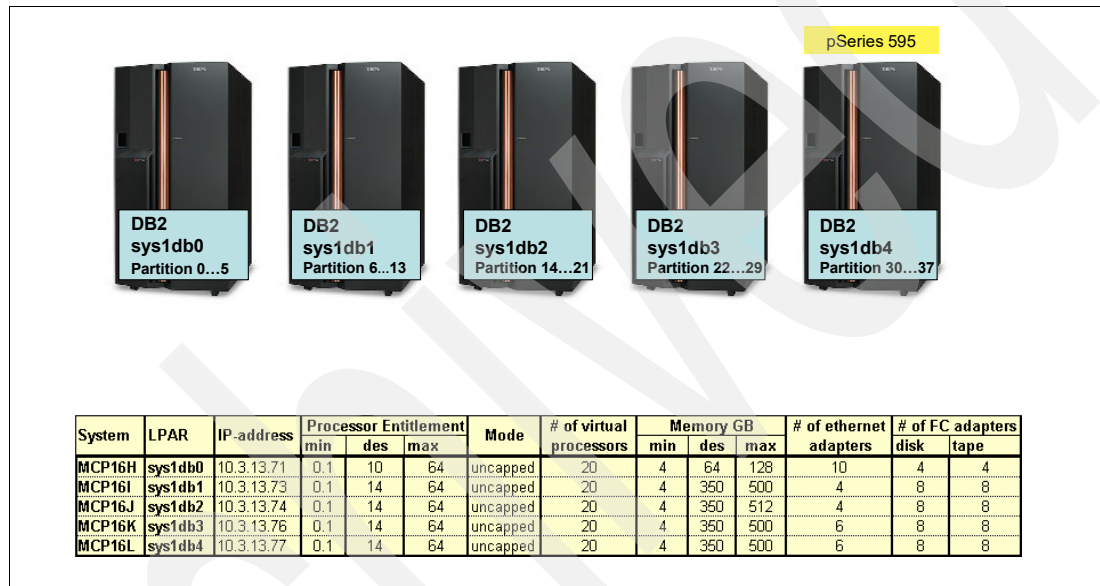


Figure 3-8 SYS1 physical servers and DB2 database LPARs

- For the SYS2, the AIX LPARs are grouped in one System p595. The same database design was implemented in SYS1 and SYS2. SYS2 AIX LPARs descriptions are shown in Figure 3-9.

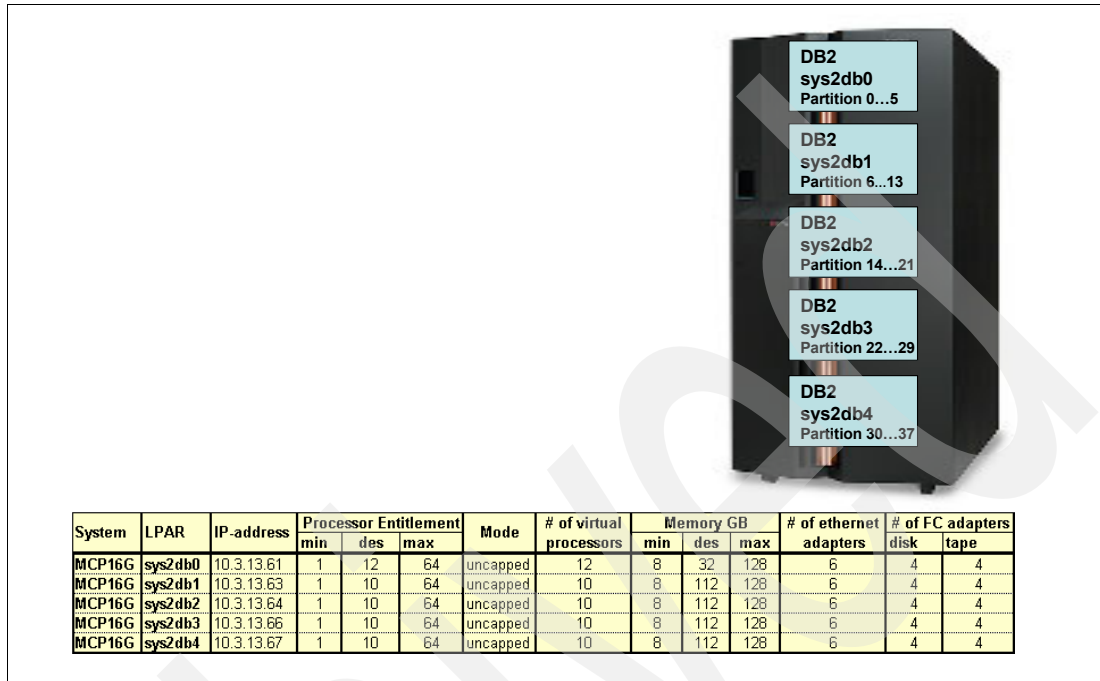


Figure 3-9 SYS2 physical server and DB2 database LPARs

The network was configured with an Ether-channel that groups network devices to improve bandwidth. A specific network was created for DB2 partitions communication and an additional network for SAP instances (for example, the central instance and the application server for DB2 partition 0).

The network created for the DB2 partitions can also be used for TSM. The architecture was created to have a very flexible environment and to facilitate tests so that they can be executed in parallel without a considerable impact.

3.2.5 Database partitions

Following SAP recommendations, the database was partitioned in 32+1 partitions in DPF environments. The DB partition 0 holds the dimension and the catalog table. The 32 remaining partitions hold larger objects like the fact tables, operational data store (ODS) tables, and persistent storage area (PSA) tables.

In our environment, you can notice that we have 33 DB2 partitions from 0 to 37. Using basic math, it is missing 5 partitions. The reason for not having partitions 1 to 5 is related with some history about how the infrastructure was created.

At the beginning we had an 8-TB database backup image taken in an environment with six partitions. The database was restored following this layout, and the new design was created to best accommodate and handle the massive increase in size we would generate. Large objects like fact tables, operation data store tables, and persistent storage area tables initially stored on partitions 1 to 5 were redistributed across new node groups using partitions 6 to 37.

After this redistribution, we could have deleted partitions 1 to 5, but, due to Tivoli Data Protection constraints, we had to keep them since TDP requires that all partitions must be sequential without missing peers.

Important: Consider DB2 partitions 1 to 5 as void, since no critical or sensitive data is stored on them. From a performance point of view they are not used.

Basically, to achieve better results and improve scalability, the number of DB2 partitions dedicated to InfoCubes fact tables, aggregate tables, and operational data store tables was increased to 32 partitions. The main node, DB2 partition 0, was configured with a higher number of CPUs because it is the partition that suffers the highest impact and load.

DB2 partitions and LPARs used in our environment as depicted in Figure 3-10. The stand-alone LPAR 0 was dedicated to DB2 partition 0, and the remaining LPARs spread with eight DB2 partitions each.

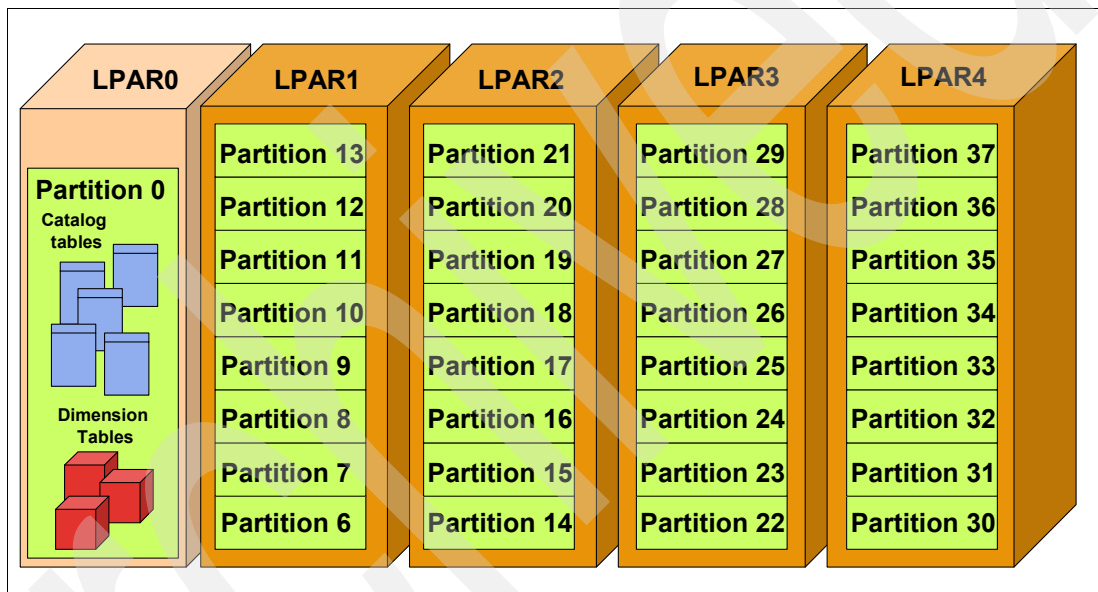


Figure 3-10 DB2 partitions and LPARs

You may think that 32 database partitions may be a very high number compared to usual database environments. The main reason for using 32 DB2 partitions was to improve scalability for some infrastructure maintenance tasks like backup, restore, and roll forward. This configuration may bring some overhead and complexity for maintenance. You may consider in your own environment whether having a higher number of nodes would bring better benefits.

3.2.6 Tablespaces and disk layout

Tablespace and disk layout can be a subject of much discussion. Nowadays it is more common to rely on storage to handle I/O balancing and prefetching techniques working as a virtual well-performing disk.

As described in Chapter 5, “The system storage perspective” on page 317, the storage used in our tests is based on IBM DS8300. Normally, storage prefetching is aggressive, since it occurs on the storage cache. With the latest DS8300, the large amount of cache considerably improves the storage response time. The latency for writing data is not very dependent on the

stripe size, array type, and how data is cached in storage or on different layers, because data is written directly into cache and asynchronously written to disk. However, even understanding that both operations cannot be split, accessing data in an optimal way can become a challenge. Let us discuss why.

Considering a basic layout of a DB2 database with a DMS tablespace using files as containers and file system caching, as shown in Figure 3-11, you can see which part is common for all layers — the cache. Will this be a problem? The rule of thumb is that we should have as much cache as possible to improve response time. That is what we may think, but that is not always true. You are not always improving overall caching by having it work independently in each layer.

Over cache can be as optimal as it can be expensive. You have to consider that to cache data, at any layer, extra resources are needed for CPU cycles to prefetch data, memory usage, or to asynchronously or synchronously remove data from cache.

Poor disk layout for write operations normally impacts in extra space needed, but no considerable performance impact can be seen. But the drawbacks of it would bring an unbalanced scenario for read operations, which would need to read more data blocks and data on a higher granularity, and in the way that it is accessed and requested in the database, may not be sequentially stored, causing bad prefetching and cache hit ratios.

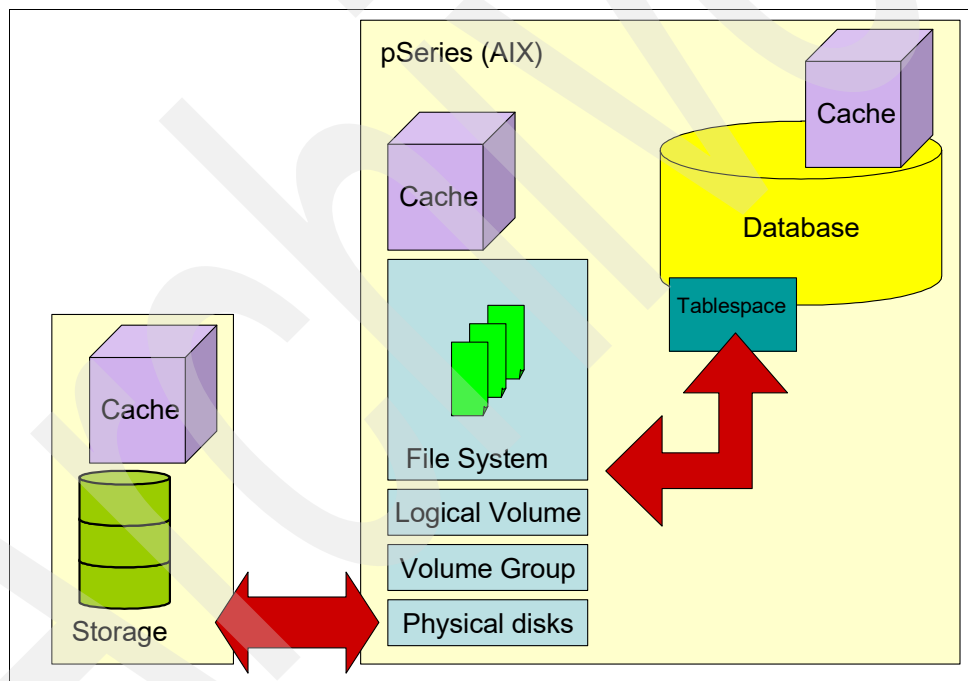


Figure 3-11 Macro view of database and storage layers

Tuning cache in all layers is heavily dependent on the infrastructure design and hardware used. We provide thoughts and recommendations about what we used in our design and you should use your experiences and infrastructure design to possibly match them.

When dealing with virtualization of the storage using concepts of virtual disks, it is important to design your storage layout in a way in which I/O is balanced across physical disks. In our case, the largest environment, SYS1, was generated from SYS2. We cover the layout as a single instance and you can consider it true for both systems due to their unique roots.

Tablespace layout

Tablespace layout was created based on DMS tablespaces. We chose the use of files on JFS2 file systems turning off file system cache for each tablespace.

By disabling file system cache, we enabled concurrent I/O, which bypasses file cache and transfers data directly from disk into the user space buffer. It also bypasses the inode lock, which allows multiple threads to read and write to the same file concurrently.

For further information about DB2 file systems and how they were created, refer to Chapter 4, “The IBM System p perspective” on page 287.

The use of concurrent I/O was chosen to decrease CPU usage and memory consumption by file system cache for database data, focusing more on the DB2 mechanisms of controlling and prefetching data. By doing this, one of the cache layers for file system was disabled. We did some tests with file system cache, comparing it with concurrent I/O.

Each DB2 tablespace partition had four LPARs created. So, for a tablespace spread over eight DB2 partitions, it would have eight times four containers, a total of 32 containers. Since that is at least two DB2 partitions per LPAR, eight containers per LPAR from each would have their own file system on a specific array in DS8300. Balancing between containers and storage is described in “Balancing tablespace layout and disk layout” on page 171.

In our environment, disregarding DB2 partition 0, which holds small objects, the remaining partitions hold four groups of tablespaces:

- ▶ Tablespaces for ODS
- ▶ Tablespaces for PSA
- ▶ Tablespaces for Infocubes
- ▶ Tablespaces for aggregates

It is important to highlight that different tablespaces for each of the groups listed were created over different node groups to ensure a well-balanced scenario from data distribution and usage points of view.

To achieve the balanced scenario, we spread these groups over our DB2 partitions 6 to 37. In Figure 3-12 we highlight a set of tablespaces used in our scenario to depict the tablespace distribution. Each tablespace inside the tablespace group had the same number of objects. Although being a difficult task to achieve in a real production environment, it is important to take some extra time in the architecture phase to ensure balance between partitions and the environment.

| Type | Tablespace Name | LPAR and DB2 partition | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|------------------|-----------------|------------------------|-------|---|---|----|----|----|----|----|-------|----|----|----|----|----|----|----|-------|----|----|----|----|----|----|----|-------|----|----|----|----|----|----|----|--|--|--|
| | | LPAR0 | LPAR1 | | | | | | | | LPAR2 | | | | | | | | LPAR3 | | | | | | | | LPAR4 | | | | | | | | | | |
| | | 0 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | | | |
| ODS | YMODSD01 | | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | | | |
| | YMODSD02 | | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | | | | |
| PSA | YPPSAD01 | | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | | | |
| | YPPSAD02 | | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | | | | |
| Infocubes Fact | YMFACD01 | | 6 | | | | 10 | | | | 14 | | | | 18 | | | | 22 | | | | 26 | | | | 30 | | | | 34 | | | | | | |
| | YMFACD02 | | | 7 | | | | 11 | | | | 15 | | | 19 | | | | | 23 | | | 27 | | | | | 31 | | | 35 | | | | | | |
| | YMFACD03 | | | | 8 | | | | 12 | | | | 16 | | | 20 | | | | | 24 | | | 28 | | | | | 32 | | | 36 | | | | | |
| | YMFACD04 | | | | | 9 | | | | 13 | | | | 17 | | | 21 | | | | | 25 | | | 29 | | | | | 33 | | | 37 | | | | |
| Aggregate Tables | YMAGGRD04 | | 6 | | | | 10 | | | | 14 | | | | 18 | | | | 22 | | | | 26 | | | | 30 | | | | 34 | | | | | | |
| | YMAGGRD05 | | | 7 | | | | 11 | | | | 15 | | | 19 | | | | | 23 | | | 27 | | | | | 31 | | | 35 | | | | | | |
| | YMAGGRD06 | | | | 8 | | | | 12 | | | | 16 | | | 20 | | | | | 24 | | | 28 | | | | | 32 | | | 36 | | | | | |
| | YMAGGRD07 | | | | | 9 | | | | 13 | | | | 17 | | | 21 | | | | | 25 | | | 29 | | | | | 33 | | | 37 | | | | |

Figure 3-12 Group of tablespaces and LPARs

In our case, having four distinct node group patterns for aggregates and InfoCubes and two node group patterns for PSA and ODS was enough because it is a controlled environment and the workload is highly predictable. Normally, there are two concerns when you consider balancing a DB2 partitioned database:

- ▶ Processing
- ▶ Space

Processing and space balancing are main design concerns because they may be tight together or they may be totally apart. When you have a good design, you use at least one partition that could extract most of the server (or LPAR) resources. As you would normally spread file systems (or specific raw devices per partition) to tighten the other end to balance I/O, even if only reaching one DB2 partition per server and extracting most of the server's resources, if you reach only one node group you may unbalance it from the space usage point of view, consequently voiding all your efforts to balance I/O.

The higher the number of partitions in a node group, the lower the risk of having the processing balancing affected. Why is this, though you used different node group patterns for PSA and ODS than the ones used for InfoCubes fact tables and aggregates? As most things in performance and tuning, the option is based on a trade off: if you chose to have a higher number of partitions in your node group you will also get a more difficult maintenance solution.

In our test, for each InfoCube, two specific tablespaces were set: data and indexes. Sharing the same tablespace between objects, and dropping and creating infocubes on the fly quite often would make the high watermark value totally unpredictable and could affect other tasks like backups and restores. Also, there are extra costs for maintenance.

For all tablespaces, each node has four containers created on a separate file system. In the example shown in Figure 3-13, highlighting tablespace YMFACT21, 32 files can be seen, spread over eight DB2 partitions. A column shows the percentage of sequentiality. It does not refer to anything inside DB2, but it shows how it is stored on the logical volume. This also shows whether the file is fragmented. In our case, such high sequentiality is only possible because we are not under production stress tasks that would require resizes of the tablespaces, which might cause a fragmentation of the file.

| Container (File) | Filename | Tablespace | Partition | Sequentiality (%) |
|---|------------------------|------------|-----------|-------------------|
| /db2/EB8/sapdata1/NODE0006/YMFACTD21.container000 | YMFACTD21.container000 | YMFACTD21 | 6 | 100.00% |
| /db2/EB8/sapdata2/NODE0006/YMFACTD21.container001 | YMFACTD21.container001 | YMFACTD21 | 6 | 100.00% |
| /db2/EB8/sapdata3/NODE0006/YMFACTD21.container002 | YMFACTD21.container002 | YMFACTD21 | 6 | 100.00% |
| /db2/EB8/sapdata4/NODE0006/YMFACTD21.container003 | YMFACTD21.container003 | YMFACTD21 | 6 | 100.00% |
| /db2/EB8/sapdata1/NODE0010/YMFACTD21.container000 | YMFACTD21.container000 | YMFACTD21 | 10 | 100.00% |
| /db2/EB8/sapdata2/NODE0010/YMFACTD21.container001 | YMFACTD21.container001 | YMFACTD21 | 10 | 100.00% |
| /db2/EB8/sapdata3/NODE0010/YMFACTD21.container002 | YMFACTD21.container002 | YMFACTD21 | 10 | 100.00% |
| /db2/EB8/sapdata4/NODE0010/YMFACTD21.container003 | YMFACTD21.container003 | YMFACTD21 | 10 | 100.00% |
| /db2/EB8/sapdata1/NODE0014/YMFACTD21.container000 | YMFACTD21.container000 | YMFACTD21 | 14 | 100.00% |
| /db2/EB8/sapdata2/NODE0014/YMFACTD21.container001 | YMFACTD21.container001 | YMFACTD21 | 14 | 100.00% |
| /db2/EB8/sapdata3/NODE0014/YMFACTD21.container002 | YMFACTD21.container002 | YMFACTD21 | 14 | 100.00% |
| /db2/EB8/sapdata4/NODE0014/YMFACTD21.container003 | YMFACTD21.container003 | YMFACTD21 | 14 | 100.00% |
| /db2/EB8/sapdata1/NODE0018/YMFACTD21.container000 | YMFACTD21.container000 | YMFACTD21 | 18 | 100.00% |
| /db2/EB8/sapdata2/NODE0018/YMFACTD21.container001 | YMFACTD21.container001 | YMFACTD21 | 18 | 100.00% |
| /db2/EB8/sapdata3/NODE0018/YMFACTD21.container002 | YMFACTD21.container002 | YMFACTD21 | 18 | 100.00% |
| /db2/EB8/sapdata4/NODE0018/YMFACTD21.container003 | YMFACTD21.container003 | YMFACTD21 | 18 | 100.00% |
| /db2/EB8/sapdata1/NODE0022/YMFACTD21.container000 | YMFACTD21.container000 | YMFACTD21 | 22 | 100.00% |
| /db2/EB8/sapdata2/NODE0022/YMFACTD21.container001 | YMFACTD21.container001 | YMFACTD21 | 22 | 100.00% |
| /db2/EB8/sapdata3/NODE0022/YMFACTD21.container002 | YMFACTD21.container002 | YMFACTD21 | 22 | 100.00% |
| /db2/EB8/sapdata4/NODE0022/YMFACTD21.container003 | YMFACTD21.container003 | YMFACTD21 | 22 | 100.00% |
| /db2/EB8/sapdata1/NODE0026/YMFACTD21.container000 | YMFACTD21.container000 | YMFACTD21 | 26 | 100.00% |
| /db2/EB8/sapdata2/NODE0026/YMFACTD21.container001 | YMFACTD21.container001 | YMFACTD21 | 26 | 100.00% |
| /db2/EB8/sapdata3/NODE0026/YMFACTD21.container002 | YMFACTD21.container002 | YMFACTD21 | 26 | 100.00% |
| /db2/EB8/sapdata4/NODE0026/YMFACTD21.container003 | YMFACTD21.container003 | YMFACTD21 | 26 | 100.00% |
| /db2/EB8/sapdata1/NODE0030/YMFACTD21.container000 | YMFACTD21.container000 | YMFACTD21 | 30 | 100.00% |
| /db2/EB8/sapdata2/NODE0030/YMFACTD21.container001 | YMFACTD21.container001 | YMFACTD21 | 30 | 100.00% |
| /db2/EB8/sapdata3/NODE0030/YMFACTD21.container002 | YMFACTD21.container002 | YMFACTD21 | 30 | 100.00% |
| /db2/EB8/sapdata4/NODE0030/YMFACTD21.container003 | YMFACTD21.container003 | YMFACTD21 | 30 | 100.00% |
| /db2/EB8/sapdata1/NODE0034/YMFACTD21.container000 | YMFACTD21.container000 | YMFACTD21 | 34 | 100.00% |
| /db2/EB8/sapdata2/NODE0034/YMFACTD21.container001 | YMFACTD21.container001 | YMFACTD21 | 34 | 100.00% |
| /db2/EB8/sapdata3/NODE0034/YMFACTD21.container002 | YMFACTD21.container002 | YMFACTD21 | 34 | 100.00% |
| /db2/EB8/sapdata4/NODE0034/YMFACTD21.container003 | YMFACTD21.container003 | YMFACTD21 | 34 | 100.00% |

Figure 3-13 YMFACT21 tablespace containers and sequentiality

An InfoCube fact table or aggregates tablespace would have 32 containers. This would be slightly different for ODS and PSA objects since they are spread over 16 partitions, resulting in a total of 48 containers.

Tip: To check fragmentation of a file on AIX use the command `fileplace`. You need root authority to do so.

Extent and prefetch size

The extent size for a tablespace represents the number of pages of data that will be written to a container before data is written to the next container. When selecting an extent size, you should consider:

- ▶ The size and type of tables in the tablespace.
- ▶ For DMS tablespaces, the space is allocated to a table one extent at a time. As the table is populated and an extent becomes full, a new extent is allocated. DMS tablespace container storage is pre-reserved, which means that new extents are allocated until the container is completely used.

Reading several consecutive pages into the buffer pool using a single I/O operation can greatly reduce application overhead. In addition, multiple parallel I/O operations to read several ranges of pages into the buffer pool can help reduce I/O wait time.

Prefetching starts when the database manager determines that sequential I/O is appropriate and will improve performance. In cases like table scans and table sorts, the database manager can easily determine that sequential prefetch will improve I/O performance. In these cases, the database manager automatically starts sequential prefetch.

It is a good practice to explicitly set the PREFETCHSIZE value as a multiple of the number of tablespace containers or the number of physical disks under each container (if a RAID device is used) based on the EXTENTSIZE value for your tablespace. In our case, where the extent size is eight pages and each tablespace has four files, the prefetch quantity can be set to at least 32 pages. As we had an array of seven physical disks per container, you could set the prefetch quantity to 256 pages. In our case, we considered a high amount of data and decided to stick only with the number for containers.

Disk layout

When reading data from or writing data to tablespace containers, DB2 (for Linux, UNIX, and Windows) may use parallel I/O if the number of containers in the database is greater than one. However, there are situations when it would be beneficial to have parallel I/O enabled for single container tablespaces. For example, if the container is created on a single RAID device that is composed of more than one physical disks, you may want to issue parallel read and write calls. This is our case, and for each file, a storage data array was set.

To force parallel I/O for a tablespace that has a single container, you can use the DB2_PARALLEL_IO registry variable. This variable was set to an asterisk (*), meaning every tablespace. This variable was set in our environment, as shown in Example 3-19.

Example 3-19 Setting parallel I/O in our environment

```
db2set DB2_PARALLEL_IO=*
```

In SYS2, on the 16 TB database, we configured only one DS8300 storage sub-system. For SYS1, on the larger 60 TB database, we configured four DS8300 storage sub-systems.

For further details about how the disk layout and the DS8300 storage sub-system were configured in our solution, refer to Chapter 5, “The system storage perspective” on page 317.

Balancing tablespace layout and disk layout

The task of balancing I/O may be time intensive. As storage and databases gets renewed, new techniques and self-improvements are created to help with the design and maintenance of such solutions.

As an example in the following, we pick one tablespace, YMFACTD21, to get into some details about the balancing between the tablespace and disk.

As discussed before, SYS1 was generated from SYS2 and some tests were executed in SYS2 before generating SYS1 to fine-tune and improve the design created in the first phase, and a design change was done on the AIX disk layout.

The lowest space allocation granularity in an AIX Volume Group (VG) is a physical partition (PP). This is observable through an `lsvg` command, as shown in Figure 3-14. In our case data VGs were set with a PP of 64 MB.

```

$lsvg rootvg
VOLUME GROUP:      rootvg                VG IDENTIFIER: 000c42bd00004c0000000111470521b6
VG STATE:          active                 PP SIZE:       32 megabyte(s)
VG PERMISSION:    read/write             TOTAL PPs:     2168 (69376 megabytes)
MAX LVs:          256                    FREE PPs:     550 (17600 megabytes)
LVs:              11                     USED PPs:     1618 (51776 megabytes)
OPEN LVs:         10                    QUORUM:        1
TOTAL PVs:        4                     VG DESCRIPTORS: 4
STALE PVs:        0                     STALE PPs:    0
ACTIVE PVs:       4                     AUTO ON:       no
MAX PPs per VG:  32512                  MAX PVs:       32
MAX PPs per PV:  1016                   AUTO SYNC:     no
LTG size:         128 kilobyte(s)       BB POLICY:     relocatable
HOT SPARE:        no
  
```

Figure 3-14 `lsvg` with the physical partition size highlighted

As a macro view and infrastructure perspective, all disk arrays for all DB2 partitions would use the storage sub-systems in SYS2 because only one DS8300 was used on this environment, as follows: DS1: one DS8300 used for DB2 partitions 0 to 5 in AIX LPAR sys1db0, DB2 partitions 6 to 13 in AIX LPAR sys1db1, DB2 partitions 14 to 21 in AIX LPAR sys1db2, DB2 partitions 22 to 29 in AIX LPAR sys1db3 and DB2 partitions 30 to 37 in AIX LPAR sys1db4.

In SYS2, each AIX file system and, consequently, each logical volumes was created in 16 disks spreading over four disk arrays in the storage, as shown in Figure 3-15. Disk numbers and container names are just examples to easily understand the design. This configuration shows tablespace containers for one DB2 partition, since each tablespace partition would have four containers. All remaining partitions and tablespaces would have the same pattern.

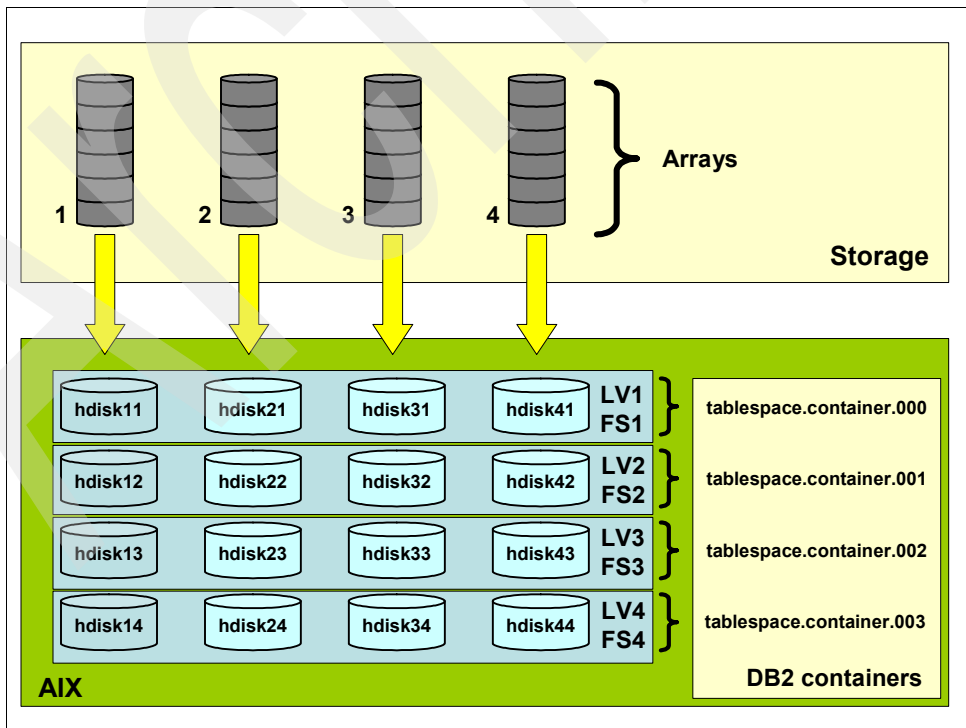


Figure 3-15 SYS2 storage and AIX layout for DB2 containers

Looking at the picture, you may not notice any issue or problem with this configuration. However, after designing this layout, we identified that it was not balancing I/O evenly as expected.

DB2 data LVs were configured to use Logical Volume Manager (LVM) spreading. If you select the minimum inter-disk setting (range = minimum), the PPs assigned to the LV are located on a single disk. If you select the maximum inter-disk setting (range = maximum), PPs are located on multiple disks to enhance performance, and this was the option used.

The maximum setting, considering other constraints, spreads the physical partitions of the logical volume as evenly as possible over as many physical volumes as possible. This is a performance-oriented option, because spreading the physical partitions over several disks tends to decrease the average access time for the logical volume.

Have you seen the gap? Let us go further on the analysis. The issue can be better seen when we cross AIX and DB2 perspectives. DMS tablespaces stripe data extents across containers. It will write one extent per container until it moves to the next one.

The problem with the environment was the over stripping. As data was spread in one PP granularity (64 MB) across disks, DB2 would also stripe data across containers, resulting in a scenario in which it would only access one array, as shown in Table 3-16. As shown in the picture, array 4 would be only accessed due to the LVM spreading. hdisk41, hdisk42, hdisk43, and hdisk44 would host all four containers for a specific DB2 partition. In the DB2 perspective, it would spread extents and pages across its containers but it would not have any effect, since all of them are reaching the same disk array.

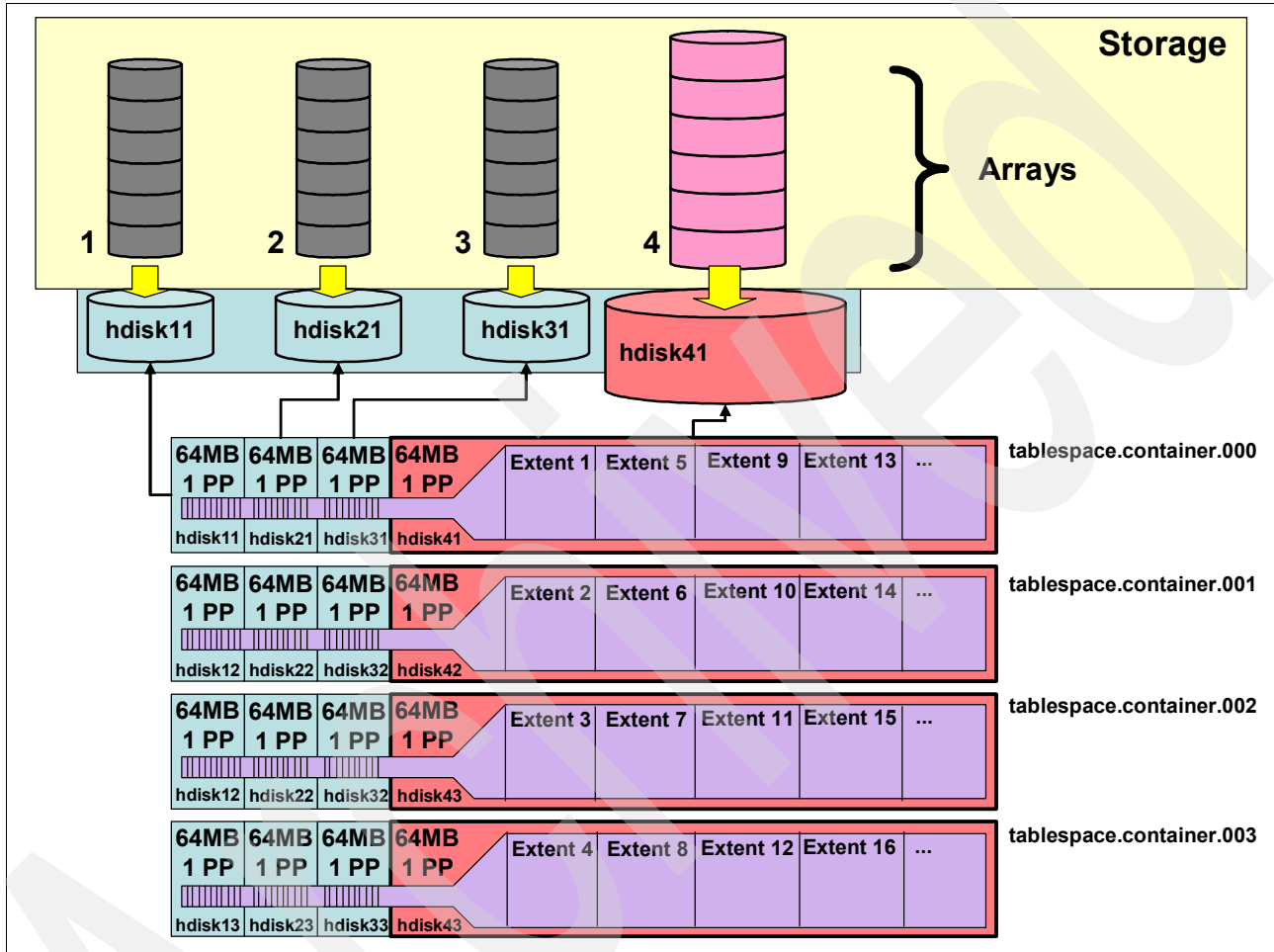


Figure 3-16 SYS2 over stripping issue

Based on this design and identified issue, an improved scenario was designed for SYS1 and the 60 TB database.

Important: Even briefly describing storage parts from the DB2 perspective, at this point, we strongly recommend that you take some time to read and understand the storage layout in Chapter 5, “The system storage perspective” on page 317, for a better understanding of the SYS1 disk layout.

Most of the configuration for SYS1 remained the same. The only change executed was to accommodate the higher number of DS8300 storage sub-systems.

From a macro perspective and infrastructure point of view, all arrays for a specific DB2 partition would be allocated in the same storage sub-system as they were split over DB2 AIX LPARs. The four storage sub-system DS8300 balancing in SYS1 was based on the following:

- ▶ DS1: one DS8300 used for DB2 partitions 0 to 5 in AIX LPAR sys1db0 and DB2 partitions 6 to 13 in AIX LPAR sys1db1
- ▶ DS2: one DS8300 used for DB2 partitions 14 to 21 in AIX LPAR sys1db2
- ▶ DS3: one DS8300 used for DB2 partitions 22 to 29 in AIX LPAR sys1db3
- ▶ DS4: one DS8300 used for DB2 partitions 30 to 37 in AIX LPAR sys1db4

The major change on the disk layout was to make sure that we would not get into the problem we had on SYS2. To fix that, other than using 16 disks from four disk arrays, we designed the solution with eight disks from eight disk arrays. Also, a change was executed on how the file system spreading would occur to accommodate only two disks other than four. This is shown in Figure 3-17. As you can see in the figure, eight disk arrays were created in the storage from which four LVs and file systems were created. The major difference is that the spreading is now against two arrays per DB2 container, allowing each container to work on at least one array. The figure only highlights four files, which would be the number of containers for one tablespace per partition, therefore being reasonable to scale this out for each tablespace and its partition in the database.

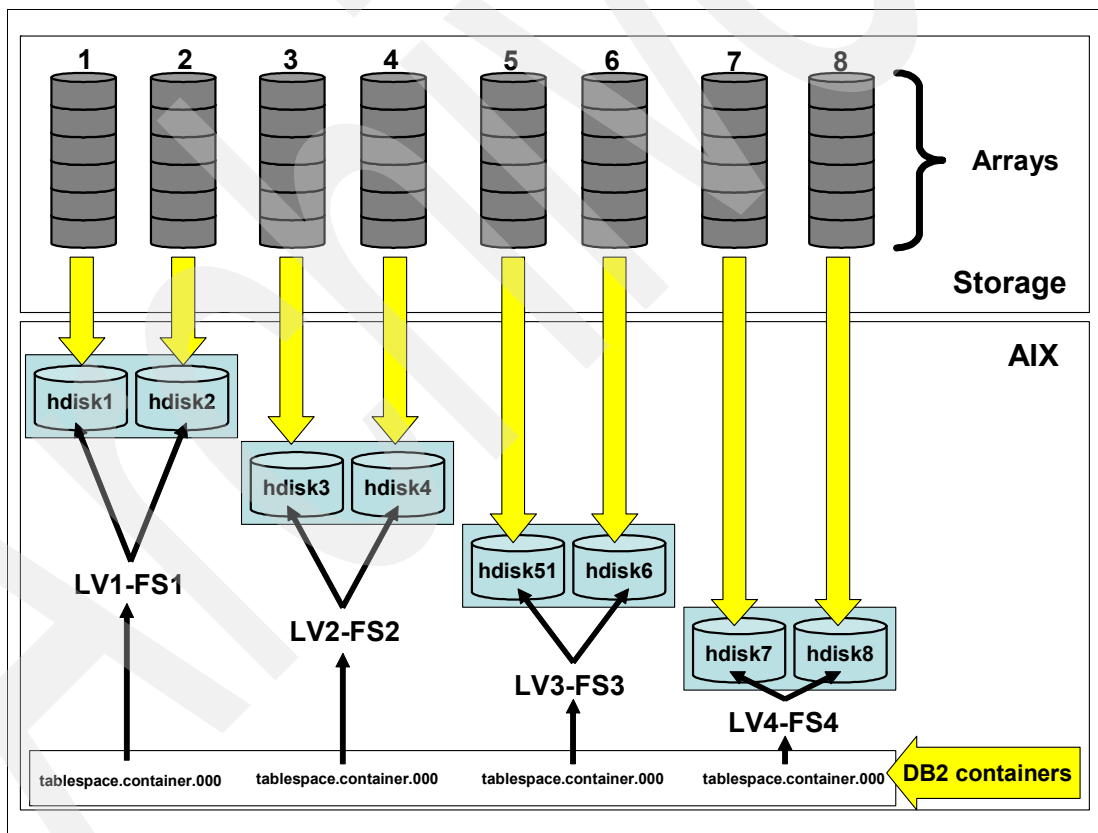


Figure 3-17 LVM file systems spreading for DB2 containers

As you may have noticed, having a larger number of disk arrays and a lower number of disks available on AIX allowed us to balance I/O in a better way. The DB2 data would be striped as depicted in Figure 3-18. In this scenario, optimized I/O balancing since tablespace extents are being spread over different disk arrays. The logical volume spreading was the same as that used in SYS1 spreading 1 PP (64 MB) per disk on each logical volume and file system.

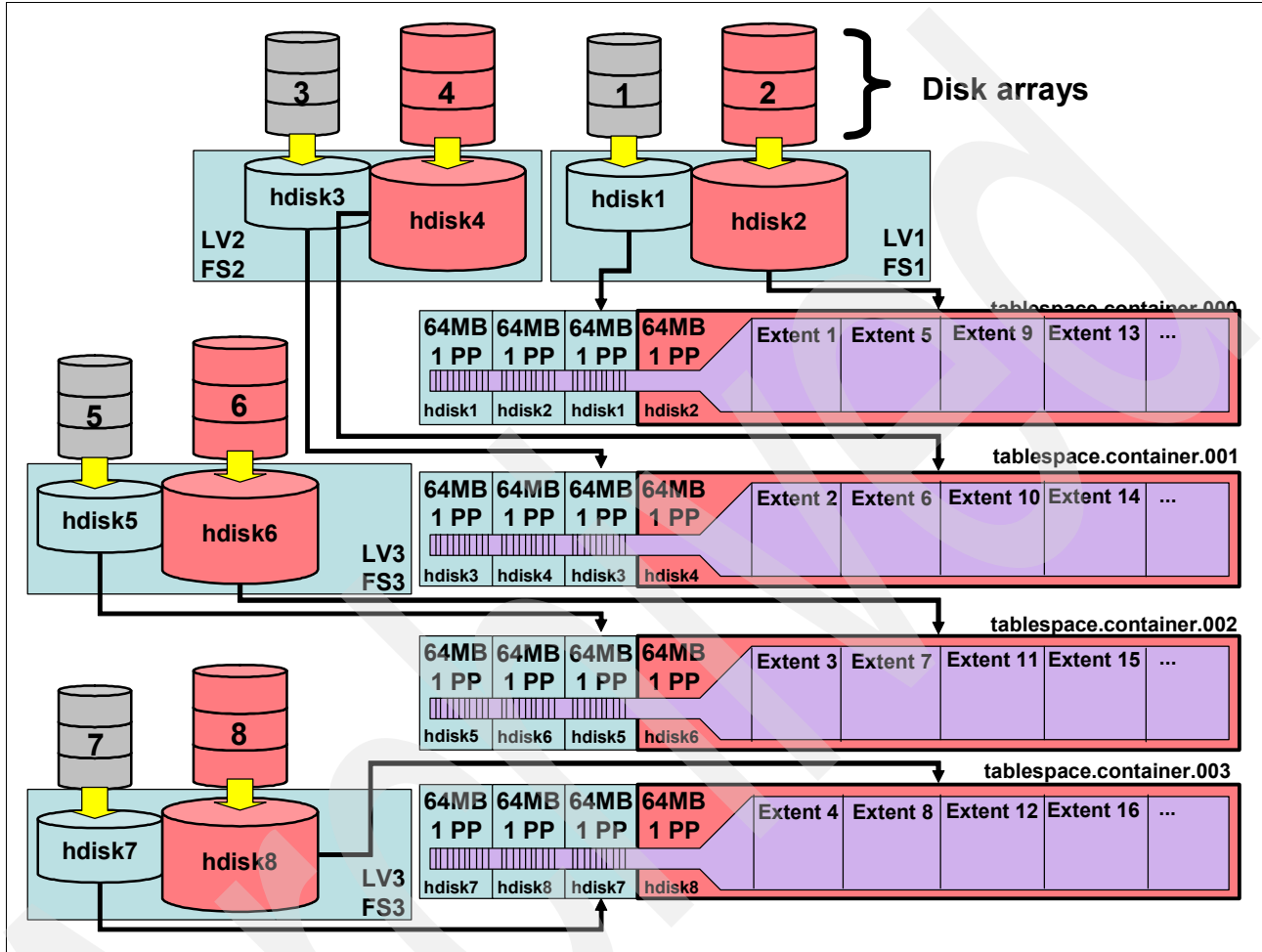


Figure 3-18 SYS1 spreading and DB2 extents

3.2.7 Object distribution across partitions

In our environment, DB2 partition 0 was used to store SAP NetWeaver BI temporary tables, system catalog tables, and dimension tables. The remaining 32 DB2 partitions were used to balance InfoCubes fact tables, aggregate tables, PSAs, and ODS objects.

A pattern was created to balance evenly processing and disk usage, as shown in Table 3-8, and explained in the next sections.

Table 3-8 Pattern used for balancing evenly disk usage

| Object | Number of DB2 partitions |
|------------------------------|--------------------------|
| ODS | 16 |
| InfoCube aggregate table | 8 |
| InfoCube fact table | 8 |
| PSA for fact tables and ODSs | 16 |

To make it easier to understand the design and balancing of SAP BW objects across DB2 partitions, we describe only a set of objects created: 100 InfoCubes (fact tables and aggregate tables), and a small set of ODS and PSA objects. All remaining objects were created following the same pattern.

Operational data stores and persistent staging area

The operational data stores (ODSs) and persistent staging data (PSA) tables were spread over 16 DB2 partitions with two distinct DB2 node groups, the first one starting in DB2 partition six and the second starting in DB2 partition 7. Two tablespaces and two index spaces for each were created (we use ODS as an example, but this same pattern was used for PSA):

- ▶ Data tablespaces
 - YMODSD01
 - YMODSD02
- ▶ Index tablespaces
 - YMODSID01
 - YMODSID02

The DB2 partition group and nodes for these tablespaces used as examples are shown in Example 3-9. The only restriction was to match node groups for data tablespace and index tablespace.

Table 3-9 ODS distribution

| Tablespace | Nodegroup | Partitions |
|------------|-----------|--|
| YMODSD01 | NG_YMO01 | 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36 |
| YMODSI01 | NG_YMO01 | 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36 |
| YMODSD02 | NG_YMO02 | 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37 |
| YMODSI02 | NG_YMO02 | 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37 |

To better understand how ODS objects were spread over DB2 partitions, we show the distribution for the ones used in the previous example, as shown in Figure 3-19. All ODS objects follow the same pattern as the ones illustrated in the figure.

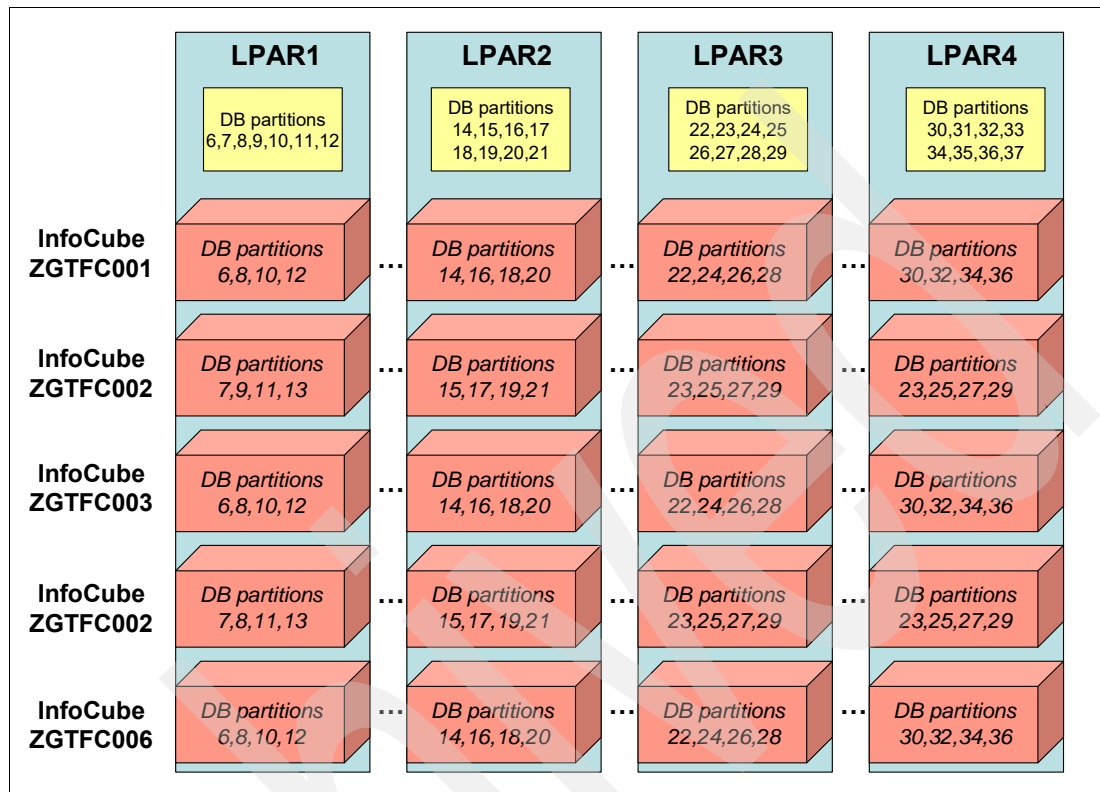


Figure 3-19 ODS data and index structure

The table name for active rows for ODS in SAP BW follows the naming pattern described in Example 3-20.

Example 3-20 ODS naming pattern

Naming pattern:
/BIC/A<ODS_NAME>00

where:

<ODS_NAME> is the ODS name

for example, the active rows table for ODS ZGTFC001 would be:

/BIC/AZGTFC00100

Figure 3-20 shows the balancing for ODS ZGTF001. The distribution starts at DB2 partition 6 and goes through all the even partitions. The same figure could be drawn for the odd partitions.

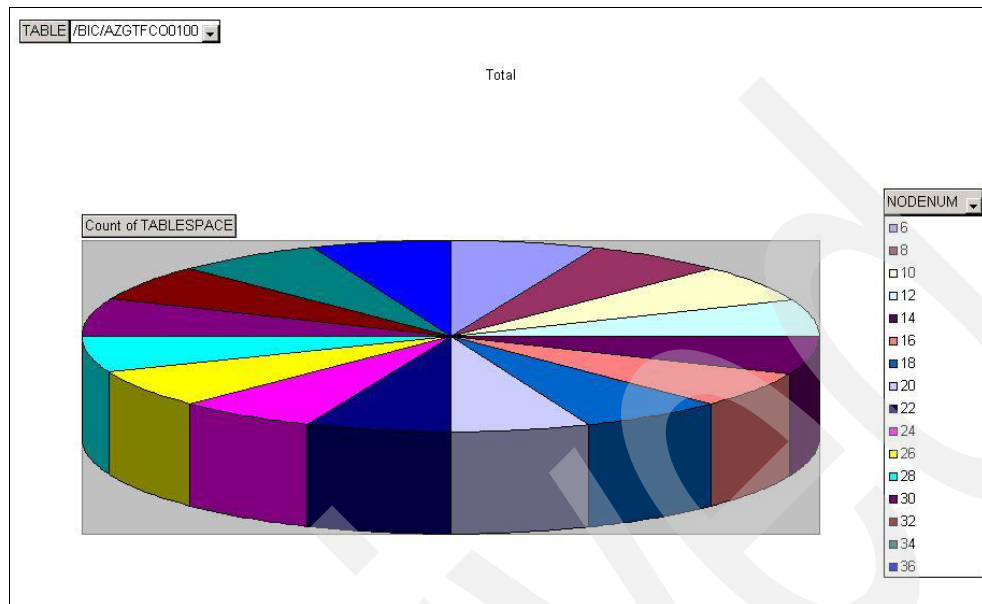


Figure 3-20 ODS table spread over DB2 partitions in node group NG_YMO01

InfoCubes fact tables

The InfoCubes fact table balancing was spread linearly over four distinct DB2 partition group patterns. In this structure, the division of the set we chose, 100 InfoCubes by four partition groups, was designed as follow:

- ▶ DB2 partition pattern 1 - 26 InfoCubes
- ▶ DB2 partition pattern 1 - 26 InfoCubes
- ▶ DB2 partition pattern 1 - 24 InfoCubes
- ▶ DB2 partition pattern 1 - 24 InfoCubes

In Figure 3-21, we highlight the distribution of each InfoCube fact tablespace on the first partition group pattern that includes DB2 partition 6, 10, 14, 18, 22, 26, 30, and 34. In this partition group we spread 26 InfoCubes.

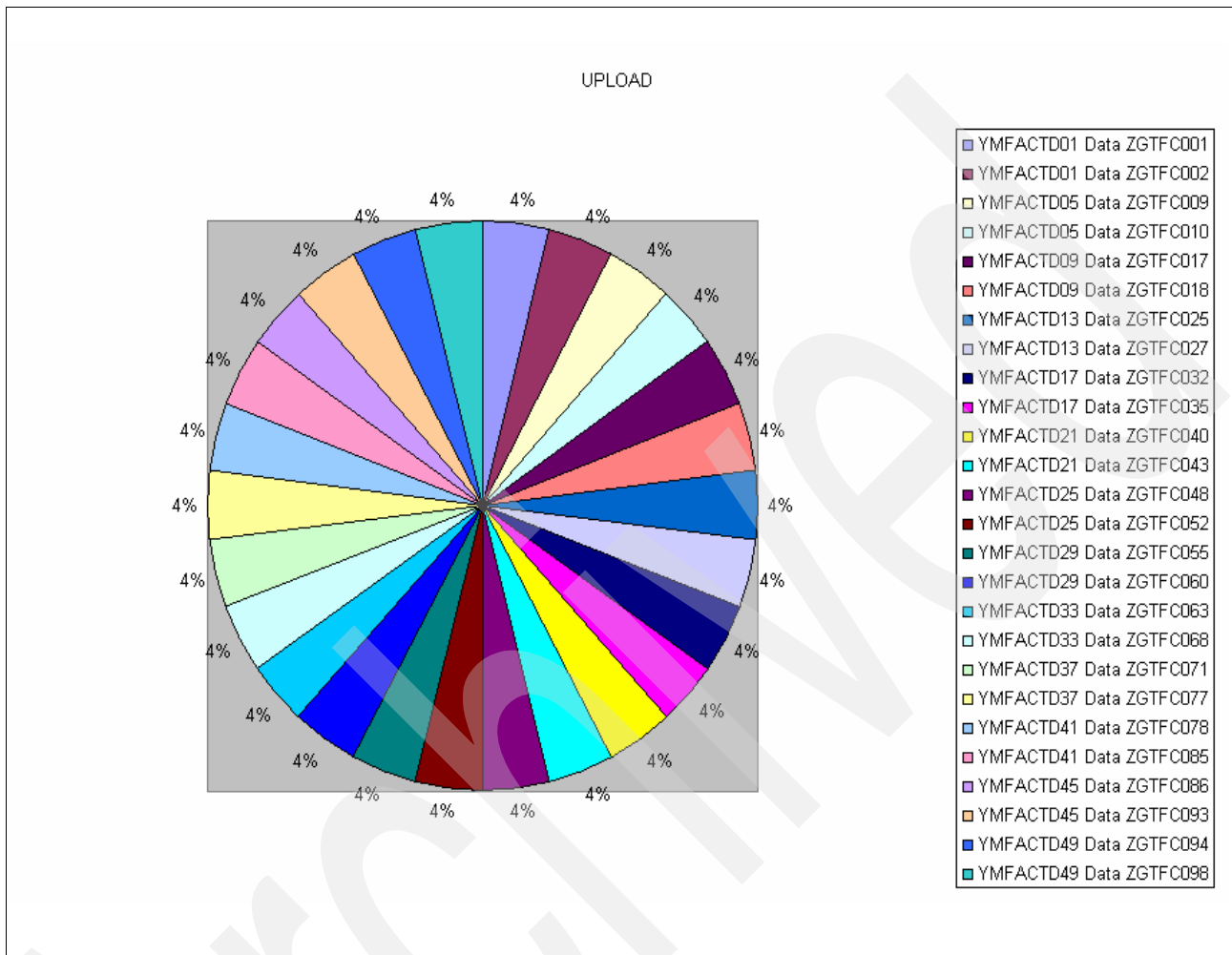


Figure 3-21 26 InfoCubes fact tablespaces distributed in DB2 partition 6, 10, 14, 18, 22, 26, 30, and 34

Identical figures could be shown for all the other DB2 partitions.

The number of InfoCubes per tablespace was evenly distributed, with two InfoCube fact tables per tablespace, as shown in Figure 3-22. In this figure, we chose only the InfoCubes and tablespaces that were spread over the partition group pattern spread over DB2 partitions 6, 10, 14, 18, 22, 26, 30, and 34. The same distribution was done for the remaining InfoCubes and partition group patterns.

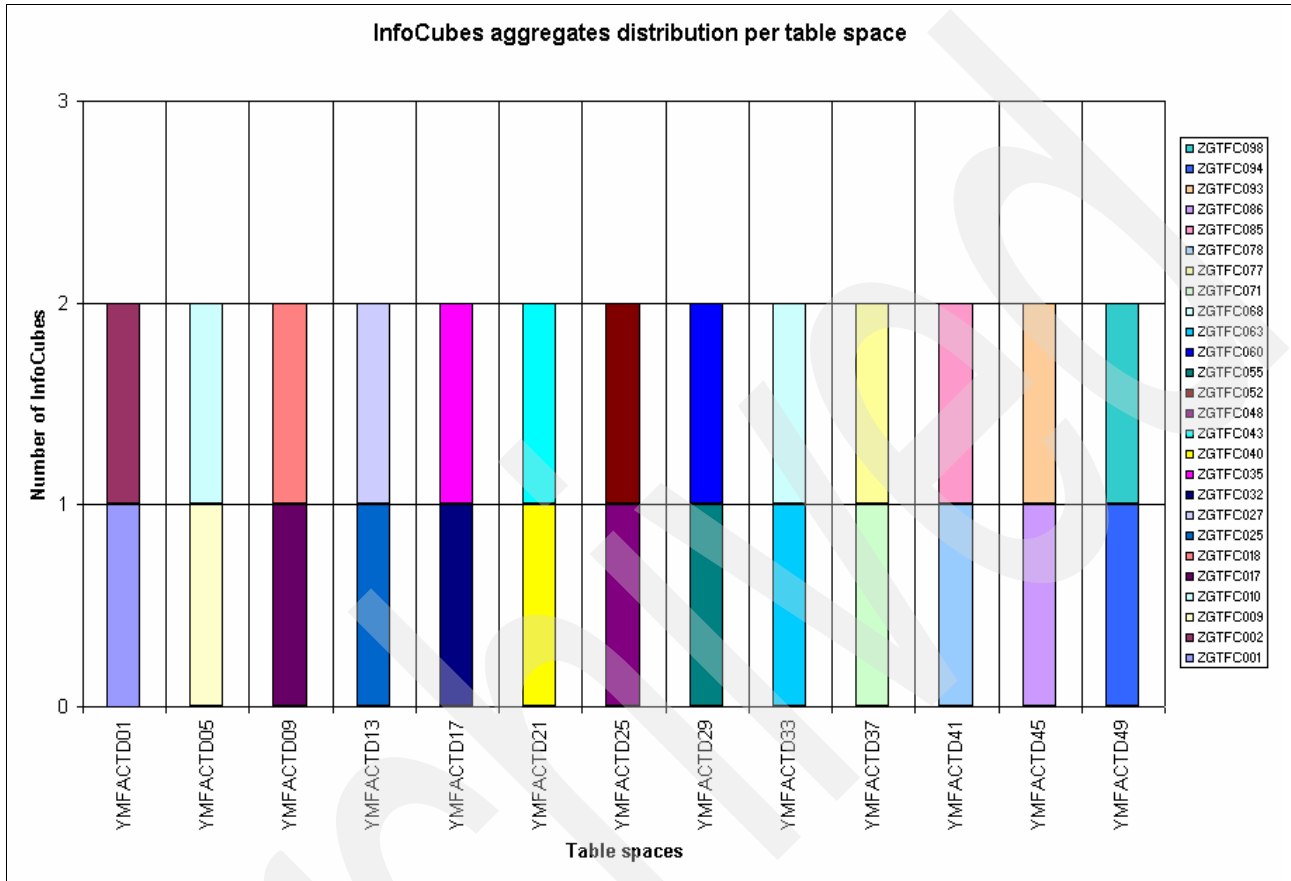


Figure 3-22 InfoCubes fact tables distribution per tablespace

InfoCubes aggregates tables

InfoCubes aggregates tables were spread over four distinct DB2 partition group patterns.

In Figure 3-23, we highlight the distribution of each InfoCube aggregates tablespace on the first partition group pattern, which includes DB2 partitions 6, 10, 14, 18, 22, 26, 30, and 34. In this partition group we spread 35 InfoCubes aggregates tables.

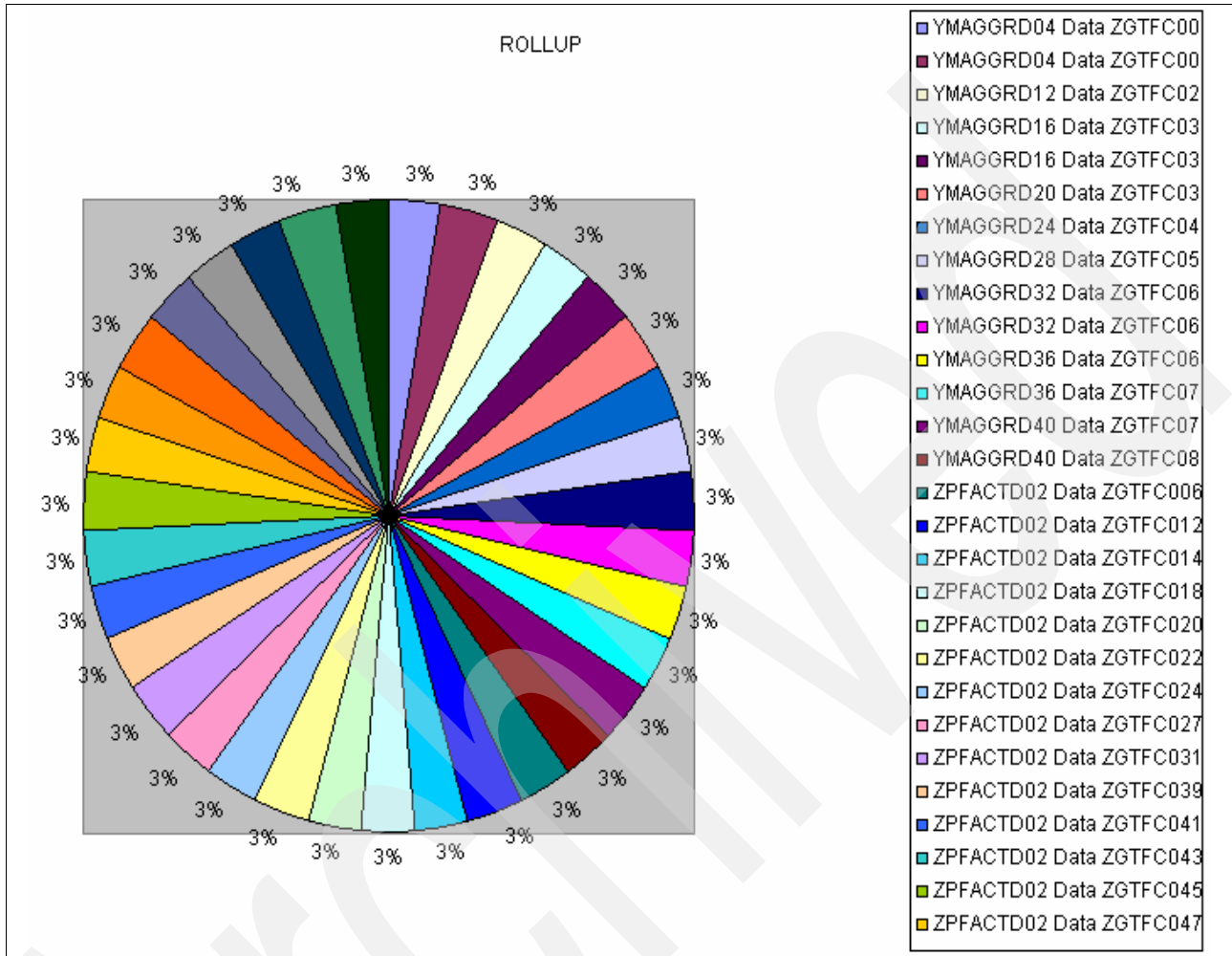


Figure 3-23 Thirty-five InfoCubes aggregates distributed in DB2 partitions 6, 10, 14, 18, 22, 26, 30, and 34

Identical figures could be drawn for the other DB2 partitions.

Table 3-10 shows summarized numbers for InfoCubes and their partition group pattern distribution.

Table 3-10 Summarized number of Infocubes aggregates per DB2 partition group pattern

| Number of InfoCubes | DB2 partition group pattern |
|---------------------|----------------------------------|
| 35 | 6, 10, 14, 18, 22, 26, 30 and 34 |
| 26 | 7, 11, 15, 19, 23, 27, 31 and 35 |
| 21 | 8, 12, 16, 20, 24, 28, 32 and 36 |
| 18 | 9, 13, 17, 21, 25, 29, 33 and 37 |

The number of InfoCubes aggregates tablespaces were spread as even as possible. This is depicted in Figure 3-24. In the figure, as an example, we extracted the configuration for InfoCubes aggregates tablespaces spread on the DB2 partition group patterns 7, 11, 15, 19, 23, 27, 31, and 35. In the figure you can see that, in some cases, the same tablespace was used by two InfoCubes.

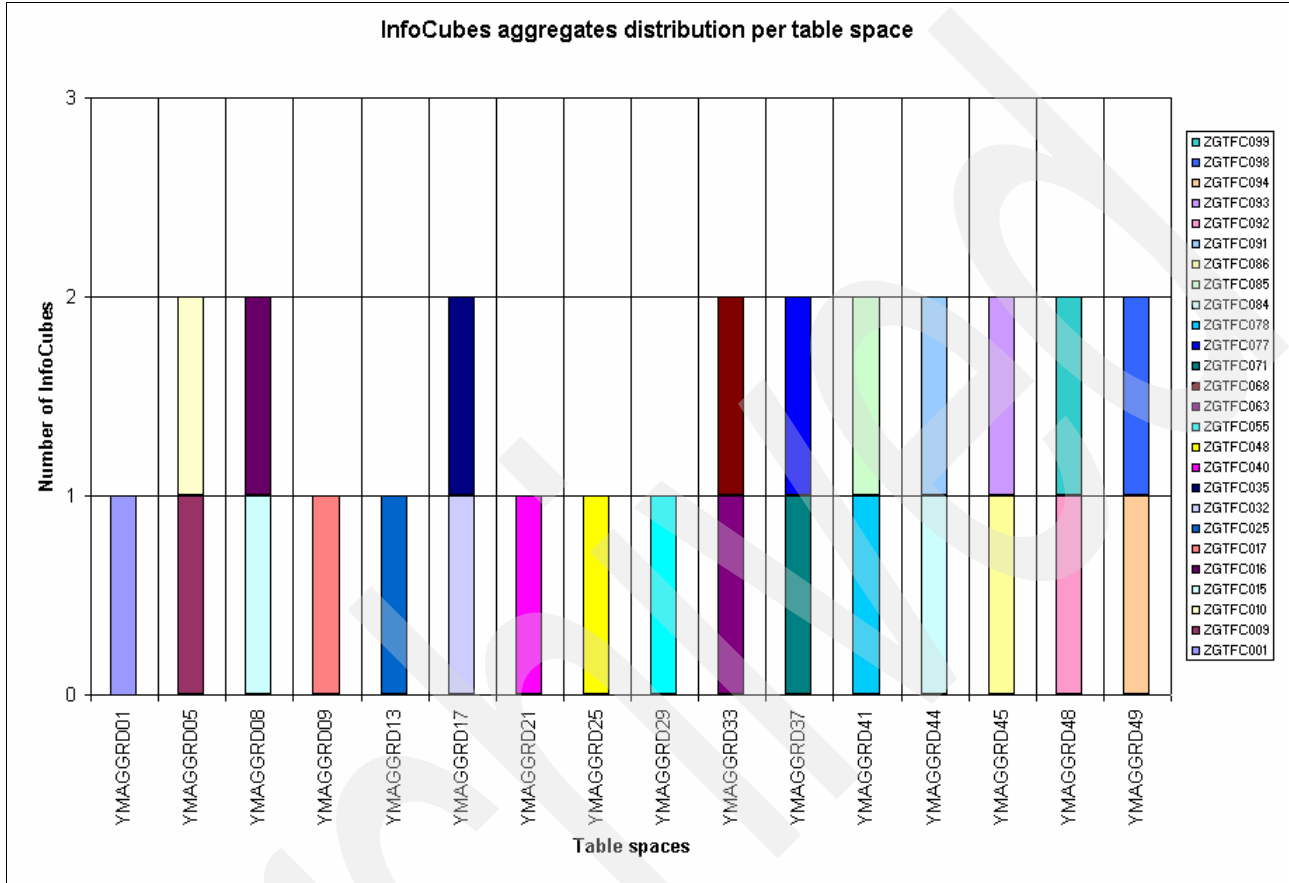


Figure 3-24 InfoCubes aggregates distribution per tablespace

3.2.8 Buffer pools

In our scenario, we split buffer pools based on SAP NetWeaver BI objects. Environments SYS1 and SYS2 differ on the database and tables size: larger buffer pools can be found in SYS1. We split the buffer pool into three major classes:

- ▶ For aggregates - BP_AGGR_16K
- ▶ For dimensions - BP_DIM_16K
- ▶ For all remaining objects (InfoCubes fact tablespace, PSA tablespaces, and ODS) - BP_STD_16K

An additional buffer pool, called IBMDEFAULTBP with a page size of 4 KB, was used for catalog tables.

We detail their sizes and partitions in the following sections.

IBMDEFAULTBP

This is a DB2 default buffer pool and was used for the catalog tablespace. It had a page size of 4,096 bytes, and the size and distribution for SYS1 and SYS2 is shown in Figure 3-25.

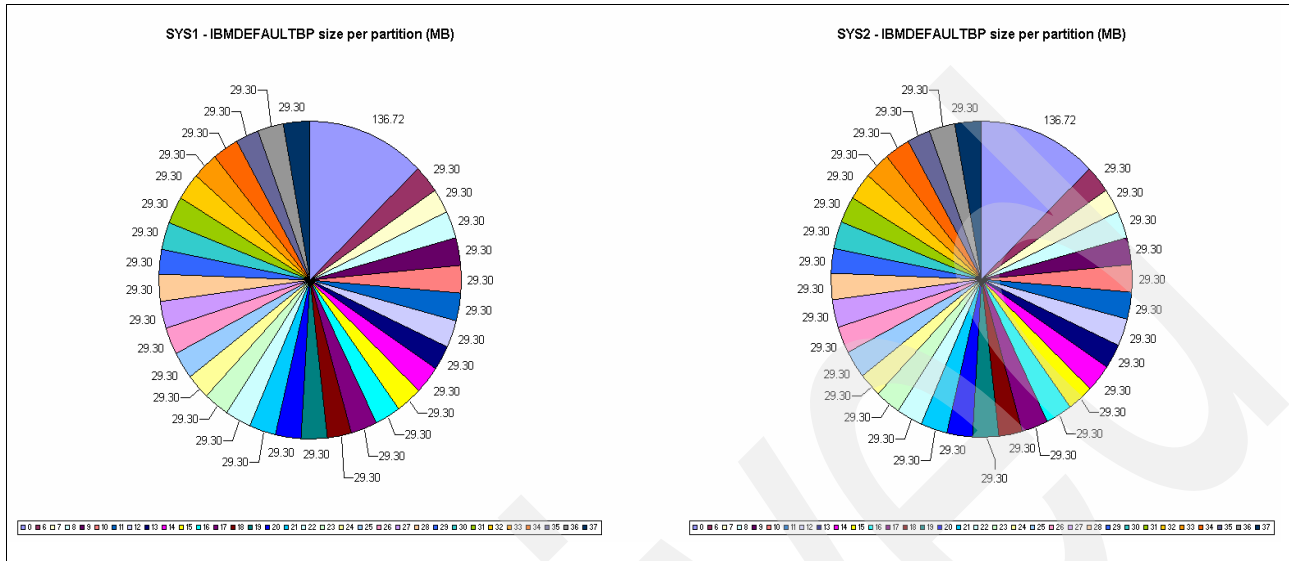


Figure 3-25 IBMDEFAULTBP sizes per DB2 partitions for SYS1 and SYS2

BP_AGGR_16K

BP_AGGR_16K was used for InfoCubes aggregates tablespaces. They were spread over DB2 partitions 6 to 37 and a small piece on DB2 Partition 0, which can be disregarded since there were no aggregates tablespaces on it. BP_AGGR_16K size and distribution for SYS1 and SYS2 is depicted in Figure 3-26. As you may notice, for SYS1 they were set to 225,000 pages of 16 KB, a total of 3.6 GB per partition. For SYS2, they were slightly smaller, being set to 180,000 pages of 16 KB, a total of 2.8 GB.

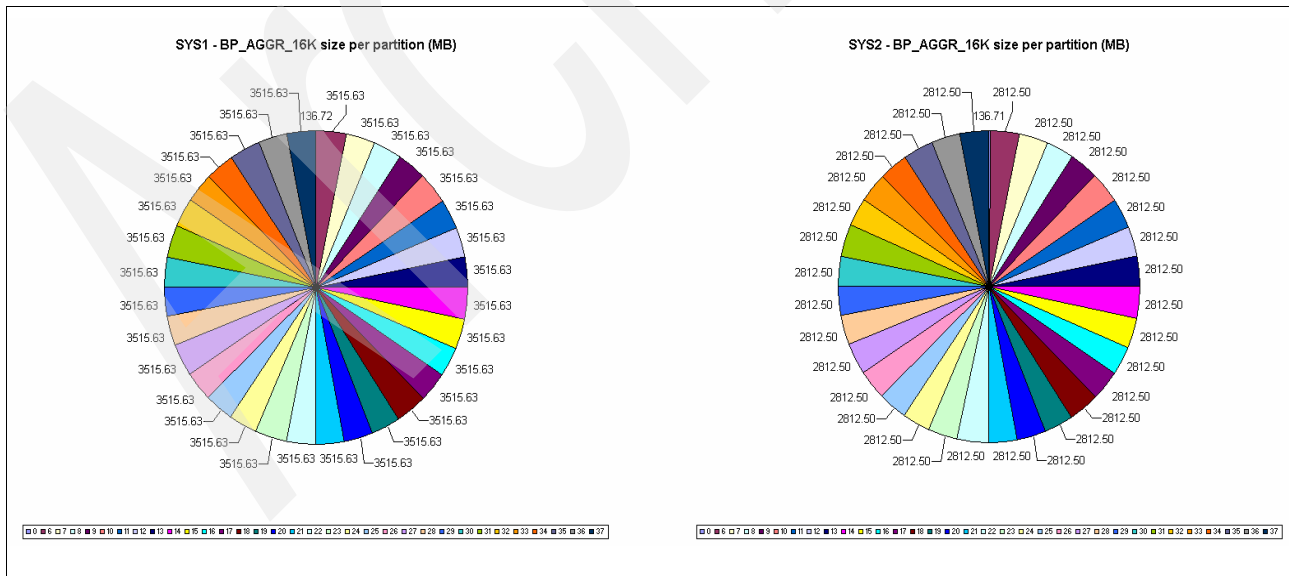


Figure 3-26 BP_AGGR_16K sizes per DB2 partitions for SYS1 and SYS2

BP_DIM_16K

This buffer pool was only created on DB2 partition 0. This is because of the SAP NetWeaver BI design and since it was used only dimension tablespaces. For SYS1, it was set to 312,500 pages of 16 KB, a total of 4.8 GB. For SYS2, it was set to 43750 pages of 16 KB, a total of 680 MB.

BP_STD_16K

BP_STD_16K was used for InfoCubes fact tablespaces, PSA, and ODS tablespaces. This buffer pool was spread over DB2 partitions 6 to 37. BP_STD_16K size and distribution for SYS1 and SYS2 is depicted in Figure 3-27. For SYS1 they were set to 2,000,000 pages of 16 KB, a total of 31.25 GB per partition. For SYS2, they were significantly smaller since the largest objects were considerably smaller on SYS2, being set to 400,000 pages of 16 KB, a total of 6.25 GB.

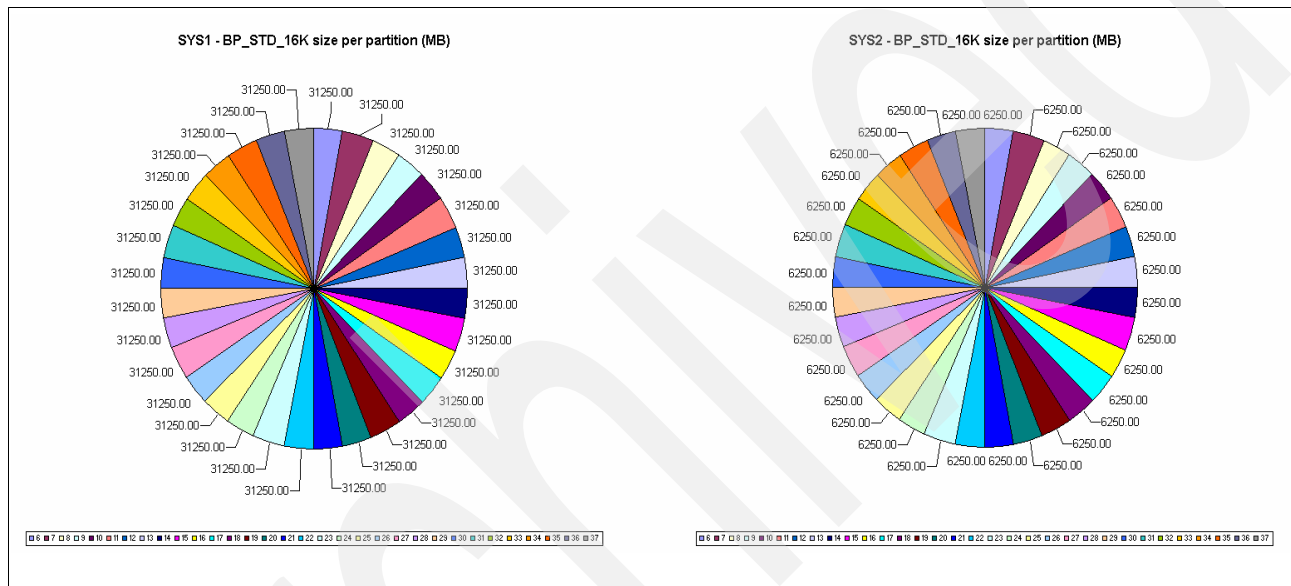


Figure 3-27 BP_STD_16K sizes per DB2 partitions for SYS1 and SYS2

3.2.9 Monitoring tools

To monitor our environment, we used both well-known tools (such as IBM DB2 Performance Expert and NMON) and other tools that were built to satisfy specific requests. This section describes the tools and scripts used to monitor our environment, and presents some of the key reports that were generated.

IBM DB2 Performance Expert

DB2 Performance Expert is software to manage performance of DB2 landscapes. It provides a server component to run on a central machine, which monitors a set of DB2 instances remotely without any agents being deployed on the monitored DB2 systems. The server maintains performance databases to hold the performance data. A separate client component is typically installed on the user's workstation. It connects to the server component to retrieve all monitoring data necessary.

The set of features of DB2 Performance Expert comprise different monitoring approaches, including real-time monitoring, history-based monitoring, performance alerting, long-term performance and trend analysis and performance visualization. DB2 Performance Expert collects not only information about the monitored DB2 system, but also about the underlying

operating system, including CPU times, memory consumption, disk I/O, storage, and processes.

We used our tests for the DB2 PE version described in Example 3-21.

Example 3-21 DB2 PE version used

```
$ pelevel
=====
IBM DB2 Performance Expert Server V2
=====
IBM (c) DB2 Performance Expert Server for Multiplatforms
IBM (c) DB2 Performance Expert Server for Workgroups
Version 2.2.1.0.419, code level U221_GAHOT-U419,R221_GAHOT-L1299,N221_GAHOT-E401
=====
```

DB2 Performance Expert monitors DB2 databases independently from the application using the database. If this monitored data can be correlated somehow to the applications running, performance issues can be narrowed to a user or functional module of the application.

The DB2 snapshot monitoring allows the retrieval of application information, and this snapshot exposes certain client attributes to identify users or transactions within the application that is connecting to the DB2 database. These available client attributes are:

- ▶ `tpmon_client_app`, which identifies the application performing the transaction
- ▶ `tpmon_client_userid`, which is the application-level client user ID that currently uses the connection
- ▶ `tpmon_client_wkstn`, which identifies the client's system or workstation.\
- ▶ `tpmon_acc_str`, which contains customizable client information from the application, passed to the target database for logging and diagnostic purposes

The SAP database support layer (dbsl) uses this information to provide the SAP-specific user and application information to DB2.

A sample instance information report from DB2 PE is displayed in Figure 3-28. This picture shows the DB2 information split in nodes in our database environment.

| Performance Counter | PART0 | PART1 | PART2 | PART3 | PART4 | PART5 | PART6 | PART7 | PART8 | PART9 |
|--|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Server instance name | db2eb8 | db2eb8 | db2eb8 | db2eb8 | db2eb8 | db2eb8 | db2eb8 | db2eb8 | db2eb8 | db2eb8 |
| Product version | 8.1.1.112 | 8.1.1.112 | 8.1.1.112 | 8.1.1.112 | 8.1.1.112 | 8.1.1.112 | 8.1.1.112 | 8.1.1.112 | 8.1.1.112 | 8.1.1.112 |
| Service level | U807381 | U807381 | U807381 | U807381 | U807381 | U807381 | U807381 | U807381 | U807381 | U807381 |
| Current connections | 370 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Attempted connections for DB2 Connect | 1,287 | 0 | 0 | 0 | 0 | 0 | 287 | 0 | 0 | 0 |
| Current connections for DB2 Connect | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Conn. waiting for host reply | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Conn. waiting for client to send request | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Remote connections to DBM | 370 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Remote connections executing in the DBM | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Local connections | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Local connections executing in the DBM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Local Databases with current connects | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Total heap allocated (pages) | 253 | 0 | 0 | 0 | 0 | 0 | 418 | 958 | 139 | 316 |
| Hash join threshold | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Post threshold sorts | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Piped sorts requested | 265,184 | 0 | 0 | 0 | 0 | 0 | 483 | 991 | 603 | 538 |
| Piped sorts accepted | 265,184 | 0 | 0 | 0 | 0 | 0 | 483 | 991 | 603 | 538 |
| Sort private heap high water mark | 214,432 | 0 | 0 | 0 | 0 | 0 | 50,415 | 33,471 | 50,831 | 32,995 |
| Piped sorts (%) | 100.00 | N/C | N/C | N/C | N/C | N/C | 100.00 | 100.00 | 100.00 | 100.00 |
| Agents registered | 399 | 6 | 6 | 6 | 6 | 6 | 81 | 72 | 73 | 56 |
| Agents waiting for token | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Maximum agents registered | 411 | 6 | 6 | 6 | 6 | 6 | 83 | 72 | 73 | 56 |
| Maximum agents waiting | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Committed private memory (bytes) | 1,769,472 | 1,572,864 | 1,572,864 | 1,572,864 | 1,572,864 | 1,572,864 | 3,899,392 | 3,751,936 | 3,735,552 | 3,047,424 |
| Agents assigned from pool | 63,416 | 2,564 | 2,564 | 2,564 | 2,564 | 2,564 | 768,839 | 252,354 | 504,231 | 163,832 |
| Agents created due to empty pool | 6,377 | 7 | 7 | 7 | 7 | 7 | 113 | 73 | 74 | 57 |
| Stolen agents | 43,966 | 0 | 0 | 0 | 0 | 0 | 6,462 | 4,450 | 7,969 | 2,881 |
| Agents created (%) | 6.09 | 0.27 | 0.27 | 0.27 | 0.27 | 0.27 | 0.01 | 0.02 | 0.01 | 0.03 |
| Maximum coordinating agents | 390 | 2 | 2 | 2 | 2 | 2 | 4 | 2 | 2 | 2 |
| Connection switches | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Total inactive DRDA agents | N/P | N/P | N/P | N/P | N/P | N/P | N/P | N/P | N/P | N/P |
| Idle agents | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| Maximum agent overflows | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 3-28 Instance information report from DB2 PE

A sample and very useful report about buffer pool hit ratio, sort overflows, catalog and package cache hit ratio, and application waiting on locks generated by DB2 PE is shown in Figure 3-29.

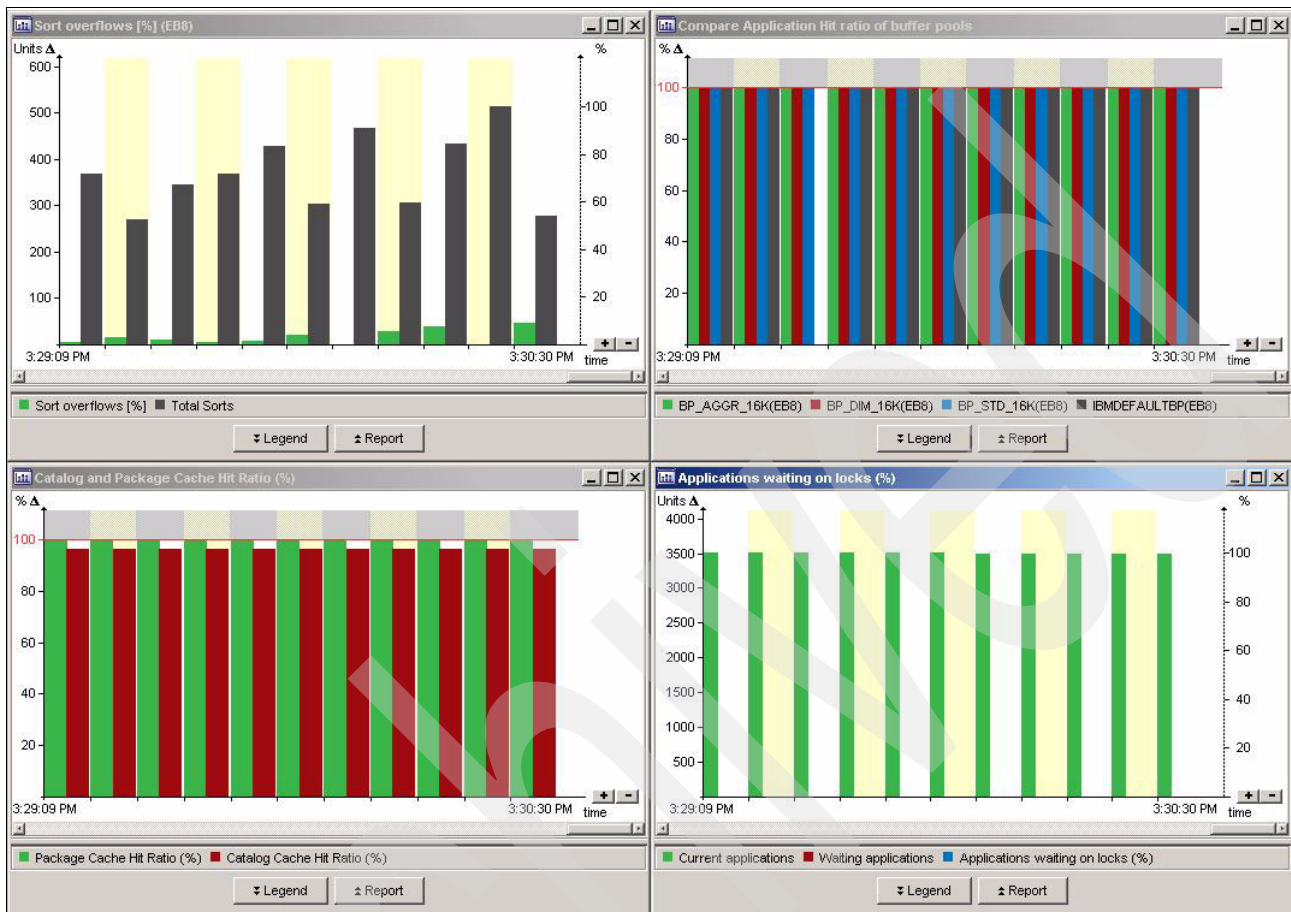


Figure 3-29 An example of DB2 PE report

NMON

NMON, a free tool that analyzes AIX and Linux performance, is not officially supported. The following site provides information about NMON:

http://www-128.ibm.com/developerworks/eserver/articles/analyze_aix/

NMON was used in this project to collect critical information such as memory usage, CPU usage, I/O activity, and network. NMON was started in all DB2 LPARs as part of a group of servers. One master server was configured with rsh access throughout all partitions, and it was responsible for starting the NMON daemon on all other servers that belonged to the system. Information was generated every two minutes and was dumped into a file. The script that we used to start an NMON instance on our DB2 servers is shown in Figure 3-30. The following fields are displayed:

- ▶ The script is based on a time stamp formatted by 12 digits in the format YYYYmmddHHMMSS. For example, if the instance were started at 21:13:17 on July 3 2006, the time stamp would read 20060703211317 (see line 3).
- ▶ The MACHINES_FILES variable (see line 10) lists all servers that must have the NMON instance running, and it is used on the for loop.
- ▶ The LOCK_FILE variable (see line 14) is used to hold the start time stamp so that it can be visible to the stop script for finding find the correct process on the target server and stopping it.
- ▶ As you may have noticed, the remote shell was configured between our servers to facilitate the usage of tools like this.

```

1 DAY=$(date +%d-%m-%Y)
2 TIME=$(date +%H_%M_%S)
3 DATE=$(date +%Y%m%d%H%M%S)
4
5 RUN_NAME=CUSTOMER
6 mkdir -p /XmRec/CUSTOMER/SEB/results/${DATE}
7
8 DIR_RESULTS=/XmRec/CUSTOMER/SEB/results/${DATE}
9
10 MACHINES_FILES=/XmRec/CUSTOMER/SEB/group/1
11
12 DIR_REMOTE_TMP=/tmp/m2/CUSTOMER/${DATE}
13
14 LOCK_FILE="/XmRec/CUSTOMER/SEB/${1}.direct.lock"
15
16 NMON_CMD="nmon -s120 -c2700 -T -t -L -W -N -S -A -m ${DIR_REMOTE_TMP} -F ${DATE}"
17 CHECK_OS_CMD="lscfg > ${DIR_REMOTE_TMP}/${DATE}.os.settings.initial"
18
19 #####
20
21 start_nmon () {
22     echo ""
23     echo "#####"
24     echo "> STARTING RUN ${RUN_NAME} ${NEW_NUM_RUN}"
25     echo "#####"
26     echo ""
27
28     for HOSTNAME in `awk '1 !~/^$/ && 1 !~/^#/ { print $1 }' ${MACHINES_FILES}`; do
29         echo ""
30         echo "-----[ ${HOSTNAME} ]-----"
31         echo "# ${NMON_CMD}"
32         echo ""
33
34         rsh ${HOSTNAME} "[ ! -d ${DIR_REMOTE_TMP} ] && mkdir ${DIR_REMOTE_TMP} ; ${NMON_CMD}"
35         rsh ${HOSTNAME} "[ ! -d ${DIR_REMOTE_TMP} ] && mkdir ${DIR_REMOTE_TMP} ; ${CHECK_OS_CMD}"
36
37         echo ""
38         echo ""
39     done
40 }
41
42 #####
43
44
45 start_nmon
46
47 echo ${DATE} > ${LOCK_FILE}

```

Figure 3-30 NMON start script

Starting NMON

The `nmon` command is started with the following options, which are described in Table 3-11:

```
nmon -s120 -c2700 -T -t -L -W -N -S -A -m ${DIR_REMOTE_TMP} -F ${DATE}
```

Table 3-11 NMON start command options

| Option | Description |
|--------------------|-------------------------------------|
| -s 120 | Refreshed every 120 seconds |
| \${DIR_REMOTE_TMP} | nmon directory on the target server |
| -F \${DATE} | File name: YYYYmddHHMMSS |

The script also runs a `lscfg` on the AIX to check the configuration at the start and stop of the run. The script is very straightforward and is based on `rsh` enablement between servers. To run the script, a parameter must be specified, which is the server group, in our case, varying from `SYS1`, `SYS2`, and `SYS3` as being the pool of servers in our environment. For example, to start NMON monitoring on `SYS1`, we executed this command as root:

```
[root@cusdata]/XmRec/CUSTOMER/SEB: start_direct_run.sh sys1
```


Stopping NMON

The stop nmon script was written following the same concepts, and is shown in Figure 3-31. Based on the time stamp previously stored on the lock file, the stop script goes through servers in the list used in the start script and kills the nmon process. The output file is then copied to the local server and moved to a Samba shared area where we could access with Windows workstations.

```
1 DAY=$(date '+%d-%m-%Y')
2 TIME=$(date '+%H_%M_%S')
3 LOCK_FILE="/XmRec/CUSTOMER/SEB/${1}.direct.lock"
4 DATE=$(cat ${LOCK_FILE})
5
6 THIS_MACHINE=10.3.13.160
7
8 DIR_RESULTS="/XmRec/CUSTOMER/SEB/results/${DATE}"
9
10 MACHINES_FILES="/XmRec/CUSTOMER/SEB/group/${1}"
11 DIR_REMOTE_TMP="/tmp/m2/CUSTOMER/${DATE}"
12 NMON_CMD=$(DATE)
13 CHECK_OS_CMD="lsdfg > ${DIR_REMOTE_TMP}/${DATE}.os.settings.final"
14
15 #####
16
17 stop_nmon () {
18
19     echo ""
20     echo "#####"
21     echo " STOPPING RUN ${RUN_NAME}_${NUM_RUN}"
22     echo "#####"
23
24
25     for HOSTNAME in `awk '{ $1 !~/^#/ && $1 !~/^#/ { print $1 }' ${MACHINES_FILES}`; do
26         echo ""
27         echo "-----[ ${HOSTNAME} ]-----"
28         echo "# Stopping NMON"
29         echo ""
30
31
32         echo "ps -eaf | grep ${NMON_CMD}"
33         rsh ${HOSTNAME} "ps -eaf | grep \"${NMON_CMD}\" | grep -v grep | awk '{ print \$2 }' | xargs
34         ${DIR_REMOTE_TMP}/${DATE} ${THIS_MACHINE}:${DIR_RESULTS}/${DATE}_${HOSTNAME}.nmon"
35         rsh ${HOSTNAME} "${CHECK_OS_CMD}"
36         rsh ${HOSTNAME} rcp ${DIR_REMOTE_TMP}/${DATE}.os.settings.final
37         ${THIS_MACHINE}:${DIR_RESULTS}/${DATE}.${HOSTNAME}.os.settings.final"
38         rsh ${HOSTNAME} rcp ${DIR_REMOTE_TMP}/${DATE}.os.settings.initial
39         ${THIS_MACHINE}:${DIR_RESULTS}/${DATE}.${HOSTNAME}.os.settings.initial"
40
41         DIR_SAVE=/CUSTOMER/`echo ${HOSTNAME} | cut -c 1-4`/NMON/${DATE}
42
43         [ ! -d ${DIR_SAVE}/${DATE} ] && mkdir -p ${DIR_SAVE}/${DATE}
44         cp ${DIR_RESULTS}/${DATE}_${HOSTNAME}.nmon ${DIR_SAVE}/${DATE}/${DATE}_${HOSTNAME}_${TIME}.nm
45
46         echo ""
47         echo ""
48     done
49 }
50 #####
51 stop_nmon
52 echo "-----"
53 echo "All the Data are in ${DIR_RESULTS} ..."
54 echo ""
```

Figure 3-31 NMON stop script

To stop NMON on SYS1, we executed this command as root:

```
[root@cusdata]/XmRec/CUSTOMER/SEB: stop_direct_run.sh sys1
```

Collecting the data

The collected data is parsed and analyzed by a free tool called NMON analyzer. More information about this tool can be obtained at:

<http://www-941.haw.ibm.com/collaboration/wiki/display/Wikiptype/nmonanalyzer>

This tool is based on a Microsoft® Excel® spreadsheet with Visual Basic® for Application (VBA) responsible for parsing and creating all NMON graphics and charts. The base version used in our project is Version 3.1.7 from May 5th, 2006. We modified this version and two

new buttons were added: the Analyze and Collect Data and the Collect Analyzed Data facilities.

