Comprehensive, End-to-End Insight into Oracle's Flagship Database Machine

Oracle Exadata

EXPERT'S HANDBOOK

Tariq Farooq | Charles Kim | Nitin Vengurlekar
Sridhar Avantsa | Guy Harrison | Syed Jaffar Hussain

FREE SAMPLE CHAPTER

Oracle Exadata Expert's Handbook

This page intentionally left blank

Oracle Exadata Expert's Handbook

Tariq Farooq
Charles Kim
Nitin Vengurlekar
Sridhar Avantsa
Guy Harrison
Syed Jaffar Hussain

Addison-Wesley

New York • Boston • Indianapolis • San Francisco
Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2015935451

ISBN-13: 978-0-321-99260-4
ISBN-10: 0-321-99260-1
Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana
First printing, June 2015

Contents

Preface

Blazingly fast, Exadata is Oracle's complete database machine—with unparalleled performance brought about by engineering hardware and software technologies from Oracle and Sun. Exadata has been widely embraced by enterprise users worldwide, including government, military, and corporate entities.

Authored by a world-renowned veteran author team of Oracle ACEs/ACE directors with a proven track record of multiple bestselling books and an active presence on the Oracle speaking circuit, this book is a blend of real-world, hands-on operations guide and expert handbook for Exadata Database Machine administrators (DMAs).

Targeted for Oracle Exadata DBAs and DMAs, this expert's handbook is intended to serve as a practical, technical, go-to reference for performing administration operations and tasks for Oracle's Exadata Database Machine. This book is a

- Practical, technical guide for performing setup and administration of Oracle's Exadata Database Machine
- Expert, pro-level Exadata handbook
- Real-world, hands-on Exadata operations guide
- Expert deployment, management, administration, support, and monitoring guide and handbook
- Practical, best-practices advice from real-life Exadata DMAs

The authors have written this handbook for an audience of intermediate-level, power, and expert users of the Exadata Database Machine.

This book covers both 11*g* and 12*c* versions of the underlying Exadata software.

Acknowledgments

Tariq Farooq

To begin with, I would like to express my endless gratitude and thanks for anything and everything in my life to the Almighty ALLAH, the lord of the worlds, the most gracious, the most merciful.

I dedicate this book to my parents, Mr. and Mrs. Abdullah Farooq; my amazing wife, Ambreen; and my wonderful kids, Sumaiya, Hafsa, Fatima, and Muhammad-Talha; my nephews, Muhammad-Hamza, Muhammad Saad, Abdul-Karim, and Ibrahim. Without all of their perpetual support, this book would not have come to fruition. My endless thanks to them as I dedicated more than two years of my spare time to this book, most of which was on airplanes and in late nights and weekends at home.

My heartfelt gratitude to my friends at the Oracle Technology Network (OTN), colleagues in the Oracle ACE fellowship, my co-workers and everyone else within the Oracle community, as well as my workplace for standing behind me in my quest to bring this project to completion, especially Jim Czuprynski, Mike Ault, Bert Scalzo, Sandesh Rao, Bjoern Rost, Karen Clark, Vikas Chauhan, Suri Gurram, John Casale, and my buddy Dave Vitalo.

Considering that Exadata was the hot new kid on the Oracle block, I had been contemplating and reaching out to a lot of folks about writing a book on Exadata for over a year, before the stars got aligned and we started working on this project. From inception to writing to technical review to production, authoring a book is a complex, labor-intensive, lengthy, and at times painful process; this book would

not have been possible without the endless help and guidance of the awesome Addison-Wesley team. A very special thank-you to Greg Doench, the executive editor, and all the other folks at Addison-Wesley, who stood like a rock behind this project. Kudos to the book review and editorial team at Addison-Wesley for a job well done. A special thanks to Nabil Nawaz for contributing and helping out with the authoring process.

Finally, many appreciative thanks to my buddies and co-authors—Charles, Nitin, Sridhar, Syed, and Guy—for the amazing team effort that allowed us to bring this book to you, my dear reader. My sincerest hope is that you will learn from this book and that you will enjoy reading it as much as we did researching and authoring it.

Charles Kim

I dedicate this book to my father, who passed away to be with the Lord earlier this year. I thank my wonderful wife, Melissa, who always supported my career aspirations no matter how crazy they seemed. Last, I would like to thank my three precious sons, Isaiah, Jeremiah, and Noah, for always making me smile.

Nitin Vengurlekar

I would like to thank my family, Nisha, Ishan, Priya, and Penny, and especially my mother, father, and Marlie.

Sridhar Avantsa

Any success that I have had or will have is primarily due to the support, encouragement, and guidance I have received from people I have had the distinct honor, pleasure, and good luck to have in my life. There are a lot more folks who have had a profound impact on my life, but the individuals listed here have been my Rocks of Gibraltar over the years. I dedicate this book to these individuals as a small token of my appreciation and thanks. I am forever indebted to them for the love, encouragement, and support they have provided over the years.

To my parents, Mr. Avantsa Krishna and Mrs. Avantsa Manga, who for years denied themselves any pleasures or luxuries to ensure that they provided me with the education, facilities, and opportunities to build a solid foundation for success in the future. To my grandfather, Mr. A. V. Rayudu, whose unflinching faith in his grandchildren helped us to really believe in ourselves.

To my elder brother, Srinivas Avantsa, who understood and knew me better than I ever did, for guiding and supporting me through my formative years. If not for him, I might never have made it to college. To my cousin, Nand Kishore Avantsa, who introduced me to the fascinating world of computer science.

To my wife of the last 19 years, Gita Avantsa, the love of my life and the very best mother my children could ask for. She has stood by me and supported me over the years, through thick and thin, with a smile that always reminds me that "everything is going to be all right." Without her support and understanding, balancing work with writing a book would have been impossible.

To my sons, eight-year-old Nikhil and seven-year-old Tarun, who have put up with me on a daily basis. Your innocence, intelligence, and resiliency never cease to amaze me.

Last, but not least, I want to thank my coauthors for including me in this journey, and my team at Rolta AdvizeX for helping me—among them Rich Niemiec, Robert Yingst, and Michael Messina stand out.

Thank you ever so much, everybody.

Guy Harrison

I dedicate this work to the memory of Catherine Maree Arnold (1981–2010).

Thanks as always to Jenni, Chris, Kate, Mike, and William Harrison who give my life meaning and happiness. Thanks Tariq and Greg for giving me the opportunity to work with Sridhar, Syed, Charles, Sahid, Bert, Michael, Nitin, Nabil, and Rahaman.