Based on this tool, we created an hybrid version of the spreadsheet provided by implementing a third button to generate all default reports and also create a consolidated report in a Microsoft Word format. An example of the spreadsheet used in our tests is shown in Figure 3-32.

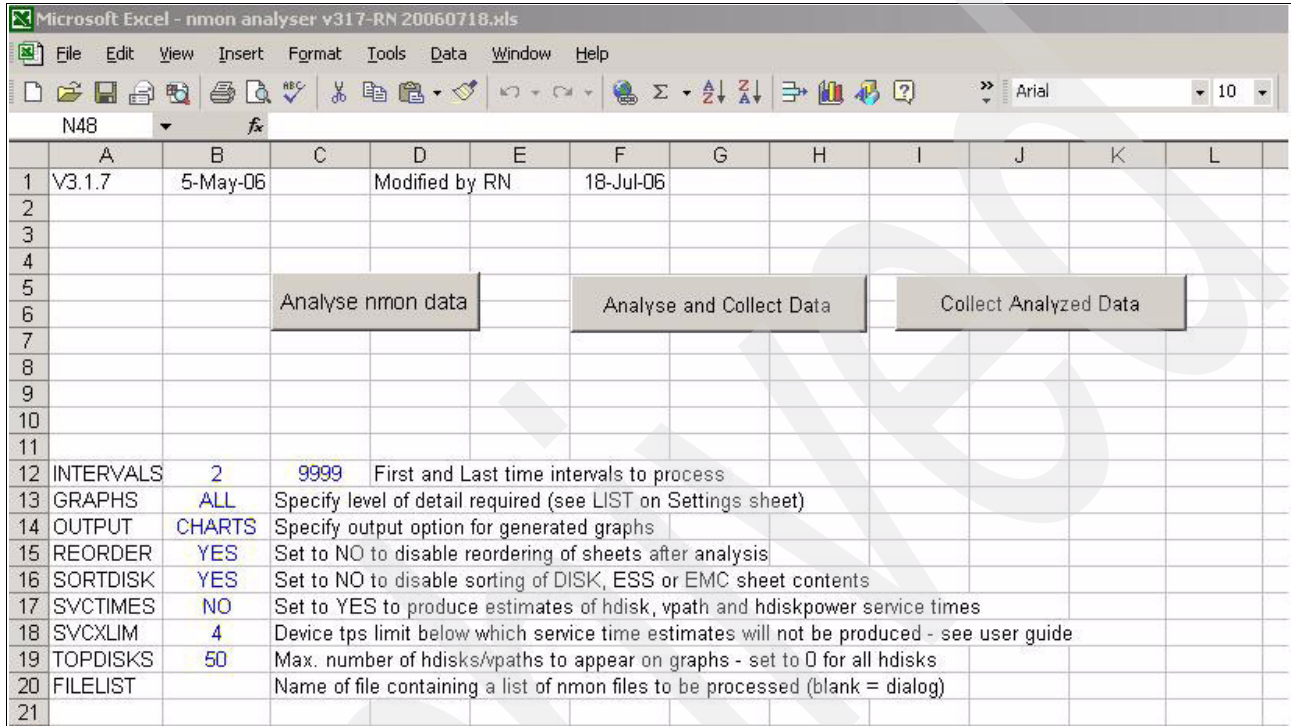


Figure 3-32 NMON analyzer tool modified for the project

The Analyze and Collect Data button was implemented over the logic created on the public available tool plus an implementation to generate a report with consolidated data about all servers in the server group. This button would invoke the function that parses nmon files and generate all the regular reports and a consolidated report with all the information together, which helped us to analyze the whole structure. The last button would generate the consolidated report-based reports that were generated by the first button, so no nmon files would need to be parsed again since the first report generated by the analyze part would feed the consolidation of a second report.

An example of one chart generated by the consolidation feature created is shown in Figure 3-33.

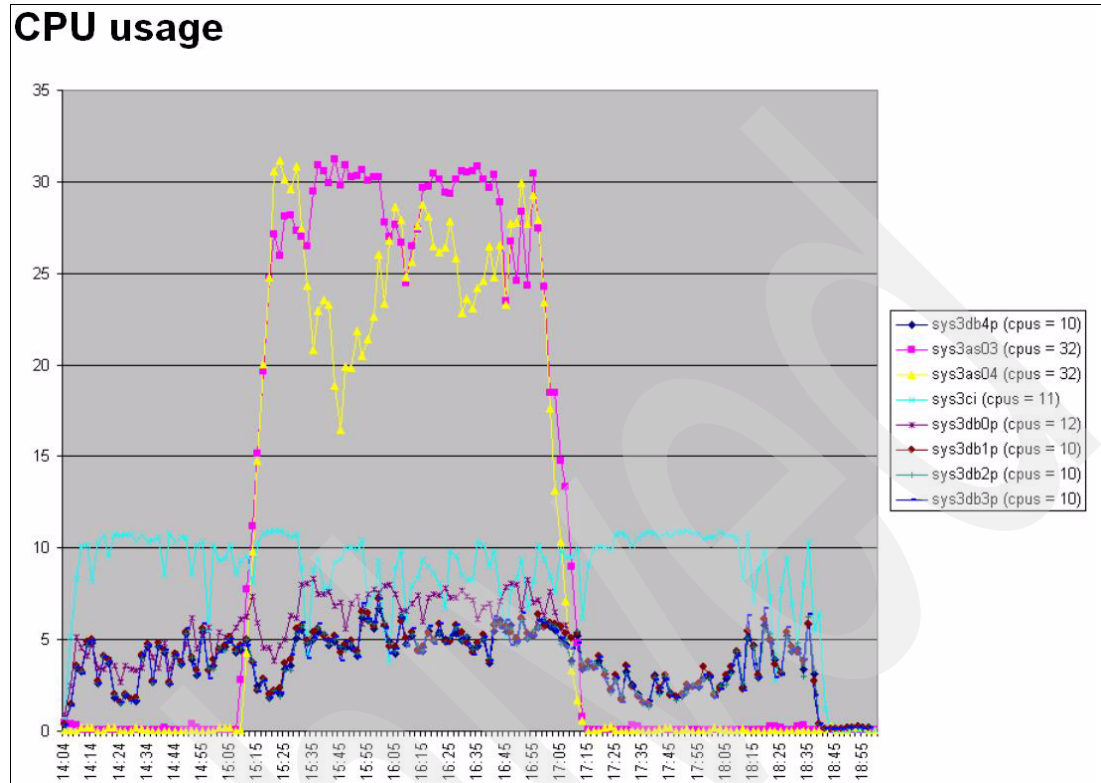


Figure 3-33 NMON analyzer customized report

3.3 Options and parameters discussions

In this section we go through the parameters and options used in our environment. Some of them may be straightforward and normally enabled in most of the SAP NetWeaver BI environments. Others were configured with a specific purpose. The scope of this discussion is the DB2 instance and database parameters.

The DB2 9 release used for the tests is depicted in the Example 3-22. As you can see, fix pack 2 was installed. Only one instance was created, under user db2eb8 and updated to the latest level shown, as it was migrated from DB2 Version 8.

Example 3-22 DB2 9 release level

```

DB2 Release Info
Server is running MPP/EEE.
DB21085I Instance "db2eb8" uses "64" bits and DB2 code release "SQL09012" with
level identifier "01030107".
Informational tokens are "DB2 v9.1.0.2", "s070210", "U810940", and Fix Pack
"2".
Product is installed at "/opt/IBM/db2/V9.1".

```

3.3.1 Registry and environment variables

Some other parameters that influence database and instance performance are set in registry and environment levels that are called *registry* and *environment* parameters.

Environment variables DB2INSTANCE, DB2NODE, DB2PATH, and DB2INSTPROF are stored in the DB2 profile registries, and they all depend on your operating system. In our case, these variables are set when the file named db2profile under sqllib directory is called.

In DB2 Version 8.2 a new registry setting was implemented to optimize the settings for SAP environments. There are a number of internal and external features and changes designed for SAP workloads in DB2 Version 8.2.2 and currently in DB2 9. Many of these features are enabled via registry variables or configuration parameters. To make the configuration and maintenance of SAP on DB2 more transparent, the SAP tuning knob can be used in DB2 9 to set the registry environments optimized for SAP. This feature is implemented as a single registry variable, DB2_WORKLOAD. To give you an idea about how the integration and optimization between DB2 and SAP is important, this variable is the only aggregate registry variable currently available. This feature alleviates the complexity of ensuring that all of the SAP features are enabled for your environment, as well as correctly setting the corresponding registry variables. Quite simply, just set DB2 on the SAP setting.

You can use the `db2set -gd DB2_WORKLOAD=SAP` command to list the default settings for this aggregate registry variable, as shown in Example 3-23. A new variable can be noticed when you set it on DB2 9. It is called DB2_OPT_MAX_TEMP_SIZE.

You can use the new DB2_OPT_MAX_TEMP_SIZE registry variable to limit the amount of space that queries can use in temporary tablespaces.

All registry variables that are set with the use of the aggregate registry variable DB2_WORKLOAD=SAP have the tag [DB2_WORKLOAD] after the value.

When you execute a `db2set -a11` command, it brings variables from five different scopes:

- ▶ The environment, denoted by [e].
- ▶ The user level registry, denoted by [u].
- ▶ The node level registry, denoted by [n].
- ▶ The instance level registry, denoted by [i].
- ▶ The global level registry, denoted by [g].

Example 3-23 Registry and environment parameters

```
[e] DB2CHKPTR=OFF
[e] DB2CODEPAGE=1208
[e] DB2COMM=TCPIP
[e] DB2COUNTRY=1
[e] DB2DBDFT=EB8
[g] DB2ADMINSERVER=dasusr1
[g] DB2INSTDEF=db2eb8
[g] DB2SYSTEM=sys1db0d

[i] DB2_ANTIJOIN=EXTEND [DB2_WORKLOAD]
[i] DB2_APM_PERFORMANCE=1,2,4,5,6,7,8,9
[i] DB2_BLOCK_ON_LOG_DISK_FULL=ON
[i] DB2_CORRELATED_PREDICATES=S
[i] DB2_EVALUNCOMMITTED=YES [DB2_WORKLOAD]
[i] DB2_FORCE_APP_ON_MAX_LOG=YES
```

```

[i] DB2_FORCE_FCM_BP=YES [DB2_WORKLOAD]
[i] DB2_HASH_JOIN=YES
[i] DB2_IMPLICIT_UNICODE=YES [DB2_WORKLOAD]
[i] DB2_INLIST_TO_NLJN=YES [DB2_WORKLOAD]
[i] DB2_INTERESTING_KEYS=YES [DB2_WORKLOAD]
[i] DB2_MDC_ROLLOUT=YES [DB2_WORKLOAD]
[i] DB2_MINIMIZE_LISTPREFETCH=YES [DB2_WORKLOAD]
[i] DB2_NUM_CKPW_DAEMONS=0
[i] DB2_OBJECT_TABLE_ENTRIES=65532 [DB2_WORKLOAD]
[i] DB2_OPT_MAX_TEMP_SIZE=10240 [DB2_WORKLOAD]
[i] DB2_OPTPROFILE=YES [DB2_WORKLOAD]
[i] DB2_PARALLEL_IO=*
[i]
DB2_REDUCED_OPTIMIZATION=4,INDEX,JOIN,NO_TQ_FACT,NO_HSJN_BUILD_FACT,STARJN_CARD_SK
EW,NO_SORT_MGJOIN [DB2_WORKLOAD]
[i] DB2_RR_TO_RS=YES [DB2_WORKLOAD]
[i] DB2_SKIPINSERTED=YES [DB2_WORKLOAD]
[i] DB2_STATVIEW=YES
[i] DB2_STRIPED_CONTAINERS=ON
[i] DB2_TRUNCATE_REUSESTORAGE=IMPORT [DB2_WORKLOAD]
[i] DB2_UPDATE_PART_KEY=YES [DB2_WORKLOAD]
[i] DB2_USE_ALTERNATE_PAGE_CLEANING=ON
[i] DB2_USE_FAST_PREALLOCATION=ON

[i] DB2_VENDOR_INI=/db2/EB8/dbs/tsm_config/vendor.env
[i] DB2_VIEW_REOPT_VALUES=YES [DB2_WORKLOAD]
[i] DB2_WORKLOAD=SAP
[i] DB2ATLD_PORTS=6000:6500
[i] DB2CODEPAGE=1208
[i] DB2COMM=TCPIP [DB2_WORKLOAD]
[i] DB2DBDFT=EB8
[i] DB2MEMDISCLAIM=YES
[i] DB2MEMMAXFREE=2000000 [DB2_WORKLOAD]
[i] DB2NOTIFYVERBOSE=YES [DB2_WORKLOAD]
[i] DB2RSHCMD=/usr/bin/ssh

```

-
- ▶ Two new keywords, `NO_SORT_MGJOIN` and `NO_SORT_NLJOIN`, are added to the `DB2_REDUCED_OPTIMIZATION` registry variable.
 - The new `NO_SORT_MGJOIN` keyword instructs the optimizer to generate query plans that do not force sorts for merge scan joins (MSJN).
 - The new `NO_SORT_NLJOIN` keyword instructs the optimizer to generate query plans that do not force sorts for nested loop joins (NLJN).
 - ▶ DB2 started using `J2_METAMAP` space pre-allocation in 64-bit DB2 UDB Version 8.1 fix pack 7 for DMS files on JFS2 file systems. It has changed the behavior regarding the use of `J2_METAMAP` space pre-allocation in Version 8.1 fix pack 9, due to issues that have been encountered. By default, `J2_METAMAP` space pre-allocation is not utilized as of Version 8.1 fix pack 9. As a result, space allocation using the `DB2 ALTER` or `CREATE TABLESPACE` commands may take a longer time on 64-bit DB2 UDB Version 8.1 fix pack 9 and later as compared with fix packs 7 and 8. The faster space allocation method may still be enabled via the registry variable `DB2_USE_FAST_PREALLOCATION` using the command `db2set DB2_USE_FAST_PREALLOCATION=YES`.

Note that space that was allocated while running 64-bit DB2 UDB Version 8.1 fix packs 7 and 8 may still be vulnerable to problems due to the use of the JFS2 pre-allocation feature.

- ▶ The MAX_LOG database configuration parameter controls the percentage of log space that a unique application can use.

If the environment variable DB2_FORCE_APP_ON_MAX_LOG is set to TRUE, the application that exceeds the percentage configured on MAX_LOG is forced off the database and the unit of work is rolled back.

If this parameter is set to FALSE, the current statement fails. The application can still commit the work completed by the previous statements in the unit of work, or it can roll back the work completed to undo the unit of work.

- ▶ The DB2_APM_PERFORMANCE variable was set for latch contention issues.
- ▶ By default, DB2 UDB uses the first extent of each DMS container (file or device) to store a container tag. The container tag is DB2 metadata for the container. In earlier versions of DB2 UDB, the first page was used for the container tag, instead of the first extent, and as a result less space in the container was used to store the tag. (In earlier versions of DB2 UDB, the DB2_STRIPED_CONTAINERS registry variable was used to create tablespaces with an extent sized tag. However, because this is now the default behavior, this registry variable no longer has any affect.)
- ▶ The default for the DB2_CORRELATED_PREDICATES variable is ON. When there are unique indexes on correlated columns in a join, and this registry variable is ON, the optimizer attempts to detect and to compensate for correlation of join predicates. When this registry variable is ON, the optimizer uses the KEYCARD information of unique index statistics to detect cases of correlation, and dynamically adjusts the combined selectivity of the correlated predicates, thus obtaining a more accurate estimate of the join size and cost.
- ▶ The DB2ATLD_PORTS registry variable will replace the value of the PORT_RANGE load configuration option. The default range is from 6000 to 6063. For the DB2ATLD_PORTS registry variable, the range should be provided in the following format:
`<lower-port-number>:<higher-port-number>`
- ▶ DB2_HASH_JOIN specifies hash-join as a possible join method when compiling an access plan. The DB2_HASH_JOIN registry variable should be used, but it needs to be tuned to get the best performance. Hash-join performance is best if you can avoid hash loops and overflow to disk. To tune hash-join performance, estimate the maximum amount of memory available for the sheapthres configuration parameter, and then tune the sortheap configuration parameter. Increase its value until you avoid as many hash loops and disk overflows as possible, but do not reach the limit specified by the sheapthres configuration parameter.
- ▶ DB2MEMMAXFREE specifies the maximum number of bytes of unused private memory that is retained by DB2 processes before unused memory is returned to the operating system.
- ▶ On AIX, memory used by DB2 processes may have some associated paging space. This paging space may remain reserved even when the associated memory has been freed, and it depends on the AIX system's (tunable) virtual memory management allocation policy. The DB2MEMDISCLAIM registry variable controls whether DB2 agents explicitly request that AIX disassociates the reserved paging space from the freed memory.

A DB2MEMDISCLAIM setting of YES results in smaller paging space requirements, and possibly less disk activity from paging. A DB2MEMDISCLAIM setting of NO results in larger paging space requirements, and possibly more disk activity from paging. In some

situations, such as though paging space is plentiful and real memory is so plentiful that paging never occurs, a setting of NO provides a minor performance improvement.

- ▶ The DB2ENVLIST=INSTHOME SAPSYSTEMNAME dbs_db6_schema DIR_LIBRARY LIBPATH variable lists specific variable names for either stored procedures or user-defined functions. By default, the **db2start** command filters out all user environment variables except those prefixed with DB2 or db2. If specific environment variables must be passed to either stored procedures or user-defined functions, you can list the variable names in the DB2ENVLIST environment variable. Separate each variable name by one or more spaces.
- ▶ DB2_BLOCK_ON_LOG_DISK_FULL is a registry variable that you can set to prevent *disk full* errors from being generated when DB2 cannot create a new log file in the active log path. DB2 attempts to create the log file every five minutes and writes a message to the db2diag.log file after each attempt.
- ▶ The DB2_FORCE_FCM_BP=YES [DB2_WORKLOAD] variable is applicable to DB2 UDB ESE for AIX with multiple logical partitions. When DB2START is issued, DB2 allocates the FCM buffers either from the database global memory or from a separate shared memory segment if there is not enough global memory available. These buffers are used by all FCM daemons for that instance on the same physical machine. The kind of memory allocated is largely dependent on the number of FCM buffers to be created, as specified by the fcm_num_buffers database manager configuration parameter.

If the DB2_FORCE_FCM_BP variable is set to YES, the FCM buffers are always created in a separate memory segment so that communication between FCM daemons of different logical partitions on the same physical node occurs through shared memory. Otherwise, FCM daemons on the same node communicate through UNIX Sockets. Communicating through shared memory is faster, but there is one fewer shared memory segment available for other uses, particularly for database buffer pools. Enabling the DB2_FORCE_FCM_BP registry variable reduces the maximum size of database buffer pools.

- ▶ DB2DBDFT=EB8 specifies the database alias name of the database to be used for implicit connects. If an application has no database connection, but SQL statements are issued, an implicit connect will be made if the DB2DBDFT environment variable has been defined with a default database.
- ▶ The DB2COMM registry variable allows you to set communication protocols for the current DB2 instance. If the DB2COMM registry variable is undefined or set to null, no protocol connection managers are started when the database manager is started.
- ▶ DB2CODEPAGE=1208 specifies the code page of the data presented to DB2 for database client application. The user should not set DB2CODEPAGE unless explicitly stated in DB2 documents, or asked to do so by DB2 service. Setting DB2CODEPAGE to a value not supported by the operating system can produce unexpected results. Normally, you do not need to set DB2CODEPAGE because DB2 automatically derives the code page information from the operating system.
- ▶ The DB2_PARALLEL_IO registry variable is used to change the way DB2 UDB calculates the I/O parallelism of a tablespace. When I/O parallelism is enabled (either implicitly, by the use of multiple containers, or explicitly, by setting DB2_PARALLEL_IO), it is achieved by issuing the correct number of prefetch requests. Each prefetch request is a request for an extent of pages.
 - If this registry variable is not set, the degree of parallelism of any tablespace is the number of containers of the tablespace. For example, if DB2_PARALLEL_IO is set to null and a tablespace has four containers, there will be four extent-sized prefetch requests issued.

- If this registry variable is set, the degree of parallelism of the tablespace is the ratio between the prefetch size and the extent size of this tablespace. For example, if DB2_PARALLEL_IO is set for a tablespace that has a prefetch size of 160 and an extent size of 32 pages, there will be five extent-sized prefetch requests issued. A wildcard character can be used to tell DB2 UDB to calculate the I/O parallelism for all tablespaces this way.

In I/O subsystems that support striping, the physical spindles beneath each DB2 UDB container (for example, with a RAID device), the number of disks underneath each DB2 UDB container should be taken into account when choosing a prefetch size for the tablespace. If the prefetch size of the tablespace is AUTOMATIC, DB2 UDB automatically calculates the prefetch size of a tablespace using the following equation:

Prefetch size = (number of containers)*(number of disks per container)*extent size

The number is then used by DB2 UDB to fill in the number of disks per container in the equation.

If only an asterisk is used but a number is not specified, a default of six disks per container is used.

The DB2_PARALLEL_IO registry variable can be used to tell DB2 UDB the number of disks per container. For example, if DB2_PARALLEL_IO="1:4" and tablespace 1 have three containers, extent size 32, and prefetch size AUTOMATIC, then the prefetch size is calculated as 3 * 4 * 32 = 384 pages. The I/O parallelism of this tablespace is 384 divided by 32 = 12. If the prefetch size of a tablespace is not AUTOMATIC, this information about the number of disks per container is not used.

Any tablespace that is specified under the DB2_PARALLEL_IO variable is assumed to be using six as the number of disks per container. For example, if DB2_PARALLEL_IO="*,1:3, all tablespaces will use six as the number of disks per container, except for tablespace 1, which will use three. Values other than six can be specified in the registry variable.

3.3.2 Instance configuration

The instance configuration is described in Example 3-24 for SYS1 and in Example 3-25 on page 200 for SYS2. Configurations between environments have minor changes.

Example 3-24 SYS1 database manager configuration

```

Database Manager Configuration
Node type = Enterprise Server Edition with local and remote clients

Database manager configuration release level           = 0x0b00
CPU speed (millisec/instruction)                      (CPUSPEED) = 3.857478e-07
Communications bandwidth (MB/sec)                    (COMM_BANDWIDTH) = 2.000000e+02
Max number of concurrently active databases           (NUMDB) = 6
Federated Database System Support                    (FEDERATED) = NO
Transaction processor monitor name                    (TP_MON_NAME) =
Default charge-back account                           (DFT_ACCOUNT_STR) =
Java Development Kit installation path                 (JDK_PATH) =
/db2/db2eb8/sqllib/java/jdk64
Diagnostic error capture level                        (DIAGLEVEL) = 3
Notify Level                                          (NOTIFYLEVEL) = 3
Diagnostic data directory path                        (DIAGPATH) = /db2/EB8/db2dump
Default database monitor switches
  Buffer pool                                          (DFT_MON_BUFPOOL) = ON
  Lock                                                (DFT_MON_LOCK) = ON

```


| | | |
|---|-------------------------|-------------------|
| Sort | (DFT_MON_SORT) | = ON |
| Statement | (DFT_MON_STMT) | = ON |
| Table | (DFT_MON_TABLE) | = ON |
| Timestamp | (DFT_MON_TIMESTAMP) | = ON |
| Unit of work | (DFT_MON_UOW) | = ON |
| Monitor health of instance and databases | (HEALTH_MON) | = OFF |
| SYSADM group name | (SYSADM_GROUP) | = DBEB8ADM |
| SYSCTRL group name | (SYSCTRL_GROUP) | = DBEB8CTL |
| SYSMAINT group name | (SYSMAINT_GROUP) | = |
| SYSMON group name | (SYSMON_GROUP) | = |
| Client Userid-Password Plugin | (CLNT_PW_PLUGIN) | = |
| Client Kerberos Plugin | (CLNT_KRB_PLUGIN) | = |
| Group Plugin | (GROUP_PLUGIN) | = |
| GSS Plugin for Local Authorization | (LOCAL_GSSPLUGIN) | = |
| Server Plugin Mode | (SRV_PLUGIN_MODE) | = UNFENCED |
| Server List of GSS Plugins | (SRVCON_GSSPLUGIN_LIST) | = |
| Server Userid-Password Plugin | (SRVCON_PW_PLUGIN) | = |
| Server Connection Authentication | (SRVCON_AUTH) | = NOT_SPECIFIED |
| Database manager authentication | (AUTHENTICATION) | = SERVER |
| Cataloging allowed without authority | (CATALOG_NOAUTH) | = NO |
| Trust all clients | (TRUST_ALLCLNTS) | = YES |
| Trusted client authentication | (TRUST_CLNTAUTH) | = CLIENT |
| Bypass federated authentication | (FED_NOAUTH) | = NO |
| Default database path | (DFTDBPATH) | = /db2/EB8 |
| Database monitor heap size (4KB) | (MON_HEAP_SZ) | = 2048 |
| Java Virtual Machine heap size (4KB) | (JAVA_HEAP_SZ) | = 2048 |
| Audit buffer size (4KB) | (AUDIT_BUF_SZ) | = 64 |
| Size of instance shared memory (4KB) | (INSTANCE_MEMORY) | = AUTOMATIC |
| Backup buffer default size (4KB) | (BACKBUFSZ) | = 1024 |
| Restore buffer default size (4KB) | (RESTBUFSZ) | = 1024 |
| Sort heap threshold (4KB) | (SHEAPTHRES) | = 500000 |
| Directory cache support | (DIR_CACHE) | = NO |
| Application support layer heap size (4KB) | (ASLHEAPSZ) | = 16 |
| Max requester I/O block size (bytes) | (RQRIOBLK) | = 65535 |
| Query heap size (4KB) | (QUERY_HEAP_SZ) | = 4096 |
| Workload impact by throttled utilities | (UTIL_IMPACT_LIM) | = 10 |
| Priority of agents | (AGENTPRI) | = SYSTEM |
| Max number of existing agents | (MAXAGENTS) | = 1500 |
| Agent pool size | (NUM_POOLAGENTS) | = 50 |
| Initial number of agents in pool | (NUM_INITAGENTS) | = 5 |
| Max number of coordinating agents | (MAX_COORDAGENTS) | = (MAXAGENTS - |
| NUM_INITAGENTS) | | |
| Max no. of concurrent coordinating agents | (MAXCAGENTS) | = MAX_COORDAGENTS |
| Max number of client connections | (MAX_CONNECTIONS) | = MAX_COORDAGENTS |
| Keep fenced process | (KEEPFENCED) | = NO |
| Number of pooled fenced processes | (FENCED_POOL) | = 5 |
| Initial number of fenced processes | (NUM_INITFENCED) | = 0 |
| Index re-creation time and redo index build | (INDEXREC) | = ACCESS |
| Transaction manager database name | (TM_DATABASE) | = 1ST_CONN |
| Transaction resync interval (sec) | (RESYNC_INTERVAL) | = 180 |
| SPM name | (SPM_NAME) | = sys1db0d |
| SPM log size | (SPM_LOG_FILE_SZ) | = 256 |
| SPM resync agent limit | (SPM_MAX_RESYNC) | = 20 |
| SPM log path | (SPM_LOG_PATH) | = |
| TCP/IP Service name | (SVCENAME) | = sapdb2EB8 |

```

Discovery mode (DISCOVER) = DISABLE
Discover server instance (DISCOVER_INST) = DISABLE
Maximum query degree of parallelism (MAX_QUERYDEGREE) = 4
Enable intra-partition parallelism (INTRA_PARALLEL) = NO
Maximum Asynchronous TQs per query (FEDERATED_ASYNC) = 0
No. of int. communication buffers(4KB) (FCM_NUM_BUFFERS) = AUTOMATIC
No. of int. communication channels (FCM_NUM_CHANNELS) = AUTOMATIC
Node connection elapse time (sec) (CONN_ELAPSE) = 10
Max number of node connection retries (MAX_CONNRETRIES) = 5
Max time difference between nodes (min) (MAX_TIME_DIFF) = 60
db2start/db2stop timeout (min) (START_STOP_TIME) = 10

```

One change was done on the SYS1 environment to increase the number of MAXAGENTS and NUM_POOLAGENTS due to some specific changes on the SAP NetWeaver BI process chain and the number of dispatchers running that required more active connections on the database.

Example 3-25 SYS2 database manager configuration

```

Database Manager Configuration
Node type = Enterprise Server Edition with local and remote clients
Database manager configuration release level = 0x0b00
CPU speed (millisec/instruction) (CPUSPEED) = 3.778755e-07
Communications bandwidth (MB/sec) (COMM_BANDWIDTH) = 1.000000e+02
Max number of concurrently active databases (NUMDB) = 6
Federated Database System Support (FEDERATED) = NO
Transaction processor monitor name (TP_MON_NAME) =
Default charge-back account (DFT_ACCOUNT_STR) =
Java Development Kit installation path (JDK_PATH) =
/db2/db2eb8/sqllib/java/jdk64
Diagnostic error capture level (DIAGLEVEL) = 3
Notify Level (NOTIFYLEVEL) = 3
Diagnostic data directory path (DIAGPATH) = /db2/EB8/db2dump
Default database monitor switches
Buffer pool (DFT_MON_BUFPOOL) = ON
Lock (DFT_MON_LOCK) = ON
Sort (DFT_MON_SORT) = ON
Statement (DFT_MON_STMT) = ON
Table (DFT_MON_TABLE) = ON
Timestamp (DFT_MON_TIMESTAMP) = ON
Unit of work (DFT_MON_UOW) = ON
Monitor health of instance and databases (HEALTH_MON) = OFF
SYSADM group name (SYSADM_GROUP) = DBEB8ADM
SYSCTRL group name (SYSCTRL_GROUP) = DBEB8CTL
SYSMAINT group name (SYSMAINT_GROUP) =
SYSMON group name (SYSMON_GROUP) =
Client Userid-Password Plugin (CLNT_PW_PLUGIN) =
Client Kerberos Plugin (CLNT_KRB_PLUGIN) =
Group Plugin (GROUP_PLUGIN) =
GSS Plugin for Local Authorization (LOCAL_GSSPLUGIN) =
Server Plugin Mode (SRV_PLUGIN_MODE) = UNFENCED
Server List of GSS Plugins (SRVCON_GSSPLUGIN_LIST) =
Server Userid-Password Plugin (SRVCON_PW_PLUGIN) =
Server Connection Authentication (SRVCON_AUTH) = NOT_SPECIFIED
Database manager authentication (AUTHENTICATION) = SERVER
Cataloging allowed without authority (CATALOG_NOAUTH) = NO

```

| | |
|---|-------------------------------------|
| Trust all clients | (TRUST_ALLCLNTS) = YES |
| Trusted client authentication | (TRUST_CLNTAUTH) = CLIENT |
| Bypass federated authentication | (FED_NOAUTH) = NO |
| Default database path | (DFTDBPATH) = /db2/EB8 |
| Database monitor heap size (4KB) | (MON_HEAP_SZ) = 4096 |
| Java Virtual Machine heap size (4KB) | (JAVA_HEAP_SZ) = 2048 |
| Audit buffer size (4KB) | (AUDIT_BUF_SZ) = 64 |
| Size of instance shared memory (4KB) | (INSTANCE_MEMORY) = AUTOMATIC |
| Backup buffer default size (4KB) | (BACKBUFSZ) = 1024 |
| Restore buffer default size (4KB) | (RESTBUFSZ) = 1024 |
| Sort heap threshold (4KB) | (SHEAPTHRES) = 500000 |
| Directory cache support | (DIR_CACHE) = NO |
| Application support layer heap size (4KB) | (ASLHEAPSZ) = 16 |
| Max requester I/O block size (bytes) | (RQRIOBLK) = 65535 |
| Query heap size (4KB) | (QUERY_HEAP_SZ) = 4096 |
| Workload impact by throttled utilities | (UTIL_IMPACT_LIM) = 10 |
| Priority of agents | (AGENTPRI) = SYSTEM |
| Max number of existing agents | (MAXAGENTS) = 1024 |
| Agent pool size | (NUM_POOLAGENTS) = 100 |
| Initial number of agents in pool | (NUM_INITAGENTS) = 5 |
| Max number of coordinating agents | (MAX_COORDAGENTS) = (MAXAGENTS - |
| NUM_INITAGENTS) | |
| Max no. of concurrent coordinating agents | (MAXCAGENTS) = MAX_COORDAGENTS |
| Max number of client connections | (MAX_CONNECTIONS) = MAX_COORDAGENTS |
| Keep fenced process | (KEEPFENCED) = NO |
| Number of pooled fenced processes | (FENCED_POOL) = 5 |
| Initial number of fenced processes | (NUM_INITFENCED) = 0 |
| Index re-creation time and redo index build | (INDEXREC) = ACCESS |
| Transaction manager database name | (TM_DATABASE) = 1ST_CONN |
| Transaction resync interval (sec) | (RESYNC_INTERVAL) = 180 |
| SPM name | (SPM_NAME) = sys2db0p |
| SPM log size | (SPM_LOG_FILE_SZ) = 256 |
| SPM resync agent limit | (SPM_MAX_RESYNC) = 20 |
| SPM log path | (SPM_LOG_PATH) = |
| TCP/IP Service name | (SVCENAME) = sapdb2EB8 |
| Discovery mode | (DISCOVER) = DISABLE |
| Discover server instance | (DISCOVER_INST) = DISABLE |
| Maximum query degree of parallelism | (MAX_QUERYDEGREE) = 2 |
| Enable intra-partition parallelism | (INTRA_PARALLEL) = NO |
| Maximum Asynchronous TQs per query | (FEDERATED_ASYNC) = 0 |
| No. of int. communication buffers(4KB) | (FCM_NUM_BUFFERS) = AUTOMATIC |
| No. of int. communication channels | (FCM_NUM_CHANNELS) = AUTOMATIC |
| Node connection elapse time (sec) | (CONN_ELAPSE) = 10 |
| Max number of node connection retries | (MAX_CONNRETRIES) = 5 |
| Max time difference between nodes (min) | (MAX_TIME_DIFF) = 60 |
| db2start/db2stop timeout (min) | (START_STOP_TIME) = 10 |

One change executed during the tests that helped to solve an issue was the change of FCM_NUM_BUFFERS and FCM_NUM_CHANNELS to AUTOMATIC. We set these parameters to automatic due to the same enhancement on the communication sub-system architecture in DB2 9.

The re-architecture of the DB2 9 communications subsystem has resulted in several enhancements involving the Database Partitioning Feature (DPF):

- ▶ Separate sender and receiver communications daemons are now used to improve communication speed.
- ▶ The number of FCM buffers and FCM channels can be dynamically changed during the execution, eliminating the need for tuning. There are several other dynamic configuration improvements that automatically monitor resource usage without your involvement.

These enhancements affect several configuration parameters and monitor elements.

- ▶ The `fcm_num_buffers` parameter has a new `AUTOMATIC` setting that allows the DB2 database system to attempt to tune the configured parameter value if the database is not making full use of resources. This setting is turned on by default.
- ▶ The new `fcm_num_channels` configuration parameter specifies the number of FCM channels. It replaces the deprecated `fcm_num_rqb`, `fcm_num_anchors`, and `fcm_num_connect` parameters. This parameter is set to `AUTOMATIC` by default.

Two new monitor elements replace a number of deprecated ones:

- ▶ The new `ch_free` monitor element indicates the number of inter-node communication channels that are currently free. It replaces the deprecated `ma_free`, `ce_free`, and `rb_free` monitor elements.
- ▶ The new `ch_free_bottom` monitor element indicates the lowest number of free inter-node communication channels reached during processing. It replaces the deprecated `ma_free_bottom`, `ce_free_bottom`, and `rb_free_bottom` monitor elements.

Doing a small test using `FCM_NUM_BUFFERS` and `FCM_NUM_CHANNELS` to automatic, we identified a lower memory usage comparing DB2 Version 8 and DB2 9. This is described in Table 3-12.

Table 3-12 Memory usage comparison between DB2 Version 8 and DB2 9

| Host name | Memory (MB) automatic FCM parameters in DB2 Version 8 | Memory(MB) automatic FCM parameters in DB2 9 | Memory (MB) difference |
|-----------|--|--|---------------------------|
| sys2db0 | 31624 | 31768 | 143.92 |
| sys2db1 | 115454 | 115072 | -381.48 |
| sys2db2 | 115397 | 115097 | -300.38 |
| sys2db3 | 115445 | 115115 | -330.72 |
| sys2db4 | 115409 | 115067 | -342.63 |

Although being a minor difference, this denotes clearly how the FCM architecture was improved together with the communication architecture in DB2 9. An example, extracted from `db2diag.log`, of FCM dynamic adjustments is shown in Example 3-26. As you can see, DB2 9 would change values according to an overall resource analysis.

Example 3-26 FCM dynamic adjustment message extracted from `db2diag.log`

```

2007-03-01-17.41.18.752172+060 I208963A438          LEVEL: Warning
PID      : 590186          TID   : 1          PROC  : db2sysc 11
INSTANCE: db2eb8          NODE  : 011
FUNCTION: DB2 UDB, fast comm manager, sqkfDynamicResourceMgr::AdjustResources, probe:100
MESSAGE : FCM Automatic/Dynamic Resource Adjustment (Buffer): 41170 will be

```

released; new total will be 123510 (for 8 logical & 2 failover partitions).

3.3.3 Database configuration

The database configuration is described in Example 3-27 for SYS1. Because the database configuration in SYS2 is very similar to the one used for SYS1, we only discuss the SYS1 database configuration.

In SYS1 and SYS2 most database configuration variables were the same across DB2 partitions, besides the log archive directory, which was configured to point to a specific directory for each partition.

Example 3-27 SYS1 database configuration

```
Database Configuration for Database EB8
Database configuration release level      = 0x0b00
Database release level                   = 0x0b00
Database territory                        = en_US
Database code page                        = 1208
Database code set                         = UTF-8
Database country/region code             = 1
Database collating sequence              = IDENTITY_16BIT
Alternate collating sequence              (ALT_COLLATE) =
Database page size                       = 4096
Dynamic SQL Query management             (DYN_QUERY_MGMT) = DISABLE
Discovery support for this database       (DISCOVER_DB)   = DISABLE
Restrict access                           = NO
Default query optimization class         (DFT_QUERYOPT)  = 5
Degree of parallelism                    (DFT_DEGREE)   = 1
Continue upon arithmetic exceptions      (DFT_SQLMATHWARN) = NO
Default refresh age                      (DFT_REFRESH_AGE) = 0
Default maintained table types for opt   (DFT_MTTB_TYPES) = SYSTEM
Number of frequent values retained       (NUM_FREQVALUES) = 10
Number of quantiles retained             (NUM_QUANTILES) = 20
Backup pending                           = NO
Database is consistent                   = NO
Rollforward pending                      = NO
Restore pending                          = NO
Multi-page file allocation enabled        = YES
Log retain for recovery status            = RECOVERY
User exit for logging status              = YES
Self tuning memory                       (SELF_TUNING_MEM) = OFF
Size of database shared memory (4KB)     (DATABASE_MEMORY) = COMPUTED
Database memory threshold                 (DB_MEM_THRESH)  = 10
Max storage for lock list (4KB)          (LOCKLIST)      = AUTOMATIC
Percent. of lock lists per application    (MAXLOCKS)      = AUTOMATIC
Package cache size (4KB)                 (PCKCACHESZ)    = 20480
Sort heap thres for shared sorts (4KB)   (SHEAPTHRES_SHR) = 200000
Sort list heap (4KB)                     (SORTHEAP)      = 65000
Database heap (4KB)                      (DBHEAP)        = 159000
Catalog cache size (4KB)                 (CATALOGCACHE_SZ) = 10240
Log buffer size (4KB)                    (LOGBUFSZ)      = 16384
Utilities heap size (4KB)                (UTIL_HEAP_SZ)  = 30720
Buffer pool size (pages)                  (BUFFPAGE)      = 3500000
```

```

Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 128000
Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 25
Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 2000
SQL statement heap (4KB) (STMHEAP) = 60000
Default application heap (4KB) (APPLHEAPSZ) = 4096
Statistics heap size (4KB) (STAT_HEAP_SZ) = 20480
Interval for checking deadlock (ms) (DLCHKTIME) = 300000
Lock timeout (sec) (LOCKTIMEOUT) = 3600
Changed pages threshold (CHNGPGS_THRESH) = 45
Number of asynchronous page cleaners (NUM_IOCLEANERS) = AUTOMATIC
Number of I/O servers (NUM_IOSERVERS) = AUTOMATIC
Index sort flag (INDEXSORT) = YES
Sequential detect flag (SEQDETECT) = YES
Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32
Track modified pages (TRACKMOD) = OFF
Default number of containers = 1
Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 16
Max number of active applications (MAXAPPLS) = AUTOMATIC
Average number of active applications (AVG_APPLS) = AUTOMATIC
Max DB files open per application (MAXFILOP) = 1950
Log file size (4KB) (LOGFILSIZ) = 262144
Number of primary log files (LOGPRIMARY) = 115
Number of secondary log files (LOGSECOND) = 13
Changed path to log files (NEWLOGPATH) =
Path to log files = /db2/EB8/log_dir/NODE0000/
Overflow log path (OVERFLOWLOGPATH) =
Mirror log path (MIRRORLOGPATH) =
First active log file = S0102327.LOG
Block log on disk full (BLK_LOG_DSK_FUL) = YES
Percent max primary log space by transaction (MAX_LOG) = 21
Num. of active log files for 1 active UOW (NUM_LOG_SPAN) = 115
Group commit count (MINCOMMIT) = 1
Percent log file reclaimed before soft ckcpt (SOFTMAX) = 300
Log retain for recovery enabled (LOGRETAIN) = RECOVERY
User exit for logging enabled (USEREXIT) = OFF
HADR database role = STANDARD
HADR local host name (HADR_LOCAL_HOST) =
HADR local service name (HADR_LOCAL_SVC) =
HADR remote host name (HADR_REMOTE_HOST) =
HADR remote service name (HADR_REMOTE_SVC) =
HADR instance name of remote server (HADR_REMOTE_INST) =
HADR timeout value (HADR_TIMEOUT) = 120
HADR log write synchronization mode (HADR_SYNCMODE) = NEARSYNC
First log archive method (LOGARCHMETH1) =
VENDOR:/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a
Options for logarchmeth1 (LOGARCHOPT1) = /db2/EB8/dbs/tsm_config/vendor.env.0
Second log archive method (LOGARCHMETH2) = OFF
Options for logarchmeth2 (LOGARCHOPT2) =
Failover log archive path (FAILARCHPATH) = /db2/EB8/log_archive/
Number of log archive retries on error (NUMARCHRETRY) = 10
Log archive retry Delay (secs) (ARCHRETRYDELAY) = 600
Vendor options (VENDOROPT) = /db2/EB8/dbs/tsm_config/vendor.env.0
Auto restart enabled (AUTORESTART) = ON
Index re-creation time and redo index build (INDEXREC) = RESTART
Log pages during index build (LOGINDEXBUILD) = OFF

```

```

Default number of loadrec sessions (DFT_LOADREC_SES) = 1
Number of database backups to retain (NUM_DB_BACKUPS) = 12
Recovery history retention (days) (REC_HIS_RETENTN) = 10
TSM management class (TSM_MGMTCLASS) = DB2_DBWT
TSM node name (TSM_NODENAME) =
TSM owner (TSM_OWNER) =
TSM password (TSM_PASSWORD) =
Automatic maintenance (AUTO_MAINT) = OFF
Automatic database backup (AUTO_DB_BACKUP) = OFF
Automatic table maintenance (AUTO_TBL_MAINT) = OFF
Automatic runstats (AUTO_RUNSTATS) = OFF
Automatic statistics profiling (AUTO_STATS_PROF) = OFF
Automatic profile updates (AUTO_PROF_UPD) = OFF
Automatic reorganization (AUTO_REORG) = OFF

```

Specifically for SYS1, another variable varies according to DB2 partitions, the variable LOGARCHOPT1, which points to a specific configuration file per partition. As shown in Example 3-27 on page 203, it shows the value /db2/EB8/dbs/tsm_config/vendor.env.0 for DB2 partition 0. The last digit of the file name changes accordingly to the partition, as shown in Table 3-13.

Table 3-13 TSM servers pattern for DB2 partitions

| DB2 partition | LOGARCHOPT1 value |
|---------------|---------------------------------------|
| 0 | /db2/EB8/dbs/tsm_config/vendor.env.0 |
| 1 | /db2/EB8/dbs/tsm_config/vendor.env.1 |
| 2 | /db2/EB8/dbs/tsm_config/vendor.env.2 |
| (nn) | /db2/EB8/dbs/tsm_config/vendor.env.nn |
| 37 | /db2/EB8/dbs/tsm_config/vendor.env.37 |

The reason to use a specific value for each partitions is that a different TSM node names is used per partition number, as depicted in Table 3-14.

Table 3-14 TSM node names pattern for DB2 partitions

| DB2 partition | TSM node names |
|---------------|----------------|
| 0 | tsmserver_00 |
| 1 | tsmserver_01 |
| 2 | tsmserver_02 |
| (nn) | tsmserver_nn |
| 37 | tsmserver_37 |

The idea behind this is to store logs for different DB2 partitions on different tape cartridges using a technique called *collocation*. This configuration is illustrated in Figure 3-34. The collocation is achieved by use of the DB2 partition number to reference a specific TSM node. For further information about the TSM and storage design, go to Chapter 5, “The system storage perspective” on page 317.

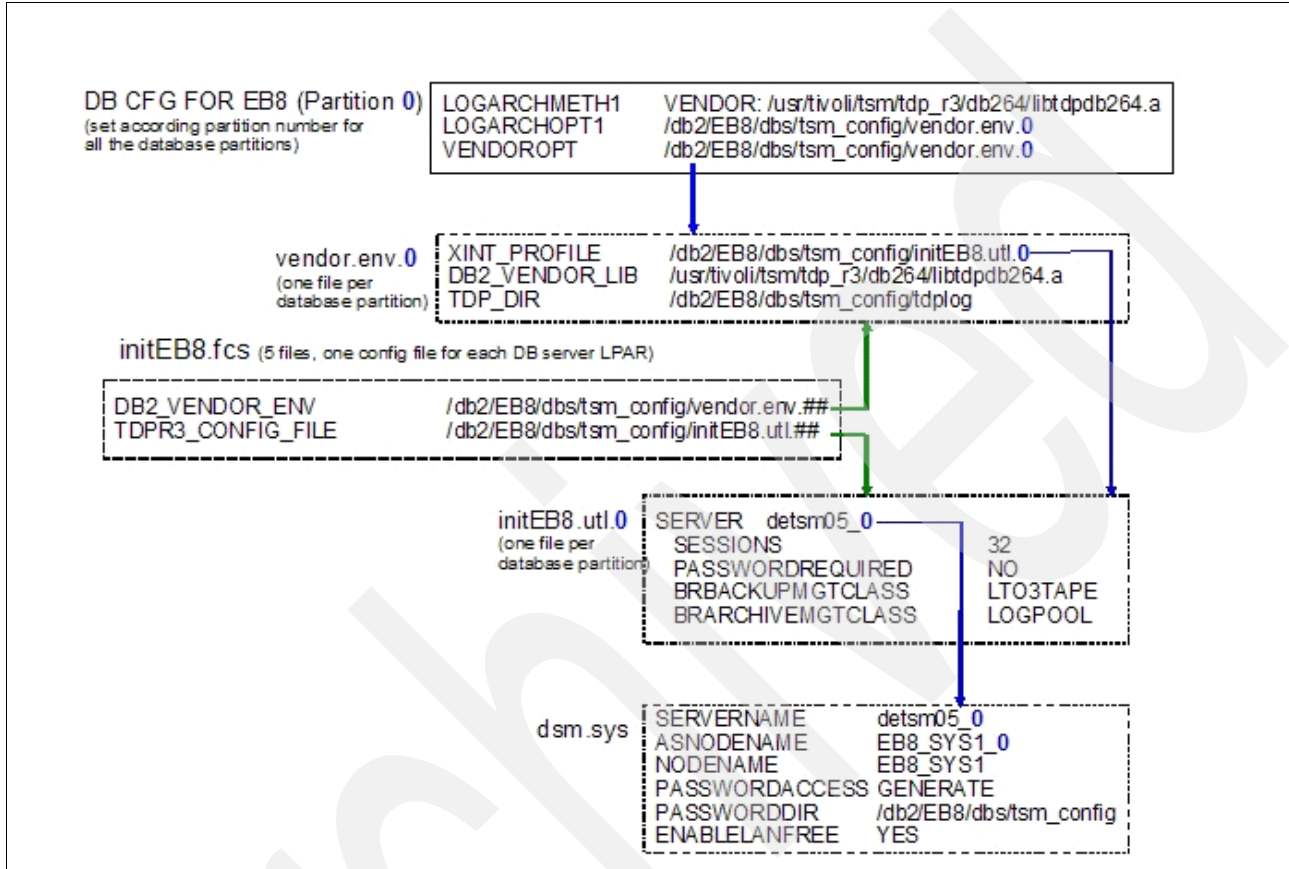


Figure 3-34 Configuration to assign different TSM nodes per partition

3.4 Results from a DB2 specialist’s perspective

This section describes some of the tests and their results from a DB2 specialist’s perspective. Some of the results were very specific to test scalability or design resilience, and others to test new features that were not used previously or are new in DB2 9.

Important: As in any large test, the measurements and results we got in our environment may vary from your environment due to the large and complex number of variables involved.

3.4.1 Stress test results - KPI-G

The main focus of KPI-G was to achieve a set of measurements for upload, aggregation, and online processing in SAP BW.

- ▶ Upload 125 million records per hour
- ▶ Aggregate 25 million records per hour
- ▶ For online processing: queries with average of 2.08 transactions per second and average response time of less than 20 seconds

For further information about this KPI, refer to Chapter 1, “Project overview: business objectives, architecture, infrastructure, and results” on page 1.

This test was executed in SYS1 using the 60 TB database. The database servers involved in the test are:

- ▶ sys1db0: DB2 partition 0
- ▶ sys1db1: DB2 partition 6 to 13
- ▶ sys1db2: DB2 partition 14 to 21
- ▶ sys1db3: DB2 partition 22 to 29
- ▶ sys1db4: DB2 partition 30 to 37

The measurements we discuss in this section were taken during what we called the high load phase: we collected the data from the system while all three processes were running at their full loads. This was done to make it a more consistent measurement and get the closest results to a production environment as possible. The high load period we discuss in this section started at 18:50 and finished at 20:43. For our analysis, we consider 18:50 as our point zero.

Discussions about sys1db0 (DB2 partition 0)

Sys1db0 contained only the catalog, the dimension, and the SAP NetWeaver temporary tables. As the coordinator and catalog DB2 partition, all communication between DB2 AIX LPARs and partitions was coordinated by this partition. The volume of data used in SAP NetWeaver BI is shown in Table 3-15.

Table 3-15 Volume of objects in sys1db0

| Use | Number of tablespaces | Number of tables | Number of pages |
|------------------|-----------------------|------------------|-------------------|
| Catalog | 1 | 105 | 256,991 (4 KB) |
| Dimensions | 1 | 12,503 | 727,022 (16 KB) |
| SAP BW temporary | 17 | 19,008 | 9,872,256 (16 KB) |

Memory consumption

The following figures describe the memory consumption for sys1db0.

Figure 3-35 shows that the memory usage does not vary significantly during the test.

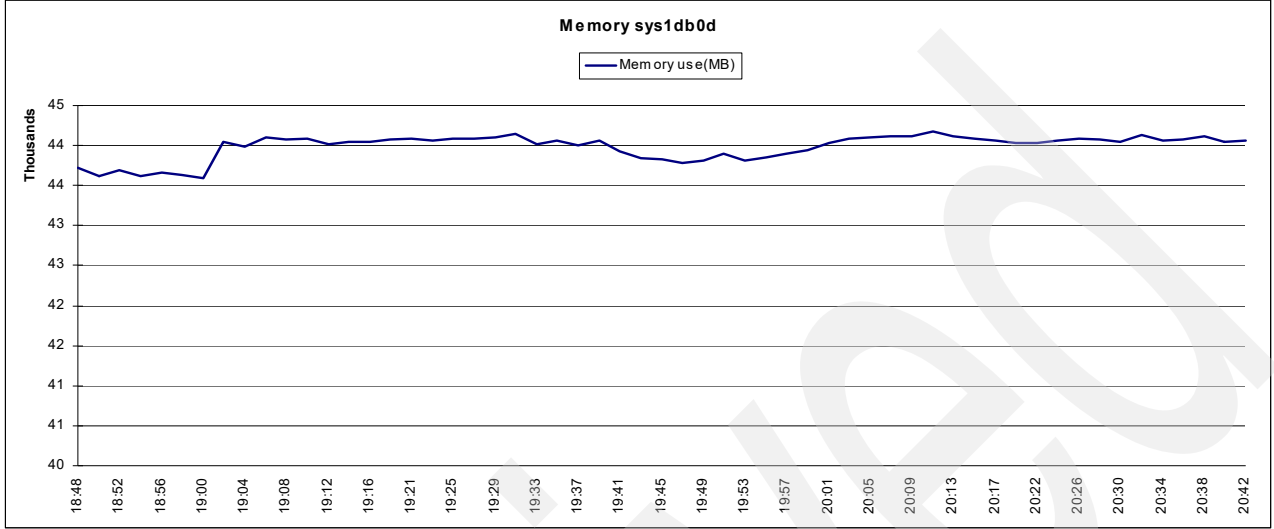


Figure 3-35 Memory usage for sys1db0 during the stress test

Figure 3-36 shows that a considerable slice of memory was used by the file system cache. Although it was put off for the data tablespaces, the catalog tablespaces and the database logs were using the file systems and the file system cache was used for them.

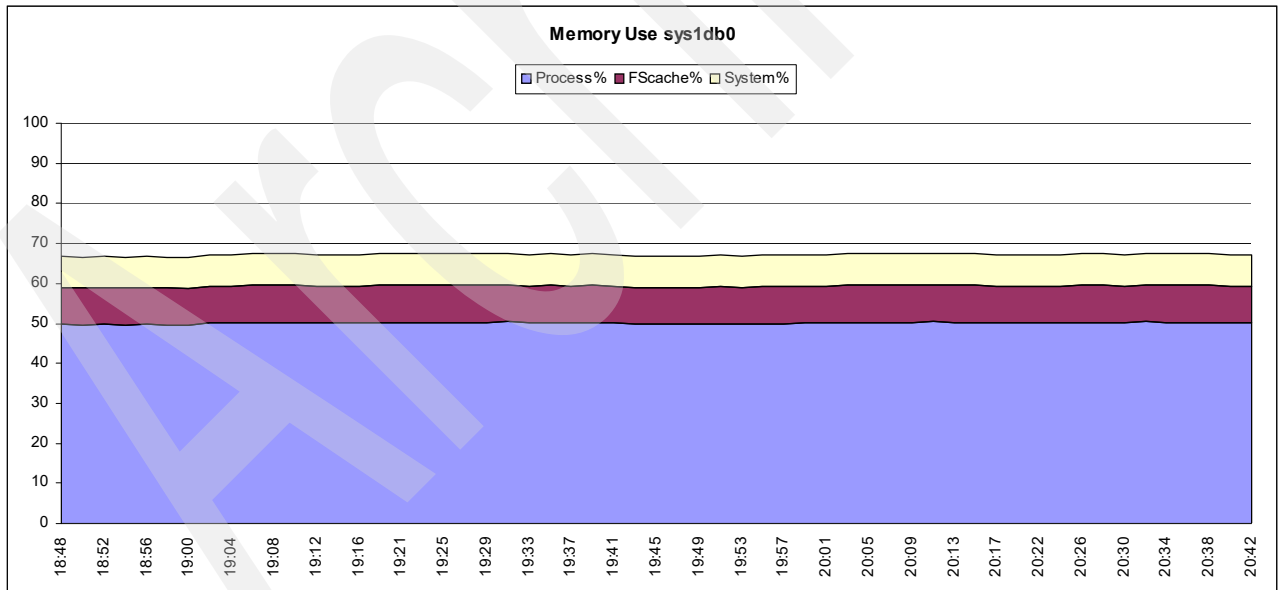


Figure 3-36 Process, file system cache, and system memory usage

I/O usage

The following figures describe the I/O consumption for sys1db0.

Figure 3-37 shows the total disk usage. If you compare the disk usage of this specific server with the remaining servers in the environment (shown in Figure 3-51 on page 219), you can observe that it is much lower, because no large objects are being stored in this partition.

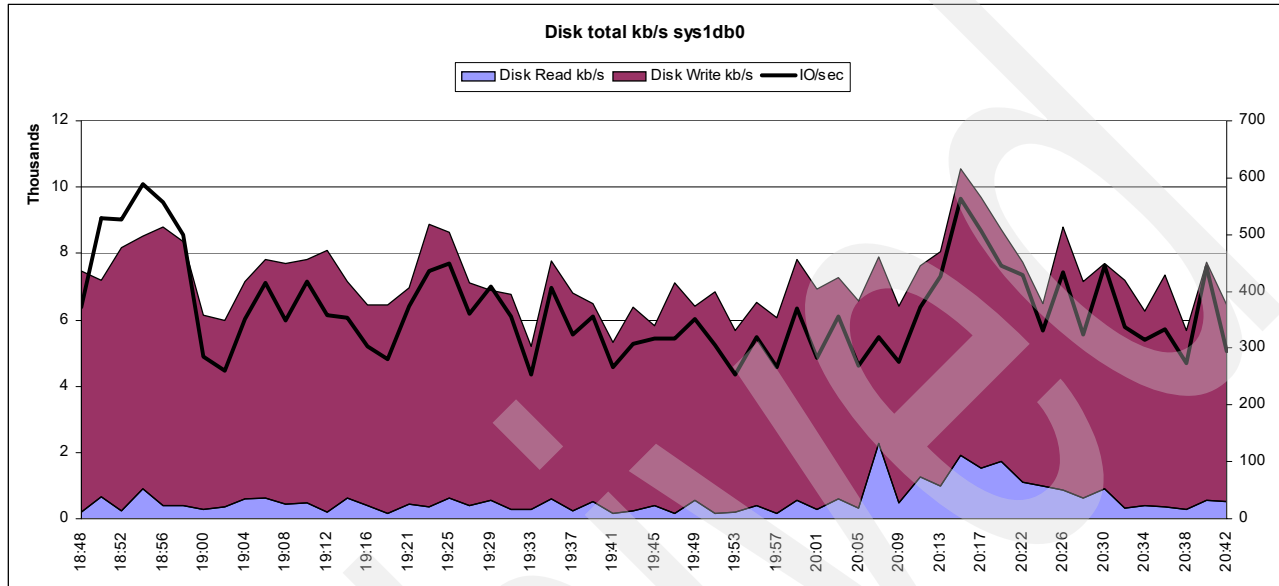


Figure 3-37 Disk total usage in sys1db0

Figure 3-38 is a view of the database disk read usage. This figure shows the top disk read during the run. We can see two distinct sets of disks:

- ▶ The first 12 disks, with the highest number of reads, were used in file systems, hosting the catalog tablespaces. This can explain such a high read usage.
- ▶ The second set of disks was used for the database logs.

To better understand the graph, you should consider that the higher the difference between the average value and the maximum value, the more inconsistency and spikes you will see during the high load period.

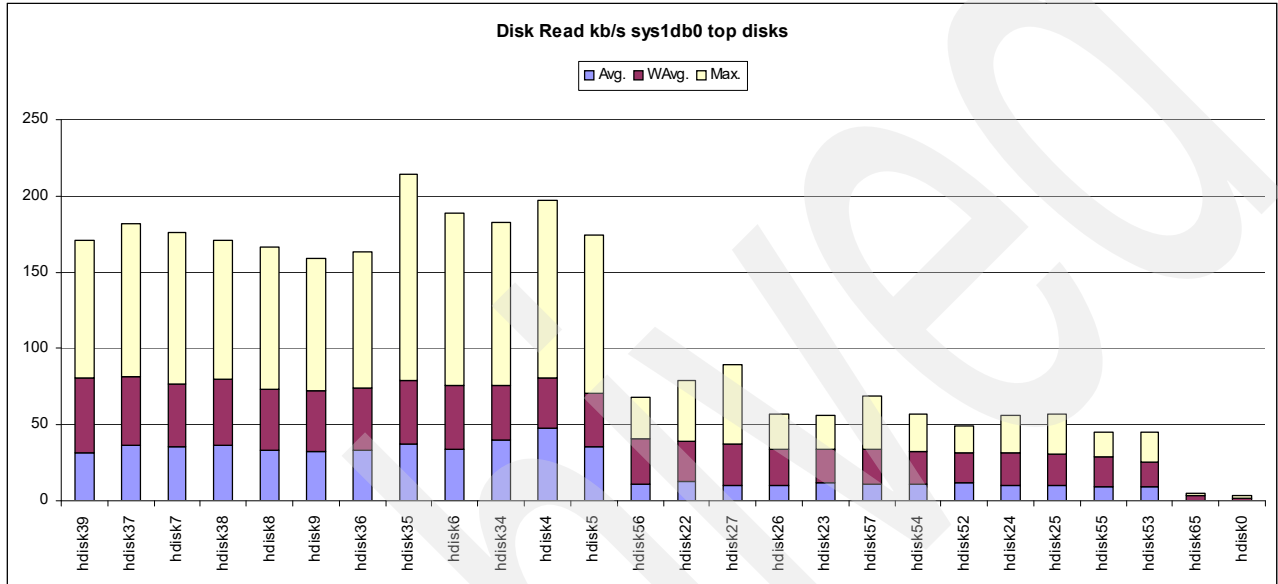


Figure 3-38 Top disk reads in sys1db0

Figure 3-39 shows the disk write usage, where you can see two groups again, but they are inverted compared to the reads. This is because the log activity was intense during the test high load phase and a small amount of data was written on the remaining objects.

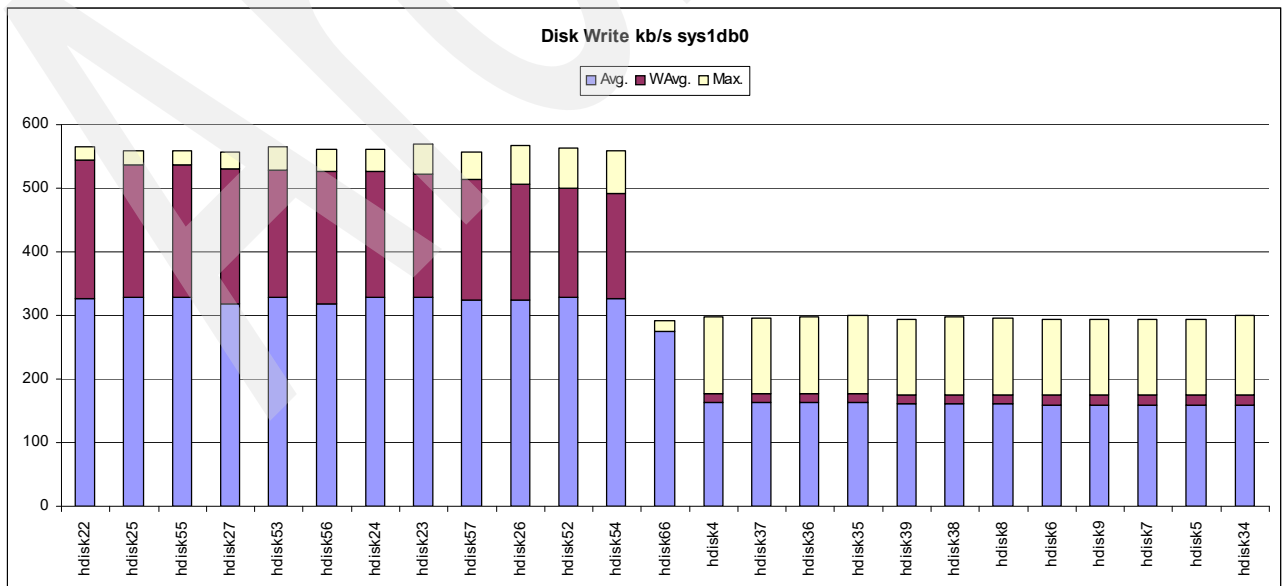


Figure 3-39 Top disk writes in sys1db0

Look at Figure 3-39 on page 210 again. Do you see something different? There are 25 disks. The first 12 disks were allocated for the database log file systems, and the last 12 disks were allocated for the catalog, the SAP NetWeaver BI temporary and dimension tablespaces. The disk that does not match with the remaining is hdisk66. It is used by the file system where DB2 writes its db2diag.log file. You need to consider that the write volume is considerably low compared to the other DB2 AIX LPARs, and that the DB2 dump file system is shared across all DB2 AIX LPARs and is used for the log messages in db2diag.log by all DB2 partitions. This also explains the consistency of hdisk66 writes. Also, the maximum value is very close to the average for the high load period.

Network usage

The network usage in sys1db0 should be the highest one comparing to the remaining DB2 AIX LPARs, because it hosts the coordinator and the catalog DB2 partition. This means that all remaining DB2 partitions communicate with the coordinator, causing network usage to be much higher on this server than on the others. Also, the SAP application server connects and gets information from the database by DB2 partition 0. It is important to highlight though that networks used on these tasks were different ones, to avoid any bottleneck.

The following figures describe the network usage.

Figure 3-40 shows the total network usage. We see that the network writes are higher than reads due to the aggregation of the DB2 partitions communications network with others.

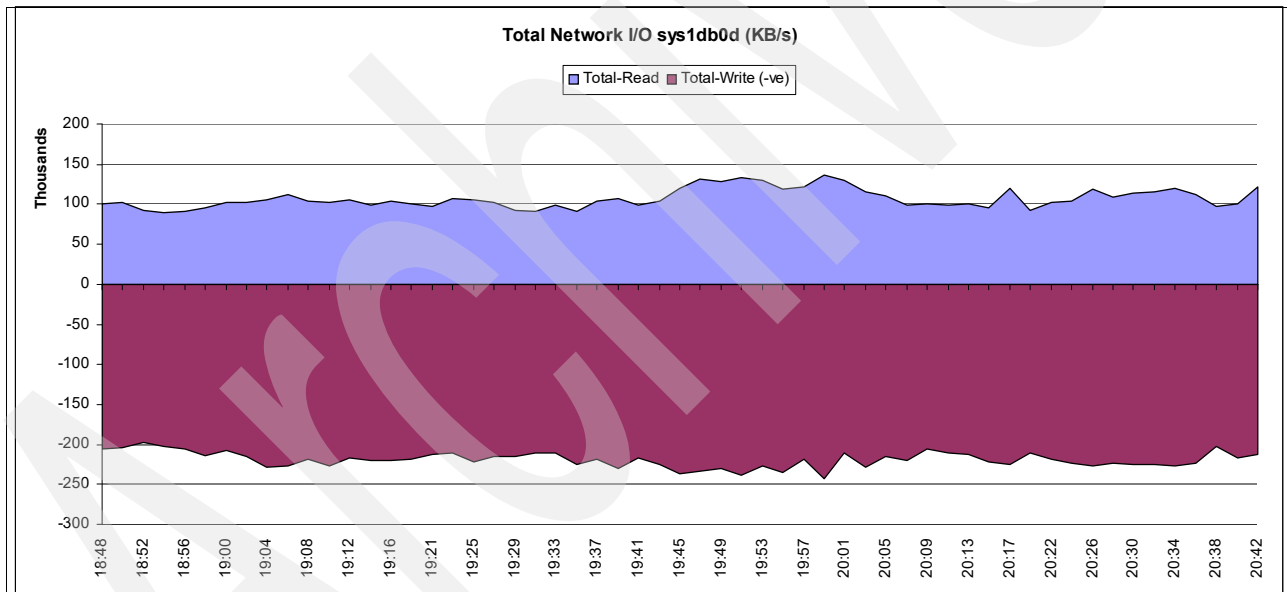


Figure 3-40 Total network usage in sys1db0

To better understand the network usage, we split its usage through different back-ends:

- Back-end 1: communication between DB2 partitions

The back-end 1 network was used only to enable the communication between the DB2 partitions. The usage of this particular network is shown in Figure 3-41 on page 212. We see that the volume of writes is considerably higher than the volume of reads. This is due to the high volume of data being inserted and hashed over DB2 partitions. The data either comes from SAP NetWeaver BI as records from the upload task or they are inserting data from selects over other tables (like aggregate task) where most of it is executed without any data getting out from the database. Data must be retrieved from the partitions, but the volume of data sent back to the coordinator is considerably lower since the optimizer

chooses the access plan that can do as much processing as possible on each DB2 partition and then send data back to the coordinator.

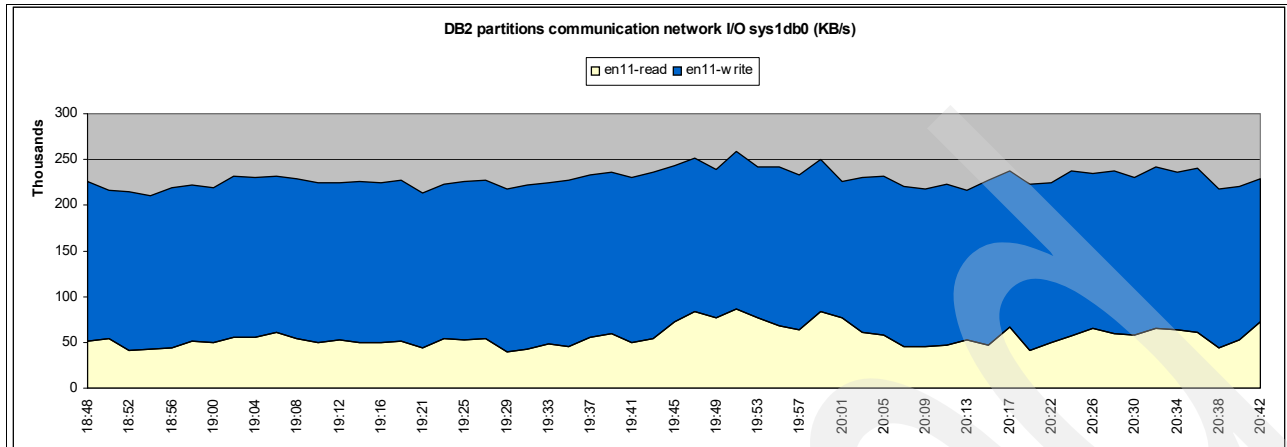


Figure 3-41 DB2 partitions communication usage

The best case to illustrate this behavior is a collocated join. A collocated join occurs locally on the database partition where the data resides. The database partition sends the data to the other database partitions after the join is complete inside the DB2 partition. For the optimizer to consider a collocated join, the joined tables must be collocated, and all pairs of the corresponding distribution key must participate in the equality join predicates. This is most of the case for tables used in our test.

- ▶ Back-end 2: communication between SAP BW instances and DB2 coordinator partition

The second back-end network created was used to enable the communication between the SAP NetWeaver instances and the DB2 coordinator partition. Its usage during the run is shown in Figure 3-42. We observe that the usage is much lower than the network usage for the DB2 partitions communication.

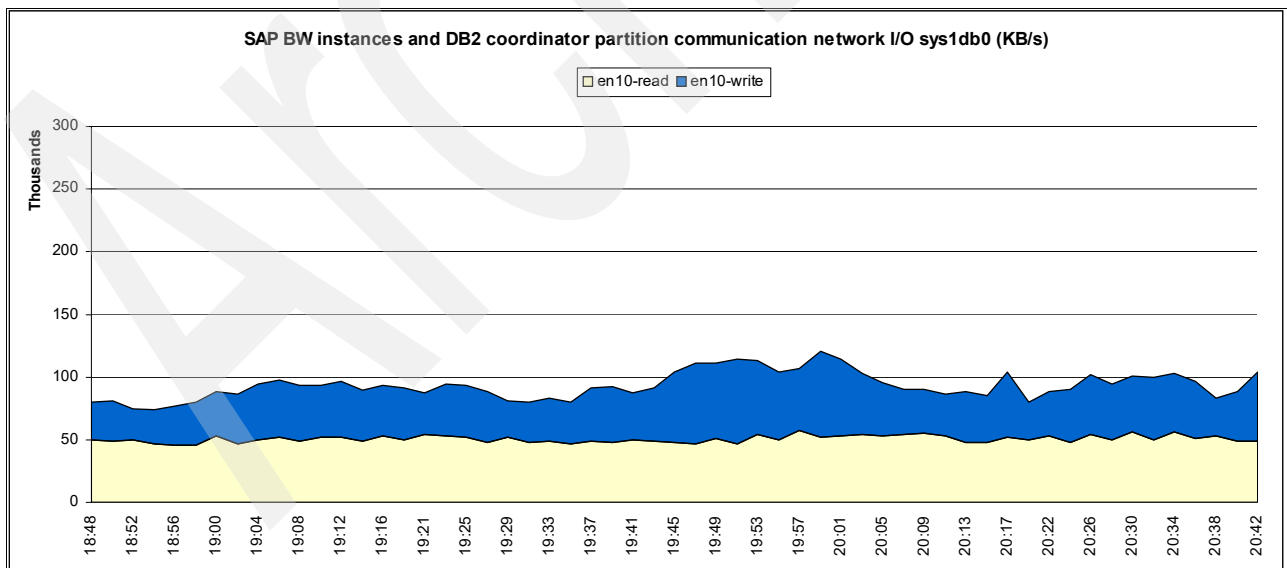


Figure 3-42 SAP BW instances and DB2 coordinator partitions communication network usage

- ▶ Front-end: administration, control, and user access network

The last network used in our DB2 AIX LPARs was used only for administration and controlling purposes, basically only ssh, sftp, Telnet, and ftp were executed over this network. Its usage is shown in Figure 3-43. For an effective view of it, due to its small usage, we changed the scale in the graphic. Most of the spike seen for writing was due to monitoring tools that, on a frequent basis, refreshed database information.

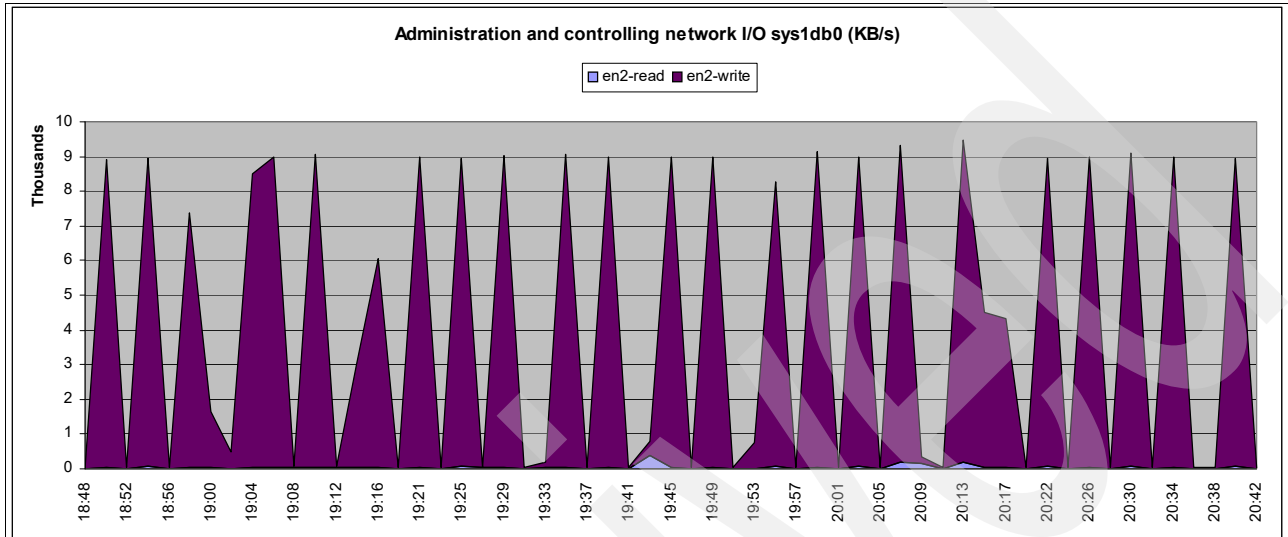


Figure 3-43 Front-end administration and controlling network usage

CPU usage

CPU usage on sys1db0 is expected to be considerably lower than the remaining DB2 AIX LPARs. It is shown in Figure 3-44. We see that two graphics are shown in the figure. It would not be fair if we use only the CPU percentage usage, since we rely on the power of virtualization to optimize use of resources. The second graphic shows how the number of logical CPUs varied during the run, giving a good idea of the CPU percentage versus the number of logical CPUs. It is worth checking the dynamic and the resilience this feature brings to the entire environment.

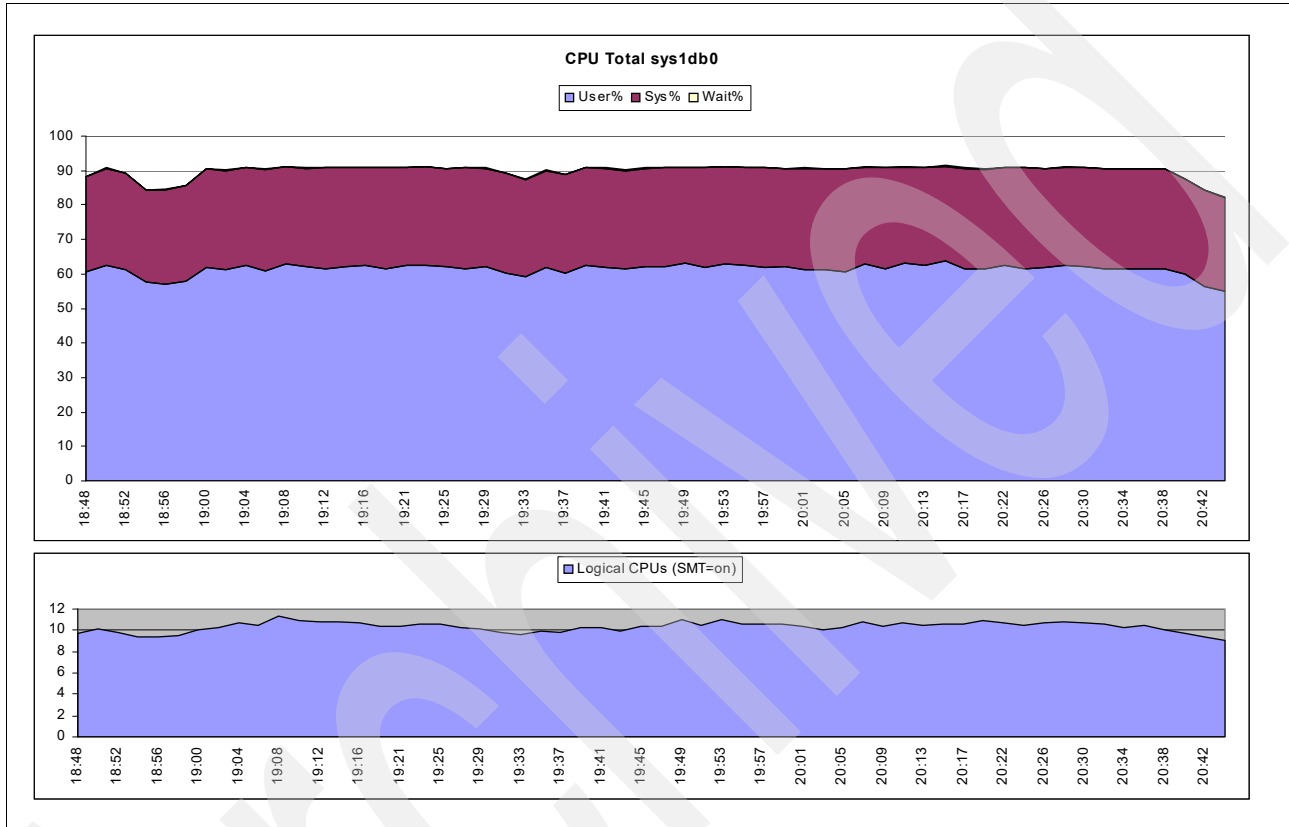


Figure 3-44 CPU usage for sys1db0

Database performance and usage

This section discusses the database behavior, and the following figures highlight the most important aspects.

Figure 3-45 shows the overall buffer pool hit ratio for all buffer pools for the DB2 partition 0. This ratio was high, with a hit ratio higher than 99%. This is mainly because buffer pools were well tuned with objects for this partition. Keep in mind as well that the objects in this partition are small compared to the remaining objects used in SAP NetWeaver BI.

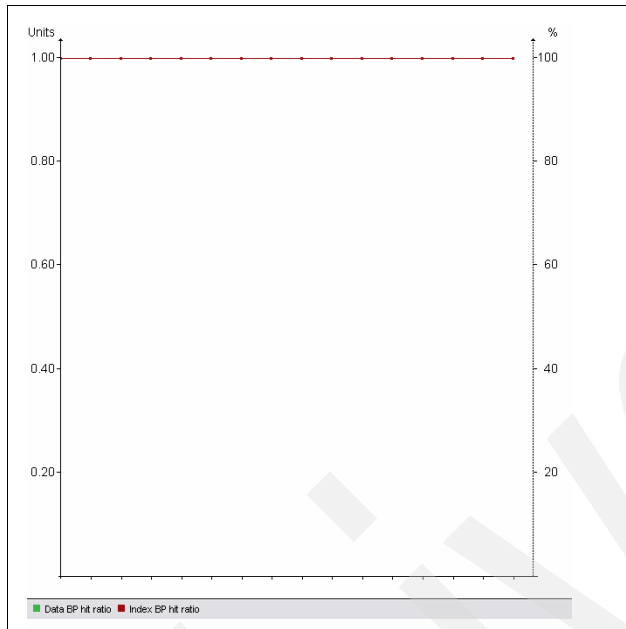


Figure 3-45 Buffer pool hit ration for DB2 Partition 0

Figure 3-46 shows the number of applications connected to the database.

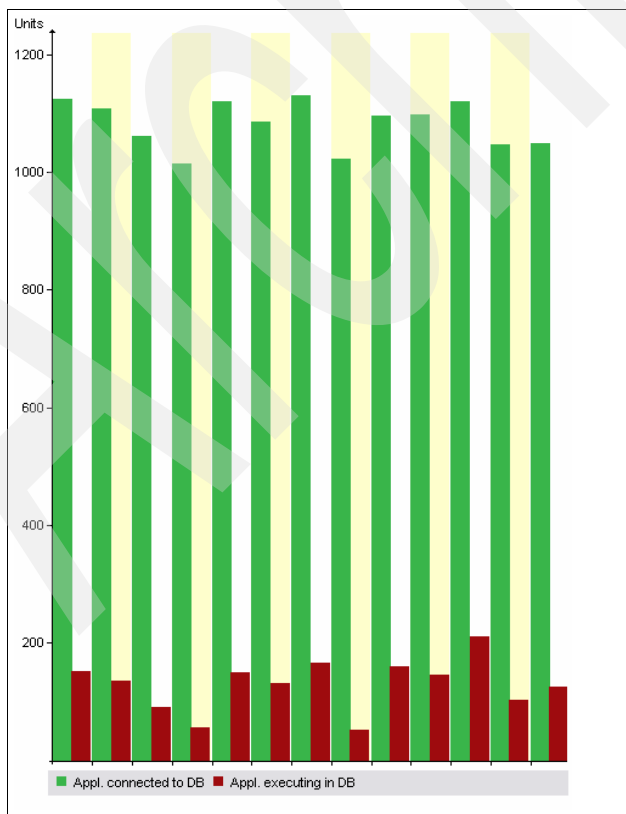


Figure 3-46 Number of applications connected versus executing

Figure 3-47 shows the average number of locks held per application during the high load phase.

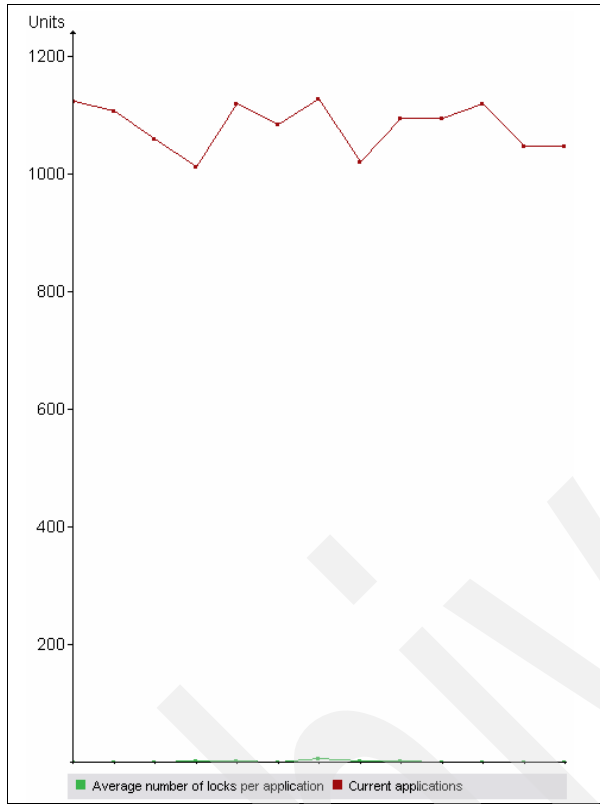


Figure 3-47 Average number of locks per application

Figure 3-48 shows the very low amount of sort overflow that occurred during the run, compared to the number of sorts.

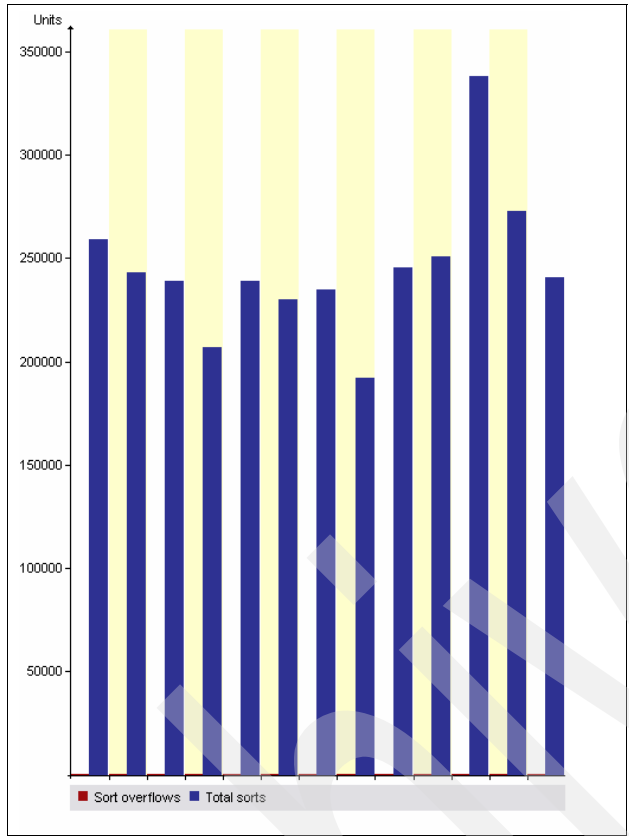


Figure 3-48 Sort overflows versus total sorts

Discussions about sys1db1(DB2 partitions 6 to 13)

All large objects used in SAP NetWeaver BI are spread over eight DB2 partitions (6 to 13).

Memory consumption

The memory consumption during the tests is described in the following figures.

Figure 3-49 shows the memory usage, which did not vary considerably comparing to the variance percentage with the total size used.

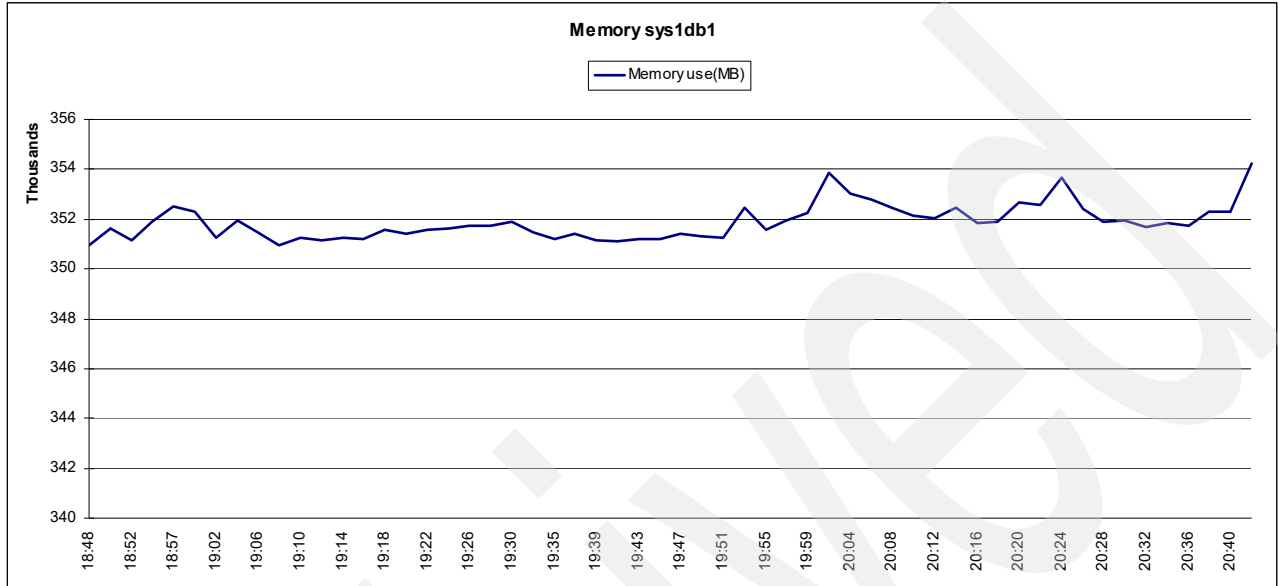


Figure 3-49 Memory usage for sys1db1 during the stress test

As shown in Figure 3-61 on page 227, a slice of memory was used by the file system cache. The database logs used file systems and a file system cache. As the opposite of sys1db0, memory was configured in DB2 partitions to use approximately 90% of the total real memory available.

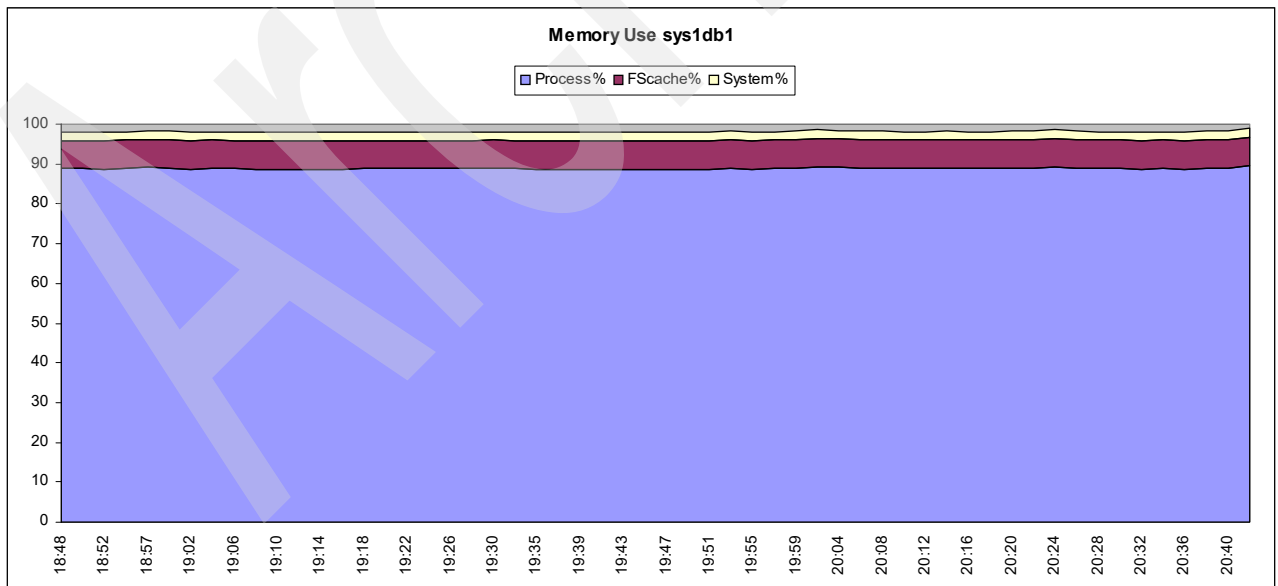


Figure 3-50 Process, file system cache, and system memory usage

I/O usage

I/O usage on sys1db1 is much higher than sys1db0 because all SAP NetWeaver BI large objects are spread over the DB2 partitions that are hosted on this particular server.

Figure 3-51 shows the I/O consumption. If you compare the disk usage of this specific server with the remaining servers in the environment, you can see that it is much lower, because no large objects are stored on it.

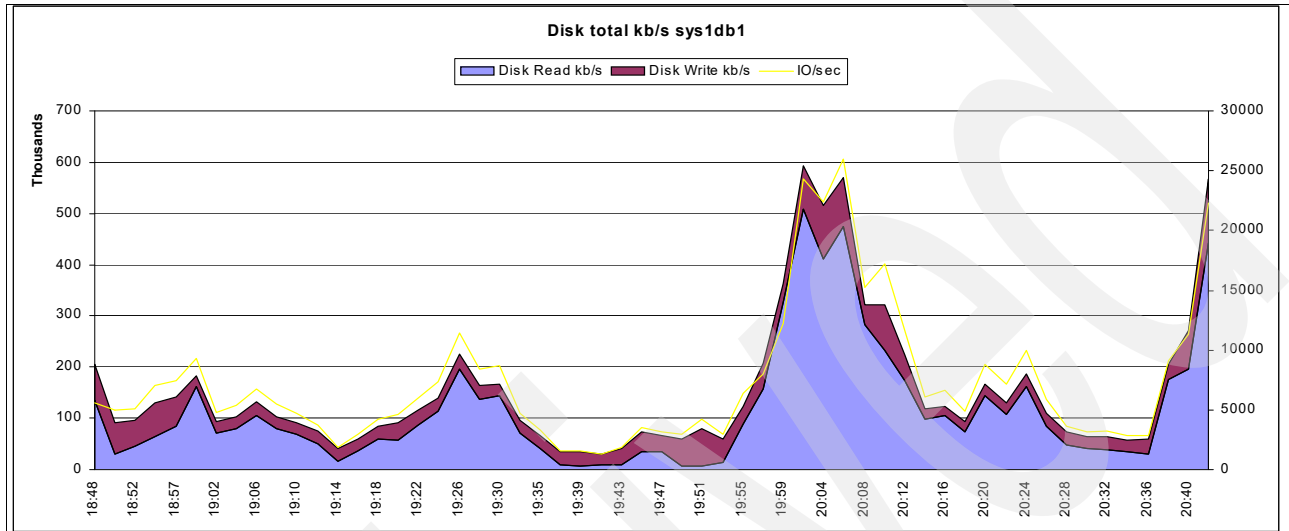


Figure 3-51 Disk total usage in sys1db1

Let us take a different approach from the one we used to analyze disk usage on sys1db0. We go through the analysis of their volume groups. This is easier because of the usage of naming rules tells us in which DB2 partition the volume group was being used, as shown in Figure 3-52. Three volume groups per DB2 partition can be analyzed:

- ▶ The first one, called DB2PXX_DATA_VG (where XX represents the partition number) is the one used by PSA, ODS, and InfoCube fact tablespaces containers. This is why the highest amount of read is concentrated on these volume groups.
- ▶ The second volume group created, called DB2PXX_TEMP_VG, is the one used for a tablespace, named PSATEMP16K, and is the default temporary tablespace created and used by all DB2 partitions
- ▶ The third, named DB2PXX_LOG_VG, is used to store the partition logs.

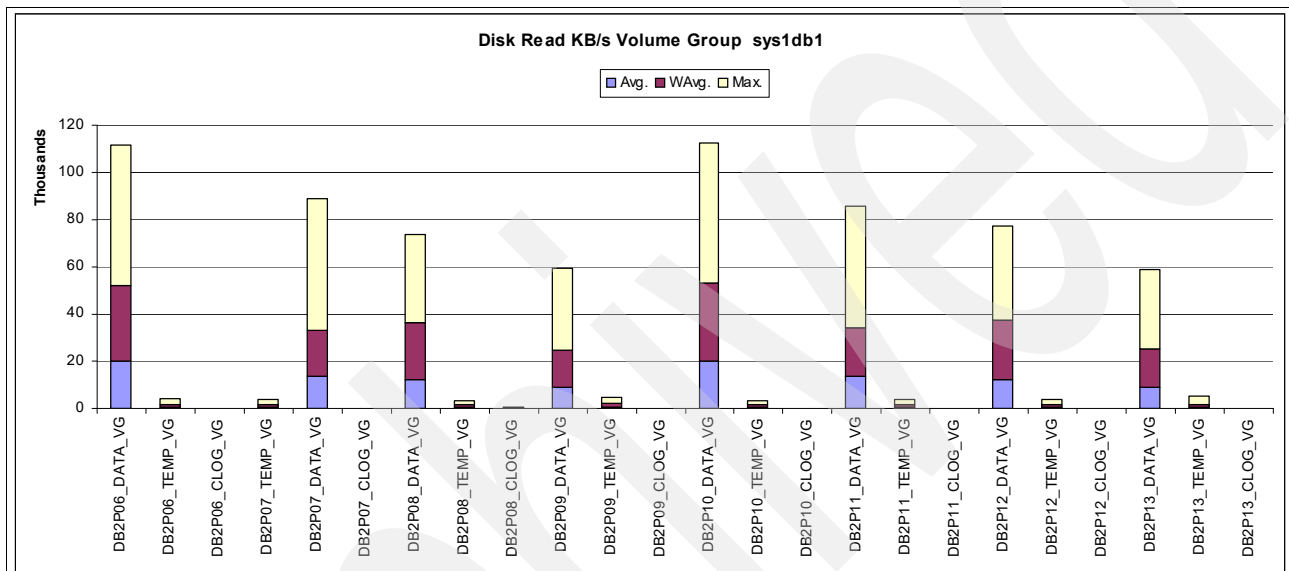


Figure 3-52 Disk read volume group in sys1db1

We observe that the read balancing did not achieve the best result. DB2 partitions 6 and 10 show the highest disk read usage.