Syed Jaffar Hussain

I would like to dedicate this book to my parents, Mr. and Mrs. Saifulla; my wife, Ayesha; my three little champs, Ashfaq, Arfan, and Aahil; and the entire Oracle community.

First and the foremost, I thank the Almighty for giving me everything in life and my parents for giving me wonderful life and making me what I am today. Also, I owe a very big thank-you to my family for allowing me to concentrate on writing assignments and helping me complete the project on time. Beyond a doubt, the project wouldn't have been possible without the tremendous moral support and encouragement of my wife, friends, and colleagues. Thank you, everyone, once again.

I would like to thank my management (Khalid Al-Qathany, Hussain Mobarak AlKalifah, Majed Saleh AlShuaibi), my dearest friends (Khusro Khan, Mohammed Farooqui, Naresh Kumar, Shaukat Ali, Chand Basha, Gaffar Baig, Hakeem, Mohsin, Inam Ullah Bukhari, Rizwan Siddiqui, Asad Khan), my brother Syed Shafiullah, fellow colleagues, well-wishers, supporters, nears, and dears for their immense support and constant encouragement. I can't forget thanking Mr. Vishnusivathej, Nassyam Basha, YV Ravi Kumar, Aman Sharma, and Karan and Asad Khan for helping me while writing this book.

I am also thankful from the bottom of my heart to the official technical reviewers (Sandesh Rao and Javed) for taking some valuable time from their busy schedules to review our book and for providing great input. I can't conclude the list without mentioning the members of the Addison-Wesley team who put this project together.

Nabil Nawaz

I would like to first thank my wife Rabia and the kids for being patient while I was busy contributing to this book and away from family for several weekends—this project ended up taking more time than I expected. I am very lucky to have an understanding family that supported me on my first book!

I am grateful to Charles Kim for inviting me to be part of this amazing book; he also spent time revising the contributions I made and I really appreciate his guidance and all of his help. Charles has been an excellent mentor and is always willing to help anyone learn about technology.

Thank you also to Bane Radulovic from Oracle Support for all of his time to discuss and review the Exadata Stack upgrade process in detail. Without him I would have never been able to contribute to the upgrade chapter.

About the Authors

Tariq Farooq is an Oracle Technologist/Architect/Problem-Solver and has been working with various Oracle Technologies for more than 24 years in very complex environments at some of the world's largest organizations. Having presented at almost every major Oracle conference/event all over the world, he is an award-winning speaker, community leader/organizer, author, forum contributor, and tech blogger. He is the founding president of the IOUG Virtualization & Cloud Computing Special Interest Group and the BrainSurface social network for the various Oracle communities. Tariq founded, organized, and chaired various Oracle conferences including, among others, the OTN Middle East and North Africa (MENA) Tour, the OTN Europe Middle East and Africa (EMEA) tour, VirtaThon (the largest online-only conference for the various Oracle domains), the CloudaThon and RACaThon series of conferences, and the first ever Oracle-centric conference at the Massachusetts Institute of Technology (MIT) in 2011. He was the founder and anchor/show-host of the VirtaThon Internet Radio series program. Tariq is an Oracle RAC Certified Expert and holds a total of 14 professional Oracle Certifications. Having authored over 100 articles, white-papers, and other publications, Tariq is the coauthor of the *Expert Oracle RAC 12c*, *Building DB Clouds in Oracle 12c*, *Oracle Problem-Solving*, and *Troubleshooting Oracle* books. Tariq has been awarded the Oracle ACE and ACE Director awards from 2010–2015.

Charles Kim is an architect in Hadoop/Big Data, Linux infrastructure, cloud, virtualization, and Oracle Clustering technologies. He holds certifications in Oracle, VMware, Red Hat Linux, and Microsoft and has over 23 years of IT experience on mission- and business-critical systems. Charles presents regularly at Oracle OpenWorld, VMWorld, IOUG, and various local/regional user group conferences. He is an Oracle ACE director, VMware vExpert, Oracle Certified DBA, Certified Exadata Specialist, and a RAC Certified Expert. His books include *Oracle Database 11g: New Features for DBAs and Developers*, *Linux Recipes for Oracle DBAs*, *Oracle Data Guard 11g Handbook*, *Virtualize Oracle Business Critical Databases*, *Oracle ASM 12c Pocket Reference Guide*, and *Virtualizing Hadoop.* Charles is the current president of the Cloud Computing (and Virtualization) SIG for the Independent Oracle User Group and blogs regularly at DBAExpert.com/blog.

Nitin Vengurlekar is the cofounder and CTO of Viscosity North America where he is responsible for partner relationships and end-to-end solution deployment. Prior to joining Viscosity, he worked for Oracle for more than 17 years, mostly in the RAC engineering group and in RAC product management. He spent his last three years at Oracle as database cloud architect/evangelist in Oracle's Cloud Strategy Group in charge of private database cloud messaging. Nitin is a well-known speaker in the areas of Oracle storage, high availability, Oracle RAC, and private database cloud. He has written or contributed to *Database Cloud Storage*, *Oracle Automatic Storage Management*, and *Oracle Data Guard 11g Handbook,* and has written many papers and contributed to Oracle documentation as well as Oracle educational material. With more than 28 years of IT experience, Nitin is a seasoned systems architect who has successfully assisted numerous customers in deploying highly available Oracle systems.

Sridhar Avantsa started his career with Oracle in 1991 as a developer. Over the years he progressed to become a DBA and an architect. Currently he runs the National Oracle Database Infrastructure Consulting Practice for Rolta AdvizeX (formerly known as TUSC), which he joined in 2006 as a technical management consultant. His specific areas of interest and expertise include infrastructure architecture, database performance tuning, high availability/disaster recovery and business continuity planning, Oracle RAC and Clustering, and the Oracle engineering systems. Sridhar has been an active member of the Oracle community as a presenter and as a member of Oracle Expert Panels at conferences.

Guy Harrison is an Oracle ACE and Executive Director of research and development at Dell Software. He is the author of *Oracle Performance Survival Guide* and (with Steven Feuerstein) *MySQL Stored Procedure Programming* as well as other books, articles, and presentations on database technology. He also writes a monthly column for *Database Trends and Applications* (www.dbta.com).