Figure 3-53 is a pie chart that shows the percentage of all data read over the high load phase and the volume groups percentages.

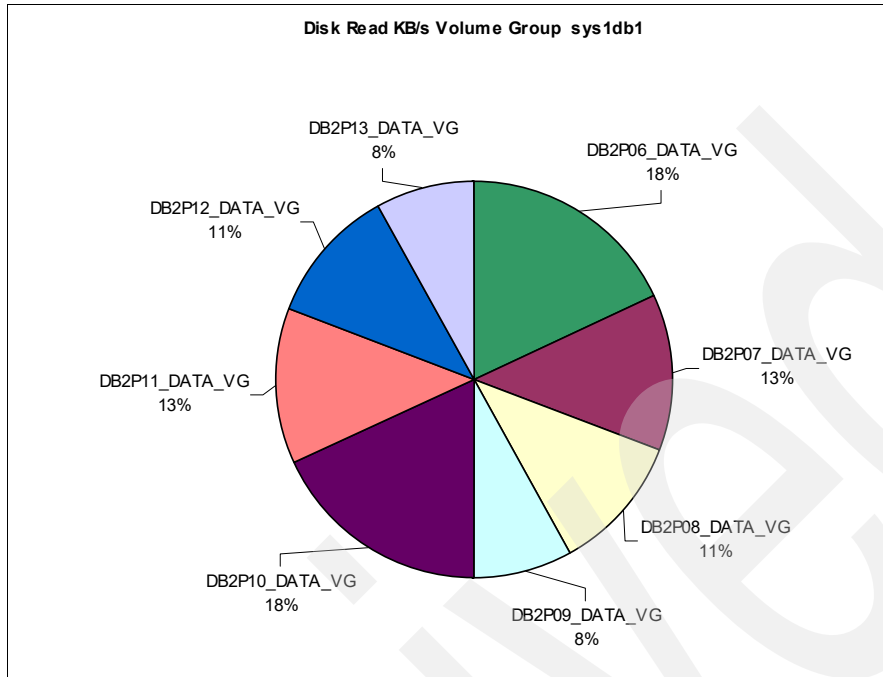


Figure 3-53 Disk read per data volume group for sys1db1

This can be extended to the remaining DB2 AIX LPARs. To make sure that this assumption was correct, we went through the disk read for data volume groups in all DB2 AIX LPARs, as shown in Figure 3-54.

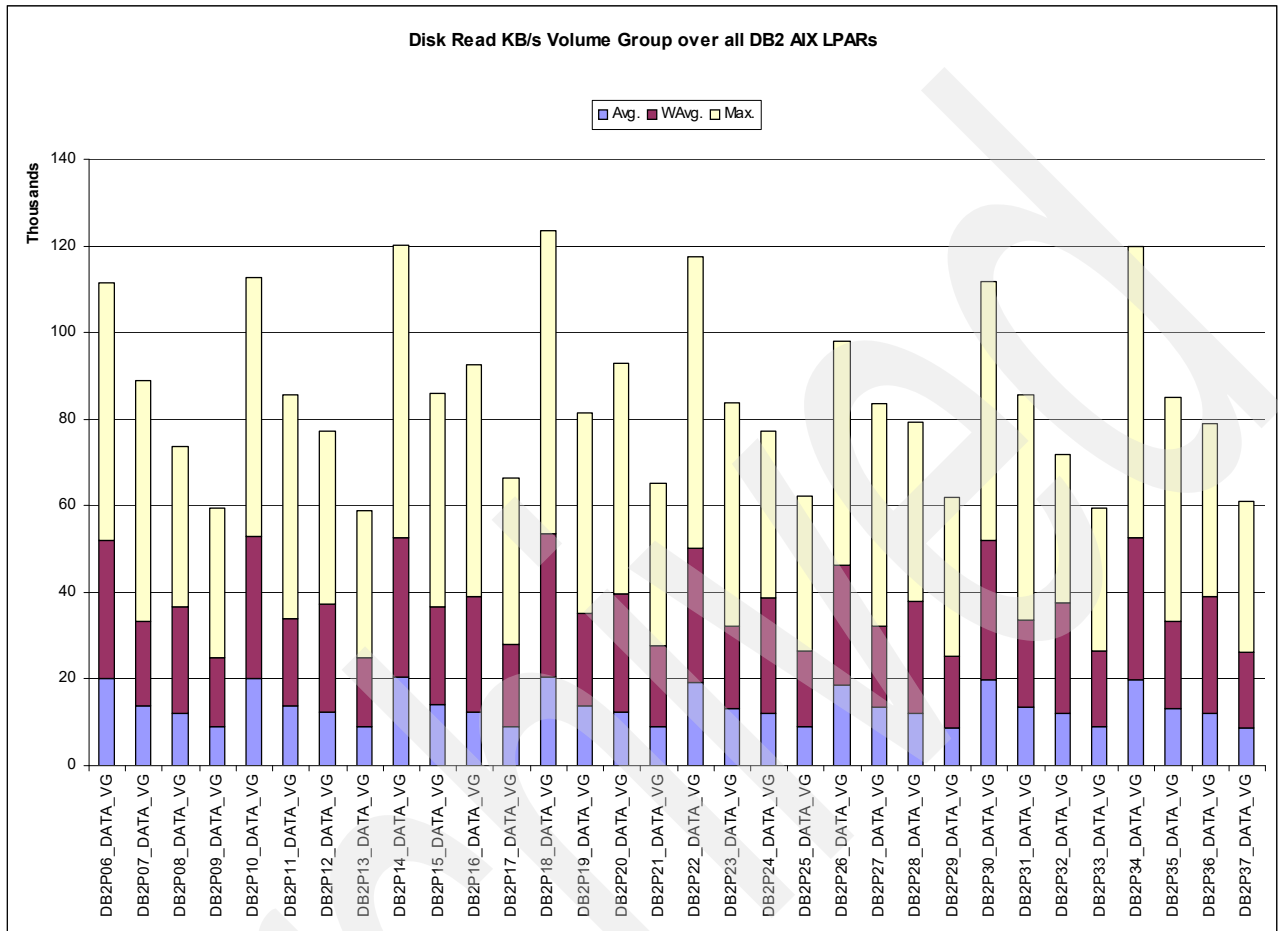


Figure 3-54 Disk read usage for data volume groups over all DB2 AIX LPARs

To identify the balance deviation, Figure 3-55 shows the overall distribution for disk read of large objects.

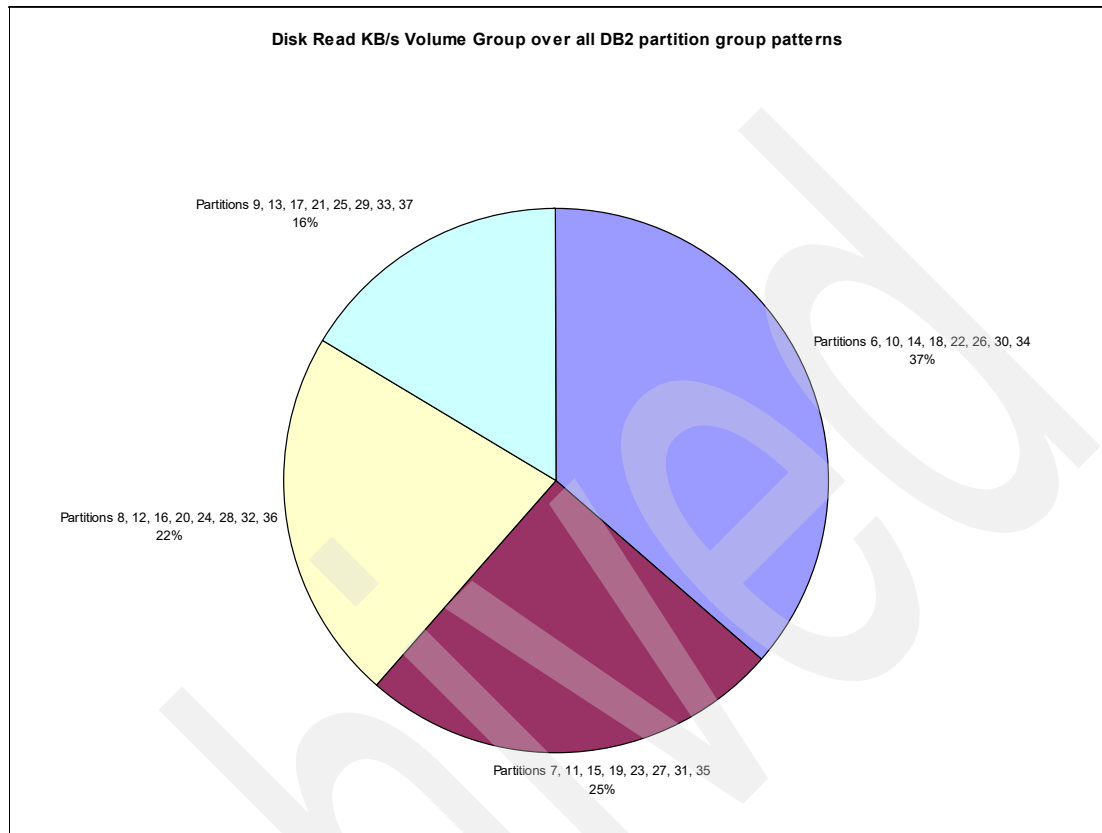


Figure 3-55 DB2 partition group patterns and disk read volume group balancing

Some optimizations may have helped to distribute more evenly disk read, but due to time constraints, no further tests were executed.

The disk write volume groups usage is shown in Figure 3-56. From this disk write perspective, balancing is much more efficient. The highest usage is for temporary tablespaces due to the amount of activities running on the system.

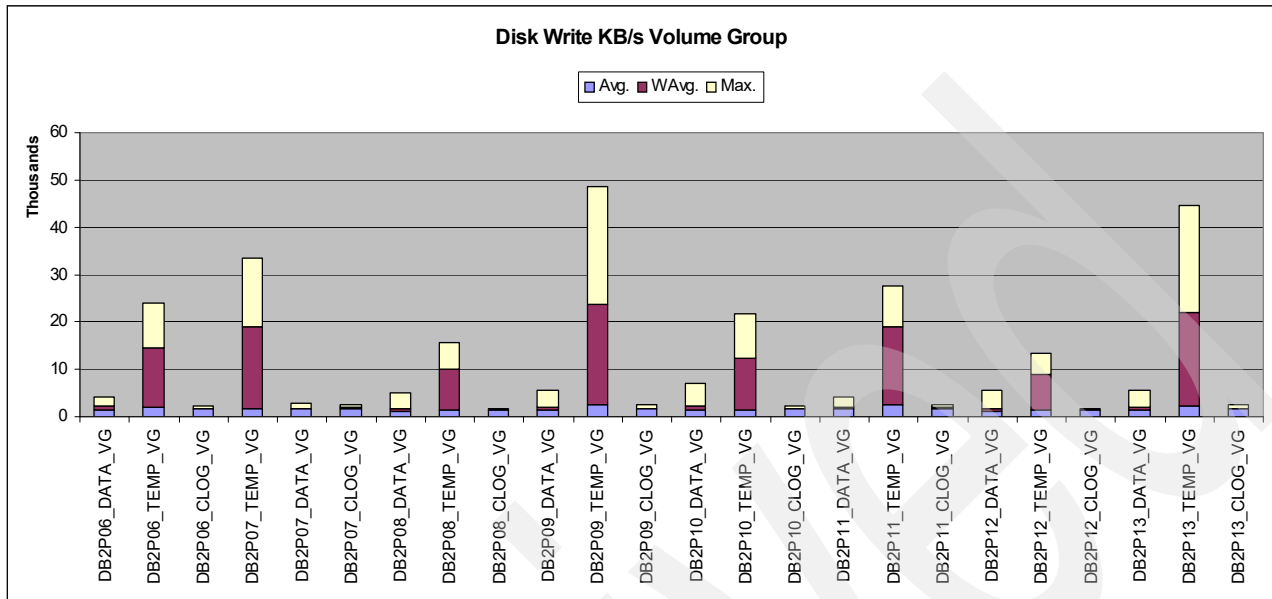


Figure 3-56 Top disk writes in sys1db1

The average disk write usage in all three distinct volume groups is very similar, as shown in Figure 3-57. We see that the usage is slightly higher on the temporary volume groups, followed by the logs, and then the data.

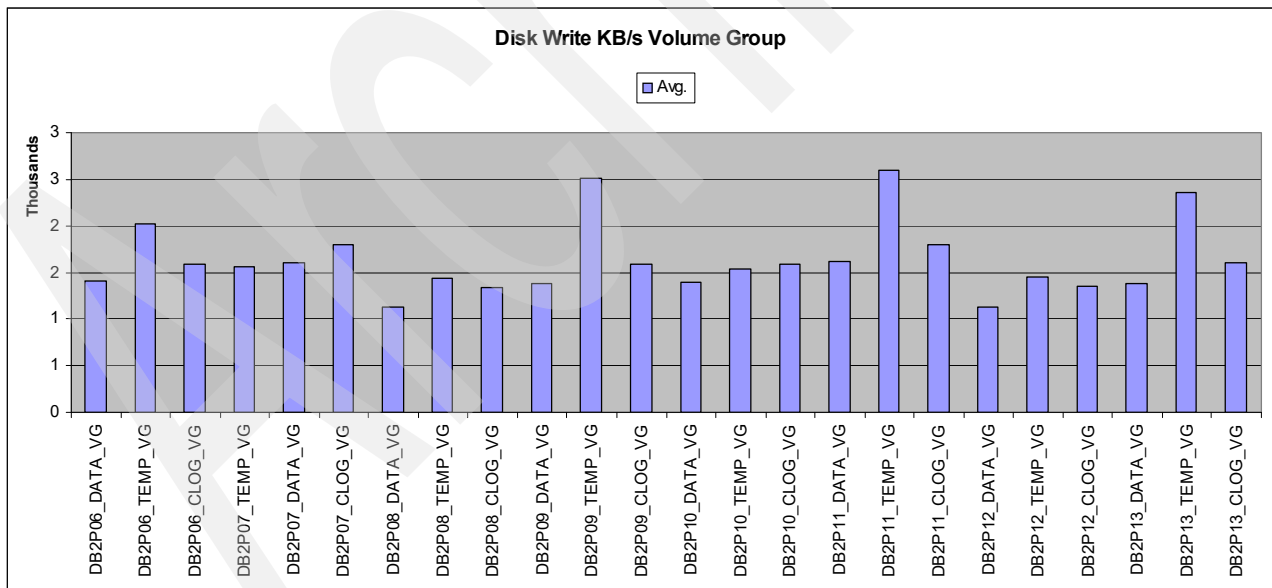


Figure 3-57 Average disk write for volume groups

Network usage

Network usage on this specific DB2 AIX LPAR is considerably lower than the usage on sys1db0 since the only communication was with the coordinator node.

Figure 3-58 shows the summary for the total network usage. We see that the network usage is more balanced between read and write, compared to sys1db0. There is a slightly higher number of network reads than writes.

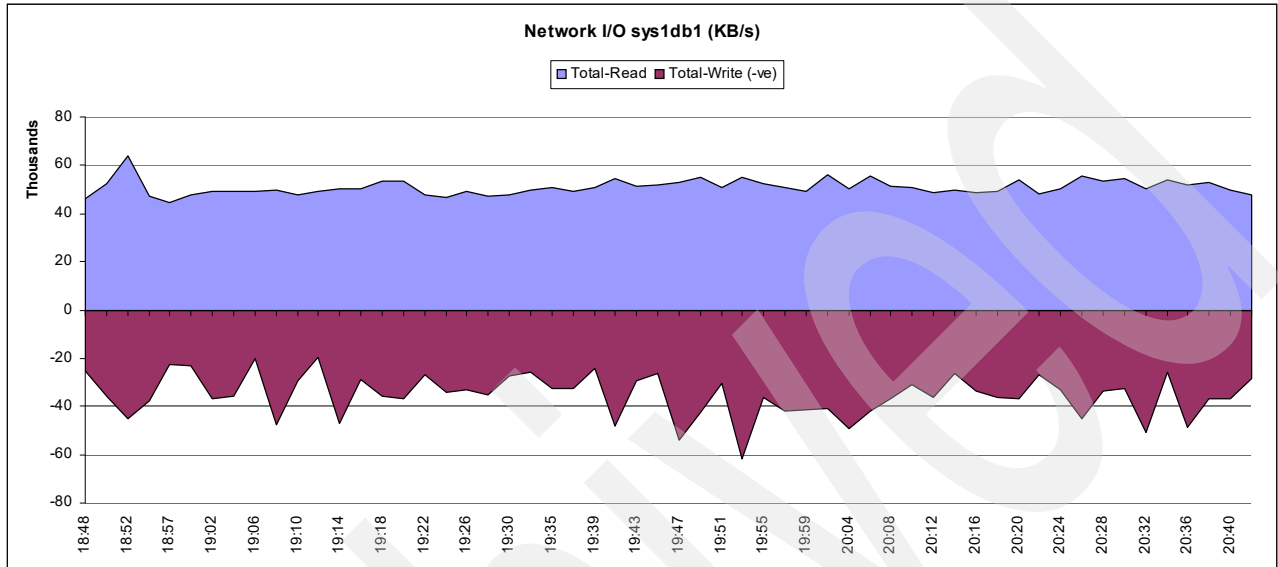


Figure 3-58 Total network usage in sys1db1

In sys1db1 we have two different networks:

- ▶ A back-end network that is a DB2 partitions communication network

The back-end network was used only to enable communication between local DB2 partitions with the DB2 coordinator partition. Most of the network usage was derived from this network, as shown in Figure 3-59.

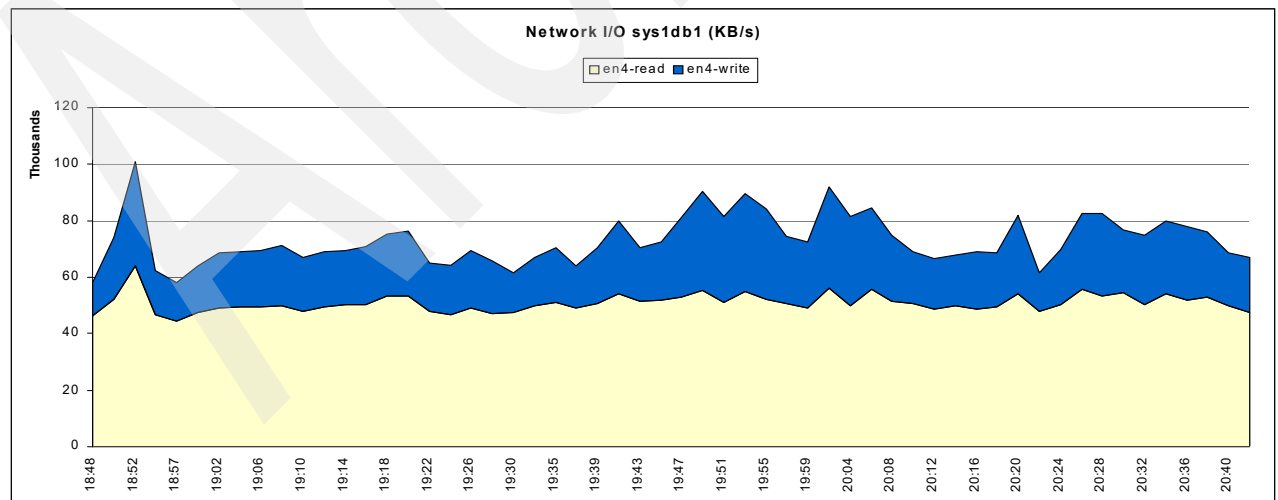


Figure 3-59 DB2 partitions communication usage

- ▶ A front-end network for administration, control, and user access network

This was used only for administration and controlling purposes, basically, only ssh, sftp, Telnet, and ftp were executed over this network. This is shown in Figure 3-60, but to have an effective view of it, due to its small usage, we changed the scale in the graphic. Most of the spike look for writing was caused by monitoring tools that refreshed database information on a frequent basis.

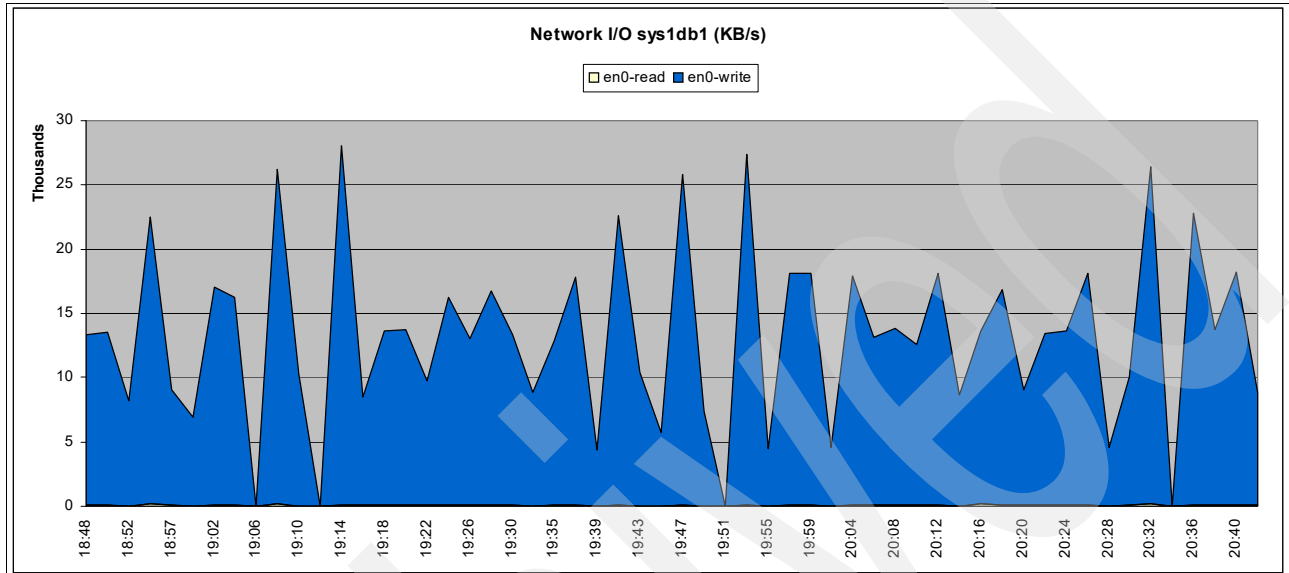


Figure 3-60 Front-end administration and controlling network usage

CPU usage

CPU usage on sys1db1 was higher than sys1db0, and followed the same pattern as the remaining DB2 AIX LPARs, which denotes a well-balanced environment from the CPU usage point of view. This is shown in Figure 3-61 on page 227. Two graphics are shown in the figure. It would not be fair if we used only the CPU percentage usage, since we rely on the power of virtualization to optimize use of resources. The second graphic is how the number of logical CPUs varied during the run, giving a good idea of the CPU percentage versus the number of logical CPUs. Due to the distinct activities running against the database, the variance is considerably high, even being only evaluated during the high load phase. It is worth checking the dynamic and resilience that virtualization brings to the entire environment. A slice of the graph also gets some attention, the file system cache. The reason for having this slice even disabling file system cache for tablespaces can be explained by the high amount of logs generated, which still use the file system cache.

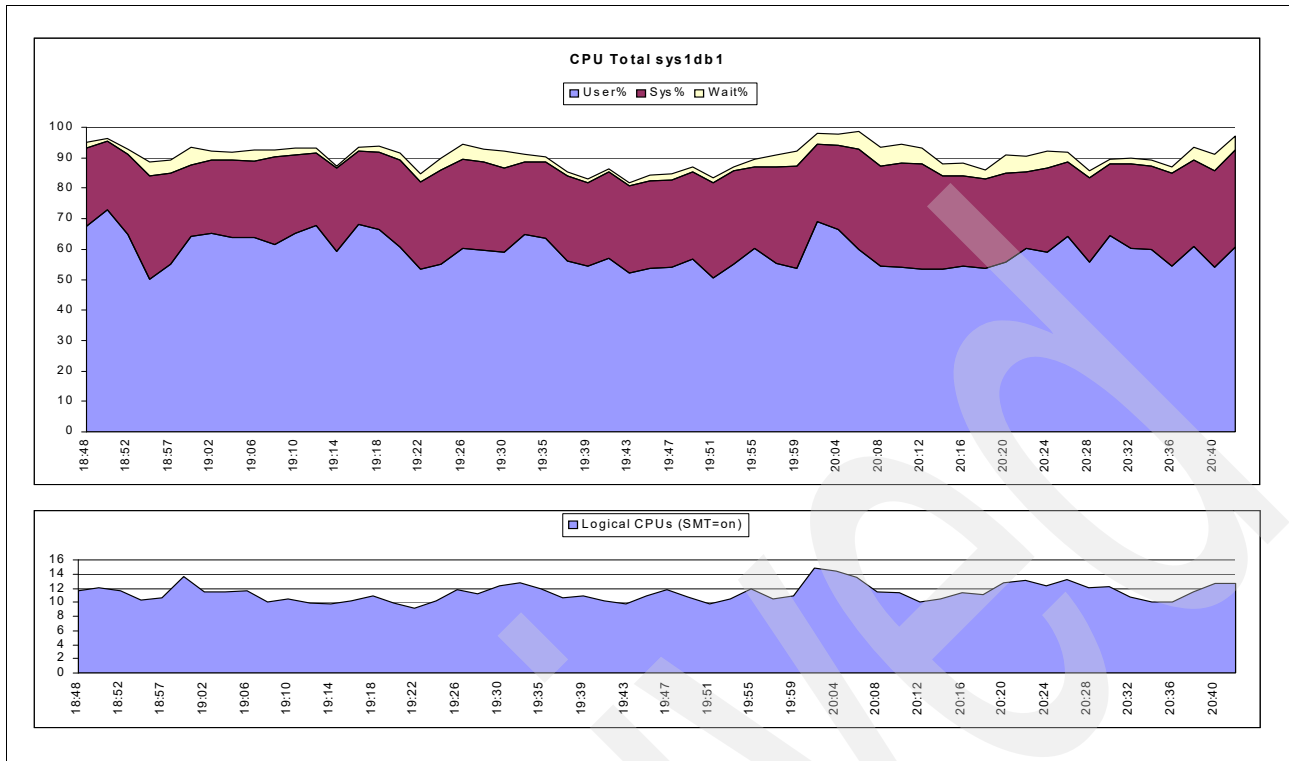


Figure 3-61 CPU usage for sys1db1

Database performance and usage

The data buffer pool hit ratio had the most critical measurements. Since objects distributed across the DB2 partitions are considerable large, the BP_STD_16K buffer pool (used by ODS, PSA, and InfoCubes fact tablespaces) does not have a good hit ratio on these partitions. For InfoCubes aggregates, the buffer pool hit ratio (either for data or index) was high.

Some things may have helped increase the data buffer pool hit ratio for BP_STD_16K:

- ▶ Split objects that are so large and accessed so sporadically that they would never be able to take advantage of buffer pool, simplicity resulting in a high volume of data being spilled out of buffer pool memory.
- ▶ Create larger buffer pools in an attempt to accommodate memory for these objects.
- ▶ Use DB2 9 data row compression (we did some tests on this direction. For further information refer to “Stress test comparison with and without compression” on page 247.

Note: The same observations regarding the memory, the network, the CPU usage, and the database performance could be done for the sys1db2, sys2db3, and sys1db4 DB partitions.

Conclusion

The results for this test came up with a positive achievement of all pre-established measurements. The overall results are shown in Table 3-16.

Table 3-16 Test results overall numbers

| Metric | Value |
|---------------------------------------|------------------------------|
| Average load per hour | 125.06 million records |
| Average aggregation per hour | 25.03 million records |
| Average query transactions per second | 2.16 transactions per second |
| Average query response time | 15.5 seconds |

The overall snapshot for the run is depicted in Figure 3-62.

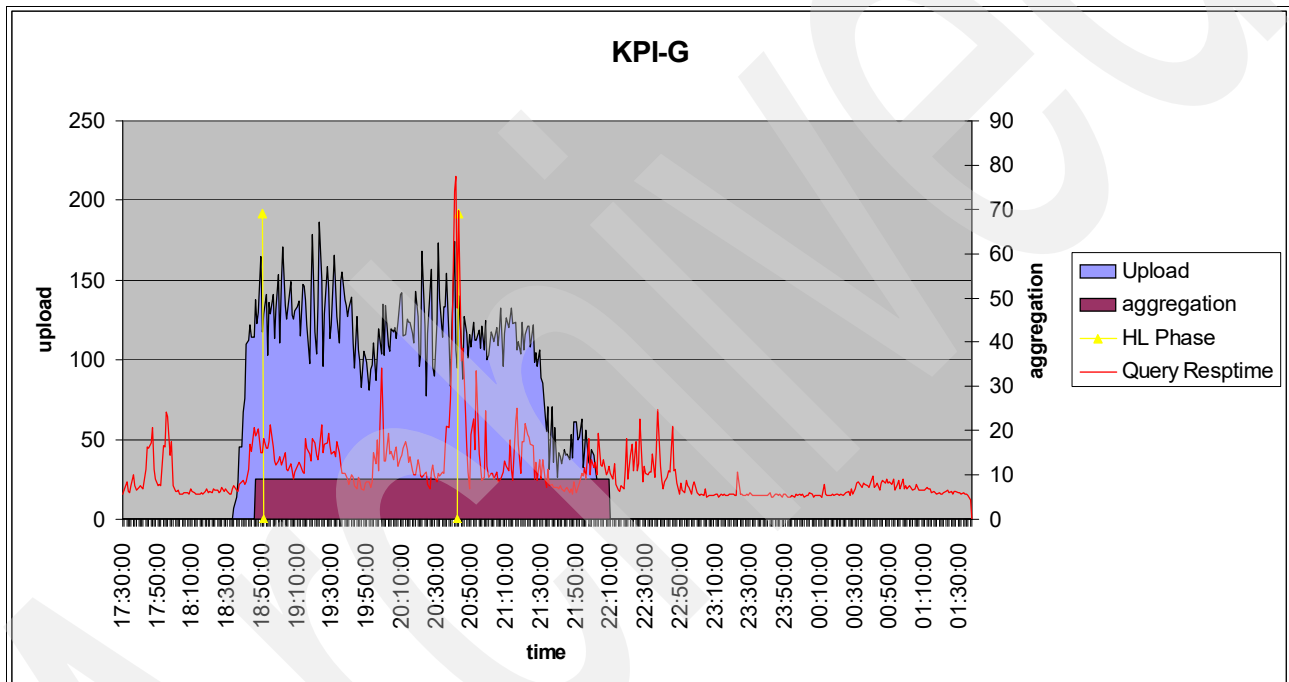


Figure 3-62 KPI-G overall snapshot

3.4.2 Tests with DB2 data row compression

DB2 9 introduces several improvements and new features like self-tuning memory manager, pureXML, and LBAC. Row compression is a very powerful new feature. In the following sections we discuss tests and results when using the row compression feature.

SYS2 was used for these tests, with the following DB2 AIX LPARs:

- ▶ sys2db0: DB2 partition 0
- ▶ sys2db1: DB2 partitions 6 to 13
- ▶ sys2db2: DB2 partitions 14 to 21
- ▶ sys2db3: DB2 partitions 22 to 29
- ▶ sys2db4: DB2 partitions 30 to 37

The art of compressing data

Compressing data is based on a dictionary that maps a data row value with shorter symbol strings, allowing you to reduce space usage. Data row compression uses a static dictionary-based compression algorithm to compress data by row. Compressing data at the row level allows you to repeat patterns that span multiple column values within a row and to replace them with shorter symbol strings.

- ▶ In order to compress table data, the table COMPRESS attribute must be set to YES and a compression dictionary must exist for the table.
- ▶ To build a compression dictionary (and subsequently compress a table), you have to perform a classic (offline) table reorganization. All the data rows that exist in a table participate in the building of the compression dictionary. The dictionary is stored along with the table data rows in the data object portions of the table.
- ▶ To decompress a table, set the table COMPRESS attribute to NO and then perform a classic (offline) table reorganization.

A data row that is inserted into a page may be compressed if the COMPRESS attribute for the table is set to YES and a dictionary exists. This applies to any insert row operation, including an insert through the import or LOAD operations. Compression is enabled for an entire table. However, each row is compressed individually. Therefore, a table could contain both compressed and non-compressed rows at the same time.

Only permanent data objects can be compressed. Data row compression is not applicable to index, long, LOB, and XML data objects.

Row compression is not compatible with table data replication support.

Row compression statistics can be generated using the RUNSTATS command and are stored in the system catalog table SYSCAT.TABLES. A compression estimation option is available with the INSPECT utility. The query optimizer includes decompression cost in its costing model.

SAP NetWeaver BI tables compression

By the time that this book was being written, no full integration of DB2 data row compression was provided in DBACOCKPIT. SAP though, in its SAP-note 980067¹, provides two tools, a script and an ABAP-based tool that can be imported into SAP NetWeaver BI.

We chose to use in our compression test parts of the scripts provided. We used one stored procedure provided to estimate the compression ratio. This stored procedure is depicted in Example 3-28 on page 230 and can be downloaded with the SAP note.

Valid values for stored procedure parameters are:

- ▶ dbpartitionnum:
 - -2 inspects calls in all partitions
 - -1 inspects calls for current partition
 - *n*, where *n* is any integer greater or equal to 0; inspects calls executed in partition *n*
- ▶ table_schema:
 - NULL or empty string, assumes current schema
 - Schema name

¹ More about this SAP note at:

<ftp://ftp.software.ibm.com/software/data/pubs/papers/DB2-SAP-compression.pdf>
http://www.sbm.com.sa/files/IBM_DB2_Optimized_for_SAP.pdf

- ▶ `table_name`:
 - NULL or empty string, assumes all tables for the schema
 - Table name

Note: We strongly recommend that you follow the SAP procedures and instructions.

Example 3-28 SAP stored procedure to estimate data row compression ratio

```
CREATE PROCEDURE SAPR3.INSPECT_TABLE_ROWCOMPESTIMATE(  
    IN DBPARTITIONNUM INTEGER,  
    IN TABLE_SCHEMA VARCHAR(128),  
    IN TABLE_NAME VARCHAR(128)  
)  
    SPECIFIC INSPECT_TABLE_ROWCOMPESTIMATE  
    DYNAMIC RESULT SETS 1  
    MODIFIES SQL DATA  
    NOT DETERMINISTIC  
    CALLED ON NULL INPUT  
    LANGUAGE C  
    EXTERNAL NAME 'db2sap!inspect_table_rowcompeestimate'  
    FENCED NOT THREADSAFE  
    PARAMETER STYLE SQL  
    PROGRAM TYPE SUB  
    DBINFO;
```

We used this stored procedure to check the average compression ratio that we would achieve for some objects. SAP recommends compressing only tables that are not already compressed by SAP. In our test, a non-production environment, we created a design to compress a selection of the following tables:

- ▶ ODS tables
- ▶ PSA tables
- ▶ InfoCubes fact tables

We do not go through the execution for all the tables. We provide an example for the main objects when passing the parameter *dbpartitionnum equal -2*.

Example 3-29 provides an example of the command to check the data row compression ratio for the ODS ZGTFCO01 in all database partitions.

Example 3-29 Calling the stored procedure to inspect ZGTFCO01 compression ratio

```
CALL SAPR3.INSPECT_TABLE_ROWCOMPESTIMATE(-2, 'SAPR3', '/BIC/AZGTFCO0100')
```

The output of the command is shown in Figure 3-63.

| TABLE_SCHEMA | TABLE_NAME | DBPARTITIONNUM | SAVED_PAGES_PCT | SAVED_BYTES_PCT | ROWS_TOO_SMALL_PCT | COMPRESSION_DICT_SIZE | EXPANSION_DICT_SIZE |
|--------------|-----------------|----------------|-----------------|-----------------|--------------------|-----------------------|---------------------|
| SAPR3 | /BIC/AZGTF00100 | 6 | 84 | 84 | 0 | 59136 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 8 | 84 | 84 | 0 | 59648 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 10 | 84 | 84 | 0 | 59904 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 12 | 84 | 84 | 0 | 61440 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 14 | 84 | 84 | 0 | 60928 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 16 | 84 | 84 | 0 | 60416 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 18 | 84 | 84 | 0 | 59648 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 20 | 84 | 84 | 0 | 62080 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 22 | 84 | 84 | 0 | 59008 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 24 | 84 | 84 | 0 | 59008 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 26 | 84 | 84 | 0 | 59520 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 28 | 84 | 84 | 0 | 59264 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 30 | 84 | 84 | 0 | 59904 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 32 | 84 | 84 | 0 | 59648 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 34 | 84 | 84 | 0 | 61568 | 32768 |
| SAPR3 | /BIC/AZGTF00100 | 36 | 84 | 84 | 0 | 60416 | 32768 |

Figure 3-63 Inspect results for compression of ZGTF001

Example 3-30 shows the command to check the compression ratio for a PSA table.

Example 3-30 Calling the stored procedure to inspect a PSA table compression ratio

```
CALL SAPR3.INSPECT_TABLE_ROWCOMPESTIMATE(-2, 'SAPR3', '/BIC/B0002675000')
```

The output of the command is shown in Figure 3-64.

| TABLE_SCHEMA | TABLE_NAME | DBPARTITIONNUM | SAVED_PAGES_PCT | SAVED_BYTES_PCT | ROWS_TOO_SMALL_PCT | COMPRESSION_DICT_SIZE | EXPANSION_DICT_SIZE |
|--------------|------------------|----------------|-----------------|-----------------|--------------------|-----------------------|---------------------|
| SAPR3 | /BIC/B0002675000 | 6 | 82 | 82 | 0 | 52352 | 32768 |
| SAPR3 | /BIC/B0002675000 | 8 | 82 | 82 | 0 | 53248 | 32768 |
| SAPR3 | /BIC/B0002675000 | 10 | 82 | 82 | 0 | 56192 | 32768 |
| SAPR3 | /BIC/B0002675000 | 12 | 82 | 82 | 0 | 54656 | 32768 |
| SAPR3 | /BIC/B0002675000 | 14 | 82 | 82 | 0 | 54144 | 32768 |
| SAPR3 | /BIC/B0002675000 | 16 | 82 | 82 | 0 | 55552 | 32768 |
| SAPR3 | /BIC/B0002675000 | 18 | 82 | 82 | 0 | 54272 | 32768 |
| SAPR3 | /BIC/B0002675000 | 20 | 82 | 82 | 0 | 55936 | 32768 |
| SAPR3 | /BIC/B0002675000 | 22 | 82 | 82 | 0 | 52352 | 32768 |
| SAPR3 | /BIC/B0002675000 | 24 | 82 | 82 | 0 | 53248 | 32768 |
| SAPR3 | /BIC/B0002675000 | 26 | 82 | 82 | 0 | 56192 | 32768 |
| SAPR3 | /BIC/B0002675000 | 28 | 82 | 82 | 0 | 54656 | 32768 |
| SAPR3 | /BIC/B0002675000 | 30 | 82 | 82 | 0 | 53888 | 32768 |
| SAPR3 | /BIC/B0002675000 | 32 | 82 | 82 | 0 | 52864 | 32768 |
| SAPR3 | /BIC/B0002675000 | 34 | 82 | 82 | 0 | 52224 | 32768 |
| SAPR3 | /BIC/B0002675000 | 36 | 82 | 82 | 0 | 54400 | 32768 |

Figure 3-64 Inspect results for compression of PSA table

Example 3-31 shows the command to check the compression ratio for the InfoCube ZGTF002 fact table.

Example 3-31 Calling the stored procedure to inspect a ZGTF002 fact table compression ratio

```
CALL SAPR3.INSPECT_TABLE_ROWCOMPESTIMATE(-2, 'SAPR3', '/BIC/FZGTF002')
```

The output of the command is shown in Figure 3-65.

| TABLE_SCHEMA | TABLE_NAME | DBPARTITIONNUM | SAVED_PAGES_PCT | SAVED_BYTES_PCT | ROWS_TOO_SMALL_PCT | COMPRESSION_DICT_SIZE | EXPANSION_DICT_SIZE |
|--------------|---------------|----------------|-----------------|-----------------|--------------------|-----------------------|---------------------|
| SAPR3 | /BIC/FZGTF002 | 6 | 86 | 86 | 0 | 39424 | 32768 |
| SAPR3 | /BIC/FZGTF002 | 10 | 86 | 86 | 0 | 39296 | 32768 |
| SAPR3 | /BIC/FZGTF002 | 14 | 86 | 86 | 0 | 39040 | 32768 |
| SAPR3 | /BIC/FZGTF002 | 18 | 86 | 86 | 0 | 39168 | 32768 |
| SAPR3 | /BIC/FZGTF002 | 22 | 86 | 86 | 0 | 39680 | 32768 |
| SAPR3 | /BIC/FZGTF002 | 26 | 86 | 86 | 0 | 39040 | 32768 |
| SAPR3 | /BIC/FZGTF002 | 30 | 86 | 86 | 0 | 39608 | 32768 |
| SAPR3 | /BIC/FZGTF002 | 34 | 86 | 86 | 0 | 39552 | 32768 |

Figure 3-65 Inspect results for compression of ZGTF002 fact table

As you may notice, for these three objects the compression ratio, according to the **inspect** command, would be about 82%.

Compression enablement, execution, and results

We created a simple outline to enable the compression for tables in our database. This outline can be split into the following steps:

1. We update the statistics for all tables. Before starting the compression, we executed RUNSTATS in all tables so our measurements would be based on the latest information.
2. We export the SYSCAT.TABLES view to have a snapshot of the tables and the number of pages used without compression. We could have exported the SYSIBM.SYSTABLES table, but since it would refer to the same information, we chose the SYSCAT.TABLES view (because it is more readable in our case).
3. We execute an ALTER TABLE command to alter selected PSA, ODS, and InfoCube fact tables specifying the clause COMPRESS YES. This command executes the table reorganization to create the compression dictionaries, and afterwards we execute RUNSTATS to collect the statistics for the new compressed table. A script was created and used for this task and it is discussed in detail in 3.4.3, “REORG and RUNSTATS comparison with and without compression” on page 235.
4. We export the SYSCAT.TABLES view to have a snapshot of the tables and the number of pages used after the compression.

We focus on pages used for data (because it is not meaningful to check any other objects, as they would be the same at the same size).

Results:

- ▶ Before compression, a total of 711,188,655 pages of 16 KB were being used for the tables.
- ▶ After compressing the tables, it went down to 307,257,099 pages.

Transforming this in bytes, an initial size of 11.11 GB reduced to 4.8 GB — a total reduction of 56.8%. If you transpose this only over the objects we compressed, we achieved a compression ratio of 85.84% mixing PSA, ODS, and InfoCubes fact tables.

About the compression ratio

We wanted to analyze some results in detail:

- ▶ First, we analyzed the ODS table ‘/BIC/AZGTFCO0100’. Since we are using the output from SYSCAT.TABLES we felt that all columns would not fit and would not be necessary, so we truncated and removed some columns, keeping the ones related to the analysis.

Example 3-32 shows all the commands executed over this table. (We specified a different tablespace to execute the table reorganization.)

Example 3-32 Enabling compression for ZGTFCO01 table

```
Fri Mar 2 11:19:34 NPT 2007
ALTER TABLE SAPR3."/BIC/AZGTFCO0100" COMPRESS YES
DB20000I The SQL command completed successfully.
Fri Mar 2 11:19:35 NPT 2007
REORG TABLE SAPR3."/BIC/AZGTFCO0100" USE PSAPTEMP16 ON ALL DBPARTITIONNUMS
DB20000I The REORG command completed successfully.
Fri Mar 2 11:22:00 NPT 2007
RUNSTATS ON TABLE SAPR3."/BIC/AZGTFCO0100" and indexes all
DB20000I The RUNSTATS command completed successfully.
Fri Mar 2 11:23:58 NPT 2007
```

Specifically for this table, and as shown in Figure 3-66, the table was reduced from 4,368,464 pages to 638,000, which is a compression ratio of 85.40%. This is very close to what the table inspection presented, as shown previously, which was 84%.

If you compare the average row size, it was reduced to 84.38%. As shown, to identify whether the table has data row compression enabled, you have to check whether the field compression is set to 'R' (row).

| | TABSHEMA | TABNAME | STATS_TIME | COLCOUNT |
|------------------|----------|------------------|----------------------------|----------|
| Regular table | SAPR3 | /BIC/AZGTFC00100 | 2007-02-27-11.49.50.900715 | 218 |
| Compressed Table | SAPR3 | /BIC/AZGTFC00100 | 2007-03-02-11.23.56.238624 | 218 |

| CARD | NPAGES | FPAGES | OVERFLOW | COMPRESSION | AVGROWSIZE | PCTROWSCOMPRESSED |
|----------|---------|---------|----------|-------------|------------|-------------------|
| 35397216 | 4368464 | 4368480 | 0 | N | 1863 | 0.00 |
| 35397216 | 638000 | 638080 | 0 | R | 291 | 100.00 |

Figure 3-66 Statistics compression comparison for ZGTFC001

- ▶ A second object is a PSA table, '/BIC/B0002675000'. Example 3-33 shows the commands executed over this table.

Example 3-33 Enabling compression for PSA table

```

Fri Mar 2 11:35:58 NPT 2007
ALTER TABLE SAPR3."/BIC/B0002675000" COMPRESS YES
DB20000I The SQL command completed successfully.
Fri Mar 2 11:35:58 NPT 2007
REORG TABLE SAPR3."/BIC/B0002675000" USE PSAPTEMP16 ON ALL DBPARTITIONNUMS
DB20000I The REORG command completed successfully.
Fri Mar 2 11:50:48 NPT 2007
RUNSTATS ON TABLE SAPR3."/BIC/B0002675000" and indexes all
DB20000I The RUNSTATS command completed successfully.
Fri Mar 2 11:59:39 NPT 2007

```

Specifically for this table, and as shown in Figure 3-67, the table was reduced from 14,389,548 pages to 2,409,672, which is a compression ratio of 83.25%. Again, the inspection estimated a compression ratio very close to the real one, which was 82%. If you compare to the average row size, it consequently dropped as well to 82.26%.

| | TABSHEMA | TABNAME | STATS_TIME | COLCOUNT |
|------------------|----------|------------------|----------------------------|----------|
| Regular table | SAPR3 | /BIC/B0002675000 | 2007-02-27-12.25.26.390280 | 220 |
| Compressed Table | SAPR3 | /BIC/B0002675000 | 2007-03-02-11.59.36.832404 | 220 |

| CARD | NPAGES | FPAGES | OVERFLOW | COMPRESSION | AVGROWSIZE | PCTROWSCOMPRESSED |
|----------|----------|----------|----------|-------------|------------|-------------------|
| 1.15E+08 | 14389548 | 14389560 | 0 | N | 1900 | 0.00 |
| 1.15E+08 | 2409672 | 2409732 | 0 | R | 337 | 100.00 |

Figure 3-67 Statistics compression comparison for PSA table

- ▶ A third object is the InfoCube ZGTFC002 fact table. Example 3-34 shows the commands executed over this table.

Example 3-34 Enabling compression for InfoCube ZGTFC002 fact table

```

Mon Mar  5 20:23:11 NFT 2007
ALTER TABLE SAPR3."/BIC/FZGTFC002" COMPRESS YES
DB20000I The SQL command completed successfully.
Mon Mar  5 20:23:11 NFT 2007
REORG TABLE SAPR3."/BIC/FZGTFC002" USE PSAPTEMP16 ON ALL DBPARTITIONNUMS
DB20000I The REORG command completed successfully.
RUNSTATS ON TABLE SAPR3."/BIC/FZGTFC002" and indexes all
DB20000I The RUNSTATS command completed successfully.
Mon Mar  5 20:24:07 NFT 2007

```

Specifically for this table, and as shown in Figure 3-68, the table was reduced from 101,456 pages to 12,728, which is a compression ratio of 87.45%. Again, the inspection estimated a compression ratio very close to the real one, which was 86%. If you compare the average row size, it consequently dropped as well to 87%.

| | TABSCHEMA | TABNAME | STATS_TIME | COLCOUNT |
|------------------|-----------|----------------|----------------------------|----------|
| Regular table | SAPR3 | /BIC/FZGTFC002 | 2007-03-05-20.13.43.596583 | 122 |
| Compressed Table | SAPR3 | /BIC/FZGTFC002 | 2007-03-05-20.24.05.264265 | 122 |

| CARD | NPAGES | FPAGES | OVERFLOW | COMPRESSION | AVGROWSIZE | PCTROWSCOMPRESSED |
|---------|--------|---------|----------|-------------|------------|-------------------|
| 1521544 | 101456 | 1264696 | | O/N | 1038 | 0.00 |
| 1521544 | 12728 | 12760 | | O/R | 135 | 100.00 |

Figure 3-68 Statistics compression comparison for InfoCube ZGTFC002 fact table

Compression time

To provide some details about the overall time to enable the compression, we tightly monitored a specific set of tables. The tables selected had a total of 4.2 TB — 266,698,696 pages of 16 KB spread over 544 tables.

The total duration of the tasks to compress this volume of data is shown in Table 3-17.

Table 3-17 Task duration to compress tables

| Task | Seconds | Hours |
|----------------|---------|-------------|
| Table REORG | 160,639 | 44.62 hours |
| Table RUNSTATS | 18,719 | 5.2 hours |

For the average page per second, we consider the table REORG and the table RUNSTATS times since after a table REORG. To make the optimizer aware of compression, updating statistics is practically obligatory. The average number of pages per second processed in our database enabling compression is shown in Table 3-18, where we consider the start time for the table REORG and the end time for the table RUNSTATS.

Table 3-18 Average pages and MB per second

| 16 KB page per second | MB per second |
|-----------------------|---------------|
| 1,487 | 23.23 |

This may not be the best metric to check the performance, since the number of pages get changed just after the table reorganization. However, this is the best metric to help measure for how much time you would take to switch on compression over an existing table and database. These results are very sensitive and specific to the environment and the data layout.

Important: It is important to note that we did not execute table reorganizations and statistics updates in parallel. Doing that in parallel may improve the required time, but to do so we need to create a SMS temporary tablespace. The one we used was created as DMS.

3.4.3 REORG and RUNSTATS comparison with and without compression

This section highlights some of the benefits and overhead of compression of InfoCubes fact tables reorganizations and statistics update.

Reorg and Runstats test description

We wanted to create some way to measure how maintenance tasks are impacted when using compression. To check that, a small test scenario was designed. We defined a set of tables and the scope that we would check:

- ▶ When executing a table reorganization
- ▶ When collecting table and indexes statistics
- ▶ When collecting table and detailed indexes statistics

A script was created to generate alter table statements, enabling or disabling compression, when executing the table reorganization. The way the script is developed, it is supposed to do all tests, and at the end, leave the table in the same state in which it was found. For example, if you have a table that is already compressed and is listed in the file with table names to be used on the test, it executes the following tasks:

- ▶ Disables compression and checks:
 - Alters table to disable compression
 - Executes table reorganization to move it to a natural un-compressed state
 - Collects table statistics
 - Collects table statistics with detailed index information
- ▶ Regular table checks:
 - Executes table reorganization
 - Collects table statistics
 - Collects table statistics with detailed index information
- ▶ Enables compression checks:
 - Alters table to enable compression back
 - Executes table reorganization to move it to a natural un-compressed state
 - Collects table statistics
 - Collects table statistics with detailed index information

On the other hand, if the table is not compressed yet, it executes the following sequence:

- ▶ Enables compression checks:
 - Alters table to enable compression back
 - Executes table reorganization to move it to a natural un-compressed state
 - Collects table statistics
 - Collects table statistics with detailed index information

- ▶ Disables compression and checks:
 - Alters table to disable compression
 - Executes table reorganization to move it to a natural un-compressed state
 - Collects table statistics
 - Collects table statistics with detailed index information
- ▶ Regular table checks:
 - Executes table reorganization
 - Updates table statistics
 - Updates table statistics with detailed index information

These steps not only helped us to keep the current state, but they left the same state with data reorganization and statistics up to date.

To measure the table reorganization time when the table is compressed, we figure out that the time it takes to execute the table reorganization to create the dictionary is the same as it takes when the table is compressed. We have made a set of tests, and Figure 3-69 shows that the numbers are roughly the same for the table /BIC/FYGTDS_ELE.

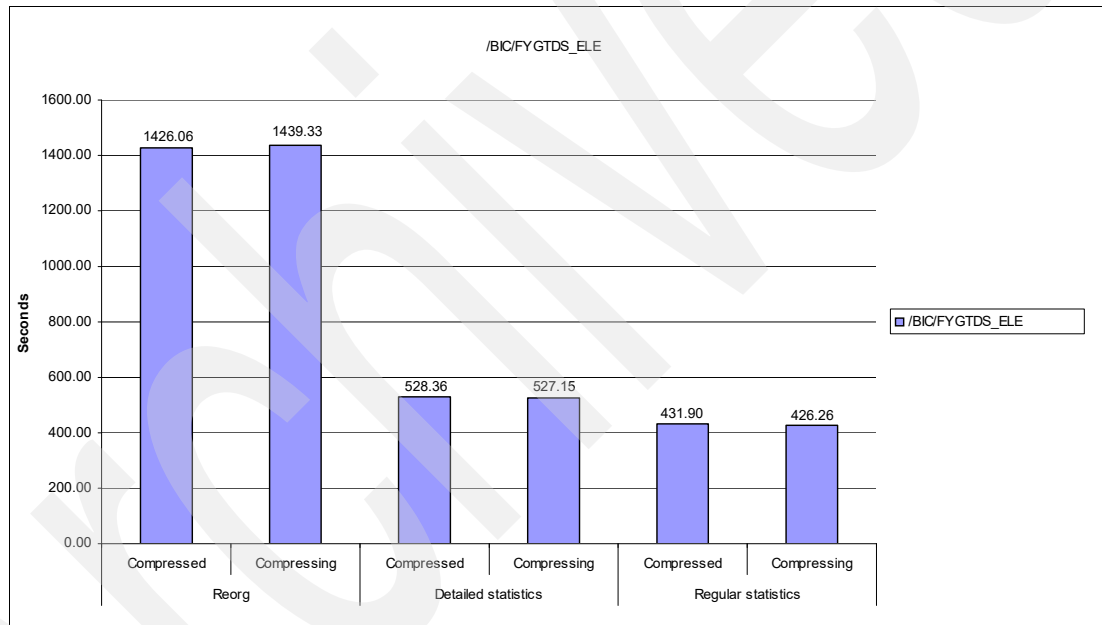


Figure 3-69 Comparison of tasks during compression and after a table is compressed

Based on this assumption, we used the table reorganization time taken at compression enablement as our measurement to eliminate redundancy and improve test time. An important thing to highlight is that the results may vary accordingly to your environment, so if you have an opportunity to do some tests prior moving to a data row compression in a full scale environment, this will help you have some direction as to how it will behave with this feature.

The tests were executed over tables that did and that did not have compression enabled, to make sure that the sequence would not impact our measurements. Also, we chose a small set of large tables since they would be, in most cases, the major concern for these tasks.

Scripts and control tables

We designed a script to help us execute this test. Two control tables were also created, and they are shown in Example 3-35. The first table is used to store the duration of each step and status. The second table has a list of tables that somehow fail during the test execution. We had some cases where tables would not take advantage of compression since data was already compressed, failing when trying to create the compression dictionary.

Example 3-35 create_tables.ddl file to create control tables

```
CREATE TABLE REORG.CONTROL_COMPRESSION (
    TABNAME VARCHAR(128) ,
    COMPRESS INTEGER ,
    NPAGES_UNCOMPRESSED BIGINT ,
    REORG_START TIMESTAMP ,
    REORG_END TIMESTAMP ,
    NPAGES_COMPRESSED BIGINT ,
    RUNSTATS_DETAILED_START TIMESTAMP ,
    RUNSTATS_DETAILED_END TIMESTAMP ,
    RUNSTATS_START TIMESTAMP ,
    RUNSTATS_END TIMESTAMP );

CREATE TABLE REORG.CONTROL_COMPRESSION_FAILED (
    TABNAME VARCHAR(128) ,
    DESCRIPTION VARCHAR(255) );
```

The script was based on a master script and three model files. Each model corresponds to a phase:

- ▶ Enable table compression: model file called `enable_compress_model.ksh`
- ▶ Disable table compression: model file called `disable_compress_model.ksh`
- ▶ Regular table execution: model file called `regular_table_model.ksh`

The master script generates a second script, replacing values from model files for each table used on the test. The master script is depicted in Example 3-36. The file generated, when executed, would then perform all tasks and store control information in control tables.

Example 3-36 Master script generate_compress_script.sh

```
# setting the name of the generated script
SCRIPT_GENERATED_NAME="compress_tables.ksh"
# ensuring that we will not use something from the past
> ${SCRIPT_GENERATED_NAME}
# setting files where we have our tables names
FILENAMES="tables.txt"
# setting files where we have our tables names
DATABASE="EB8"
# setting tabschema
TABSCHEMA="SAPR3"
# control table name
CONTROL_TABLE="REORG.CONTROL_COMPRESSION"
CONTROL_TABLE_FAILED="REORG.CONTROL_COMPRESSION_FAILED"
# use temp space
USE_TEMPSPACE=" use PSAPTEMP16K "
# USE_TEMPSPACE=" "

for file in ${FILENAMES}
do
```

```

NTABLES_TOTAL=$(cat ${file} | wc -l | sed "s/ //g")
let NTABLE=1
echo "echo \"CONNECT TO ${DATABASE}\"" >> ${SCRIPT_GENERATED_NAME}
echo "db2 \"CONNECT TO ${DATABASE}\"" >> ${SCRIPT_GENERATED_NAME}
cat ${file} | while read line
do
    db2 "connect to ${DATABASE}"
    IS_COMPRESSED=$(db2 -x "select count(1) from syscat.tables where tabschema='${TABSHEMA}'
and tabname='${line}' and COMPRESSION='R'")
    IS_COMPRESSED=$(echo $IS_COMPRESSED | sed "s/ //g")
    ## If table is compressed, it will generate the backward sequence, disabling
compression and then, re-enabling it.
    if [ ${IS_COMPRESSED} = "1" ];then
        echo "## Table: ${line}      Table ${NTABLE} of ${NTABLES_TOTAL}" >>
${SCRIPT_GENERATED_NAME}
        echo "echo \"Table: ${line}      Table ${NTABLE} of ${NTABLES_TOTAL}\"" >>
${SCRIPT_GENERATED_NAME}
        TABLE_NAME=$(echo $line | sed "s/\\/\\\\\\\\/g")
        cat disable_compress_model.ksh | sed "s/tabschema/${TABSHEMA}/g" | sed
"s/control_table_failed/${CONTROL_TABLE_FAILED}/g" | sed "s/control_table/${CONTROL_TABLE}/g" |
sed "s/use_temptablespace/${USE_TEMP_TABLESPACE}/g" | sed "s/table/${TABLE_NAME}/g">>
${SCRIPT_GENERATED_NAME}
        cat regular_table_model.ksh | sed "s/tabschema/${TABSHEMA}/g" | sed
"s/control_table_failed/${CONTROL_TABLE_FAILED}/g" | sed "s/control_table/${CONTROL_TABLE}/g" |
sed "s/use_temptablespace/${USE_TEMP_TABLESPACE}/g" | sed "s/table/${TABLE_NAME}/g">>
${SCRIPT_GENERATED_NAME}
        line x
        cat enable_compress_model.ksh | sed "s/tabschema/${TABSHEMA}/g" | sed
"s/control_table_failed/${CONTROL_TABLE_FAILED}/g" | sed "s/control_table/${CONTROL_TABLE}/g" |
sed "s/use_temptablespace/${USE_TEMP_TABLESPACE}/g" | sed "s/table/${TABLE_NAME}/g">>
${SCRIPT_GENERATED_NAME}
        let NTABLE=${NTABLE}+1
    fi
    ## If table is not compressed, it will compress it and then disable compress.
    if [ ${IS_COMPRESSED} = "0" ];then
        echo "## Table: ${line}      Table ${NTABLE} of ${NTABLES_TOTAL}" >>
${SCRIPT_GENERATED_NAME}
        echo "echo \"Table: ${line}      Table ${NTABLE} of ${NTABLES_TOTAL}\"" >>
${SCRIPT_GENERATED_NAME}
        TABLE_NAME=$(echo $line | sed "s/\\/\\\\\\\\/g")
        cat enable_compress_model.ksh | sed "s/tabschema/${TABSHEMA}/g" | sed
"s/control_table_failed/${CONTROL_TABLE_FAILED}/g" | sed "s/control_table/${CONTROL_TABLE}/g" |
sed "s/use_temptablespace/${USE_TEMP_TABLESPACE}/g" | sed "s/table/${TABLE_NAME}/g">>
${SCRIPT_GENERATED_NAME}
        cat disable_compress_model.ksh | sed "s/tabschema/${TABSHEMA}/g" | sed
"s/control_table_failed/${CONTROL_TABLE_FAILED}/g" | sed "s/control_table/${CONTROL_TABLE}/g" |
sed "s/use_temptablespace/${USE_TEMP_TABLESPACE}/g" | sed "s/table/${TABLE_NAME}/g">>
${SCRIPT_GENERATED_NAME}
        cat regular_table_model.ksh | sed "s/tabschema/${TABSHEMA}/g" | sed
"s/control_table_failed/${CONTROL_TABLE_FAILED}/g" | sed "s/control_table/${CONTROL_TABLE}/g" |
sed "s/use_temptablespace/${USE_TEMP_TABLESPACE}/g" | sed "s/table/${TABLE_NAME}/g">>
${SCRIPT_GENERATED_NAME}
        let NTABLE=${NTABLE}+1
    fi
fi

```


The model scripts used in the master script are shown in Example 3-37, where:

1. The `enable_compress_model.ksh` file is used to enable compression for a table.
 - a. Alter the table using the clause `COMPRESS YES` to enable compression.
 - b. Execute the table reorganization to create the dictionary.
 - c. Collect detailed statistics for the table and indexes.
 - d. Collect statistics for the table and indexes.
2. The `regular_table_model.ksh` file is used to execute only table reorg and collect statistics over a regular table.
 - a. Execute the table reorganization to create the dictionary.
 - b. Collect detailed statistics for the table and indexes.
 - c. Collect statistics for the table and indexes.
3. The `disable_compress_model.ksh` file is used to disable table compression for a table.
 - a. Alter the table using the clause `COMPRESS NO` to disable compression.
 - b. Execute the table reorganization to drop the dictionary and uncompress data.
 - c. Collect detailed statistics for the table and indexes.
 - d. Collect statistics for the table and indexes.

Example 3-37 Model files used by the master script

```

1          enable_compress_model.ksh
date
echo "EXPORT TO export.enable.before.${(echo table | sed "s/\\/.g").ixf} OF IXF SELECT * FROM
SYSCAT.TABLES WHERE TABNAME='table'"
db2 "EXPORT TO export.enable.before.${(echo table | sed "s/\\/.g").ixf} OF IXF SELECT * FROM
SYSCAT.TABLES WHERE TABNAME='table'"
date
echo "SELECT * FROM SYSCAT.TABLES WHERE TABNAME='table'"
db2 "SELECT * FROM SYSCAT.TABLES WHERE TABNAME='table'"
date
echo "ALTER TABLE tabschema.\"table\" COMPRESS YES"
1.a.      db2 "ALTER TABLE tabschema.\"table\" COMPRESS YES"
date
db2 "INSERT INTO control_table(TABNAME,COMPRESS,REORG_START,NPAGES_UNCOMPRESSED)
VALUES('table',1,CURRENT_TIMESTAMP,(SELECT NPAGES FROM SYSCAT.TABLES WHERE TABNAME='table'))"
echo "REORG TABLE tabschema.\"table\" use temptablespace ON ALL DBPARTITIONNUMS"
1.b.      REORG_RC=$(db2 "REORG TABLE tabschema.\"table\" use temptablespace ON ALL
DBPARTITIONNUMS")
echo ${REORG_RC}
REORG_OK=$(echo ${REORG_RC} | grep "The REORG command completed successfully" | wc -l | sed "s/
//g" )
if [ "${REORG_OK}" = "1" ];then
  db2 "UPDATE control_table SET REORG_END=CURRENT_TIMESTAMP,RUNSTATS_DETAILED_START=CURRENT
TIMESTAMP WHERE COMPRESS=1 and TABNAME='table'"
  echo "RUNSTATS ON TABLE tabschema.\"table\" with distribution and detailed indexes all"
1.c.      db2 "RUNSTATS ON TABLE tabschema.\"table\" with distribution and detailed indexes
all"
  db2 "UPDATE control_table SET RUNSTATS_DETAILED_END=CURRENT
TIMESTAMP,NPAGES_COMPRESSED=(SELECT NPAGES FROM SYSCAT.TABLES WHERE TABNAME='table') WHERE
COMPRESS=1 AND TABNAME='table'"
date

```

```

db2 "UPDATE control_table SET RUNSTATS_START=CURRENT TIMESTAMP WHERE COMPRESS=1 AND
TABNAME='table'"
echo "RUNSTATS ON TABLE tabschema.\"table\" and indexes all"
1.d. db2 "RUNSTATS ON TABLE tabschema.\"table\" and indexes all"
db2 "UPDATE control_table SET RUNSTATS_END=CURRENT TIMESTAMP WHERE COMPRESS=1 AND
TABNAME='table'"
date
date
echo "EXPORT TO export.enable.after.${(echo table | sed "s/\\/.g").ixf} OF IXF SELECT * FROM
SYSCAT.TABLES WHERE TABNAME='table'"
db2 "EXPORT TO export.enable.after.${(echo table | sed "s/\\/.g").ixf} OF IXF SELECT * FROM
SYSCAT.TABLES WHERE TABNAME='table'"
date
echo "SELECT * FROM SYSCAT.TABLES WHERE TABNAME='table'"
db2 "SELECT * FROM SYSCAT.TABLES WHERE TABNAME='table'"
date

else
db2 "DELETE FROM control_table WHERE TABNAME='table' AND COMPRESS=1" > /dev/null
echo "REORG FAILED!"
db2 "INSERT INTO control_table_failed(TABNAME,DESCRIPTION) VALUES('table','${REORG_RC}')" >
/dev/null
fi
2 regular_table_model.ksh
date
echo "EXPORT TO export.regular.before.${(echo table | sed "s/\\/.g").ixf} OF IXF SELECT * FROM
SYSCAT.TABLES WHERE TABNAME='table'"
db2 "EXPORT TO export.regular.before.${(echo table | sed "s/\\/.g").ixf} OF IXF SELECT * FROM
SYSCAT.TABLES WHERE TABNAME='table'"
date
echo "SELECT * FROM SYSCAT.TABLES WHERE TABNAME='table'"
db2 "SELECT * FROM SYSCAT.TABLES WHERE TABNAME='table'"
date
db2 "INSERT INTO control_table(TABNAME,COMPRESS,REORG_START,NPAGES_UNCOMPRESSED)
VALUES('table',0, CURRENT TIMESTAMP, (SELECT NPAGES FROM SYSCAT.TABLES WHERE TABNAME='table'))"
echo "REORG TABLE tabschema.\"table\" use temptablespace ON ALL DBPARTITIONNUMS"
2.a. REORG_RC=$(db2 "REORG TABLE tabschema.\"table\" use temptablespace ON ALL
DBPARTITIONNUMS")
echo ${REORG_RC}
REORG_OK=$(echo ${REORG_RC} | grep "The REORG command completed successfully" | wc -l | sed "s/
//g" )
if [ "${REORG_OK}" = "1" ];then
db2 "UPDATE control_table SET REORG_END=CURRENT TIMESTAMP,RUNSTATS_DETAILED_START=CURRENT
TIMESTAMP WHERE COMPRESS=0 AND TABNAME='table'"
echo "RUNSTATS ON TABLE tabschema.\"table\" with distribution and detailed indexes all"
2.b. db2 "RUNSTATS ON TABLE tabschema.\"table\" with distribution and detailed indexes
all"
db2 "UPDATE control_table SET RUNSTATS_DETAILED_END=CURRENT
TIMESTAMP,NPAGES_COMPRESSED=(SELECT NPAGES FROM SYSCAT.TABLES WHERE TABNAME='table') WHERE
COMPRESS=0 AND TABNAME='table'"
### regular index
db2 "UPDATE control_table SET RUNSTATS_START=CURRENT TIMESTAMP WHERE COMPRESS=0 AND
TABNAME='table'"
echo "RUNSTATS ON TABLE tabschema.\"table\" and indexes all"
2.c. db2 "RUNSTATS ON TABLE tabschema.\"table\" and indexes all"

```

```

db2 "UPDATE control_table SET RUNSTATS_END=CURRENT TIMESTAMP WHERE COMPRESS=0 AND
TABNAME='table'"
date
echo "EXPORT TO export.regular.after.$(echo table | sed "s/\\./.g").ixf OF IXF SELECT * FROM
SYSCAT.TABLES WHERE TABNAME='table'"
db2 "EXPORT TO export.regular.after.$(echo table | sed "s/\\./.g").ixf OF IXF SELECT * FROM
SYSCAT.TABLES WHERE TABNAME='table'"
date
echo "SELECT * FROM SYSCAT.TABLES WHERE TABNAME='table'"
db2 "SELECT * FROM SYSCAT.TABLES WHERE TABNAME='table'"
date
else
db2 "DELETE FROM control_table WHERE TABNAME='table' AND COMPRESS=0" > /dev/null
echo "REORG FAILED!"
db2 "INSERT INTO control_table_failed(TABNAME,DESCRIPTION) VALUES('table','${REORG_RC}')" >
/dev/null
fi
3 disable_compress_model.ksh
date
echo "EXPORT TO export.disable.before.$(echo table | sed "s/\\./.g").ixf OF IXF SELECT * FROM
SYSCAT.TABLES WHERE TABNAME='table'"
db2 "EXPORT TO export.disable.before.$(echo table | sed "s/\\./.g").ixf OF IXF SELECT * FROM
SYSCAT.TABLES WHERE TABNAME='table'"
date
echo "SELECT * FROM SYSCAT.TABLES WHERE TABNAME='table'"
db2 "SELECT * FROM SYSCAT.TABLES WHERE TABNAME='table'"
date
echo "ALTER TABLE tabschema.\"table\" COMPRESS NO"
3.a. db2 "ALTER TABLE tabschema.\"table\" COMPRESS NO"
date
db2 "INSERT INTO control_table(TABNAME,COMPRESS,REORG_START,NPAGES_COMPRESSED) VALUES('table',2,
CURRENT TIMESTAMP, (SELECT NPAGES FROM SYSCAT.TABLES WHERE TABNAME='table'))"
echo "REORG TABLE tabschema.\"table\" use_temp tablespaces ON ALL DBPARTITIONNUMS"
REORG_RC=$(db2 "REORG TABLE tabschema.\"table\" use_temp tablespaces ON ALL DBPARTITIONNUMS")
echo ${REORG_RC}
REORG_OK=$(echo ${REORG_RC} | grep "The REORG command completed successfully" | wc -l | sed "s/
//g" )
if [ "${REORG_OK}" = "1" ];then
db2 "UPDATE control_table SET REORG_END=CURRENT TIMESTAMP,RUNSTATS_DETAILED_START=CURRENT
TIMESTAMP WHERE COMPRESS=2 and TABNAME='table'"
echo "RUNSTATS ON TABLE tabschema.\"table\" with distribution and detailed indexes all"
3.b. db2 "RUNSTATS ON TABLE tabschema.\"table\" with distribution and detailed indexes
all"
db2 "UPDATE control_table SET RUNSTATS_DETAILED_END=CURRENT
TIMESTAMP,NPAGES_UNCOMPRESSED=(SELECT NPAGES FROM SYSCAT.TABLES WHERE TABNAME='table') WHERE
COMPRESS=2 AND TABNAME='table'"
date
db2 "UPDATE control_table SET RUNSTATS_START=CURRENT TIMESTAMP WHERE COMPRESS=2 AND
TABNAME='table'"
echo "RUNSTATS ON TABLE tabschema.\"table\" and indexes all"
3.c. db2 "RUNSTATS ON TABLE tabschema.\"table\" and indexes all"
db2 "UPDATE control_table SET RUNSTATS_END=CURRENT TIMESTAMP WHERE COMPRESS=2 AND
TABNAME='table'"
date

```

```

    echo "EXPORT TO export.disable.after.$(echo table | sed "s/\\./.g").ixf OF IXF SELECT * FROM
SYSCAT.TABLES WHERE TABNAME='table'"
    db2 "EXPORT TO export.disable.after.$(echo table | sed "s/\\./.g").ixf OF IXF SELECT * FROM
SYSCAT.TABLES WHERE TABNAME='table'"
    date
    echo "SELECT * FROM SYSCAT.TABLES WHERE TABNAME='table'"
    db2 "SELECT * FROM SYSCAT.TABLES WHERE TABNAME='table'"
    date
else
    db2 "DELETE FROM control_table WHERE TABNAME='table' AND COMPRESS=2" > /dev/null
    echo "REORG FAILED!"
    db2 "INSERT INTO control_table_failed(TABNAME,DESCRIPTION) VALUES('table','${REORG_RC}')" >
/dev/null
fi

```

Extracts of the execution for InfoCube ZGTFC098 of the file generated by the master script are shown in Example 3-38. It executes the first phase for a table that was already compressed by setting COMPRESS NO at the very end. The last phase is executed by setting COMPRESS YES for the table. The text in bold italic only replaces the SELECT executed on SYSCAT.TABLES.

Example 3-38 Sample snippet of the execution's output for InfoCube ZGTFC098

```

Thu Mar  8 15:14:39 NFT 2007
EXPORT TO export.before..BIC.FZGTFC098.ixf OF IXF SELECT * FROM SYSCAT.TABLES WHERE
TABNAME='/BIC/FZGTFC098'
SQL3132W The character data in column "STATISTICS_PROFILE" will be truncated
to size "32700".
SQL3104N The Export utility is beginning to export data to file
"export.before..BIC.FZGTFC098.ixf".
SQL3105N The Export utility has finished exporting "1" rows.
Number of rows exported: 1
Thu Mar  8 15:14:39 NFT 2007
SELECT * FROM SYSCAT.TABLES WHERE TABNAME='/BIC/FZGTFC098'
=====etc (lines removed)
ALTER TABLE SAPR3."/BIC/FZGTFC098" COMPRESS NO
DB20000I The SQL command completed successfully.
Thu Mar  8 15:14:39 NFT 2007
DB20000I The SQL command completed successfully.
REORG TABLE SAPR3."/BIC/FZGTFC098" USE PSAPTEMP16 ON ALL DBPARTITIONNUMS
DB20000I The REORG command completed successfully.
DB20000I The SQL command completed successfully.
RUNSTATS ON TABLE SAPR3."/BIC/FZGTFC098" with distribution and detailed indexes all
DB20000I The RUNSTATS command completed successfully.
DB20000I The SQL command completed successfully.
Thu Mar  8 15:29:48 NFT 2007
DB20000I The SQL command completed successfully.
RUNSTATS ON TABLE SAPR3."/BIC/FZGTFC098" and indexes all
DB20000I The RUNSTATS command completed successfully.
DB20000I The SQL command completed successfully.
Thu Mar  8 15:32:31 NFT 2007
EXPORT TO export.after..BIC.FZGTFC098.ixf OF IXF SELECT * FROM SYSCAT.TABLES WHERE
TABNAME='/BIC/FZGTFC098'
SQL3132W The character data in column "STATISTICS_PROFILE" will be truncated
to size "32700".
SQL3104N The Export utility is beginning to export data to file

```

```

“export.after..BIC.FZGTFC098.ixf”.
SQL3105N The Export utility has finished exporting “1” rows.
Number of rows exported: 1
Thu Mar 8 15:32:31 NPT 2007
SELECT * FROM SYSCAT.TABLES WHERE TABNAME='/BIC/FZGTFC098'
=====etc (lines removed)
Thu Mar 8 15:32:31 NPT 2007
EXPORT TO export.before..BIC.FZGTFC098.ixf OF IXF SELECT * FROM SYSCAT.TABLES WHERE
TABNAME='/BIC/FZGTFC098'
SQL3132W The character data in column “STATISTICS_PROFILE” will be truncated
to size “32700”.
SQL3104N The Export utility is beginning to export data to file
“export.before..BIC.FZGTFC098.ixf”.
SQL3105N The Export utility has finished exporting “1” rows.
Number of rows exported: 1
Thu Mar 8 15:32:32 NPT 2007
SELECT * FROM SYSCAT.TABLES WHERE TABNAME='/BIC/FZGTFC098'
=====etc (lines removed)
Thu Mar 8 15:32:32 NPT 2007
DB20000I The SQL command completed successfully.
REORG TABLE SAPR3.”/BIC/FZGTFC098” USE PSAPTEMP16 ON ALL DBPARTITIONNUMS
DB20000I The REORG command completed successfully.
DB20000I The SQL command completed successfully.
RUNSTATS ON TABLE SAPR3.”/BIC/FZGTFC098” with distribution and detailed indexes all
DB20000I The RUNSTATS command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
RUNSTATS ON TABLE SAPR3.”/BIC/FZGTFC098” and indexes all
DB20000I The RUNSTATS command completed successfully.
DB20000I The SQL command completed successfully.
Thu Mar 8 15:51:18 NPT 2007
EXPORT TO export.after..BIC.FZGTFC098.ixf OF IXF SELECT * FROM SYSCAT.TABLES WHERE
TABNAME='/BIC/FZGTFC098'
SQL3132W The character data in column “STATISTICS_PROFILE” will be truncated
to size “32700”.
SQL3104N The Export utility is beginning to export data to file
“export.after.BIC.FZGTFC098.ixf”.
SQL3105N The Export utility has finished exporting “1” rows.
Number of rows exported: 1
Thu Mar 8 15:51:18 NPT 2007
SELECT * FROM SYSCAT.TABLES WHERE TABNAME='/BIC/FZGTFC098'
=====etc (lines removed)
Thu Mar 8 15:51:18 NPT 2007
EXPORT TO export.before.BIC.FZGTFC098.ixf OF IXF SELECT * FROM SYSCAT.TABLES WHERE
TABNAME='/BIC/FZGTFC098'
SQL3132W The character data in column “STATISTICS_PROFILE” will be truncated
to size “32700”.
SQL3104N The Export utility is beginning to export data to file
“export.before.BIC.FZGTFC098.ixf”.
SQL3105N The Export utility has finished exporting “1” rows.
Number of rows exported: 1
Thu Mar 8 15:51:18 NPT 2007
SELECT * FROM SYSCAT.TABLES WHERE TABNAME='/BIC/FZGTFC098'
=====etc (lines removed)
Thu Mar 8 15:51:18 NPT 2007

```

```

ALTER TABLE SAPR3."/BIC/FZGTFC098" COMPRESS YES
DB20000I The SQL command completed successfully.
Thu Mar 8 15:51:18 NPT 2007
DB20000I The SQL command completed successfully.
REORG TABLE SAPR3."/BIC/FZGTFC098" USE PSAPTEMP16 ON ALL DBPARTITIONNUMS
DB20000I The REORG command completed successfully.
DB20000I The SQL command completed successfully.
RUNSTATS ON TABLE SAPR3."/BIC/FZGTFC098" with distribution and detailed indexes all
DB20000I The RUNSTATS command completed successfully.
DB20000I The SQL command completed successfully.
Thu Mar 8 16:02:33 NPT 2007
DB20000I The SQL command completed successfully.
RUNSTATS ON TABLE SAPR3."/BIC/FZGTFC098" and indexes all
DB20000I The RUNSTATS command completed successfully.
DB20000I The SQL command completed successfully.
Thu Mar 8 16:05:23 NPT 2007
Thu Mar 8 16:05:23 NPT 2007
EXPORT TO export.after.BIC.FZGTFC098.ixf OF IXF SELECT * FROM SYSCAT.TABLES WHERE
TABNAME='/BIC/FZGTFC098'
SQL3132W The character data in column "STATISTICS_PROFILE" will be truncated
to size "32700".
SQL3104N The Export utility is beginning to export data to file
"export.after.BIC.FZGTFC098.ixf".
SQL3105N The Export utility has finished exporting "1" rows.
Number of rows exported: 1
Thu Mar 8 16:05:23 NPT 2007
SELECT * FROM SYSCAT.TABLES WHERE TABNAME='/BIC/FZGTFC098'

```

Table reorganization and statistics update tests results

We did not create a clear pattern for the behavior of these tasks. They are definitely dependent on each table layout, size, and infrastructure. We got overall results summarizing all of the tables for all of these tasks. Collecting statistics execution time though is so similar to the table without compression that we assume that it did not have either negative or positive effects in our environment.

Figure 3-70 shows a table reorganization example. After compressing the table, we reduced the time from 5,040 seconds to 3,291, which is a reduction of almost 35%.

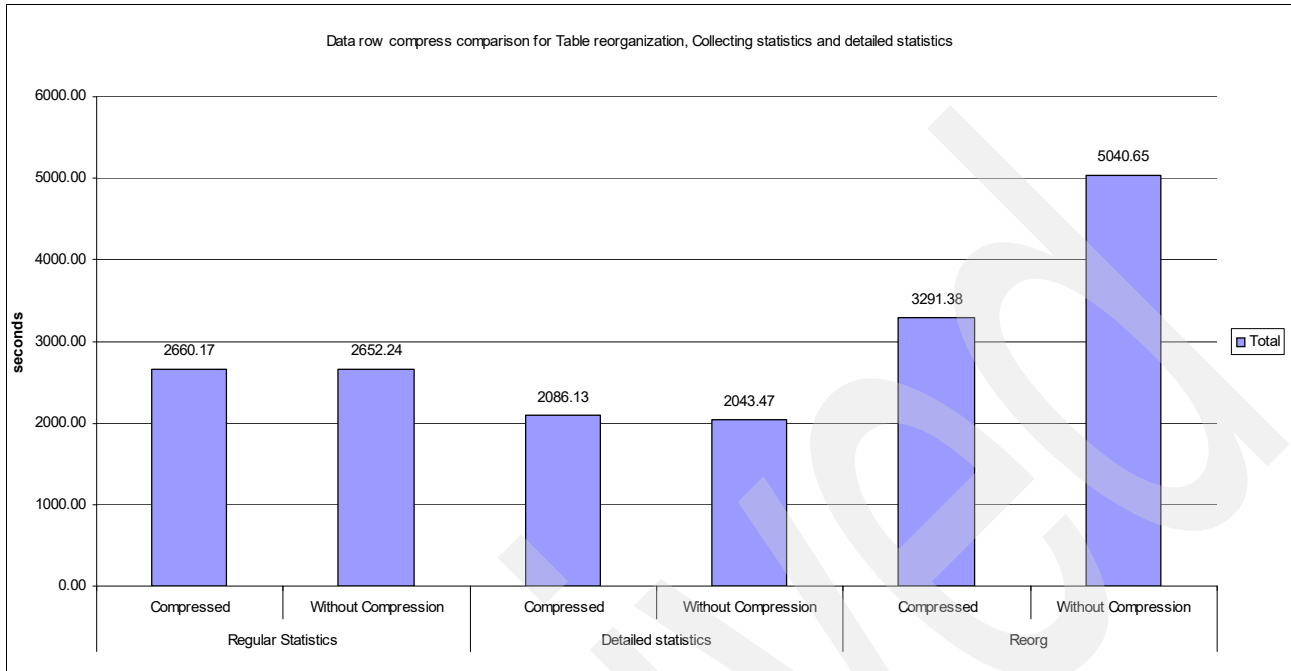


Figure 3-70 Data row compress comparison for table reorganization, collecting statistics, and detailed statistics

This behavior is not seen for all the tables. It may vary even on tables for the same database that differ in size or compression ratio that you achieved. Analyzing only InfoCubes fact tables in Figure 3-71, when you drill down on distinct tables, collecting statistics is roughly the same for both compression and without compression. Although having this pattern for statistics, this cannot be seen for tables reorganization, where, in most of them, the duration was reduced and improved significantly.

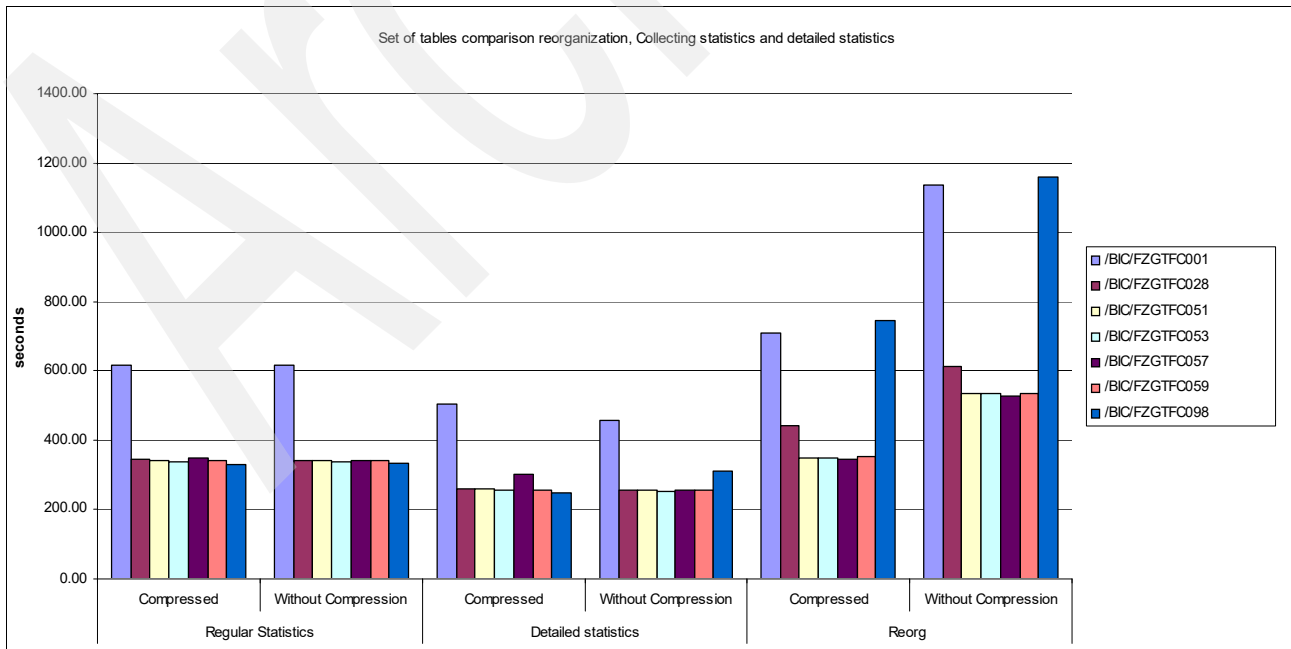


Figure 3-71 InfoCubes fact tables behavior for table reorganization and statistics update

In Figure 3-72, we show tables used for ODS, InfoCubes fact tables, and so on. The same behavior can be noticed where we got a better result for table reorganizations without considerably impacting statistics update.

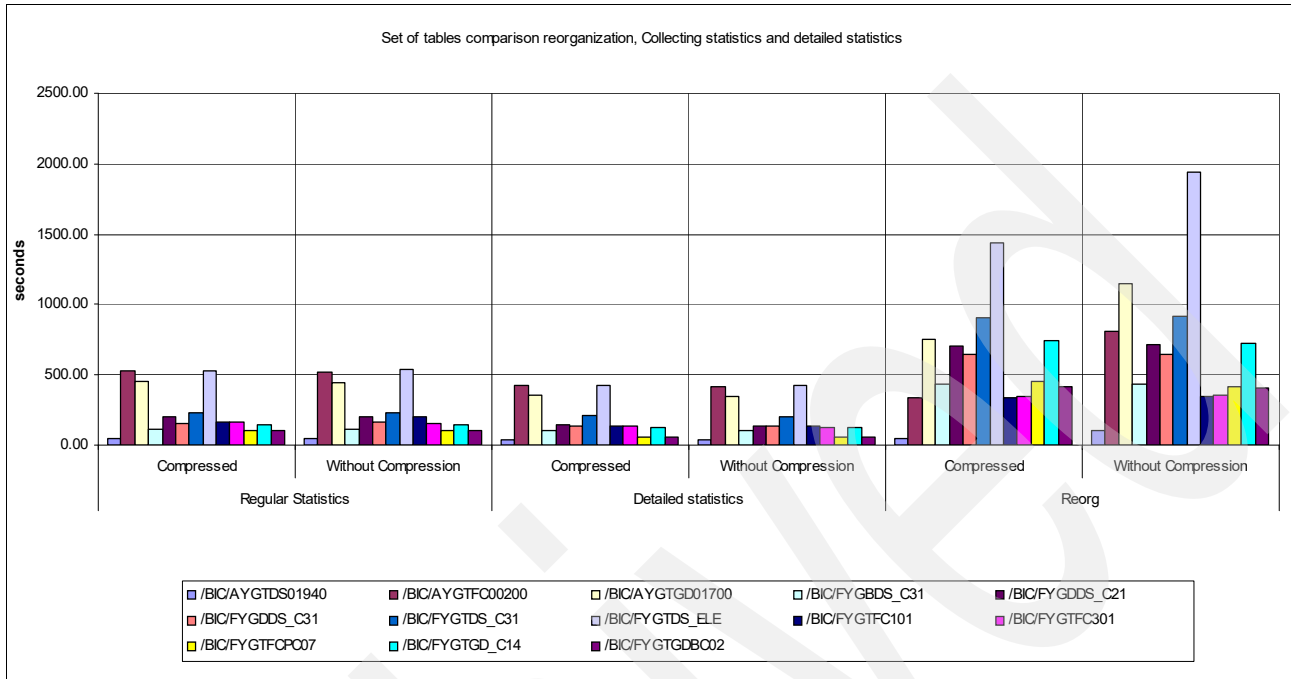


Figure 3-72 Aleatory table behavior for table reorganization and statistics update

Moving back: uncompress tables

In our environment shown in Figure 3-73, we notice that it takes less time to move back than to reorganize over a flat regular table, but more time than for the table with compression. This behavior is observed for all of the remaining objects we used in our test.

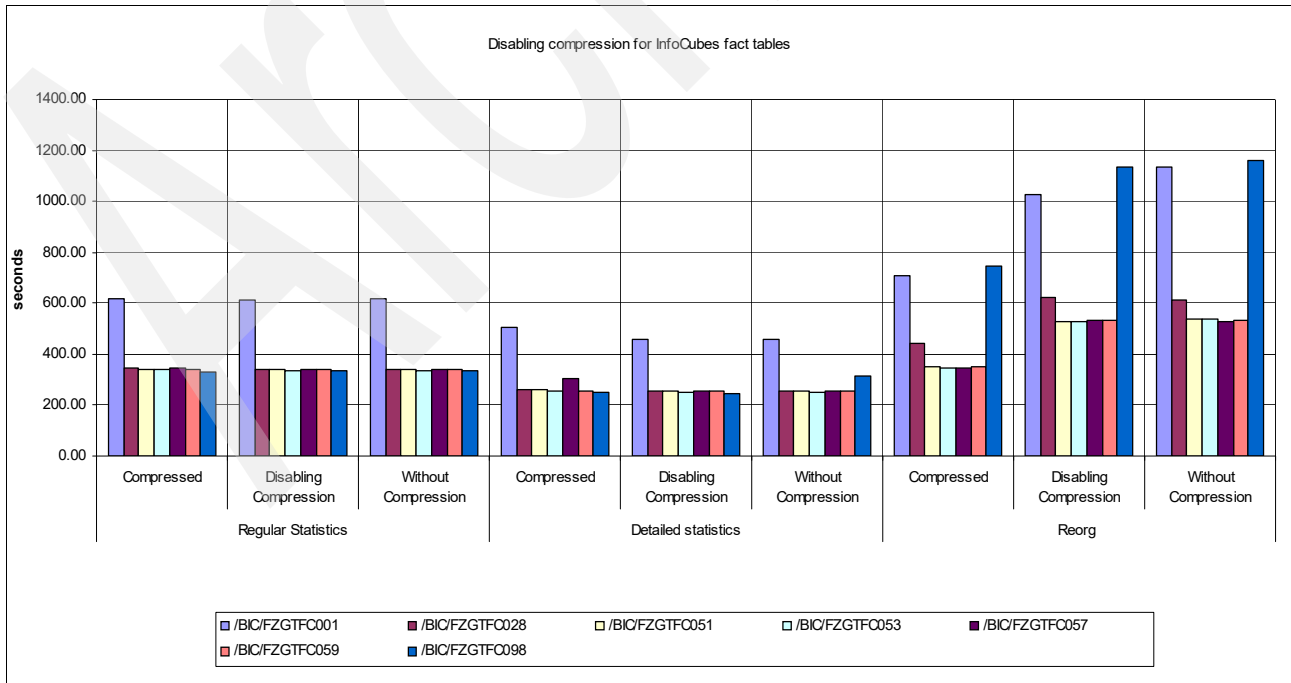


Figure 3-73 Disabling compression behavior over distinct tables

Conclusions and thoughts

There are several improvements when using data row compression:

- ▶ It considerably improves the data buffer pool hit ratio because more data fits on the same amount of memory pool.
- ▶ It reduces the I/O usage because you can get more data from memory and because data rows are smaller.
- ▶ It can considerably reduce storage costs and maintenance tasks.

There are some things to consider if you have a system that is CPU bound. For example, because it relies on CPU cycles to uncompress and compress data on the fly, data row compression may behave differently in this case.

One thing that you should consider and that you may have already noticed is that the compression dictionary created with the table reorganization is based on the data available on the table at that specific time. What happens if you create an empty table and that creates a compression dictionary? Nothing. The compression dictionary will be empty as well. When you have a table and you enable the compression, you will have a compression over your existing data. If you start adding different sets of data, the data row compression ratio for the new data will not be the same as the old one, and you will only achieve the highest compression ratio when another table reorganization invoking a compression dictionary reset is executed. This may require you to execute more table reorganizations more frequently to take full advantage of compression.

There is also some overhead for inserts and updates because it requires extra CPU cycles to compress the data while it is inserting or updating data. You may consider extra CPU cycles if this feature is a good option for your design and architecture.

3.4.4 Stress test comparison with and without compression

In this section we describe some stress tests that we have done using different features. All these tests were executed on SYS2 due to test and time requirements. Since this test was executed on SYS2, the following DB2 AIX LPARs used were:

- ▶ sys2db0: DB2 partition 0
- ▶ sys2db1: DB2 partitions 6 to 13
- ▶ sys2db2: DB2 partitions 14 to 21
- ▶ sys2db3: DB2 partitions 22 to 29
- ▶ sys2db4: DB2 partitions 30 to 37

The goal of the tests was to test how the DB2 9 new feature of data row compression would improve or degrade the response time. The tests were performed by running three different loads:

- ▶ Upload of one InfoCube
- ▶ Aggregate in parallel 8 distinct InfoCubes
- ▶ Online queries

This test is very similar to the load we used on KPI-G, described in 3.4.1, “Stress test results - KPI-G” on page 207. Only the volume and the parallelization differ between these tests.

Before compressing objects as described in 3.4.2, “Tests with DB2 data row compression” on page 228, we executed a baseline stress test to be able to compare it afterwards with the same test executed with data row compression.

Stress test results with regular tables

The main figure of the overall stress test on the system using regular tables is shown in Figure 3-74. You see that we are only considering the high load phase, which means that the ramp-ups and ramp-downs for all three tasks are being removed from the result window. The measurements of this scenario are shown in Table 3-19.

Table 3-19 Stress test results with regular tables

| Metric | Value |
|---------------------------------------|------------------------------|
| Average load per hour | 5.91 million records |
| Average aggregation per hour | 44.5 million records |
| Average query transactions per second | 1.39 transactions per second |
| Average query response time | 10.58 seconds |

Notice on the chart that the query response time has very inconstant behavior.

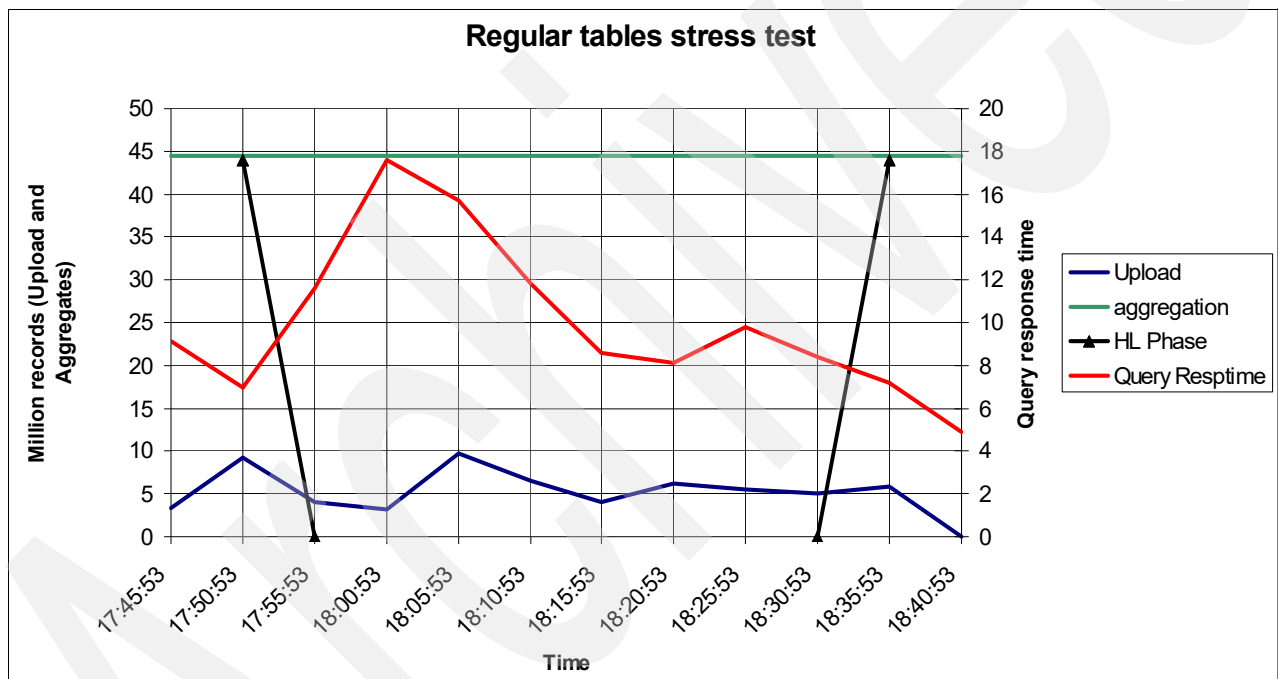


Figure 3-74 Regular tables stress test chart

The queries' inconstant behavior is caused by the same problem we had on the test described in 3.4.1, "Stress test results - KPI-G" on page 207, where the data buffer pool hit ratio was low. Since we only compressed ODS, PSA, and InfoCubes fact tables, the only buffer pool we analyze is BP_STD_16K, which is used for these objects. Also, we focus on DB2 AIX LPARs where these objects were spread over from sys2db1 to sys2db4.

The data buffer pool hit ratio for BP_STD_16K for DB2 partitions 6 to 37 is described in Figure 3-75.

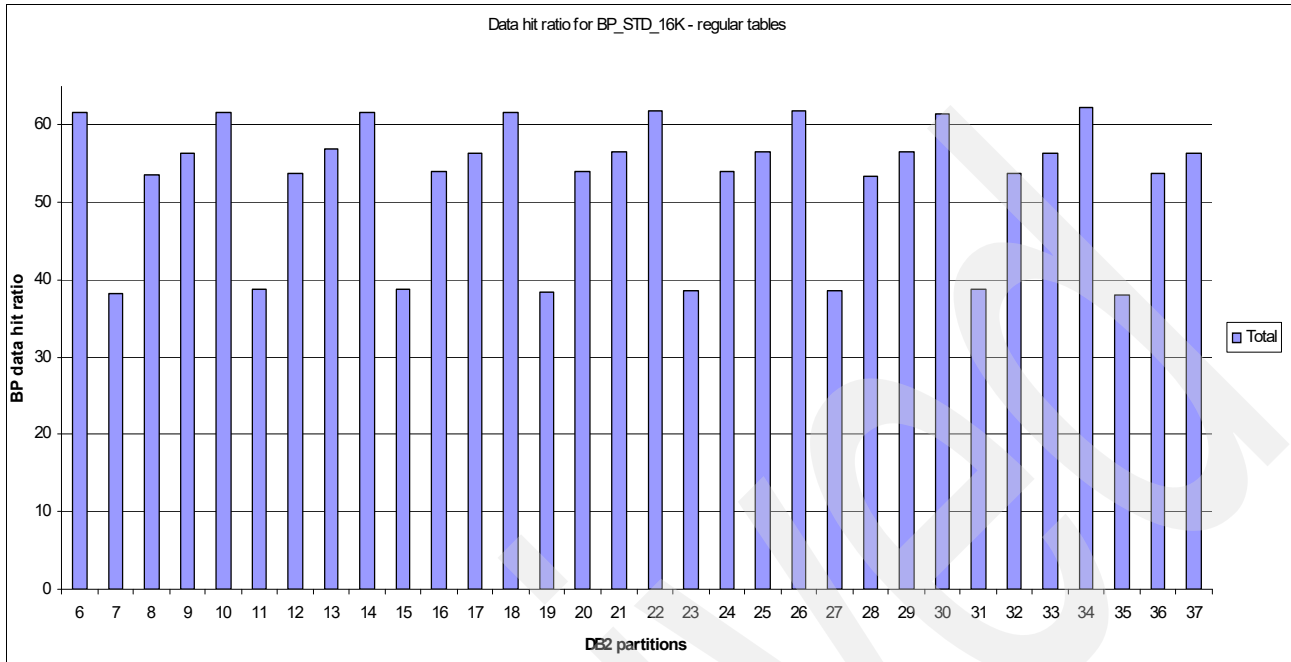


Figure 3-75 BP_STD_16K data buffer pool hit ratio for DB2 partitions 6 to 37

As seen in Figure 3-75, the data buffer pool hit ratio is poor. Each DB2 partition pattern follows the same behavior. For example, if you analyze DB2 partitions 6, 10, 14, 18, 22, 26, 30, and 34 you will see that the values are the same. This is understandable because the objects that are spread over this set of DB2 partitions and the buffer pool sizes are the same.

The major impact of this poor buffer pool hit ratio would be over online queries and InfoCubes aggregation. In Figure 3-76, we demonstrate the buffer pool data hit ratio behavior for DB2 partitions 6 to 9, as discussed previously, and to facilitate our analysis over the graphic, you can assume the same trend line between following DB2 partitions:

- ▶ DB2 partition 6 same as partitions 10, 14, 18, 22, 26, 30, 34
- ▶ DB2 partition 7 same as partitions 11, 15, 19, 23, 27, 31, 35
- ▶ DB2 partition 8 same as partitions 12, 16, 20, 24, 28, 32, 36
- ▶ DB2 partition 9 same as partitions 13, 17, 21, 25, 29, 33, 37

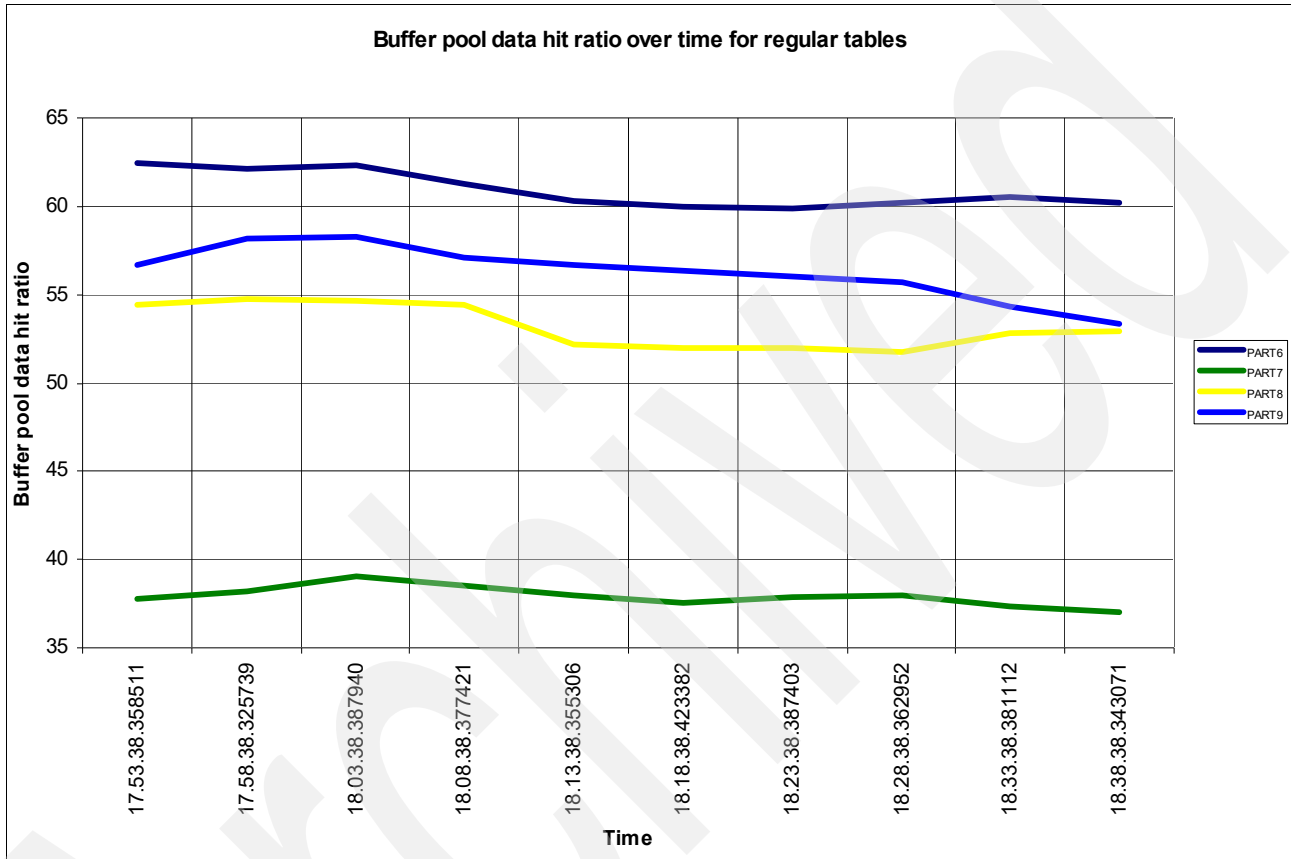


Figure 3-76 Buffer pool data hit ratio and DB2 partition group patterns trend line for high load phase

Table 3-20 summarizes the most actively read tables separated in two categories: tables used for queries and tables used in the aggregation of infocubes.

Table 3-20 Major tables from read rows perspective for queries and aggregates

| InfoCube | Table | Queries | Aggregates |
|----------|----------------|---------|------------|
| ZGTFC013 | /BIC/FZGTFC013 | Yes | No |
| ZGTFC015 | /BIC/FZGTFC015 | Yes | No |
| ZGTFC019 | /BIC/FZGTFC019 | Yes | No |
| ZGTFC021 | /BIC/FZGTFC021 | Yes | No |
| ZGTFC023 | /BIC/FZGTFC023 | Yes | No |
| ZGTFC025 | /BIC/FZGTFC025 | Yes | No |
| ZGTFC040 | /BIC/FZGTFC040 | Yes | No |

| InfoCube | Table | Queries | Aggregates |
|----------|----------------|---------|------------|
| ZGTFC042 | /BIC/FZGTFC042 | Yes | No |
| ZGTFC044 | /BIC/FZGTFC044 | Yes | No |
| ZGTFC046 | /BIC/FZGTFC046 | Yes | No |
| ZGTFC048 | /BIC/FZGTFC048 | Yes | No |
| ZGTFC050 | /BIC/FZGTFC050 | Yes | No |
| ZGTFC063 | /BIC/FZGTFC063 | Yes | No |
| ZGTFC065 | /BIC/FZGTFC065 | Yes | No |
| ZGTFC067 | /BIC/FZGTFC067 | Yes | No |
| ZGTFC069 | /BIC/FZGTFC069 | Yes | No |
| ZGTFC071 | /BIC/FZGTFC071 | Yes | No |
| ZGTFC073 | /BIC/FZGTFC073 | Yes | No |
| ZGTFC075 | /BIC/FZGTFC075 | Yes | No |
| ZGTFC090 | /BIC/FZGTFC090 | Yes | No |
| ZGTFC092 | /BIC/FZGTFC092 | Yes | No |
| ZGTFC094 | /BIC/FZGTFC094 | Yes | No |
| ZGTFC096 | /BIC/FZGTFC096 | Yes | No |
| ZGTFC098 | /BIC/FZGTFC098 | Yes | No |
| ZGTFC030 | /BIC/FZGTFC030 | No | Yes |
| ZGTFC032 | /BIC/FZGTFC032 | No | Yes |
| ZGTFC034 | /BIC/FZGTFC034 | No | Yes |
| ZGTFC038 | /BIC/FZGTFC038 | No | Yes |
| ZGTFC060 | /BIC/FZGTFC060 | No | Yes |
| ZGTFC062 | /BIC/FZGTFC062 | No | Yes |
| ZGTFC076 | /BIC/FZGTFC076 | No | Yes |
| ZGTFC078 | /BIC/FZGTFC078 | No | Yes |

There are a total of 24 major tables for queries and eight major tables for InfoCubes aggregates.

The volume of data read only for queries during the high load phase is shown in Figure 3-77.

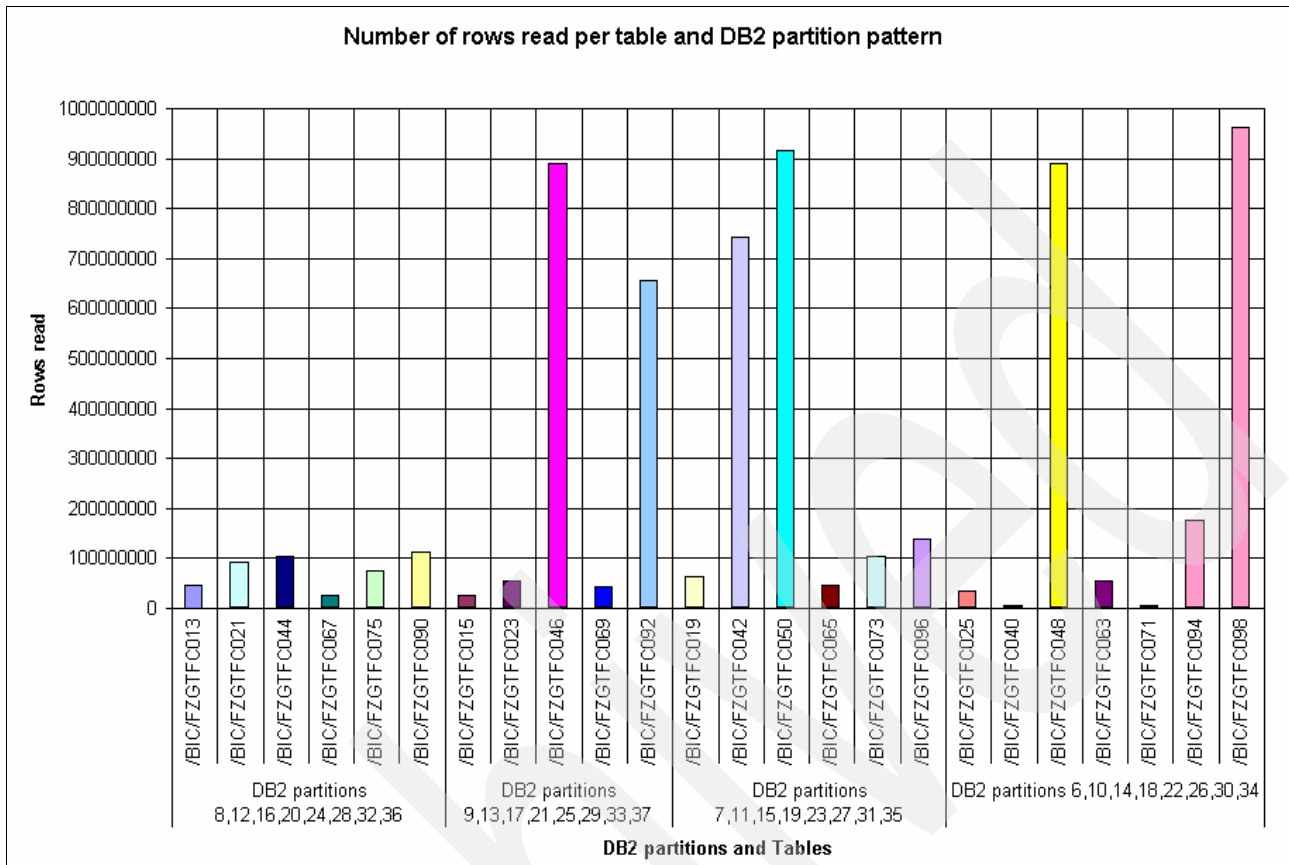


Figure 3-77 Number of rows read by queries per table

The volume of read only rows for aggregation of Infocubes during the high load phase is shown in Figure 3-78. You can see that the volume of rows read by the InfoCube aggregation process is much lower than the volume read in queries. Also, we could have used a better balance on the scenario because we could have reached tables over DB2 partitions 8, 12, 16, 20, 24, 28, 32, and 36.

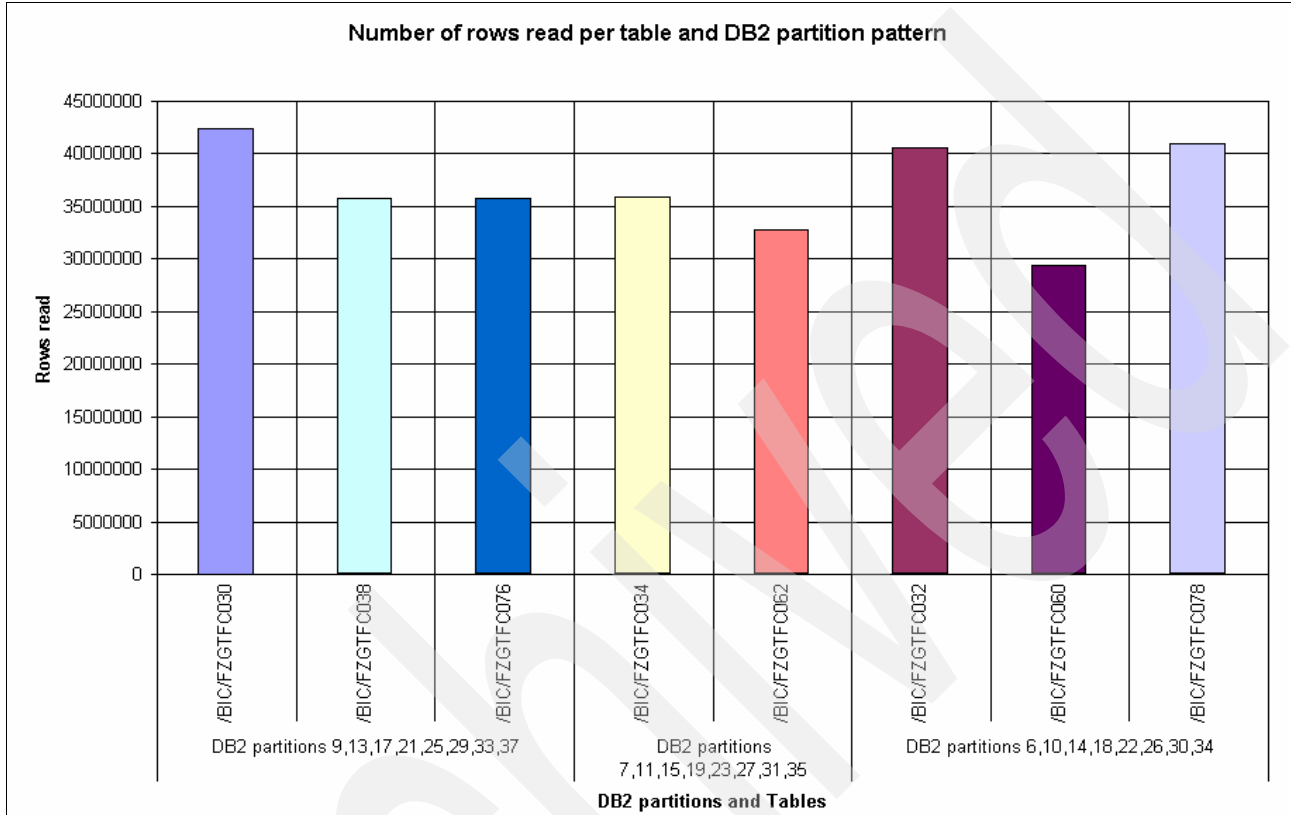


Figure 3-78 Number of rows read by InfoCubes aggregation per table

In Figure 3-79, analyzing the average disk read, we can see that read usage for DB2 partition group patterns that start on partition 6 and partition 7 are much more used than the remaining partition groups. This could have been changed, but since our target is to check compression benefits or overloads, as long we use the same test procedure and use this test as a baseline, our measurements will be transparent.

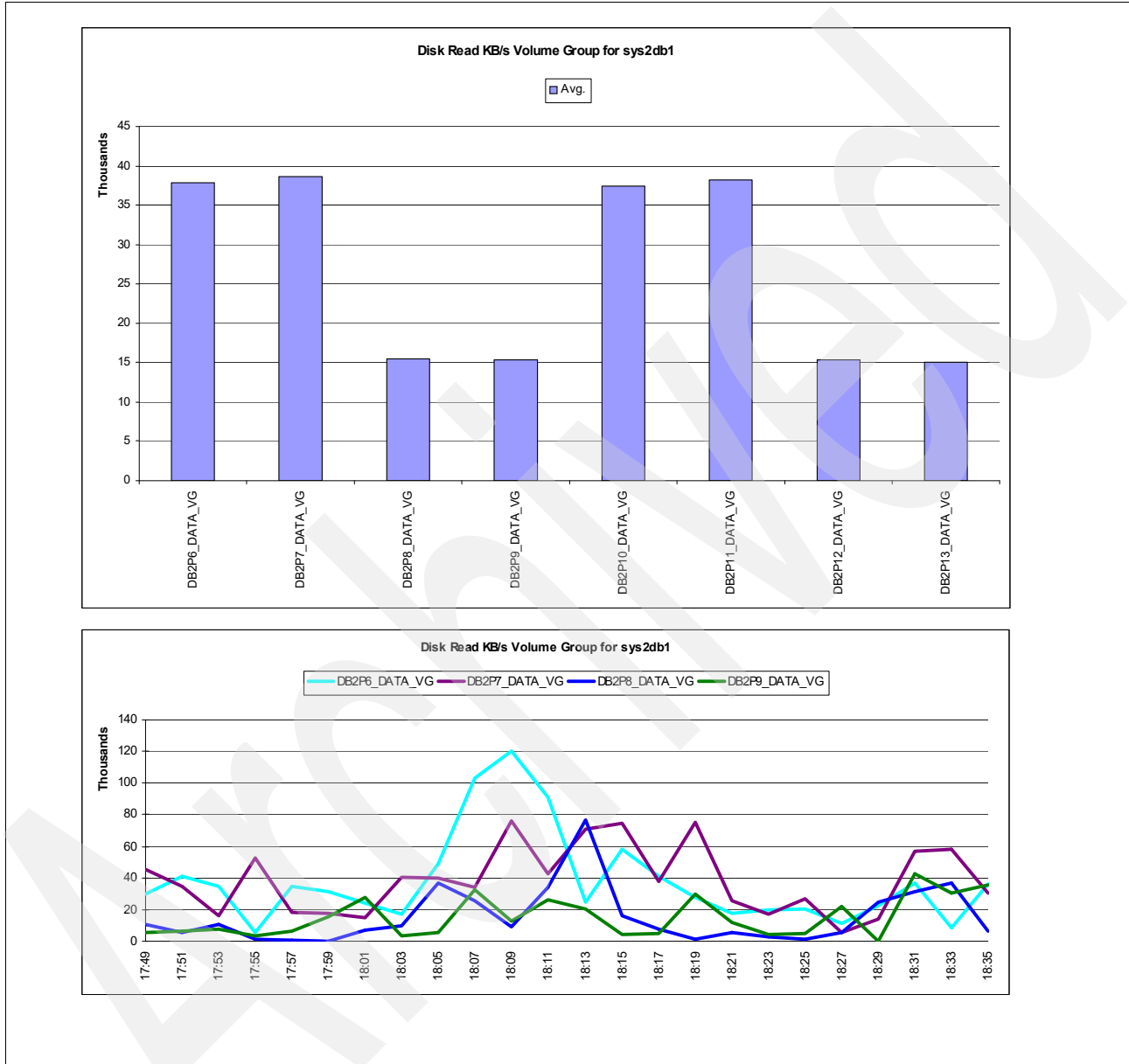


Figure 3-79 I/O usage for sys2db1

Stress test results with compressed tables

After compressing the tables for SAP NetWeaver BI objects like PSA, ODS, and InfoCubes fact tables, we executed another test following the same procedure depicted in “Stress test results with regular tables” on page 248.

The main results of this test using data row compression are shown in Figure 3-80. The measurements are described in Table 3-21.

Table 3-21 Stress test results with data row compression

| Metric | Value |
|---------------------------------------|------------------------------|
| Average load per hour | 5.65 million records |
| Average aggregation per hour | 46.25 million records |
| Average query transactions per second | 1.45 transactions per second |
| Average query response time | 7.12 seconds |

You see on the chart in Figure 3-80 that the query response time has a smoother behavior. We configured the tool that was used to automate query workload with a maximum of 1.45 transactions per second, which was exactly the same value that we achieved. This demonstrates that, most likely, without setting this maximum value, we could have had a better result.

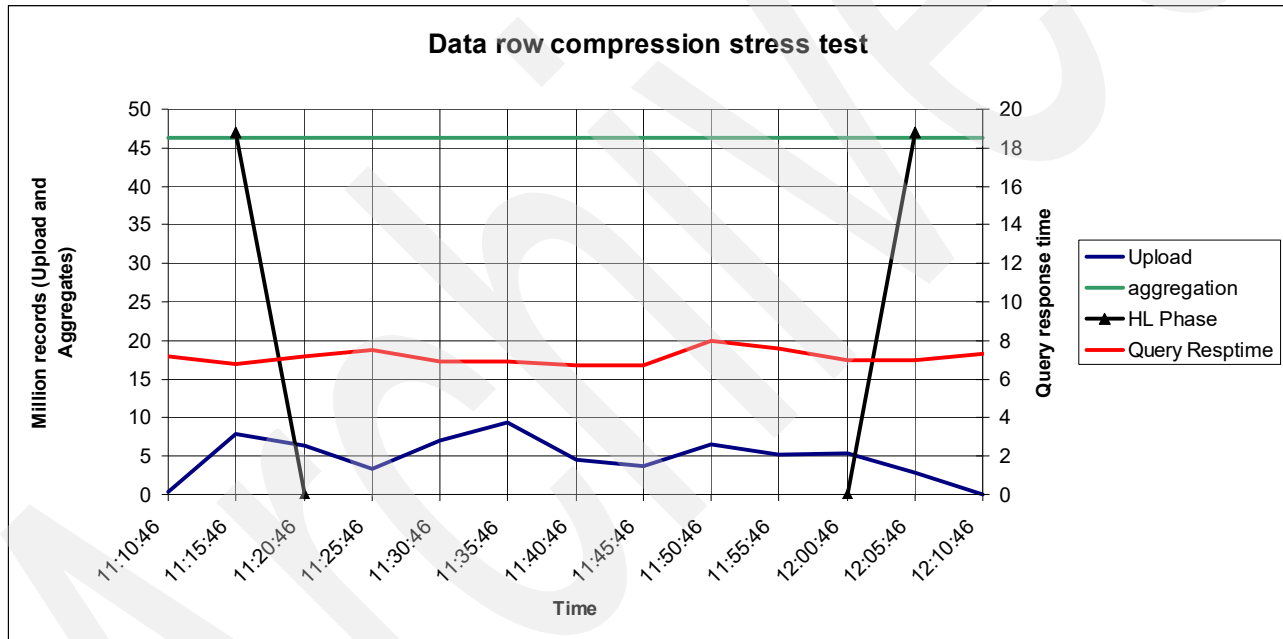


Figure 3-80 Data row compression stress test chart

The data buffer pool hit ratio for BP_STD_16K for DB2 partitions 6 to 37 is described in Figure 3-81. As seen in the figure, the data buffer pool hit ratio was better. Each DB2 partition pattern follows the same behavior. For example, if you analyze DB2 partitions 6, 10, 14, 18, 22, 26, 30, and 34 you will see that the values are the same. This is understandable because the objects that are spread over this set of DB2 partitions and the buffer pool size are the same.

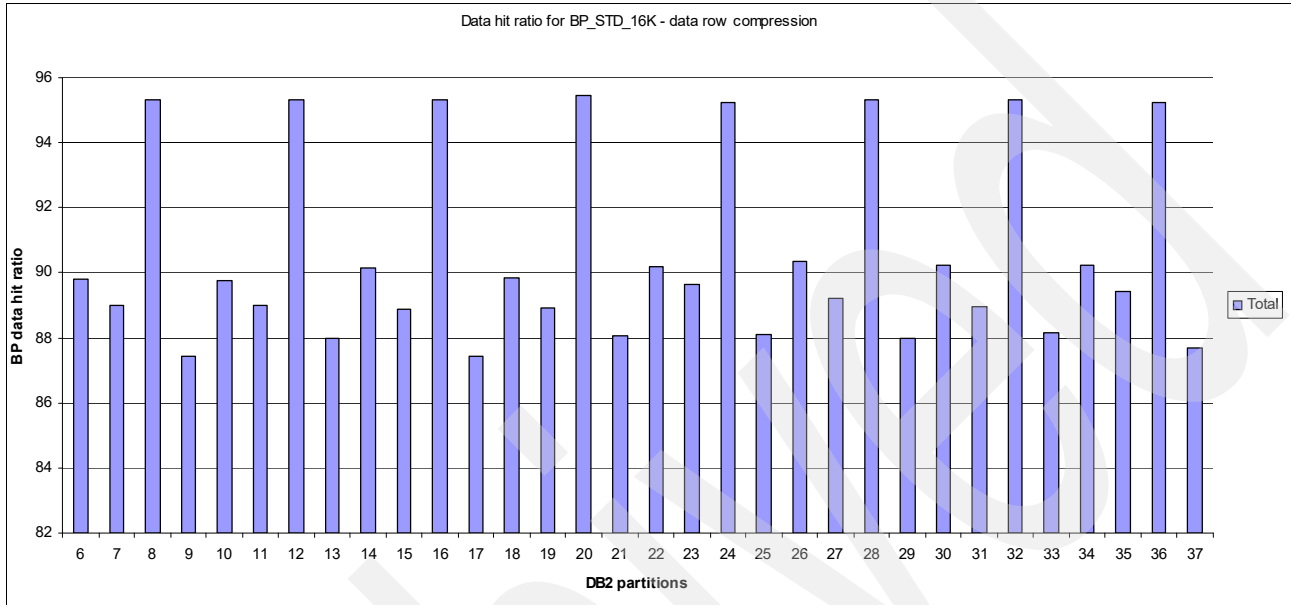


Figure 3-81 BP_STD_16K data buffer pool hit ratio for DB2 partitions 6 to 37

As a result, we had a much data buffer pool hit ratio, as shown on a time scale in Figure 3-82.

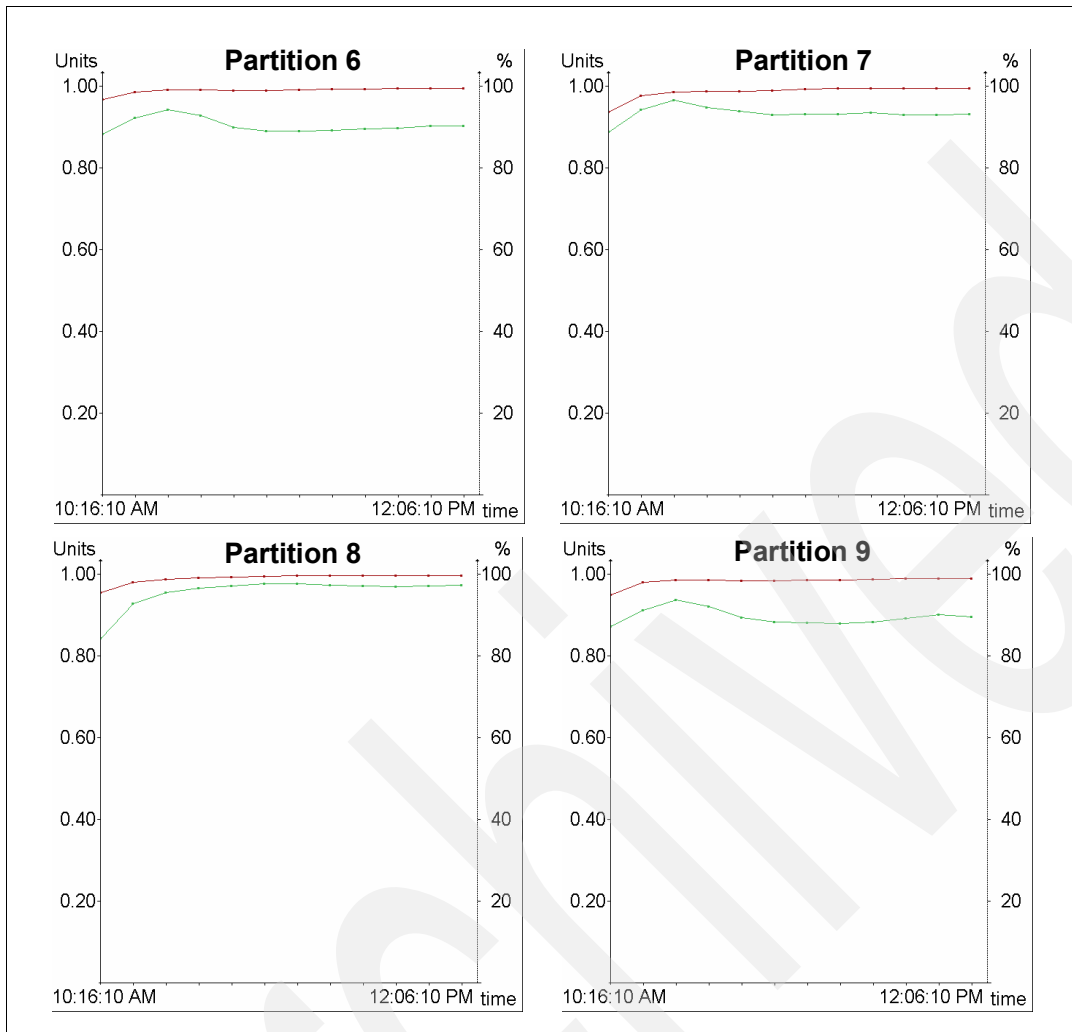


Figure 3-82 Buffer pool data hit ratio trend line for high load phase

Stress test compression thoughts and conclusions

As shown in Figure 3-83, comparing both results shows a massive improvement for queries. The main reason is that this is the most massive read-intensive task in our test scenario.

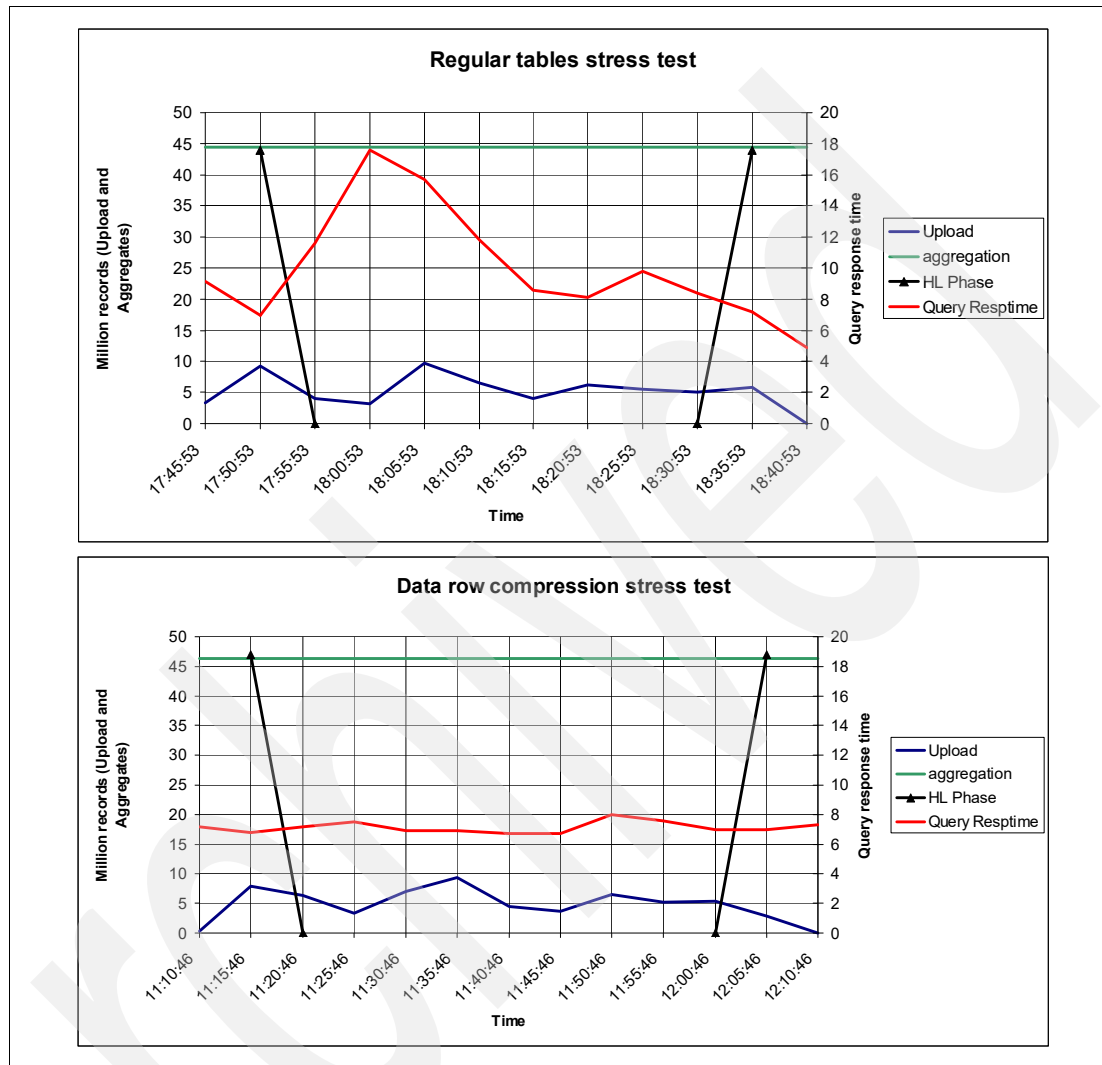


Figure 3-83 Test results comparison

As shown in Table 3-22, the response time behavior after data row compression is much better due to the new higher amount of data that fits in buffer pools in a compressed state. We had a better result in the InfoCubes aggregation workload simulation as well. This is because it also relies on the summarization of InfoCube fact table data directly inserted into InfoCubes aggregates tables. For upload, which is more write intensive task, we faced a slight degradation, which may be caused by the compression overhead in terms of CPU cycles. The test results are detailed in Table 3-22.

Table 3-22 Data row compression compared to regular tables results

| Metric | Regular table | Data row compression | Result |
|---------------------------------------|------------------------------|------------------------------|-----------------------|
| Average load per hour | 5.91 million records | 5.65 million records | Overhead of 4.4% |
| Average aggregation per hour | 44.5 million records | 46.25 million records | Improvement of 3.93% |
| Average query transactions per second | 1.39 transactions per second | 1.45 transactions per second | Improvement of 4.32% |
| Average query response time | 10.58 seconds | 7.12 seconds | Improvement of 32.70% |

It is important to analyze the impact from the server perspective since the idea of data row compression is to decrease I/O usage, but would also bring higher CPU usage.

Let us start comparing disk read usage for both tests, as shown in Figure 3-84, where we observe that the volume of disk read is reduced dramatically.

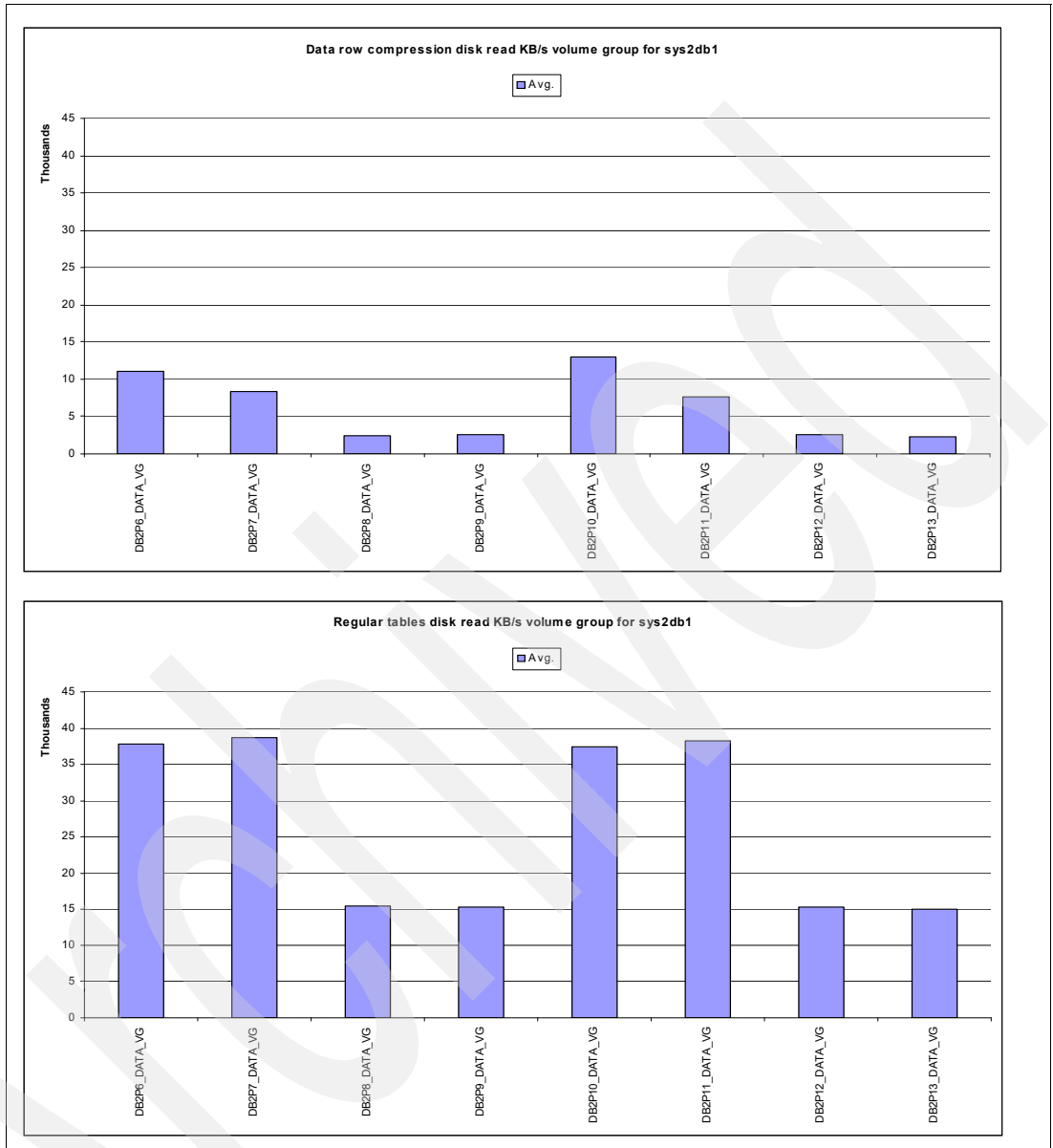


Figure 3-84 Disk read usage for both tests

To measure the percentage of disk read usage reduction, we created another chart, shown in Figure 3-85. We can correlate this number to the disk usage reduction, which was approximately the same as the compression ratio. (The exact same test was run before and after enabling data row compression.)

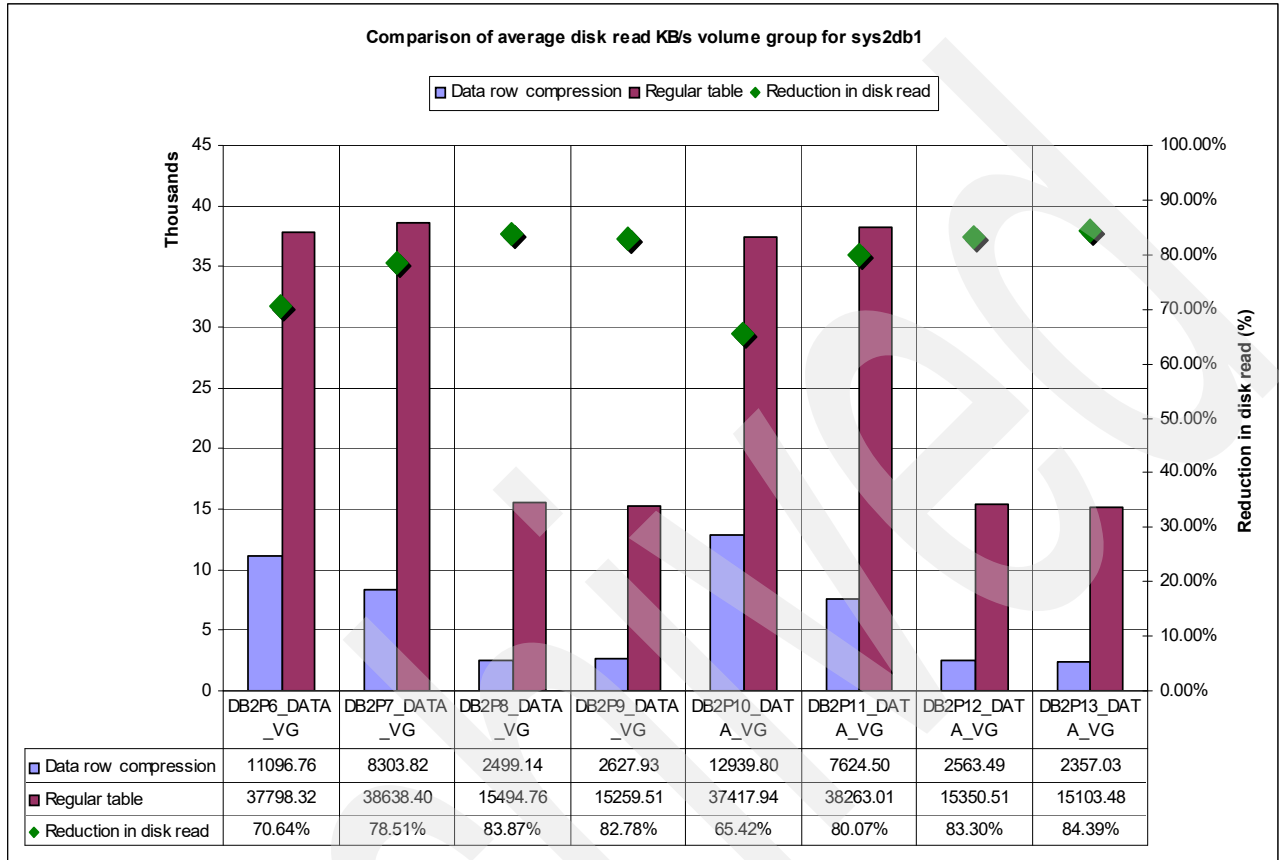


Figure 3-85 Percentage of disk read usage

Let us now go to the other side, the CPU usage. Enabling data row compression should cause higher CPU usage. The total CPU usage comparison for our tests is shown in Figure 3-86. As you see, and as expected, there is a slight increase in the CPU usage when compression is enabled, because it will have to expend extra CPU cycles to compress and uncompress data.

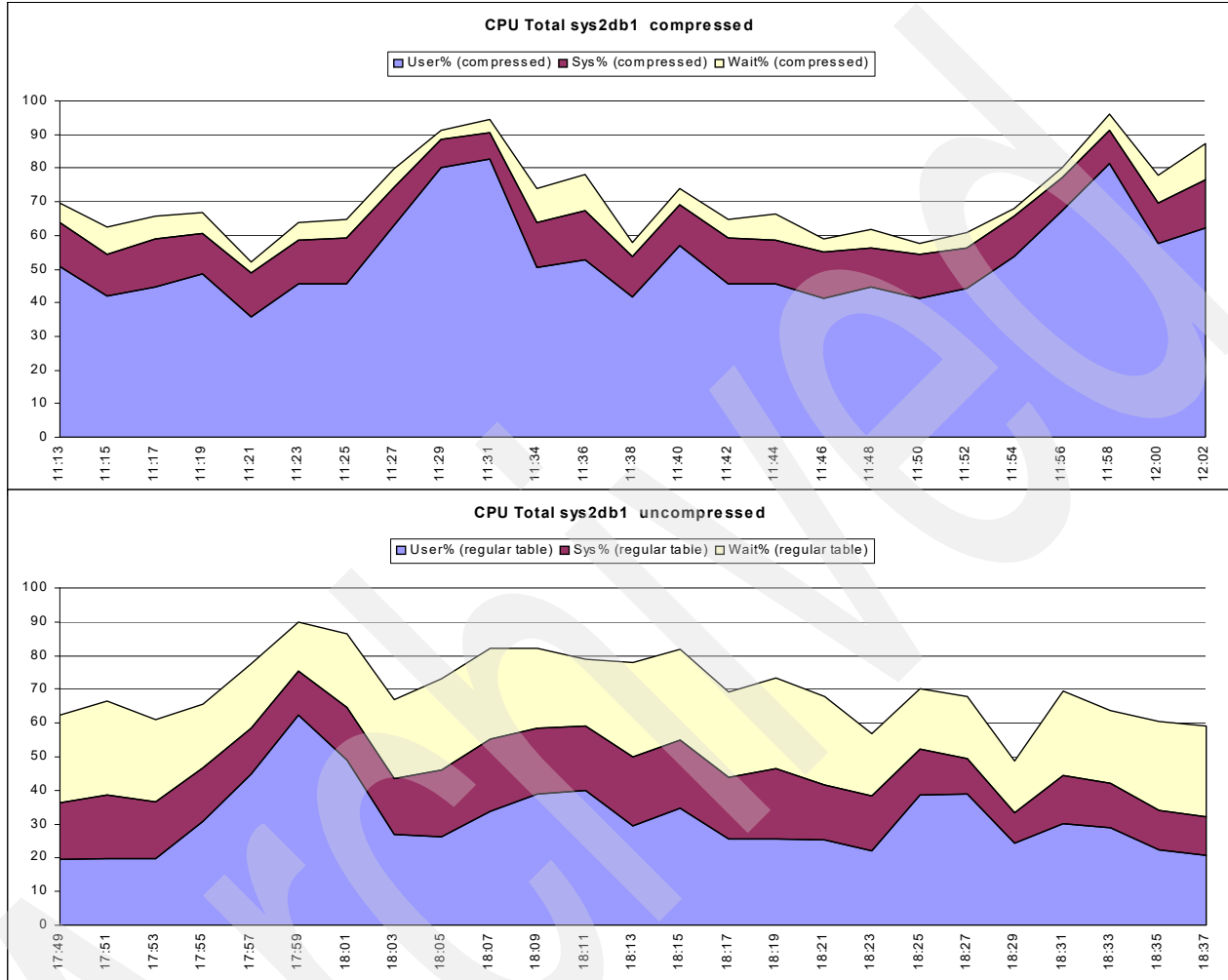


Figure 3-86 Total CPU usage comparison

From this graphic we can extract the real user CPU usage, as shown in Figure 3-87. The difference now is more clear and easier to visualize. We used two distinct Y axis to enable a well-formatted analysis over the high load phase for both runs. What is important though is to realize that enabling compression, unconditionally, will be more expensive in user CPU usage than using a regular uncompressed table.

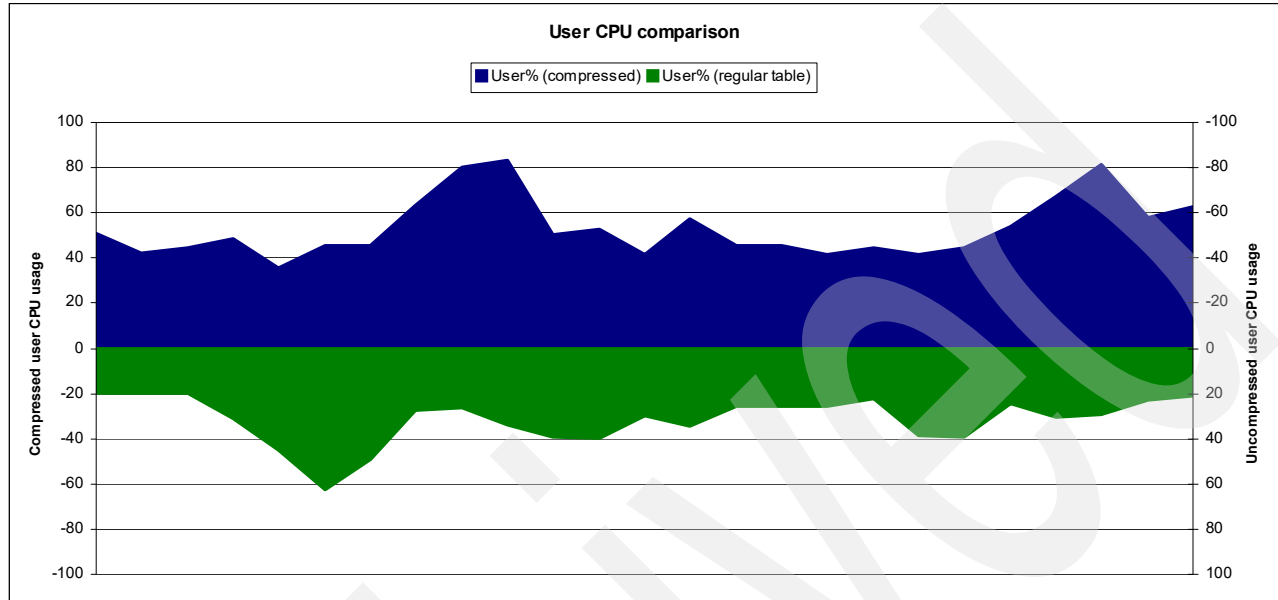


Figure 3-87 Comparison of user CPU usage

Translated in numbers, this picture would result in the values shown in Table 3-23.

Table 3-23 Average user CPU usage comparison

| State | Average user CPU usage |
|-------------------------------------|------------------------|
| Data row compression enabled tables | 53.01% |
| Regular tables | 31.17% |

However, unless you are using virtualization, I/O wait CPU is also considered for total CPU usage since it is being locked. When enabling compression, I/O wait is reduced dramatically. In our environment, we used virtualization. Therefore, this analysis would not count, because if another LPAR on the System p server required more CPU, it would use this CPU slice.

We also had a decrease of an average of 30% for the system CPU usage using compression, but we do not count on this to show our point.

Figure 3-88 shows the CPU usage, the user CPU, and the wait CPU. The results appear to be more tuned. We can explain this if we consider that the gains we have for reducing I/O roughly overlaps the needs for CPU to compress/uncompress data. This should be more sensitive on systems where there are fewer disks or storage sub-systems. The reverse is also true. On a system in which we have high CPU usage, compression can be impacted and can cause a CPU bottleneck. So consider carefully, based on your design and benefits and trade-offs presented here, whether it will have a positive response and improvement for you in your own environment.

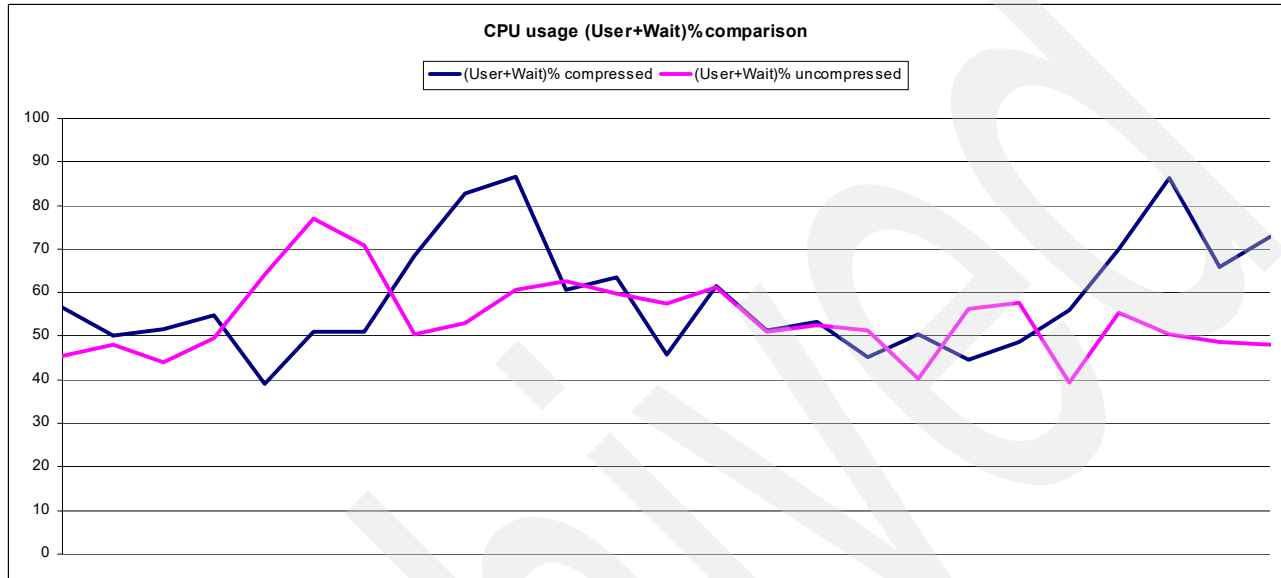


Figure 3-88 Comparison for compressed and uncompressed CPU usage (user and I/O wait)

The average total CPU usage comparison in both tests is illustrated in Table 3-24.

Table 3-24 Average total CPU usage numbers for both tests

| State | Average user CPU usage | Average wait CPU usage | Average system CPU usage | Average total CPU usage |
|----------------------|------------------------|------------------------|--------------------------|-------------------------|
| Data row compression | 53.01% | 5.72% | 12.29% | 71.03% |
| Regular tables | 31.17% | 23.04% | 16.12% | 70.33% |

A detailed memory usage comparison does not seem to be necessary since it is stable in both tests. The average memory usage is shown in Table 3-25.

Table 3-25 Average total memory usage numbers for both tests

| State | Average used memory |
|----------------------|---------------------|
| Data row compression | 114,753 MB |
| Regular tables | 114,418 MB |

We only analyzed sys2db1, but you can assume the same behavior for all remaining DB2 AIX LPARs. You can assume the same behavior for DB2 partitions that belong to the same DB2 partition group pattern.

3.4.5 Stress test comparison with concurrent I/O and file system cache

There is some discussion about the benefit of using file system cache or other file system access methods like concurrent I/O or direct I/O.

In all previous tests we ran without file system cache, using concurrent I/O. You can alter this feature on the tablespace level by executing a `ALTER TABLESPACE` command using the clause:

- ▶ `FILE SYSTEM CACHING`: All I/O operations in the target tablespace will be cached at the file system level.
- ▶ `NO FILE SYSTEM CACHING`: All I/O operations will bypass the file system level cache.

We decided to execute the same stress test procedure, enabling the file system cache to evaluate the benefits and overheads of this feature.

The overall figures we got with tablespaces with and without file system cache are shown in Figure 3-88 on page 264. The measurements have shown a considerable degradation only over-achieving one metric, the queries workload.

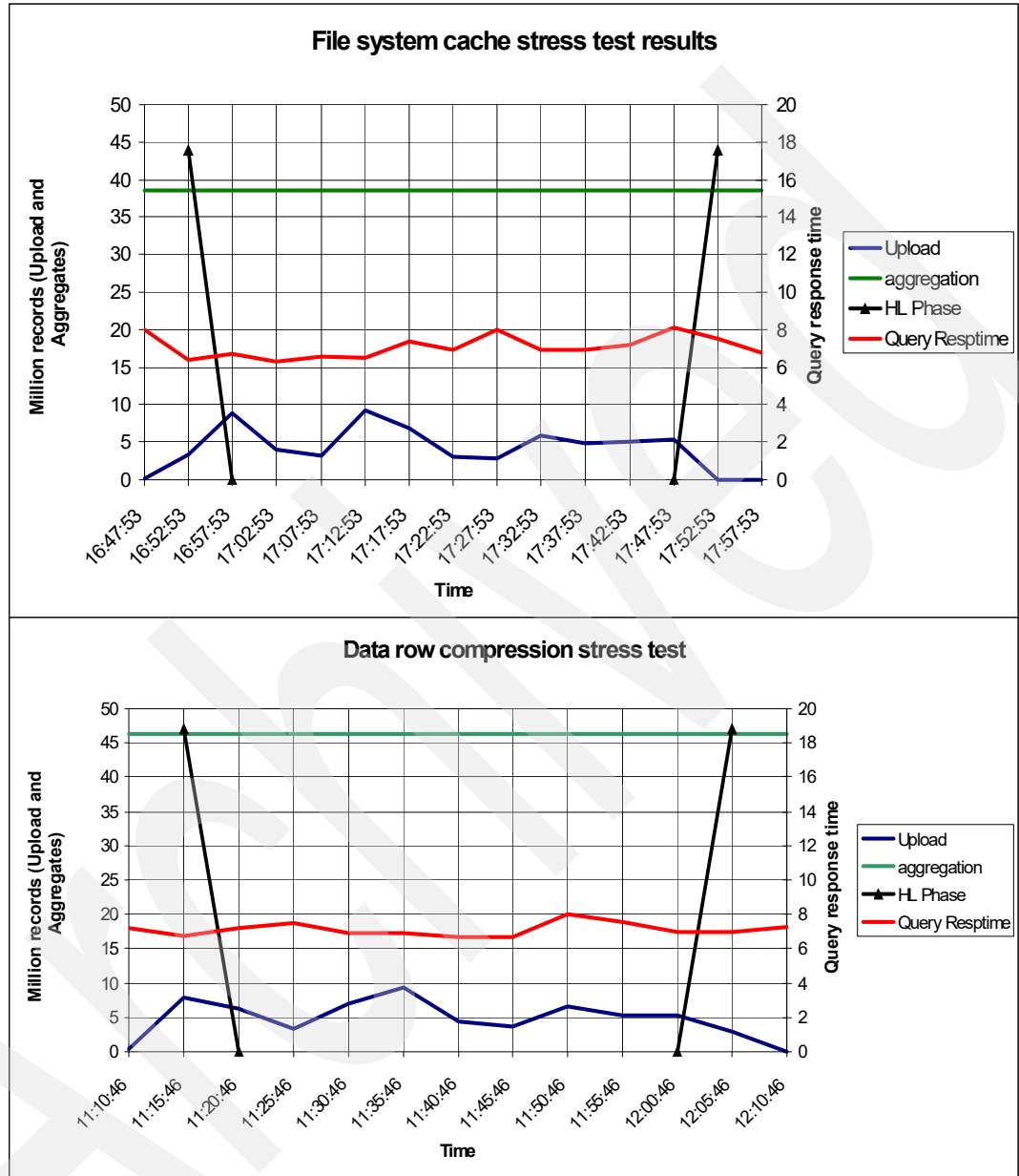


Figure 3-89 Test results snapshot comparison

Table 3-26 summarizes the results.

Table 3-26 Stress test results with file system cache

| Metric | Value |
|---------------------------------------|------------------------------|
| Average load per hour | 4.8 million records |
| Average aggregation per hour | 38.55 million records |
| Average query transactions per second | 1.45 transactions per second |
| Average query response time | 7.03 seconds |

Table 3-27 compares these values with the test we executed without file system cache. We notice a high impact over the InfoCube upload and aggregation workloads.

Table 3-27 Comparison of stress test with and without file system cache

| Metric | With file system cache | Without file system cache (CIO) | Result |
|---------------------------------------|------------------------------|---------------------------------|-----------------------|
| Average load per hour | 4.8 million records | 5.65 million records | Overhead of 15% |
| Average aggregation per hour | 38.55 million records | 46.25 million records | Improvement of 16.65% |
| Average query transactions per second | 1.45 transactions per second | 1.45 transactions per second | Same performance |
| Average query response time | 7.03 seconds | 7.12 seconds | Improvement of 1.2% |

The following figures are an analysis of these effects.

Figure 3-90 is a summary of both tests. We observe a higher I/O and CPU usage on the test executed with the file system cache.

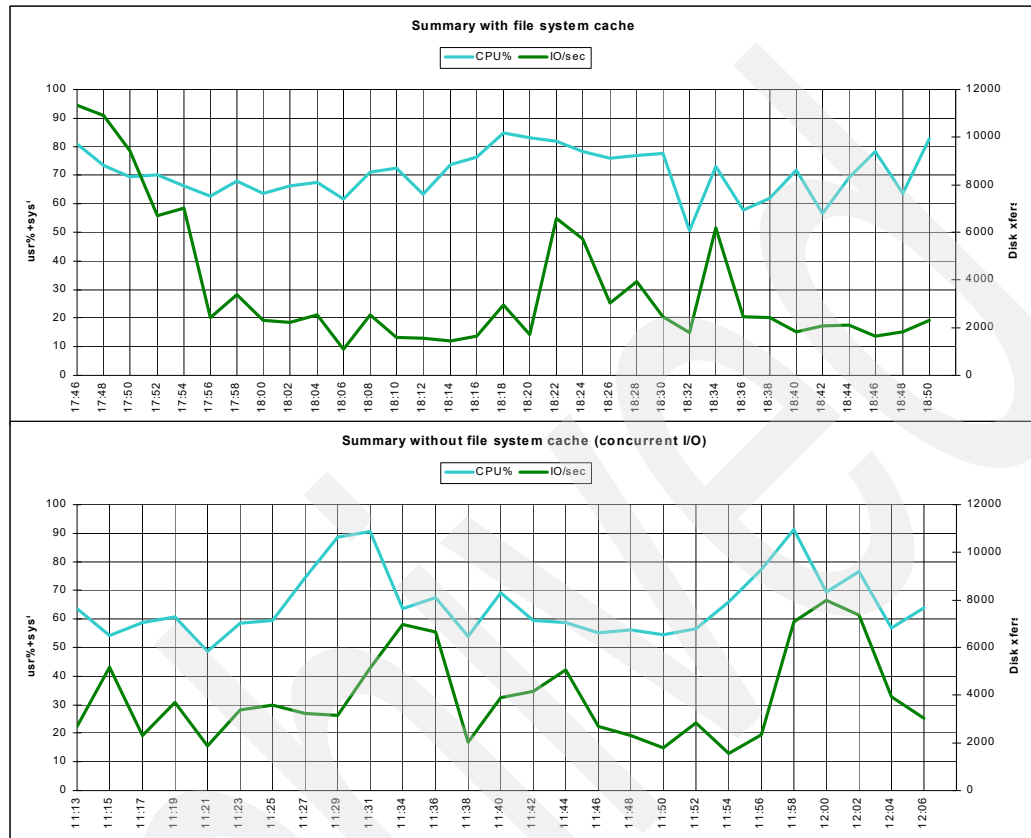


Figure 3-90 Comparison of summary usage for CPU and I/O transfer

As shown in Table 3-28, the average CPU usage denotes the overhead of over-caching data. When enabling file system cache, extra CPU cycles are required to maintain the cache, either prefetching data or cleaning it asynchronously. A slight improvement can be noticed for I/O wait CPU usage, which may relate to the small improvement we got for the query workload response time.

Table 3-28 Comparison of average CPU usage

| State | Average user CPU | Average system CPU | Average I/O wait CPU |
|---------------------------|------------------|--------------------|----------------------|
| Without file system cache | 50.7% | 11.9% | 5.5% |
| With file system cache | 54.02% | 14.07% | 4.5% |

Due to the use of file system cache, the usage of asynchronous I/O processes is higher, as shown in Figure 3-91.

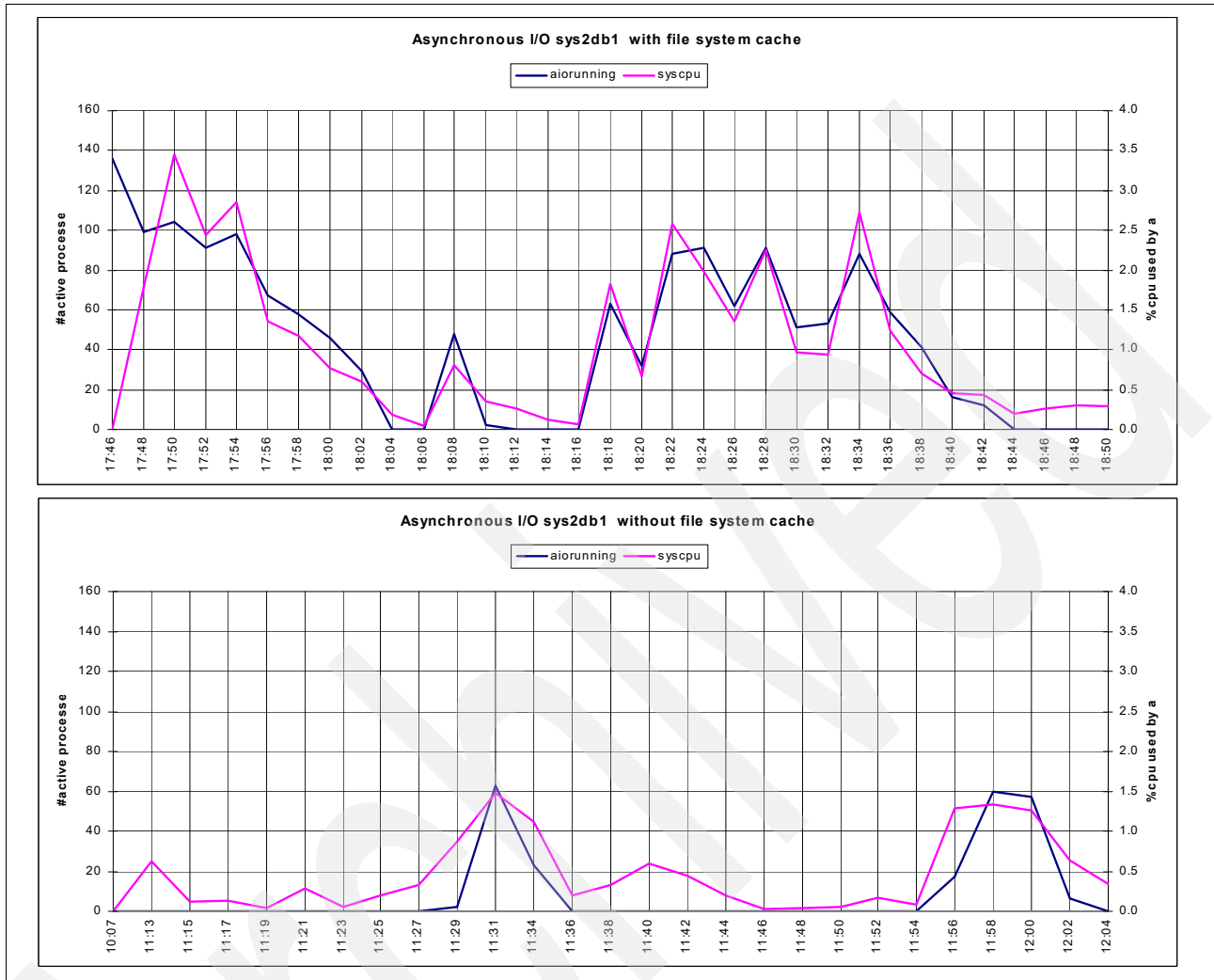


Figure 3-91 Comparison of CPU usage and number of processes for asynchronous I/O

Analyzing the overall behavior on the server, the major change is that it reads much more data than necessary. Comparing the amount of volume group disk read, shown in Figure 3-92, it becomes more clear that over-caching in this case does not help, and it actually creates an overhead. Reading and prefetching data into memory in distinct layers may cause this unwanted behavior.

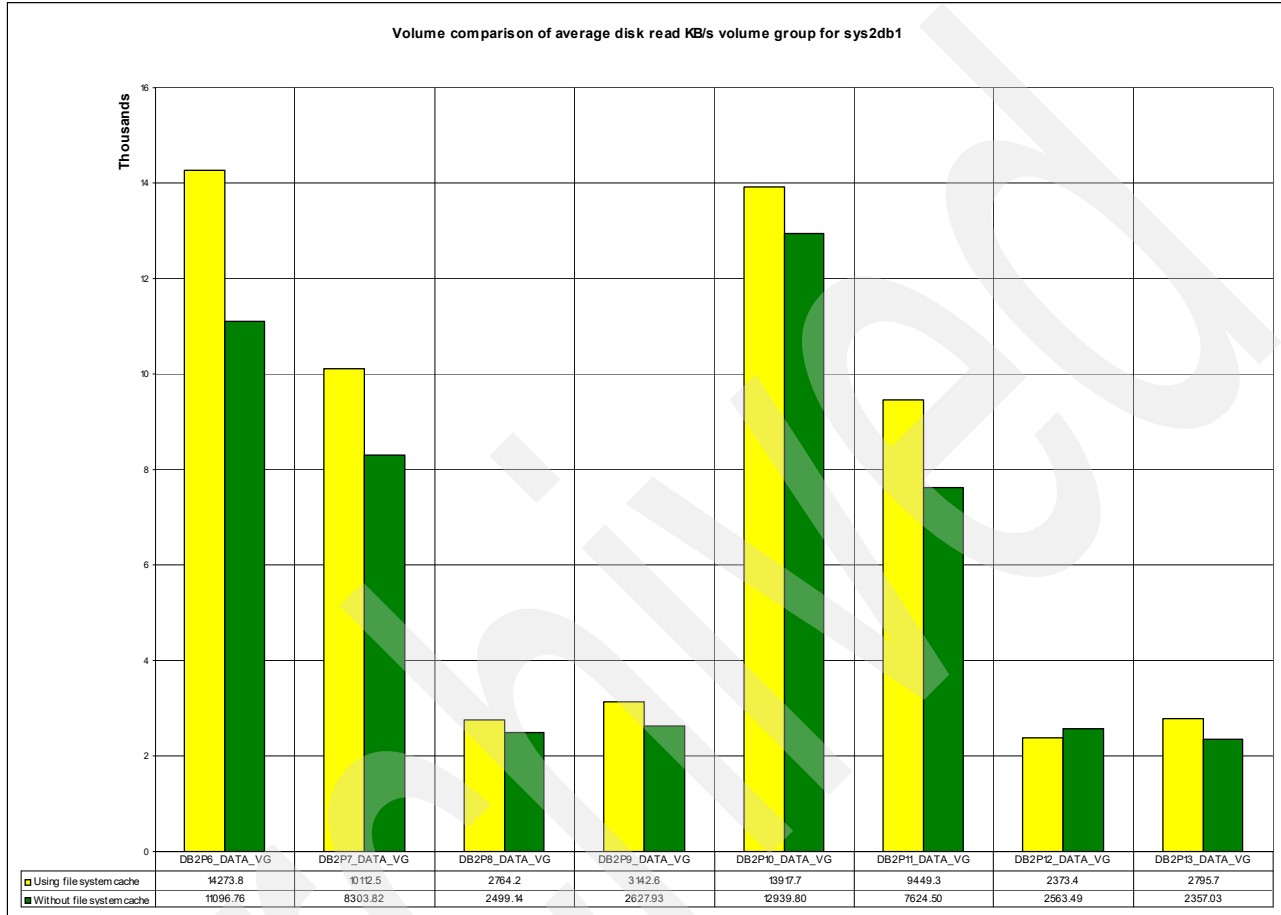


Figure 3-92 Volume comparison of average read disk for volume groups

To finalize our analysis of this test, let us check some numbers of I/O statistics depicted in Figure 3-93. The numbers show an excessive volume of read, slight overhead for transactions per second, and an improvement for maximum transactions per second. However, all remaining measurements show an overhead, reading 46.15% more data with bad results. Repair the read/write ratio, using file system cache it went up to 3.16 where without it, it were 2.02.

| Total System I/O Statistics | With file system cache | Without file system cache | Improvement/Overhead | Ratio |
|---------------------------------|------------------------|---------------------------|----------------------|--------|
| Avg tps during an interval: | 3612.52 | 3927.47 | Overhead | 8.02% |
| Max tps during an interval: | 11328.60 | 8001.30 | Improvement | 41.58% |
| Max tps interval time: | 01:04:42 | 00:52:17 | N/A | N/A |
| Total number of Mbytes read: | 317503.63 | 217239.73 | Overhead | 46.15% |
| Total number of Mbytes written: | 100504.24 | 107599.56 | Overhead | 6.59% |
| Read/Write Ratio: | 3.16 | 2.02 | Overhead | 56.47% |

Figure 3-93 I/O statistics comparison

3.4.6 Multi Dimension Clustering tables test results

Some tests were executed to test the performance value turning using the Multi Dimension Clustering (MDC) for an InfoCube fact table. We chose one specific table, one that was being reached by a heavy query and causing a high response time for the queries. We did not run any stress test, only the execution of the query measuring the time it would take.

The InfoCube fact table we used in this test was ZGTFC021. The changes we executed and measured results for were:

- ▶ Create a second table based on /BIC/FZTFC021 and cluster it on the dimension 3 field only.
- ▶ Create a third table based on /BIC/FZTFC021 and cluster it on dimension 3 and dimension time fields.

The query SQL was extracted from the DB2 catalog cache and used by SAP BW queries. This query is shown in Example 3-39. the highlighted part called A is the part that varies according to the table that we want to measure. In the example we reference the regular table SAPR3./BIC/FZGTFC021.

Example 3-39 Query used to test execution time and results

```
WITH SAPR3."/BIO/0700000004" AS ( ( SELECT SUCC , PRED , FACTOR FROM SAPR3."/BIO/0200000007"
WHERE SEQ_NR = 0 ) ) , SAPR3."/BIO/0305354308" AS ( SELECT "DU"."SID_OG_UABSMG" AS "S___232"
,"DU"."SID_OG_UVV002" AS "S___334" ,"DU"."SID_OG_UVV010" AS "S___233" ,"DU"."SID_OCURRENCY" AS
"S___045" ,"D8"."SID_OMAT_SALES" AS "S___368" ,"D9"."SID_OCUST_SALES" AS "S___587"
,"DT"."SID_OFISCYEAR" AS "S___180" , SUM ( "H4"."FACTOR" * "F"."G_AVV020" ) AS "OG_AVV020" ,
SUM ( "H4"."FACTOR" * "F"."G_AVV031" ) AS "OG_AVV031" , SUM ( "H4"."FACTOR" * "F"."G_AVV041"
) AS "OG_AVV041" , SUM ( "H4"."FACTOR" * "F"."G_AVV051" ) AS "OG_AVV051" , SUM (
"H4"."FACTOR" * "F"."G_QABSMG" ) AS "OG_QABSMG" , SUM ( "H4"."FACTOR" * "F"."G_QVV002" )
AS "OG_QVV002" , SUM ( "H4"."FACTOR" * "F"."G_QVV010" ) AS "OG_QVV010" , COUNT( * ) AS
"1ROWCOUNT"
FROM SAPR3."/BIC/FZGTFC021" "F" , A
SAPR3."/BIC/DZGTFC021U" "DU" ,
SAPR3."/BIC/DZGTFC0218" "D8" ,
SAPR3."/BIC/DZGTFC0219" "D9" ,
SAPR3."/BIC/DZGTFC021T" "DT" ,
SAPR3."/BIC/DZGTFC021P" "DP" ,
SAPR3."/BIC/DZGTFC0213" "D3" ,
SAPR3."/BIO/XMAT_SALES" "X4" ,
SAPR3."/BIC/DZGTFC0214" "D4" ,
SAPR3."/BIO/SPLANT" "S5" ,
SAPR3."/BIO/0700000004" "H4"
WHERE "F"."KEY_ZGTFC021U" = "DU"."DIMID" AND "F"."KEY_ZGTFC0218" = "D8"."DIMID" AND
"F"."KEY_ZGTFC0219" = "D9"."DIMID" AND "F"."KEY_ZGTFC021T" = "DT"."DIMID" AND
"F"."KEY_ZGTFC021P" = "DP"."DIMID" AND "F"."KEY_ZGTFC0213" = "D3"."DIMID" AND
"D8"."SID_OMAT_SALES" = "X4"."SID" AND "F"."KEY_ZGTFC0214" = "D4"."DIMID" AND "D4"."SID_OPLANT"
= "S5"."SID" AND "D3"."SID_OREC_TYPE" = "H4"."SUCC" AND ((( "DP"."SID_OCHNGID" = 0 )) AND ((
"D3"."SID_OCURTYPE" = 2 )) AND (( "DT"."SID_OFISCVARNT" = 19 )) AND (( "X4"."S_OPROD4" =
3320168 )) AND (( "S5"."PLANT" BETWEEN '0000' AND 'ZZZZ' )) AND (( "DP"."SID_ORECORDTP" = 0 ))
AND (( "DP"."SID_OREQUID" <= 558044 )) AND (( "D3"."SID_OVERSION" = 1 )) AND ((
"D3"."SID_OVTYPE" = 10 )) AND (( "D4"."SID_YGT_ERKRS" = 19 )))) AND ((( "DT"."SID_OFISCYEAR" =
190002004 ))) OR ((( "DT"."SID_OFISCYEAR" = 190002005 ))) AND "X4"."OBJVERS" = 'A' AND
"H4"."SUCC" <> 2000008999 GROUP BY "DU"."SID_OG_UABSMG" , "DU"."SID_OG_UVV002"
,"DU"."SID_OG_UVV010" , "DU"."SID_OCURRENCY" , "D8"."SID_OMAT_SALES" , "D9"."SID_OCUST_SALES"
,"DT"."SID_OFISCYEAR" ) SELECT "S___232" , "S___334" , "S___233" , "S___045" , "S___368" ,
```

"S___587" , "S___180" , "OG_AVV020" , "OG_AVV031" , "OG_AVV041" , "OG_AVV051" , "OG_QABSMG" , "OG_QVV002" , "OG_QVV010" , "IROWCOUNT" FROM SAPR3." /BIO/0305354308"

It is important to note that when having MDC, space usage can become an issue if you do not chose the correct dimensions (or the correct extent size) in the tablespace. MDC tables allocate one extent per dimension to ensure clustering, and whenever more space is needed, new extents will be allocated.

Important: We strongly recommend that you read SAP note 942909^a “DB6: Multi-Dimensional Clustering for SAP BW 3.x” before enabling this feature in your system. We enabled MDC directly in DB2 because we used a test environment, not a production environment.

a. More information about this SAP note is available on the following Web site:
<https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/webcontent/uuid/e02ce76b-588c-2910-7dab-faa1e94ad012>

All objects were copied and created over the two new tables. Only the clustering index creation was changed (to be created as a regular index because the new table would already be clustered by other defined columns). The definition for SAPR3./BIC/FZFTFC021 is shown in Example 3-40. In the highlighted code, you see that we have compression enabled for this table. Since we did all tests in a chronological fashion, we used the baseline as a compressed table. This did not compromise any results since the two tables created were also compressed.

Example 3-40 InfoCube ZGTFC021 fact table definition

```
CREATE TABLE "SAPR3"."./BIC/FZGTFC021" (  
    "KEY_ZGTFC021P" INTEGER NOT NULL WITH DEFAULT 0 ,  
    "KEY_ZGTFC021T" INTEGER NOT NULL WITH DEFAULT 0 ,  
    ===== lines removed =====  
    "KEY_ZGTFC021C" INTEGER NOT NULL WITH DEFAULT 0 ,  
    "KEY_ZGTFC021D" INTEGER NOT NULL WITH DEFAULT 0 ,  
    "G_QVV010" DECIMAL(17,3) NOT NULL WITH DEFAULT 0 ,  
    "G_QVV002" DECIMAL(17,3) NOT NULL WITH DEFAULT 0 ,  
    ===== lines removed =====  
    "G_AVV390" DECIMAL(17,2) NOT NULL WITH DEFAULT 0 ,  
    "G_AVV901" DECIMAL(17,2) NOT NULL WITH DEFAULT 0 )  
COMPRESS YES  
DISTRIBUTE BY HASH("KEY_ZGTFC0211",  
    "KEY_ZGTFC0212",  
    "KEY_ZGTFC0213",  
    "KEY_ZGTFC0214",  
    "KEY_ZGTFC0215",  
    "KEY_ZGTFC0216",  
    "KEY_ZGTFC0217",  
    "KEY_ZGTFC0218",  
    "KEY_ZGTFC0219",  
    "KEY_ZGTFC021A",  
    "KEY_ZGTFC021B",  
    "KEY_ZGTFC021C",  
    "KEY_ZGTFC021D",  
    "KEY_ZGTFC021T",  
    "KEY_ZGTFC021U")  
    IN "YMFACTD11" INDEX IN "YMFACTI11" ;
```

The following steps were executed in SAPR3./BIC/FZGTFC021 prior the creation of the new tables:

1. Execution of a table reorganization
2. Collection of statistics for the tables and indexes

The following steps were executed for each table:

1. Creation of the new table on a different schema (but on the same tablespace) and partitioning group, clustering the table on dimensions, and creation of all indexes used by the table
2. Execution of a load from the cursor from the InfoCube fact table SAPR3./BIC/FZTFC021 into the new created table
3. Execution of a table reorganization on the new table created
4. Collection of statistics for the table and indexes on the newly created table

Table clustered on dimension 3 creation

Example 3-41 shows the table definition. The highlighted line A shows the schema name for the table. The highlighted line B shows the difference in the definition that made it clustered by dimension 3.

Example 3-41 ZGTFC021 MDC table clustered on dimension 3

```

A CREATE TABLE "FABI0D3_NEW"."/BIC/FZGTFC021" (
    "KEY_ZGTFC021P" INTEGER NOT NULL WITH DEFAULT 0 ,
    "KEY_ZGTFC021T" INTEGER NOT NULL WITH DEFAULT 0 ,
    ===== lines removed
    "KEY_ZGTFC021C" INTEGER NOT NULL WITH DEFAULT 0 ,
    "KEY_ZGTFC021D" INTEGER NOT NULL WITH DEFAULT 0 ,
    "G_QVV010" DECIMAL(17,3) NOT NULL WITH DEFAULT 0 ,
    "G_QVV002" DECIMAL(17,3) NOT NULL WITH DEFAULT 0 ,
    ===== lines removed
    "G_AVV390" DECIMAL(17,2) NOT NULL WITH DEFAULT 0 ,
    "G_AVV901" DECIMAL(17,2) NOT NULL WITH DEFAULT 0 )
    COMPRESS YES
    DISTRIBUTE BY HASH("KEY_ZGTFC0211",
    "KEY_ZGTFC0212",
    "KEY_ZGTFC0213",
    "KEY_ZGTFC0214",
    "KEY_ZGTFC0215",
    "KEY_ZGTFC0216",
    "KEY_ZGTFC0217",
    "KEY_ZGTFC0218",
    "KEY_ZGTFC0219",
    "KEY_ZGTFC021A",
    "KEY_ZGTFC021B",
    "KEY_ZGTFC021C",
    "KEY_ZGTFC021D",
    "KEY_ZGTFC021T",
    "KEY_ZGTFC021U")
    IN "YMFACTD11" INDEX IN "YMFACTI11"
B ORGANIZE BY DIMENSIONS
    ( "KEY_ZGTFC0213" )

```

Example 3-42 describes the command executed to load data into the newly created table.

Example 3-42 Load from cursor to load data into the new table

```
db2 "declare cur cursor for select * from SAPR3.\"/BIC/FZGTFC021\""  
db2 "load from cur of cursor insert into FABI0D3_NEW.\"/BIC/FZGTFC021\""  
NONRECOVERABLE"
```

The load from the cursor took approximately 40 minutes to load 48,637,880 records. Example 3-43 shows the output of the load.

Example 3-43 Load from cursor command output

| Agent Type | Node | SQL Code | Result |
|------------|--------------------------------------|-----------|----------|
| LOAD | 008 | +00000000 | Success. |
| LOAD | 012 | +00000000 | Success. |
| LOAD | 016 | +00000000 | Success. |
| LOAD | 020 | +00000000 | Success. |
| LOAD | 024 | +00000000 | Success. |
| LOAD | 028 | +00000000 | Success. |
| LOAD | 032 | +00000000 | Success. |
| LOAD | 036 | +00000000 | Success. |
| PARTITION | 000 | +00000000 | Success. |
| RESULTS: | 8 of 8 LOADs completed successfully. | | |

Summary of Partitioning Agents:

Rows Read = 48637880
Rows Rejected = 0
Rows Partitioned = 48637880

Summary of LOAD Agents:

Number of rows read = 48637880
Number of rows skipped = 0
Number of rows loaded = 48637880
Number of rows rejected = 0
Number of rows deleted = 0
Number of rows committed = 48637880

DB20000I The SQL command completed successfully.

Afterwards a table reorganization and a statistics update was executed.

Table clustered on dimension 3 and dimension time creation

Example 3-41 on page 273 shows the table definition. The highlighted line A shows the schema name for the table. The highlighted line B shows the difference in the definition, which caused it to be clustered by dimension 3 and dimension time.

Example 3-44 ZGTFC021 MDC table clustered on dimension 3 and dimension time

```

A CREATE TABLE "FABI0D3_DT" "."/BIC/FZGTFC021" (
    "KEY_ZGTFC021P" INTEGER NOT NULL WITH DEFAULT 0 ,
    "KEY_ZGTFC021T" INTEGER NOT NULL WITH DEFAULT 0 ,
    ===== lines removed
    "KEY_ZGTFC021C" INTEGER NOT NULL WITH DEFAULT 0 ,
    "KEY_ZGTFC021D" INTEGER NOT NULL WITH DEFAULT 0 ,
    "G_QVV010" DECIMAL(17,3) NOT NULL WITH DEFAULT 0 ,
    "G_QVV002" DECIMAL(17,3) NOT NULL WITH DEFAULT 0 ,
    ===== lines removed
    "G_AVV390" DECIMAL(17,2) NOT NULL WITH DEFAULT 0 ,
    "G_AVV901" DECIMAL(17,2) NOT NULL WITH DEFAULT 0 )
    COMPRESS YES
    DISTRIBUTE BY HASH("KEY_ZGTFC0211",
    "KEY_ZGTFC0212",
    "KEY_ZGTFC0213",
    "KEY_ZGTFC0214",
    "KEY_ZGTFC0215",
    "KEY_ZGTFC0216",
    "KEY_ZGTFC0217",
    "KEY_ZGTFC0218",
    "KEY_ZGTFC0219",
    "KEY_ZGTFC021A",
    "KEY_ZGTFC021B",
    "KEY_ZGTFC021C",
    "KEY_ZGTFC021D",
    "KEY_ZGTFC021T",
    "KEY_ZGTFC021U")
    IN "YMFACTD11" INDEX IN "YMFACTI11"
B ORGANIZE BY DIMENSIONS
    ( "KEY_ZGTFC0213",
    "KEY_ZGTFC021T")
  
```

Example 3-42 on page 274 shows the command executed to load data into the newly created table.

Example 3-45 Load from cursor to load data into the new table

```

db2 "declare cur cursor for select * from SAPR3.\./BIC/FZGTFC021\"
db2 "load from cur of cursor insert into FABI0D3_DT.\./BIC/FZGTFC021\"
NONRECOVERABLE"
  
```

The load from the cursor took approximately 17 minutes to load 48,637,880 records. Example 3-43 on page 274 shows the output of the load.

Example 3-46 Load from cursor command output

| Agent Type | Node | SQL Code | Result |
|------------|------|-----------|----------|
| LOAD | 008 | +00000000 | Success. |

| | | | |
|-----------|--------------------------------------|-----------|----------|
| LOAD | 012 | +00000000 | Success. |
| LOAD | 016 | +00000000 | Success. |
| LOAD | 020 | +00000000 | Success. |
| LOAD | 024 | +00000000 | Success. |
| LOAD | 028 | +00000000 | Success. |
| LOAD | 032 | +00000000 | Success. |
| LOAD | 036 | +00000000 | Success. |
| PARTITION | 000 | +00000000 | Success. |
| RESULTS: | 8 of 8 LOADs completed successfully. | | |

Summary of Partitioning Agents:

Rows Read = 48637880
 Rows Rejected = 0
 Rows Partitioned = 48637880

Summary of LOAD Agents:

Number of rows read = 48637880
 Number of rows skipped = 0
 Number of rows loaded = 48637880
 Number of rows rejected = 0
 Number of rows deleted = 0
 Number of rows committed = 48637880

Afterwards, a table reorganization and a statistics update were executed.

Compression insight

Some insights about compression can be illustrated here. As we discussed previously in 3.4.2, “Tests with DB2 data row compression” on page 228, the compression rate of a table may change over time. Loading additional data in such a table will be compressed based on the current dictionary. However, the structure of the data may change over time, and thus rebuild of the compression dictionary may change the compression rate.

This worst case behavior can be seen if compression is enabled on an empty table. No compression dictionary is built, and thus compression dramatically changes after reorganization.

Let us use one table as an example, the newly created table FABIOD3_NEW./BIC/FZGTFC021.

After loading data and collecting statistics we got the figures illustrated in Figure 3-94, row number 1, and after a table reorganization to create the compression dictionary and collecting statistics, row number 2. The size of the table before creating the dictionary would be exactly the same as an uncompressed table since we did not have any data and no compression dictionary was created.

| | TABSCHEMA | TABNAME | STATS_TIME | TABLEID |
|---|-------------|----------------|----------------------------|---------|
| 1 | FABIOD3_NEW | /BIC/FZGTFC021 | 2007-03-14-19.13.07.747728 | 10 |
| 2 | FABIOD3_NEW | /BIC/FZGTFC021 | 2007-03-14-19.23.47.649930 | 10 |

| CARD | NPAGES | FPAGES | COMPRESSION | AVGROWSIZE |
|----------|---------|---------|-------------|------------|
| 48590352 | 3239448 | 3240320 | R | 1037 |
| 48590352 | 422480 | 423424 | R | 141 |

Figure 3-94 Comparison of compressed table before and after creating compression dictionary

For large loads, you should consider that you would have a better result for compression if you load a portion of data and create the compression dictionary and then, afterwards, load the remaining portion of data. This would help data load to get a better compression ratio. On the other hand, it could cause some performance degradation (because it would need some extra CPU cycles to compress data). SAP R3load has a new option that allows you to load a portion of data to create the compression dictionary. This is `-loadprocedure fast COMPRESS`. For further information refer to SAP note 905614, "DB6: R3load -loadprocedure fast COMPRESS"².

Test results

We executed two runs for each table. The first run is a cold run with no data in the buffer pools.

² For more information about the SAP-note, please go to <ftp://ftp.software.ibm.com/software/data/pubs/papers/DB2-SAP-compression.pdf>

The response times we got for all three tables are shown in Figure 3-95. The regular table in the SAPR3 schema is much more dependent on the buffer pool, and you can see how response time drops for the second run. Both MDC tables have a slight improvement on the second run.

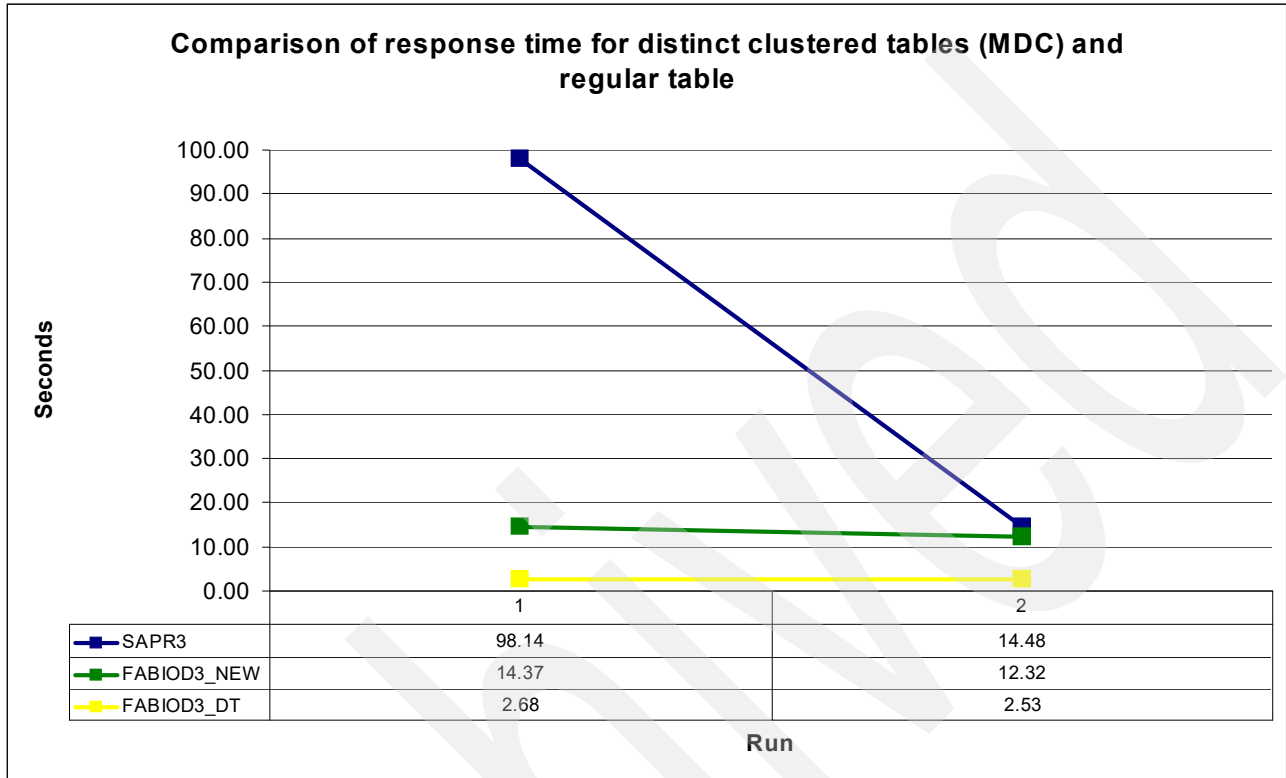


Figure 3-95 MDC clustered tables and regular table response time comparison

Some other comparisons must be done to build your own conclusions. From the CPU usage point of view, MDCs present better results for the user and system CPU time, as shown in Figure 3-96.