Syed Jaffar Hussain is an Oracle Database expert with more than 20 years of IT experience. In the past 15 years he has been involved with several local and large-scale international banks where he implemented and managed highly complex cluster and non-cluster environments with hundreds of business-critical databases. Oracle awarded him the prestigious "Best DBA of the Year" and Oracle ACE director status in 2011. He also acquired industry-best Oracle credentials, Oracle Certified Master (OCM), Oracle RAC Expert, OCP DBA 8*i*,9*i*,10*g*, and 11*g* in addition to ITIL expertise. Syed is an active Oracle speaker who regularly presents technical sessions and webinars at many Oracle events. You can visit his technical blog at http://jaffardba.blogspot.com. In addition to being part of the core technical review committee for Oracle technology oriented books, he also coauthored *Oracle 11g R1/R2 Real Application Clusters Essentials* and *Oracle Expert RAC*.

This page intentionally left blank

About the Technical Reviewers and Contributors

Dr. Bert Scalzo is a world-renowned database expert, Oracle ACE, author, Chief Architect at HGST, and formerly a member of Dell Software's TOAD dev team. With three decades of Oracle database experience to draw on, Bert's webcasts garner high attendance and participation rates. His work history includes time at both Oracle Education and Oracle Consulting. Bert holds several Oracle Masters certifications and has an extensive academic background that includes a BS, MS, and Ph.D. in computer science, as well as an MBA, and insurance industry designations.

Javid Ur Rahaman has more than 15 years of experience with various Oracle technologies working in the APAC, USA, and African regions. He currently works as a Practice Lead–Oracle Managed Services at Rapidflow Apps Inc., a California-based VCP Specialized Oracle Partner. He contributes to various seminars on Oracle technologies at different forums. Javid's areas of focus include large-scale national and international implementations of Oracle Exadata, Exalogic, Exalyitcs, ODA, RAC, OBIEE, SOA, OTM, Web Center, Oracle Demantra, Cloud Integration with EBS, HCM Cloud Implementation, and EBS among other Oracle technologies. Javid can be followed on his blog http://oraclesynapse.com, on Twitter @jrahaman7.

Nabil Nawaz started his career with Oracle in 1997 and currently works as a senior consultant at Viscosity North America. He has more than 17 years' experience working as an Oracle DBA starting with version 7.1.6; he is an OCP, is Exadata certified; and is also an Oracle ACE associate. His background is quite vast with Oracle and he has had the opportunity to work as a consultant in many large Fortune 500 companies focusing on architecting high availability solutions such as RAC, Dataguard, and most recently Oracle Exadata and Virtualization technologies. Nabil is a native of Dallas, Texas, and resides there with his wife Rabia and three children. Nabil regularly speaks at Oracle Users groups and can be followed at his blog http://nnawaz.blogspot.com/ and on Twitter @Nabil_Nawaz.

Sandesh Rao is an Oracle technologist and evangelist, solving critical customer escalations, and has been part of the Oracle RAC Development organization for the past 8 years, developing several products to help Oracle customers. Sandesh regularly speaks at Oracle OpenWorld, COLLABORATE, and webinars/seminars as part of the Oracle RAC Special Interest Group on products related to Oracle high availability like RAC, ASM, and Grid Infrastructure. Involved with working on Engineered systems and best practices for the same through tools like exachk, he is also responsible for Diagnosability across the Oracle Database product stack and development of tools in this space. Sandesh has been working with customers, architecting, and designing solutions and solving critical problems and leading four different teams across different products like Database, Enterprise Manager, Middleware, and now the Cloud space for customer private and public clouds. Learn more about Sandesh at http://goo.gl/t6XVAQ.

This page intentionally left blank

3

The Secret Sauce: Exadata Storage Cells

The core focus of this chapter is to examine the **Exadata Storage Server**, popularly known as **Exadata Cell**, in more detail. Storage Cells are arguably the "secret sauce" or critical element that makes Exadata scale and perform so well. While the previous chapters explained the features and advantages of Exadata in 360-degree view, in this chapter we'll complete the picture by focusing on the highly specialized storage.

DBAs have almost universally acknowledged that historically the Achilles' heel for database performance was disk I/O. The faster the database and/or the more spindles handling database requests, the better the performance. Of course, newer storage technologies such as Flash disk or SSD (solid-state disk) don't fit very nicely into that generalization; however, they are still new enough and remain relatively expensive such that the bulk of storage remains on magnetic disk. Hence I/O performance remains an issue that Exadata storage or cell server attempts to address. This chapter focuses more on the abstract concepts of *what* the cell storages are and *how* they operate in the setup.

An Overview of Exadata Storage Server

An Exadata Database Machine is typically shipped with three preconfigured hardware components: Compute (database) Nodes, cell storage, and ultrafast InfiniBand storage network. The Exadata Storage Server is not simply another storage server.

It is capable of delivering unique features and provides more functionality than any third-party traditional storage server. It plays a significant role in the Exadata Database Machine by provisioning storage capacity and the very unique Exadata features.

Exadata Storage Server is not just another typical storage area network (SAN) storage device or black box that facilitates and fulfills the storage requirements. It has the intelligent Exadata Storage Software that provides the capabilities of Cell Offload processing (Smart Scan), Storage Indexes, Hybrid Columnar Compression (HCC or EHCC), I/O Resource Management (IORM), Smart Flash Cache, fast file creation, and so on. The Exadata software features are covered in detail in other chapters.

In general, each Exadata machine comes in various configurations. The number of database and cell servers purely depends on the Exadata rack capacity you choose. It comes in Eighth, Quarter, Half, and Full Racks. Depending on your choice of configuration, the cell server range can be three, seven, or 14. You have the flexibility to choose the size that best suits your needs, and you can scale up as the demand rises in the future. Figure 3.1 represents the cell server count and capacity of different Exadata racks:

- A Quarter Rack comes with two Compute (DB) Nodes and three storage servers.
- A Half Rack comes with four Compute (DB) Nodes and seven storage servers.
- A Full Rack comes with eight Compute (DB) Nodes and 14 storage servers.

Figure 3.1 The storage server count is based on rack size.