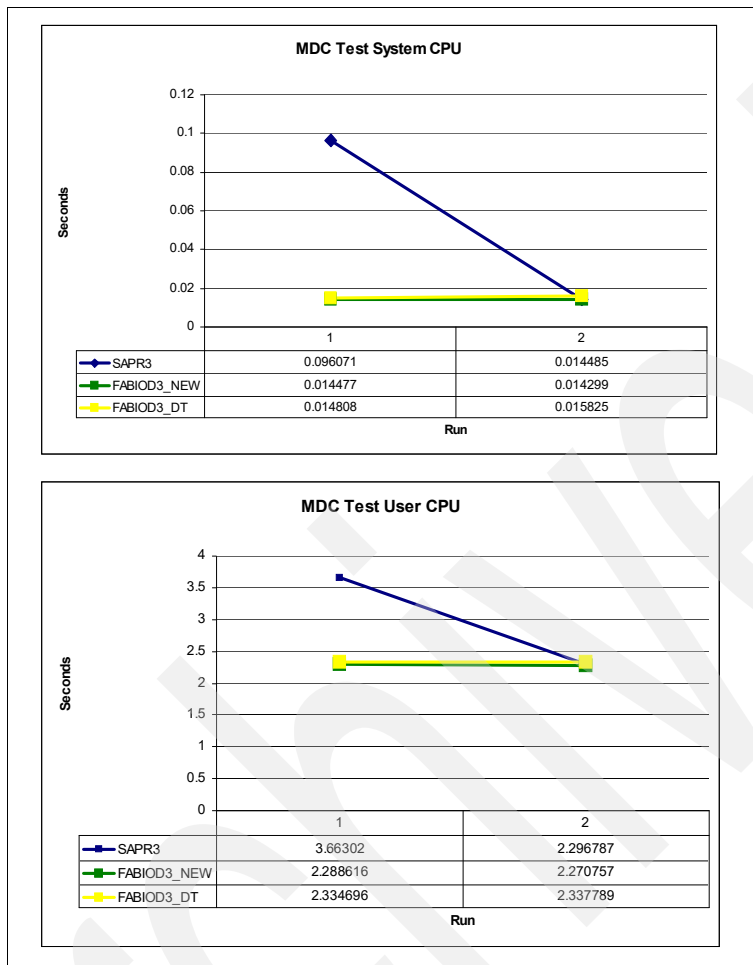


Figure 3-96 User and system CPU time comparison between MDC tables and regular table

The last thing that we consider for this MDC test is space usage. Let us compare the information gathered from the catalog tables shown in Figure 3-97. The CLUSTERED column values for the last two records are MDC tables. That can be also checked by the number in the ACTIVE_BLOCKS column. From the space point of view we need to check FPAGES.

| | TABSCHEMA | TABNAME | STATS TIME | CARD |
|--|-------------|----------------|----------------------------|----------|
| Regular table | SAPR3 | /BIC/FZGTFC021 | 2007-03-02-16.16.15.773225 | 48590352 |
| Clustered on Dimension 3 | FABIOD3_NEW | /BIC/FZGTFC021 | 2007-03-14-19.23.47.649930 | 48590352 |
| Clustered on Dimension 3 and Dimension 8 | FABIOD3_DT | /BIC/FZGTFC021 | 2007-03-15-14.55.05.436289 | 48590352 |

| | NPAGES | FPAGES | TBSPACE | COMPRESSION | CLUSTERED | ACTIVE_BLOCKS |
|---|--------|--------|-----------|-------------|-----------|---------------|
| ? | 422416 | 422448 | YMFACTD11 | R | - | 0 |
| ? | 422480 | 423424 | YMFACTD11 | R | Y | 52920 |
| ? | 429456 | 485184 | YMFACTD11 | R | Y | 60640 |

Figure 3-97 Table statistics comparison

The number shown in column ACTIVE_BLOCKS is the number of MDC blocks created. Each block is exactly the same size as the tablespace extent, which in our case is eight pages. We compare the space overhead and improvement for response time considering only the second run since on the first one. A regular table is strongly dependent on the buffer pool. The comparison table is depicted in Figure 3-98. We used the regular compressed table as a baseline to measure the percentage of space overhead and the response time improvement.

| | TABSCHEMA | TABNAME | Overhead of space used (%) | Improvement for response time |
|--|-------------|----------------|----------------------------|-------------------------------|
| Regular table | SAPR3 | /BIC/FZGTFC021 | Baseline | Baseline |
| Clustered on Dimension 3 | FABIOD3_NEW | /BIC/FZGTFC021 | 0.22% | 14.93% |
| Clustered on Dimension 3 and Dimension 8 | FABIOD3_DT | /BIC/FZGTFC021 | 14.59% | 83% |

Figure 3-98 Overheads and improvements comparison for MDC tables compared to the regular table

The overhead of space used varies with the dimensions chosen. There are tools available to help you evaluate that, like DB2 Design Advisor and RSDB6_MDC, which is a report tool built in SAP BW. For further information about DB2 Design Advisor and RSDB6_MDC refer to:

- ▶ DB2 Design Advisor: DB2 Info Center
<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/c0005144.htm>
- ▶ RSDB6_MDC and how to implement it on SAP: SAP note 942909: "DB6: Multi-Dimensional Clustering for SAP BW 3.x"

3.4.7 STMM and DPF (SAP NetWeaver BI)

When self-tuning is enabled in partitioned database environments, there is a single database partition, known as the tuning partition, that monitors the memory configuration and propagates any configuration changes to all other database partitions to maintain a consistent configuration across all the participating database partitions.

The tuning partition is selected based on a number of characteristics, such as the number of database partitions in the partition group and the number of buffer pools defined.

To determine which database partition is currently specified as the tuning partition, use the following ADMIN_CMD command:

```
CALL SYSPROC.ADMIN_CMD('get stmm tuning dbpartitionnum')
```

To change the tuning partition, use the following ADMIN_CMD command:

```
CALL SYSPROC.ADMIN_CMD('update stmm tuning dbpartitionnum <db_partition_num>')
```

When you issue this command, the tuning partition will be updated asynchronously or at the next database startup.

To have the memory tuner automatically re-select the tuning partition, enter -1 for the <db_partition_num> value.

The memory tuner will only be started in a DPF environment if the database is activated by an explicit ACTIVATE DATABASE command because self-tuning requires all partitions to be active before it can properly tune the memory on a multi-partition system.

To disable self-tuning for a subset of database partitions, set the self_tuning_mem configuration parameter to OFF for the database partitions that you want to leave untuned.

To disable self-tuning for a subset of the memory consumers controlled by configuration parameters on a particular database partition, set the value of the relevant configuration parameter or buffer pool size to `MANUAL` or a specific value on that database partition.

To disable tuning for a particular buffer pool on a database partition, issue an `ALTER BUFFER POOL` command specifying a size value and a value for the `PARTITIONNUM` parameter for the partition where self-tuning is to be disabled.

An `ALTER BUFFERPOOL` statement that specifies the size on a particular database partition will create an exception entry for the given buffer pool in the `SYSCAT.SYSBUFFERPOOLNODES` catalog, or update the exception entry if one already exists. When an exception entry exists for a buffer pool in this catalog, that buffer pool will not participate in self-tuning when the default buffer pool size is set to `AUTOMATIC`. To remove an exception entry so that a buffer pool can be re-enabled for self-tuning:

- ▶ Disable tuning for this buffer pool by issuing an `ALTER BUFFERPOOL` statement, setting the buffer pool size to a specific value.
- ▶ Issue another `ALTER BUFFERPOOL` statement with the `PARTITIONNUM` clause specified to set the size of the buffer pool on this database partition to the default buffer pool size.
- ▶ Enable self-tuning by issuing another `ALTER BUFFERPOOL` statement, setting the size to `AUTOMATIC`.

Ideally, your data should be distributed evenly across all of your database partitions, and the workload running on each partition should have similar memory requirements. If the data distribution is skewed, so that one or more of your database partitions contains significantly more or less data than other database partitions, these anomalous database partitions should not be enabled for self-tuning. The same is true if the memory requirements are skewed across the database partitions. It can happen, for example, if resource-intensive sorts are only performed on one partition, or if some database partitions are associated with different hardware with more available memory. Self-tuning can still be enabled on some database partitions in this type of environment. To take advantage of self-tuning memory in environments with skew, identify a set of database partitions that have similar data and memory requirements and enable them for self-tuning. Memory configuration in the remaining partitions should be configured manually.

STMM may not be an option for SAP NetWeaver BI because the distribution and workload are different between the partitions. In this case, with uneven partitions workload and distribution configuring, we do not recommend STMM.

In our environment, we have defined STMM for a set of DB2 partitions. In our case, partitions 6 to 37 were where we had the largest objects. To enable this process we went through the following steps:

1. We set up STMM ON for DB2 partitions 6 to 37 by executing the commands depicted in Example 3-47.

Example 3-47 DB2 commands executed for DB2 partitions 6 to 37

```
db2 update database configuration for EBB using self_tuning_mem ON
db2 update database configuration for EBB using locklist automatic
db2 update database configuration for EBB using maxlocks automatic
db2 update database configuration for EBB using pckcachesz automatic
db2 update database configuration for EBB using sheapthres automatic
db2 update database configuration for EBB using sortheap automatic
```

2. We set the tuning partition to be partition 8 by issuing the following command in the concentrator partition:
CALL SYSPROC.ADMIN_CMD('update stmm tuning dbpartitionnum 8')
3. STMM was enabled for BP_STD_16K, IBMDEFAULTBP, and BP_AGGR_16K buffer pools, and the execution for BP_STD_16K, as shown in Example 3-48.

Example 3-48 STMM enablement for BP_STD_16K

```
ALTER BUFFERPOOL BP_STD_16K SIZE 10240;

-- BE SURE THAT ALL PARTITIONS WE DEFINED WILL NOT BE ON THE EXCEPTION LIST
ALTER BUFFERPOOL BP_STD_16K DBPARTITIONNUM 6 SIZE 10240;
<...>
ALTER BUFFERPOOL BP_STD_16K DBPARTITIONNUM 37 SIZE 10240;

-- BE SURE THAT PARTITIONS 0 TO 5 WILL BE ON THE EXCEPTION LIST
ALTER BUFFERPOOL BP_STD_16K DBPARTITIONNUM 0 SIZE 5000;
<...>
ALTER BUFFERPOOL BP_STD_16K DBPARTITIONNUM 5 SIZE 5000;

ALTER BUFFERPOOL BP_STD_16K SIZE AUTOMATIC;
```

4. Finally, we recycled the instance and explicitly activated the database by issuing an `ACTIVATE DATABASE` command. This is required to enable STMM in DPF environments.

We executed a workload using SAP queries to check results and how to see how it would dynamically change and tune values for distinct memory sets.

The results we got for the query response time using STMM are shown in Figure 3-99. The top value shown for a specific query STR_D that impacted the average response time was when it went directly to the InfoCube fact table. This is heavily dependant on the buffer pool size.

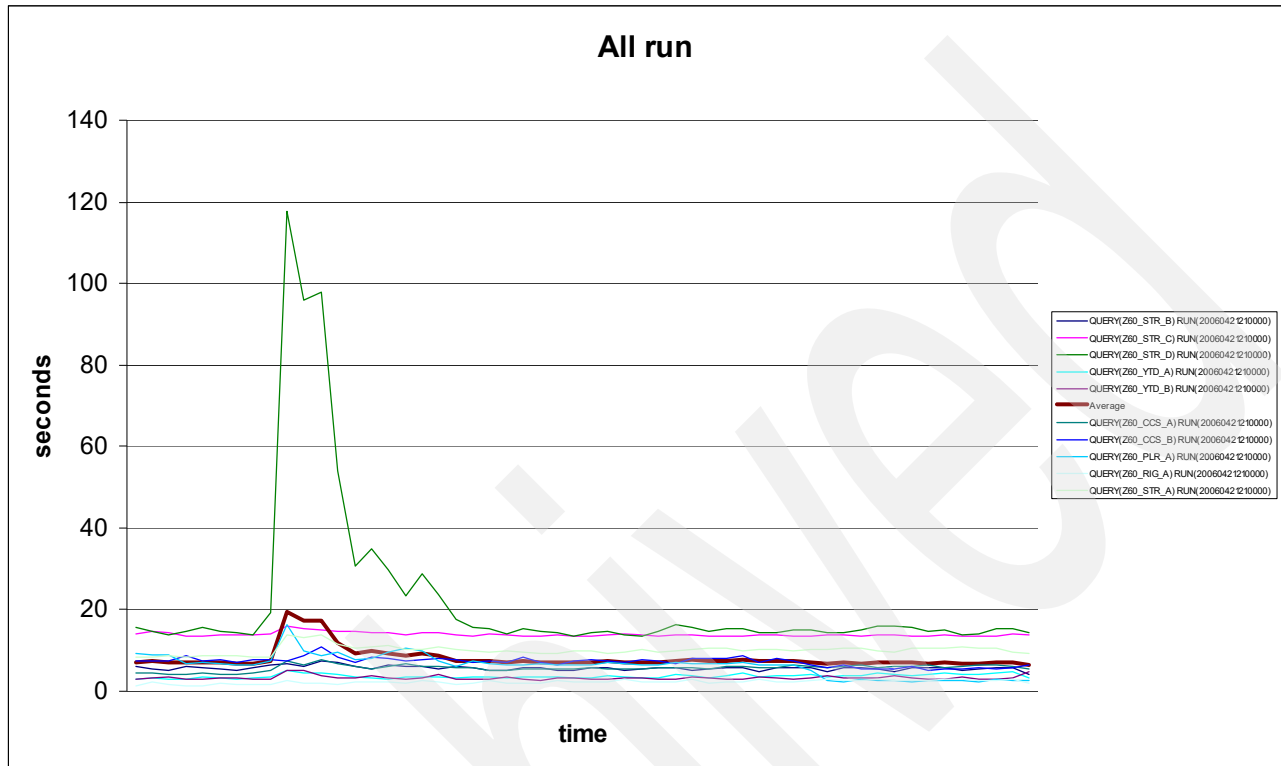


Figure 3-99 Queries response time with STMM

If you overlap the query behavior with the buffer pool response time for BP_STD_16K, which is the one used by InfoCubes fact tables and the one that was reached by this query, you will have the results shown in Figure 3-100 on page 284. This shows how STMM works and how it adapts based on the profile of one of the DB2 partitions. Choosing the correct partition is critical to having the best response time with STMM. Choose the most accessed or more stressed DB2 partition. Even in a well-balanced environment, production demands are normally unpredictable, sometimes causing a slight unbalance between partitions, unless you are spreading all objects over all partitions. It is important to highlight that all other buffer pools configured to automatic and database parameters also varied according to the workload.

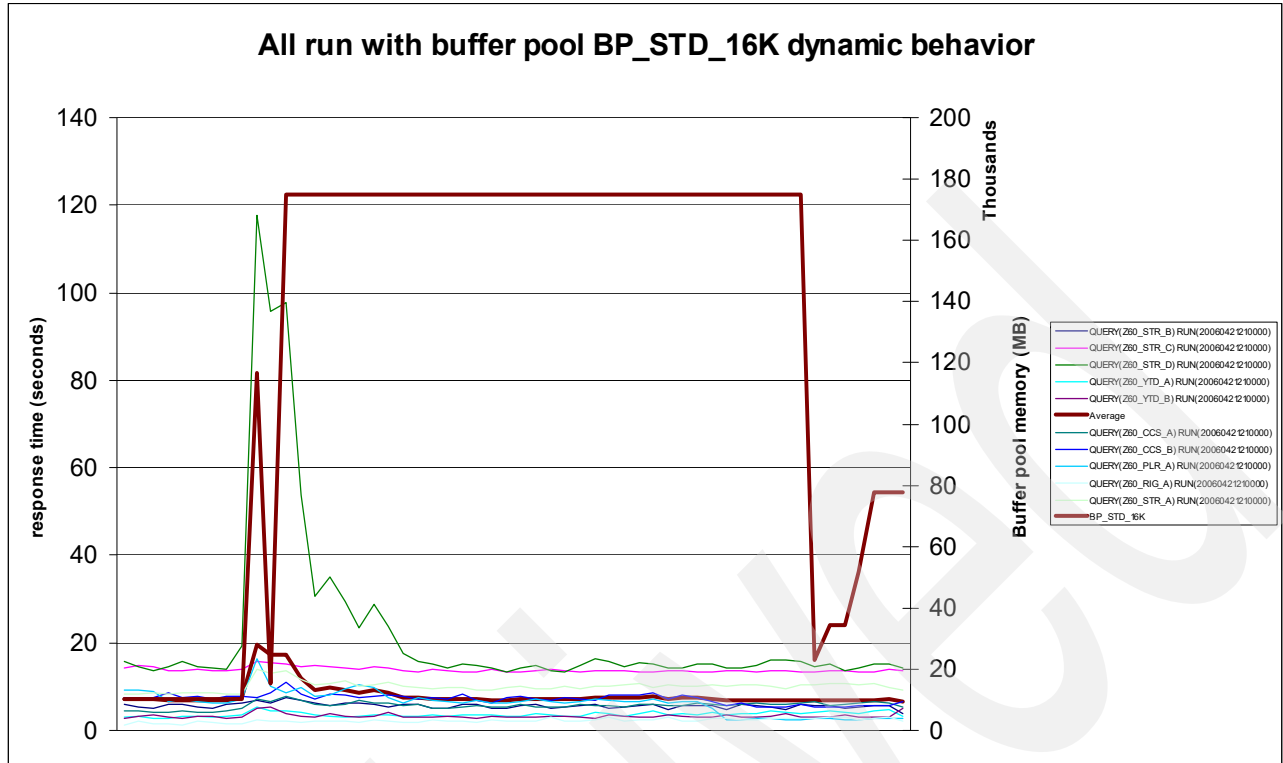


Figure 3-100 Queries response time and buffer pool dynamic size changes

The set of SAP query response time comparison tests with and without STMM over the high load phase is shown in Figure 3-101 on page 285. You can notice that response time without STMM looks more stable. This is because STMM tunes the system dynamically accordingly to the workload imposed on the database.

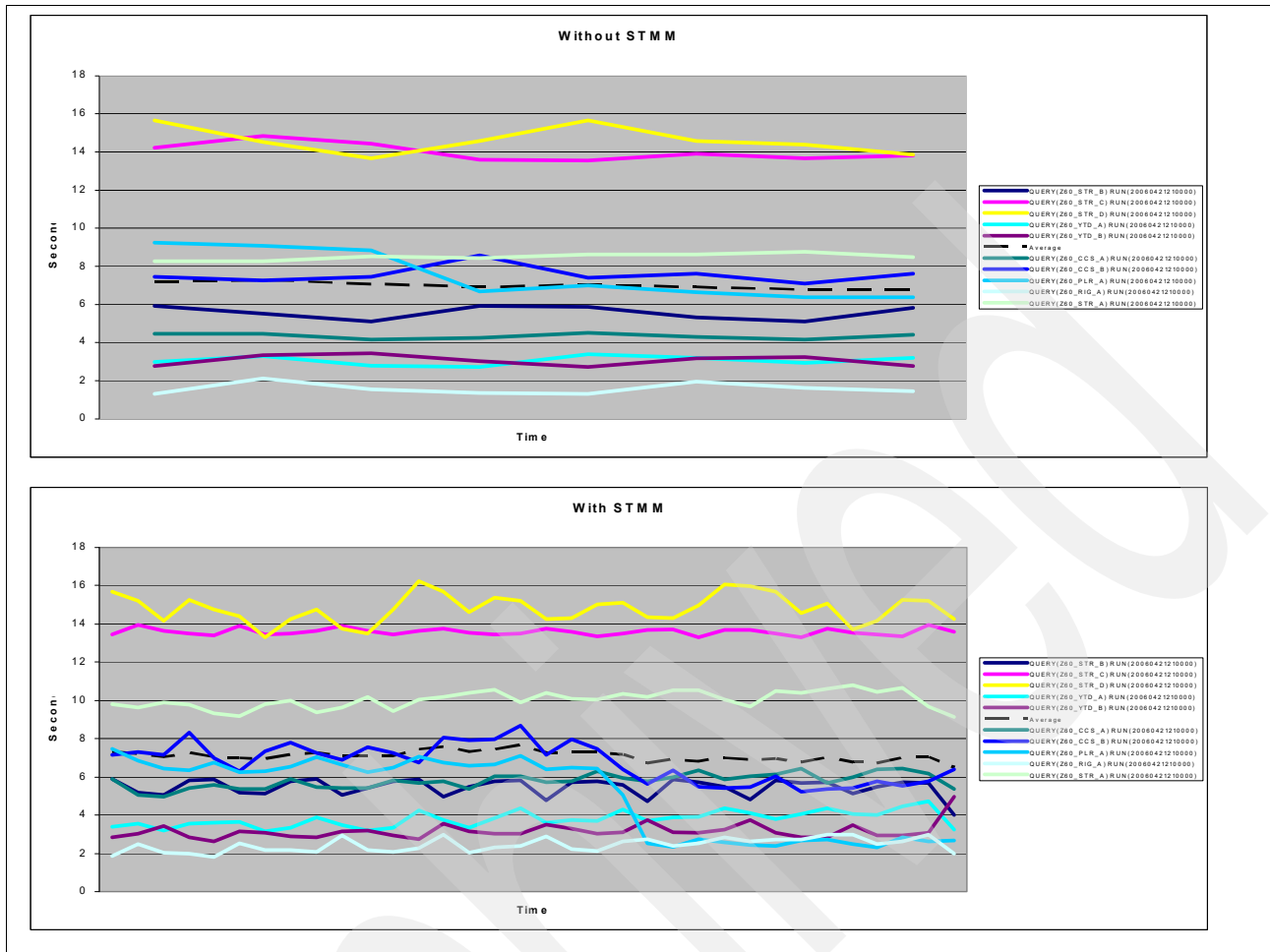


Figure 3-101 Response time comparison

For the number of transactions per second, a comparison of the results is shown in Figure 3-102. The same pattern can be seen using STMM. A slight oscillation can be seen compared to transactions per second on a manually tuned system.

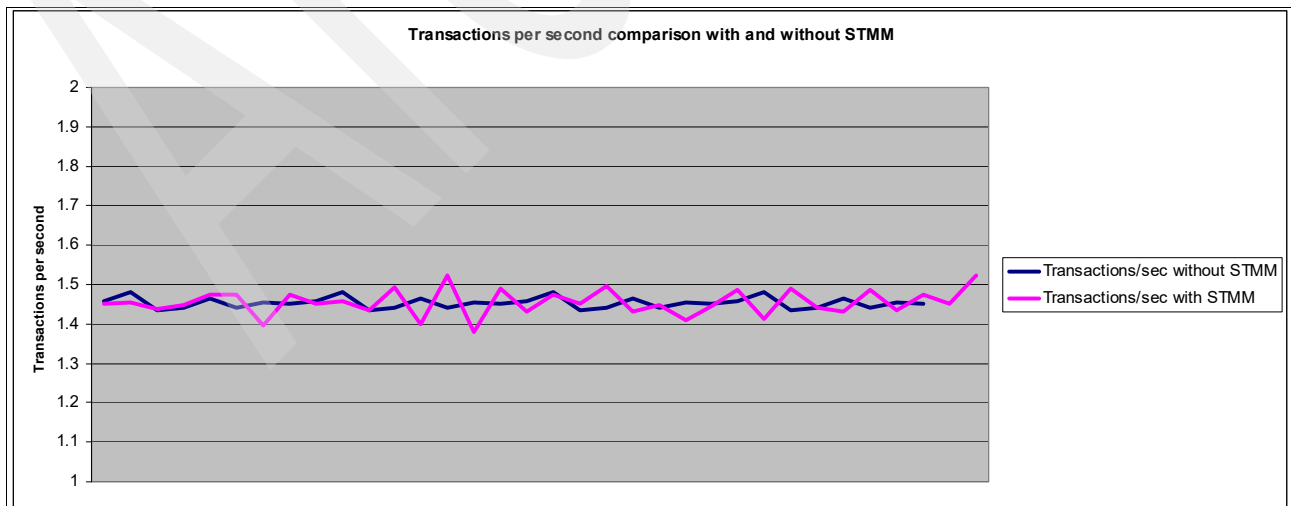


Figure 3-102 Transactions per second comparison with and without STMM

The final results and numbers from our test show a very small degradation. This can be seen in Table 3-29.

Table 3-29 STMM comparison with a tuned database

| Measure | Average/ maximum/ minimum | With STMM | Without STMM | Result |
|-------------------------|---------------------------------|--------------|--------------|----------------------|
| Response time | Minimum | 6.47 seconds | 6.77 seconds | Improvement of 4.69% |
| Response time | Average | 7.07 seconds | 6.96 seconds | Overhead of 1.57% |
| Response time | Maximum | 7.64 seconds | 7.26 seconds | Overhead of 4.9% |
| Transactions per second | Minimum | 1.38 | 1.43 | Overhead of 3.84% |
| Transactions per second | Average | 1.45 | 1.45 | Same results |
| Transactions per second | Maximum | 1.52 | 1.48 | Improvement of 2.77% |

For a large database and to help you control memory, you may chose to use the regular database tuning procedures. However, for smaller databases or databases that handle very distinct and volatile workloads, STMM may be a good option, since it will be able to dynamically change sizes for database memory pools, which is virtually impossible when using the regular database tuning procedure, where you would have to set up the database based on the most costly workload, and possibly oversize values for most of the remaining workload.



The IBM System p perspective

In this chapter, the following topics are discussed:

- ▶ Overview of the IBM System p5 hardware
- ▶ Overview of the Advanced POWER5 Virtualization features
- ▶ The hardware and operating system used in this project
- ▶ Storage used from an AIX standpoint
- ▶ Network from an AIX standpoint
- ▶ AIX kernel tunables and other settings
- ▶ Lessons learned

4.1 Introducing IBM System p model p595

This section introduces the System p for readers who are not familiar with this server. In particular, the simultaneous multi-threading and the virtualization features are highlighted because they were used in our tests.

4.1.1 The System p hardware

IBM System p5 servers provide an entire range of servers from a low-end p5-505 up to a high-end p595. At the time of writing this book, the 64-way p5-595 server is the new flagship of the product line, with nearly four times the commercial performance (based on rPerf estimates¹) and twice the capacity of its predecessor, the System p690.

These servers come with mainframe-inspired reliability, availability, and serviceability (RAS) capabilities and virtualization technology with breakthrough innovations, such as Micro-Partitioning™ technology. Micro-Partitioning technology allows as many as ten logical partitions (LPARs) per processor to be defined. Up to 254 virtual servers (LPARs) with a choice of AIX 5L, Linux, or i5/OS® operating systems can be configured in a single p5-595 server.

System p5-595 servers are equipped with advanced 64-bit 2.1 or 2.3 GHz POWER5+ processor cores packaged in multichip modules (MCMs) with eight cores per MCM providing SMP configurations from 16-way up to 64-way.

System p5-595 servers provide memory capacity from 8 GB up to 2 TB DDR2 SDRAM² running at 400 MHz or up to 1 TB DDR2 SDRAM running at 533 MHz.

Servers come with one I/O drawer, which can be expanded with eleven additional I/O drawers, each providing twenty 64-bit PCI-X³ slots and 16 disk bays.

For more information visit the IBM System p servers Web site at:

<http://www-03.ibm.com/systems/p/>

4.1.2 Simultaneous multi-threading

Simultaneous multi-threading (SMT) is a feature of the POWER5 processor that provides the ability for a single physical processor to simultaneously dispatch instructions from two hardware thread contexts. Simultaneous multi-threading is designed to take advantage of the superscalar nature of the POWER5 processor so that more instructions can be executed at the same time. The basic concept is that no single process can fully saturate the processor, which makes it possible to have two processes providing input for the processor simultaneously.

SMT is supposed to be used primarily in commercial environments where the speed of an individual transaction is not as important as the total number of transactions that can be performed.

¹ More about the relative performance metric for System p servers (rPerf) is available at:
<http://www-03.ibm.com/systems/p/hardware/rperf.html>

² More about DDR2 in System p servers is available at:
http://www-306.ibm.com/common/ssi/01X.wss?DocURL=http://d03xhttpc1001g.boulder.ibm.com/common/ssi/re_p_ca/7/897/ENUS106-577/index.html&InfoType=AN&InfoSubType=CA&InfoDesc=Announcement+Letters&panelurl=&paneltext=

³ Peripheral Component Interconnect Extended (PCI-X) is a computer bus.

When simultaneous multi-threading is activated, each physical processor appears as two logical processors to an operating system.

Note: Simultaneous multi-threading is supported on POWER5-based systems running an AIX 5L Version 5.3 operating system or a Linux operating system at an appropriate level. AIX 5L Version 5.2 does not support SMT.

Table 4-1 on page 311 provides an overview of a processor executing in single thread mode and an overview of a processor executing two threads in simultaneous multi-threading mode.

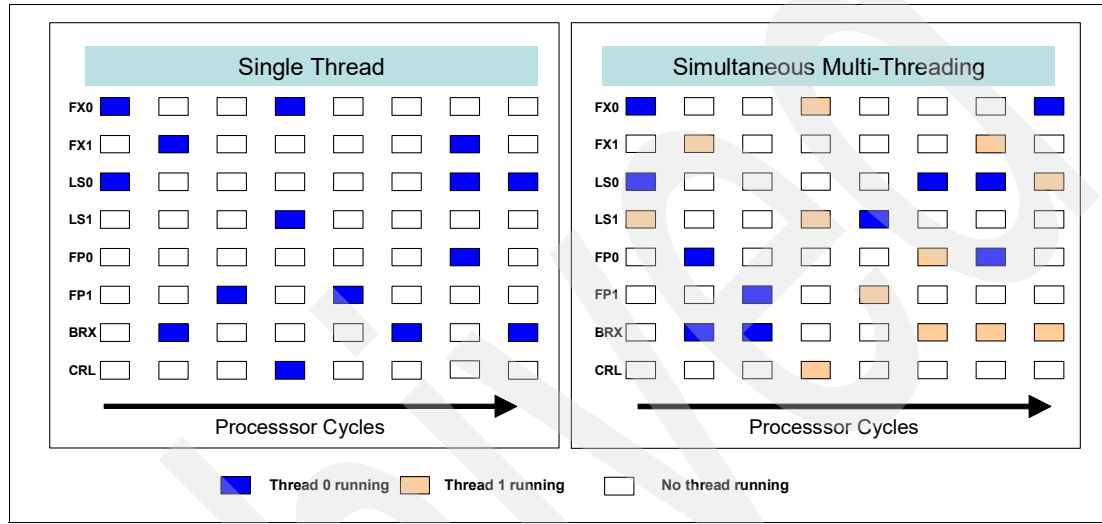


Figure 4-1 Single thread mode versus simultaneous multi-threading mode

4.1.3 POWER Hypervisor

The POWER™ Hypervisor is the foundation for virtualization on System p servers. It is a piece of firmware that resides in flash memory. POWER Hypervisor™ supports advanced functions like:

- ▶ Micropartitioning
- ▶ Virtual I/O
- ▶ Multiple operating systems on a single system
- ▶ Dynamic resource movement across partitions
- ▶ Capacity upgrade on demand

The POWER5 Hypervisor is the underlying control mechanism that resides below the operating system but above the hardware layer.

Note: POWER Hypervisor is mandatory on all POWER5-based systems. This includes single partition systems.

For more information about POWER Hypervisor read *Advanced POWER Virtualization on IBM eserver p5 Servers: Architecture and Performance considerations*, SG24-5768.

4.1.4 Virtualization features on IBM system p5

The virtualization features on POWER5 are called *Advanced POWER Virtualization (APV)* and are implemented in the POWER Hypervisor. The two essential functions of Advanced POWER Virtualization are *Logical Partitioning (LPAR)* and *Virtual I/O*.

Logical partitioning

Logical partitioning of a system allows more than one operating system's images to reside on the same system simultaneously without interfacing each other. There are two types of partitions on p5 systems and both types of partitions can coexist on the same system at any given time. The types of logical partitions are:

- ▶ Dedicated processor partition
- ▶ Shared processor partition or micro-partition

In addition to sharing processors, System p5 servers also provide sharing devices through virtual I/O.

On POWER5, systems logical partitions are created and managed by Hardware Management Console (HMC), which is a dedicated PC providing a graphical user interface for configuring and operating partitioned p5 systems. One HMC is capable of controlling multiple IBM System p servers.

Note: At the time of writing this publication, HMC is capable of managing up to 32 System p5 systems and up to 254 logical partitions.

Dedicated processor partitions

Dedicated processor partitions cannot share the processor with other partitions. These processors are owned by the partition on which they are running. The amount of processing capacity on the partition is limited by the number of processors configured in that partition and cannot be exceeded (unless you add more processors into the partition using a dynamic LPAR operation). By default, a powered-off logical partition using dedicated processors will have its processors available for use of other partitions.

When configuring dedicated processor partitions you define:

- ▶ The number of processors (as whole processors, one is minimum)
- ▶ The amount of memory
- ▶ The I/O resources

The memory assigned to partitions is dedicated when powered-on but can be dynamically moved to other partitions by DLPAR-operations and is available for other partitions when powered-off.

Shared processor partitions or micro-partitions⁴

Shared processor partitions, or micro-partitions, provide the ability to share physical processors among other shared processor partitions in the system. This allows system resources to be used more efficiently than in the case of dedicated processor partitions.

Shared processor partitioning is the mapping of *virtual processors* to physical processors. When configuring logical partitions, the virtual processors are assigned to partitions, not to physical processors. With the assistance of POWER Hypervisor, an entitlement or percentage of physical processor usage is granted to shared processor partitions. The minimum processor entitlement is 1/10 of a physical processor for a partition.

⁴ In this publication the terms *shared processor partition* and *micro-partition* refer to the same technology.

With fractional processor allocation, more partitions can be created in a given system, which enables clients to maximize the number of workloads that can be running on a server simultaneously.

The virtualization of physical processors on POWER5-based systems requires several new concepts and terminologies. The following terminology represents the types of processors and processor pools used in shared processor partitioning:

▶ Logical processor

This is a hardware thread, an operating system view of a managed processor unit. In AIX 5L V5.3, each hardware thread appears as a unique processor. The number of logical processors will be twice the number of virtual processors when simultaneous multithreading (SMT) is enabled.

▶ Virtual processor

This defines the way in which a partition's entitlement is spread over a physical processor. The virtual processor is the POWER Hypervisor dispatch unit. Currently, the maximum number of virtual processors per partition is 64.

▶ Physical processor

This is the actual physical hardware resource. Currently, the maximum number of physical processors in POWER5 systems is 64. The definition is the number of physical processor cores.

▶ Shared processor pool

A shared processor pool is a group of physical processors that are not dedicated to any logical partition. In a shared logical partition there are no fixed relationships between virtual processors and physical processors. Virtualization technology coupled with POWER Hypervisor facilitates the sharing of processing units in a shared processor pool between shared processor partitions.

▶ Capped mode

The guaranteed (entitlement) capacity given to a partition is guaranteed by the system and is never exceeded even if there are available processor resources in the shared pool.

▶ Uncapped mode

The processing capacity given to the partition at any given time may exceed the guaranteed processing capacity when there are resources available in the shared processing pool.

▶ Uncapped weight

If multiple uncapped partitions require idle processing units from a shared processor pool, processing units are distributed to requesting partitions in proportion to each partition's *uncapped weight*. The higher the uncapped weight of a partition, the more processing units it gets. The uncapped weight is an integer between 0 and 255. The default value is 128.

For more information about logical partitioning on POWER5 systems read *Advanced POWER Virtualization on IBM System p5*, SG24-7940-01.

Virtual I/O on POWER5-based systems

Virtual I/O provides the capability for a single I/O adapter to be used by multiple logical partitions on the same server, enabling consolidation of I/O resources and minimizing the number of required I/O adapters.

The POWER hypervisor provides the interconnection for the partitions by use of *virtual adapters*. A virtual adapter may or may not correspond to a physical adapter. Inter-partition communication using virtual Ethernet is based on virtual Ethernet adapters provided by POWER Hypervisor. While using Ethernet connections between partitions that reside on separate systems or while using *virtual SCSI* connections, a special hosting partition is needed as a link between virtual resources and physical resources. This hosting partition is called *IBM Virtual I/O Server* and is supported only on POWER5-based systems.

IBM Virtual I/O Server provides the following functions:

- ▶ Shared Ethernet Adapter

A Shared Ethernet Adapter (SEA) can be used to connect a physical Ethernet network to a virtual Ethernet network. It provides the ability for several client partitions to share one physical adapter. The SEA hosted in IBM Virtual I/O Server (VIOS) acts as layer-2 bridge between a virtual and an external network.

Figure 4-2 shows an example of a virtual Ethernet and a Shared Ethernet Adapter.

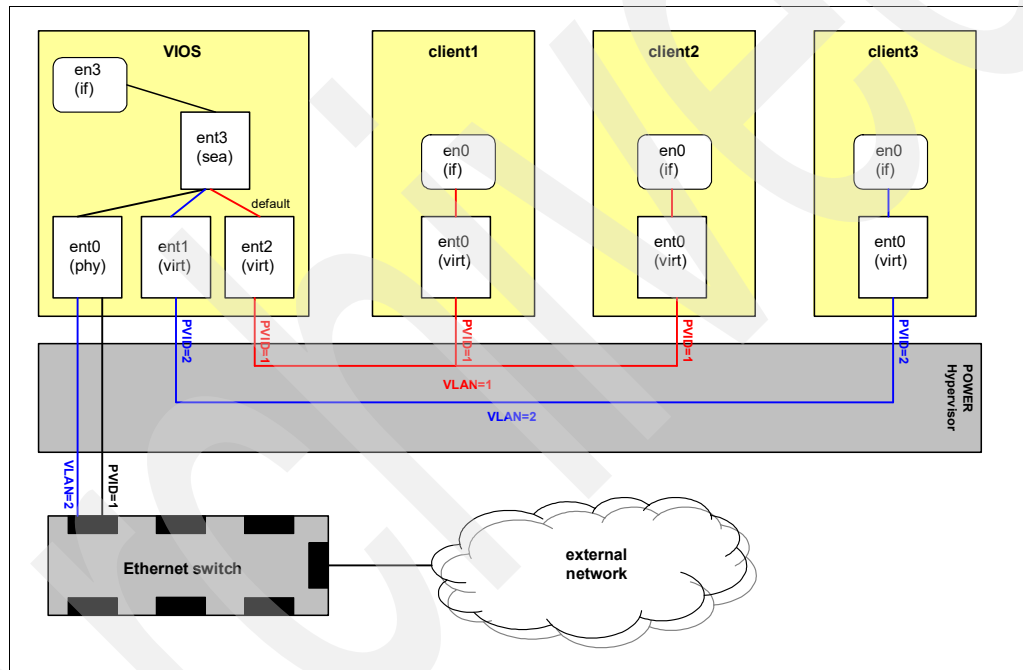


Figure 4-2 Virtual Ethernet and Shared Ethernet adapter

- ▶ Virtual SCSI

Virtual SCSI is a virtualized implementation of the SCSI protocol. Virtual SCSI is based on a client/server relationship. The Virtual I/O Server owns the physical resources and acts as a server. The logical partitions accessing the Virtual SCSI resources act as clients. Physical disks owned by the Virtual I/O Server can either be exported and assigned to a clients as whole, or can be partitioned into logical volumes. The logical volumes can then be assigned to different logical partitions.

Figure 4-3 shows an example where one physical disk is split into two logical volumes inside the Virtual IO Server. The logical volumes are assigned to one of the client partitions, which accesses them through a virtual SCSI client adapter. Inside the client partition the disks are seen as normal hdisks. For the other client partition the Virtual I/O Server assigns a physical disk as a whole. The client partition accesses the disk through a virtual SCSI client adapter as a normal hdisk.

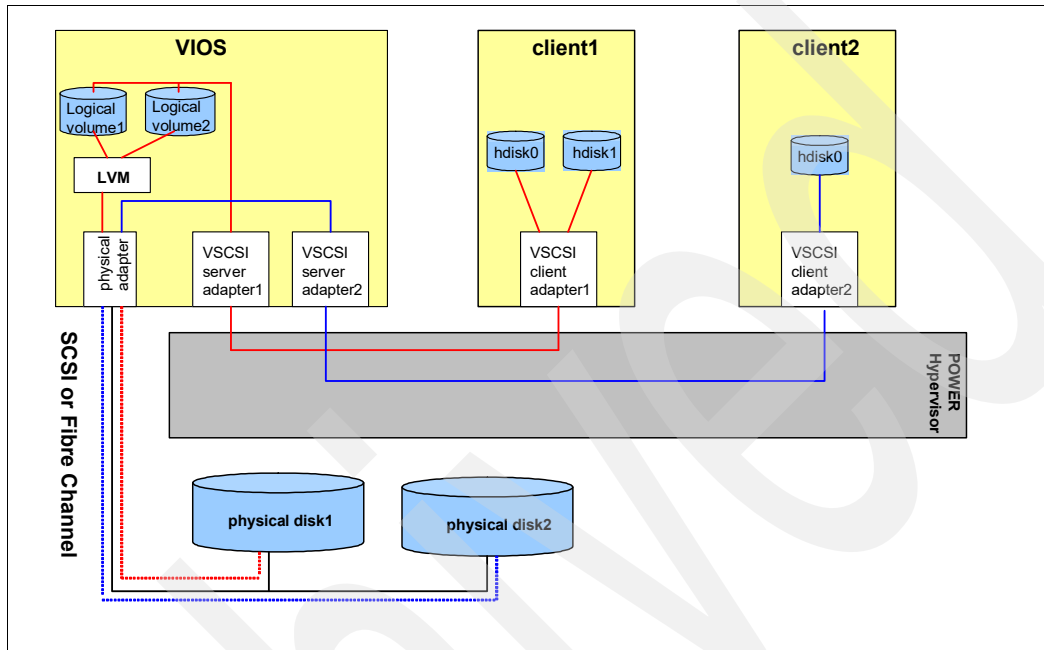


Figure 4-3 Overview of Virtual SCSI architecture

4.1.5 Operating systems

At the time of writing this publication, two versions of the IBM AIX 5L operating system and three versions of the Linux operating system are supported on System p5 systems:

- ▶ Supported AIX 5L operating system versions:
 - AIX 5L Version 5.2
 - Supports dynamic logical partitioning (DLPAR) on Systems p4 and p5 based systems
 - Does not support SMT
 - Does not support Advanced Power Virtualization (APV)
 - AIX 5L Version 5.3
 - Supports SMT
 - Supports all features of APV
- ▶ Supported Linux operating system versions
 - SUSE Linux Enterprise Server 9
 - SUSE Linux Enterprise Server 10
 - Red Hat Enterprise Linux AS 4

For more information about supported operating systems visit the IBM System p Web site:

<http://www-03.ibm.com/systems/p/>

4.2 The project environment

This section describes the IBM System p hardware used in this project phase, and demonstrates the scalability and manageability of the SAP NetWeaver BI infrastructure.

The objective in phase 1 was to test a 20 TB system running on one single p5-p595 and using one single DS8300 storage. In phase 1, no virtualization features of System p5 hardware were used. The distribution of CPU and memory resources was based on DLPAR operations. The series of tests at 20 TB were designed to provide the basis for the follow-on 60 TB system.

In phase 2 the logical architecture of the system was the same as in phase 1, but the physical architecture changed to five separate System p595 servers and four separate DS8300 storage systems. Shared processor partitions and Advanced POWER Virtualization were also utilized. Virtual I/O was not implemented.

The operating system used in phase 2 of the project was AIX 5L Version 5.3 TL5.

Figure 4-4 describes the high-level component model of the infrastructure used in this test. SAP connects via HTTP to Web-based front-ends or via TCP/IP connections to SAP GUI. From here all connectivity to the DB server is managed by Web application servers. One or more application servers were used during the tests to scale the solution.

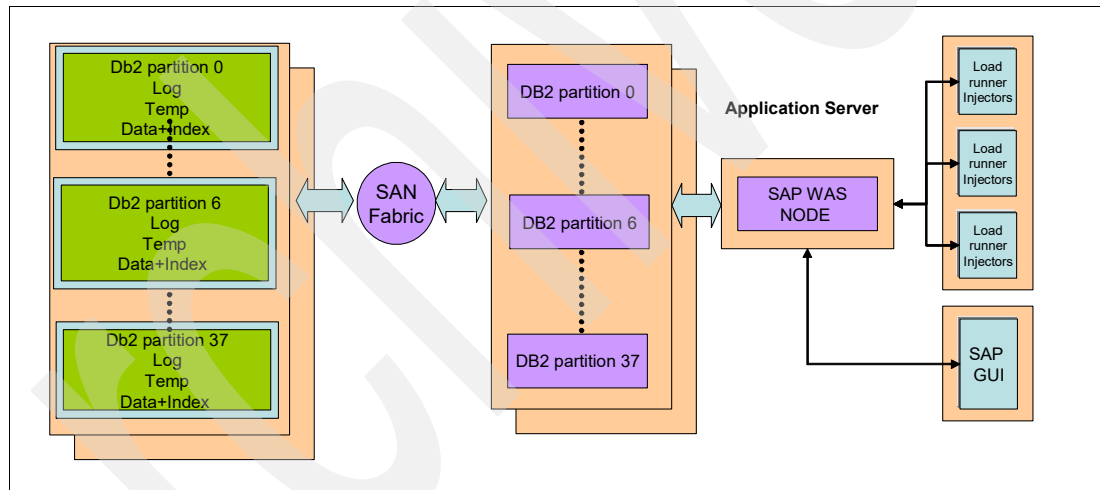


Figure 4-4 Logical architecture of 60 TB system

In the 60 TB physical architecture the components of the logical architecture were placed on five separate System p595 servers and on four separate DS8300 storage subsystems, instead of one System p595 and one DS8300 used in the 20 TB system (phase 1).

Figure 4-5 shows the server, storage, and SAN network components used in this project. We had five IBM p5 p595 servers dedicated to this proof of concept. The additional two p595s seen in the figure were housing TSM server LPAR and NIM server LPAR.

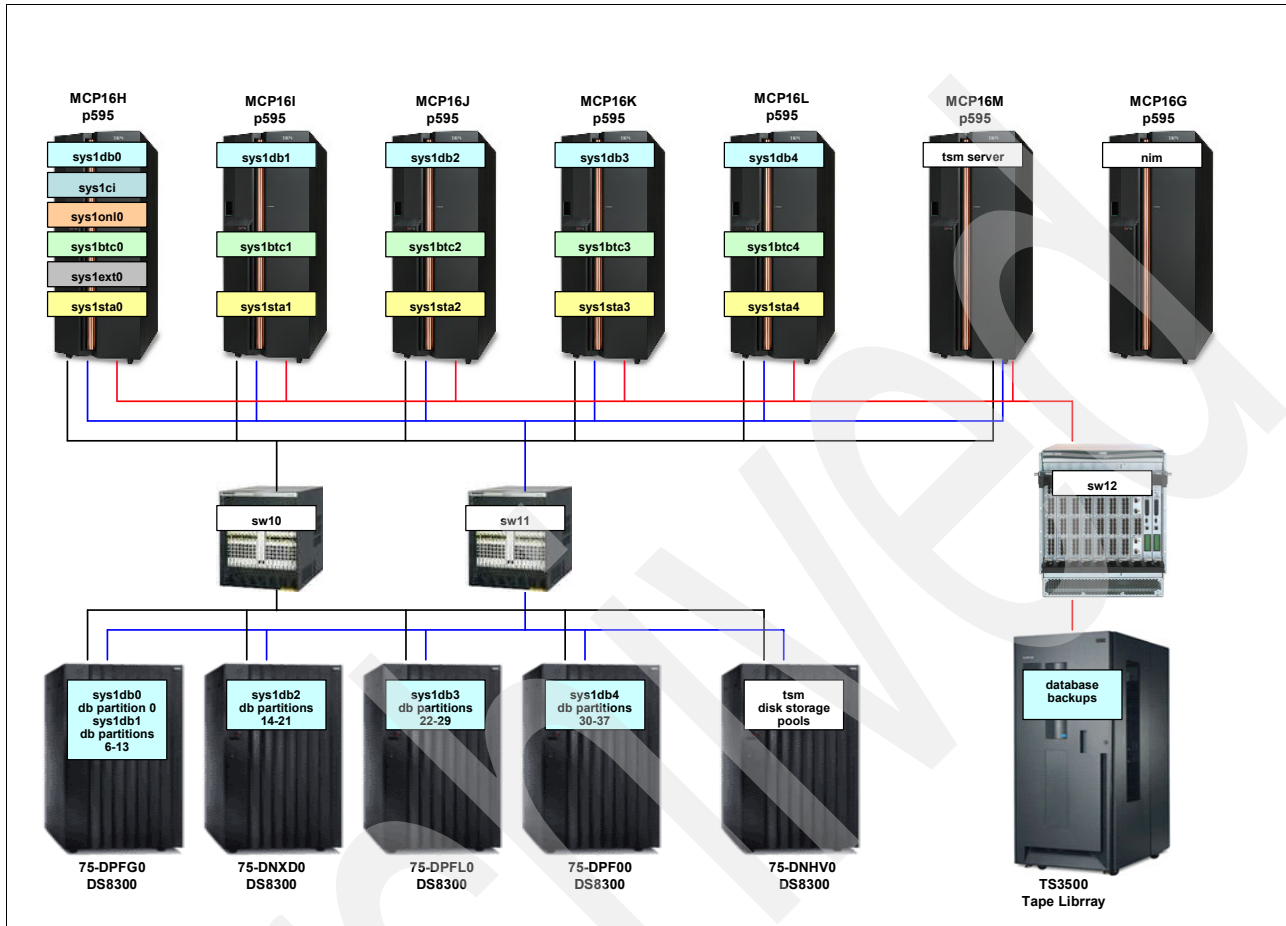


Figure 4-5 The physical architecture of 60 TB environment

Four IBM DS8300 storage subsystems were dedicated to the database. The fifth DS8300 seen in the figure provided storage for the disk pools of the Tivoli Storage Manager server.

There are two separate SAN networks in the figure — one for disk connections and one for connecting the tape drives to the LPARs running storage agents used to back up the database.

4.2.1 IBM System p595 servers and logical partitioning

In this project we used the latest p595 technology equipped with 2.3 GHz POWER5+ processors. Each of the five servers had 64 processors and 512 GB of memory. Logical partitioning was used to separate various load profiles. The CPU virtualization capability of POWER5 Advanced Power Virtualization (APV) was used to implement dynamic distribution of CPU resources based on predefined policies between logical partitions.

All logical partitions (except the TSM server and the NIM server) were configured as uncapped. The CPU usage was limited by the number of virtual processors and by the uncapped weight value of each partition.

On the first p595 system the following logical partitions (LPARs) were defined:

- ▶ One LPAR for SAP Central Instance (CI)
- ▶ One LPAR for DB2 partition 0 (catalog node)
- ▶ One LPAR for SAP online queries
- ▶ One LPAR for SAP batch (dedicated for aggregate)
- ▶ One LPAR for SAP batch (dedicated for data-load extractors)
- ▶ One LPAR for Tivoli Storage Agent (used as data mover for backups)

Each of the remaining four p595 systems had the following LPARs defined:

- ▶ One LPAR for DB2 server running eight DB2 partitions
- ▶ One LPAR for SAP batch (dedicated for data load)
- ▶ One LPAR for Tivoli Storage Agent (used as data mover for backups)

Figure 4-6 shows the detailed CPU, memory, and I/O resource configurations of logical partitions. It also shows the priority (uncapped weight) to the shared pool CPU resources of each LPAR.

| 1 | A | C | D | E F G | | | H | I | J | K | L M N | | | O | P | Q R | |
|----|--------|----------|-------------|-----------------------|-----|-----|-----------|------|-----------------|-------------------------|-----------|-----|-----|------------------------|------------------|------|--|
| | System | LPAR | IP-address | Processor Entitlement | min | des | max | Mode | Uncapped weight | # of virtual processors | Memory GB | | | # of ethernet adapters | # of FC adapters | | |
| 2 | | | | | | | | | | | min | des | max | | disk | tape | |
| 3 | | | | | | | | | | | | | | | | | |
| 4 | MCP16H | sys1ci | 10.3.13.72 | 0.1 | 3 | 64 | uncapped | 64 | 4 | 1 | 12 | 64 | 2 | | | | |
| 5 | MCP16H | sys1btc0 | 10.3.13.118 | 0.1 | 16 | 64 | uncapped | 32 | 42 | 8 | 96 | 128 | 4 | | | | |
| 6 | MCP16H | sys1db0 | 10.3.13.71 | 0.1 | 10 | 64 | uncapped | 128 | 20 | 4 | 64 | 128 | 10 | 4 | 4 | | |
| 7 | MCP16H | sys1ext0 | 10.3.13.68 | 0.1 | 6 | 64 | uncapped | 32 | 16 | 4 | 96 | 128 | 2 | | | | |
| 8 | MCP16H | sys1onl0 | 10.3.13.50 | 0.1 | 10 | 64 | uncapped | 64 | 32 | 8 | 90 | 128 | 4 | | | | |
| 9 | MCP16H | sys1sta0 | 10.3.13.81 | 0.1 | 4 | 64 | uncapped | 0 | 6 | 4 | 20 | 64 | 4 | 4 | 4 | | |
| 10 | | | | | | | | | | | | | | | | | |
| 11 | MCP16I | sys1btc1 | 10.3.13.119 | 0.1 | 32 | 64 | uncapped | 16 | 48 | 8 | 96 | 128 | 4 | | | | |
| 12 | MCP16I | sys1db1 | 10.3.13.73 | 0.1 | 14 | 64 | uncapped | 128 | 20 | 4 | 350 | 500 | 4 | 8 | 8 | | |
| 13 | MCP16I | sys1sta1 | 10.3.13.82 | 0.1 | 7 | 64 | uncapped | 0 | 8 | 4 | 20 | 64 | 4 | 8 | 8 | | |
| 14 | | | | | | | | | | | | | | | | | |
| 15 | MCP16J | sys1btc2 | 10.3.13.120 | 0.1 | 32 | 64 | uncapped | 16 | 48 | 8 | 96 | 128 | 4 | | | | |
| 16 | MCP16J | sys1db2 | 10.3.13.74 | 0.1 | 14 | 64 | uncapped | 128 | 20 | 4 | 350 | 512 | 4 | 8 | 8 | | |
| 17 | MCP16J | sys1sta2 | 10.3.13.83 | 0.1 | 7 | 64 | uncapped | 0 | 8 | 4 | 20 | 64 | 4 | 8 | 8 | | |
| 18 | | | | | | | | | | | | | | | | | |
| 19 | MCP16K | sys1btc3 | 10.3.13.121 | 0.1 | 32 | 64 | uncapped | 16 | 48 | 8 | 96 | 128 | 4 | | | | |
| 20 | MCP16K | sys1db3 | 10.3.13.76 | 0.1 | 14 | 64 | uncapped | 128 | 20 | 4 | 350 | 500 | 6 | 8 | 8 | | |
| 21 | MCP16K | sys1sta3 | 10.3.13.84 | 0.1 | 7 | 64 | uncapped | 0 | 8 | 4 | 20 | 64 | 4 | 8 | 8 | | |
| 22 | | | | | | | | | | | | | | | | | |
| 23 | MCP16L | sys1btc4 | 10.3.13.122 | 0.1 | 32 | 64 | uncapped | 16 | 48 | 4 | 96 | 128 | 4 | | | | |
| 24 | MCP16L | sys1db4 | 10.3.13.77 | 0.1 | 14 | 64 | uncapped | 128 | 20 | 4 | 350 | 500 | 6 | 8 | 8 | | |
| 25 | MCP16L | sys1sta4 | 10.3.13.85 | 0.1 | 7 | 64 | uncapped | 0 | 8 | 4 | 20 | 64 | 4 | 8 | 8 | | |
| 26 | | | | | | | | | | | | | | | | | |
| 27 | | | | | | | | | | | | | | | | | |
| 28 | MCP16M | tsm | 10.3.13.5 | 1 | 8 | 64 | dedicated | | | 2 | 128 | 256 | 6 | 16 | 16 | | |
| 29 | | | | | | | | | | | | | | | | | |
| 30 | MCP16G | nim | 10.3.13.101 | 1 | 3 | 64 | dedicated | | | 1 | 6 | 8 | 4 | | | | |

Figure 4-6 Definitions of logical partitions used in the project

In the project environment there were different workloads that had to be separated from each other. Therefore, an equal policy for distributing CPU resources was implemented on each System p595 system. The peaks in different load types led to effective CPU resource balancing even in high-load situations.

The database had the highest priority for CPU resources over all types of workloads. The next higher priority was given to SAP CI and online queries to guarantee good and constant online response times.

The aggregates drive the database, but are limited by the number of concurrent batch jobs running the aggregation process. The number of concurrent batch jobs is defined inside SAP. The aggregates had a priority that was lower than online queries but higher than data load.

The data load had two different workload profiles. On the extractor side it was batch oriented, on the load side massive parallel. The massive parallel part was prioritized lower than aggregates. It had relatively low CPU entitlement but a high number of virtual processors, allowing utilization of all unused CPU resources, as far as higher priority LPARs do not require them. The extractor side of data upload had higher priority than the load side.

Figure 4-7 shows the priorities for CPU resources in shared pool and resource distribution over logical partitions. The higher the priority (uncapped weight), the more CPU resources the LPAR will get from the shared pool if contention occurs.

| MCP16H p595 64 cpu, 512 GB mem | MCP16I p595 64 cpu, 512 GB mem | MCP16J p595 64 cpu, 512 GB mem | MCP16K p595 64 cpu, 512 GB mem | MCP16L p595 64 cpu, 512 GB mem | MCP16M p595 64 cpu, 512 GB mem |
|---|---|---|---|---|--|
| <p>sys1db0 db2 catalogue node 10 ent, vp 20, 64 GB mem uncapped weight 128</p> | <p>sys1db1 db2 partitions 6-13 14 ent, vp 20, 350 GB mem uncapped weight 128</p> | <p>sys1db2 db2 partitions 14-21 14 ent, vp 20, 350 GB mem uncapped weight 128</p> | <p>sys1db3 db2 partitions 22-29 14 ent, vp 20, 350 GB mem uncapped weight 128</p> | <p>sys1db4 db2 partitions 30-37 14 ent, vp 20, 350 GB mem uncapped weight 128</p> | |
| <p>sys1ci central instance 3 ent, vp 4, 12 GB mem uncapped weight 64</p> | | | | | |
| <p>sys1oni0 application server online queries 10 ent, vp 32, 90 GB mem, uncapped weight 64</p> | | | | | |
| <p>sys1btc0 application server aggregates 16 ent, vp 42, 96 GB mem uncapped weight 32</p> | <p>sys1btc1 application server uploads 16 ent, vp 48, 96 GB mem uncapped weight 16</p> | <p>sys1btc2 application server uploads 16 ent, vp 48, 96 GB mem uncapped weight 16</p> | <p>sys1btc3 application server uploads 16 ent, vp 48, 96 GB mem uncapped weight 16</p> | <p>sys1btc4 application server uploads 16 ent, vp 48, 96 GB mem uncapped weight 16</p> | |
| <p>sys1ext0 application server extractor 6 ent, vp 16, 96 GB mem uncapped weight 32</p> | | | | | |
| <p>sys1sta0 tivoli storage agent backups 4 ent, vp 6, 20 GB mem uncapped weight 0</p> | <p>sys1sta1 tivoli storage agent backups 7 ent, vp 8, 20 GB mem uncapped weight 0</p> | <p>sys1sta2 tivoli storage agent backups 7 ent, vp 8, 20 GB mem uncapped weight 0</p> | <p>sys1sta3 tivoli storage agent backups 7 ent, vp 8, 20 GB mem uncapped weight 0</p> | <p>sys1sta4 tivoli storage agent backups 7 ent, vp 8, 20 GB mem uncapped weight 0</p> | <p>tsm tsm server 8 cpu, 128 GB mem dedicated</p> |

Figure 4-7 Priorities for CPU resources in shared pool and resource distribution of logical partitions

The graph in Figure shows the CPU consumption of the DB2 server LPARs (sys1dbd1–sys1db4) as the number of physical processors during one of the KPI-G test runs. The DB2 load was evenly distributed over the four LPARs, and CPU consumption was most of the time below the entitled capacity of 14, although DB2 server LPARs had the highest priority to the shared pool.

Note: For more information about test results see Chapter 1, “Project overview: business objectives, architecture, infrastructure, and results” on page 1.

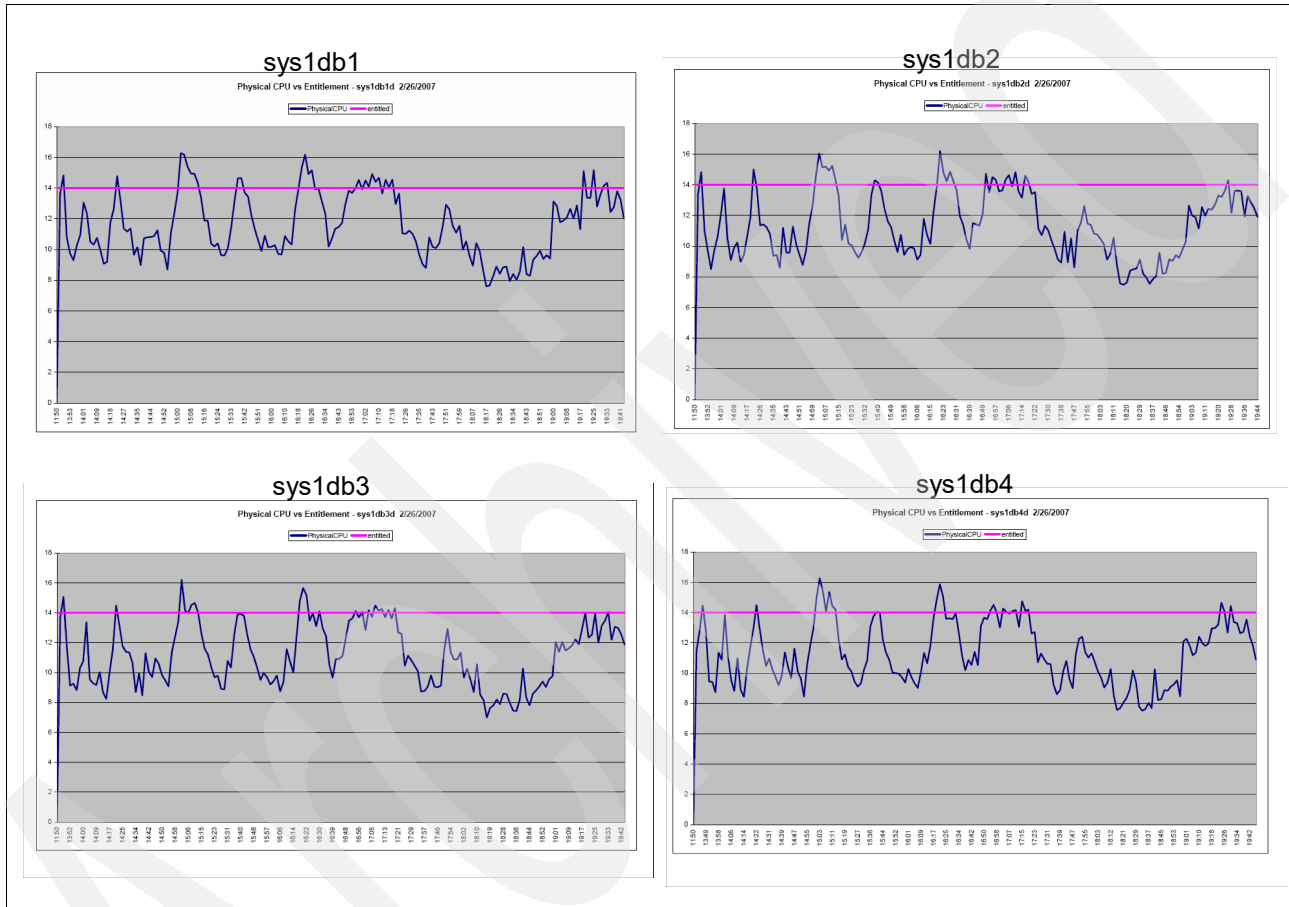


Figure 4-8 Physical CPU versus entitlement - DB2 servers sys1db1–sys1db4

The graph in Figure 4-9 shows the CPU consumption of LPARs (sys1btc1–sys1btc4) as the number of physical processors. These LPARs ran the parallel part of the data upload and had the lowest priority over all LPARs to the shared pool except the ones running backups. The CPU entitlement of these LPARs was relatively low, but the number of virtual processors was high. In the graph we can see that these LPARs took all unused capacity not required by higher priority LPARs from the shared processor pool.

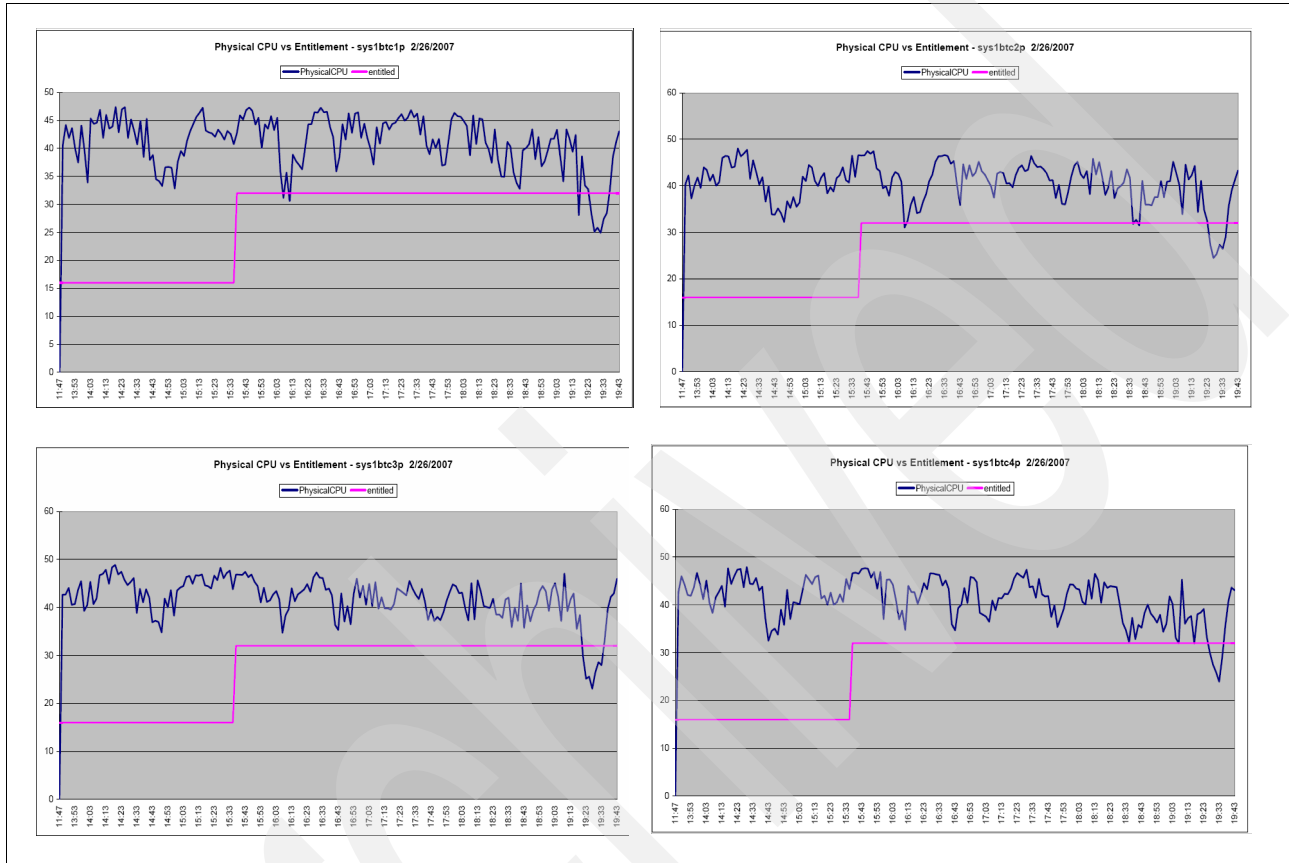


Figure 4-9 Physical CPU versus entitlement - application server LPARs sys1btc1–sys1btc4

The graph in Figure 4-10 shows the CPU consumption of the rest of the LPARs. The CPU consumption of sys1db0, the DB2 catalogue node, was stable during the entire run. There were no major peaks in CPU consumption, although the uncapped weight of the LPAR was high. The CPU consumption of sys1onl0, the online queries, varied. During the peak load it exceeded its entitlement by 40%. CPU consumption of the other three LPARs was relatively low.

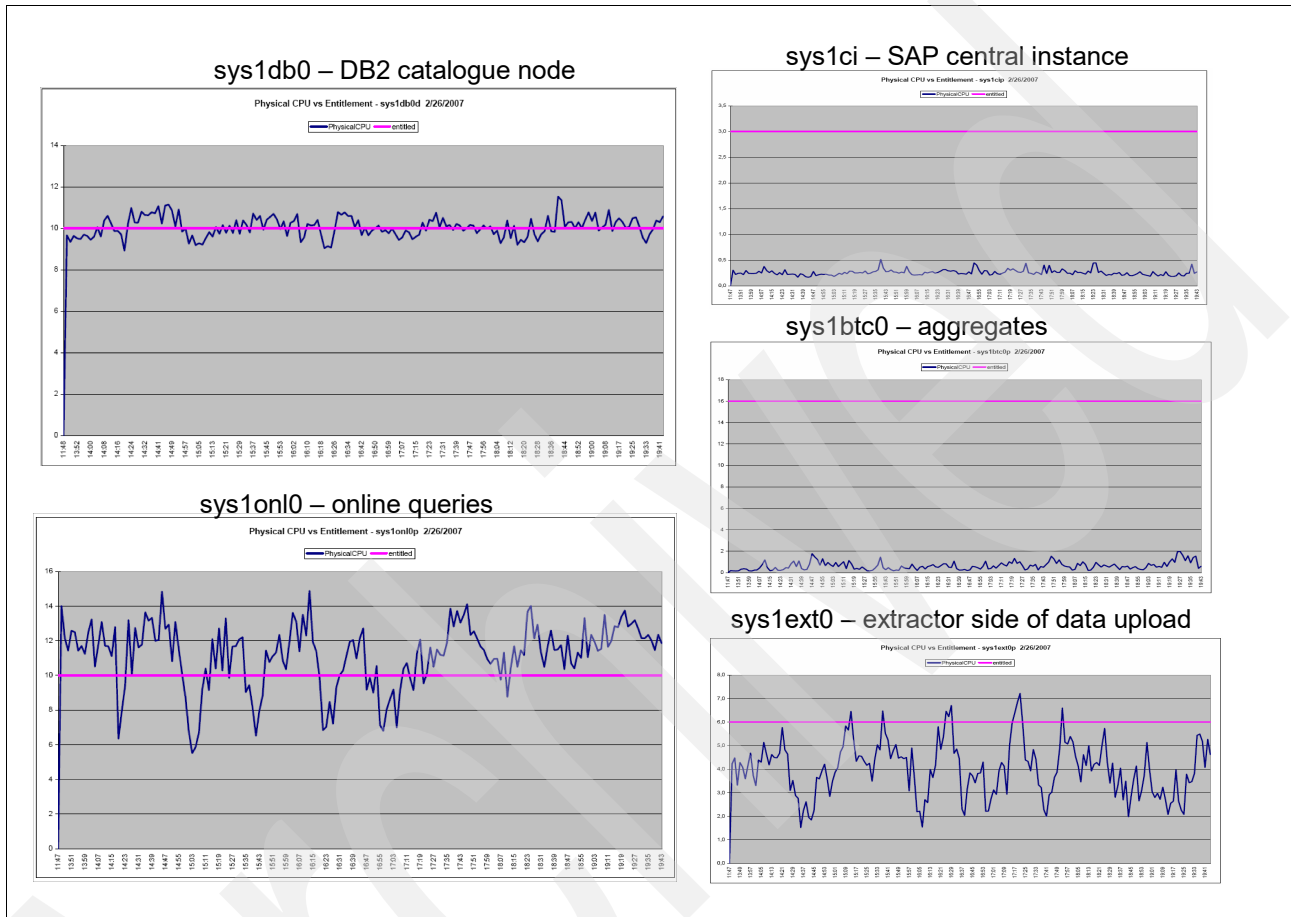


Figure 4-10 CPU consumption of LPARs sys1db0, sys1onl0, sys1ci, sys1btc0, and sys1ext0

The LPARs sys1sta0–sys1sta4 used as data movers for backups are covered in more detail in Chapter 6, “The Tivoli Storage Manager perspective” on page 347.

4.2.2 Storage and file systems

The target was to design and implement DB2 multi-partition shared nothing architecture. This means that each DB2 partition should have its own dedicated storage resources. An architecture like this provides the following advantages:

- ▶ It provides a nearly linear scalable architecture with the number of partitions.
- ▶ It increases DB2 parallelism and performance.
- ▶ It gives a flexible architecture by means of adding/moving DB2 partitions based on capacity needs.
- ▶ It gives the flexibility to move storage resources between LPARs and physical servers.

This section covers only the storage used by DB2 partitions, since all DS8300 storage was assigned to DB2 partitions. Each LPAR had two internal SCSI disks on which operating system and other system software was installed.

We assumed that the workload was equally distributed over all DB2 partitions and that I/O activity on each of the 32 DB2 partitions would be the same.

The storage of sys1db0 (DB2 catalog node) resided on the same DS8300 system as the storage of sys1db1 (DB2 partitions 6–13). All other DB2 servers had dedicated storage subsystems.

Figure 4-11 shows the mapping of storage between database server LPARs and appropriate DB2 partitions.

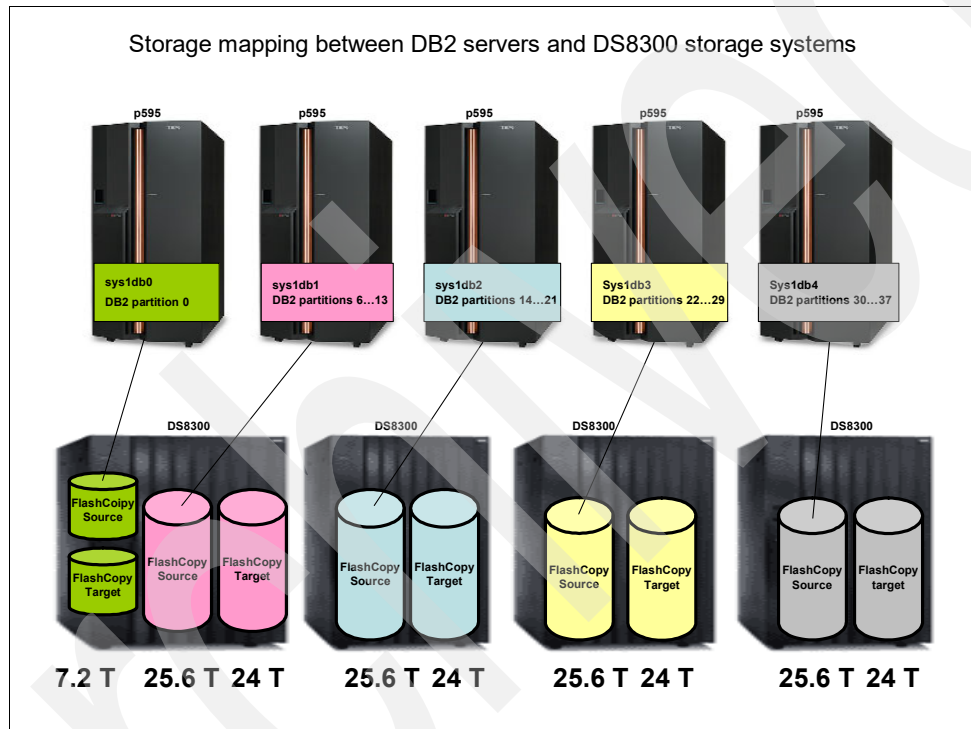


Figure 4-11 Storage mapping between database servers and DB2 partitions

The DB2 partition 0 (catalog node) had 16 dedicated disk arrays and other DB2 partitions (6–37) had eight dedicated disk arrays each. Three different sizes of LUNs were created as follows:

- ▶ 25 GB LUN for logs and temporary tablespaces
- ▶ 150 GB LUN for data and index (used on DB2 partition 0 only)
- ▶ 350 GB LUN for data and index (used DB2 partitions 6–37)

One of the targets of the design was to minimize the total number of LUNs or hdisks from the AIX standpoint.

The DS8300 LUNs appear to AIX as hdisks, each having eight paths over the SAN-network to the appropriate LUN on the storage subsystem. On AIX we used the Subsystem Device Driver Path Control Module (SDDPCM) for IBM System Storage DS8000 and DS6000 as the path control module for AIX MPIO to control multiple paths.

For more information about SDDPCM visit the SDD Web site:

<http://www-1.ibm.com/support/search.wss?rs=540&tc=ST52G7&dc=D600&dtm>

Figure 4-12 shows the layers of the I/O stack.

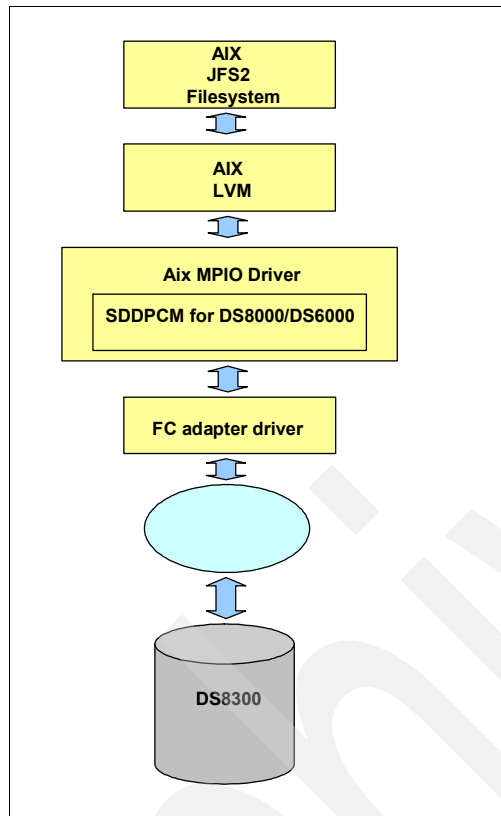


Figure 4-12 Layers of disk I/O stack

AIX Logical Volume Manager (LVM)

At the AIX LVM level we defined three *volume groups* for each DB2 partition. The physical partition (PP, an allocation unit inside a volume group) size was set to 64 MB on each volume group. For each DB2 partition the following volume groups were defined:

- ▶ DB2Pxx_DATA_VG for data and index
- ▶ DB2Pxx_TEMP_VG for temporary tablespaces
- ▶ DB2Pxx_CLOG_VG for database logs

For DB2 partitions 6–37 the logical volumes (and file systems on top of them) for data/index and temporary tablespaces were created on two hdisks in a way that physical partitions allocated from a volume group were evenly spread over the two hdisks. For DB2 logs the logical volume (and file system on top of it) was created on eight hdisks, spreading PPs evenly over all eight disks.

For the DB2 partition 0, the logical volumes and file systems were created in a similar way, with the exception that only the logical volume for DB2 logs had its PPs spread over all underlying disks.

AIX file system layout for DB2 partition 0 (sys1db0)

Twelve file systems for data and index tablespaces (sapdata1 to sapdata12) were defined. Each file system mapped to one 150 GB hdisk without LVM spreading (LV INTER-POLICY set to minimum, with a total number of 12 hdisks). Each file system had its own journal log (jfs2log). They were named:

- ▶ /db2/EB8/sapdata1/NODE00
- ▶ /db2/EB8/sapdata2/NODE00
- ▶ ..
- ▶ /db2/EB8/sapdata11/NODE00
- ▶ /db2/EB8/sapdata12/NODE00

Twelve file systems for temporary tablespaces (sapdatat1 to sapdatat12) were defined. Each file system mapped to two 25 GB hdisks without LVM spreading (LV INTER-POLICY set to minimum). Each file system has its own dedicated journal log (jfs2log) for a total number of 24 hdisks. They were named:

- ▶ /db2/EB8/sapdatat1/NODE00
- ▶ /db2/EB8/sapdatat2/NODE00
- ▶ ..
- ▶ /db2/EB8/sapdatat11/NODE00
- ▶ /db2/EB8/sapdatat12/NODE00

One file system for database logs mapped to 24 25 GB hdisks with LVM spreading (LVM INTER-POLICY set to maximum) was defined. It was named /db2/EB8/log_dir/NODE0000.

AIX file system layout for DB2 partitions 6–37 (sys1db1–sys1db4)

Each of the 32 DB2 partitions has eight dedicated disk arrays on one of the four DS8300 storage subsystems. On each array we have defined one 350 GB LUN and two 25 GB LUNs. To AIX they appear as one 350 GB hdisk and two 25 GB hdisks. For each of the 32 DB2 partitions, we defined the following file system structure:

- ▶ Four file systems for data and index
 - /db2/EB8/sapdata1/NODE00x
 - /db2/EB8/sapdata2/NODE00x
 - /db2/EB8/sapdata3/NODE00x
 - /db2/EB8/sapdata4/NODE00x

Each of these file systems had two 350 GB hdisks, which reside on separate DS8300 disk arrays. LVM spreading was used when creating underlying logical volumes for the file systems.

- ▶ Four file systems for temp
 - /db2/EB8/sapdatat1/NODE00x
 - /db2/EB8/sapdatat2/NODE00x
 - /db2/EB8/sapdatat3/NODE00x
 - /db2/EB8/sapdatat4/NODE00x

Each of these file systems had two 25 GB hdisks, which reside on separate DS8300 disk arrays. LVM spreading was used when creating underlying logical volumes for the file systems.

- ▶ One file system for logs
 - /db2/EB8/log_dir/NODE00x

This file system had 8 x 25 GB hdisks, which reside on separate DS8300 disk arrays. LVM spreading was used when creating underlying logical volumes for the file systems.

Figure 4-13 describes the storage mapping of data/index tablespaces between DB2, AIX, and the DS8300 storage subsystem. A tablespace is spread over four containers, each of which is mapped to an AIX file system. A file system is spread on two hdisks that reside on separate disk arrays.

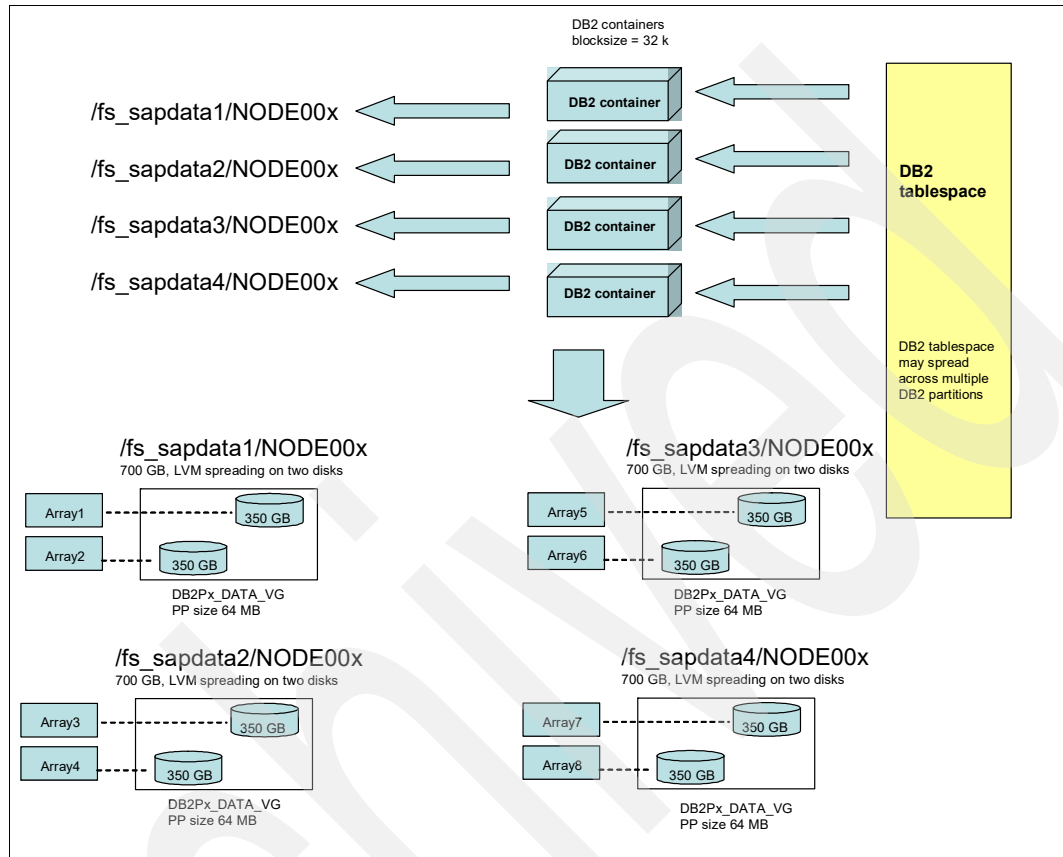


Figure 4-13 Mapping between DB2, AIX, and DS8300 storage subsystem

The DB2 catalog node had 25 file systems with a total of 60 hdisks, and DB2 partitions 6–37 each had nine file systems with 24 hdisks. In total, for DB2 we had 313 file systems and 828 hdisks residing on five LPARS on five separate p595 systems.

Figure 4-14 describes the disk I/O rate on the four DB2 server LPARs sys1db1–sys1db4 during a KPI-G test run. The I/O load was evenly distributed over all four DB2 servers and the peak I/O rates on each server were over 700 MBps, meaning that the aggregate peak I/O rate was 2.8 GBps. The maximum number of I/O operations per second during the peak load was slightly over 30,000 on each server.

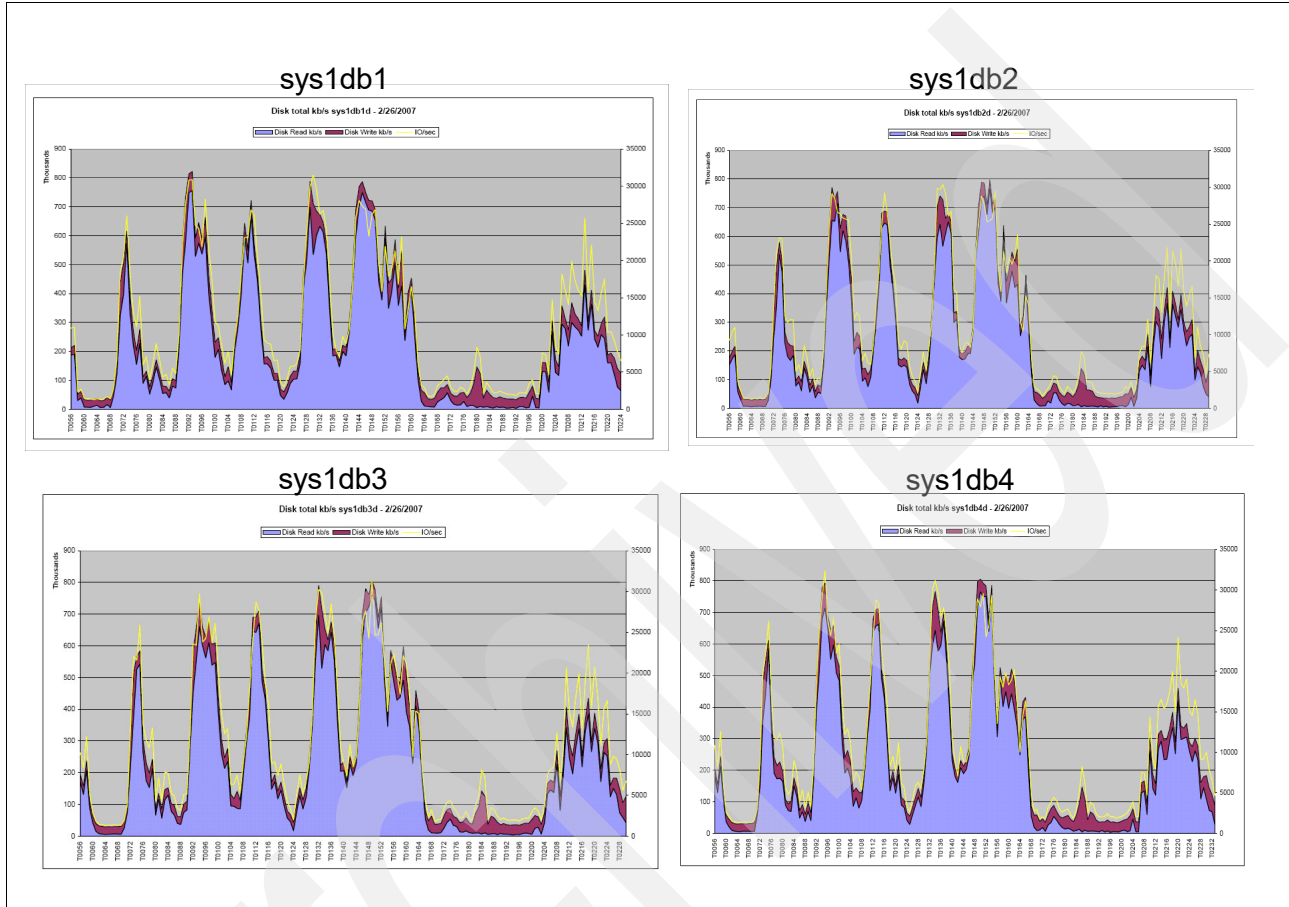


Figure 4-14 Disk I/O during a KPI-G test run on DB2 servers sys1db1–sys1db4

4.2.3 Network

Normally, in an SAP environment there are at least two networks, one for front-end and one for a backbone supporting server-to-server communications. This is for performance and also for security reasons, as you do not want users to have direct access to the database servers. Also, when various SAP components have dedicated networks, it is easier to monitor network load of a specific component.

Network architecture used in this project

We used three separate networks:

- ▶ A front-end network for the workload tools, administrators, Tivoli Storage Manager, and external connections to system: - MTU 1500
- ▶ A backbone1 network for communication between DB2 partition 0 (catalogue node) and other DB2 partitions: - MTU 9000
- ▶ A backbone2 network for communication between application servers and DB2 partition 0: - MTU 9000

Figure 4-15 describes this network architecture. The application servers connect to the DB2 partition 0 only. All client-oriented traffic went between the application servers and the DB2 partition 0. DB2 partition 0 had a unique role in the environment and functioned as a type of master coordinator for the other DB2 instances.

All communication between the DB2 partitions was between DB2 partition 0 and the other DB2 partitions.

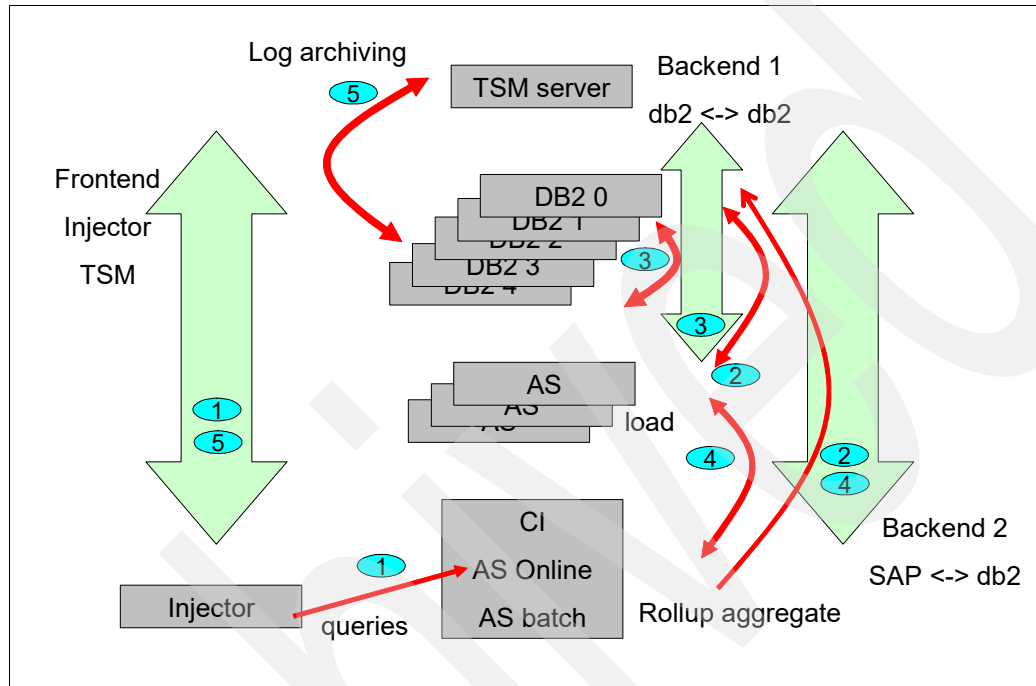


Figure 4-15 The network architecture

We used link aggregates (EtherChannel) of 4 GB Ethernet interfaces to connect DB2 partition 0 (sys1db0) to both of the backbone networks. The four other DB2 server LPARs were connected to the backbone1 network with a link aggregate of two Ethernet interfaces. The TSM server was connected to the front-end network with a two-interface EtherChannel connection.

Figure 4-16 describes how the LPARs were connected to the various network segments. In the picture several connectors from an LPAR to the same network describe an EtherChannel link aggregate.

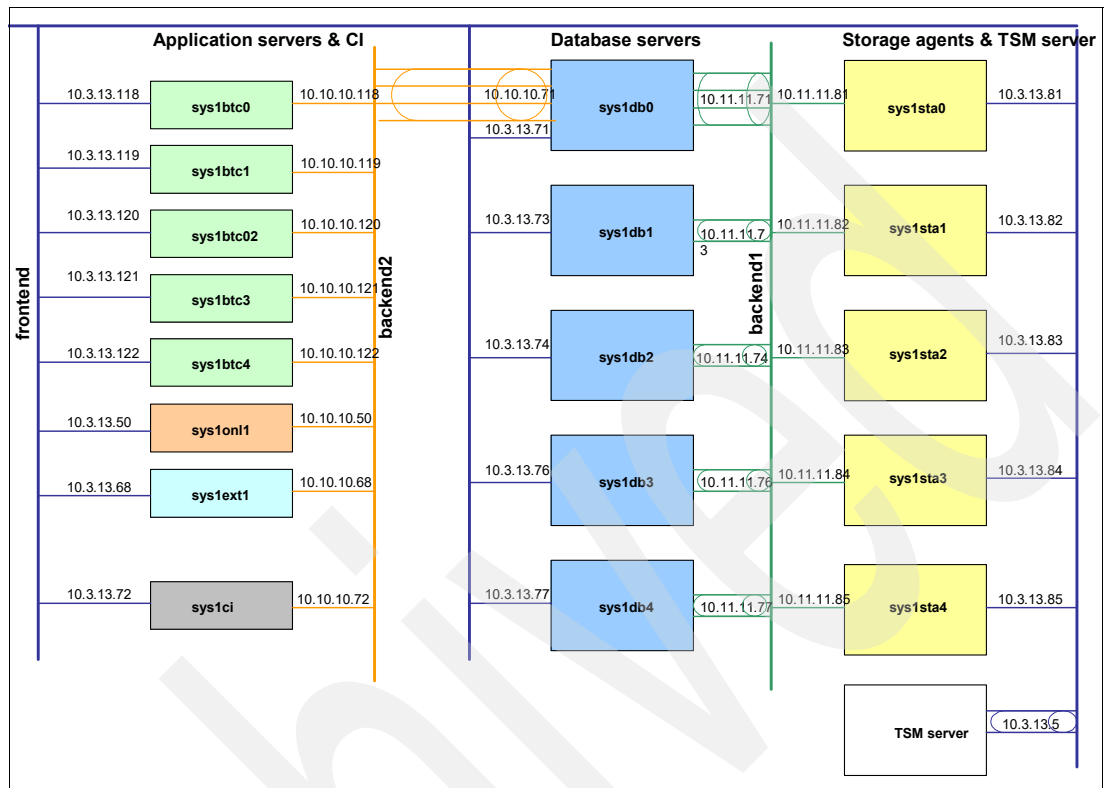


Figure 4-16 The network structure

Network hardware

In this environment we choose GB Ethernet as the base technology for LAN networks. The other possible network technologies were:

- ▶ 10 GB Ethernet
- ▶ InfiniBand®
- ▶ Ethernet adapter with RDMA⁵

Ethernet adapters

At server side we used 2-port Gigabit Ethernet adapters. The packet size is critical for performance. We used jumbo frames on backbone networks. They were set up at the with level.

Example 4-1 shows output of the `lscfg` command.

Example 4-1 Output of the `lscfg` command

```
sys1db0d:root}/root -> lscfg -vl ent0
ent0    U5791.001.91803F1-P1-C09-T1 2-Port Gigabit Ethernet-SX PCI-X Adapter
(14108802)
2-Port Gigabit Ethernet-SX PCI-X Adapter:
Part Number..... 03N6973
```

⁵ Remote Direct Memory Access (RDMA) allows data to move directly from the memory of one computer into that of another without involving either one's operating system. This permits high-throughput, low-latency networking, which is especially useful in massively parallel computer clusters.

```

FRU Number.....03N6973
EC Level..... H14078
Manufacture ID.....YL1021
Network Address..... 00096BEEECB2
ROM Level.(alterable).....DV0210
Device Specific.(YL).....U5791.001.91803F1-P1-C09-T1

```

Example 4-2 shows output of the `lsattr` command, which shows attributes of an GB Ethernet adapter. Those attributes that are in highlighted were changed from their default values.