For instance, an Exadata X4 Storage Server has the following software and hardware capacity:

- A mandatory preconfigured Oracle Enterprise Linux (OEL) operating system
- Three default system user configurations: root, celladmin, and cellmonitor
- Intelligent Exadata Storage Server Software
- Two six-core Intel Xeon E5-2630 v2 processors (2.6GHz)
- 12 x 1.2TB SAS disks with High Performance (HP) and 10,000 RPM or 12 x 4TB disks with High Capacity (HC) and 7200 RPM
- 4 x 800GB Sun Flash Accelerator 480 PCI cards
- Dual-port, 2 x InfiniBand, and 4 x QDR (40GB) active-active connectivity
- 96GB memory per server
- 3.2TB Exadata Smart Flash Cache
- CELLSRV, Restart Server (RS), and Management Server (MS) background services

Each Exadata Storage Server is managed and treated individually. The Cell Control Command-Line Interface (CellCLI) utility is used to administrate the local cell, and the Distributed Command-Line Interface (dcli) utility is used to administrate the remote Exadata cell operations. The CellCLI is used to perform most of the administration and management tasks on a local cell server. The dcli utility (noninteractive) has the capability to centralize cell management across all cell servers on an Exadata machine.

Storage Server Architecture

In addition to any traditional storage server components like CPU, memory, network interface controllers (NICs), storage disks, and so on, an Exadata Storage Cell comes preloaded with the OEL operating system and intelligent Exadata Storage Server Software.

Figure 3.2 depicts the typical Quarter Rack Exadata Storage Cell architecture details, two Compute Nodes and three Storage Cells, and how the communication and relation between a Compute Node and Storage Cells are established.

An ASM instance running on the Compute (database) Node communicates with a storage server through an InfiniBand network connection using the special Intelligent Database (iDB) protocol. Additionally, the iDB protocol provides aggregation and failover to the interconnect network bandwidth. An Eighth or Quarter Rack

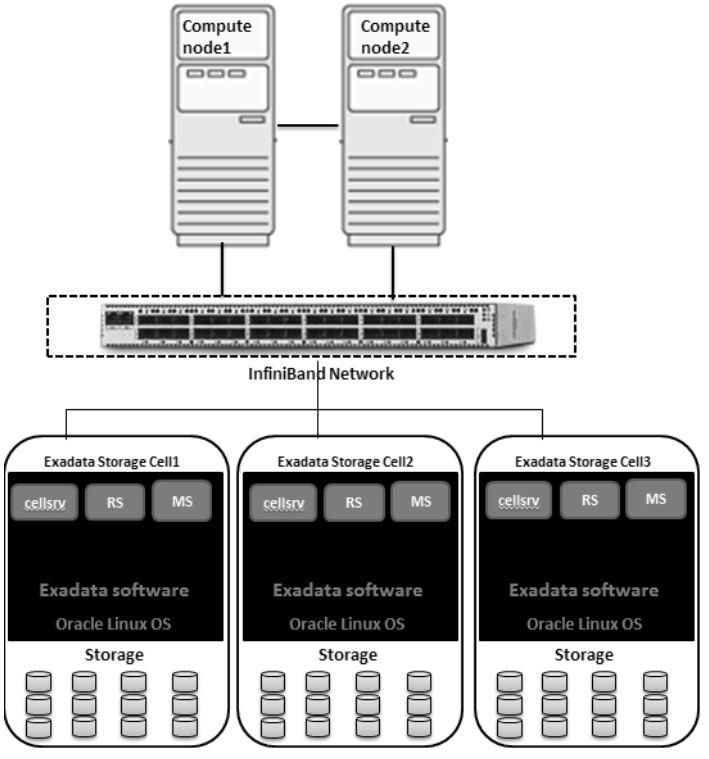

Figure 3.2 Eighth/Quarter Rack Exadata Database Machine compute and cell server architecture

comes with two InfiniBand network switches, known as leaf switches, configured between a cell and Compute Nodes to provide a communication path tolerating any switch failure. A third switch (spine) is provided only in Half and Full Rack capacity.

Each Exadata Storage Server comes with a fixed 12 uniform High Performance (HP) or High Capacity (HC) physical disks, preconfigured OEL operating system, Exadata Storage Software, and three key background services. The storage server can be accessed with three options: local login, secure shell (SSH), and KVM switch.

Cell Software Components and Management

Three key software components that run in the background are responsible for delivering the core functionality of the cell server: Cell Server (CELLSRV), Management Server (MS), and Restart Server (RS).

Cell Server

Cell Server (CELLSRV) is a multithreaded process. Arguably the heaviest among the three processes, it uses the most CPU cycles, and it also uses the special iDB protocol over InfiniBand (Oracle data transfer protocol) for communication between an ASM instance and Storage Cells. It is the primary component running on the cell and is responsible for performing Exadata advanced responsibilities, such as SQL Offloading (Smart Scans), prioritizing and scheduling an I/O on the underlying disks, implementing IORM, and so on.

It is recommended that you set the high limits of soft and hard values for the celladmin user to avoid as few ORA-600 errors as possible. As part of the disk management, when the CELLSRV process discovers that a particular disk is performing poorly, it will notify an ASM instance immediately to take the grid disk offline. Each time a database is started, it gets registered with the cell service on the cell server, and the limit of database connection to each cell service is up to 255.

The following query helps you identify the CELLSRV hang incidents on the cell:

```
# CellCLI> list alerthistory where alertMessage like ".*CELLSRV hang.*" detail
```

To diagnose CELLSRV issues, such as when CELLSRV is hung, consuming a significant amount of CPU and memory, memory leaks, and so on, you can generate a state dump of the CELLSRV process with the following command to troubleshoot the issue:

```
# CellCLI> alter cell events = "immediate cellsrv.cellsrv_statedump(0,0)"
# CellCLI> alter cell events = "immediate cellsrv.cellsrv_statedump(2,0)"
```

The following output is generated upon execution of the command, which can be referred to for further analysis of the current CELLSRV situation:

```
Dump sequence #1 has been written to /opt/oracle/cell11.2.3.3.0_LINUX.X64_131014.1/log/
diag/asm/cell/cell2/trace/svtrc_31243_80.trc


Cell usdwilo03 successfully altered
Cell cell2 successfully altered
```

Each time a state dump is performed, the sequence count for dump is increased. The trace file name in the preceding example is svtrc_18140_21.trc. The trace file contains detailed information about the cell, that is, cell software version, dump sequence, memory information, cell parameters, statistics, disk owner information, InfiniBand information, and so on. At any point in time, if you want to know the internal working condition of a CELLSRV process, you can generate a state dump to get the complete details.

As mentioned earlier, each cell is managed individually with the CellCLI utility. The CellCLI utility provides a command-line interface to the cell management functions, such as cell initial configuration, cell disk and grid disk creation, and performance monitoring. The CellCLI utility runs on the cell and is accessible from a client computer that has network access to the Storage Cell or is directly connected to the cell. The CellCLI utility communicates with Management Server to administer the Storage Cell.