Example 4-2 Output of the lsattr command

Example 4-3 lsattr -El ent0

```

alt_addr.....0x000000000000..Alternate ethernet address.....True
busintr.....180309.....Bus interrupt level.....False
busmem.....0xe8120000.....Bus memory address.....False
chksum_offload.....yes...Enable hardware transmit and receive checksum...True
opy_bytes.....2048.....Copy packet if this many or less bytes..True
delay_open.....no.Enable delay of open until link state is known ...True
failback.....yes.....Enable auto failback to primary.....True
failback_delay.....15.....Failback to primary delay timer..... True
failover.....disable.....Enable failover mode..... True
flow_ctrl.....yes.....Enable Transmit and Receive FlowControl...True
intr_priority.....3.....Interrupt priority.....False
intr_rate.....10000...Max rate of interrupts generated by..... True
jumbo_frames.....yes.....Transmit jumbo frames.....True
large_send.....no.....Enable hardware TX TCP resegmentation.....True
media_speed.....Auto_Negotiation Media speed.....True
rom_mem.....0xe80c0000...ROM memory address..... False
rx_hog.....1000..Max rcv buffers processed per rcv interrupt .....True
rxbuf_pool_sz.....2048.....Rcv buffer pool, make 2X rxdesc_que_sz.....True
rxdesc_que_sz.....1024.....Rcv descriptor queue size.....True
slih_hog.....10...Max Interrupt events processed per interrupt.....True
tx_que_sz.....8192.... Software transmit queue size .....True

```

LPAR sys1db0, which was running DB2 partition 0 (catalogue node), had two EtherChannel adapters defined, both having four underlying gigabit adapters. Other DB2 LPARs and the TSM server LPAR had one EtherChannel adapter each based on two underlying gigabit adapters.

Example 4-4 shows the output of the `lsattr` command against an EtherChannel adapter. Attributes that are in bold were changed from their default values.

Example 4-4 Output of lsattr command against an EtherChannel adapter

```

{sys1db0:root}/root -> lsattr -El ent10
adapter_names....ent0,ent4,ent6,ent8.....EtherChannel Adapters.....True
alt_addr.....0x000000000000.....Alternate EtherChannel Address...True
auto_recovery....yes.....Enable automatic recovery after failover True
backup_adapter...NONE.....Adapter used when whole channel fails....True
hash_mode.....src_dst_port....Determines how outgoing adapter is chosen True
mode.....8023ad.....EtherChannel mode of operation.....True
netaddr.....0.....Address to ping.....True
no_loss_failover...yes.....Enable lossless failover after pingfailure True
num_retries.....3.....Times to retry ping before failing.....True

```

```
retry_time.....1..... Wait time (in seconds) between pings....True
use_alt_addr.....no .....Enable Alternate EtherChannel Address....True
use_jumbo_frame...yes.....Enable Gigabit Ethernet Jumbo
Frames.....True
```

Ethernet switch

The Ethernet switch was Cisco 7609. Example 4-5 shows the EtherChannel configuration on the switch.

Example 4-5 EtherChannel configuration on switch

```
Global config of the switch
port-channel per-module load-balance
port-channel load-balance src-dst-port
Per portchannel config
interface Port-channel14
description 1/37 1/39 1/41 1/43
switchport
switchport access vlan 128
switchport mode access
mtu 9216
Per portchannel config
interface GigabitEthernet1/37
switchport
switchport access vlan 128
switchport mode access
mtu 9216
channel-group 14 mode active
interface GigabitEthernet1/39
switchport
switchport access vlan 128
switchport mode access
  mtu 9216
channel-group 14 mode active
interface GigabitEthernet1/41
switchport
switchport access vlan 128
switchport mode access
mtu 9216
channel-group 14 mode active
interface GigabitEthernet1/43
switchport
switchport access vlan 128
switchport mode access
mtu 9216
channel-group 14 mode active
```

Network load

We monitored the network during a KPI-G test run in order to see what are the typical I/O rates between the various components.

Figure 4-17 shows which I/O rates were the highest between DB2 partition 0 and the other DB2 partitions. There was also some load between the DB2 partition 0 and SAP application servers and between the DB2 partition 0 and the SAP Central Instance (CI). The network load on the remaining LPARs was not meaningful.

The network latency was not significant due to the dedicated Ethernet switch.

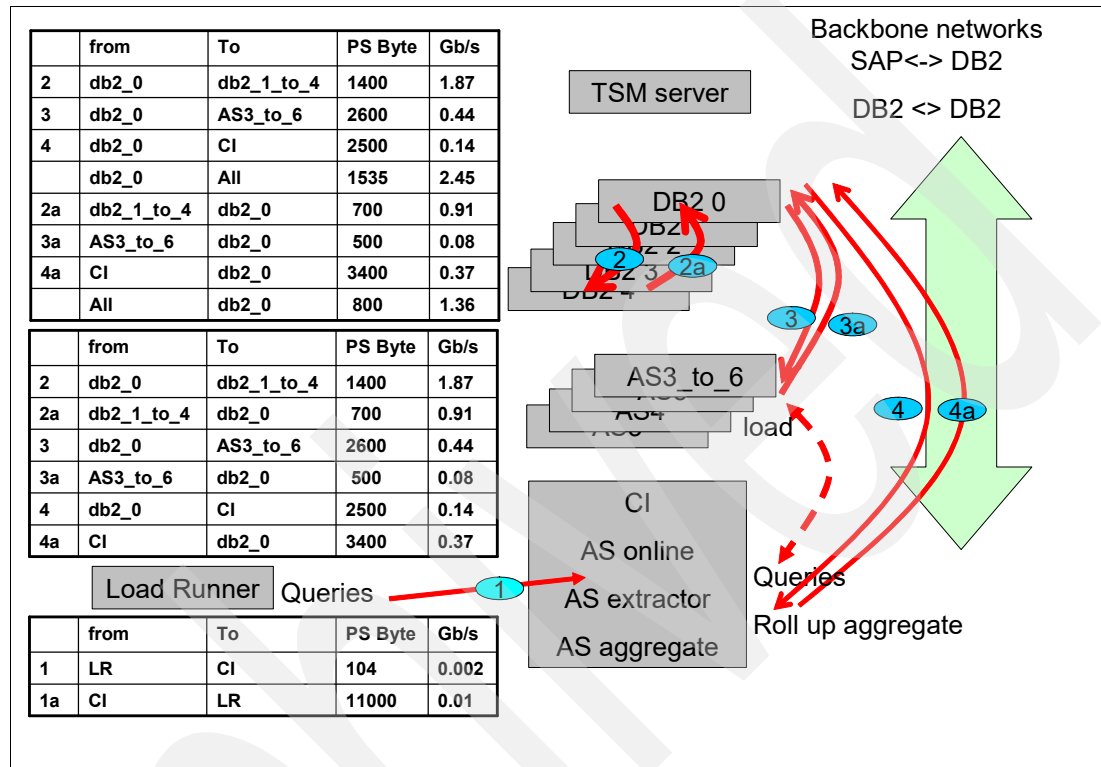


Figure 4-17 Results of network throughput study

4.3 Settings and options

This section lists the operating system settings and other options implemented in the project environment. These settings and recommendations are the results of tests in this project and from other large projects. They represent current recommendations from OSS (IBM for SAP), AIX recommendations, and typical settings at large SAP customer sites.

4.3.1 General options and recommendations

In this project environment only jfs2 file systems were used as sapdata file systems. For optimal control of memory behavior, mixing jfs and jfs2 should be avoided.

In our environment, concurrent I/O (CIO) was used for sapdata. For DB2 log file systems, CIO was not used, as it could cause archives and rollbacks to go slower. The log writers were writing synchronously to log files that were cached in the AIX file cache, which caused LRU to run constantly.

Using release-behind for writes would not work, since the writes were synchronous. However, using release-behind for reads helped because whenever the archiver reads a log file, the

pages got discarded from memory and reduced the need for LRU to run as often. Release-behind for reads was activated by the 'rbr' mount option for log file systems.

4.3.2 AIX kernel and network

Table 4-1 shows the kernel tunables that have been changed from their default values. Most of the settings are based on recommendations from SAP support or typical settings at large SAP customer sites.

Table 4-1 AIX kernel tunables used in this project

| Type | Tunable | Default value | Application servers | Database servers | Note |
|--------|--------------------------|---------------|---------------------|------------------|------|
| ioo | j2_MaxPageReadAhead | 128 | 512 | 512 | |
| ioo | j2_maxRandomWrite | 0 | 128 | 128 | |
| ioo | j2_minPageReadAhead | 2 | 32 | 32 | |
| ioo | j2_nBufferPerPagerDevice | 512 | 2048 | 2048 | |
| ioo | lvm_bufcnt | 9 | 16 | 16 | |
| ioo | Maxpgahead | 8 | 512 | 512 | |
| ioo | Maxrandwrt | 0 | 128 | 128 | |
| ioo | Minpgahead | 2 | 128 | 128 | |
| ioo | Numfsbufs | 196 | 4096 | 4096 | |
| ioo | sync_release_ilock | 0 | 1 | 1 | |
| schedo | vpm_xvcpus | 1 | -1 | -1 | 1) |
| vmo | lru_file_repage | 1 | 0 | 0 | |
| vmo | maxclient% | 80 | 8 | 7 | 3) |
| vmo | Maxfree | 1088 | 17920 | 2076 | |
| vmo | maxperm% | 80 | 10 | 10 | 3) |
| vmo | Minfree | 960 | 14336 | 1920 | |
| vmo | minperm% | 20 | 3 | 3 | |
| vmo | strict_maxclient | 1 | 0 | 1 | 3) |
| vmo | memory_affinity | 1 | 1 | 0 | 2) |
| no | lpqmaxlen | 100 | 250 | 250 | |
| no | nbc_pseg_limit | 3145728 | 20447232 | 20447232 | |
| no | rfc1323 | 0 | 1 | 1 | |
| no | sb_max | 1048576 | 1310720 | 1310720 | |
| no | tcp_nodelayack | 0 | 0 | 1 | |
| no | tcp_pmtu_discover | 1 | 0 | 0 | |
| no | tcp_recvspace | 16384 | 262144 | 262144 | |
| no | tcp_sendspace | 16384 | 262144 | 262144 | |

| Type | Tunable | Default value | Application servers | Database servers | Note |
|------|-------------------|---------------|---------------------|------------------|------|
| no | udp_pmtu_discover | 1 | 0 | 0 | |
| no | udp_recvspace | 42080 | 65536 | 65536 | |
| no | udp_sendspace | 9216 | 65536 | 65536 | |

Note 1

Virtual CPU (VCPU) folding is implemented to reduce the latency in the Hypervisor scheduler by logically deactivating VCPUs that are idle. Normally, one idle VCPU is deactivated per second. By setting `vpm_xvcpus` to “num”, when the activity starts, “num” of VCPUs are activated immediately, and then one VCPU is activated per second. For a burst type of activity, this value can be set to equal the number of VCPUs configured in the LPAR. So all VCPUs will be activated immediately when the activity starts.

Originally, the decision to turn off processor folding was for two reasons:

- ▶ Because we thought that the database pattern was CPU-bound.
- ▶ We were worried about a race condition with interrupts being sent to processors that were about to be folded (this has since been fixed).

However, in general, having processor folding can increase performance by reducing wasted dispatches of idle VCPUs, as well as reduce the chance of scalability issues in the applications when the number of configured VCPUs is high.

Note 2

To ensure that there was always enough memory pages on freelist, we set `maxclient%` and `maxperm%` to small values, and we set `strict_maxclient` to 1. With these settings there was always enough memory on the freelist. However, due to these settings, LRU was running constantly during the entire test run.

4.3.3 About `lsattr` parameters

Table 4-2 describes the `lsattr` parameters.

Table 4-2 Settings of `lsattr` tunables used in this project.

| Type | Tunable | Default value | Application servers | Database servers | Note |
|------|--------------|---------------|---------------------|------------------|------|
| aio | autoconfig | 0 | | available | |
| aio | maxreqs | 4096 | | 40960 | 1 |
| aio | maxservers | 10 | | 200 | 1 |
| aio | minservers | 1 | | 100 | |
| sys0 | maxuproc | 10000 | 5000 | 5000 | |
| entx | jumbo_frames | no | yes | yes | 2 |

Note 1

It is critical that enough AIO servers be available and that the `maxreqs` are sufficient. On AIX 5.3, the `iostat -AQ` command shows AIO statistics. The recommendation on AIX 5.3 is 20 AIO servers per CPU (per SMT thread).

Note 2

MTU 9000 on backbone networks only.

Note: For more information about AIX kernel tunables see *AIX 5L Practical Performance Tools and Tuning Guide*, SG24-6478.

4.4 Lessons learned

This section covers the experience gained and lessons learned when implementing some system tuning methods. There are also some observations of general system behavior made during the test runs.

4.4.1 Memory affinity and paging

In this section we discuss memory affinity and paging.

Recommendation

Our recommendations are:

- ▶ One of the first recommendations was to disable memory affinity by setting the vmo parameter `memory_affinity` to 0. This recommendation was made primarily to solve the issue of paging out to paging space during the runs. After memory affinity was disabled, it was noticed that paging was still there — perhaps not as frequently as before, but it was still there. Also, disabling memory affinity seemed to have little or no impact on the overall benchmark numbers.
- ▶ It was observed that there were frequent page-ins from paging space but no page-outs to paging space. Page-outs are important due to the fact that if we can prevent page-outs to paging space, the applications do not have to wait on page-ins. For that reason, it was important to determine the root cause of the page-outs.
- ▶ After changing the monitoring interval to 1 second in `vmstat` and monitoring the system for longer periods, it was observed that the page-outs occurred in bursts within a 1-second interval. At the same time the freelist had decreased up to 750 MB in some cases. Also at that time, the LRU activity was very high, as LRU was scanning millions of page frames per second to free up pages. Based on this observation, the conclusion is that the freelist was completely used by a huge allocation of memory. Perhaps a DB2 in-process heap sort was running. A check on the DB2 parameters indicated that up to 1 GB could be used for a private sort. When the freelist went down to 0 and if VMM had to backtrack, it is obvious that LRU had to steal computational pages.
- ▶ Assuming that this theory was correct, there were two tunables that could be implemented without reboot. Both methods had the same goal, which was to always have enough memory on the freelist to ensure that these periodic bursts of memory allocation would not cause the freelist to go to 0.
 - One way to do that is to tune `minfree` and `maxfree` so that `minfree` is large enough to accommodate these huge instantaneous allocations, which meant that there should be approximately 1 GB of free memory on the freelist. `minfree` had to be set so that there were always 250,000 pages on the freelist.

This method of tuning was not done because the number of CPUs per partition was not fixed, which could affect the number of memory pools.

- The other method was to decrease `maxclient%` and `maxperm%` values to 8 and 10, respectively, and to set `strict_maxclient` to the default value of 1. Previously, the values of `maxclient%` and `maxperm%` were set to 7 and 8 with `strict_maxclient` set to 0.

With this new tuning, there was always more than enough memory on the freelist, and the paging disappeared completely. However, we noticed that LRU was running constantly during the entire test run. The overall impact of these settings on benchmark performance was not noticeable.

Future recommendations

There is another tuning method that would require a reboot. This might be the best solution for LRU and the paging issue, but it requires further testing. This is to implement list-based-LRU and set `maxperm%`, `maxclient%`, `minfree`, and `maxfree` to system defaults. The idea here is that with list-based-LRU, it is much easier for `lrud` thread to find free pages, and therefore the freelist will not go to zero so often. Also, list-based-LRU will probably be the default in AIX 6.1. To enable it, set `vmo` parameter `page_steal_method` to 1. This requires a system reboot. The expectation is that there will be lower LRU overhead.

4.4.2 Tcp_NodelayAck

In this section we discuss `Tcp_NodelayAck`.

Recommendation

Our recommendation is to set `tcp_nodelayack` back to the default of 0 on all partitions except for the CI partition. Setting `tcp_nodelayack` can, in some cases, result in huge improvements in performance. In some cases it can hurt performance noticeably. And in other cases, it can make no difference at all. Setting `tcp_nodelayack` with the `no` command is dynamic and takes effect immediately.

Background

Setting `tcp_nodelayack` to 1 results in sending back an acknowledgment packet for every packet received. If it is set to 0, then an acknowledgment is sent back, either when we have received enough data (a multiple of the MTU size) or when the application has read the data.

The advantage of sending an ACK back immediately is that if the sender will not send another packet until it receives an ACK from the previous packet, then this could cause performance issues. However, if the sender disables the Nagle algorithm⁶ by setting `TCP_NODELAY` (which is not the same as `tcp_nodelayack`) on its socket, then the sender's packets should not get delayed on transmit.

The disadvantage of setting `tcp_nodelayack` is that the packet rate goes up significantly on the transmit side on our partitions. If you have many CPUs trying to send at the same time through the same adapter, then this could cause lock contention in the network driver. Also, on constrained networks or switches, the additional network traffic could cause further performance degradation.

Observations

By looking at the existing network connections with the `netstat -ao` command, it was observed that the connections were already setting `TCP_NODELAY` on the sockets. Without

⁶ Nagle's algorithm is a means of improving the efficiency of TCP/IP networks by reducing the number of packets that need to be sent over the network. Nagle's algorithm works by coalescing a number of small outgoing messages, and sending them all at once. Specifically, as long as there is a sent packet for which the sender has received no acknowledgment, the sender should keep buffering its output until it has a full packet's worth of output, so that output can be sent all at once.

knowing what was set on the workload tool machine, we chose to leave `tcp_nodelayack` as 1 on the CI partition until further tests could be performed.

We observed that setting `tcp_nodelayack` to the default value of 0 reduced the number of transmit packets by half. However, this does not seem to have any noticeable impact on the benchmark performance.

Future recommendations

Since it is not known what type of external systems will communicate with the CI partition, it may be better to leave that partition with the `tcp_nodelayack` as 1 unless some testing can be done first. For the other partitions, there has been no evidence to indicate that setting it to 1 helps, and in some cases it could hurt.

4.4.3 Use of large pages

In this section we discuss the use of large pages.

Recommendation

Enable the use of large (16 MB) pages for the DB2 buffer cache.

Background

CPU profiles collected on the DB2 partitions indicated that about 5% of the DB2 CPU time was spent in pinning and unpinning memory frames (lookup, pin, unpin routines). Also, time was spent in the locking code when the segments are locked for a pin. With large pages, there would be no need to pin a buffer, since the pages were already pinned. Also, there is the added advantage of reduced TLB misses by using large pages, which can improve performance even more by eliminating the extra instructions needed to do address translation (by looking up the address in the page frame table).

Observations

Large pages were initially implemented by setting the number of large page regions to 600. This would have provided 96 GB of large pages per partition.

The vmo parameters set were:

- ▶ `lgpg_regions=6000`
- ▶ `lgpg_size=16777216`
- ▶ `v_pinshm=1, maxpin%=90`

In addition, the following command was run to give the user permission to use large pages:

```
chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE <db2username>
```

It was observed that DB2 was using approximately 90 GB of buffer cache.

However, upon starting the database, the system started paging heavily. It turned out that DB2 had allocated over 105 GB of memory in its shared memory segments. So another test was done with 7,700 large pages (which required adding more real memory to each partition), and this time the system did not page.

Performance data gathered showed that the pin/unpin overhead went away almost completely. Also, hardware counter data showed that the TLB⁷ misses were reduced by 32% on the main DB2 partitions and up to 50% on the DB0 partition. Even though kernel time was reduced in the DB2 partitions, there was no noticeable impact to the overall benchmark performance.

⁷ Translation Lookaside Buffer, a memory buffer to improve the performance of virtual memory systems.

Future recommendations

Unless there is a significant benefit observed with large pages, we recommend that you do not use them because of the added complexities of system administration.

There may be some benefits of using medium (64 KB) pages, particularly on the batch partitions, since those tend to run heavily utilized. This is easy to do, as it only requires setting an environment variable. The environment variable could be set as follows:

```
LDR_CNTRL=DATAPSIZE=64K@TEXTPSIZE=64K@STACKPSIZE=64K
```

If batch performance is important, then we recommend using medium pages on those partitions. There may be a small benefit to using medium pages on the DB2 partitions as well, but given that the overall benchmark performance did not improve with large pages, it may not be enough to try unless there is sufficient time for a test.

4.4.4 Power5 virtualization

In our environment the use of Shared Processor Partitions (Micropartitioning) was planned from the beginning. In an environment like this where there are many different workloads running on separate LPARs, CPU virtualization provides a good way of optimizing and balancing CPU usage dynamically between the logical partitions.

The use of Shared Processor Partitions had some overhead. An average of 10% of CPU time was spent on the POWER Hypervisor (measured on DB2 LPARs). This overhead is small compared to benefits gained by the dynamic distribution of CPU capacity between the LPARs.

4.4.5 JFS2 lock contention

There was some lock contention on the JFS2 dioCache pile lock. Unfortunately, there is no resolution for this. The more CPUs there are, the worse this could be.

4.4.6 DB2 FCM

We observed that in DB2 V8, db2fcmd processes were consuming several percent of the CPU time because on the select system calls they were executing with a 1 millisecond timeout value. DB2 V9 splits up the tasks of the FCM processes into multiple processes so that it should not have to loop in select, so there should be some improvement in DB2 9.



The system storage perspective

This chapter discusses the following:

- ▶ An introduction to the storage products used in the project
- ▶ A description of the project environment
- ▶ A discussion of options and parameters
- ▶ Results of the project goals as influenced by the storage
- ▶ A discussion of lessons learned

5.1 Introducing the system storage

IBM storage products were instrumental in achieving the test objectives. The project members chose IBM System Storage DS8300 for the disk storage, IBM System Storage TS3500 Tape Library for the tape automation, IBM System Storage TS1030 Tape Drive for the LTO-3 tape drive, and IBM TotalStorage SAN256M and IBM TotalStorage SAN140M for the SAN switches.

The storage infrastructure was sized to ensure that it would not be the limiting factor in the performance efforts. More storage was available than was used in achieving the test objectives. Not all of the DDMs in the DS8300s nor all the LTO-3 tape drives in the IBM TS3500 library were utilized. At no point in any of the tests was the tape or disk storage the limiting factor in the performance runs.

The storage infrastructure was connected with 2 GB Fibre Channel switch ports and adapters. At the start of the project, 4 GB Fibre Channel was not widely available. If one were to put together a similar environment, but using 4 GB technology, the number of storage devices, interconnections, and switch ports could be reduced, while achieving the same or better performance.

5.1.1 Disk - IBM System Storage DS8300

Many different models and features exist in IBM System Storage DS8000 series of products. This project utilized the DS8300 Turbo Model 932. The major licensed feature exploited in this project for the backup methodology was the DS8300 Copy Services FlashCopy function. For more details about the DS8000 series of products, refer to *IBM System Storage DS8000 Series: Architecture and Implementation*, SG24-6786.

The DS8300 Turbo Model 932 features a dual four-way processor complex implementation and includes a Model 932 base frame and up to two optional Model 92E expansion frames. A diagram of a DS8000 unit is provided in Table 5-1 on page 335. In the first frame, there is space for a maximum of 128 disk drive modules (DDMs), and each expansion frame can contain 256 DDMs. Each of the five DS8300s utilized in the project included the two optional Model 92E expansion frames, for a total of 640 available DDMs in each DS8300.

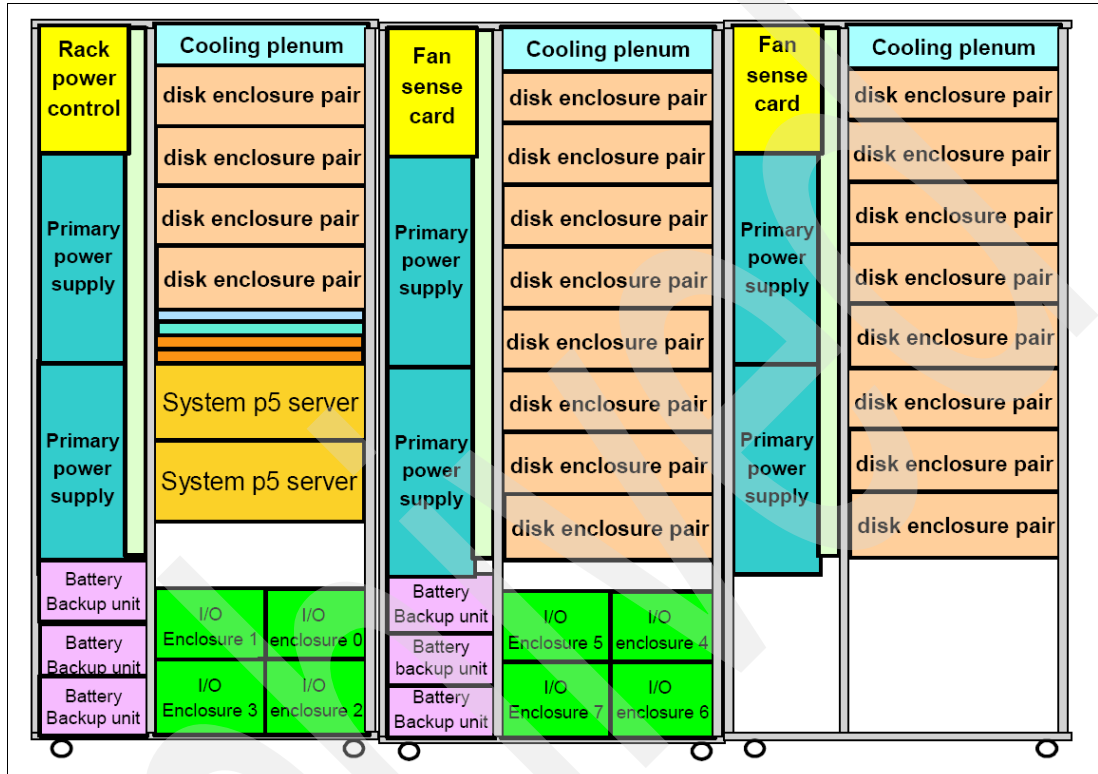


Figure 5-1 DS8300 showing major components

A DS8300 Turbo Model 932 can support up to a maximum of 32 host adapters, which provide up to 128 Fibre Channel/FICON ports. Each DS8300 had eight quad-ported 4 GB Fibre Channel cards. Not all the ports available were used, and the ports in use were attached to a 2 GB switch port. This is described in greater detail in 5.2, “The project environment” on page 324.

The left side of the base frame (viewed from the front of the machine) is the frame power area. Only the base frame contains rack power control cards (RPC) to control power sequencing for the storage unit. It also contains a fan sense card to monitor the fans in that frame. The base frame contains two primary power supplies (PPSs) to convert input AC into DC power. The power area also contains two or three battery backup units (BBUs), depending on the model and configuration. The base frame can contain up to eight disk enclosures, each of which can contain up to 16 disk drives. In a maximum configuration, the base frame can hold 128 disk drives. Above the disk enclosures are cooling fans located in a cooling plenum. Between the disk enclosures and the processor complexes are two Ethernet switches, a Storage Hardware Management Console (an S-HMC), and a keyboard/display module.

The base frame contains two processor complexes. These System p5 servers contain the processor and memory that drive all functions within the DS8000. Finally, the base frame contains four I/O enclosures. These I/O enclosures provide connectivity between the adapters and the processors. The adapters contained in the I/O enclosures can be either

device adapters (DAs) or host adapters (HAs). The communication path used for adapter to processor complex communication is the RIO-G loop. This loop not only joins the I/O enclosures to the processor complexes, it also allows the processor complexes to communicate with each other.

The left side of each expansion frame (viewed from the front of the machine) is the frame power area. The expansion frames contain a fan sense card to monitor the fans in that frame. Each expansion frame contains two PPSs to convert the AC input into DC power. Finally, the power area can contain three BBUs, depending on the model and configuration. Each expansion frame can hold up to 16 disk enclosures, which contain the disk drives. They are described as 16-packs, because each enclosure can hold 16 disks. In a maximum configuration, an expansion frame can hold 256 disk drives. Above the disk enclosures are cooling fans located in a cooling plenum. An expansion frame can contain I/O enclosures and adapters if it is the first expansion frame that is attached to a 9x2 model. The second expansion frame in a 9x2 model configuration cannot have I/O enclosures and adapters, nor can any expansion frame that is attached to a model 9x1. If the expansion frame contains I/O enclosures, the enclosures provide connectivity between the adapters and the processors. The adapters contained in the I/O enclosures can be either device or host adapters.

5.1.2 Library - IBM System Storage TS3500 Tape Library

IBM System Storage TS3500 Tape Library, formerly known as IBM TotalStorage 3584 Tape Library, is designed for medium to large automated tape storage and backup solutions and is part of an entire family of tape libraries for small to large tape automation solutions. Originally designed for IBM Linear Tape Open Ultrium Technology drives, the library was enhanced to also support the IBM TotalStorage Enterprise Tape Drive 3592 in preparation of System z™ attachment. Together with IBM System Storage 3953 Tape System, the TS3500 Tape Library can attach to System z as well as to open systems hosts.

Many different configurations are possible with the TS3500 library. For more details on the TS3500, refer to *IBM System Storage Tape Library Guide for Open Systems*, SG24-5946-04, or *IBM TS3500 Tape Library with System z Attachment: A Practical Guide to TS1120 Tape Drives and TS3500 Tape Automation*, SG24-6789. The tape library used in this project consists of one L53 frame and five D53 frames. The L53 frame and one of the D53 frames had twelve LTO-3 drives installed. The other four D53 frames had 11 LTO-3 drives installed. Sixty-eight LTO-3 drives were available. However, only 32 drives were utilized in achieving the performance targets.

The components shown in Figure 5-2 are:

1. Library frame
2. x-Rail system
3. Cartridge accessory with optimized dual-gripper transport mechanism
4. Accessor controller
5. Cartridge storage slots
6. IBM LTO Ultrium Tape Drives or 3592 Tape Drives
7. Front door
8. Door safety switch
9. I/O stations
10. Operator panel and operator panel controller

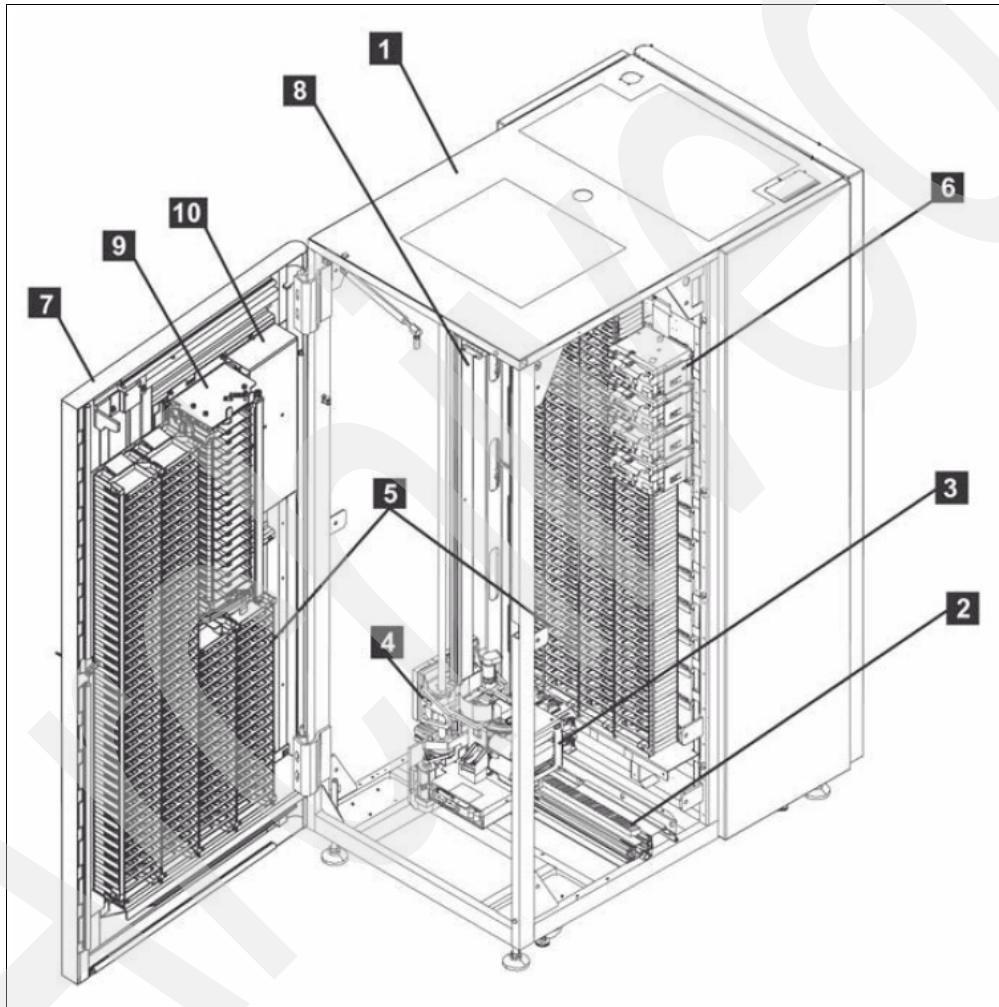


Figure 5-2 IBM TS3500 Tape Library L-frame showing major components

5.1.3 Tape - IBM System Storage TS1030 Tape Drive

The TS1030 LTO Tape Drive drive offers high capacity, performance, and technology designed for the midrange Open Systems environment. The TS1030 LTO Tape Drive has a 4 GB Fibre Channel interface for either point-to-point or Fibre Channel-Arbitrated Loop attachment. The native data transfer rate is 80 MBps. With data that compresses at a ratio of 2:1, the tape drive can achieve 160 MBps. The drive uses the IBM TotalStorage LTO Ultrium 400 GB data cartridge, which provides up to 800 GB of storage with 2:1 compression.

For additional information about the TS1030 tape drive refer to *IBM System Storage Tape Library Guide for Open Systems*, SG24-5946-04.

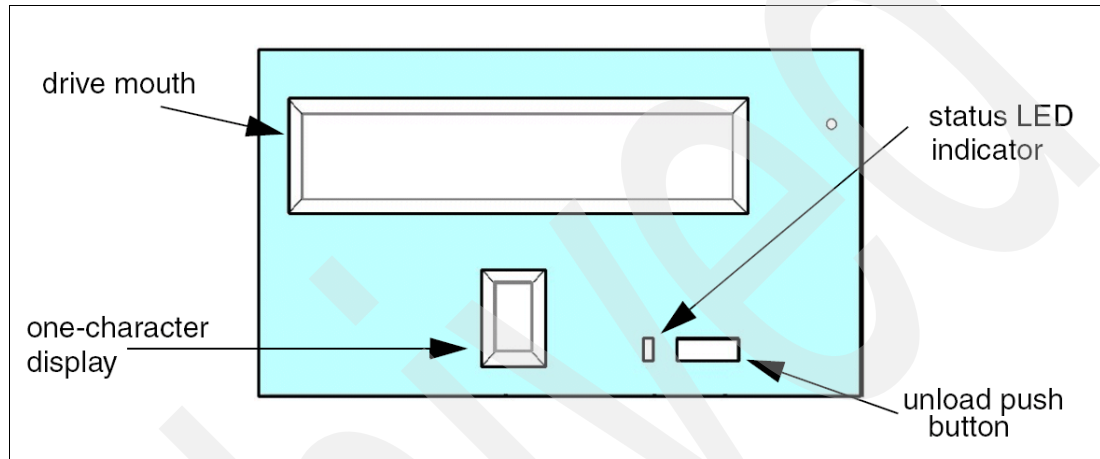


Figure 5-3 Drive front interface components

The status LED indicator uses color and lighting to indicate that:

- ▶ The tape is in motion for reading or writing.
- ▶ The drive is rewinding, locating, or unloading the cartridge.
- ▶ The drive is in maintenance mode.
- ▶ A failure occurred and the drive or media requires service.
- ▶ A microcode update is occurring.

The unload push button enables the operator to:

- ▶ Unload a cartridge.
- ▶ Enter maintenance mode and execute maintenance operations.
- ▶ Force a drive dump operation.

The one-character display indicates errors and communicates messages, such as requests for cleaner tapes. The operator uses the single-character display for diagnostic and maintenance functions. The drive mouth is the location for insertion of the tape cartridge.

5.1.4 Storage Area Network

Two IBM Storage Area Network switches were utilized in this project, the IBM TotalStorage SAN256M and the IBM TotalStorage SAN140M. Both switches were configured with 2 GB switch modules. For additional information about these switches refer to *IBM System Storage Solutions Handbook*, SG24-5250.

IBM TotalStorage SAN256M

IBM TotalStorage SAN256M, also known as the i10K, is designed to contain up to eight Line Modules (LIM), each with up to 32 Fibre Channel (FC) ports. A fully populated SAN256M comprises up to 256 FC ports in a 14U rack mount chassis. A variety of LIM types are available that enable a combination of 2/4-GB FC ports for connection to server and storage resources, as well as 10-GB FC ports for the Inter-Switch Link (ISL) between SAN256M directors. The project utilized one SAN256M for tape-to-host connectivity. Figure 5-4 shows a front view of the IBM SAN256M.



Figure 5-4 Front view of an IBM SAN256M

IBM TotalStorage SAN140M

IBM TotalStorage SAN140M is a 140-port product that provides dynamic switched connections between Fibre Channel servers and devices in a SAN environment. It takes 12U of rack space and scales from 16 to 140 ports. The project utilized two SAN140Ms for disk-to-host connectivity. Figure 5-5 shows a front view of the IBM SAN140M.



Figure 5-5 Front view of an IBM SAN140M

5.2 The project environment

The purpose of this section is to describe the project environment from the system storage perspective. The following is a list of the main storage hardware components of the project environment.

- ▶ Six p595s
 - One p595 for the catalog node
 - Four p595s for the DB2 data partitions
 - One p595 TSM server
- ▶ Five DS8300 Turbo Model 932s
 - Each with two Model 92E expansion frames
- ▶ One 6-frame TS3500 Tape Library
 - One L53 frame
 - Five D53 frames
- ▶ Sixty-eight TS1030 LTO Tape Drives
 - LTO-3 based tape drive
- ▶ One SAN256M
 - For tape-to-host connectivity
- ▶ Two SAN140M
 - For disk-to-host connectivity

5.2.1 Storage area network architecture

The storage area network architecture was designed so that there would be a sufficient number of Fibre Channel ports to achieve the project goals.

Host Fibre Channel ports were designated for either tape or disk access. Disk and tape devices were not accessible over the same Fibre Channel port.

In Figure 5-6 only the LPARs that had adapters connecting to the storage area network are shown. Other LPARs involved in the overall architecture, that did not have Fibre connection to the storage area network, were removed from the picture for clarity. We used the following naming convention for the LPARs:

- ▶ Sys1db{0–4} are the LPARs that hosted the production copy of DB2.
 - Sys1db0 was the catalogue node,
 - Sys1db{1–4} were the database nodes.
- ▶ Each of those LPARs had a related LPAR for facilitating the FlashCopy-based backup, Sys1sta{0-4}. The sta is short for storage agent. However, both sets of LPARs had the Tivoli Storage Manager Storage Agent software installed, as well as DB2.

For the FlashCopy backup, the LPARs Sys1sta{0–4} were used. But restores from tape were done directly to Sys1db{0–4}.

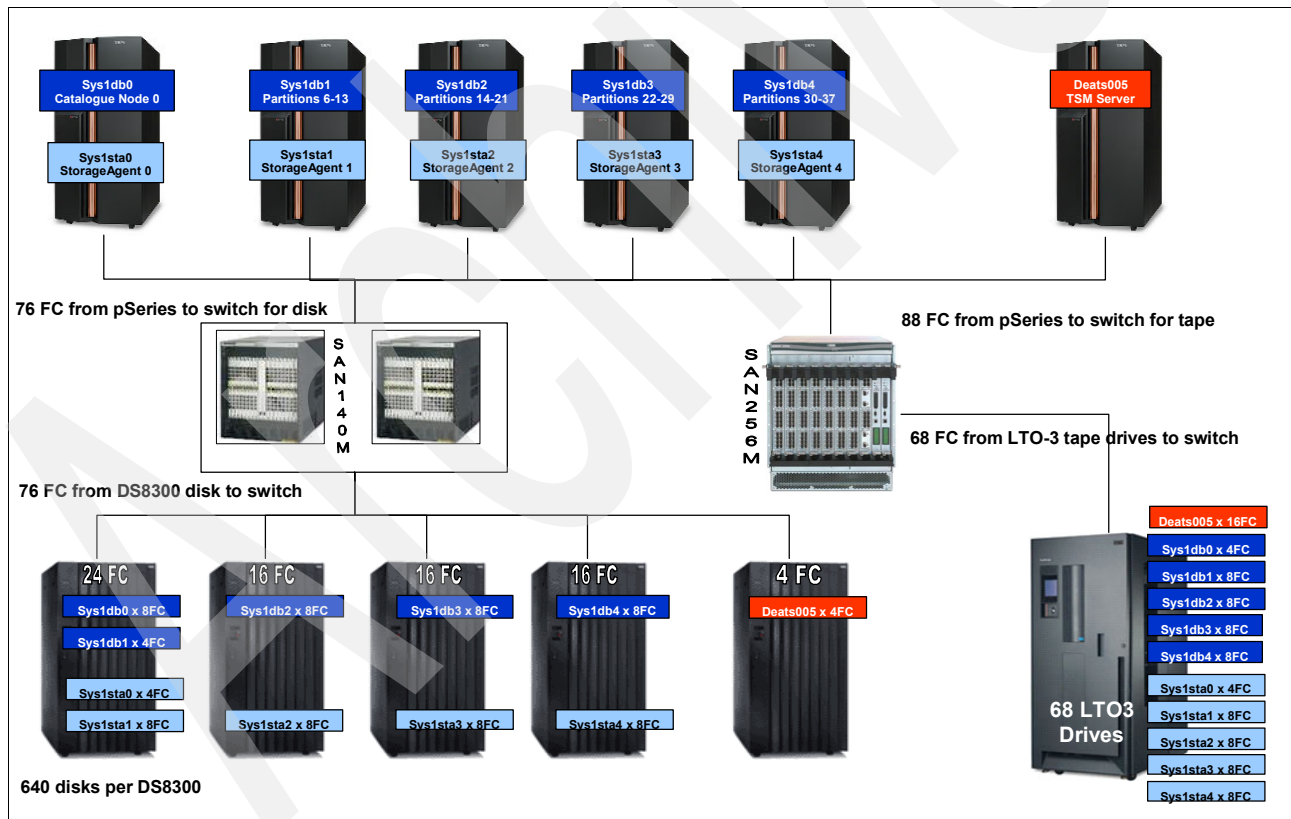


Figure 5-6 Simplified view of the storage area network architecture

Figure 5-6 shows the relationship and number of Fibre Channel connections between the p595 systems and the storage devices. For the disk connectivity, there was the same number of Fibre Channel ports on each p595 as on the disk. The ports were split between the two LPARs shown. However, for the Sys1db{0–4} systems, the pathing was set up so that each of

the adapters had paths to two adapters on the DS8300. This setup is explained in further detail in 5.2.3, “Disk configuration” on page 330.

Sys1db0 and Sys1sta0 each had four Fibre connections to disk that were split evenly to the two SAN140M switches. Sys1db0 and Sys1sta0 also had four fibre connections to the SAN256M switch for access to tape.

Sys1db{1–4} and Sys1sta{1–4} each had Eight fibre connections to disk that were split evenly to the two SAN140M switches. Sys1db{1–4} and Sys1sta{1–4} also had eight fibre connections to the SAN256M switch.

Deats005, the LPAR that had the Tivoli Storage Manager Server software, had four fibre connections to the disk storage. Deats005 also had sixteen fibre connections to the SAN256M switch.

In Figure 5-6 on page 325, from left to right, the DS8300s had the following number of FC ports connected to the SAN: 24, 16, 16, 16, 4. The DS8300 that was used by Deats005 for the TSM diskpool was also used by other hosts. These other hosts were not involved or active during the tests.

5.2.2 Zoning

Different approaches for zoning were taken between the disk and tape zoning. Disk and tape devices are accessed and shared differently. Disk subsystems can provide LUN masking, which tape devices typically do not have (virtual tape products are an exception). Tape drives and the tape library in a TSM environment can be shared, and the tape cartridge handling can be managed by a primary server. These inherent differences in the storage devices naturally lead to different approaches in zoning.

Disk zoning

There were relatively few zones involved with the disk storage. There was one zone for each LPAR that had fibre connections to the disk storage. The number of paths to each LUN was controlled via the DS8300’s LUN masking functionality. This approach kept the complexity of managing the paths to the storage device. It would have been possible to have many zones and precisely match the desired paths with the LUN masking. But in this environment that would have added only complexity without any other benefits.

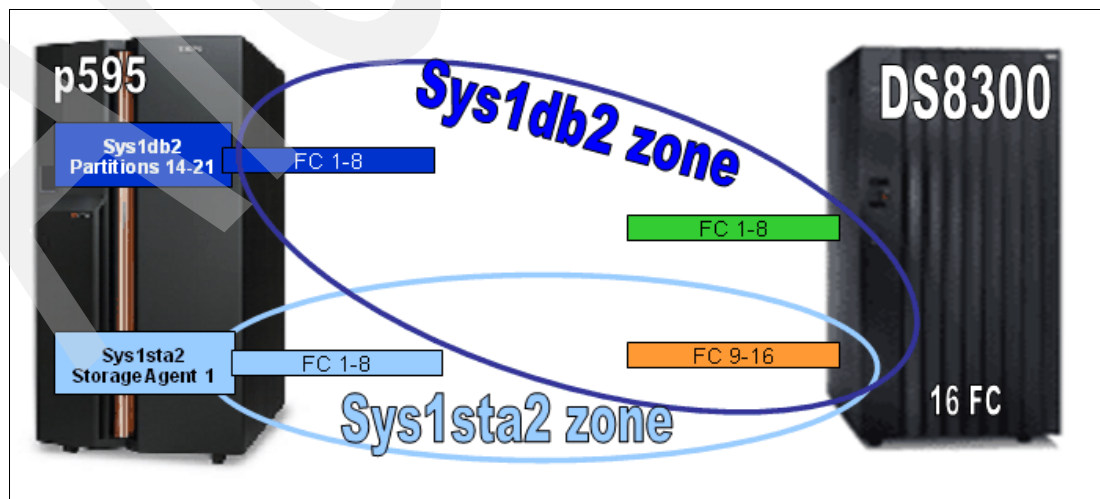


Figure 5-7 Basic disk zoning

For three of the four DS8300s that hosted the DB2 data, the zoning was identical to that shown in Figure 5-7 on page 326. These were the DS8300s that hosted LPARS Sys1db2, Sys1db3, and Sys1db4 (and the respective storage agent LPARs). For the storage agent LPARs, the eight fibre ports for disk were zoned with eight ports of the DS8300. For the DB LPARs the eight fibre ports for disk were zoned with sixteen ports of the DS8300, eight of which were the same as the storage agents. The idea behind having more ports on the disk side was to make sure that the disk was capable of providing more bandwidth to the p595. (A comparison test was not run to see how much of a benefit this provided, so it is not known how much improvement this provided compared to using one to one.)

Sys1db{0-1} and Sys1sta{0-1} shared the same DS8300. They used separate ports on the DS8300. In Figure 5-8 we see that the zoning is similar to the other DS8300s. The main difference is that Sys1db0 and Sys1sta0 have four fibre ports to the disk instead of eight.

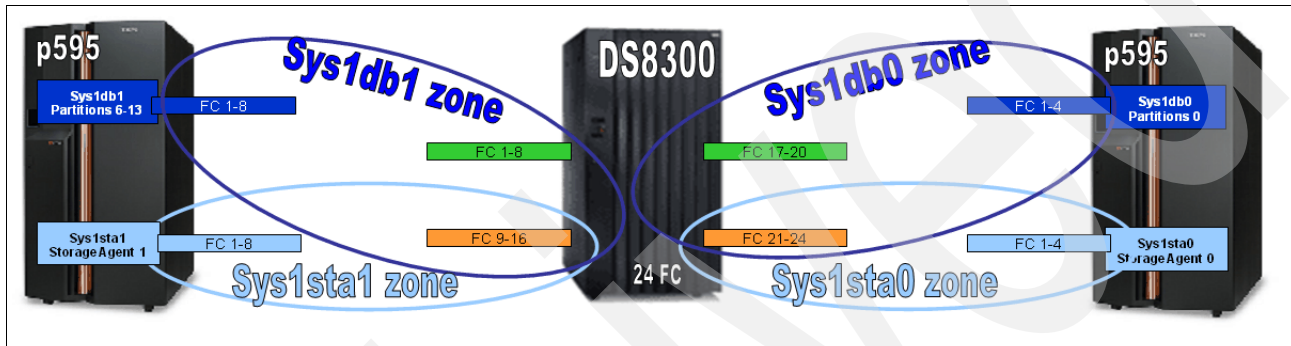


Figure 5-8 Disk zoning with catalogue node Sys1db0

Tape zoning

Sixty-eight tapes were available in the TS3500 library, but for the backup and restore tests only thirty-two were ever in use at one time. There was one zone for each tape drive. Each host adapter to have access to a tape drive was placed in that tape drive's zone.

In Figure 5-9 we see an example of the zoning that was used for most of the tape drives. In this case the zone included the tape drive, the TSM Server, one of the DB LPARs, and the correlating storage agent LPAR. There was no contention for this resource during the testing. The TSM Server was not accessing tape drives during the backup and restore KPIs. The db LPARs were used for restoring, while the storage agent LPARs were used for backup. The backup and restore tests were not run concurrently, so both hosts could have connectivity to the same tape resources.

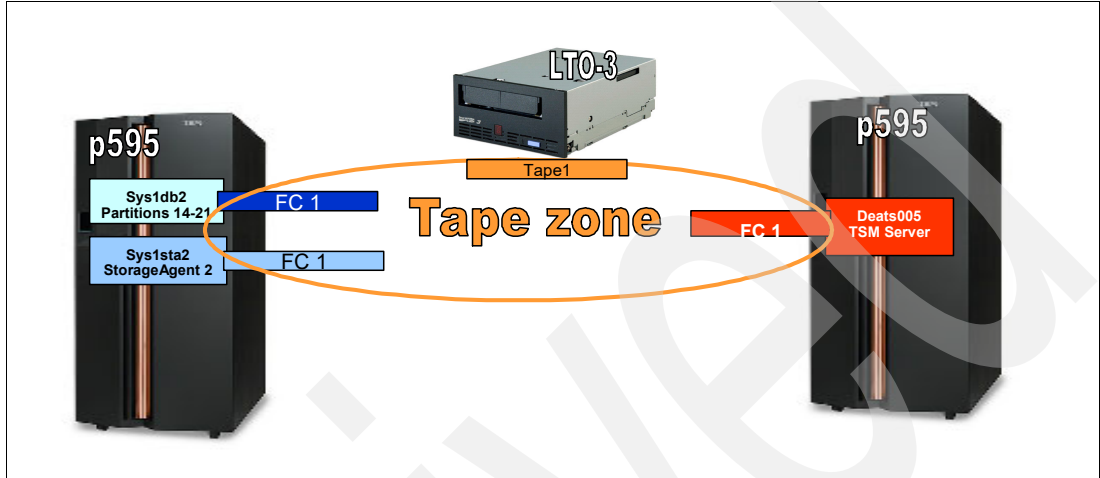


Figure 5-9 Basic tape zoning

The DB2 catalogue partition's fibre adapters were included in the tape drive zones along with other DB partitions. In Figure 5-10 we see an example of this. Since the DB2 catalogue partition was always backed up or restored before the rest of the nodes, there was no contention for these tape resources either.

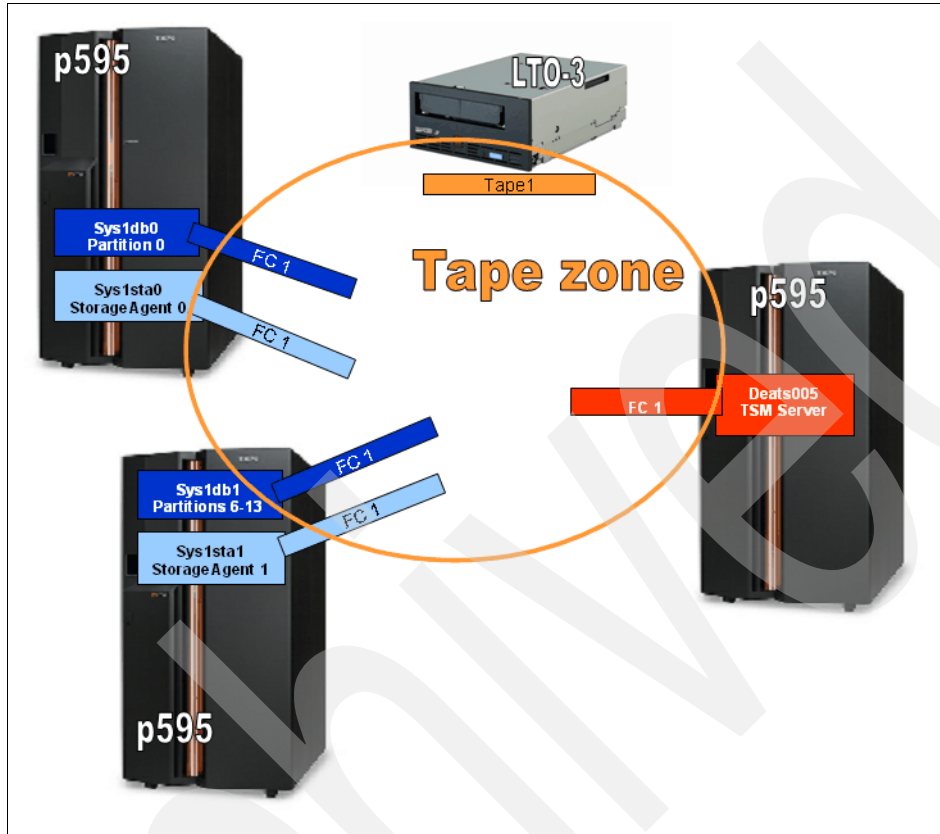


Figure 5-10 Tape zoning with catalogue node

From a host perspective, each host fibre adapter designated for tape was zoned with two tape drives¹. As mentioned, only one tape path on an adapter was active at a given point in time. This was controlled through the TSM software. It is much easier and less disruptive to change a path on the TSM server to be off-line than it is to modify the zoning of the switches.

¹ One adapter in each LPAR was zoned with one extra tape drive. The TSM drive path was never online, so for clarity this special condition is not discussed.

From the point of view of the host during a backup or restore, Figure 5-11 provides an example of what was visible on the host's fibre ports. The dotted line for the Tape2 zone represents the drive in that zone being offline. It is still zoned, but not accessible.

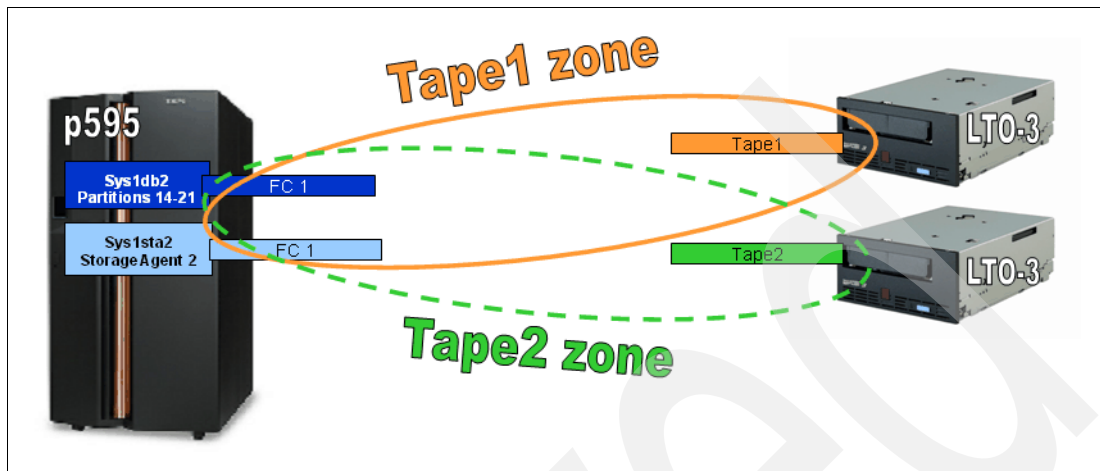


Figure 5-11 Tape access- host view

For Sys1db{1-4} and Sys1sta{1-4} there were eight fibre ports designated for tape. Therefore, sixteen drives were accessible by the system, but only eight had an online path via the TSM software.

Sys1db0 and Sys1st0 each had four ports designated for tape. Eight tape drives were accessible by the system, and four had online paths via the TSM software.

5.2.3 Disk configuration

This section assumes that the reader is familiar with the basic disk virtualization capabilities of the DS8300. If the terms and concepts described in this section are unfamiliar, refer to chapter six of *Virtualization Concepts in IBM System Storage DS8000 Series: Architecture and Implementation*, SG24-6786.

The DS8300's disk virtualization capabilities provide for a large degree of flexibility in configuring the storage. This flexibility results in an infinite number of ways in which the disk storage can be configured and assigned. While it is true that it is easier to manage fewer things than more things, when working with storage and things of this nature, it is a good idea to keep the number of items to be managed to a minimum. There is no point in having complexity for the sake of complexity. Even outside the human aspect of being difficult to manage many items, system operations also complete quicker with fewer items.

A primary goal of the project was to show the scalability of the solution. To this end, the implementation of DB2 followed a multi-partition shared nothing architecture. Ideally, each DB2 partition has its own dedicated CPUs, memory, disks, and so on. This was not followed exactly. The catalogue node Sys1db0 shared the same DS8300 as Sys1db1. However, the load characteristics of the catalogue node Sys1db0 and Sys1db1 did not impact each other. These two hosts had exclusive access to their own storage arrays, and for the infrastructure tests, the backup and restore of the catalogue node was not in parallel with the other host systems.

DS8300 array sites, RAID arrays, ranks, extent pools

Each of the DS8300s had two Model 92E expansion frames and were fully populated with 640 DDMs. In order to have control over the disk access/placement and to adhere to the

shared-nothing architecture, the disk was configured so that there was one array site, one RAID array, one rank, and one extent pool for each set of eight disks. With 640 available DDMs, this made 80 array sites, 80 RAID-5 arrays, 80 ranks, and 80 extent pools. Only 64 of the ranks were used in the four DS8300s for DB2 partition production data. Half of these RAID arrays were 7+P, with the other half 6+S+P. Since eight DB2 partitions were located on each DS8300, each DB2 partition had exclusive access to eight of the 64 ranks.

The remaining sixteen ranks were unused for three of the four DS8300s. The catalogue node partition 0 used twelve of these extra sixteen ranks from one DS8300.

Figure 5-12 illustrates the configuration of the DS8300 that included the catalogue node. The color corresponds to one of the eight device adapters (DAs). Based on the color and the numbering of the DAs, DA 2 and DA 0 are connected to the disk in the base frame and the second expansion frame. The disk in the second expansion frame attached to DA2 and DA0 corresponds to the extra sixteen ranks.

In Figure 5-12 are colored boxes that indicate which database partition the ranks were associated with. Each partition used space from the eight ranks attached to a different device adapter. Partition 0 is the special case, and shared DA 0 and DA 2. For the other three DS8300s, no volumes were defined for the extra sixteen ranks on DA 0 and DA 2. Each rank also included FlashCopy targets, which are illustrated in Figure 5-14 on page 333.

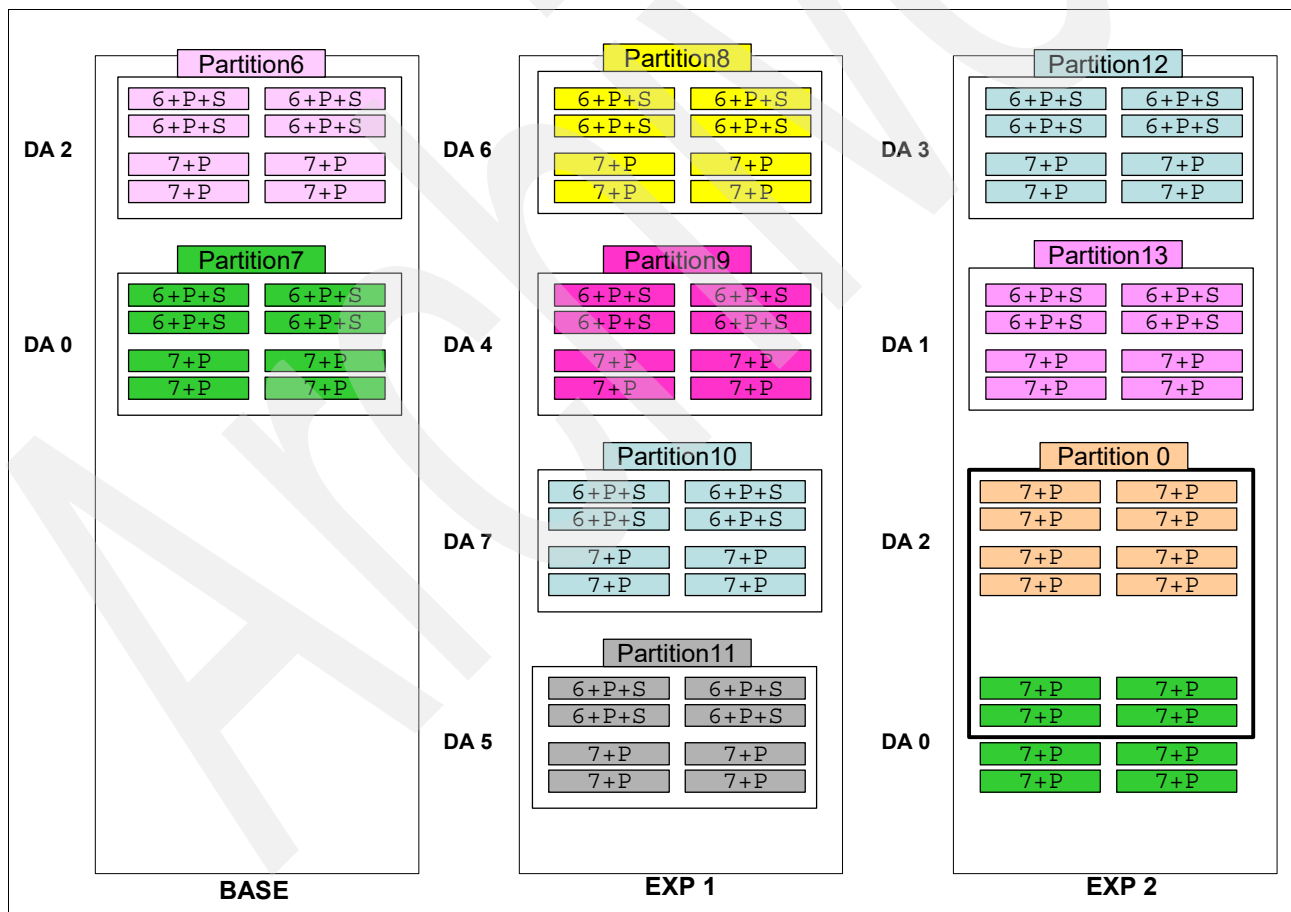


Figure 5-12 Array site, rank, and extent pool by disk adapter

DB2 production data volumes and FlashCopy targets

For the DB2 production data partitions, five open systems volumes per extent pool were created on each of the 64 extent pools. With 320 volumes per DS8300, this gives 1280

volumes for all the DB2 production data (five volumes per extent pool * 64 extent pools * 4 DS8300s).

Figure 5-13 shows the volume and RAID array sizes based on the 146 GB DDMs used. Total available capacity for the RAID arrays with and without spares was 779 GB and 909 GB, respectively. The RAID arrays without the spare had 130 GB more available capacity, but this capacity was not used. Of the five volumes, two were 350 GB in size, and the other three were 25 GB in size. One of the 350 GB volumes and one of the 25 GB volumes were used as FlashCopy targets.

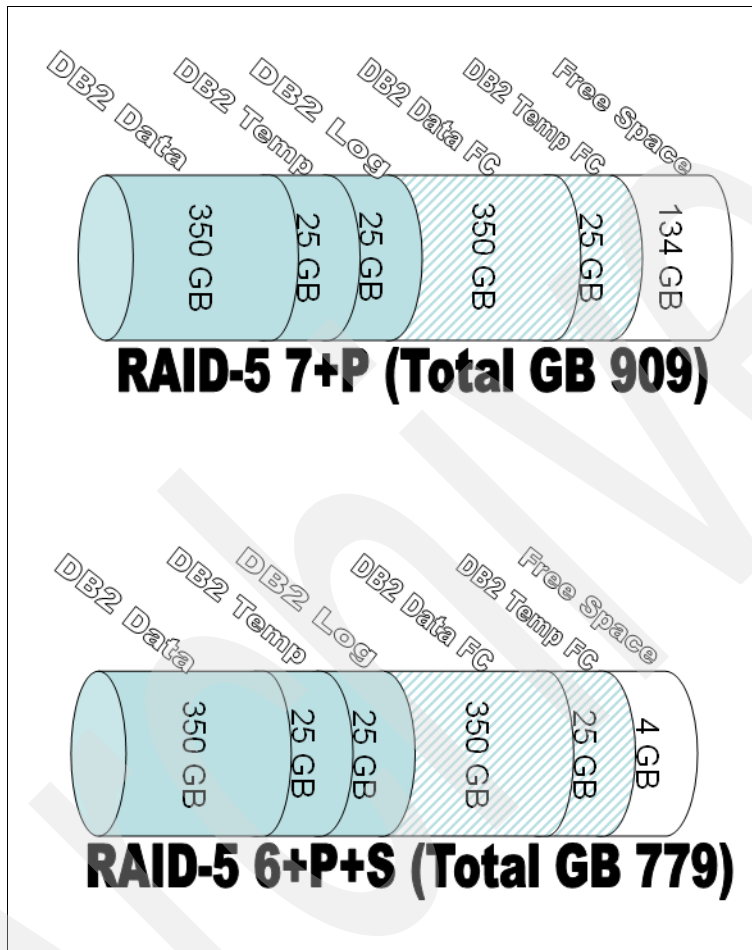


Figure 5-13 Volume sizes for RAID array with and without spare

If we combine everything discussed so far in this section and look at just the base frame for one of the DS8300s, then we get what is shown in Figure 5-14. The solid red parts of the cylinder represent the volumes that are assigned to one DB2 partition. These volumes would be further configured by the host system and used by AIX LVM to create file systems for subsequent use by DB2. The red striped parts on the separate DA represent the FlashCopy targets for the solid red. Conversely, the green solid is for a different DB2 partition, and the green striped, the corresponding flashcopy target. Also illustrated is the association to either server 0 or server 1 in the DS8300. Half the storage is assigned to one, and half to the other.

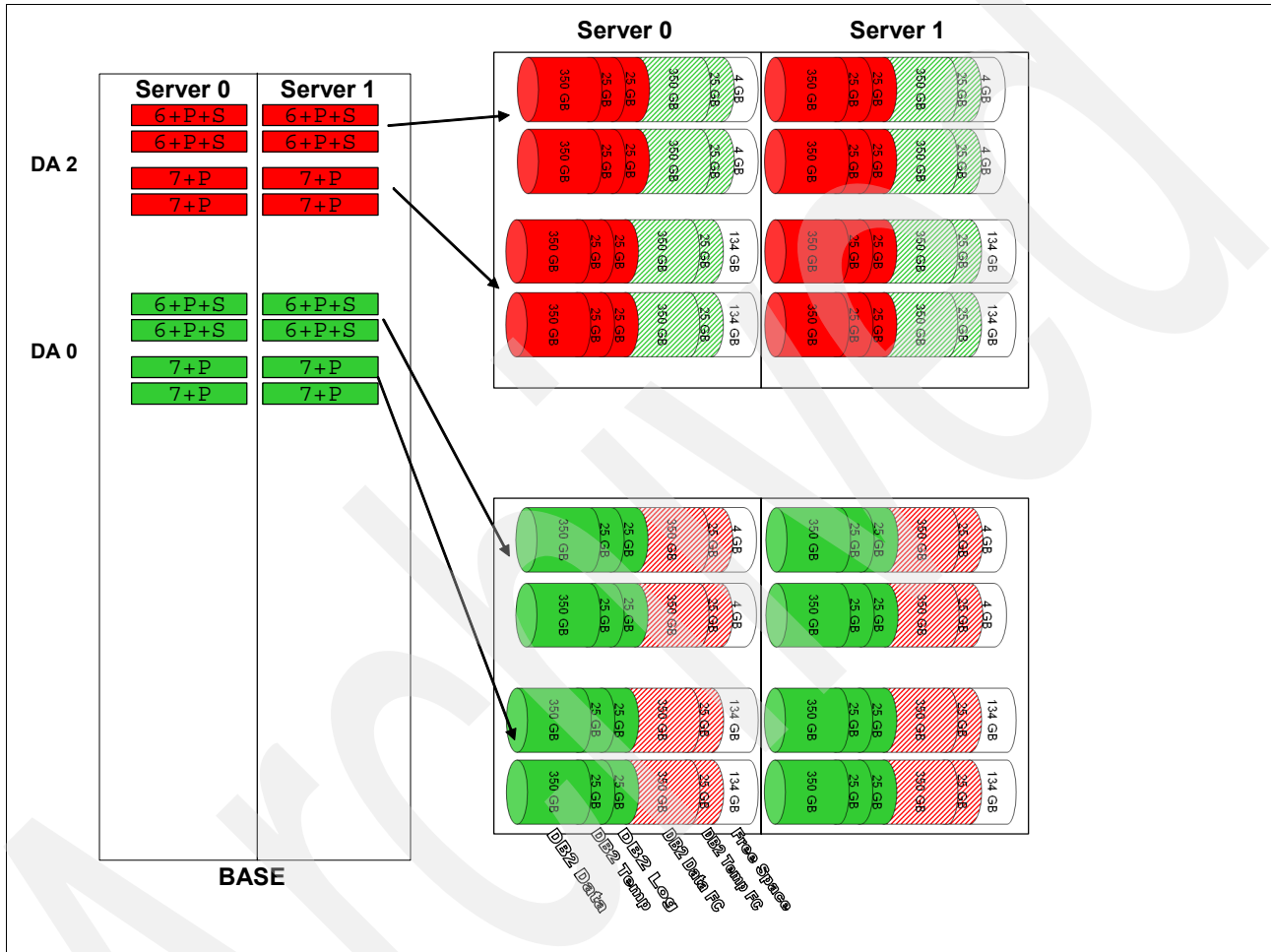


Figure 5-14 FlashCopy source and target volume sizes

Catalogue partition/DB2 partition 0

The catalogue partition was configured differently from the other partitions. The catalogue partition shared a DS8300 with DB2 partitions 6–13. It did use separate ranks within the DS8300. Figure 5-15 shows how the volumes were configured for the twelve ranks associated with the catalogue node.

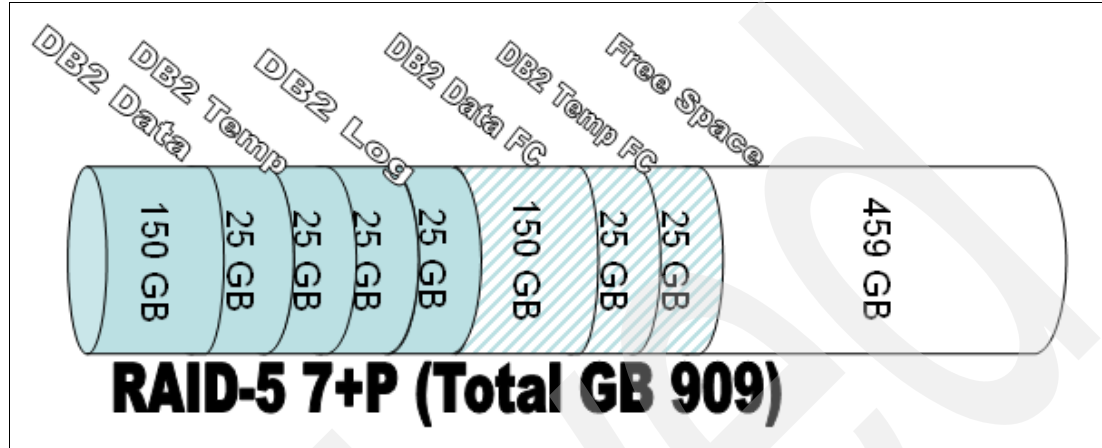


Figure 5-15 Volume sizes for RAID arrays assigned to catalogue node

The size of the volumes used for the data and index for the catalogue were smaller, at 150 GB. There were two 25 GB volumes per rank for the log and two 25 GB volumes per rank for the temporary tablespaces. The data and temp volumes had corresponding FlashCopy target volumes.

DS8300 volume groups and host attachments

The combination of zoning in the SAN and the DS8300 LUN masking capability determined the number of paths accessible by the host p595s to each LUN.

With five volumes per rank, eight ranks per DB partition, and eight DB partitions per DS8300, this makes 320 volumes per DS8300, 192 volumes for the data partition, and 128 volumes for FlashCopy targets. The DS8300 with the catalogue partition is a special case and has an additional 100 volumes (64 for the catalogue partition Sys1db0, and 36 for the FlashCopy LPAR Sys1sta0). The 320 volumes for the data partitions were grouped into four volume groups. Two volume groups for the data LPAR and two volume groups for the storage agent LPAR. The number of host attachments for each LPAR were also split in two. This reduced the total number of paths from which the volumes were accessible from the host.

As discussed in 5.2.2, “Zoning” on page 326, sixteen fibre ports were available on the DS8300s for each data LPAR. An additional eight were available on the DS8300 that hosted the catalogue node. Given the number of available ports and the fact that the volumes were split into two volume groups, this leads to eight paths per volume for the data LPARs and four paths per volume for the storage agent LPARs. Table 5-1 shows the output from the DS8300 when listing the volume groups.

Table 5-1 Open systems volume groups for DS2

| Nickname | Number | Volumes | Host attachments |
|-------------------|--------|---------|------------------|
| DS2_VG_P14_P17 | 0 | 96 | 8 |
| DS2_VG_P14_P17_FC | 2 | 64 | 4 |
| DS2_VG_P18_P21 | 1 | 96 | 8 |
| DS2_VG_P18_P21_FC | 3 | 64 | 4 |

The DS8300 that hosted the partition, in addition to eight data partitions, had two more volume groups, as depicted in Table 5-2. The data LPAR’s volumes were split into two volume groups consisting of 96 volumes each with eight host attachments. The corresponding FlashCopy volume groups were split into two volume groups consisting of 64 volumes each with four host attachments. P10_P13 represents the four partitions, 10 through 13. DS1_VG_P10_P13_FC was used for disk system 1, volume group, partitions 10 through 13, and FlashCopy targets.

Sys1db0 or partition 0 LPAR’s volumes were put into one volume group, consisting of 64 volumes with eight host attachments (DS1_VG_P0). The corresponding FlashCopy volume group (DS1_VG_FC) consisted of 36 volumes with four host attachments.

Table 5-2 Open systems volume groups for DS1 with partition 0

| Nickname | Number | Volumes | Host attachments |
|-------------------|--------|---------|------------------|
| DS1_VG_P0 | 4 | 64 | 8 |
| DS1_VG_P0_FC | 5 | 36 | 4 |
| DS1_VG_P10_P13 | 1 | 96 | 8 |
| DS1_VG_P10_P13_FC | 3 | 64 | 4 |
| DS1_VG_P6_P9 | 0 | 96 | 8 |
| DS1_VG_P6_P9_FC | 2 | 64 | 4 |

Disk configuration summary

So far the way the DS8300s were configured has been discussed, but little has been mentioned concerning how the host system configured the volumes allocated.

Figure 5-16 is slightly different from Figure 5-14 on page 333. In Figure 5-14 on page 333 the sizes for the volumes have been replaced with file system information. There are four file systems for data containers. A file system for a data container comprises two ranks, which span both of the servers in the DS8300. There are also four file systems for the temp containers. A file system for the temp containers also comprises two ranks that span both servers. There is only one file system for the log, which includes all eight ranks.

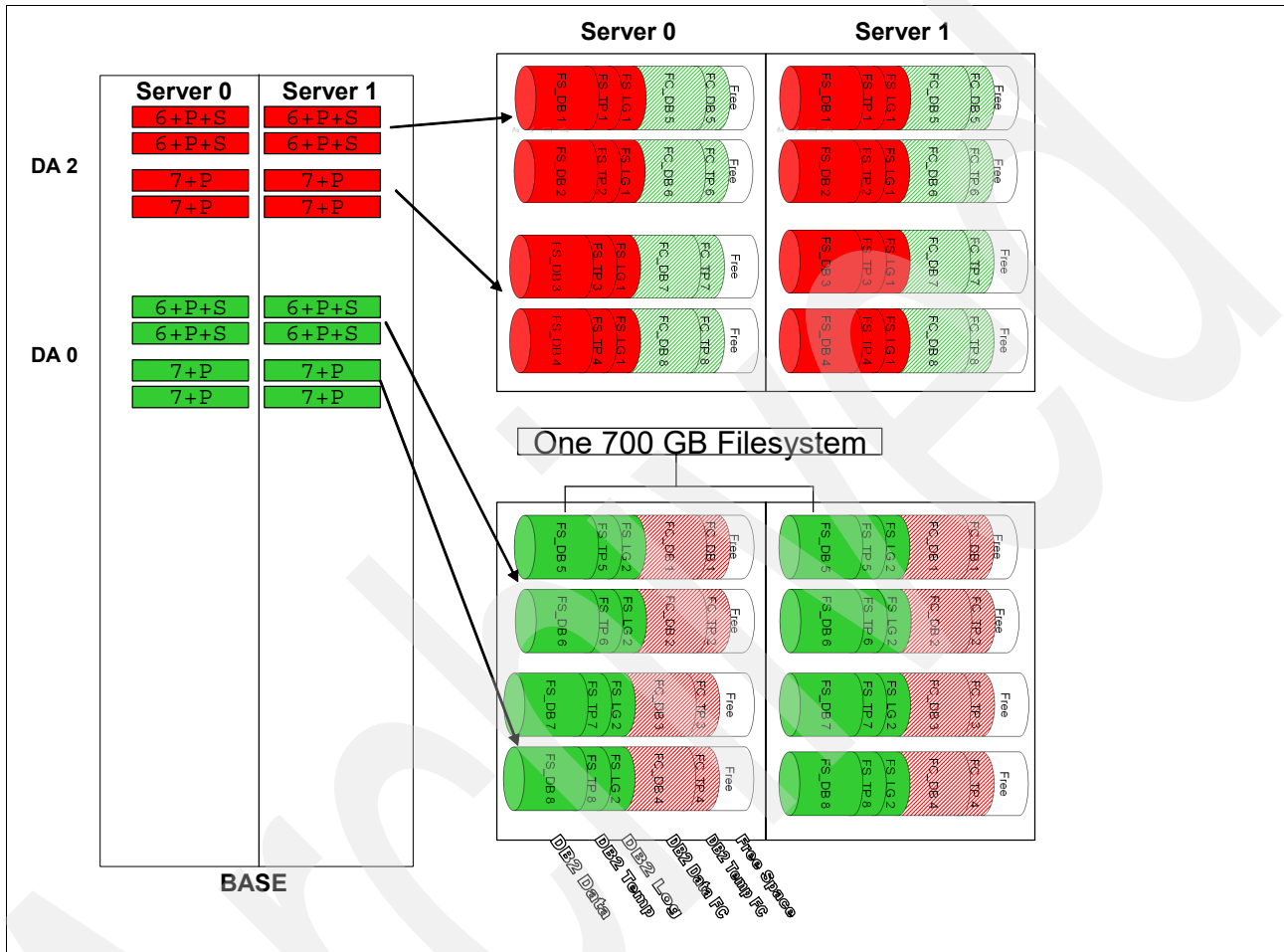


Figure 5-16 FlashCopy source and target file system specified

A more detailed look into the file system and the AIX view is covered in Chapter 4, “The IBM System p perspective” on page 287.

If we omit the catalogue partition and look at the 32 data partitions, then Figure 5-17 summarizes the disk configuration.

| <ul style="list-style-type: none"> Filesystems <ul style="list-style-type: none"> For each DB2 partition <table border="1"> <thead> <tr> <th></th> <th>Production</th> <th>FlashCopy</th> </tr> </thead> <tbody> <tr> <td>• 4 Filesystems of 700 GB for DATA</td> <td>2,800 GB</td> <td>2,800 GB</td> </tr> <tr> <td>• 4 Filesystems of 50 GB for TEMP</td> <td>200 GB</td> <td>200 GB</td> </tr> <tr> <td>• 1 Filesystems of 200 GB for LOG</td> <td>200 GB</td> <td></td> </tr> </tbody> </table> Total 288 Filesystems Total Production = $3\ 200\ \text{GB} * 32 = 100\ \text{TB}$ (25% free space) Total FlashCopy = $3\ 000\ \text{GB} * 32 = 93.75\ \text{TB}$ 4 x DS 8300 of 64 ranks of 8 disks 146 GB 15KRPM in Raid 5 <ul style="list-style-type: none"> 8 ranks are dedicated to <ul style="list-style-type: none"> a DB2 partition production data a FlashCopy target of another DB2 partition Each rank is divide in 5 LUNs, 40 LUNs per DB2 partitions 1 filesystem of Data on 2 ranks 320 LUNs per DS8300 | | | | Production | FlashCopy | • 4 Filesystems of 700 GB for DATA | 2,800 GB | 2,800 GB | • 4 Filesystems of 50 GB for TEMP | 200 GB | 200 GB | • 1 Filesystems of 200 GB for LOG | 200 GB | |
|---|------------|-----------|--|------------|-----------|------------------------------------|----------|----------|-----------------------------------|--------|--------|-----------------------------------|--------|--|
| | Production | FlashCopy | | | | | | | | | | | | |
| • 4 Filesystems of 700 GB for DATA | 2,800 GB | 2,800 GB | | | | | | | | | | | | |
| • 4 Filesystems of 50 GB for TEMP | 200 GB | 200 GB | | | | | | | | | | | | |
| • 1 Filesystems of 200 GB for LOG | 200 GB | | | | | | | | | | | | | |

Figure 5-17 Disk layout 60 TB with 32 DB2 partitions

5.2.4 Tape configuration

The tape library used in this project consisted of one L53 frame and five D53 frames. The L53 frame and one of the D53 frames had twelve LTO-3 drives installed. The other four D53 frames had 11 LTO-3 drives installed. Sixty-eight LTO-3 drives were available. However, only 32 drives were utilized in achieving the performance targets.

Figure 5-18 shows a picture of what a six-frame TS3500 tape library looks like. Only one logical library containing all the resources was configured. The single logical library contained all 68 tape drives and all of the storage slots.



Figure 5-18 Six-frame TS3500 Tape Library

The sharing of the drives and media between the various LPARs was handled and coordinated by the TSM software. Only LTO-3 media was used.

The TS3500 has additional functions and features that were not used in this project. In a typical customer environment consisting of this many tape drives and frames, the physical library would be split into multiple logical libraries. Even without using the advanced functions of the TS3500 library, the basic functions provided by the library and drives were capable of meeting the extreme throughput demands required by the backup and restore requirements.

5.3 Options and parameters discussion

This project's size and scope resulted in situations that closely mirrored those encountered in real customer environments. Even after careful planning, internal deliberation, customer meetings, and many discussions with various experts, many decisions had to be made with imperfect information. It is not possible to foresee all the issues to be encountered. At some point you have to start the actual implementation and deal with problems as they arise.

Each of the storage products involved has enough features and configuration possibilities to warrant its own book. This section focuses on just a few of the major options and parameters that were important to the particular implementation of this project.

5.3.1 Disk options and parameters

With DB2 configured in a multi-partition shared nothing architecture, it is important that the disk resources available to each partition have equivalent performance. DB2 handles distributing the same amount of work to each partition, but if the partitions are unequal in their available system resources, the overall performance of the DB2 database suffers.

Configuring and allocating disk volumes

One of the main architecture decisions is the size, number, and allocation of disk volumes. One possible method for ensuring that each DB2 partition has equal resources is to distribute each partition across all the ranks in all the DS8300s.

With 32 data partitions, this would require 32 volumes per rank just for the data volumes. Add 32 more for the temp file system and another 32 for the logs and you are already at 96. FlashCopy targets are required for the 32 data and 32 temp volumes, so an additional 64 volumes are needed. This adds up to a total of 160 volumes per rank. With 64 ranks to be used, this gives (160*64) 10,240 volumes per DS8300. Multiply that by four DS8300s and you would have to manage almost 50,000 volumes just for the 32 data partitions. Distribution in this manner, while equal, would be unmanageable.

A better approach, and the one used in the project, is to dedicate equivalent disk resources to each partition and keep the number of volumes to a minimum. The design described in 5.2.3, "Disk configuration" on page 330, dedicated eight physical ranks to each partition. These eight physical ranks were serviced by their own device adapter. They had equivalent resources, but fewer total volumes. With the flexibility of the virtualization capabilities of the DS8300, other layouts could have been implemented that would keep the total number of volumes to a reasonable amount. Different disk storage would necessitate a similar study of the underlying architecture to ensure that each partition has equivalent access to resources.

FlashCopy target considerations

There are different options to consider when deciding on how to allocate the volumes to be used as FlashCopy targets. One obvious choice is to have the production volumes and

FlashCopy target volumes on separate ranks. This would reduce the total number of volumes to be managed and be simple to implement. The big drawback to this approach is that during production workloads, half of the disks and ranks would be idle.

Placing production volumes and FlashCopy target volumes on the same rank provides access to all the DDMs for the production workload. The number of volumes may be double than in a separate array implementation, but the substantial increase in overall performance for production workloads is worth the trade-off.

Only the data/index and temporary tablespace LUNs need to be flashcopied. The logs are already managed and sent to the TSM server automatically.

Although production volumes and FlashCopy volumes were on the same rank, the volumes on the same rank were not configured as source and target pairs for each other. The target FlashCopy volumes were on different arrays from the source FlashCopy volumes. The FlashCopy volume pairs were split across device adapters but on the same internal server (0 or 1).

5.3.2 Tape options and parameters

Tape drives by their very nature lend themselves to a shared nothing architecture. An individual tape drive can only be opened and accessed by one process at a time. During the session with the tape drive, the process has exclusive access to all of the drive's resources.

Tape devices per host bus adapter

It is possible to have multiple tape devices share the same host bus adapter. This can lead to contention for the bandwidth of the host bus adapter. For this implementation tape devices were configured and paths set so that during the backup and restore workloads, each tape device used a dedicated host bus adapter.

The tape drives used were LTO-3 with 4 GB interfaces. The SAN256M was configured with 2 GB switch ports, so the tape drive was forced to operate at a 2 Gb rate. With compressible data and 2 GB Fibre Channel, a single LTO-3 tape drive can fully utilize an entire 2 GB path. In this environment the tape drives average 140 MBps. The interface rate for the 2 GB Fibre Channel is roughly 200 MBps, but a portion of that rate has to be used for command/communication overhead. One hundred and seventy-five MBps is close to the maximum data rate that is possible. With each tape drive running at 140 MBps, the backup would have been Fibre Channel limited if two of the tape drives were active on the same host bus adapter.

In a 4 GB environment, while still possible for two LTO-3 tape devices to have contention on a single 4 Gb interface, it would be more likely for other parts of the data path to be the limiting factor.

In summary, with LTO-3 tape drives, we recommend only having one LTO-3 tape drive per 2 GB fibre connection.

5.3.3 DB2 logs and tape devices

DB2 logs and tape devices, if not managed, can cause significant delays during restore situations. In a multi-partition DB2 environment each partition creates its own logs. During a rollforward process, each DB2 partition opens its own session with the TSM server and requests its logs. This is done in parallel, and if the logs for multiple partitions are on the same

tape cartridge, then partitions have to wait for access to the resource. This can lead to time outs.

A common approach to log archiving is to send them to a disk storage pool that is later migrated to tape. This approach, which works fine during the archive log operation, leads to the problem mentioned previously of having contention for the same tape cartridge. Even if collocation is used and the logs are on separate tape cartridges, if the number of tape drives is less than the number of DB2 partitions, you can also run into contention at the tape drive level. Three approaches are discussed:

- ▶ Large TSM diskpool
- ▶ One tape drive per DB2 partition
- ▶ TS7500 virtual tape server

Large TSM diskpool

One approach for managing DB2 logs in a multi-partition implementation is to keep the logs on a large TSM diskpool. This is a simple straightforward method for ensuring that each DB2 partition can request access to its own logs without waiting for a tape drive or cartridge to become available.

One major downside to this approach is that the logs would be transferred over the local area network and would not use the bandwidth of the storage area network. Another disadvantage is the amount of dedicated disk space that would be required to host all the DB2 logs.

One tape drive per DB2 partition

Another approach and the one used in this project is to have a dedicated tape drive per DB2 partition. This approach, when combined with collocation, allows each drive to archive and restore logs without contention from other partitions. LAN-free communication can be configured and used for transferring the logs to and from the tape devices.

One problem with this approach is the number of tape drives required. Not every environment has enough tape drives to dedicate to this type of approach. But depending on the importance of the data and in order to satisfy service level agreements, the number of drives could easily be justified. We also recommend having an extra drive or two available in case of drive failure.

TS7500 virtual tape server

The nature of archiving DB2 logs in a multi-partition environment would be well served by the functionality of the TS7500 line of products. These products take disk resources and virtualize them so that they are accessed and viewed as tape resources by the host.

A major advantage of a virtual tape library over a dedicated TSM disk pool is the ability to share the storage resources across multiple servers and applications. The disk storage in a TSM disk pool can only be utilized by that TSM server. The disk storage in a TS7500 virtual tape library is accessible by any host with access to the TS7500. Each host can have its own virtual tape library and virtual tape drives, but share the common disk capacity.

A virtual tape drive per DB2 partition could be configured. And the virtual tape drives can be set up to use LAN-free. Collocation would be required, but easy to do with the product.

The base behavior for the virtual tape cartridges is to start at a beginning size of 5 GB and grow in 5 GB segments up to the virtual limit for the cartridge type. When the cartridge is written from the beginning of tape, the size shrinks back to the starting 5 GB.

Tape motion operations are in milliseconds (mounts, loads, locates, rewinds, and so on).

One major difference between physical tape drives and virtual tape drives is throughput scaling. Each physical tape drive provides linear throughput scaling. With virtual tape drives there is some scaling with a small number of tape drives, but at some point you start sharing the available disk bandwidth as well as the fibre attachment bandwidth. Thirty-two physical tape drives attached in a 4 GB environment running at 240 MBps provide up to 7,680 MBps of compressed throughput. The TS7510 has a total throughput of 600 MBps, which, if shared across a corresponding 32 virtual drives, would be 19 MBps per drive. This is much faster than access over a local area network, but significantly less than the tape storage. This is common to virtualization products. You get better system utilization, but not as good maximum performance as dedicated products.

5.4 Results

The system storage provided the backbone for the 60 terabytes of data implemented in the project. With the amount of storage available and the adherence to a shared-nothing implementation, it is possible to use the hosts view of the storage for monitoring and determining results. Access to the disk system was for the most part only done by one system at a time. The NMON reporting, which includes the host's access to disk storage, in this project environment provides a good view of the load and performance of the disk system.

For the tape storage, the Fibre Channel SAN256M reporting tools used, which consolidated the port throughput at 15-minute intervals, provided a good view of the tape storage. Only one tape drive was active per port, and the duration of the backup was 4–5 hours.

Since the system storage was fundamental to the achievement of the KPIs, and since the other sections provide the customer and application view of the system storage, some of the results are left to be discussed in those sections. In this chapter a few of the results specific to tape storage are discussed.

5.4.1 Tape KPI results

This section discusses the KPI results from a storage IT preoperative.

KPI-G

Tape storage was not involved in this KPI.

KPI-1- flashback then rollforward 500 GB of logs

Tape was not involved in this KPI. Flashback is done within the DS8300 subsystem. Logs for the rollforward were located on a TSM disk storage pool.

KPI-2 - restore from tape then rollforward 2 TB of logs

For the restore of the database from tape, the bulk of the restore was processed at a rate of 4,127 MBps for over four hours. As is typical in restores, the last bit of processing takes additional time, and the throughput decreases as some jobs finish earlier than others.

Figure 5-19 illustrates the timeline for the restore portion of KPI2, and omits the rollforward timeline. From 22:39 to 23:38 the system processed the restore of partition 0 using four tape drives. From 23:38 to 3:42 (four hours, four minutes) the bulk of the restore was processed at a peak average of 4127 MBps using 32 tape drives. From 3:42 to the end of the backup, longer running restores were processed.

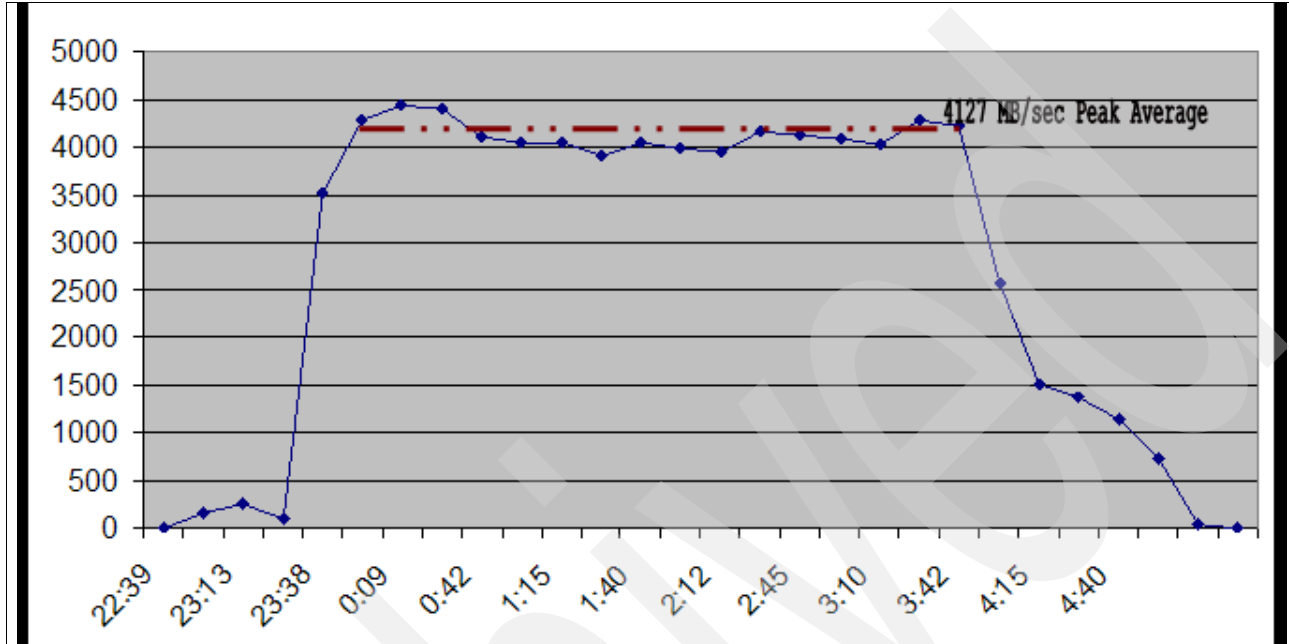


Figure 5-19 KPI-2 tape restore average peak throughput

Figure 5-20 provides a look at two ports attached to LTO-3 tape drives used during the restore. The top chart in the figure shows a drive that was also involved in the restore of partition 0 from 22:39 to 23:38. The bottom chart is for a drive that only was used in restoring one of the data partitions. No activity occurred for this drive during the time period of 22:39 to 23:38.

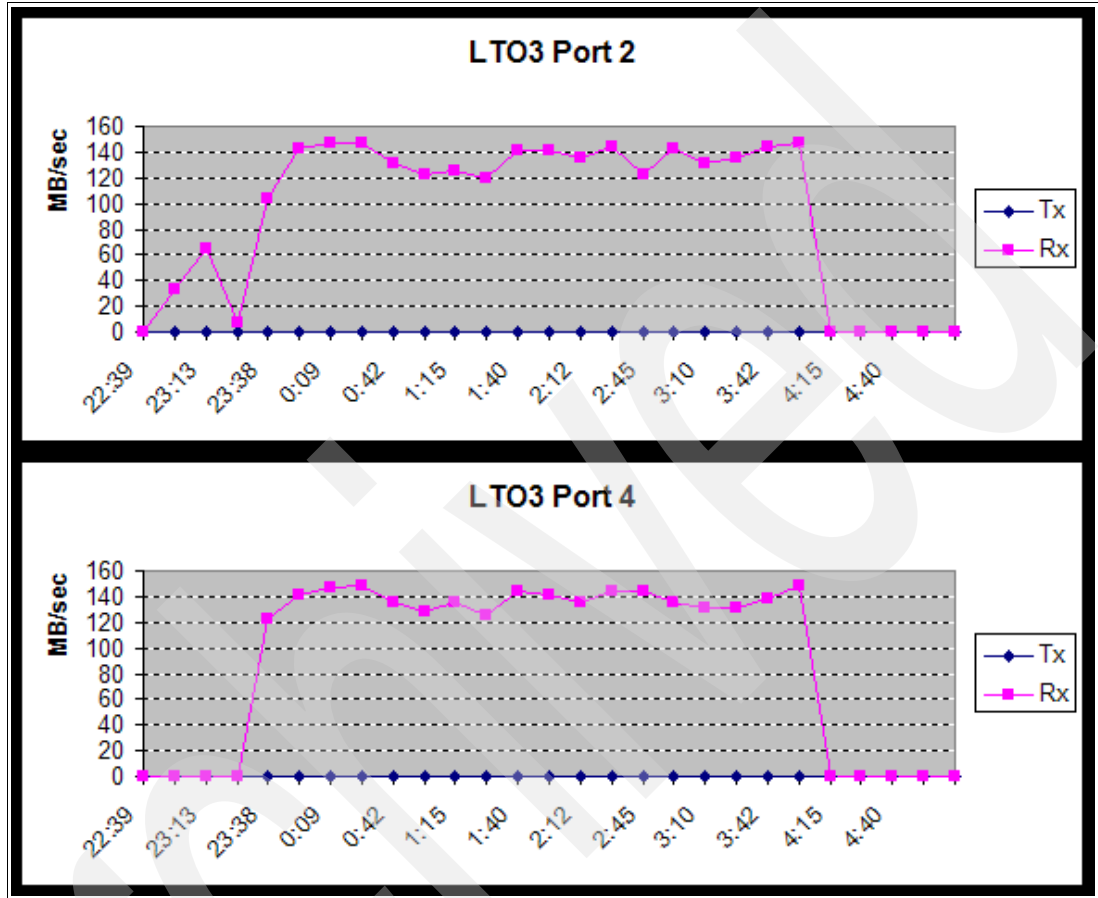


Figure 5-20 LTO-3 ports restoring the database

KPI-3a tape backup

Thirty-two LTO-3 tape drives were the foundation for the backup of the system. During the backup of the 32 database partitions, the tape storage sustained over 4,500 MBps for over four hours — more than sixteen terabytes of data an hour.