If you want to manually stop, start, or restart the CELLSRV service on the cell, use the following commands:

```
# CellCLI> alter cell shutdown services cellsrv [FORCE]
```

If you encounter any issue while shutting down the CELLSRV service, use the `FORCE` option to shut down the service forcefully.

```
# CellCLI> alter cell startup services cellsrv
```

This will start the CELLSRV service on the local cell.

```
# CellCLI> alter cell restart services cellsrv
```

This will stop/start (restart) the CELLSRV service on the local cell.

```
# CellCLI> list cell attributes cellsrvStatus detail
```

This prints the current status of the CELLSRV process on the local cell.

Management Server

Management Server (MS) provides standard cell configuration and management functionality in coordination with CellCLI. It performs the following additional tasks:

- Periodically parses the symbolic links in the /dev/disk/by-path corresponding to the FMOD Flash Disks, to verify their presence and visibility to the underlying OS.
- Tracks down the hardware-level changes on the cell server and notifies the CELLSRV through an `ioctl` system call.
- Collects, computes, and manages storage server metrics.
- Rebuilds the virtual drives when a disk is replaced.
- Typically, when a disk performs poorly, the associated grid disk and cell disk will be taken offline, and MS service will notify the CELLSRV service.

Apart from these characteristics, MS also triggers the following automated tasks every hour:

- Deletes files older than seven days from the ADR directory, $LOG_HOME, and all metric history.
- Performs alert log file auto-maintenance whenever the file size reaches 10MB in size and deletes previous copies of the alert log when they become seven days old.
- Notifies when file utilization reaches 80%.

The MS service can start-stop-restart and verify the current status with the following commands:

```
# CellCLI> alter cell shutdown services ms
```

This shuts down the MS service on the local cell.

```
# CellCLI> alter cell startup services ms
```

This starts up the MS service on the local cell.

```
# CellCLI> alter cell restart services ms
```

This stops/starts (restarts) the MS service on the local cell.

```
# CellCLI> list cell attributes msStatus detail
```

This prints the current MS service status.

Restart Server

Restart Server (RS) monitors other services on the cell server and restarts them automatically in case any service needs to be restarted. Also, it handles planned service restarts as part of any software updates. The cellrssrm is the main RS process and spans three child processes: cellrsomt, cellrsbmt, and cellesmmt.

The RS service can start-stop-restart and verify the current status with the following commands:

```
# CellCLI> alter cell shutdown services rs
# CellCLI> alter cell startup services rs
# CellCLI> alter cell restart services rs
# CellCLI> list cell attributes rsStatus detail
```

All three component services are automatically started and stopped whenever the cell server is powered off or on. However, sometimes you might need to stop the service(s) manually; for instance, to enable the write-back Flash Cache feature, you need to stop the cell service.

The `alter cell shutdown services all [FORCE]` command shuts down all services together, and the `alter cell startup services all` command starts up all services together. All grid disks and related ASM disks will become inactive and go offline respectively upon stopping either all services or just the cell server, and the communication between the cell and ASM/RDBMS instances will be disturbed.

The following commands can be used to verify the current status of all three background processes on the cell:

```
# /etc/init.d/celld status
# /etc/init.d/service cell status

    rsStatus:         running
    msStatus:         running
    cellsrvStatus:    running
```

Configuring Mail Server for Alert Notifications

After the Exadata Database Machine initial deployment, configure the SMTP server settings on each cell to receive notification whenever the storage server generates alerts and warnings. The following piece of code shows an example to configure SMTP server settings on the local cell server:

```
# CellCLI > ALTER CELL realmName=ERP_HO,-
   smtpServer= 'your_domain.com',-
   smtpFromAddr='prd.cell01@domain.com', -
   smtpPwd='password123',-
   smtpToAddr='dba_group@domain.com',-
   notificationPolicy='clear, warning, critical',-
notificationMethod='email,snmp'
```

Once the SMTP settings are configured, use the following command to validate the cell:

```
# CellCLI> ALTER CELL VALIDATE MAIL
```

Displaying Cell Server Details

The following command displays cell server comprehensive details, such as cell services status, cell name, ID, interconnect details, and so on:

```
# CellCLI> list cell detail

name:                   cel01
bbuTempThreshold:       60
bbuChargeThreshold:     800
bmcType:                IPMI
cellVersion:            OSS_11.2.3.2.1_LINUX.X64_130109
cpuCount:               24
diagHistoryDays:        7
fanCount:               12/12
fanStatus:              normal
flashCacheMode:         WriteBack
id:                     1210FMM04Y
interconnectCount:      3
interconnect1:          bondib0
iormBoost:              9.2
ipaddress1:             192.168.10.19/22
kernelVersion:          2.6.32-400.11.1.el5uek
locatorLEDStatus:       off
makeModel:              Oracle Corporation SUN FIRE X4270 M2 SERVER SAS
metricHistoryDays:      7
notificationMethod:     mail,snmp
notificationPolicy:     critical,warning,clear
offloadEfficiency:      53.7
powerCount:             2/2
powerStatus:            normal
releaseVersion:         11.2.3.2.1
upTime:                 376 days, 19:02
cellsrvStatus:          running
msStatus:               running
rsStatus:               running
```

Cell Metrics and Alert History

Cell metrics and alert history provide valuable statistics for optimizing the Exadata storage resources and components on the cell. Using the `metricdefinition`, `metriccurrent`, and `metrichistory` commands, you can display the historical and current metrics of any Exadata component, such as cell disk, Flash Cache, grid disks, I/O, host, and so on:

```
CellCLI> list metricdefinition cl_cput detail
         name:                   CL_CPUT
         description:            "Percentage of time over the previous
         metricType:             Instantaneous
         objectType:             CELL
         unit:                   %

CellCLI> list metriccurrent where objecttype = 'CELL' detail
         name:                   CL_BBU_CHARGE
         alertState:             normal
         collectionTime:         2015-01-14T18:34:40+03:00
         metricObjectName:       usdwilo18
         metricType:             Instantaneous
         metricValue:            0.0 %
         objectType:             CELL

         name:                   CL_BBU_TEMP
         alertState:             normal
         collectionTime:         2015-01-14T18:34:40+03:00
```

```
         metricObjectName:       usdwilo18
         metricType:             Instantaneous
         metricValue:            0.0 C
         objectType:             CELL

         name:                   CL_CPUT_CS
         alertState:             normal
         collectionTime:         2015-01-14T18:34:40+03:00
         metricObjectName:       usdwilo18
         metricType:             Instantaneous
         metricValue:            1.6 %
         objectType:             CELL

         name:                   CL_CPUT_MS
         alertState:             normal
         collectionTime:         2015-01-14T18:34:40+03:00
         metricObjectName:       usdwilo18
         metricType:             Instantaneous
         metricValue:            0.0 %
         objectType:             CELL

CellCLI> list metriccurrent cl_cput detail
         name:                   CL_CPUT
         alertState:             normal
         collectionTime:         2015-01-14T18:34:40+03:00
         metricObjectName:       usdwilo18
         metricType:             Instantaneous
         metricValue:            2.0 %
         objectType:             CELL
```

Querying Cell Alert History

Best practices suggest periodically querying the alert history. The alert history notifications are categorized as Informal, Warning, or Critical. The `activerequest`, `alertdefinition`, and `alerthistory` commands display current and historical alert details. In order to display the alert history that occurred on the cell or a particular component, use one of the following commands:

```
CellCLI> list alerthistory detail
         name:                   7_1
         alertDescription:       "HDD disk controller battery in learn cycle"
         alertMessage:        "The HDD disk controller battery is
                              performing a learn cycle. Battery Serial
                              Number : 591  Battery Type          : ibbu08
                              Battery Temperature   : 29 C  Full Charge
                              Capacity  : 1405 mAh  Relative Charge
                              : 100 %  Ambient Temperature   : 24 C"
         alertSequenceID:        7
         alertShortName:         Hardware
         alertType:              Stateful
         beginTime:              2014-10-17T13:51:44+03:00
         endTime:                2014-10-17T13:51:47+03:00
         examinedBy:
         metricObjectName:       Disk_Controller_Battery
         notificationState:      0
         sequenceBeginTime:      2014-10-17T13:51:44+03:00
         severity:               info
         alertAction:         "All hard disk drives may temporarily
                              enter WriteThrough caching mode as part of
```

```
                         the learn cycle. Disk write throughput might
                         be temporarily lower during this time. The
                         flash drives are not affected. The battery
                         learn cycle is a normal maintenance activity
                         that occurs quarterly and runs for
                         approximately 1 to 12 hours. Note that many
                         learn cycles do not require entering
                         WriteThrough caching mode. When the disk
                         controller cache returns to the normal
                         WriteBack caching mode, an additional
                         informational alert will be sent."

         name:                   7_2
         alertDescription:       "HDD disk controller battery back to normal"
         alertMessage:           "All disk drives are in WriteBack caching
                                 mode.  Battery Serial Number : 591  Battery
                                 Type          : ibbu08  Battery Temperature
                                 : 29 C  Full Charge Capactiy  : 1405 mAh
                                 Relative Charge       : 100 %  Ambient
                                 Temperature   : 24 C"
         alertSequenceID:        7
         alertShortName:         Hardware
         alertType:              Stateful
         beginTime:              2014-10-17T13:51:47+03:00
         endTime:                2014-10-17T13:51:47+03:00
         examinedBy:
         metricObjectName:       Disk_Controller_Battery
         notificationState:      0
         sequenceBeginTime:      2014-10-17T13:51:44+03:00
         severity:               clear
         alertAction:            Informational.


# CellCLI> list alerthistory where severity='Critical'
To view alert history of the cell categorized as 'Critical' state

# CellCLI> list alerthistory 4_1 detail
To display more details of the incident mentioned in the above example
```

Querying GV$ Views

The following Exadata-related new V$ dynamic views provide the cell and its wait events with statistical information that can be used to measure the cell state, IP address used, and so on:

- **V$CELL**—provides information about cell IP addresses mentioned in the `cellip.ora` file
- **V$CELL_STATE**—provides information about all the cells accessible from the database client
- **V$CELL_THREAD_HISTORY**—contains samples of threads in the cell collected by the cell server
- **V$CELL_REQUEST_TOTALS**—contains historical samples of requests run by the cell

Storage Architecture and Formulation

So far you have learned the fundamental concepts of cell architecture and cell management. It's time now to go for the real treat and discuss the core component of Exadata Cell, that is, the storage layer.

Before we jump in and start discussing the Exadata storage architecture and storage preparation, let's explore the basic differences between non-Exadata and Exadata environments.

A traditional Oracle Database deployment requires three major components: the Oracle Database server, the storage server, and the network layer, as shown in Figure 3.3.

In this particular setup the database is both the "engine" and the "transmission" as it both processes the raw data and delivers information to the user. Here the storage server is merely an I/O facilitator—it simply and blindly serves up requested data blocks to the database. Thus, if the database SQL optimizer decides it must perform a full table scan of a million blocks, both the network and storage server must process or handle one million blocks. Such a request could overwhelm the storage server cache, thus making it less effective for all users. Furthermore, the TCP/IP network protocol packet structure is not well optimized for such simple, massive data transfers—not even with jumbo frames enabled. The general-purpose network packets suffer other limitations, including excessive header overhead waste and processing costs. While this is the most common setup there is, it's nonetheless quite inefficient.

When it comes to Exadata, an Exadata Oracle Database deployment also contains three key hardware components: the database server, the storage server, and the network between them as shown in Figure 3.4.