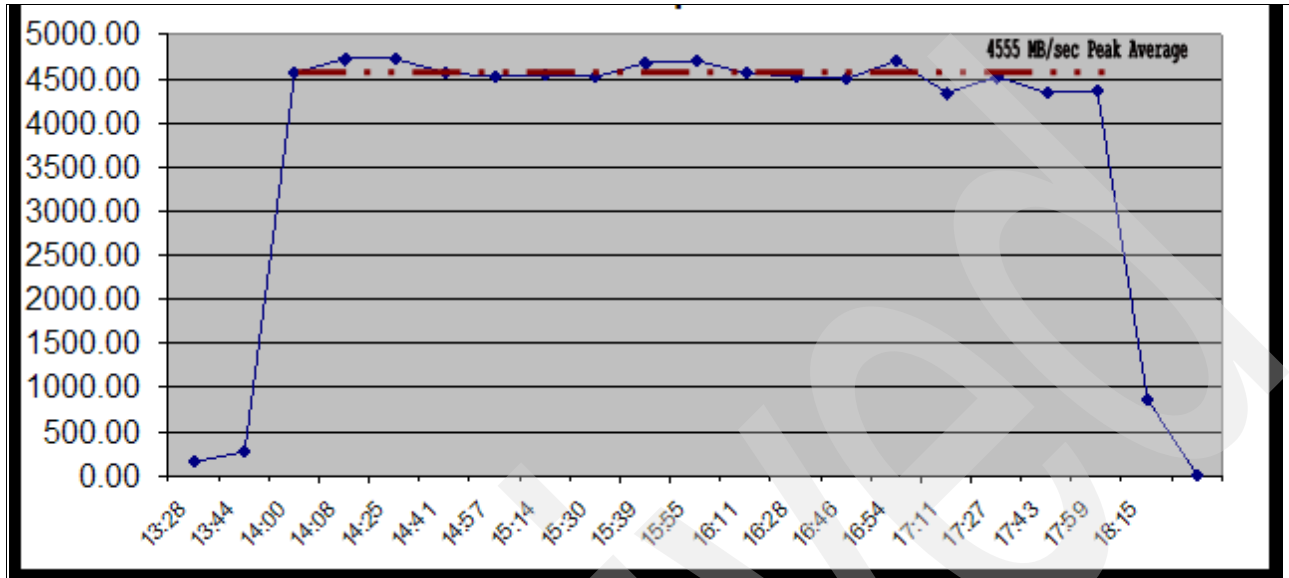


Figure 5-21 KPI-3a tape backup average peak throughput

Figure 5-21 on page 344 illustrates the throughput achieved with the tape storage. The first two data points are for when partition 0 was backed up. DB2 requires that the catalog partition be backed up for the other partitions. The software was configured so that four tape drives in parallel backed up partition 0. Once partition 0 is backed up, we see the increase in total throughput when the 32 data partitions are backed up — one tape drive per partition. The numbers for this figure were obtained from the SAN256M. The average throughput per tape drive during this four-hour period was 142 MBps with a standard deviation of 5 MBps.

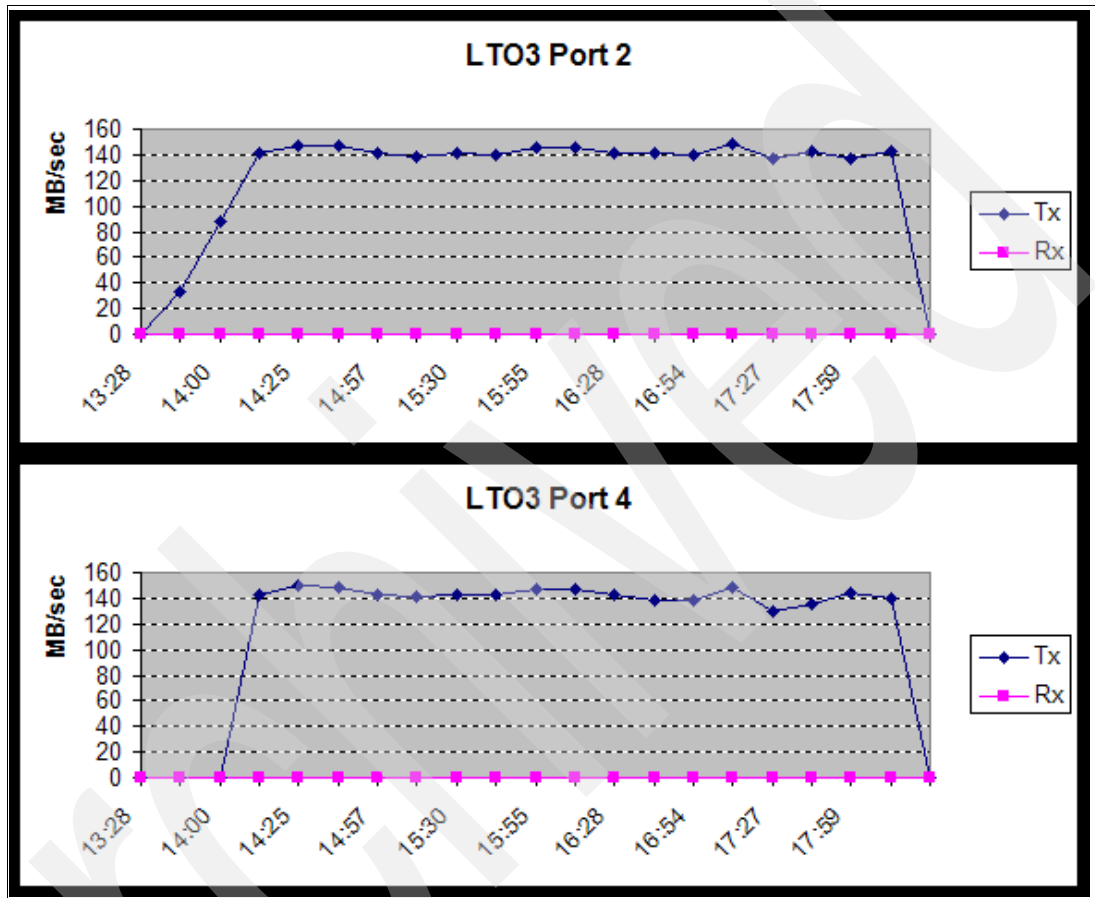


Figure 5-22 LTO-3 ports backing up the database

Figure 5-21 on page 344 compares two different ports. Port 2 was connected to one of the four tape drives used to back up partition 0, the catalogue node. Throughput to port 2 started before port 4 and was at a lower rate. After the drive connected to port 2 finished backing up partition 0, then it was used for one of the 32 data partitions. During the backup of the data partitions, the tape drives average 142 MBps.

The tape drives were able to compress the data in this environment at approximately 6:1. This was determined in two ways. First, during the testing, the LTO-3 cartridges would fill up, and TSM reported that 2.4 TB of data was contained on each cartridge. With 400 GB of native capacity, the data would need to compress 6:1 to fit 2.4 TB. The other method involved collecting logpage 32 from the tape drives after the backup. This was done with the AIX command `tapeutil -f /dev/rmtX logpage 32`, where X is the device special number of the tape drive. Using the output of that command and the SCSI reference document for the LTO-3 tape drive, the tape drive reported a compression ratio of on average 6:1.

With 2:1 data and a 2 GB or 4 GB Fibre Channel interface, the IBM LTO-3 tape drive is capable of reading and writing 256 KB blocks² at 160 MBps. With data that compresses 6:1,

the tape drive with a 2 Gb interface can run at the maximum line rate for a 2 Gb Fibre Channel. This is normally around 175 MBps³, but is dependent on the type of host bus adapter and switch technology used. Since the rate was not achieved, something in the setup was not providing data to the tape drive for it to run at maximum speed. Further tuning could have improved the average MBps for each drive and further shortened the backup window. However, the data rates achieved surpassed the requirements for the project in meeting its goals. Therefore, time was not spent in this area to try and achieve additional performance.

² Tivoli Storage Manager Server with LTO-3 devices reads and writes to tape with a 256KB blocksize

³ With a 4 Gb interface and 6:1 compressing data, the IBM LTO-3 tape drive is capable of 245 MBps.



The Tivoli Storage Manager perspective

This chapter describes the tests done to demonstrate that the proposed infrastructure is manageable in terms of data backup and restore.

- ▶ 6.1, “The project environment” on page 348, describes the project from an IT storage manager specialist perspective.
- ▶ 6.2, “Options and parameters discussions” on page 371, discusses some of the main options that can influence the management of the environment.
- ▶ 6.3, “Infrastructure tests” on page 375, presents the results of the tests requested by the customer.

6.1 The project environment

This section describes the constraints of the tests and the environment, the solutions provided by Tivoli to backup and restore the database and the components implemented in our environment to satisfy the customer requests.

6.1.1 The challenge

A major IT challenge is to manage backup, restore and recovery of databases to accommodate the requirement of rebuilding business essential data in integrity within a very short period of time after an event of failure (logical error or physical error) has always been.

The IBM DB2 UDB databases support the backbone of the enterprise business applications and the stable and reliable operation of large and fast growing databases is a key to business success

The infrastructure part of the performance test addresses the manageability of a 60 TB database from the point of view of a recovery from a disaster. This part of the test addresses the feasibility of operations like Flashcopy, backup, restore and recovery.

The customer requirements were the following:

- ▶ Tape backup of a Flashcopy in less than 12 hours.
- ▶ Flashcopy restore and simultaneous roll forward of 500 GB of logs in less than 8 hours.
- ▶ Database restore from tape using Tivoli Storage Manager and roll forward 2 TB of logs in less than 24 hours.

These test scenarios reflect the business requirements, for example being able to recover from a local issue within one business day, that can be achieved using a Flashcopy restore. Another scenario that needs to be addressed is recovering from a disaster that requires a full database restore. This recovery scenario has to be completed within a timeframe of 24 hours: During the tests, the tape restore, immediately followed by the rollforward operation for 2 TB of log files and the (re-) creation of indexes in a combined run takes about 20 hours runtime. The remaining 4 hours may be needed to set up hardware and application infrastructure.

The amount of log files reflects the average generation of them during normal production workload: 500 GB of log files may be generated as a peak amount during one day with high workload, and typically 2 TB of log files are generated during one week.

The timeframe for the backup can be met by using the FlashCopy functionality of the storage server and a subsequent backup to tape with Tivoli Storage Manager from another server/LPAR: using this approach, the backup is offloaded from the production database and so does not impact production performance. Another different approach to limit the impact on the production environment is to test the capability of DB2 to throttle a database backup.

In summary, all the infrastructure tests address the requirement of backing-up the 60 TB database daily, without any significant impact on production and the capability to recover from a system failure within a business day or to completely build a new database environment within 24 hours.

6.1.2 Introducing the backup/restore solution

Multiple solutions exist to backup and restore databases. this section discusses these solutions.

Tivoli Storage Manager for dummies

IBM Tivoli Storage Manager is a centralized policy-based data backup and recovery software. The software enables a user to backup, restore, archive, and retrieve data from a hierarchy of data storage areas. The storage areas, known as pools, can be a hierarchy of disk, optical, and tape-based media.

A typical Tivoli Storage Manager installation allows data backups from Tivoli Storage Manager clients to be spooled to random-access disk storage attached to the Tivoli Storage Manager server. That data is then migrated at a later time to less-expensive sequential-access storage, such as tape or optical disk.

Tivoli Storage Manager also enables LAN-Free and server-free backup in SAN fabric environments.

In Table 6-1 we describe briefly the Tivoli Storage Manager products and the features we used.

Table 6-1 Tivoli Storage Manager products and features

| Product or Feature | Function |
|--|---|
| Tivoli Storage Manager Server | Provides backup and archive management services. It is the core of the solution. Provides tape library management. |
| Tivoli Storage Manager Backup/Archive Client | Allows users to backup, archive or restore their files |
| Tivoli Storage Manager Client API | Allows users to enhance existing application programs with Tivoli Storage Manager services (for example DB2) |
| Tivoli Data Protections for my SAP/DB2 | Works with Tivoli Storage Manager server to protect mySAP/DB2 exploiting the backup utilities provided for DB2 |
| Tivoli Data Protections for Advanced Copy Services (aka Tivoli Data Protections for hardware) | Integrates Tivoli Storage Manager server, TDP for my SAP/DB2 and DS8000 flashcopy all together to provide a "near zero-impact" data backup and "near instant" recovery solution |
| Storage Agent | Enables Tivoli Storage Manager to perform LAN-free and Server-free operations on a Production or data mover server |

Backup techniques for database

There are several techniques you can use to back up data managed by an RDBMS. These techniques are, at least at a conceptual level, common to most RDBMSs. A combination of the following techniques may be used:

- ▶ Disk mirroring

Mirroring is the process of writing the same data to multiple storage devices at the same time. Disk mirroring is a useful technique to maximize the database availability, as it means users can continue working when a media failure has occurred. This solution is more dedicated to high availability than backup solution: it is still necessary anyway to back up databases.

- ▶ Database export

All RDBMSs provide export and import utilities, which operate on logical objects as opposed to physical objects. For example, you can use an export command to copy an individual table to a file system file. You might want to restore the table at some later time, in which case you would use the import command. Export and import are time consuming and are not designed as backup and restore utilities, but instead they can be used to move data for workload balancing or migration, for example

- ▶ Offline backup

To make an offline backup, you need to shut down the database before starting the backup and restart it after the backup is complete. The obvious but significant disadvantage is that neither users nor batch processes can access the database (read or write) while the backup is in progress. Most databases do not require that you perform offline backups if you perform online backups; online backups (along with the log files) are sufficient to recover the database.

- ▶ Online backup

Most RDBMSs enable backups to be performed while the database is started and in use. Clearly, if a database is being backed up while users are updating it, it is likely that the backed up data will be inconsistent. The RDBMSs that support online backup use log files during the recovery process to recover the database to a fully consistent state. This approach requires that you retain the RDBMS log files and indicate to the RDBMS when you are about to start the backup and when you have completed it. This method impacts negatively the RDBMS performances.

- ▶ Full database backup

A full database backup is a copy of all of the data files used to hold user data. In some database products, full database backups also include copies of the data files that hold tables used by the RDBMS itself, RDBMS log files, and any control files and parameter files that the RDBMS uses. Many RDBMSs provide both full online and offline database backups; however, the backup is different in each case.

An offline full backup can be done using operating system utilities, RDBMS utilities, or the IBM Tivoli Storage Manager backup-archive client to back up the data files that constitute the database. An online backup requires an RDBMS utility to create data files containing a copy of the database. You can then use IBM Tivoli Storage Manager to back up these data files along with the parameter files that you use to start the RDBMS.

The simplest approach to database backup is to perform only full, offline backups at regular intervals. This approach is relatively easy to administer, and recovery is relatively straightforward. However, it may not be practical to take databases offline for the period of time necessary to perform full backups at the frequency you need. You may have to adopt a more flexible approach.

Some database products provide incremental backup, which only backs up changed database pages or blocks. This is called a true incremental backup, as opposed to a simulated incremental backup (called a log file backup).

Understanding what incremental backup means for a database is critical.

- ▶ Partial database backup

Many RDBMSs allow both online and offline partial database backups. A partial database backup is a backup of a subset of the full database (such as a tablespace or data files that make up a tablespace). It is often not the best approach to back up only a subset of a database, because you must ensure that what you back up represents a complete logical unit of recovery from the perspective of both the application and the RDBMS itself.

If you have added a new data file to a tablespace, you must ensure that any control file that the RDBMS uses to define the relationship between data files and tablespaces is also backed up. You may need to back up data files that the RDBMS does not manage

► Log file backup

For some applications, the units of recovery are too large to be backed up on a daily basis (for example, performing a full daily backup). The constraining factor might be the backup window, or the network and CPU overhead of transferring all the data, for example.

An alternative is to capture only the changes to the database by backing up the RDBMS log files. This type of backup is sometimes called an incremental backup (versus a full daily backup), but it is really a simulated incremental backup, as opposed to a "true" incremental backup. A true incremental backup backs up changed database blocks or pages, whereas a simulated incremental backup backs up the database transactions.

Recovery from a simulated incremental can be much longer than from a true incremental because you must reapply all of the transactions in the logs.

► Incremental backup

Some RDBMSs provide for backing up data that has changed since the last offline or online database backup. This saves tape or disk space, but might not reduce the backup duration because the RDBMS still has to read each data block to determine whether it has changed since the last backup. When recovery is needed, the database backup and incremental backups are required to fully recover the full database. Incremental backups are useful for saving space or for saving bandwidth when backing up over the network

► Backup of RDBMS supporting files

Most RDBMSs require certain files to operate but do not back them up when using their backup utilities. These files can be initialization parameter files, password files, files that define the environment, or network configuration files.

They are external files and are not part of the database because they must be accessible for reading or editing even when the database is down. For example, the password file provides authentication in order to administer a database, especially for starting up a database from a remote site.

You must ensure that these files are also backed up using operating system tools or third-party tools such as IBM Tivoli Storage Manager

► Backup using storage server advanced copy services

A backup may potentially degrade the performance of a production system. In a 24x7 environment or with very large databases, it is particularly important to run backups without interfering with normal operation. To free the production system from the overhead of backup, it is valuable to have a copy of the database for backup, reporting, or other purposes.

Some intelligent storage servers, such as IBM Total Storage DS6000, DS8000, and SAN Volume Controller, provide an advanced copy service, FlashCopy. A FlashCopy is an identical and independent copy of one or more disk volumes, called a FlashCopy pair, which is created within the storage server. Normally these copies can be established in a very short time (five to 20 seconds, depending on the machine).

If the database resides on a storage server that supports FlashCopy, a copy of the disk volumes can be established and assigned to another (backup) machine.

On the backup machine, the (backup) database can be accessed exclusively for backup or other purposes.

It is important that the data on the disk volumes is consistent while creating the FlashCopy volumes. One way to achieve this is to shut down the database and synchronize to disk,

all of the data that may reside in memory. After the FlashCopy is established the database can be started again.

If the database cannot be stopped, then the database itself must provide features to ensure that the data on the disk will be in a consistent state when establishing the FlashCopy pair

Important: For this performance tests we decide to use the flashcopy backup and restore technique (flashback) , in order to achieve the expected results in terms of performance and availability; Tivoli Data Protection for Advanced Copy Services is the core of our backup/restore solution.

DB2 UDB backup and restore

This section focus is the DB2 backup and restore processes.

Backup and restore commands are integrated directly in DB2 UDB. While performing a backup/ restore DB2 UDB will fork additional processes:

- ▶ Buffer manipulator processes read data blocks from the tablespaces and store them into intermediate buffers. The number of buffer manipulator processes forked during the backup is controlled via the PARALLELISM clause in the DB2 BACKUP/ DB2 RESTORE command.
- ▶ Media controller processes read the content of the intermediate buffers and transmit them to the target device. For Tivoli Storage Manager, the number of media controller processes is controlled via the OPEN SESSIONS clause. Sufficient resources (tape drives) need to be available on the Tivoli Storage Manager server, to open all sessions in parallel.
- ▶ To be able to allocate the required memory for the specified buffers, the DB2 database configuration parameter UTIL_HEAP_SZ needs to correspond to the memory usage.

Number and size of the intermediate buffers is controlled via the WITH BUFFERS clause. Figure 6-1 shows this process.

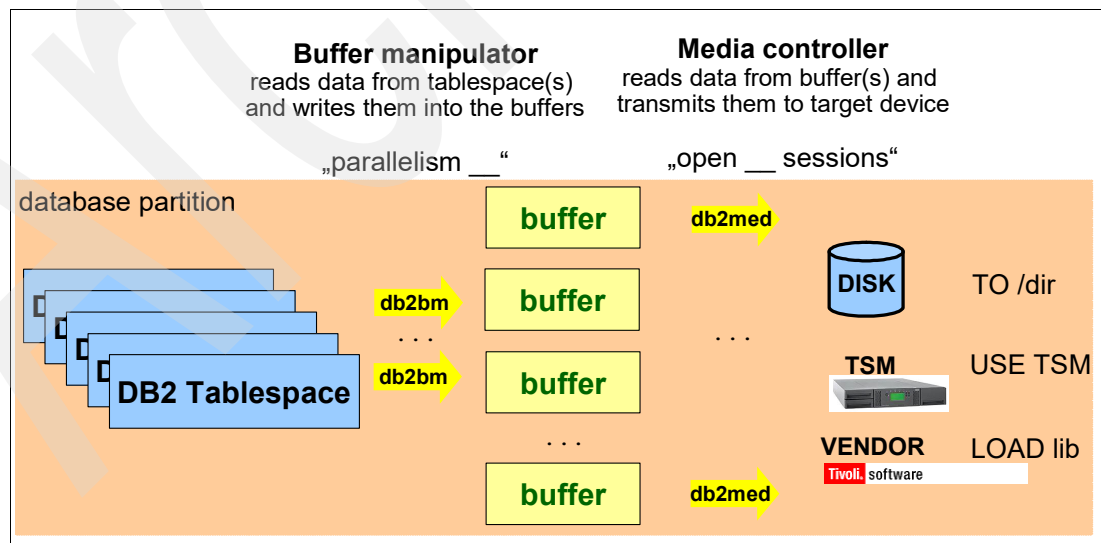


Figure 6-1 DB2 backup process

The database manager configuration parameter `util_impact_lim` can limit the performance degradation of the database workload during the runtime of the backup. No throttling for backup or another utility is enabled (`util_impact_lim = 100` is the default setting).

During the run time of the backup, the database workload may be affected. If the impact on the application during backup run time is undesirable or too high, the backup may be throttled down. Backup throttling is enabled by setting up `util_impact_lim` to a value less than 100, and the backup has to be invoked with a non-zero priority. Specifying a `util_impact_lim` value of 10, for example, should restrict the maximum impact of the backup on the database workload to 10% (or less). A throttled backup usually takes longer to complete than a non-throttled one.

Important: In a DPF setup, DB2 backup/restore commands need to be invoked on all database partitions. The database containing the database catalog is always backed up/restored first and sequentially before the others. All further partitions can be backed up/restored sequentially or in parallel.

Log file management in a DB2 UDB environment

Before DB2 Version 8, a user exit was called to archive or retrieve inactive database logs, either to an archive file system or directly to a Tivoli Storage Manager server. In SAP implementations, the inactive log files were moved to an archive file system first. Another process (called `brarchive`) was scheduled regularly to transfer them to the Tivoli Storage Manager server. An admin database (created within the same DB2 instance) took care about the history for an eventual required retrieve action.

Since DB2 UDB V8, the DB2 log manager is integrated within DB2 UDB and is available to manage the inactive log files. In our test environment, the DB2 log manager is activated to manage the database log files for all database partitions independently.

The DB2 log manager is controlled by different parameters. In particular, the database configuration parameter `LOGARCHMETH1` specifies the primary destination to archive the inactive logs. A possible destination could be `DISK` (archiving to a file system), Tivoli Storage Manager (archiving using the standard Tivoli Storage Manager API), or `VENDOR` (specifying an additional load library).

Options may be specified in `LOGARCHOPTS1`. If the method for archiving the log file was unsuccessful for `NUMARCHRETRY` times (having a delay of `ARCHRETRYDELAY` seconds between two attempts), the log files are archived to an alternate (disk) destination `FAILARCHPATH`. It is also possible to archive a second copy of the log file by specifying an optional redundant second method `LOGARCHMETH2`. During a rollforward recovery, the log files that are required for recovery but are not present are retrieved to the `OVERFLOWLOGPATH` from the DB2 log manager independently for each database partition.

Figure 6-2 illustrates the DB2 log management process.

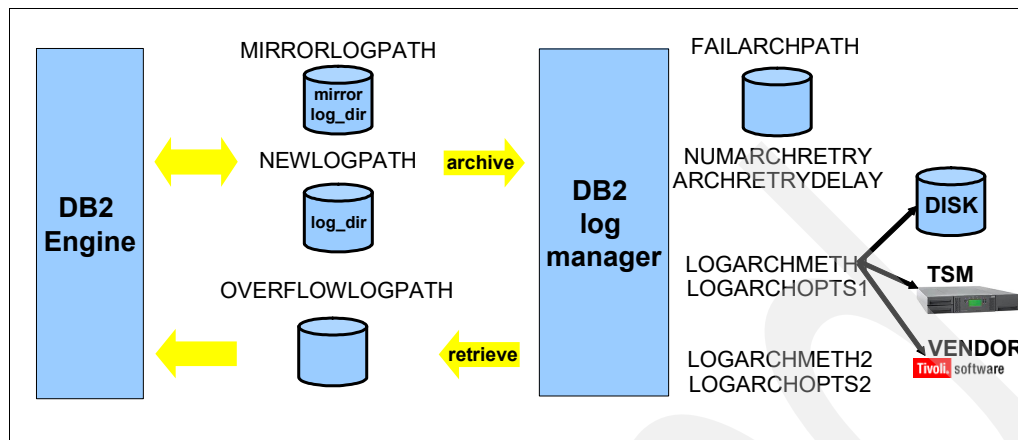


Figure 6-2 DB2 log management

A DB2 log file may have one of the following four states during its life cycle:

- ▶ Online active
DB2 UDB currently logs transactions in the log file.
- ▶ Online retained
The log file is no longer used for logging transactions, but it contains transactions corresponding to data pages changed in the buffer pool. Currently, these buffer pool blocks have not been written to disk. DB2 UDB needs the log entries in case of a crash recovery or a rollback. The DB2 log manager nevertheless copies the filled online log file to the archive location.
- ▶ Offline retained
The log file is no longer used by DB2 and does not contain transactions with any unwritten data page. The log is not crucial for a crash recovery or a rollback. The DB2 log manager copies the log to the archive location. DB2 UDB reclaims the space in the file system by deleting or overwriting the offline retained log files in the database log directory.
- ▶ Archived
A filled or closed log file that was successfully archived (to Tivoli Storage Manager).

During a rollforward recovery following a restore operation, all archived log files required for a rollforward or rollback activity have to be retrieved.

In our test environment, Tivoli Data Protection for mySAP™ is used to send/retrieve the archived logs to/from Tivoli Storage Manager. With the database distributed across several database partitions, each database partition has its own DB2 log manager. However, the log files for different database partitions may be related to each other. The following scenario has to be avoided while running a rollforward operation on multiple database partitions in parallel:

- ▶ The logs are archived independently to the Tivoli Storage Manager server for all database partitions, and eventually to a disk storage pool first. Finally, the logs were migrated to tape cartridges in a tape storage pool.
- ▶ One tape cartridge contains log files for multiple and different database partitions.

The rollforward operation runs in parallel on all different database partitions. Each of the database partitions has its own log manager. Each log manager retrieves the logs required for recovery of its partition from the Tivoli Storage Manager server. As all the log managers

run independently from each other, it may happen that two or more database partitions request log files located on one and the same tape cartridge.

The log manager/rollforward processes for some database partitions need to wait until the tape cartridge is released from the first session. So, the rollforward activity on one database partition may serialize the rollforward activities on further database partitions. Timeouts and termination of rollforward processes may occur on the database partitions, which need to wait for the resources.

To avoid that risk, the logs of different databases have to be retrieved either from:

- ▶ A common DISK storage pool. A disk storage pool allows you to retrieve logs for different database partitions in parallel.
- ▶ The logs for different partitions need to be stored on different tape cartridges. This can be achieved by using a unique Tivoli Storage Manager node name for each database partition. In addition, collocation has to be activated for the tape storage pool of the archived logs.

These settings assure us that logs of different database partitions are stored on different tape cartridges. As the log recovery is done for all database partitions in parallel, all database partitions may request logs at the same point of time during the rollforward operation. Sufficient tape drives need to be available in parallel. For best log retrieval performance, a one-to-one relationship between the number of tape drives and the number of database partitions has to be met.

DB2 UDB database split-mirror

Important: A split mirror is an independent and instantaneous copy of a DB2 UDB database on a second set of disk volumes. The second set of disk volumes can then be attached to a different set of servers (or LPARs), and an identical image of the DB2 UDB database can be started to perform database backups, initial setup of a standby database, or for database system cloning purposes.

To have an independent and consistent image of the DB2 UDB database, all control and data files need to be in the second disk set. This includes the entire content of the database directory and all directories hosting the tablespace containers. To have a consistent state in the second disk set, it is important to ensure that there are no partial page writes occurring on the source database. DB2 UDB provides the ability to suspend I/O for the source database, which allows us to split the mirror while the database is still online, without requiring any further downtime. After a **set write suspend for database <SID>** is issued on all database partitions of the source database, all write activities to the tablespaces are suspended. The FlashCopy operations are started afterwards. To continue with normal operation, the command **set write resume for database <SID>** is issued on all database partitions.

To open the split mirror database by an application or a DBA for maintenance, the database must be initialized first. To do so, the DB2 command **db2inidb** is used. The command can be used to initialize the database for certain purposes. The command **db2inidb <SID> as standby** sets up a shadow database by *log shipping*. This option also takes a DB2 backup with the DB2 **backup database** command.

To be able to access the split mirror database by an application or a DBA for maintenance, the database has to be initialized first. The initialization is done using the **db2inidb** DB2 command. Dependent on the state of the split mirror image and the intended purpose, several options are available for this command:

- ▶ **db2inidb <SID> as standby**

This option sets the split mirror image into rollforward pending state, and so allows a further rollforward of the database applying the logs. It is possible also to perform a DB2 backup with the DB2 backup database command.

► **db2inidb <SID> as snapshot**

The database is initialized, and all in-flight transactions off the current split mirror image are rolled back. This option requires that the active log directory is part of the split mirror image). A new log chain sequence is started immediately, so the relationship between the former source database and the split mirror image is broken. The split mirror database is now fully independent. It is not possible to rollforward any logs from the production DB.

► **db2inidb <SID> as mirror**

This option is used for a fast restore of a database. If the split mirror image taken before is intended for a restore, the active log directory must not be part of it. If the active logs directory would be present in the split mirror image and this image is copied back to the original disks, the last available active logs are overwritten. It would be impossible to recover to the latest point in time, as the logs required for recovery to the actual point in time would be lost.

The instantaneous copy from the first to the second is dependent on the storage vendor's implementation. Using an IBM DS8000 storage subsystem, the split-mirror from the first volume set to the second volume set can be done using FlashCopy.

Table 6-2 resumes the tasks to back up DB2 using a split mirror.

Table 6-2 Backup of DB2 using split mirror

| Task | Performed on | Description |
|--------------------------------|-------------------|--|
| db2> set write suspend for ... | Production server | All write operations on the source DB are stopped. |
| flashcopy | DS8000 | DB directory and file systems containing tablespaces have to be part of the image. |
| db2> set write resume for ... | Production server | Write operations on the source DB allowed. |
| access of flashcopied volumes | Backup server | Import of the volume groups and mount of the file systems. |
| db2inidb ... as standby | Backup server | DB is initialized on the backup server. |
| db2> backup db ... | Backup server | DB is backed up. |

Attention: When using a partitioned database, the following tasks have to be performed for each database partition:

1. Set to write suspend (catalog partition last).
2. Invoke FlashCopy for all relevant LUNs in the storage subsystem.
3. Set to write resume (catalog partition first).
4. Run **db2inidb** on the backup servers.
5. Run **db backup** on the backup servers.

Tivoli Data Protection for Advanced Copy Services are able to manage all these tasks.

Tivoli Data Protection for mySAP 5.4

Tivoli Data Protection for mySAP is used to back up and to restore database contents, control files, and offline DB2 log files.

A backup or a restore operation is started in DB2 via the DB2 Command Line Processor (CLP). CLP interprets the backup/restore commands for the DB2 database and passes control to a DB2 server process. This process triggers the backup or restore, loads the Tivoli Data Protection for mySAP shared library dynamically, and communicates through the vendor API.

- ▶ During a BACKUP DATABASE command, the DB2 server process creates a unique time stamp for the backup, reads the DB2 configuration files, and loads Tivoli Data Protection for mySAP dynamically as a shared library. DB2 buffer manipulator processes read data from the database containers and write the data into the buffers. DB2 media controller processes then move the data blocks from the buffers to the data mover part of Tivoli Data Protection for mySAP. The Tivoli Data Protection for mySAP shared library passes the data blocks to the Tivoli Storage Manager server (via the Tivoli Storage Manager API client). The Tivoli Storage Manager server then writes the data blocks to the storage media (tape or disk devices). At the end of the backup process, the DB2 server process logs the backup in the recovery history file. Backup images stored on the Tivoli Storage Manager server can be queried using the backup object manager query commands.
- ▶ During a RESTORE DATABASE command, the DB2 server process loads Tivoli Data Protection for mySAP dynamically as a shared library and requests the backup data (based on the backup time stamp information). The Tivoli Data Protection for mySAP shared library checks with the Tivoli Storage Manager server if the backup image is available. If available, it retrieves the data blocks from Tivoli Storage Manager and passes them to the DB2 server process. The DB2 server process restores the DB2 data to the database containers and logs the restore in the recovery history file.

The DB2 Log Manager loads Tivoli Data Protection for mySAP as a shared library also. When a log file is ready to be archived, the DB2 Log Manager transfers all the data blocks to Tivoli Data Protection for mySAP. Tivoli Data Protection for mySAP then passes the data to Tivoli Storage Manager. In case of a rollforward recovery operation, the DB2 Log Manager first checks whether the log files are already available in the log path or the overflow log path. If the log files cannot be found, the DB2 Log Manager checks with Tivoli Data Protection for mySAP as to whether the log images can be found on the Tivoli Storage Manager server and requests the data from there. Tivoli Data Protection for mySAP retrieves the data from Tivoli Storage Manager and passes it to the DB2 Log Manager. The DB2 Log Manager writes the log files to the log directory in the file system. The log files are applied to the database during the real part of the rollforward recovery.

Tivoli Data Protection for Advanced Copy Services

Important: The major functionality of Tivoli Data Protection for Advanced Copy Services is to enhance the database backup and restore processes by embedding the FlashCopy capabilities into those processes.

Tivoli Data Protection for FlashCopy is able to manage the entire workflow required for such a procedure and supports backups for DB2 UDB environments for both logical and physical database partitions.

Tivoli Data protection for FlashCopy uses the DS Open Application interface to interact with the FlashCopy services of the DS8000 storage server. The DS Open API is based on the CIM model.

Figure 6-3 describes the component model and the specific hardware and software needed to use the product.

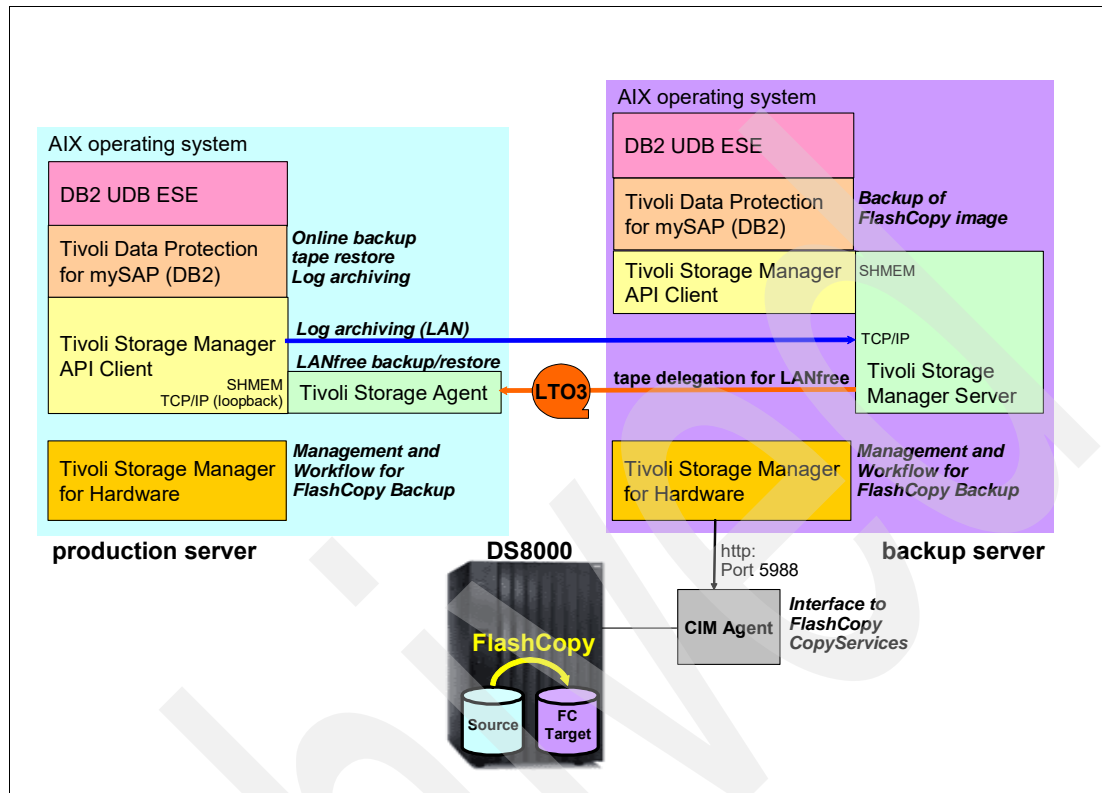


Figure 6-3 Building blocks of the solution

Tivoli Data Protection for Snapshot™ Devices is split into the two components, *splitint* and *tdphdwdb2*.

- ▶ *splitint* represents the part controlled by the storage system, which is database independent.
- ▶ *tdphdwdb2* is the DB2 dependent part and the user interface.

Table 6-3 describes the main functions of the product.

Table 6-3 Description of main functions

| Function | Allowed to run on | Description |
|-----------|------------------------------|---|
| Back up | Production and backup system | Perform a backup of the SAP DB2 UDB database. |
| Restore | Production system only | Restore and recover the production database. |
| FlashCopy | Backup system only | Perform a disk-only FlashCopy of the SAP system DB2 UDB database. |
| Unmount | Backup system only | Free up the disk environment on the backup system. |
| Withdraw | Backup system only | Break the relationship between source/target volumes. |

Important: Tivoli Data Protection for Advanced Copy Services integrates Tivoli Storage Manager server, Tivoli Data Protection for mySAP/DB2, and the DS8000 FlashCopy feature. The intent is to provide a near zero-impact data backup and near-instant recovery solution.

FlashCopy backup to disk only

The production database is set to write suspend mode, and the FlashCopy operations are invoked for all source-target volume pairs. Then the database is set to write resume and continues with normal operation. All blocks of the source volumes are copied to the target volumes during the background copy in one or multiple storage subsystems. Typically, the FlashCopy is scheduled in incremental mode. Only the blocks changed since the last FlashCopy backup are copied. The FlashCopy image of the database (target LUNs) is accessed on one or more backup servers to check the integrity of the volume groups and the file systems. After the background copy is complete, the image is available for a potential FlashCopy restore operation.

The following tasks are performed:

1. `tdphdwb2` determines all the tablespace containers for all tablespaces (using a DB2 client connection to the production database). A list containing all these file names and the local database directory is compiled.
2. For all entries in the list generated in the previous step, all file systems and corresponding disk volumes/LUNs are identified (splitint).
3. `tdphdwb2` sets the DB2 database on the production system into write suspend mode for all the database partitions.
4. FlashCopy is invoked for all database volumes (source/target).
5. `tdphdwb2` sets the DB2 database on the production system into write resume mode. The production system is now fully operational again.
6. The flashcopied image is accessed and mounted on the backup system.

Note: It is possible to keep multiple backup generations on disk. Tivoli Data Protection for Snapshot Devices supports this capability by managing multiple target sets. Configuring multiple target sets as backup generations on disk increases the availability of a best-fit backup on disk for FlashBack Restore.

FlashCopy backup followed by a database backup to Tivoli Storage Manager server

The production database is set to write suspend mode, and the FlashCopy operations are invoked for all source-target volume pairs. Then the database is set to *write resume* and continues with normal operation. Dependent on the scenario, the FlashCopy is started either with "NOCOPY" (transfer of changed blocks to the target volumes only) or with the "COPY" option (start background copy). To be able to use the FlashCopy image for a FlashCopy restore operation, the "COPY" option is mandatory. The FlashCopy is typically scheduled in incremental mode also. The FlashCopy image of the database is accessed on one or more backup servers. On the backup servers, DB2 UDB is started and a database backup of the FlashCopy image is started. DB2 UDB uses IBM Tivoli Storage Manager for ERP Data Protection for mySAP (Tivoli Data Protection for mySAP) as a load library to send the data to the Tivoli Storage Manager API Client during the DB2 backup.

The following tasks are performed:

1. tdphdwb2 determines all the tablespace containers for all tablespaces (using a DB2 client connection to the production database). A list containing all these file names and the local database directory is compiled.
2. For all entries in the list generated in the previous step, all file systems and corresponding disk volumes/LUNs are identified (splitint).
3. tdphdwb2 sets the DB2 database on the production system into write suspend mode for all the database partitions.
4. FlashCopy is invoked for all database volumes (source/target).
5. tdphdwb2 sets the DB2 database on the production system into write resume mode. The production system is now operational fully operational again.
6. The flashcopied image is accessed and mounted on the backup system. The backup system initializes the split mirror image of the database and starts the backup to tape.

FlashBack (quick restore) of the database

Tivoli Data Protection for FlashCopy provides the capability to restore target volumes, created in the FlashCopy backup, to the original source volumes. This is referred to as *FlashBack Restore*. It uses the reverse FlashCopy command of the DS8000 storage subsystem. In contrast to the FlashCopy backup, all database restore activities need to be started on the production system. Figure 6-4 illustrates the FlashBack process.

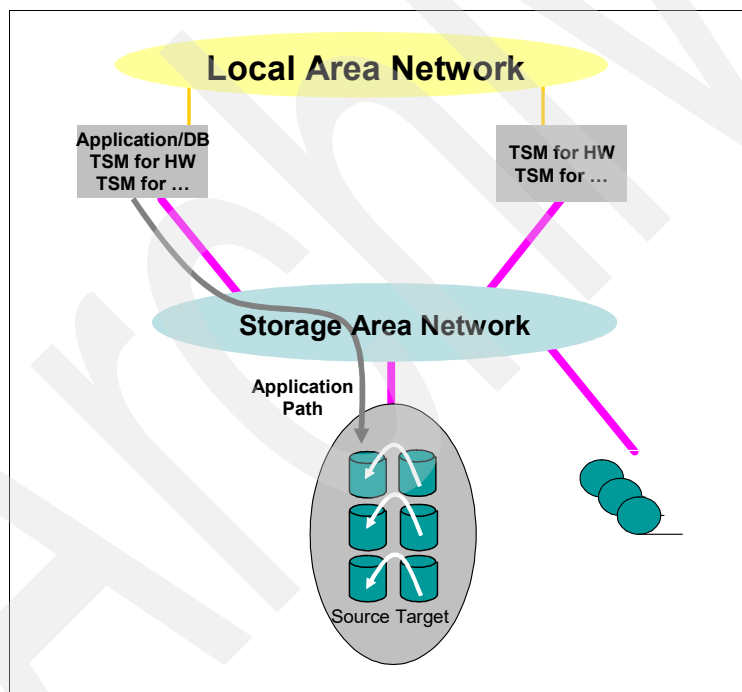


Figure 6-4 *FlashBack (quick restore)*

tdphdwd2 needs to be started on each database partition. It guides you through the restore and recovery process with a menu-driven user interface. Figure 6-5 shows the main menu.

```

IBM Tivoli Storage Manager for Hardware
Data Protection for IBM Disk Storage and SAN VC for mySAP (R) on DB2
-----
Backup History for Database
SystemID: G01
-----
Backup timestamp (ID) Type      TSM FlashCopy RTime(min) 1st active Log
-----
[1] - 21.09.2004 17:55:23 DB online ok running 32.5 S0000010.LOG
[2] - 21.09.2004 17:12:36 DB online ok invalid S0000009.LOG
[3] - 21.09.2004 17:09:58 DB online - ok S0000004.LOG
[4] - 21.09.2004 17:06:06 DB online - invalid S0000003.LOG
[5] - 21.09.2004 16:56:41 DB offline ok invalid S0000002.LOG

[d] - show details
[r] - refresh display
[o] - choose from older backups
[#] - restore the database with line number #
[f] - show FlashCopy backups only (target set state IN_USE)
[x] - exit tdphdwd2

Enter your selection:

```

Figure 6-5 flashback main menu

The following steps are executed:

1. Start a menu from the SAP database administrator.
2. Select the backup to restore.
3. Enter the time stamp for the rollforward.
4. Perform the FlashBack and rollforward.

Important: While Tivoli Data Protection for Snapshot Devices initiates the FlashCopy restore operation, the production system is cleaned up (all DB file systems are unmounted, all DB volume groups are exported). After the cleanup, splitint initiates the FlashCopy from the target volumes (from a previously taken FlashCopy backup) to the production system source volumes. Then all DB volume groups are imported and all DB file systems are mounted again. This greatly reduces the downtime of the production database server.

6.1.3 Components and settings of the backup solution

This section describes the components used in our tests to back up and restore the database.

Tivoli Storage Manager and Tivoli Data Protection design

One LPAR located on one of the p595 servers hosts the Tivoli Storage Manager server. The Tivoli Storage Manager server is connected to the tape library and to all the LTO3 tape drives using FC adapters. The tape library and the drives are shared with the Tivoli Storage Manager Storage Agents. The access and the operation are managed by the Tivoli Storage Manager server. The Tivoli Storage Manager server receives backup data via LAN (client data, archival of the database logs) and stores the data either on disk storage pools or directly on LTO3 cartridges within the tape library. Mass data transfer (database backups/database restores) is delegated to Tivoli Storage Manager Storage agents. A Tivoli Storage Manager agent is installed on each of the LPARs hosting DB2 UDB DPF partitions. During DB2 backup/restore operations the storage agent transfers the data LANfree from/to tape.

Attention: Data mover, storage agent, and backup server all refer to the same AIX partitions used to back up DB2 with Tivoli Data Protection Advanced Copy Services in server-free and lan-free mode.

The distribution of the DB2 UDB DPF partitions is the same for the FlashCopy database image and the production database. 32+1 partitions are distributed across five LPARs.

Figure 6-6 illustrates this Tivoli Storage Manager/TDP configuration.



Figure 6-6 Tivoli Storage Manager and Data Protection design

Each of the five LPARs hosting DB2 UDB DPF partitions is connected to a subset of the tape drives using eight FC adapters. For all DB2 UDB DPF partitions located within one single LPAR, the FlashCopy target volumes are attached to one data mover LPAR (storage agent). The data mover LPAR is installed on the same physical p595 server as the LPAR hosting the production database and has the DB2 UDB ESE and Tivoli Storage Manager Storage Agent software installed also. The DB2 backup of the FlashCopy image is done in the same manner as an (online) backup happens on the production LPARs. The workflow of the FlashCopy invocation and the backup of the FlashCopy database are controlled by Tivoli Storage Manager for Hardware and Tivoli Data Protection for Enterprise Resource Planning.

Figure 6-7 shows the configuration details for one System p595 server.

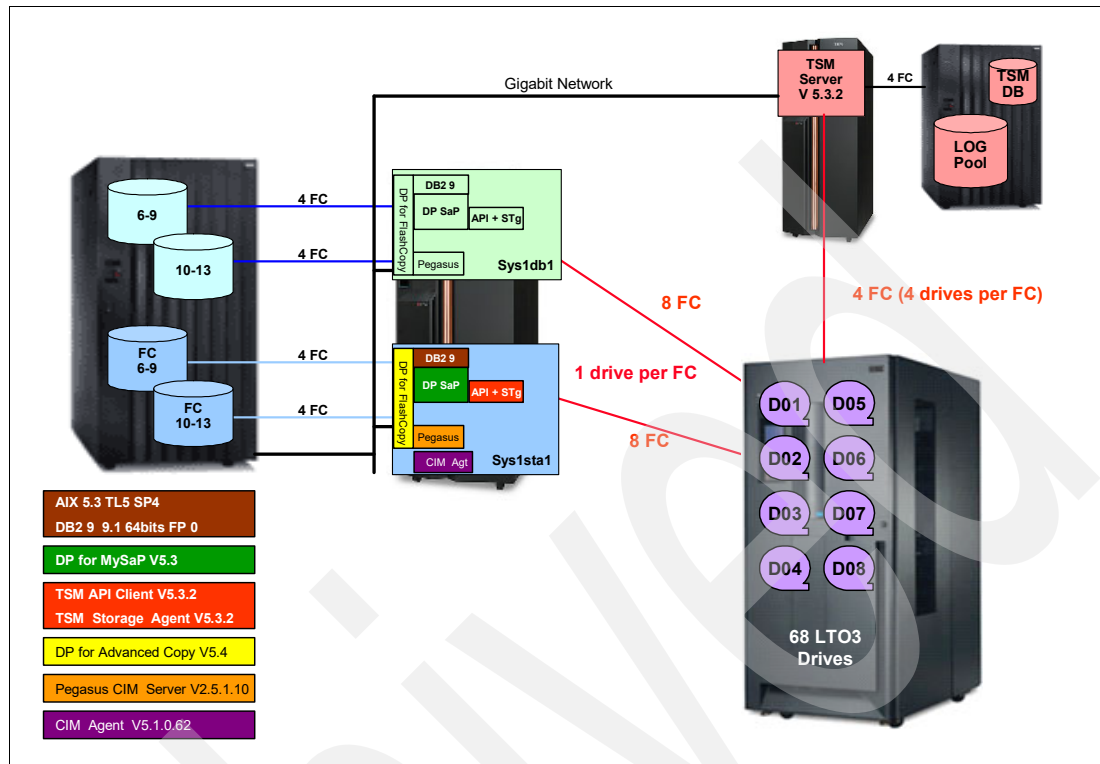


Figure 6-7 Architecture detail for one p595

The data flow during the different backup and restore processes is the following:

1. For the DB2 data backup:
 - a. FlashCopy from the sources volume to the target volumes.
 - b. The target volumes are mounted on the data mover LPAR.
 - c. DB2 backup from data mover to tapes using lanfree.
2. For the DB2 restore data:
 - a. Restore from tape to the production server using lanfree.
 - b. Restore to the sources volumes.
3. For the DB2 restore from disk:
 - a. Reverse flashcopy from the target to the source volumes.
4. For the DB2 log backup:
 - a. The production server archives the logs to the Tivoli Storage Manager server.
 - b. The Tivoli Storage Manager server stores the logs on a disk storage pool (LOGPOOL).
 - c. The logs are migrated to tape when a high threshold is reached.
5. To retrieve log from the disk storage pool:
 - a. The logs are retrieved on the Tivoli Storage Manager server from the disk storage pool.
 - b. They are transferred from the Tivoli Storage Manager server to the production server.
6. To retrieve the log from tape, the logs are retrieved directly from tape using lanfree on a production server.

Figure 6-8 illustrates these processes.

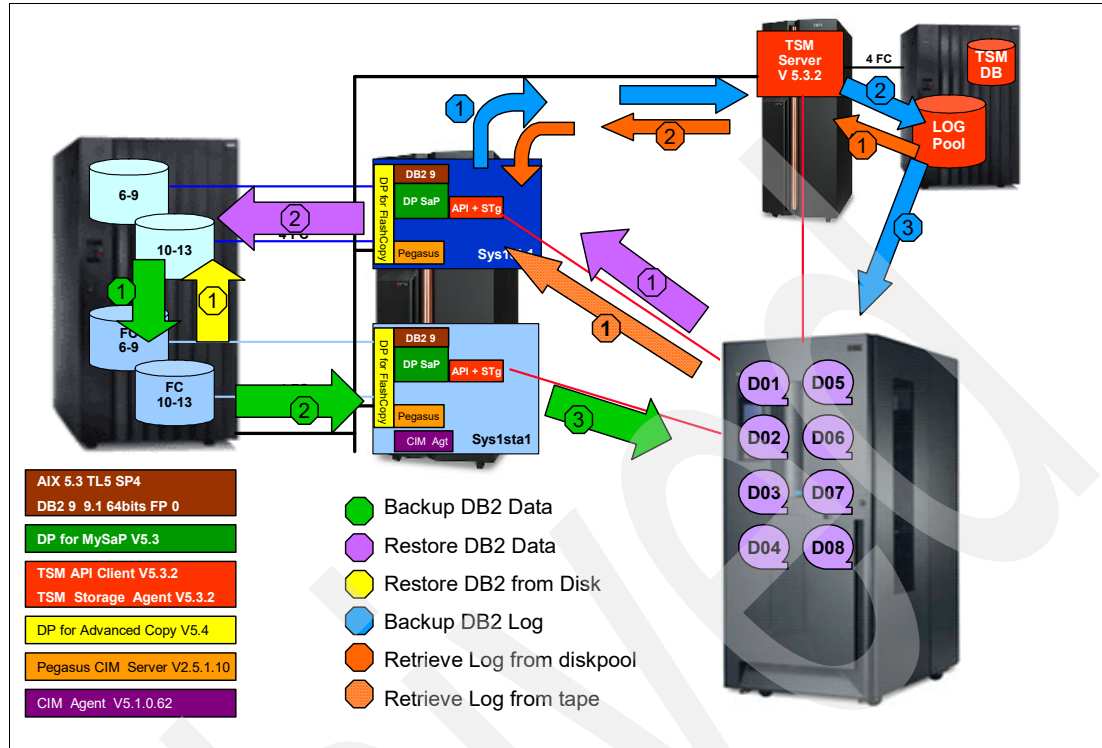


Figure 6-8 Data flows during the backup and restore process

Tivoli Storage Manager Server configuration

For the Tivoli Storage Manager Database and Recovery log, the Tivoli Storage Manager database size is 15 GB and the recovery log size is 5 GB. Both are located on a DS8000 and mirrored in order to provide a safeguard against media failure.

Figure 6-9 illustrates the Tivoli Storage Manager policy.

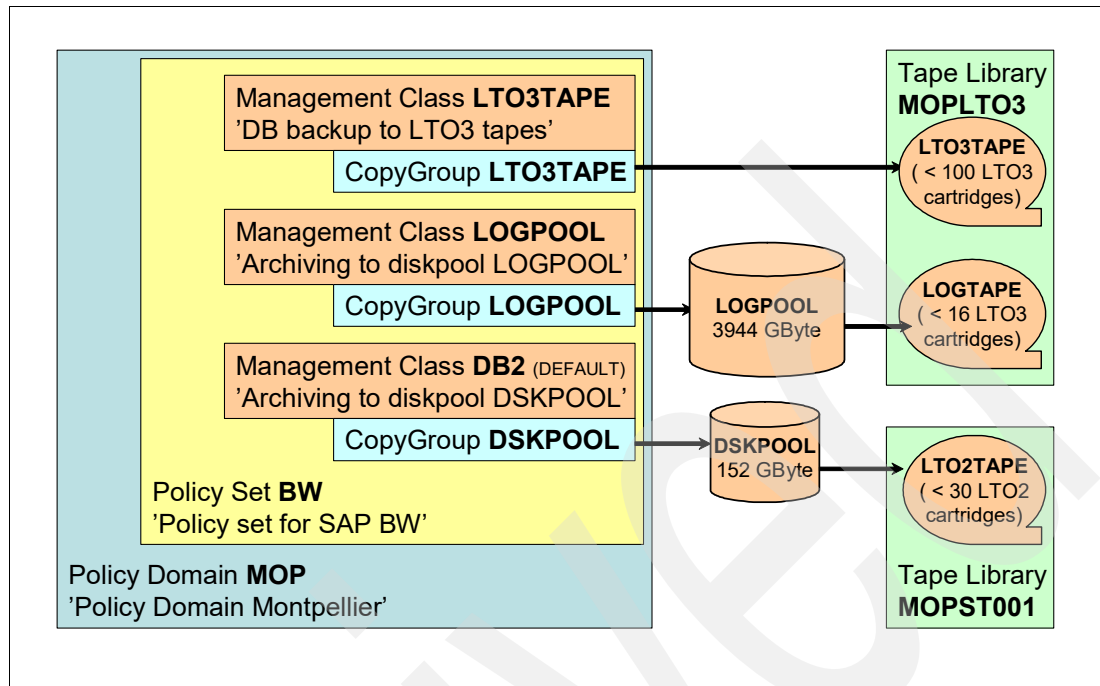


Figure 6-9 Tivoli Storage Manager Policy overview

For the Tivoli Storage Manager policy, we create two domains:

- ▶ The *BACKUP_SYS* domain for the incremental backups of our systems (at the file level) using the Tivoli Storage Manager client

The *BACKUP_SYS* management class is assigned as the default for the incremental backup of the systems (at the file level) using the Tivoli Storage Manager client. The backup copy group of this management class uses the *DSKPOOL* storage pool.

- ▶ The *MOP* domain for the DB2 backup and log archiving using Tivoli Data protection
 - The *LTO3TAPE* management class is defined for the database backup using the *LTO3TAPE* storage pool.
 - The *LOGPOOL* management class is defined for the log archiving using the *LOGPOOL* storage pool.

We create the following storage pools:

- ▶ *LTO3TAPE* use for the database backup
- ▶ *LOGPOOL* use for the log archiving and migrate to *LOGPTAPE*
- ▶ *LOGPTAPE* use for the log archiving
- ▶ *DSKPOOL* use from the incremental backups of the systems and migrate to *DSKTAPE*
- ▶ *DSKTAPE* use from the incremental backups of the systems

An IBM 3584 Tape Library is used with:

- ▶ One base frame
- ▶ Five expansion frames
- ▶ Sixty-eight LTO3 drives
- ▶ Sixty-eight Fibre Channel attachments

The Tivoli Storage Manager server has to manage the library inventory and the mount/dismount of the volumes during a lanfree backup or a TSM process (reclaim,

migration, db backup, and so on). To do that we define for the Tivoli Storage Manager server one path to the library and one path per drive.

The managed system for the SAN storage agent is a feature of Tivoli Storage Manager that enables the LAN-free Tivoli Storage Manager client. When this storage agent is installed on a backup server (or data mover) it also enables server-free data movement. The Tivoli Storage Manager client moves the data directly to/from a storage device attached to a storage area network (SAN), thereby off-loading the data movement from the LAN and from the Tivoli Storage Manager server.

The storage agent is installed on the client machine and shares storage resources with the Tivoli Storage Manager server. The storage agent can write directly to the storage media in a format that is consistent with that used by the server. The Tivoli Storage Manager server controls the storage devices and keeps track of the data that the client stored. Tivoli Storage Manager continues to use a LAN connection to exchange control information, such as policy information and data about the objects that are backed up. Using the SAN for client data movement decreases the load on the Tivoli Storage Manager server and allows the server to support a greater number of simultaneous client connections.

In order to be able to back up 32 DB2 nodes in parallel and to optimize the lanfree backup we create:

- ▶ Five storage agents, one per data mover LPAR
- ▶ Eight paths per storage agent, using eight tape drives and eight Fibre Channel attachments (one tape drive per FC attachment)

We also install the storage agents on the production servers in order to back up or restore from these servers using LAN-free.

The storage agents do not need a path to the library because the mount and dismount process is managed by the Tivoli Storage Manager server.

Tivoli Storage Manager Client API configuration

The Tivoli Storage Manager APIs are used either by Tivoli Data Protection for mySAP or Tivoli Data production for Advanced Copy Services to link them with the Tivoli Storage Manager server.

This configuration file is also located in the /db2/EB8/dbs/tsm_config folder.

The Tivoli Storage Manager Client API configuration we used is described in “The Tivoli Storage Manager API configuration file” on page 408.

Tivoli Data Protection for mySAP 5.4

Tivoli Data Protection for mySAP is installed on each production and data mover LPAR.

To share the same configuration files between the five production servers, we create an NFS file system, and the files are located in the /db2/EB8/dbs/tsm_config folder.

The vendor environment file (vendor.env) must contain the full qualified path and file name of the file init<SID>.utl and should contain the path to which the Tivoli Data Protection for mySAP run logs will be written.

Example 6-1 contains the vendor.env file we used.

Example 6-1 vendor.env

```
XINT_PROFILE=/db2/EB8/dbs/tsm_config/initEB8.utl  
TDP_DIR=/db2/EB8/dbs/tsm_config/tdplog
```

The XINT_PROFILE parameter refers to the Tivoli Data Protection for mySAP configuration file described in “The Tivoli Data Protection for mySAP configuration file” on page 396.

Tivoli Data Protection for Advanced Copy Services 5.4

Tivoli Data Protection for Advanced Copy Services is installed on each production and data mover LPAR.

The initEB8.fct configuration file contains setup information about source/target volumes, as they will be used in the FlashCopy function. This is described in “initEB8.fct” on page 399.

The initEB8.fcs configuration file contains the setup information. It is described in “The Tivoli Data Protection ACS configuration file” on page 399

These files are related to a production server and located in the /db2/EB8/dbs/sys1db(x) folder, where sys1dbx represents one of the five production servers, namely from sys1db0 to sys1db4.

CIM Agent/DS8000 open API 5.1.0.62

The Common Information Model (CIM) agent is a set of standards that was developed by the Distributed Management Task Force (DMTF). The CIM provides an open approach to the design and implementation of storage systems, applications, databases, networks, and devices. The CIM defines common object classes, associations, and methods. It contains:

- ▶ The CIM agent server

A CIM agent runs on a dedicated server and contains the CIM agent software and its components.

- ▶ The CIM agent client

A CIM agent client is an application programming interface (API) that manages storage and initiates a request to a device or a data storage server such as the IBM System Storage DS8000.

- ▶ The CIM agent object classes

Object classes are the building blocks of the CIM agent and provide functionality such as copy services, storage configuration, or logical unit number (LUN) masking.

- ▶ CIM Install

We install the CIM agents as described in the documentation "DS Open Application Programming Reference" available at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/index.jsp?topic=/com.ibm.itsmfacs.doc/fbrdb2m031.htm>

Important: We use one CIM Agent per DS8300 to decrease the time to start FlashCopy and resolve timeout errors. This agents are located on the data mover lpar.

6.1.4 Backup and restore processes

This section discusses the solutions provided by Tivoli to back up and restore the data.

Backing up with Tivoli Data Protection for mySAP

To back up using Tivoli Data Protection for mySAP, the sequence is the following:

- ▶ The catalog partition has to be backed up first.
- ▶ All other database partitions can be backed up afterwards, either sequentially or in parallel.

The scripts have been written for a parallel backup using 32 tape drives.

The `do_backup` script is the main script and is used to start the backup. It is described in “The main backup script” on page 400. We refer to other scripts, described in “The script for the backup of node 0” on page 404.

Restoring with Tivoli Data Protection for mySAP

Prior to the restore, the source database is dropped to simulate a full recreation of the database. The restore sequence is the following:

1. Use the `backom` command to get the time stamps of the backup (one time stamp per DB2 node), as shown in Example 6-2.

Example 6-2 backom query

```
su - db2eb8
backom -c q_db
```

2. The DB2 catalog partition has to be restored first.
3. All other database partitions can be restored afterwards, either sequentially or in parallel.

We create the scripts for a parallel backup using 32 tape drives. The restore script is the main script. It is described in “For the node 7 restore” on page 404. It is used to start the scripts of the other partitions, described in “For the node 7 restore” on page 404.

Backing up with Tivoli Data Protection for Advanced Copy Services

For each DB server LPAR hosting one or more database partitions, one Tivoli Data Protection for FlashCopy process (`tdphdwb2`) needs to be started. All these processes have to be started in parallel for a FlashCopy backup run. Because the production database is distributed across five LPARs, five `tdphdwb2`s are started in parallel in our case.

We create the script shown in Example 6-3 to start the process for the five partitions.

Example 6-3 Start of a FlashCopy and tape backup

```
#!/bin/ksh

host=$(hostname)
if [[ "${host}" = "sys1sta0" ]]
then
    export RAHBUFNAME=sys1sta0.buf
    cd /db2/EB8/dbs
    ./tdphdwb2 -f backup -t flashcopy unmount -p /db2/EB8/dbs/sys1db0d/initEB8.fcs
> /db2/EB8/dbs/sys1sta0.log &
    for i in 1 2 3 4; do
```

```

ssh -l db2eb8 sys1sta${i} ". /db2/db2eb8/.profile; export
RAHBUFNAME=sys1sta${i}.buf; cd /db2/EB8/dbs; ./tdphdwb2 -f backup -t flashcopy
unmount -p /db2/EB8/dbs/sys1db${i}d/initEB8.fcs > /db2/EB8/dbs/sys1sta${i}.log" &
done
else
echo "script $0 has to be run on sys1sta0 !"
fi

```

After the processes are started, one individual FlashCopy backup run using Tivoli Data Protection for FlashCopy can be structured in several phases:

- ▶ The pre-checks
- ▶ The FlashCopy invocation
- ▶ The access to the backup servers
- ▶ The backup to tapes on the backup servers
- ▶ The cleanup

Phase 1: the pre-checks

In a first phase, Tivoli Data Protection for FlashCopy checks:

- ▶ The configuration on the DB servers
- ▶ The state of the DB2 database state
- ▶ The state for all FlashCopy source-target volumes relationships

The (initial) state of the FlashCopy target volumes influences the duration of phase 1. In most cases, the FlashCopy backup is run in incremental mode.

- ▶ Running it the first time without having any existing FlashCopy source-target relationships, there is no need to check the status of all pairs.
- ▶ Repeating the FlashCopy for a second time (with all incremental FlashCopy relationships available), the check of all source-target pairs states takes longer. Checking all the 280 source-target relationships for the LUNs in our test setup takes about 30 minutes.

In our test, for this phase, all the relationships had been withdrawn, which means that no incremental FlashCopy source-target pairs are available, and so the checks are not done. After about 10 minutes, in our environment, all checks are finished and Tivoli Data Protection for FlashCopy initiates a second phase.

Phase 2: the FlashCopy invocation

In the second phase, Tivoli Data Protection for FlashCopy invokes the FlashCopy for all relevant DB volumes. All DB partitions are set to write suspend first. The FlashCopy operations are then started for all volumes. Tivoli Data protection sends the FlashCopy commands to the CIM agent that forwards them to the storage server. After the FlashCopy is invoked for all LUNs, the DB partitions are set to *write resume* and the database continues with normal operations.

The total duration of this off-line step is about a few seconds and depends on the actual workload on the database because the DB2 write suspend commands have to compete with the current workload on the database server. With a high workload on the database server, it may take more time to get the commands executed on all partitions (up to a minute). Compared to the duration of the backup to tape, this is a negligible amount of time. However, it impacts the off-line time (where the database is in write suspend mode). The production database is fully operational again, and Tivoli Data Protection for FlashCopy initiates the third phase.

Phase 3: access to the FlashCopied storage on the backup servers

In the third phase, the FlashCopied storage image is accessed on the backup servers. To avoid locking issues on the devices, the five `tdphdwd2` processes synchronize themselves and run sequentially. Each `tdphdwd2` process runs AIX configuration manager processes ("`cfgmgr`"). When all FlashCopy LUNs are available for the relevant storage set, the volume groups are redefined and varied online. Then all the file systems are mounted.

The duration of this phase depends on the number of storage elements (LUNs/disk devices, number of volume groups, and number of file systems). To optimize the overall process, a compromise between the time required to access the disks and the throughput during the backup has to be found. In our environment, two paths are defined to the LUNs.

In the test environment (having the 60 TB database distributed across five AIX LPARs across a large number of file systems and LUNs and using SDDPCM as multipathing driver), it takes about 40 minutes in total to discover all the disk devices, vary the volume groups, and mount all database file systems on the five backup servers.

If the FlashCopy is performed in "COPY" mode, and when this phase 3 is finished successfully, the backup image is made available for a FlashCopy restore.

Phase 4: the backup to tape on the backup servers

If you intend to run the backup of the Flashcopy to tape, this part is done in this phase. Tivoli Data protection for FlashCopy runs "`db2start`" and then issues the "`db2inidb as standby`" command for all database partitions. The split-mirror image is then prepared to take a backup. From a functional perspective, the database backup scheduled afterwards is similar to running an online backup of the production database to tape. The configuration capabilities available (for example, the number of tape drives available for backup purposes) influence the duration.

The backup of the coordinating partition (partition 0) has to be completed first before starting a backup for any other partition. When the backup of partition 0 is done, all remaining database partitions can be backed up. This can be done either sequentially or in parallel. The overall scheduling behavior for the remaining backups (except partition 0) is controlled by the Tivoli Data Protection profile parameter `DB2_EEE_PARALLEL_BACKUP`:

- ▶ If `DB2_EEE_PARALLEL_BACKUP` is set to NO (default setting), all database backups aligned to one `tdphdwd2` process are done sequentially. For our test, five `tdphdwd2` processes are running in parallel. The backups scheduled out of these different processes run in parallel.
- ▶ If `DB2_EEE_PARALLEL_BACKUP` is set to YES (and the backup of the coordinating partition is finished), all remaining DB2 backups are started in parallel. To be able to run all backups in parallel, sufficient tape drives need to be available. In our test environment, 16 LTO3 tape drives were available, which is less than the number of the partitions (32 partitions, with the partition 0). However, it is possible to achieve the start of 32 backups in parallel with 16 tape drives. `DB2_EEE_PARALLEL_BACKUP` has to be set to YES, and the Tivoli Storage Manager server options `RESOURCETIMEOUT` and `MAXSESSIONS` need to be set to a sufficient high value. All backups will then be started in parallel. However, the second half of the backups need to wait for the first half to be complete to get access to a tape drive. For a production environment, however, we do not recommend *overbooking* tape drive resources in such a manor.

If the backup of one set (out of one `tdphdwd2` process) is done with the same parameterization, it is sufficient to specify this value in the Tivoli Data Protection configuration profile. If individual *sessions* settings for the backups of different database partitions are used, all sessions values for all database partitions need to be set in each Tivoli Data Protection configuration profile.

Phase 5: the cleanup

Tivoli Data Protection then starts cleanup actions on the backup server: unmounting file systems, exporting volume groups, and removing disk devices.

Restoring with Tivoli Data Protection for Advanced Copy Services (FlashBack)

On each of the five LPARs hosting the database partitions, one `tdphdwd2` process is started. The five processes have to be started in parallel. They synchronize themselves among each other. All of them require some user input, so they cannot be started in background mode.

Example 6-4 shows how `tdphdwd2` can be called for a restore.

Example 6-4 restore with TDP/ACS

```
./tdphdwd2 -f restore -p /db2/EB8/dbs/tsm_config/initEB8.fcs
```

1. Tivoli Data Protection for FlashCopy checks the backup log for all available backups (either the last FlashCopy or a tape backup) and the configuration on the DB servers. One of the valid and available backup images is then selected. Tivoli Data Protection for FlashCopy checks all FlashCopy source-target volumes relationships, the file systems, and so on, to validate the status of the selected backup image. The total run time of this phase in our environment was about 15 minutes.
2. Tivoli Data Protection then queries the user before starting the *real* FlashCopy restore procedure. DB2 is stopped. all file systems are unmounted and removed. All volume groups are exported.
3. The inverse FlashCopy is then started for all volumes. All disk devices are (re-)acquired. The volume groups are imported and varied on. All file systems are mounted. This second step takes about 25 minutes in our environment. About 40 minutes after the start of the procedure, all file systems are available again.
4. Afterwards, DB2 is started again and a "db2inidb as mirror" is run for all database partitions. This last step runs very quickly, so about 55 minutes after the start of the procedure in our environment, TDP processing is finished and rollforward recovery can be started.

6.2 Options and parameters discussions

This section discusses some of the options available to optimize the backup and restore of the data.

6.2.1 Backup: parallel versus sequential

We made this test, parallel versus sequential backup, in order to choose the good settings from a performance point of view.

Parallel backup

Backup performance of the parallel backup is better compared to the sequential backup. The parallel backup for the 60 TB database had a run time of about 4 hours 25 minutes. This corresponds to a backup rate of 1100 MBps. During a parallel backup, all eight DB2 partitions of one LPAR are handled in parallel. The total I/O is distributed across 512 disks during the entire run time of the backup. This leads to a very good load distribution across the disk

subsystem. For each database partition the backup is executed using one session. One dedicated tape drive is required for each database partition. If a backup of one single partitions fails, another full total backup time is required to repeat the failing one and complete the full backup of all partitions. Extra tape drives may be required as a contingency to avoid the required repetition of a backup due to missing resources.

Figure 6-10 illustrates the parallel backup execution.

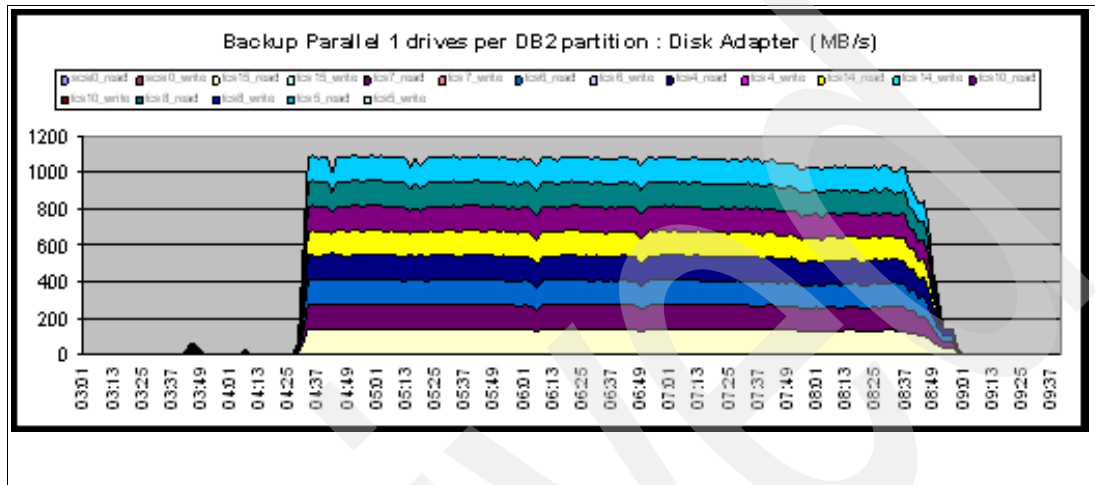


Figure 6-10 Parallel backup

Sequential backup

Price/availability for the sequential backup is better compared to the parallel backup. The parallel backup for the 60 TB database had a run time of about 7 hours 25 minutes. This corresponds to a backup rate of 700 MBps. During a sequential backup, eight DB2 partitions of one LPAR are handled in sequential order. As one database partition is handled at one point of time, the total I/O is distributed across 64 disks during run time. I/O throughput from the disks is limited by the four FC adapters involved. Planning for the number of tape drives depends on the size of the backup window to be achieved. If a backup of one single partition fails, only a subset of a full total backup time (1/8) is required to repeat the failing one and complete the full backup of all partitions. Having eight tape drives available, one failed backup could be repeated in about one hour.

Figure 6-11 illustrates the sequential backup execution.

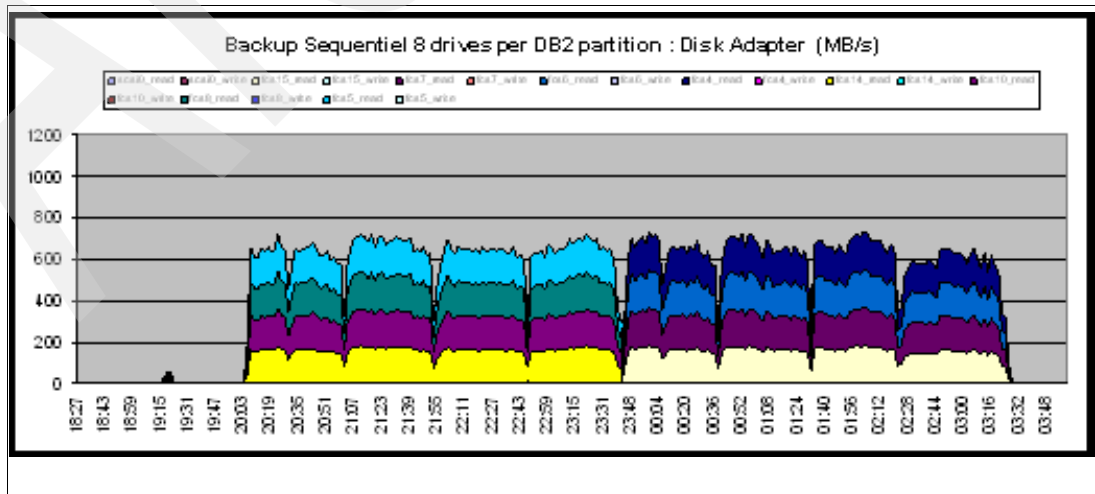


Figure 6-11 sequential backup

Conclusion: In our SAN and DS8000 configuration, the parallel backup is more performant than the sequential backup. Therefore, the parallel backup has been used.

6.2.2 Design consideration

The number of drives for a backup depends on:

- ▶ The number of LPARs, for sequential backup
For example, with DB partition 0 on one LPAR, other DB partitions on four LPARs, the possible number of drives are 4, 8, 12, 16, ... 32, 36, and so on.
- ▶ The number of DB2 partitions, for parallel backup
For example, with DB partition 0 and 32 DB partitions, the possible number of drives are 32, 64, and so on.

During the *for fast roll* forward we need:

- ▶ One drive per DB partition (including DB partition 0), with collocation. In our configuration, we needed 33 drives.
- ▶ Large diskpool for the log to avoid tape access. Keep in mind to keep logs between two backups at least.

The LTO3 throughput is about 140 MBps. In order to achieve this throughput per drive we need:

- ▶ To dedicate one FC adapter (2 GB) per LTO3 drive
- ▶ To dedicate enough FC adapter on the data mover LPAR to the disk subsystem
- ▶ To use a dedicated FC adapter on the disk subsystem for the data mover LPAR to avoid contention with other LPAR workload

Depending on the number of sessions used during a database backup/restore, one FC adapter will drive one or up to two LTO3 tapes. Although the p595 servers have FC adapters capable of 4 GBps, the SAN backbone limits them to 2 GBps. The experience shows that one LTO3 drive will be driven at about 140 MBps. Having two LTO3 drives operated by one adapter, the drives will be limited to 75–90% of their throughput.

6.2.3 Multiple volume sets

To address the requirement for minimum outage in database restoration, the time between backups must be decreased (having fewer log files to apply reduces forward recovery duration). In order to maintain the capability for FlashBack Restore, multiple backup generations must be kept on disk. Tivoli Data Protection for Snapshot Devices supports this capability by managing multiple target sets. More than one set of target volumes can become the target set (or data container) for one disk copy backup of the source volumes identified when running the SAP DBA tool brbackup together with Tivoli Data Protection for Snapshot Devices. Configuring multiple target sets as backup generations on disk increases the availability of a best-fit backup on disk for FlashBack Restore.

Having included more than one target set in a backup/restore strategy, the DBA, depending on his strategy and schedules, needs a capability to decide which target set is to be used for the different FlashCopy backups. Tivoli Data Protection DP for Snapshot Devices gives the option, for each planned backup, to choose between two selection algorithms:

- ▶ Specific target selection
- ▶ Automated target set selection

The Tivoli Data Protection for Snapshot Devices target volumes file (.fct) contains the information about the number of target sets that can be used. The file is made up of multiple volumes_set_x topics. Each topic comprises all the target volumes that will be used within one FlashCopy backup. These volumes make up one target set.

6.2.4 Multiple nodenames configuration

In the Tivoli Data Protection configuration files we use a unique Tivoli Storage Manager node name for all the 38 DB2 nodes: EB8_SYS1. However, it is possible to use one Tivoli Storage Manager node name per DB2 node.

This *multiple nodenames* configuration, combined with tape storage pool collocation, reduces the number of tape mounts and mount waits during a tape restore (and a roll forward of the database (logs retrieve)), improving the overall process of recovery.

To set up such a configuration we have to:

1. Register the 38 node names on Tivoli Storage Manager server EB8_SYS1_xx (xx is the DB2 node from 00 to 37).
2. Create 38 entries in dsm.sys, referring to the 38 nodenames.
3. Create 38 initEB8.utl.## files (## is the DB2 node from 00 to 37).
4. Create 38 vendor.env.## files (## is the DB2 node from 00 to 37).
5. Modify the initEB8.fcs files with the new .utl files.

Example 6-5 shows the different configuration files used for the DB2 node 07.

Example 6-5 vendor.env.07

```
DSMI_CONFIG=/db2/EB8/dbs/tsm_config/dsm.opt
DSMI_LOG=/db2/EB8/dbs/tsm_config
DSMI_DIR=/db2/EB8/dbs/tsm_config
XINT_PROFILE=/db2/EB8/dbs/tsm_config/initEB8.utl.07
TDP_DIR=/db2/EB8/dbs/tsm_config/tdplog
DB2_VENDOR_LIB=/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a
```

In the .utl file, the SERVER parameter only is modified, as shown in Example 6-6.

Example 6-6 finitEB8.utl.07

```
SERVER      detsm05_07
```

The TDPR3_CONFIG_FILE parameter refers to multiple utl files, as shown in Example 6-7.

Example 6-7 initEB8.fcs

```
TDPR3_CONFIG_FILE      /db2/EB8/dbs/tsm_config/initEB8.utl.##
```

Example 6-8 shows the dsm.sys file to the DB2 node 07.

Example 6-8 dsm.sys

```
servername      detsm05_07
asnodename EB8_SYS1_07
nodename        EB8_SYS1
passworddir     /db2/EB8/dbs/tsm_config
enablelanfree   YES
```

| | |
|---------------------|-----------|
| resourceutilization | 5 |
| lanfreecommmethod | SHAREDMEM |
| lanfreshmport | 1513 |

6.3 Infrastructure tests

This section summarizes the tests results.

6.3.1 KPI-3A: back up 60 TB in less than 12 hours

The purpose of this test is to show the feasibility of backing up a FlashCopy image of the 60 TB database in reasonable time to tape using Tivoli Storage Manager. Achieving a high throughput rate avoids the need for multiple FlashCopy target sets within the DS8000 storage server. The backup will be archived completely to tape before starting the next backup cycle on the FlashCopy volumes. The test simulates a cycle backup window of 12 hours, so it would be possible to take up to two backups per day.

To meet this test the following must be achieved: backup of a 60 TB database (located on flashcopy volumes) to tape in less than 12 hours. A full copy FlashCopy background copy does not need to be complete at the end of the 12 hours. During production only a small amount of data is changed (compared to the total amount of the database) per day, and by using incremental FlashCopy, the background copy of a daily cycle will of course be finished within 12 hours.

Approach

The management of both FlashCopy process and database backup up to 32 LTO3 tapes in parallel is managed by Tivoli Storage Manager for Advanced Copy Services. After the database I/O is suspended, FlashCopy operations are invoked and all database partitions resume their productive operation. The LUNs containing the FlashCopy image of the database will be accessed by further LPARs. As all database partitions are distributed across five LPARs. A symmetrical set of five LPARs will access the FlashCopy target volumes. The backup of the FlashCopy database image is started as "lanfree" backups, all in parallel to access up to 32 LTO3 tape drives in parallel. All backups are started in parallel with one session (32 LTO3 tapes).

The following software has been used:

- ▶ DB2 UDB 9.1. FP0
- ▶ Tivoli Data Protection for mySAP (DB2 UDB) 5.4.0.0
- ▶ Tivoli Data Protection for FlashCopy Devices for mySAP - DB2 5.4.0.0
- ▶ CIM agent for the IBM TotalStorage DS Open API 5.1.0.62

The following tools have been used:

- ▶ NMON
- ▶ DS8000 Monitoring tools
- ▶ DB2 Performance Expert, to monitor the DB2 backup/restore buffer quality and cache overflows.

The prerequisites for this tests are:

- ▶ The 60 TB production database is distributed across multiple database partitions and multiple logical server partitions.
- ▶ The database file systems are distributed across different volume groups, fulfilling the requirements of Tivoli Storage Manager for Hardware.
- ▶ The FlashCopy feature is activated on the DS8000 storage box, and the CIM server is set up and operational.
- ▶ Tivoli Storage Manager Server is installed and customized on one logical server partition or a standalone server.
- ▶ Tivoli Storage Manager Server accesses the tape library and 64 LTO3 tape drives with a sufficient amount of LTO3 scratch volumes in the tape library.
- ▶ Tivoli Data Protection, Tivoli Storage Manager StorageAgent, and DB2 UDB are installed on multiple LPARs. They have access to the tape library and a subset of the LTO3 tape drives (data mover).
- ▶ For the SAN setup and zoning, the source volumes are individually assigned to the multiple database partitions, and the target volumes are assigned to the multiple data mover partitions.
- ▶ Tivoli Storage Manager for Hardware software is installed and customized on all logical partitions of the DB partitions and on the data mover LPARs.
- ▶ Tivoli Storage Manager for Enterprise Resource Planning is installed and customized on all logical partitions of the database partitions and on the data mover LPARs
- ▶ A Flashcopy of the database was taken using Tivoli Storage Manager for Hardware and this FlashCopy still exists on the target volumes (incremental).
- ▶ During the tests there is no other workload.

KPI-3A results and analyses

The test is completed successfully in 6 hours and 10 minutes. Table 6-4 describes the duration of each task.

Table 6-4 KPI-3A - process duration

| Time | Task |
|-------|--|
| 00h | TDP starts on all storage agents. (one process for each LPAR hosting DB partitions) TDP checks configuration and source-target volumes relationships. |
| 00h17 | TDP invokes FlashCopy for DB volumes. Set database to „write suspend“ for all DB partitions. Start FlashCopy for all LUNs via CIM Agent. Set database to „write resume“ for all DB partitions. (some 10 seconds „offline“ time) |
| 01h06 | Mount FC LUNs on storage agents done. Acquire disk devices, volume groups, mount file systems. Run db2inidb as standby. Start backup of partition 0 (four sessions). |
| 01h27 | Backup of partition 0 finished. Start all backups for remaining partitions in parallel. Each individual backup is using one tape: (32 + 5) backups * 1 session = 37 tapes used |
| 06h01 | Backup for all partitions done. |

| Time | Task |
|-------|--|
| 06h10 | Cleanup on storage agents done. Unmount file systems, export volume groups, remove disk devices. |

Figure 6-12 highlights the three tasks that generate the I/O read throughput on storage agents and the duration of each step of the full process:

1. The FlashCopy tasks (sequential on the storage agents)
2. The tape backup of partition 0 (sys1sta0)
3. The parallel backup of the remaining partitions (sys1sta1 to sys1sta4)

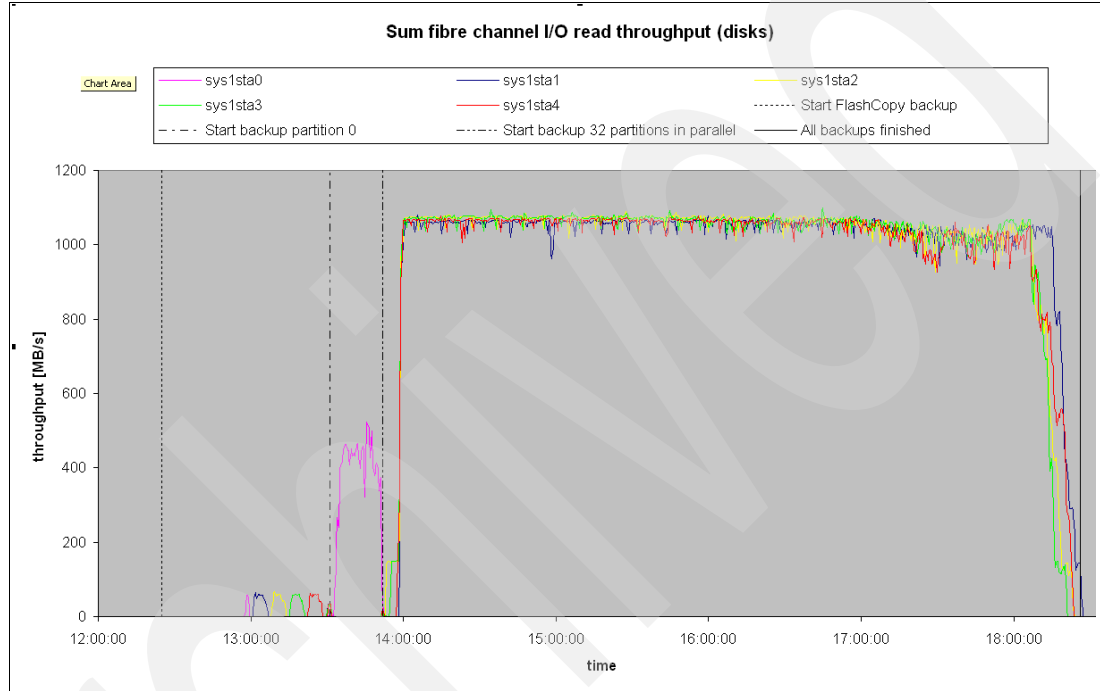


Figure 6-12 Sum Fibre Channel I/O read throughput (disks)

Figure 6-13 illustrates the throughput achieved with the tape storage. The first two data points show the time when the DB partition 0 is backed up (DB2 requires that the catalog partition be backed up before backing up the other partitions). The software was configured to have four tape drives in parallel to back up partition 0. Once partition 0 is backed up, we see the increase in the total throughput when the 32 data partitions are backed up. There is one tape drive per partition. The numbers for this figure were obtained from the SAN256M. The average throughput per tape drive during this four-hour period was 142 MBps with a standard deviation of 5 MBps.

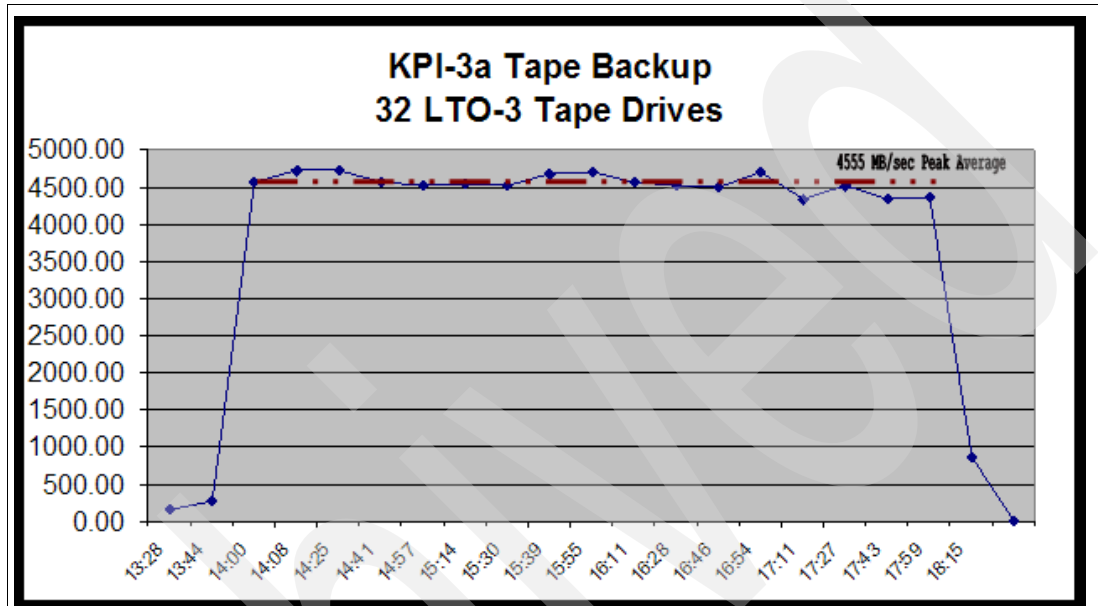


Figure 6-13 KPI-3a tape backup average peak throughput

Figure 6-14 compares two different ports. Port 2 was connected to one of the four tape drives used to back up partition 0. Port 2 started before port 4, and its throughput was lower than port 4's. After the drive connected to port 2 finished backing up partition 0, it was used for one of the 32 DB partitions. During the backup of the data partitions, the tape drive throughput was about 142 MBps.

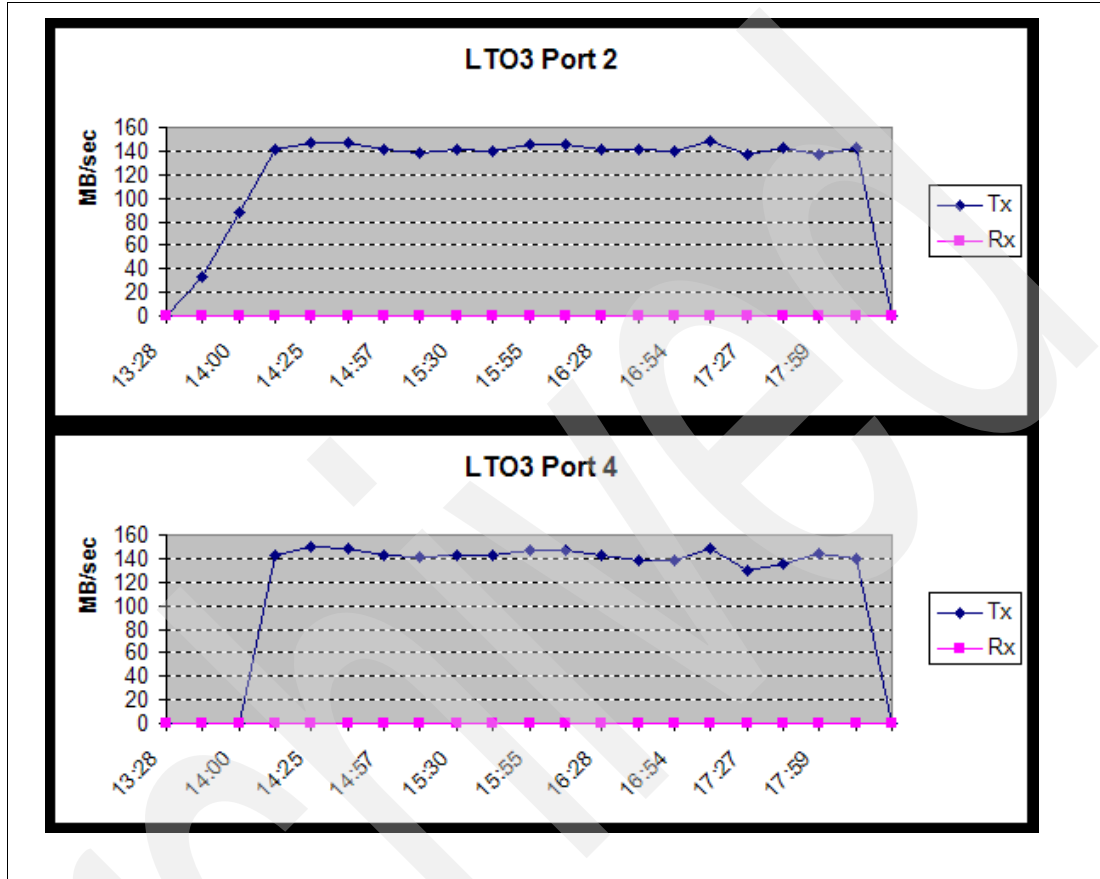


Figure 6-14 LTO-3 ports backing up the database

On one data mover (storage agent) we back up eight DB2 partitions in parallel using one LTO 3 per DB2 partition:

- ▶ 5.4 CPUs were used for a Power 5+ 2.3 GHz (or 0.6 CPU per drive)
- ▶ 12.5 GB of memory was used
- ▶ The buffer pool sizes were minimized to db2set DB2_OVERRIDE_BPF=1000

Figure 6-15 shows the processors consumed on the storage agents during the test.

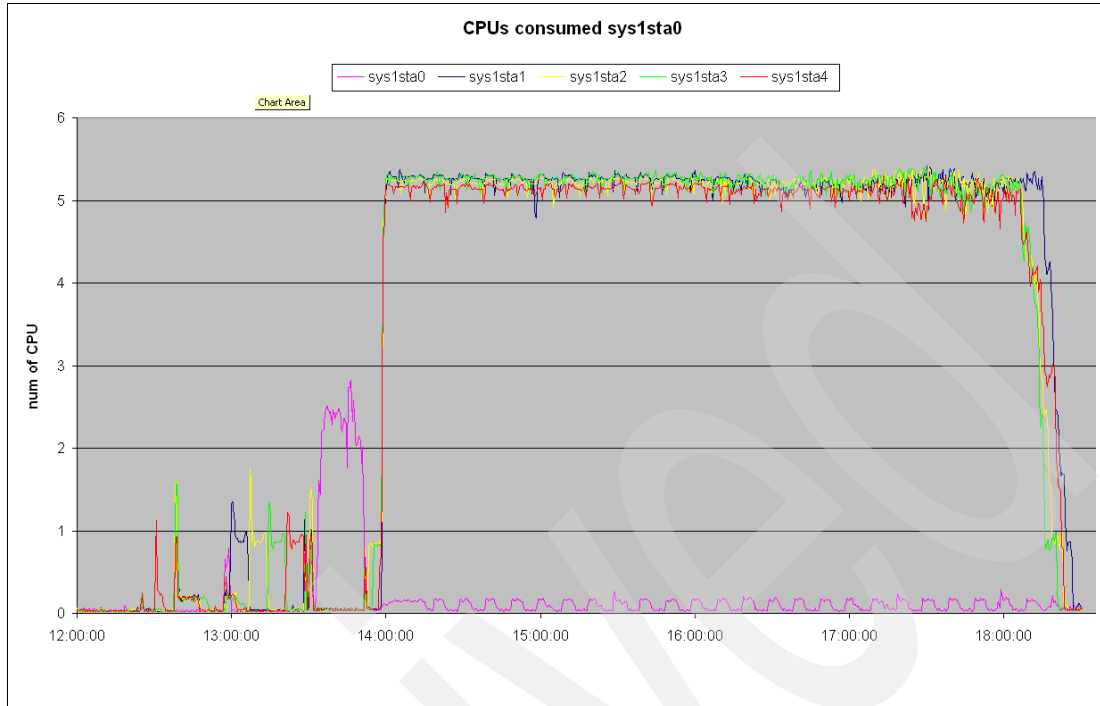


Figure 6-15 CPU usage on the storage agents

The tape usage during the backup is:

- ▶ DB partition 0 first on four drives
- ▶ Thirty-two other DB partitions after

On the Tivoli Storage Manager server, in order to reduce the number of tape mounts and dismounts, and to avoid mount waits on the library, we use the collocation at the storage pool level. Figure 6-16 highlights these behaviors.