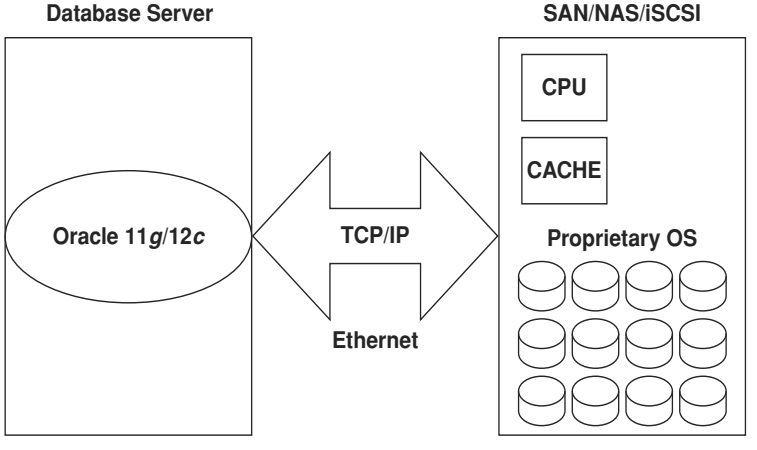

Figure 3.3 Traditional database and storage architecture

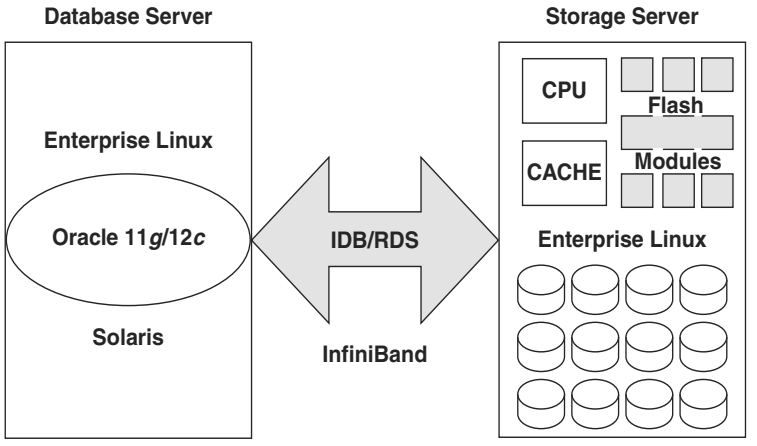

Figure 3.4 Exadata Database Machine architecture

There are four fundamental differences in the Exadata storage hardware architecture in contrast to the non-Exadata architecture, and they make all the difference in the world, especially in relation to storage scalability and performance:

- First and foremost, an Exadata cell contains **Flash Modules** that can be used either as fast disks or as additional cache (more about that later in this chapter).
- Second, the storage server is running Oracle Enterprise Linux as opposed to a proprietary OS—that's going to enable software architectural options otherwise not possible (again to be covered later in this chapter).
- Third, the high-speed, private network between the database and cell servers is based on InfiniBand rather than Ethernet.
- Fourth and finally, all communication between the database and cell servers uses the iDB protocol transmitted via Reliable Datagram Sockets (RDS).

Let's examine that last key difference in more detail since it enables or is directly responsible for some of the cell servers' "special sauce." RDS is a low-overhead, low-latency, and more CPU-efficient protocol that's been around for years (predating Exadata). So merely using the RDS protocol between database and cell servers over InfiniBand is superior to the normal deployment scenario, but while better, it's not what delivers the huge scalability and performance possible via cell servers. It's the iDB and what software architectures it makes possible that deliver most of the performance gains.

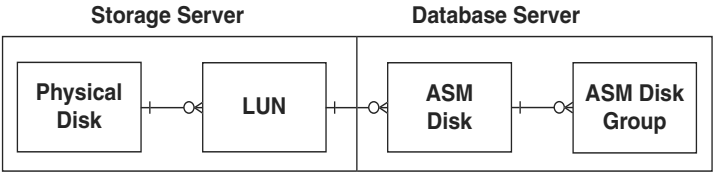

Figure 3.5 Traditional database and storage relationship

Disk Architecture in Non-Exadata

In a traditional Oracle Database deployment using Oracle's ASM, the storage server disk architecture or layout is generally organized as shown in Figure 3.5.

Typically, the physical disks (or partitions) map to logical unit numbers, or LUNs (or devices); those are then used to create Oracle ASM disks for inclusion in ASM disk groups. While ASM is an option in traditional database deployments, Oracle generally recommends it for most new databases—and especially for RAC setups. There are of course several key benefits of using ASM:

- First and foremost, as a storage mechanism it's highly integrated into the Oracle technology stack, and hence it works quite effectively and efficiently.
- Second, it eliminates the need for OS file system and logical volume managers (LVMs).
- Third, ASM offers dynamic load balancing and rebalancing of space when new disks are added or removed—something not possible with LVMs.
- Fourth and finally, it was designed from the ground up to work well with the needs and characteristics of Oracle Database I/O.

Disk Architecture in Exadata

Each Exadata Storage Server ships with 12 SAS physical disks of uniform size, either with the High Performance or the High Capacity configuration, and four Flash cards built in.

The initial two disks are mirrored using RAID (mdadm) and are used for the operating system, swap space, Exadata Storage Server software binaries, and various other Exadata configurations. The `df` command on the cell shows the following file system structure; right below the output, there is an explanation of the type of mount points and mapped file systems:

```
$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/md5              10321144   5839912   3956948  60% /
tmpfs                 49378532         0  49378532   0% /dev/shm
/dev/md7               3096272    775708   2163284  27% /opt/oracle
/dev/md4                116576     28583     81974  26% /boot
/dev/md11              5160448    205884   4692428   5% /var/log/oracle
```

- `/` is the root file system.
- `/opt/oracle` is where the Exadata software is installed.
- `/var/log/oracle` is where the cells' OS and crash logs are stored.
- The `/dev/md5` and `/dev/md6` are the system partitions, active and mirror copy.
- The `/dev/md7` and `/dev/md8` are the Exadata software installation, active and mirror copy.
- The `/dev/md11` is mapped with `/var/log/oracle`.
- At any given point in time, only four multidevice (MD) mount points can be mounted on the cell.

Approximately 29GB of space per disk is used for this purpose. In order to know whether the LUN is the system partition or not, you can use the following command:

```
CellCLI> list lun 0_0 detail
         name:                   0_0
         cellDisk:               CD_00_usdwilo18
         deviceName:             /dev/sda
         diskType:               HardDisk
         id:                     0_0
         isSystemLun:            TRUE
         lunAutoCreate:          TRUE
         lunSize:                1116.6552734375G
         lunUID:                 0_0
         physicalDrives:         20:0
         raidLevel:              0
         lunWriteCacheMode:      "WriteBack, ReadAheadNone, Direct, No
                                     Write Cache if Bad BBU"
         status:                 normal
```

There are several significant items to note here:

- First, cell servers have both Flash Cache Modules and traditional physical disks.
- Second, there's a new level of disk abstraction called the **cell disk**, which offers the ability to subdivide a LUN into partitions known as **grid disks**.
- Third, cell disks constructed from Flash Cache Modules can be further divided into Flash Cache or grid disks. Of course, physical-disk-based LUNs can map only to grid disks.
- Finally, only grid disks can be mapped to ASM disks.

At the helm of the storage layer on a cell, a physical disk is the first layer of abstraction, and each physical disk is mapped and appears as a LUN. In contrast to other storage boxes, no manual intervention is required to achieve this task as they are created automatically during Exadata Database Machine initial deployment.