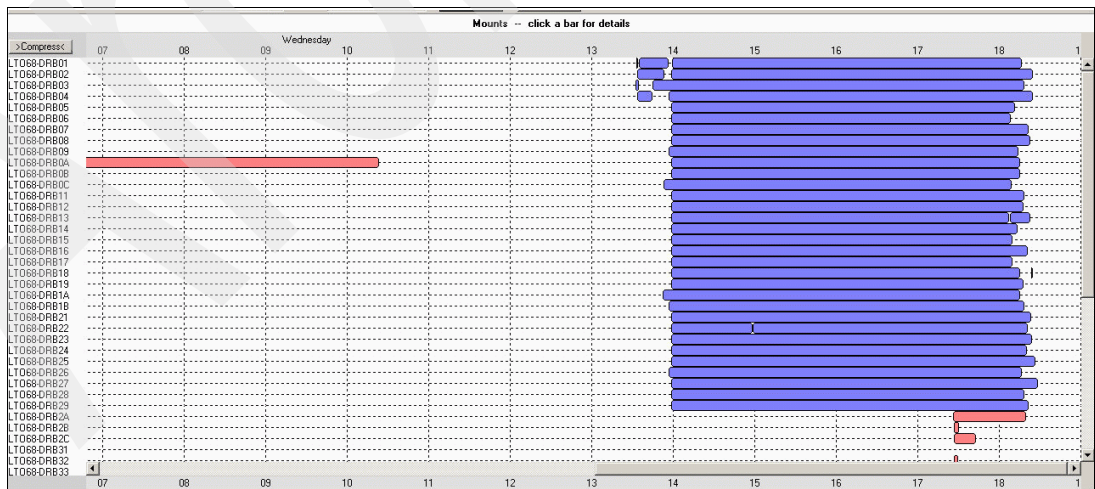


Figure 6-16 Tape mounts during the backup

Observations

We made the following observations:

- ▶ The setup was done with one Tivoli Storage Manager nodename per database partition.
- ▶ DB2_EEE_PARALLEL_BACKUP is set to YES. Values for DB2_NUM_SESSIONS have to be specified for all the 37 partitions in the Tivoli Data Protection for FlashCopy profile corresponding to the backup of DB partition 0.
- ▶ Each storage agent consumes about 5.2 CPUs during eight parallel backups to eight LTO3 tape drives in total.
- ▶ The backup rate is about 450 GB/hr for each DB2 partition, and about 3.6 TB/hr for each storage agent.
- ▶ The step to mount the FlashCopy target devices on the Tivoli Storage Manager server takes a noticeable amount of time in the overall process.
- ▶ The step to mount the FlashCopy target devices on each data mover server is done sequentially and takes about 50 minutes.
- ▶ The tape drives were able to compress the data in this environment at approximately 6:1. This was determined in two ways:
 - First, during the testing, the LTO-3 cartridges would fill up, and Tivoli Storage Manager reported that 2.4 TB of data was contained on each cartridge. With 400 GB of native capacity, the data would need to compress 6:1 to fit 2.4 TB.
 - The other method involved collecting log page 32 from the tape drives after the backup. This was done with the AIX command `tapeuti1 -f /dev/rmtX logpage 32`, where X is the device special number of the tape drive. Using the output of that command and the SCSI reference document for the LTO-3 tape drive, the tape drive reported a compression ratio of on average 6:1.

With 2:1 data and a 2 GB or 4 GB Fibre Channel interface, the IBM LTO-3 tape drive is capable of reading and writing 256 KB blocks¹ at 160 MBps. With data that compresses 6:1, a tape drive with a 2 GB interface can run at the maximum line rate for 2 GB Fibre Channel. This is normally around 175 MBps², but is dependent on the type of host bus adapter and switch technology used. Further tuning may have improved the average MBps for each drive and further shortened the backup window. However, the data rates already achieved the goal of the test.

6.3.2 KPI-1: FlashCopy restore and rollforward of 500 GB log files

This test measures the time required for a FlashCopy restore operation (flashback) and the subsequent roll forward of the DB2 database. The amount of 500 GB of archived logs reflects the volume produced during one day in the customer production systems.

To meet this test KPI, the following objectives must be achieved:

- ▶ Revert a previously taken FlashCopy from the former target volumes to the former source volumes.
- ▶ Retrieve and roll forward through 500 GB of log files.
- ▶ Successfully start of the SAP system.
- ▶ The completion time stamp is the first SAP transaction.

¹ Tivoli Storage Manager Server with LTO-3 devices reads and writes to tape with a 256 KB block size.

² With a 4 Gb interface and 6:1 compressing data, IBM LTO-3 tape drive is capable of 245 MBps.

Approach

This test requires a large amount of log files available for the database. The test runs for a long period of time, and so reset and multiple runs are time consuming.

The following software was used:

- ▶ DB2 UDB 9.1. FP0
- ▶ Data Protection for mySAP (DB2 UDB) 5.4.0.0
- ▶ Data Protection for FlashCopy Devices for mySAP - DB2 5.4.0.0
- ▶ CIM agent for the IBM TotalStorage DS Open API 5.1.0.62

The steps were:

- ▶ For the preparation
 - a. Take a FlashCopy backup of the database using Tivoli Data Protection for Hardware.
 - b. Generate 500 GB of log files by changing table entries in the database. The log files can be generated either by running the data load scenarios or running SQL scripts, generating a huge amount of create table/insert/delete/drop operations.
- ▶ For the execution:
 - a. Restore the database by reverting to a previous FlashCopy backup.
 - b. Retrieve/roll forward the database through 500 GB of logs and optimize this process by adjusting database and parallelism of roll forward processes.
 - c. Stop the roll forward and activation of each partition manually.
 - d. Start the SAP BW system and perform transactions.

The following tools were used:

- ▶ WLM passive monitoring at component level
- ▶ NMON monitoring for System p LPARs
- ▶ Storage monitor for low-level storage server
- ▶ DB2 CLP command to monitor the restore and rollforward process.
- ▶ Tivoli Storage Manager/Tivoli Data Protection log files

The prerequisites for this test are:

- ▶ FlashCopy setup and TDP customization completed
- ▶ FlashCopy setup defined for "COPY" option
- ▶ Exclusive availability of the environment (SAP BW, DB2 UDB, TDP) to perform the test
- ▶ FlashCopy backup successfully taken using TDP
- ▶ Background copy of the FlashCopy backup finished
- ▶ 500 GB of logs successfully generated and archived to Tivoli Storage Manager after the time of the invocation of the FlashCopy backup

KPI-1 results and analysis

The test is completed successfully in less than three hours. Table 6-5 describes the duration of each task.

Table 6-5 KPI1 - process duration

| Time line | Task |
|-----------|---|
| 00 | Invoke TDP processes on all DB servers: <ul style="list-style-type: none"> ▶ One process started on each server hosting DB partitions. ▶ TDP checks available (FC and tape) backups and configuration on the DB servers, and checks source-target volume relationship and file systems, and so on. |
| 15min | User entries for selecting the backup done. |
| 31min | Final confirmation for invoking the FlashCopy restore: <ul style="list-style-type: none"> ▶ Unmount/remove all file systems, export volume groups. ▶ Start FlashCopy restore for all volumes using CIM Agent. ▶ Acquire disk devices and volume groups for all FC volumes, mount file systems. ▶ Mount of all FC file systems on backup server. |
| 1h10min | Reactivate database: <ul style="list-style-type: none"> ▶ Run „db2inidb as mirror“. ▶ Manual „Retrieve of Logs“ is skipped due to direct archiving. |
| 1h20min | TDP processing finished. |
| 1h20min | Rollforward recovery of about 680 GB LOGs started. Direct archiving retrieves LOGs directly from Tivoli Storage Manager during the ROLLFORWARD. |
| 2h40min | Roll forward finished. |

Figure 6-17 highlights the duration of each task and the amount of logs we rolled forward during the flashback test.

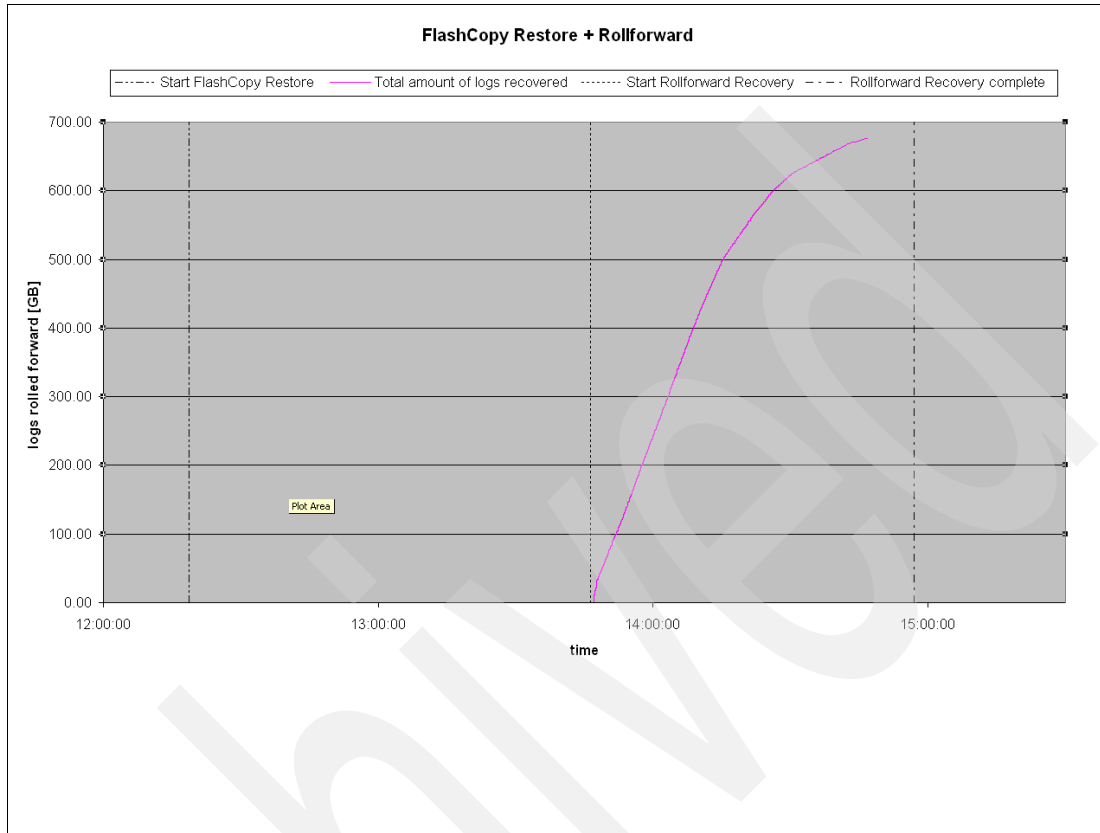


Figure 6-17 KPI1 - process duration

All the generated log files have the same size, but the rollforward of one of them varies from 20 seconds to 1 minute. The factors that may influence this point are:

- ▶ The retrieve of the log file from disk or tape storage pool
- ▶ The mount of the tape
- ▶ The tape positioning

Figure 6-18 illustrates these variations.

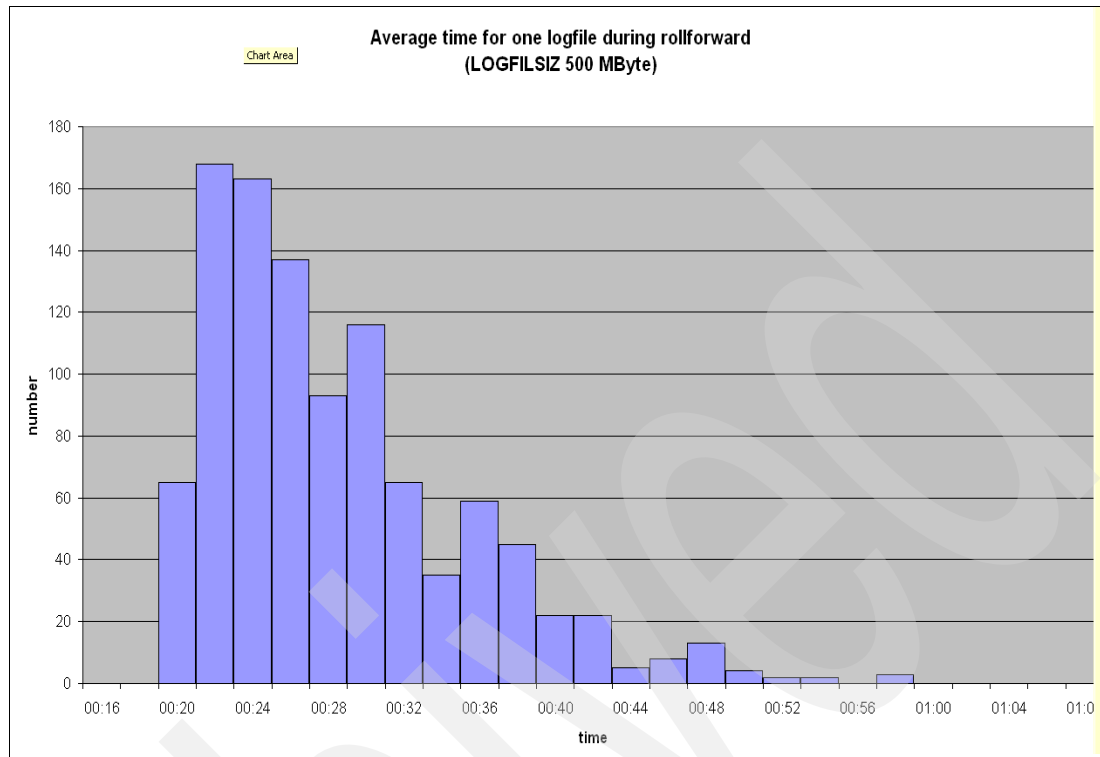


Figure 6-18 KPI1 - roll forward average time for one log file

Observations

The log file generation is not perfectly balanced across all database partitions:

- ▶ Partitions 00, 26, 32 had to retrieve/roll forward a lot of more log files.
- ▶ These partitions also retrieved them directly from tape (already migrated out of the disk storage pool).

We use different Tivoli Storage Manager nodenames per partition. Collcation for the tape storage pool of the log files is activated. The log files for different database partitions are located on different tapes to reduce mount waits.

6.3.3 KPI-2: DB restore from tape with rollforward of 2 TB archived logs

This test measures the time needed for a DB restore from tape and a subsequent retrieve/roll forward of 2 TB of archived logs. The amount of 2 TB of archived logs reflects the volume produced during several days in the customer production systems. The test simulates a disaster recovery scenario.

To meet this test KPI, the following must be achieved:

- ▶ Restore the database from tape and roll forward the database through 2 TB logs.
- ▶ Finish the rollforward.
- ▶ Successfully start the SAP system.
- ▶ The total KPI time will be the sum of the restore time and the rollforward time. This total time needs to be less than 24 hours.

Approach

The test requires a large amount of archived log files available for the database, and a well-defined, reliable, and performance-optimized setup to retrieve and apply the log files. Massive parallel restore requires access to multiple tape devices in parallel. The test runs for a long period of time and so multiple repetitions are time consuming.

The following software was used:

- ▶ DB2 UDB 9.1 FP0
- ▶ Tivoli Data Protection for mySAP (DB2 UDB) 5.4.0.0
- ▶ Tivoli Data Protection for FlashCopy Devices for mySAP - DB2 5.4.0.0
- ▶ CIM agent for the IBM TotalStorage DS Open API 5.1.0.62

The following steps are required:

- ▶ For the preparation:
 - a. Run a backup of the database to tape using Tivoli Data Protection for Hardware.
 - b. Generate 2 TB of log files by changing table entries in the database. The log files can be generated either by running the data load scenarios or running SQL scripts, generating a huge amount of create table/insert/delete/drop operations.
 - c. Drop the database to simulate a disaster recovery on a clean system.
- ▶ For the execution:
 - a. Restore the database to the clean system, having fast container allocation enabled (DB2_USE_FAST_PREALLOCATION).
 - b. The restore will be performed directly on the database server using a *LAN free* restore.
 - c. Retrieve/roll forward the database through 2 TB logs and optimize the process by adjusting the database and parallelism of rollforward processes.
 - d. Stop the rollforward and activation of each partition manually.
 - e. Start the SAP system and perform transactions (for example, se06/sm50).

The following tools were used:

- ▶ WLM passive monitoring at component level
- ▶ NMON monitoring for p5 LPARs
- ▶ Storage monitor for low-level storage server
- ▶ DB2 CLP command to monitor the restore and rollforward process.
- ▶ Tivoli Storage Manager/TDP log files

The prerequisites for this tests are:

- ▶ Tivoli Data Protection setup and customization completed
- ▶ Exclusive availability of the environment (SAP BW, DB2 UDB, Tivoli Data Protection) to perform the test
- ▶ Backup to tape successfully taken using Tivoli Data Protection and Tivoli Storage Manager
- ▶ 2 TB of logs successfully generated and archived to Tivoli Storage Manager after the time backup

KPI-2 results and analysis

The test is completed successfully in 9 hours and 45 minutes. Table 6-6 illustrates the duration of each step.

Table 6-6 KPI2 - process duration

| Time | Task |
|---|------------------------------------|
| 22:30 | Tape restore started |
| 05:10 | Tape restore finished |
| Runtime for the tape restore: 6 hours, 40 minutes | |
| 05:15 | Rollforward started |
| 07:53 | Rollforward recovery phase stopped |
| 08:01 | Rollforward stopped |
| 08:07 | SAP started |
| Runtime for the roll forward: 2 hours, 52 minutes | |

For the restore of the database from tape, the bulk of the restore was processed at an impressive rate of 4,127 MBps for over four hours. As typical in restores, the last bit of processing takes additional time, and the throughput decreases as some jobs finish earlier than others.

Figure 6-19 illustrates the time line for the restore portion of KPI-2, and omits the rollforward time line. From 22:39 to 23:38, the system processed the restore of partition 0 using four tape drives. From 23:38 to 3:42 (four hours, four minutes) the bulk of the restore was processed at a peak average of 4,127 MBps using 32 tape drives. From 3:42 to the end of the backup, longer running restores were processed.

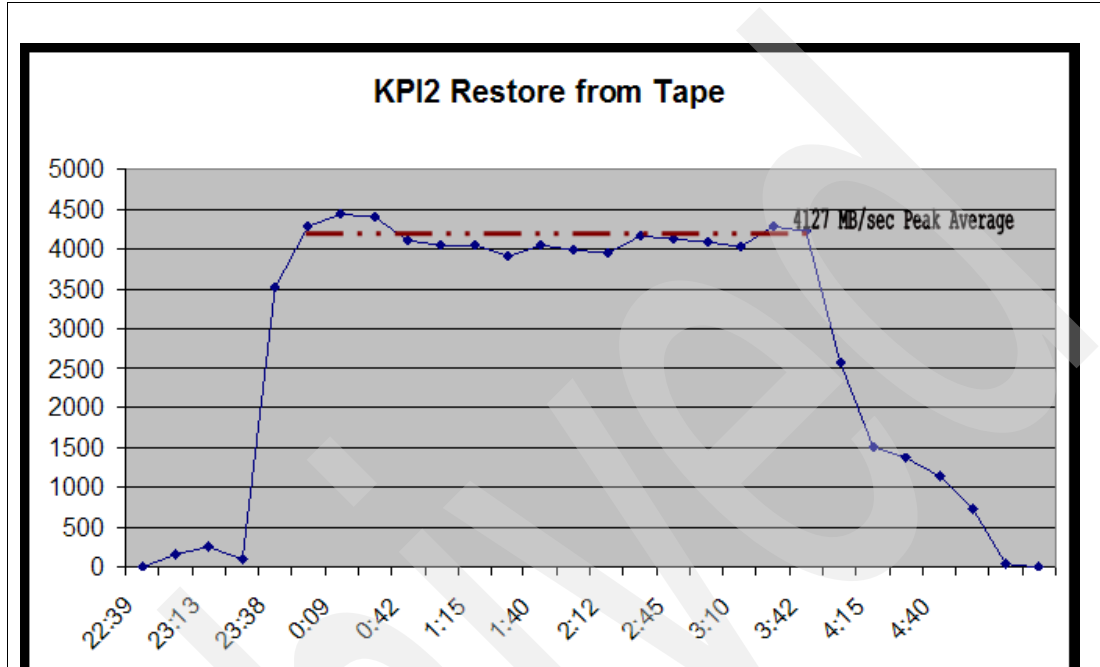


Figure 6-19 KPI-2 tape restore average peak throughput

Figure 6-20 provides a look at two ports attached to LTO-3 tape drives used during the restore. The top chart in the figure shows a drive that was also involved in the restore of partition 0 from 22:39 to 23:38. The bottom chart is for a drive that only was used in restoring one of the data partitions. No activity occurred for this drive during the time period of 22:39 to 23:38.

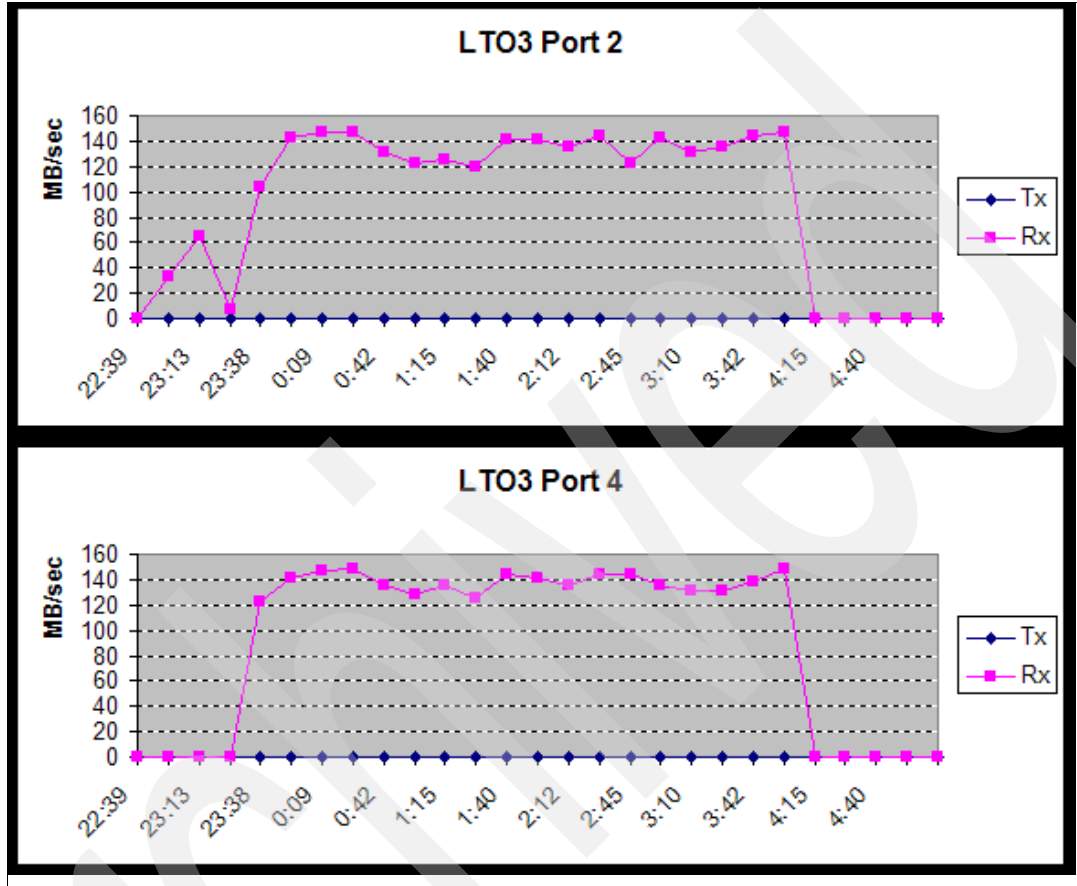


Figure 6-20 LTO-3 ports restoring the database

The restore rate varies from 350 GB/hr to 470 GB/hr for a DB2 partition. Figure 6-21 illustrates the restore data throughput per DB2 partition.

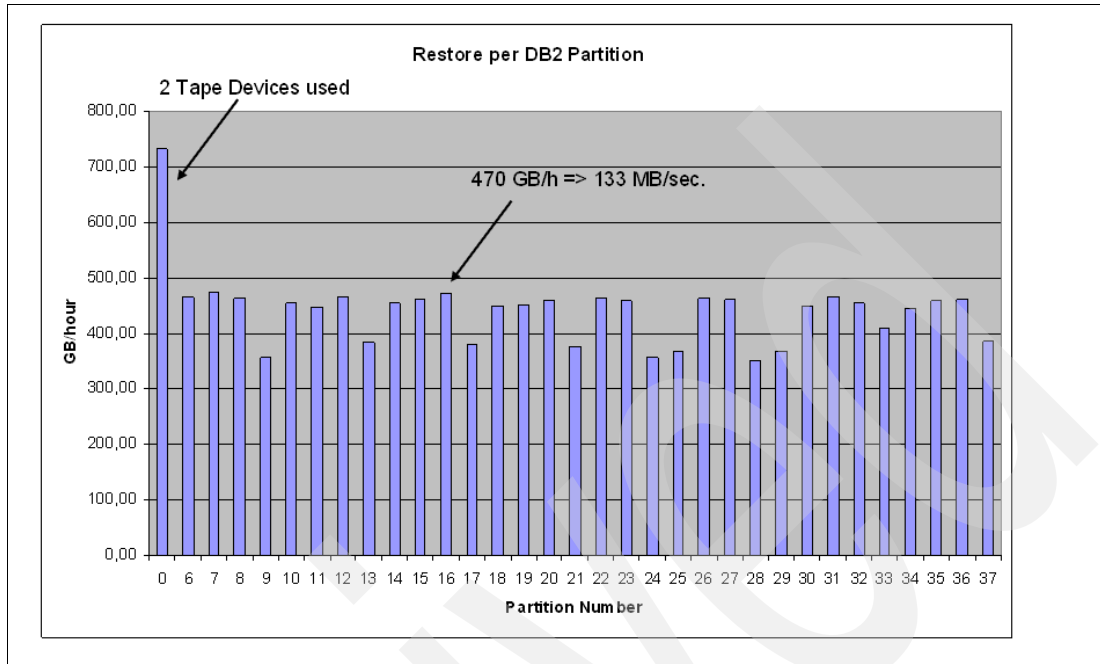


Figure 6-21 Tape restore per DB2 partition

We recover the logs on all nodes in parallel.

node0000 has a longer run time due to the nature of the workload that generates more logs on Node0000. All the logs are retrieved from a disk storage pool, and the size per log file is 512 MB.

Figure 6-22 illustrates this DB2 log recovery per DB2 node.

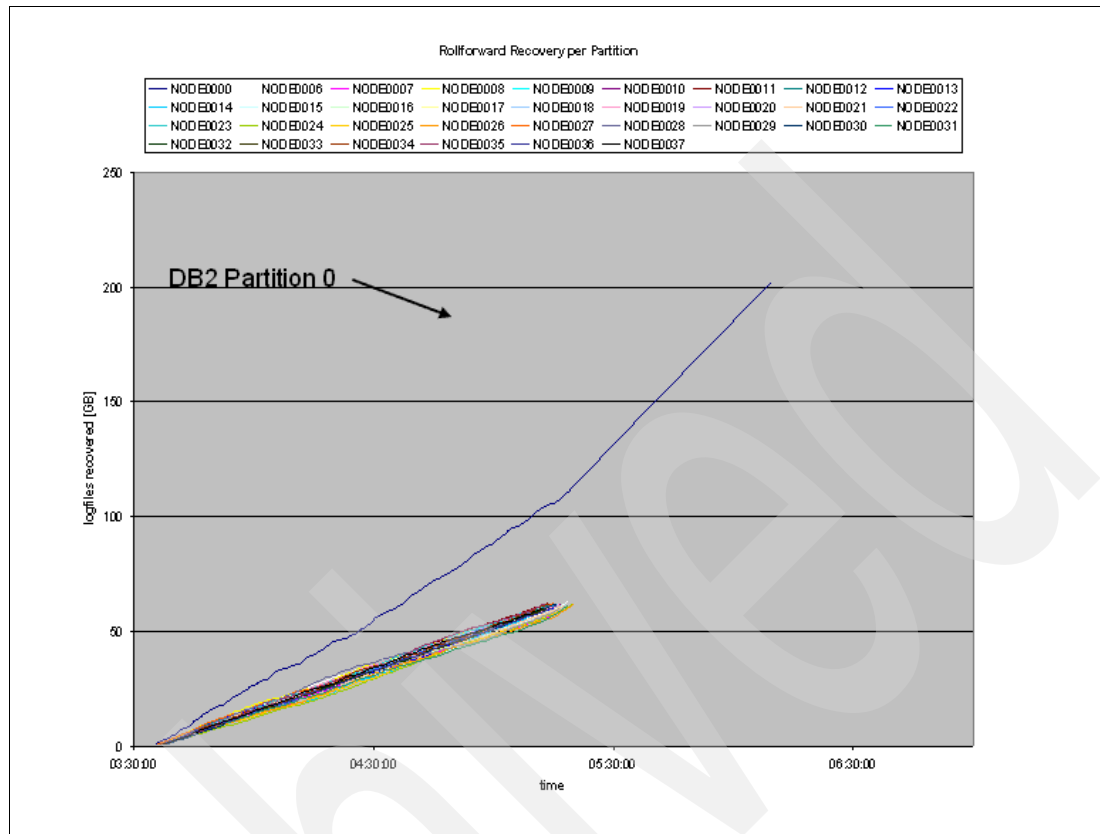


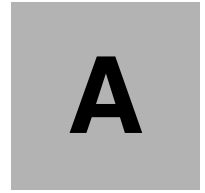
Figure 6-22 Roll forward recovery per DB2 partition

Observations

We made the following observations:

- ▶ The log file generation is not perfectly balanced across all database partitions.
- ▶ Some partitions also retrieved them directly from tape (already migrated out of the disk storage pool).
- ▶ We use different Tivoli Storage Manager nodenames per partition, collocation for the tape storage pool of the log files activated, log files for different database partitions are located on different tape drives to reduce the mount waits.

Archived



The scripts used

For the restore

Example A-1 provides the main script to restore the DB2 database.

Example: A-1 restore.ksh

```
#!/bin/ksh
nohup /db2/db2eb8/scripts/re1/restore_00.ksh >
/db2/db2eb8/scripts/re1/restore_00.log
nohup /db2/db2eb8/scripts/re1/restore_01.ksh >
/db2/db2eb8/scripts/re1/restore_01.log
nohup /db2/db2eb8/scripts/re1/restore_02.ksh >
/db2/db2eb8/scripts/re1/restore_02.log
nohup /db2/db2eb8/scripts/re1/restore_03.ksh >
/db2/db2eb8/scripts/re1/restore_03.log
nohup /db2/db2eb8/scripts/re1/restore_04.ksh >
/db2/db2eb8/scripts/re1/restore_04.log
nohup /db2/db2eb8/scripts/re1/restore_05.ksh >
/db2/db2eb8/scripts/re1/restore_05.log
nohup /db2/db2eb8/scripts/re1/restore_06.ksh >
/db2/db2eb8/scripts/re1/restore_06.log &
nohup /db2/db2eb8/scripts/re1/restore_07.ksh >
/db2/db2eb8/scripts/re1/restore_07.log &
nohup /db2/db2eb8/scripts/re1/restore_08.ksh >
/db2/db2eb8/scripts/re1/restore_08.log &
nohup /db2/db2eb8/scripts/re1/restore_09.ksh >
/db2/db2eb8/scripts/re1/restore_09.log &
nohup /db2/db2eb8/scripts/re1/restore_10.ksh >
/db2/db2eb8/scripts/re1/restore_10.log &
nohup /db2/db2eb8/scripts/re1/restore_11.ksh >
/db2/db2eb8/scripts/re1/restore_11.log &
nohup /db2/db2eb8/scripts/re1/restore_12.ksh >
/db2/db2eb8/scripts/re1/restore_12.log &
nohup /db2/db2eb8/scripts/re1/restore_13.ksh >
/db2/db2eb8/scripts/re1/restore_13.log &
nohup /db2/db2eb8/scripts/re1/restore_14.ksh >
/db2/db2eb8/scripts/re1/restore_14.log &
nohup /db2/db2eb8/scripts/re1/restore_15.ksh >
/db2/db2eb8/scripts/re1/restore_15.log &
nohup /db2/db2eb8/scripts/re1/restore_16.ksh >
/db2/db2eb8/scripts/re1/restore_16.log &
nohup /db2/db2eb8/scripts/re1/restore_17.ksh >
/db2/db2eb8/scripts/re1/restore_17.log &
nohup /db2/db2eb8/scripts/re1/restore_18.ksh >
/db2/db2eb8/scripts/re1/restore_18.log &
nohup /db2/db2eb8/scripts/re1/restore_19.ksh >
/db2/db2eb8/scripts/re1/restore_19.log &
nohup /db2/db2eb8/scripts/re1/restore_20.ksh >
/db2/db2eb8/scripts/re1/restore_20.log &
nohup /db2/db2eb8/scripts/re1/restore_21.ksh >
/db2/db2eb8/scripts/re1/restore_21.log &
nohup /db2/db2eb8/scripts/re1/restore_22.ksh >
/db2/db2eb8/scripts/re1/restore_22.log &
nohup /db2/db2eb8/scripts/re1/restore_23.ksh >
/db2/db2eb8/scripts/re1/restore_23.log &
```

```

nohup /db2/db2eb8/scripts/re1/restore_24.ksh >
/db2/db2eb8/scripts/re1/restore_24.log &
nohup /db2/db2eb8/scripts/re1/restore_25.ksh >
/db2/db2eb8/scripts/re1/restore_25.log &
nohup /db2/db2eb8/scripts/re1/restore_26.ksh >
/db2/db2eb8/scripts/re1/restore_26.log &
nohup /db2/db2eb8/scripts/re1/restore_27.ksh >
/db2/db2eb8/scripts/re1/restore_27.log &
nohup /db2/db2eb8/scripts/re1/restore_28.ksh >
/db2/db2eb8/scripts/re1/restore_28.log &
nohup /db2/db2eb8/scripts/re1/restore_29.ksh >
/db2/db2eb8/scripts/re1/restore_29.log &
nohup /db2/db2eb8/scripts/re1/restore_30.ksh >
/db2/db2eb8/scripts/re1/restore_30.log &
nohup /db2/db2eb8/scripts/re1/restore_31.ksh >
/db2/db2eb8/scripts/re1/restore_31.log &
nohup /db2/db2eb8/scripts/re1/restore_32.ksh >
/db2/db2eb8/scripts/re1/restore_32.log &
nohup /db2/db2eb8/scripts/re1/restore_33.ksh >
/db2/db2eb8/scripts/re1/restore_33.log &
nohup /db2/db2eb8/scripts/re1/restore_34.ksh >
/db2/db2eb8/scripts/re1/restore_34.log &
nohup /db2/db2eb8/scripts/re1/restore_35.ksh >
/db2/db2eb8/scripts/re1/restore_35.log &
nohup /db2/db2eb8/scripts/re1/restore_36.ksh >
/db2/db2eb8/scripts/re1/restore_36.log &
nohup /db2/db2eb8/scripts/re1/restore_37.ksh >
/db2/db2eb8/scripts/re1/restore_37.log &

```

For the node 7 backup

Example A-2 provides the script used to back up the DB node 7.

Example: A-2 backu_sys1db1d_1.ksh

```

#!/bin/ksh
date
export RAHBUFNAME=node7.buf
echo db2_all "<<+7< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions with 14 buffers buffer
1024 parallelism 8 without prompting"
retry=0
while [[ ${retry} -lt 2 ]]
do
date
db2_all "<<+7< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions with 14 buffers buffer
1024 parallelism 8 without prompting" > /db2/db2eb8/scripts/bu32/log/node7.log
grep "completed ok" /db2/db2eb8/scripts/bu32/log/node7.log
if [[ $? -eq 0 ]]
then
echo "db2 backup db EB8 ONLINE completed o.k. for node 7"
retry=2

```

```

else
  grep "completed rc" /db2/db2eb8/scripts/bu32/log/node7.log
  if [[ $? -eq 0 ]]
  then
    RC=$(grep "completed rc" /db2/db2eb8/scripts/bu32/log/node7.log | cut -d
'=' -f2)
    echo "db2 backup db EB8 ONLINE completed with RC $RC for node 7"
    if [[ ${RC} -lt 4 ]]
    then
      retry=2
    else
      (( retry=retry+1 ))
    fi
  else
    echo "seems that db2_all start failed for node 7; retrying ..."
    sleep 5
  fi
fi
done
date
sleep 30
rm
/db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_1_RUNNING

```

The Tivoli Data Protection for mySAP configuration file

Example A-3 provides the Tivoli Data Protection for mySAP configuration file used in our tests.

Example: A-3 initEB8.utl

```

#-----
# Data Protection for mySAP.com(R) technology interface for DB2 UDB
#
# Sample profile for Data Protection for mySAP.com(R) technology
# Version 3.3 for UNIX
#-----
# See the 'Data Protection for mySAP.com(R) technology Installation &
# User's Guide' for a full description.

# For comment symbol the character '#' can be used.
# Everything following this character will be interpreted as comment.
#
# Data Protection for mySAP.com(R) technology V3R3 accesses its profile
# in "read only" mode only. All variable parameters like passwords, date of
# last password change, current version number will be written into the file
# specified with the CONFIG_FILE parameter. The passwords will be encrypted.
#-----
# Prefix of the 'Backup ID' which will be stored in the description field
# of the Tivoli Storage Manager archive function.
# Maximum 6 characters.
# Default: SID___
#-----
BACKUPIDPREFIX      SY1EB8

```

```

#-----
# Number of parallel sessions to be established.
# Note: This number should correspond with the number of simultaneously
# available tape drives specified for the Tivoli Storage Manager server.
# Default: none
#-----
MAX_SESSIONS          32
#-----
# Number of backup copies of the DB2 log files.
# The valid range of REDOLOG_COPIES is from 1 to 9.
# Default: 1.
#-----
#REDOLOG_COPIES2
#-----
# Specifies the block size for disk I/O (in bytes). The valid range is
# from 4 KB to 32 MB.
# The default values have been chosen from our performance experiments in
# standard hardware environments.
# Default: 131072 (128 KB) on UNIX, 32768 (32 KB) on Windows NT.
#-----
BUFFSIZE131072        # block size in bytes
#-----
# Maximum number of data base backup versions to be kept.
# Note: Version control by Data Protection for mySAP.com(R) technology is
# only activated if the parameter MAX_VERSION is not 0.
# The valid range of MAX_VERSIONS is from 0 to 9999.
# Default: 0
#-----
MAX_VERSIONS0
#-----
# Specifies whether a null block compression of the data is to be performed
# before transmission to Tivoli Storage Manager.
# Although RL compression introduces additional CPU load, throughput can be
# improved when the network is the bottleneck. RL compression in Data
# Protection for mySAP.com(R) technology should not be used together with
# Tivoli Storage Manager API compression.
# Default: NO
#-----
RL_COMPRESSIONNO     # NO is default
#-----
# Specifies the number of DB2 buffers to be used when invoking a
# DB2 RESTORE DATABASE command via backom.
# The value of DB2_BUFFNUM is translated into the "WITH nnn BUFFERS"
# command option of the DB2 RESTORE DATABASE command.
# Default: 2
#-----
#DB2_BUFFNUM12
#-----
# Specifies the number of DB2 buffer manipulators to be started when
# invoking a DB2 RESTORE DATABASE command via backom.
# The value of DB2_BUFFNUM is translated into the PARALLELISM command
# option of the DB2 RESTORE DATABASE command.
# Default: 1
#-----
MULTIPLEXING8

```

```

#-----
# Controls generation of a trace file.
# Note: We recommend using the trace function only in cooperation with
# the hotline.
# Default: OFF
#-----
#TRACE ALL
#TRACEFILE /tmp/backint-%BID.trace
#-----
# Specify the full path of the configuration file.
# Default: none.
#-----
CONFIG_FILE /db2/EB8/dbs/tsm_config/%DB2NODE/initeB8.bki
#-----
# Denotes if Data Protection for mySAP.com(R) technology shall send
# error/status information to a Tivoli Storage Manager server.
# The servername must match one of the servers listed in a SERVER statement.
# Values for verbosity can be ERROR, WARNING, or DETAIL.
# Default: none.
#-----
#LOG_SERVER servername [verbosity]
#LOG_SERVER server_a ERROR
#-----
# Denotes if Data Protection for mySAP.com(R) technology shall send
# error/status information to a network management program via SNMP traps.
# Default: none.
#-----
#SNMPTRAP Hostname community level
#SNMPTRAPserver_a publicdetail
#*****
# Statement for multiple Servers and multiple Paths.
# may be used multiple times (one for each server).
#*****
SERVER detsm05
SESSIONS 32
PASSWORDREQUIRED NO # Use a password
# ADMSNODE NODE # Tivoli Storage Manager Nodename
BRBACKUPMGTCCLASS LTO3TAPE
BRARCHIVEMGTCCLASS LOGPOOL
# USE_AT 0 1 2 3 4 5 6 # Days for backup
#*****
# USE_AT : 0=Su 1=Mo 2=Tu 3=We 4=Th 5=Fr 6=Sa
# Default: all days
#*****
#-----
# End of profile

```

initEB8.fct

Example A-4 provides the list of source and target volumes used for the FlashCopy.

Example: A-4 initEB8.fct

```
>>> volumes_set_1
TARGET_VOLUME 75DPFG1B000 75DPFG11000 375809638400_Bytes
TARGET_VOLUME 75DPFG1B100 75DPFG11100 375809638400_Bytes
TARGET_VOLUME 75DPFG1B001 75DPFG11001 375809638400_Bytes
TARGET_VOLUME 75DPFG1B101 75DPFG11101 375809638400_Bytes
TARGET_VOLUME 75DPFG1B004 75DPFG11004 26843545600_Bytes
TARGET_VOLUME 75DPFG1B104 75DPFG11104 26843545600_Bytes
TARGET_VOLUME 75DPFG1B005 75DPFG11005 26843545600_Bytes
TARGET_VOLUME 75DPFG1B105 75DPFG11105 26843545600_Bytes
TARGET_VOLUME 75DPFG1B002 75DPFG11002 375809638400_Bytes
TARGET_VOLUME 75DPFG1B102 75DPFG11102 375809638400_Bytes
TARGET_VOLUME 75DPFG1B003 75DPFG11003 375809638400_Bytes
TARGET_VOLUME 75DPFG1B103 75DPFG11103 375809638400_Bytes
TARGET_VOLUME 75DPFG1B006 75DPFG11006 26843545600_Bytes
TARGET_VOLUME 75DPFG1B106 75DPFG11106 26843545600_Bytes
TARGET_VOLUME 75DPFG1B007 75DPFG11007 26843545600_Bytes
TARGET_VOLUME 75DPFG1B107 75DPFG11107 26843545600_Bytes
TARGET_VOLUME 75DPFG1B200 75DPFG11200 375809638400_Bytes
TARGET_VOLUME 75DPFG1B300 75DPFG11300 375809638400_Bytes
TARGET_VOLUME 75DPFG1B201 75DPFG11201 375809638400_Bytes
TARGET_VOLUME 75DPFG1B301 75DPFG11301 375809638400_Bytes
TARGET_VOLUME 75DPFG1B204 75DPFG11204 26843545600_Bytes
TARGET_VOLUME 75DPFG1B304 75DPFG11304 26843545600_Bytes
TARGET_VOLUME 75DPFG1B205 75DPFG11205 26843545600_Bytes
TARGET_VOLUME 75DPFG1B305 75DPFG11305 26843545600_Bytes
TARGET_VOLUME 75DPFG1B202 75DPFG11202 375809638400_Bytes
TARGET_VOLUME 75DPFG1B302 75DPFG11302 375809638400_Bytes
TARGET_VOLUME 75DPFG1B203 75DPFG11203 375809638400_Bytes
.....
.....
<<< volumes_set_1
```

The Tivoli Data Protection ACS configuration file

Example A-5 provides the TDP/ACS configuration file.

Example: A-5 initEB8.fcs

```
#####
#                               5.4.0.0
#####
LOGON_HOST_BACK                 sys1sta1
BACKUP_MAX                      05
IDS_CONTROL_FILE                /db2/EB8/dbs/sys1db1d/save/idssave
CONFIG_FILE                     /db2/EB8/dbs/initEB8.fcp
WORK_DIR                        /db2/EB8/dbs/sys1db1d/work
TRACE                           NO
LOG_TRACE_DIR                   /db2/EB8/dbs/sys1db1d/logtraces
```



```
nohup /db2/db2eb8/scripts/bu32/backup_sys1db1d_2.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db1d_2.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_3_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db1d_3.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db1d_3.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_4_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db1d_4.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db1d_4.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_5_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db1d_5.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db1d_5.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_6_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db1d_6.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db1d_6.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_7_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db1d_7.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db1d_7.log &
sleep 5
echo Starting backup for partitions on host sys1db2d
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_0_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db2d_0.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db2d_0.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_1_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db2d_1.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db2d_1.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_2_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db2d_2.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db2d_2.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_3_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db2d_3.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db2d_3.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_4_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db2d_4.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db2d_4.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_5_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db2d_5.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db2d_5.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_6_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db2d_6.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db2d_6.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_7_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db2d_7.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db2d_7.log &
```

```
sleep 5
echo Starting backup for partitions on host sys1db3d
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_0_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db3d_0.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db3d_0.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_1_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db3d_1.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db3d_1.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_2_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db3d_2.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db3d_2.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_3_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db3d_3.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db3d_3.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_4_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db3d_4.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db3d_4.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_5_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db3d_5.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db3d_5.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_6_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db3d_6.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db3d_6.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_7_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db3d_7.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db3d_7.log &
sleep 5
echo Starting backup for partitions on host sys1db4d
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_0_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db4d_0.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db4d_0.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_1_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db4d_1.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db4d_1.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_2_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db4d_2.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db4d_2.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_3_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db4d_3.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db4d_3.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_4_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db4d_4.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db4d_4.log &
sleep 5
```

```

touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_5_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db4d_5.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db4d_5.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_6_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db4d_6.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db4d_6.log &
sleep 5
touch /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_7_RUNNING
nohup /db2/db2eb8/scripts/bu32/backup_sys1db4d_7.ksh >
/db2/db2eb8/scripts/bu32/backup_sys1db4d_7.log &
sleep 5
#
# wait for completion of all backups
#
i=0
while [[ $i -eq 0 ]]; do
    i=1
    sleep 900
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_0_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_1_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_2_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_3_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_4_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_5_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_6_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db1d_7_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_0_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_1_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_2_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_3_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_4_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_5_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_6_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db2d_7_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_0_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_1_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_2_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_3_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_4_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_5_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_6_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db3d_7_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_0_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_1_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_2_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_3_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_4_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_5_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_6_RUNNING ]] && i=0
    [[ -e /db2/db2eb8/scripts/bu32/BACKUP_sys1db4d_7_RUNNING ]] && i=0
done
#
# collect performance data
#

```

```

par2=$(date "+ endd=%m/%d/%Y endt=%H:%M")
mv /db2/EB8/dbs/tsm_config/tdplog/*.log /db2/EB8/dbs/tsm_config/tdplog/${old}
cd /db2/EB8/dbs/tsm_config/tdplog/${old}
/db2/EB8/dbs/tsm_config/tdplog/analyze_logs > sessions.csv
/db2/EB8/dbs/tsm_config/tdplog/collect_mount ${par1} ${par2} > mount.csv

```

For the node 7 restore

Example A-7 provides the script to restore the DB2 node 7.

Example: A-7 restore_07.ksh

```

#!/bin/ksh
date
echo db2_all "<<+007< db2 restore db EB8 load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions options
/db2/EB8/dbs/tsm_config/vendor.env.07 taken at 20070228135138 with 014 buffers
buffer 01024 parallelism 008 without prompting"
export RAHBUFNAME=07.buf

db2_all "<<+007< db2 restore db EB8 load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions options
/db2/EB8/dbs/tsm_config/vendor.env.07 taken at 20070228135138 with 014 buffers
buffer 01024 parallelism 008 without prompting"
date

```

The script for the backup of node 0

Example A-8 provides the script used to back up the DB partition 0.

Example: A-8 backup_node0.ksh

```

#!/bin/ksh
date
mkdir /db2/db2eb8/scripts/bu32/log 2>/dev/null
started=0
while [[ ${started} -eq 0 ]]
do
    export RAHBUFNAME=node0.buf
    date
    echo db2_all "<<+0< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 4 sessions with 14 buffers buffer
1024 parallelism 8 without prompting"
    db2_all "<<+0< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 4 sessions with 14 buffers buffer
1024 parallelism 8 without prompting" > /db2/db2eb8/scripts/bu32/log/node0.log
    grep "completed ok" /db2/db2eb8/scripts/bu32/log/node0.log
    if [[ $? -eq 0 ]]
    then
        echo "db2 backup db EB8 ONLINE completed o.K. for node 0"
        started=1
    else

```

```

    grep "completed rc" /db2/db2eb8/scripts/bu32/log/node0.log
    if [[ $? -eq 0 ]]
    then
        RC=$(grep "completed rc" /db2/db2eb8/scripts/bu32/log/node0.log | cut -d
'=' -f2)
        echo "db2 backup db EB8 ONLINE completed with RC $RC for node 0"
        started=1
    else
        echo "seems that db2_all start failed for node 0; retrying ..."
        sleep 5
    fi
fi
done
date
date
mkdir /db2/db2eb8/scripts/bu32/log 2>/dev/null
started=0
while [[ ${started} -eq 0 ]]
do
    export RAHBUFNAME=node1.buf
    date
    echo db2_all "<<+1< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions with 14 buffers buffer
1024 parallelism 8 without prompting"
    db2_all "<<+1< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions with 14 buffers buffer
1024 parallelism 8 without prompting" > /db2/db2eb8/scripts/bu32/log/node1.log
    grep "completed ok" /db2/db2eb8/scripts/bu32/log/node1.log
    if [[ $? -eq 0 ]]
    then
        echo "db2 backup db EB8 ONLINE completed o.K. for node 1"
        started=1
    else
        grep "completed rc" /db2/db2eb8/scripts/bu32/log/node1.log
        if [[ $? -eq 0 ]]
        then
            RC=$(grep "completed rc" /db2/db2eb8/scripts/bu32/log/node1.log | cut -d
'=' -f2)
            echo "db2 backup db EB8 ONLINE completed with RC $RC for node 1"
            started=1
        else
            echo "seems that db2_all start failed for node 1; retrying ..."
            sleep 5
        fi
    fi
done
date
date
mkdir /db2/db2eb8/scripts/bu32/log 2>/dev/null
started=0
while [[ ${started} -eq 0 ]]
do
    export RAHBUFNAME=node2.buf
    date

```

```

    echo db2_all "<<<+2< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions with 14 buffers buffer
1024 parallelism 8 without prompting"
    db2_all "<<<+2< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions with 14 buffers buffer
1024 parallelism 8 without prompting" > /db2/db2eb8/scripts/bu32/log/node2.log
    grep "completed ok" /db2/db2eb8/scripts/bu32/log/node2.log
    if [[ $? -eq 0 ]]
    then
        echo "db2 backup db EB8 ONLINE completed o.k. for node 2"
        started=1
    else
        grep "completed rc" /db2/db2eb8/scripts/bu32/log/node2.log
        if [[ $? -eq 0 ]]
        then
            RC=$(grep "completed rc" /db2/db2eb8/scripts/bu32/log/node2.log | cut -d
'=' -f2)
            echo "db2 backup db EB8 ONLINE completed with RC $RC for node 2"
            started=1
        else
            echo "seems that db2_all start failed for node 2; retrying ..."
            sleep 5
        fi
    fi
done
date
date
mkdir /db2/db2eb8/scripts/bu32/log 2>/dev/null
started=0
while [[ ${started} -eq 0 ]]
do
    export RAHBUFNAME=node3.buf
    date
    echo db2_all "<<<+3< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions with 14 buffers buffer
1024 parallelism 8 without prompting"
    db2_all "<<<+3< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions with 14 buffers buffer
1024 parallelism 8 without prompting" > /db2/db2eb8/scripts/bu32/log/node3.log
    grep "completed ok" /db2/db2eb8/scripts/bu32/log/node3.log
    if [[ $? -eq 0 ]]
    then
        echo "db2 backup db EB8 ONLINE completed o.k. for node 3"
        started=1
    else
        grep "completed rc" /db2/db2eb8/scripts/bu32/log/node3.log
        if [[ $? -eq 0 ]]
        then
            RC=$(grep "completed rc" /db2/db2eb8/scripts/bu32/log/node3.log | cut -d
'=' -f2)
            echo "db2 backup db EB8 ONLINE completed with RC $RC for node 3"
            started=1
        else
            echo "seems that db2_all start failed for node 3; retrying ..."
            sleep 5
        fi
    fi
done

```



```

        fi
    fi
done
date
date
mkdir /db2/db2eb8/scripts/bu32/log 2>/dev/null
started=0
while [[ ${started} -eq 0 ]]
do
    export RAHBUFNAME=node4.buf
    date
    echo db2_all "<<+4< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions with 14 buffers buffer
1024 parallelism 8 without prompting"
    db2_all "<<+4< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions with 14 buffers buffer
1024 parallelism 8 without prompting" > /db2/db2eb8/scripts/bu32/log/node4.log
    grep "completed ok" /db2/db2eb8/scripts/bu32/log/node4.log
    if [[ $? -eq 0 ]]
    then
        echo "db2 backup db EB8 ONLINE completed o.K. for node 4"
        started=1
    else
        grep "completed rc" /db2/db2eb8/scripts/bu32/log/node4.log
        if [[ $? -eq 0 ]]
        then
            RC=$(grep "completed rc" /db2/db2eb8/scripts/bu32/log/node4.log | cut -d
'=' -f2)
            echo "db2 backup db EB8 ONLINE completed with RC $RC for node 4"
            started=1
        else
            echo "seems that db2_all start failed for node 4; retrying ..."
            sleep 5
        fi
    fi
done
date
date
mkdir /db2/db2eb8/scripts/bu32/log 2>/dev/null
started=0
while [[ ${started} -eq 0 ]]
do
    export RAHBUFNAME=node5.buf
    date
    echo db2_all "<<+5< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions with 14 buffers buffer
1024 parallelism 8 without prompting"
    db2_all "<<+5< db2 backup db EB8 ONLINE load
/usr/tivoli/tsm/tdp_r3/db264/libtdpdb264.a open 1 sessions with 14 buffers buffer
1024 parallelism 8 without prompting" > /db2/db2eb8/scripts/bu32/log/node5.log
    grep "completed ok" /db2/db2eb8/scripts/bu32/log/node5.log
    if [[ $? -eq 0 ]]
    then
        echo "db2 backup db EB8 ONLINE completed o.K. for node 5"
        started=1
    fi
done

```

```

else
  grep "completed rc" /db2/db2eb8/scripts/bu32/log/node5.log
  if [[ $? -eq 0 ]]
  then
    RC=$(grep "completed rc" /db2/db2eb8/scripts/bu32/log/node5.log | cut -d
'=' -f2)
    echo "db2 backup db EB8 ONLINE completed with RC $RC for node 5"
    started=1
  else
    echo "seems that db2_all start failed for node 5; retrying ..."
    sleep 5
  fi
fi
done
date

```

The Tivoli Storage Manager API configuration file

Example A-9 provides the system configuration file for the TSM API client.

Example: A-9 dsm.sys

```

servername          detsm05
nodename             EB8_SYS1
clusternodeno
passworddir          /db2/EB8/dbs/tsm_config
enablelanfree        YES
resourceutilization 5
lanfreecommmethod   SHAREDMEM
lanfreeshmport       1513
passwordaccess       generate
MANAGEDServices      schedule
schedlogretention    15
errorlogretention    15
schedlogname         /usr/tivoli/tsm/client/ba/bin/dsmsched.log
ERRORLOGName         /db2/EB8/dbs/tsm_config/tsmerror.log
ERRORLOGRetention N
commmethod           tcpip
tcpport              1500
tcpserveraddress     deats005
LARGECOMMBUFFERS     YES
TCPNODELAY           YES
TCPwindowsize        640
TCPbuffsize          32

servername           lanfree
*nodename             EB8_SYS3
*nodename             EB8_SYS2
nodename             EB8_SYS1
tcpport              1503
tcpserveraddress
localhos

```

Abbreviations and acronyms

| | | | |
|--------------------|---|--------------|--|
| ABAP | Advanced Business Application Programming | GHz | Giga Herz |
| AIO | Asynchronous I/O | GUI | Graphic User Interface |
| AIX | Advanced Interactive eXecutive | HA | Host Adapter |
| APAR | Authorized Program Analysis Report | HACMP | High Availability Cluster MultiProcessing |
| APD | Analysis Processor Designer | HDD | Hard Disk Drive |
| API | Application Programming Interface | HSM | Hierarchical Storage Management |
| BAPI® | Business Application Programming Interface | HTML | Hypertext Markup Language |
| BCU | Balanced Configuration Unit | HTTP | Hypertext Transfer Protocol. |
| BI | Business intelligence | HW | Hardware |
| BLOB | Binary Large Object | IBM | International Business Machines Corporation |
| CI | Central Instance | I/O | Input output |
| CIM | Common Information Model | IP | Internet Protocol |
| CIMOM | CIM Object Manager | ITSO | International Technical Support Organization |
| CIO | Concurrent I/O | J2EE™ | Java 2 Enterprise Edition |
| CLI | Command Line Interface | JCA | Java EE Connector Architecture |
| CPU | Central Processor Unit | JDBC™ | Java Database Connectivity. |
| CWMI | Common Warehouse Metadata Interchange | KPI | Key performance indicator |
| DA | Device Adapter | LAN | Local Area Network |
| DB2 HPU | DB2 High Performance Unload | LIC | Licensed Internal Code |
| DB2 PE | DB2 Performance Expert | LOB | Large object |
| DB2 UDB ESE | DB2 Universal Database™ Enterprise Server Edition | LPAR | Logical partition |
| DBA | Data Base Administrator | LRU | Last recent used |
| DDL | Data Definition Language | LSS | Logical subsystem |
| DDR | Double data rate | LUN | Logical Unit Number |
| DIA | Dialog Assembly | LVM | Logical Volume Manager |
| DMS | Database Managed Space | MB | Mega bytes |
| DPF | Data Partitioning Feature | ML | Maintenance Level |
| DSS | Decision Support System | MOLAP | Multidimensional OLAP |
| EDU | Engine Dispatch Unit | MPP | Massively Parallel Processor |
| ERP | Enterprise Resource Planning | NAS | Network Attached Storage |
| ESE | Enterprise Server Edition | NDMP | Network Data Management Protocol |
| ESS | Enterprise Storage Server® | NMON | Nigel Monitor |
| ETL | Extraction, Transformation and Loading | NTFS | NT File System |
| FC | Fibre Channel | NTP | Network Time Protocol |
| FCM | Fast Communication Manager | ODBO | OLE DB for OLAP |
| FP | FixPack | ODS | Operational Data Store |
| GB | Giga Bytes | OLAP | Online Analytical Processing |

| | | | |
|-------------------------|---|-------------|---|
| OLTP | Online Transaction Processing | WAN | Wide Area Network |
| OS | Operating System | WLM | WorkLoad Manager |
| PC | Personal Computer | WP | SAP work process |
| PDCU | Performance Data Collection Utility | XML | Extensible Markup Language |
| POC | Proof of Concept | XMLA | Extensible Markup Language for Analyses |
| PPRC | Peer-to-Peer Remote Copy | | |
| PSA | Persistent Staging Area | | |
| PSSC | Product and Solutions Support Center | | |
| PTF | Program Temporary Fix | | |
| RAID | Redundant Array of Independent Disks | | |
| RAM | Random Access Memory | | |
| RDBMS | Relational DataBase Manager | | |
| RDMA | Remote Direct Memory Access | | |
| RMAN | Oracle® Recovery Manager | | |
| ROLAP | Relational OLAP | | |
| rpm | Rounds per minute | | |
| rsh | Remote shell | | |
| SAN | Storage Area Network | | |
| SAP NetWeaver BI | SAP Business Information Warehouse | | |
| SDDPCM | Subsystem Device Driver Path Control Module | | |
| SDDPCM | Subsystem Device Driver Path Control Module | | |
| SDF | SAN Data Gateway | | |
| SDRAM | Synchronous Dynamic Random Access Memory | | |
| SMI-S | Storage Management Interface Standard | | |
| SMP | Symmetric Multiprocessing | | |
| SMS | System Managed Space | | |
| SMT | Simultaneous Multithreading | | |
| SQL | Structured Query Language | | |
| ssh | Secure shell | | |
| SVC | SAN Volume Controller | | |
| SW | Software | | |
| TB | Terra Bytes | | |
| TCP/IP | Transmission Control Protocol/Internet Protocol | | |
| TDP | IBM Tivoli Data Protection | | |
| TPC | IBM Tivoli Productivity Center | | |
| UDB | Universal DataBase | | |
| VBA | Visual Basic for Application | | |
| VPN | Virtual Private Network | | |

Glossary

Advanced Business Application Programming.

Advanced Business Application Programming (ABAP) is a high level programming language created by SAP. It is currently positioned as the language for programming SAP's Web Application Server, part of the SAP NetWeaver platform for building business applications.

Agent. An agent is a software entity that represents one or more objects by sending notifications regarding events and handling requests from managers (software or hardware entities) to modify or query the objects.

Aggregate. An aggregate stores the data set of an InfoCube in a summarized form on the database. When building an aggregate from the characteristics and navigation attributes from an InfoCube, you can group the data according to different aggregation levels. Remaining characteristics that are not used in the aggregate are summarized. New data is loaded into an aggregate using logical data packages (requests). Aggregates enable you to access InfoCube data quickly for reporting.

Application Programming Interface. An application programming interface (API) is the interface that a computer system, library or application provides in order to allow requests for services to be made of it by other computer programs, and to allow data to be exchanged between them.

array. An array is an ordered collection, or group, of physical devices (disk drive modules) that are used to define logical volumes (LVOLs) or devices. In IBM Enterprise System Storage, an array is a group of disks designated by the user to be managed with a Redundant Array of Independent Disks (RAID).

Authorized Program Analysis Report. An Authorized Program Analysis Report (APAR) is a term used in IBM for a description of a problem with an IBM program that is formally tracked until a solution is provided. An APAR is created or opened after a customer (or IBM) discovers a problem that IBM determines is due to a bug in code. The APAR is given a unique number for tracking. When the support group that maintains the code solves the problem, it develops a program temporary fix (PTF).

Balanced Configuration Unit. A Balanced Configuration Unit (BCU) is composed of software and hardware that IBM has integrated and tested as a pre-configured building block for data warehousing systems. A single BCU contains a balanced amount of disk, processing power, and memory to optimize cost-effectiveness and throughput. IT departments can use BCUs to reduce design time, shorten deployments, and maintain a strong price/performance ratio as they add building blocks to enlarge their BI systems.

Binary Large Object. A Binary Large Object (BLOB) is a collection of binary data stored as a single entity in a database management system. Blobs are typically images, audio, or other multimedia objects, though sometimes binary code is stored as a BLOB. Database support for BLOBs is not universal.

Bit. A bit is either of the digits 0 or 1 when used in the binary numeration system.

Business API. A Business API is used in mySAP to achieve business-related functionalities. It is a remote enabled function module provided by SAP.

Business intelligence. Business intelligence (BI) is a broad category of applications and technologies for gathering, providing access to, and analyzing data for the purpose of helping enterprise users make better business decisions. BI is expert information, knowledge, and technologies efficient in the management of organizational and individual business.

Byte. A byte is a group of eight adjacent binary digits that represent one EBCDIC character.

Call Level Interface. The Call Level Interface (CLI) is a de facto standard software API for SQL-based database management systems, created by The Open Group.

CIM Object Manager. The CIM Object Manager (CIMOM) is the core component of the implementation of the CIM specification. The CIMOM manages the CIM schema, instantiation, communication, and operation of the physical providers that represent the CIM classes.

Cluster. A computer cluster is a group of loosely coupled computers that work together closely so that in many respects they can be viewed as though they are a single computer. The components of a cluster are commonly, but not always, connected to each other through fast local area networks. Clusters are usually deployed to improve performance or availability over that provided by a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

In certain *file system* types like the File Allocation Table (FAT) file system of MS-DOS® or the NTFS file system of Windows NT®, a cluster is the unit of disk space allocation for files and directories. In order to reduce the overhead of managing on-disk data structures, the file system does not allocate individual disk sectors, but contiguous groups of sectors, called clusters. A cluster is the smallest logical amount of disk space that can be allocated to hold a file.

Command-line interface. A command-line interface (CLI) is an interface provided by an operating system that defines a set of commands and enables a user (or a script-like language) to issue these commands by typing text in response to the command prompt (for example, DOS commands or UNIX shell commands).

Common Information Model. The Common Information Model (CIM) is an implementation-neutral, object-oriented schema for describing network management information. The Distributed Management Task Force (DMTF) develops and maintains CIM specifications.

Concurrent copy. Concurrent copy is a facility on a storage server that enables a program to make a backup of a data set while the logical volume remains available for subsequent processing. The data in the backup copy is frozen at the point in time that the server responds to the request.

Concurrent I/O. The Concurrent I/O (CIO) feature of JFS2 provides the capability to not do inode locking of a file except in the case where the inode itself needs to change (such as when the file size is changing). Concurrent I/O also implicitly uses the Direct I/O path. Therefore, CIO is the equivalent of Direct I/O without inode locking. To enable CIO, the application can open a file with the `O_CIO` flag in the `open()` system call, or the file system can be mounted with the `CIO` mount option. The use of named mounts can also be done with Concurrent I/O as with Direct I/O mounts.

Copy Services. Copy Services is made of a collection of optional software features, with a Web browser interface, used for configuring, managing, and monitoring data-copy functions.

Data Definition Language. Data Definition Language (DDL) is a computer language for defining data. XML schema is an example of a pure DDL, a subset of SQL's instructions form another DDL.

Data mart. The data mart interface enables the user to update data from one data target to another. The data mart interface allows the user to update data within an SAP NetWeaver BI system and also between several other systems. If several SAP NetWeaver BI systems are used, the system delivering the data is called the source SAP NetWeaver BI, and the system that is receiving data is called the target SAP NetWeaver BI. The individual Business Information Warehouses in this type of setup are called data marts.

Data provider. A data provider is an object that delivers data for one or more Web items.

Data set. Record of values that belong together in a relational database table. A data set is saved in the relational database management system (DBMS) as a line.

Data warehouse. A data warehouse is a data collection or a database that is created by integrating various data sets and data from external sources. Data warehouses provide users with a global view of the data that has many applications.

DataSource. DataSource is an object that makes data for a business unit available to SAP NetWeaver BI. The DataSource contains a number of logically related fields that are provided in a flat structure for data transfer to SAP NetWeaver BI.

DDR2 SDRAM. Double-data-rate two (DDR2) synchronous dynamic random access memory (SDRAM) is a random access memory technology used for high speed storage of the working data of a computer. It is a part of the synchronous dynamic random access memory family of technologies, which is one of many dynamic random access memory (DRAM) implementations, and is an evolutionary improvement over its predecessor, DDR SDRAM. Its primary benefit is the ability to run its bus at twice the speed of the memory cells it contains, thus enabling faster bus speeds and higher peak throughputs than earlier technologies. This is achieved at the cost of higher latency.

Device Adapter. A Device Adapter (DA) is a physical component of the ESS that provides communication between the clusters and the storage devices. Multiple DAs are connected to the clusters in such a way that any cluster can access a storage device via multiple paths, providing fault tolerance and enhanced availability.

Dialog Assembly (DIA). You use the indicator DIA to determine whether the system is to display a list of missing parts with information about calculated quantities and dates, that is, the components that are not fully available. You can also use this indicator to control whether the quantities and dates of the selected components can be processed interactively in the sales order.

Enterprise Resource Planning. ERP integrates all data and processes of an organization into a single unified system. A typical ERP system will use multiple components of computer software and hardware to achieve the integration.

Extensible Markup Language. The Extensible Markup Language (XML) is a W3C-recommended general-purpose markup language that supports a wide variety of applications. Its primary purpose is to facilitate the sharing of data across different information systems, particularly systems connected via the Internet. XML allows diverse software reliably to understand information formatted and passed in multiple languages. XML is a simplified subset of Standard Generalized Markup Language (SGML).

Fact table. A fact table is a table in the center of an InfoCube star schema. The data part contains all key figures of the InfoCube, and the key is formed by links to the entries of the dimensions of the InfoCube.

Failover. Failover pertains to the process of transferring all control to a single cluster when the other cluster in the storage unit fails.

Fibre Channel. Fibre Channel (FC) is a technology for transmitting data between computer devices. It is especially suited for attaching computer servers to shared storage devices and for interconnecting storage controllers and drives.

FlashCopy. This is a function on the IBM Enterprise Storage Server that can create a point-in-time copy of data while an application is running. A FlashCopy image is a space-efficient image of the contents of part of the SAN file system at a particular moment. A FlashCopy mapping is a continuous space on a direct access storage volume, occupied by or reserved for a particular data set, data space, or file. A FlashCopy service is a copy service that duplicates the contents of a source virtual disk (VDisk) on a target VDisk. In the process, the original contents of the target VDisk are lost.

Global Copy. This is an optional capability of the DS8000 remote mirror and copy feature that maintains a fuzzy copy of a logical volume on the same DS8000 or on another DS8000. In other words, all modifications that any attached host performs on the primary logical volume are also performed on the secondary logical volume at a later point in time. The original order of update is not strictly maintained.

Global Mirror. This is an optional capability of the DS8000 remote mirror and copy feature that provides a two-site extended distance remote copy. Data that is written by the host to the storage unit at the local site is automatically maintained at the remote site.

High Availability Cluster Multiprocessing. HACMP offers robust high availability and disaster recovery, with faster failover and easier-to-use configuration facilities for IBM System p servers.

IBM TotalStorage DS8000. It is a member of IBM TotalStorage Resiliency Family of storage servers and attached storage devices (disk drive modules). The DS8000 delivers high-performance, fault-tolerant storage and management of enterprise data, affording access through multiple concurrent operating systems and communication protocols. High performance is provided by multiple symmetrical multiprocessors, integrated caching, RAID support for the disk drive modules, and disk access through a high-speed serial storage architecture interface.

IBM TotalStorage. It is the brand name used to identify storage products from IBM, including the IBM TotalStorage DS8000.

InfoArea. An InfoArea groups meta-objects together in the Business Information Warehouse. Every data target is assigned to an InfoArea. The resulting hierarchy is then displayed in the Administrator Workbench. In addition to their property as a data target, InfoObjects can also be assigned to different InfoAreas using InfoObject catalogs.

InfoCube. An InfoCube is a quantity of relational tables that are created according to the star schema. An InfoCube describes a self-contained data set (from the reporting view), for example, for a business-oriented area. InfoCubes are objects that can function as data targets as well as InfoProviders.

InfoObject. Business evaluation objects (for example, customers or sales) are called InfoObjects in SAP NetWeaver BI. InfoObjects are subdivided into characteristics, key figures, units, time characteristics, and technical characteristics (such as request numbers).

InfoPackage. An InfoPackage describes which data in a DataSource should be requested from a source system. The data can be precisely selected using selection parameters (for example, only controlling area 001 in period 10.1997). An InfoPackage can request the following types of data: transaction data, attributes for master data, hierarchies for master data, and master data texts.

InfoProvider. An InfoProvider is an analysis-relevant view of an SAP NetWeaver BI object for which queries in SAP NetWeaver BI can be created or executed. There are two types of InfoProviders. One type includes objects that contain physical data. These are known as data targets, such as InfoCubes, ODS objects, and InfoObjects (characteristics with attributes, texts, or hierarchies). The other type includes objects that display no physical data storage, such as InfoSets, RemoteCubes, SAP RemoteCubes, and MultiProviderserm3 definition.

InfoSet. An InfoSet is a semantic view of ODS objects and InfoObjects (characteristics with master data) that allows you to create reports on these objects, particularly on the joins between these objects. Unlike the classic InfoSet, this view of data is SAP NetWeaver BI-specific. In the InfoSet builder, InfoSets are created and changed. InfoSets allow you to use the query designer to define reports.

InfoSource. An InfoSource is a quantity of all available data for a business event, or type of business event (for example, cost center accounting). An InfoSource is a quantity of information that has been grouped together from information that logically belongs together. InfoSources can contain transaction data or master data (attributes, texts, and hierarchies). An InfoSource is always a quantity of InfoObjects that belong together logically. The structure where they are stored is called a communication structure.

JCA. Java EE Connector Architecture (JCA) is a Java-based technology solution for connecting application servers and enterprise information systems as part of enterprise application integration solutions. While JDBC is specifically used to connect Java EE applications to databases, JCA is a more generic architecture for connection to older systems (including databases). The BI Java Connectors are JCA-compliant.

JDBC. JDBC is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases.

Logical subsystem. A logical subsystem (LSS) represents the logical functions of a storage controller that allow one or more host I/O interfaces to access a set of devices. The controller groups the devices according to the addressing mechanisms of the associated I/O interfaces. One or more LSSs exist on a storage controller. In general, the controller associates a given set of devices with only one LSS.

Logical Unit Number. The Logical Unit Numbers (LUNs) are provided by the storage devices attached to the SAN. This number provides you with a volume identifier that is unique among all storage servers. The LUN is synonymous with a physical disk drive or a SCSI device. For disk subsystems such as IBM Enterprise Storage Server, a LUN is a logical disk drive. This is a unit of storage on the SAN that is available for assignment or unassignment to a host server.

Logical Volume Manager. The Logical Volume Manager (LVM) is a set of system commands, library routines, and other tools that allow the user to establish and control logical volume storage. The LVM maps data between the logical view of storage space and the physical disk drive module.

Master data. The master data includes the permitted values for a characteristic, also called characteristic values. Characteristic values are discrete names.

Metadata. Metadata are data that describe other data. Generally, a set of metadata describes a single set of data, called a resource.

Metro Mirror. The Metro Mirror technology is a function of a storage server that maintains a consistent copy of a logical volume on the same storage server or on another storage server. All modifications that any attached host performs on the primary logical volume are also performed on the secondary logical volume.

MOLAP. MOLAP is an analytic tool designed to allow analysis of data through the use of a multidimensional data model. MOLAP differs significantly from ROLAP in that it requires the pre-computation and storage of information in the InfoCube — the operation known as processing. MOLAP stores this data in an optimized multidimensional array storage, rather than in a relational database.

Multidimensional data. Multidimensional data are data in a multidimensional database. Data can include basic data values (loaded from an external source) that represent combinations of the lowest level of members in the dimensions of the database, data values that are calculated from the base data values, and rolled up data values that are created by combining values for members in dimension hierarchies. They are data suitable for business analytics. In the BI Java SDK, the term multidimensional data is used synonymously with OLAP data.

MultiProvider. A MultiProvider is a type of InfoProvider that combines data from several InfoProviders and makes it available for reporting. The MultiProvider itself contains no data. Its data comes exclusively from the InfoProviders on which it is based. You can assemble a MultiProvider from different combinations of InfoProviders. MultiProviders, like InfoProviders, are objects or views that are relevant for reporting.

ODS. An Object DataSource is an object that stores consolidated and cleaned transaction data on a document level. An ODS object describes a consolidated data set from one or several InfoSources. This data set can be evaluated using a BEx query. An ODS object contains a key (for example, document number, position), as well as data fields that, as key figures, can also contain character fields (for example, customer). In contrast to multidimensional data stores for InfoCubes, data in ODS objects is stored in transparent, flat database tables.

OLAP. A multidimensional, multi-user, client/server computing environment for users who need to analyze consolidated enterprise data in real time. OLAP systems feature zooming, data pivoting, complex calculations, trend analyses, and data modeling. OLAP is an approach to quickly provide the answer to analytical queries that are dimensional in nature. It is part of the broader category business intelligence, which also includes extract transform load (ETL), relational reporting, and data mining. Databases configured for OLAP employ a multidimensional data model, allowing for complex analysis and queries with a rapid execution time. The output of an OLAP query is typically displayed in a matrix format.

OLE-DB OLE DB is an API designed by Microsoft for accessing different types of data stores in a uniform manner. It is a set of interfaces implemented using the Component Object Model (COM). OLE DB separates the data store from the application that needs access to it through a set of abstractions that include the DataSource, session, command, and rowsets. The consumers are the applications that need access to the data, and the provider is the software component that implements the interface and therefore provides the data to the consumer.

Peer to Peer Remote Copy. PPRC is a remote-copy service that provides a synchronous copy of a volume or disk for disaster recovery, device migration, and workload migration.

Persistent Staging Area (PSA). PSAs are transparent database tables, in which request data is stored in the form of the transfer structure. A PSA is created per DataSource and source system. It represents an initial store in SAP NetWeaver BI, in which the requested data is saved unchanged for the source system.

Process. A process is a naturally occurring or designed sequence of changes of properties/attributes of a system/object.

Redundant Array of Independent Disks. Redundant Array of Independent Disks (RAID) is a methodology of grouping disk drives for managing disk storage to insulate data from a failing disk drive. RAID 5 is a type of RAID that optimizes cost-effective performance while emphasizing use of available capacity through data striping. RAID 5 provides fault tolerance for up to two failed disk drives by distributing parity across all the drives in the array plus one parity disk drive. The DS8000 automatically reserves spare disk drives when it assigns arrays to a device adapter pair (DA pair). RAID 10 is a type of RAID that optimizes high performance while maintaining fault tolerance for up to two failed disk drives by striping volume data across several disk drives and mirroring the first set of disk drives on an identical set. The DS8000 automatically reserves spare disk drives when it assigns arrays to a device adapter pair (DA pair).

Remote Mirror and Copy. A feature of a storage server that constantly updates a secondary copy of a logical volume to match changes made to a primary logical volume. The primary and secondary volumes can be on the same storage server or on separate storage servers.

RemoteCube. A RemoteCube is an InfoCube whose transaction data is not managed in the Business Information Warehouse, but externally. Only the structure of the RemoteCube is defined in SAP NetWeaver BI. The data for reporting is read from another system using a BAPI.

ROLAP. ROLAP is an analytic tool designed to allow analysis of data through the use of a multidimensional model. ROLAP differs from MOLAP in that it does not require the pre-computation and storage of information. ROLAP tools access the data in a relational database and generate SQL queries to calculate information at the appropriate level when a user requests it. With ROLAP, it is possible to create additional database tables (summary tables or aggregations) that summarize the data at any desired combination of dimensions.

Roll back. To remove changes that were made to database files under commitment control since the last commitment boundary.

Roll forward. To update the data in a restored database or tablespace by applying changes recorded in the database log files.

Roll up. Loads data packages (requests) for an InfoCube that are not yet available into all aggregates of the InfoCube. After it has been rolled up, the new data is used in queries.

SAP work process. SAP is a multi-process application as opposed to a multi-threaded architecture. SAP multi-tasks work on a number of defined processes. In configuring the SAP system, the administrator defines how many processes of what kind the system should manage. The types of SAP work processes are batch, dialog, enqueue, update, and update2.

Source system. System that makes the Business Information Warehouse available for data extraction.

Staging. A process that prepares (stages) data in a Data Warehouse.

Subsystem Device Driver Path Control Module.

Subsystem Device Driver Path Control Module (SDDPCM) is a loadable path control module designed to support the multipath configuration environment on IBM TotalStorage Enterprise Storage Server, IBM System Storage SAN Volume Controller, and IBM TotalStorage DS family. When the supported devices are configured as MPIO-capable devices, SDDPCM is loaded and becomes part of the AIX MPIO FCP (Fibre Channel Protocol) device driver. The AIX MPIO device driver with the SDDPCM module enhances the data availability and I/O load balancing. SDDPCM manages the paths to provide:

- High availability and load balancing of storage I/O
- Automatic path-failover protection
- Concurrent download of licensed internal code
- Prevention of a single-point-failure caused by host bus adapter, Fibre Channel cable, or host-interface adapter on supported storage

Symmetric Multiprocessing (SMP). Symmetric Multiprocessing is a multiprocessor computer architecture where two or more identical processors are connected to a single shared main memory. Most common multiprocessor systems today use an SMP architecture.

Target system. Target Business Information Warehouse SAP NetWeaver BI System to which another SAP NetWeaver BI System is connected as a source system, and into which you can load data using export DataSources.

Virtual machine facility. This is a virtual data processing machine that appears to the user to be for the exclusive use of that user, but whose functions are accomplished by sharing the resources of a shared data processing system. This is an alternate name for the VM/370 IBM operating system.

XML for Analysis. XMLA is a protocol specified by Microsoft for exchanging analytical data between client applications and servers using HTTP and SOAP as a service on the Web. XML for Analysis is not restricted to any particular platform, application, or development language. Using XML for Analysis in the Business Information Warehouse allows a third-party reporting tool that is connected to the SAP NetWeaver BI to communicate directly with the Online Analytical Processing (OLAP) processor.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 418. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *Infrastructure Solutions: Design, Manage, and Optimize a 20 TB SAP NetWeaver Business Intelligence Data Warehouse*, SG24-7289
- ▶ *Advanced POWER Virtualization on IBM eserver p5 Servers: Architecture and Performance considerations*, SG24-5768
- ▶ *Advanced POWER Virtualization on IBM System p5*, SG24-7940-01
- ▶ *AIX 5L Practical Performance Tools and Tuning Guide*, SG24-6478
- ▶ *IBM System Storage DS8000 Series: Architecture and Implementation*, SG24-6786
- ▶ *IBM System Storage Tape Library Guide for Open Systems*, SG24-5946-04
- ▶ *IBM TS3500 Tape Library with System z Attachment: A Practical Guide to TS1120 Tape Drives and TS3500 Tape Automation*, SG24-6789
- ▶ *IBM System Storage Solutions Handbook*, SG24-5250

Online resources

These Web sites are also relevant as further information sources:

- ▶ How to tune the OLAP cache
<https://www.sdn.sap.com/irj/sdn/howtoguides>
- ▶ SAP-note 980067
http://www.sbm.com.sa/files/IBM_DB2_Optimized_for_SAP.pdf
<ftp://ftp.software.ibm.com/software/data/pubs/papers/DB2-SAP-compression.pdf>
- ▶ Performance tuning of the OLAP cache
<http://service.sap.com/bi>
- ▶ SAP Note 930487 DB6
<https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/webcontent/uuid/e02ce76b-588c-2910-7dab-faa1e94ad012>
- ▶ The IBM System p servers
<http://www-03.ibm.com/systems/p/>
- ▶ Relative performance metric for System p servers (rPerf)
<http://www-03.ibm.com/systems/p/hardware/rperf.html>

- ▶ SDD Web site
<http://www-1.ibm.com/support/search.wss?rs=540&tc=ST52G7&dc=D600&dtm>
- ▶ SAP Standard Application Benchmark
<http://www.sap.com/solutions/benchmark/measuring/index.epx>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Numerics

64 bits 195

A

ABAP 151, 229
ACTIVATE DATABASE 280
ADMIN_CMD 280
administration 3
Advanced POWER Virtualization
 see APV
aggregate
 aggregate tables 166, 176
 dependent 63
aggregate rollup process 62
aggregate tree 63
Aggregates 104
aggregation 3
AGGRFILL 83
AIX 5.4 314
AIX 5L Version 5.2 289
AIX 5L version 5.2 293
AIX 5L Version 5.3 289
AIX 5L version 5.3 293–294
AIX command
 fileplace 170
 lsvg 172
ALTER BUFFER POOL 281
ALTER TABLE 232
ALTER TABLESPACE 265
Analysis Processor Designer 48
application server 66
APV 290
architecture 5
ARCHRETRYDELAY 353
ASTAT 9
ATTRIBCHAN 83

B

BACKUP DATABASE 357
base frame 319
battery backup unit
 see BBU
BBU 319
BEx 48, 53
BGD process 64
BGD_extract process 57, 62
BGD_loading process 57
BGD_roll process 63
BGD_trigger process 56
block size 81
brarchive 353
brbackup 373
Buffer manipulator process 352

buffer pool 188
bufferpool quality 45
business dimensions 50
Business Explorer
 see BEx

C

Capacity upgrade on demand 289
Capped mode 291
ch_free 202
ch_free_bottom 202
CHECKAGGR 83
chuser 315
CIM 357
CIO 310
Cisco 309
CLI 151
CLP 357
Collocation 340
collocation 355
command
 db2imigr 160
 db2set 194
 MIGRATE DATABASE 160
 RESTORE DATABASE 160
Command Line Processor
 see CLP
COMPRESS 229
compression 44
compression dictionary 229
compression estimation 229
concurrent I/O 265
Concurrent IO
 see CIO
CONDAGGR 83
CSV file 10

D

DA 320
data facts 50
data flow 52
data row compression 228
Data Sources 49
Data Target 49
DB2
 HPU 152, 155
 HPU configuration 155
 HPU control file 154
DB2 BACKUP 352
DB2 buffer cache 315
DB2 command
 db2inidb 355
 redistribute 151
DB2 log manager 353

DB2 PE 11
 DB2 Performance Expert 185
 DB2 RESTORE 352
 DB2_APM_PERFORMANCE 196
 DB2_BLOCK_ON_LOG_DISK_FULL 197
 DB2_CORRELATED_PREDICATES 196
 DB2_EEE_PARALLEL_BACKUP 370
 DB2_FORCE_APP_ON_MAX_LOG 196
 DB2_FORCE_FCM_BP 197
 DB2_HASH_JOIN 196
 DB2_OPT_MAX_TEMP_SIZE 194
 DB2_PARALLEL_IO 171, 197
 DB2_STRIPED_CONTAINERS 196
 DB2_WORKLOAD 194
 DB2ATLD_PORTS 196
 DB2CODEPAGE 197
 DB2COMM 197
 DB2DBDFT 197
 DB2ENVLIST 197
 db2Export API 141
 db2fcmd 316
 db2hpu 157
 db2Import API 141
 db2inidb 355
 db2inidb as mirror 371
 DB2INSTANCE 194
 DB2INSTPROF 194
 DB2MEMDISCLAIM 196
 DB2MEMMAXFREE 196
 db2move 151
 DB2NODE 194
 DB2PATH 194
 db6conv 151
 DBACOCKPIT 229
 dbpartitionnum 229
 DDL 153
 DDM 318
 DDR2 288
 device adapter
 see DA
 DIA_submit process 57, 62
 DIA_update 76
 DIA_update process 57, 62
 DIA_WP 19
 dimension identifier 51
 dimension table 50
 direct I/O 265
 disk drive module
 see DMM
 DLPAR 293
 DLPAR-operations 290
 DMS 195–196, 235
 DP for mySAP 354
 DPF 202, 353
 DS Open API 357
 DS8300 Turbo Model 932 318
 dynamic LPAR 290

E

environment 194

EtherChannel 306
 Ethernet 306–307
 Ethernet switch 319
 ETL 52
 Excel spreadsheet 191
 expansion frame 320
 Expert Mode 59
 Explain GUI tool 141
 export 350
 extended start schema 50
 EXTENTSIZE 171

F

fact table 16, 50
 Fact-table 166
 FAILARCHPATH 353
 fan sense card 319
 FCM 202, 316
 FCM_NUM_BUFFERS 201
 fcm_num_buffers 202
 FCM_NUM_CHANNELS 201
 fcm_num_channels 202
 Fibre Channel 319
 Fibre Channel-Arbitrated Loop 322
 FICON 319
 field compression 233
 File System cache 208
 file system cache 265
 FILE SYSTEM CACHING 265
 firmware 289
 first stage 3
 frame power 319–320

H

HA 320
 HACMP 5
 Hardware Management Console
 see HMC
 hashing key 152, 154
 hdisk 293, 301
 HMC 290
 host adapter
 see HA
 host bus adapter 339
 Hypervisor scheduler 312

I

I/O enclosure 319
 I/O wait CPU 263
 I/O wait time 171
 IBM Linear Tape Open Ultrium Technology 320
 IBM System Storage DS8300 318
 IBM System Storage TS1030 Tape Drive 318
 IBM System Storage TS3500 Tape Library 318
 IBM TotalStorage SAN140M 318
 IBM TotalStorage SAN256M 318
 IBM TPC 12
 IBM Virtual I/O Server 292

- IDoc 73
- import 350
- Infiniband 307
- Info IDoc 73
- InfoCubes 49
- InfoObjects 49
- InfoPackage 57
- InfoProvider 49
- information model 48
- InfoSets 49
- InfoSource 49
- Infrastructure 3
- Instance parameters 75
- Inter- Switch Link
 - see ISL
- inter-disk setting 173
- ISL 323

J

- J2_METAMAP 195
- JFS2 196
- JFS2 dioCache 316
- jfs2log 303

K

- KEYCARD 196

L

- LAN 307
- LAN-Free 349
- LAN-free 340
- LANfree 361
- lan-free 362
- lanfree 375
- LDR_CNTRL 316
- least recently used
 - see LRU
- LIM 323
- Line Module
 - see LIM
- load process 56, 72
- loadprocedure fast COMPRESS 277
- LOB 229
- log 202
- log file status
 - archived 354
 - offline retained 354
 - online active 354
 - online retained 354
- log shipping 355
- LOGARCHMETH1 353
- LOGARCHMETH2 353
- Logical Partitioning
 - see LPAR
- Logical Volume Manager
 - see LVM
- LPAR 290
 - dynamic LPAR 290

- LPARMon 10
- LRU 70, 313
 - list-based-LRU 314
- lrud 314
- lsattr 308
- lscfg 307
- LTO-3 drive 320
- LUN 301, 326
- LV INTER-POLICY 303
- LVM 5, 302

M

- MANUAL 281
- MAX_LOG 196
- MAXAGENTS 200
- maxclient% 312, 314
- maxfree 313
- MAXLINES 72
- maxperm% 312, 314
- MAXPROC 112
- MAXPROCS 62, 73, 76, 119, 122
- Maxprocs 19, 23
- MAXSESSIONS 370
- MAXSIZE 72, 119
- MCM 288
- MDC 271
- Media controller process 352
- Micro-Partitioning 288
- Micropartitioning 289, 316
- micro-partitioning 14
- minfree 313
- monitoring interval 313
- MPIO 301
- MSJN 195
- MTU size 314
- multichip modules
 - see MCM
- MultiProvider 49, 67

N

- Nagle algorithm 314
- NIM server 295
- NLJN 195
- NMON 10, 188, 341
 - analyser 191
- NO FILE SYSTEM CACHING 265
- NO_SORT_MGJOIN 195
- NO_SORT_NLJOIN 195
- NUM_POOLAGENTS 200
- NUMARCHRETRY 353

O

- ODS 49
- OLAP 45, 48, 53
 - processor 48
- OLAP cache 16, 70, 89, 100
- Online
 - upload 3

online
 query load 3
Online Analytical Processing
 see OLAP
OPEN SESSIONS clause 352
Operational Data Store
 see ODS
optimizer 229
OVERFLOWLOGPATH 353

P

page_steal_method 314
PARALLELISM 352
partition
 Dedicated processor 290
 micro-partition 290
 Shared processor 290
partitioning key 152
PARTITIONNUM 281
PCI-X 288
PDCU 12
performance 3
Performance Data Collection Unit
 see PDCU
Peripheral Component Interconnect Extended (\$nopage>
 see PCI-X 288
permanent data object 229
Persistent Staging Area
 see PSA
pin/unpin overhead 315
pipe 153
policy-based 349
PORT_RANGE 196
POWER Hypervisor 289, 316
POWER5 14, 288
POWER5+ 8, 288
PPS 319
prefetch 170
 sequential 171
PREFETCHSIZE 171
primary power supply
 see PPS
process chain 56, 63
Processor
 Logical 291
 Physical 291
 Shared Processor Pool 291
 Virtual 291
PSA 49

Q

query load 3

R

r3load 152, 157
rack power control
 see RPC
rdisp/rfc_max_login 76

rdisp/rfc_max_own_login 76
rdisp/rfc_max_own_used_up 76
rdisp/rfc_min_wait_dia_wp 75
rdisp/wp_no_btc 75, 82
rdisp/wp_no_dia 75
RDMA 307
recommendations 310
Red Hat 293
Redbooks Web site 418
 Contact us xii
registry 194
release-behind 310
Remote Direct Memory Access
 see RDMA
REORG 234
RESOURCETIMEOUT 370
RESTORE DATABASE 357
RFC request 75
RIO-G loop 320
ROIDOCPRMS table 57
ROLLUP 83
ROOSPRMS table 57, 74
round-robin load balancing 58
RPC 319
rPerf 288
RSA1 transaction 61, 65, 69
RSARFCLD 76
RSBATCH transaction 83
RSDDSTATWHM table 60
rsh 189
RSMONMESS table 60
RSPC transaction 56, 63
RSRCACHE transaction 102
RUNSTATS 229, 232, 234
RZ10 transaction 76
RZLLICLASS table 58

S

SAN 301
SAP
 SAPinst 157
SAP Business Intelligence 83
SAP GUI 294
SAP Job-Chains 16
SAP R3load 277
SAP transaction
 SM37 9
 ST03 9
sapdata-file systems 310
SBIW 57
SBIW transaction 73
SDDPCM 301
SDRAM 288
SE16 transaction 58, 75
SEA 292
self_tuning_mem 280
server-free 349, 362
Shared 292
Shared Ethernet Adapter 292
shared nothing architecture 300, 338

- Shared Processor Partition 316
- S-HMC 319
- Simultaneous Multiprocessing
 - see SMP
- Simultaneous multi-threading
 - see SMT
- SM37 57
- SM37 transaction 56, 64
- SM66 transaction 57
- SMLG transaction 57–58
- SMP 288
- SMS 235
- SMT 14, 114, 288
- split mirror 355
- splitint 358
- SPRO transaction 81
- SQL 53
- ST02 transaction 103
- ST03 transaction 59, 65, 71, 104
- stage
 - first 3
 - second 3
- STATFRQU 73
- STMM 281
- storage area network architecture 325
- Storage Data Array 171
- Storage Hardware Management Console
 - see S-HMC
- strict_maxclient 312
- Subsystem Device Driver Path Control Module
 - see SDDPCM
- SUSE 293

T

- table_name 230
- table_schema 229
- tape 349
- TCP_NODELAY 314
- tcp_nodelayack 314
- tdphdwb2 358–359
- TLB 315
- TotalStorage Productivity Center
 - see TPC
- tpmon_acc_str 186
- tpmon_client_app 186
- tpmon_client_userid 186
- tpmon_client_wkstn 186
- transaction SBIW 57
- transfer rule 52
- trict_maxclient 314

U

- Uncapped mode 291
- Uncapped weight 291
- uncompress 246
- update rule 52
- Update Rules 49
- upload 3
- UTIL_HEAP_SZ 352

- util_impact_lim 353
- utility
 - INSPECT 229

V

- VBA 191
- VCPU 312
- virtual adapter 292
- Virtual I/O 289–291
- virtual processors 290
- Virtual SCSI 292
- virtual SCSI 292
- virtualization 316, 330
- Visual Explain GUI 141
- vmo 315
- vmstat 313
- Volume Group 302
- vpm_xvcpus 312
- vUsers 19

W

- WITH BUFFERS 352
- write resume 359
- write resume mode 359
- write suspend 359
- write suspend mode 359

X

- XINT_PROFILE 367
- XML data object 229
- XML file 52

Archived



Infrastructure Solutions: Design, Manage, and Optimize a 60 TB SAP NetWeaver Business Intelligence Data Warehouse

Archived



Infrastructure Solutions: Design, Manage, and Optimize a 60 TB SAP NetWeaver Business Intelligence Data Warehouse



**Scalability study of
SAP NetWeaver BI on
IBM System p5 and
DB2 9**

**Architectural
description, test
results, lessons
learned**

**Management of
solution using DS8000
and Tivoli products**

In order to improve the performance and operational efficiency of businesses worldwide, a customer using SAP solutions wanted to establish a global business program to define and implement a standardized, group-wide business process architecture and associated master data for the parameterization of the group software tools. The expected growth of the number of users and the size of the database would be at a level never reached by other customers, however, so IBM was asked to undertake the following:

- ▶ Test the application to be sure that it could sustain such growth.
- ▶ Prove the manageability of the solution.
- ▶ Provide recommendations to optimize the infrastructure architecture.

This IBM Redbooks publication describes the testing that was done in terms of performance and manageability in an SAP NetWeaver BI and DB2 environment on IBM System p when scaling a client's solution to a data warehouse of 60 terabytes (TB).

This book resulted from a joint cooperative effort among the PSSC, the IBM/SAP International Competency Center, the DB2-SAP Center of Excellence, SAP AG, and a customer employing more than 250,000 employees with factories and logistics operations in almost every country in the world.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**

SG24-7385-00

ISBN 0738488534