The next setup is to configure the cell disk from the existing LUNs. A cell disk is created based on the existing LUN on the cell server. Once the cell disk is created, the disk can be subdivided into one or more grid disks to make them available for an ASM instance as ASM candidate disks.

As standard practice, when a cell disk is subdivided into multiple grid disks, you can then assign different performance characteristics to each grid disk according to business needs. For instance, you can assign a grid disk from a cell disk at the outermost track of a physical disk to gain the highest level of performance, and another grid disk can be assigned to the inner track of a physical disk to achieve moderate performance. The higher-performance grid disks across all cell servers then can be put together into a single disk group to place any hot data, whereas the lower-performance disks can be assembled into a disk group to store archive logs. For example, the higher-performance grid disks can be used for a data disk group and the lower-performance disks can be used to keep archive logs.

In an Oracle Exadata database deployment the cell servers can only use Oracle's ASM; however, the cell server disk architecture or layout is a little more complex with some rather different and unique options for organization. Figure 3.6 represents the relationship between disk storage and its entities.

The major difference between Flash Cache and Flash-based grid disks is quite simple. Flash Cache is autopilot caching of recently accessed database objects.

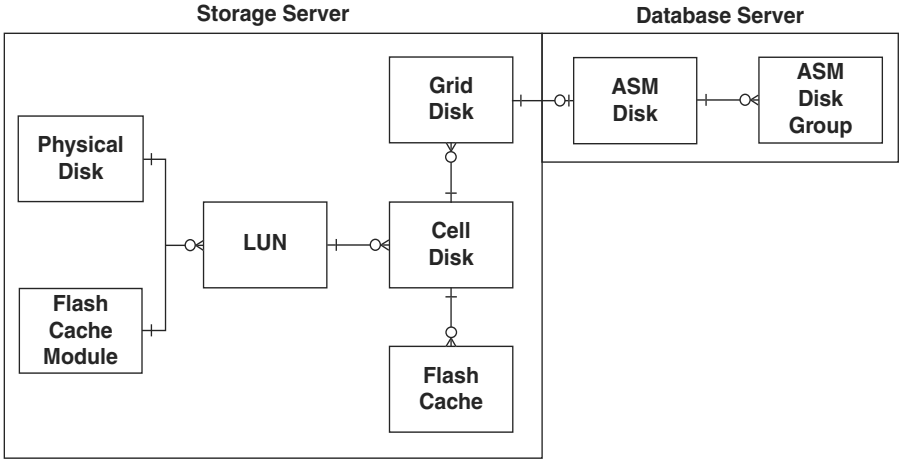

Figure 3.6 Exadata storage (physical, LUN, cell disk, grid disk, and ASM disks) formation flow

Think of it as a supersize System Global Area (SGA) at the storage level. A very similar concept known as **Smart Response Technology (SRT)** exists on some newer Intel CPUs and their chip sets, whereby an SSD can be used as front-end caching of a traditional disk drive. Flash Cache does offer the ability to manually pin database objects into it (much like pinning objects into the SGA). Here's an example of pinning the PARTS table into the Flash Cache:

```
SQL> ALTER TABLE PARTS STORAGE (CELL_FLASH_CACHE KEEP);
```

Flash grid disks, on the other hand, are simply Flash Modules organized into persistent disks for ASM use. In many ways it's like having a fast SSD disk instead of a magnetic disk on your PC. At times there will be database objects that you know will perform better if they are truly Flash based rather than contained on traditional disks (and possibly Flash cached). Hence, there are times when you'll want to create ASM disks and disk groups from Flash Modules to gain the full benefits of that speed. So for those cases you'll want to create cell and grid disks from Flash Cache Modules. The commands for doing so are covered later in this chapter.

System Users for Cell Administration

As mentioned in the beginning of the chapter, each Exadata Storage Server is typically configured with three default users with different roles. Here are the differences between the users and their capabilities:

- **root**—superuser privileges. Used to shut down and start up the storage server.
- **celladmin**—used to perform cell-level administrative tasks such as `CREATE`, `ALTER`, `MODIFY` cell objects, such as cell disks, grid disk, configure notification, and so on, using the CellCLI and dcli utilities
- **cellmonitor**—a monitoring user used to perform cell monitoring tasks. Unlike root and celladmin, it can't be used to `CREATE`, `ALTER`, or `MODIFY` any cell objects.

Following are a few practical examples.

Listing Disk Levels

To list all levels of disks, including physical disks, LUNs, cell disks, and grid disks, use the following commands:

```
Some CellCLI commands
If you want to list all the commands associated with CellCLI utility, use
the following command:
CellCLI> help

 HELP [topic]
   Available Topics:
        ALTER
        ALTER ALERTHISTORY
        ALTER CELL
        ALTER CELLDISK
        ALTER FLASHCACHE
        ALTER GRIDDISK
        ALTER IBPORT
        ALTER IORMPLAN
        ALTER LUN
        ALTER PHYSICALDISK
        ALTER QUARANTINE
        ALTER THRESHOLD
        ASSIGN KEY
        CALIBRATE
        CREATE
        CREATE CELL
```

To list the Flash Cache disks configured on the local cell, run the following command:

```
CellCLI> list lun where disktype = 'flashdisk'
         1_0     1_0     normal
         1_1     1_1     normal
         1_2     1_2     normal
         1_3     1_3     normal
         2_0     2_0     normal
         2_1     2_1     normal
         2_2     2_2     normal
```

To list the LUN details, such as to determine if the LUN is a system LUN or not, LUN size, ID, RAID level, device name, and other information on the local node, execute the following command:

```
CellCLI> list lun detail
         name:                   0_0
         cellDisk:               CD_00_usdwilo18
         deviceName:             /dev/sda
         diskType:               HardDisk
         id:                     0_0
         isSystemLun:            TRUE
         lunAutoCreate:          TRUE
         lunSize:                1116.6552734375G
         lunUID:                 0_0
         physicalDrives:         20:0
         raidLevel:              0
         lunWriteCacheMode:      "WriteBack, ReadAheadNone, Direct, No
                                       Write Cache if Bad BBU"
         status:                 normal

         name:                   0_1
         cellDisk:               CD_01_usdwilo18
         deviceName:             /dev/sdb
```

```
         diskType:               HardDisk
         id:                     0_1
         isSystemLun:            TRUE
         lunAutoCreate:          TRUE
         lunSize:                1116.6552734375G
         lunUID:                 0_1
         physicalDrives:         20:1
         raidLevel:              0
         lunWriteCacheMode:      "WriteBack, ReadAheadNone, Direct, No
                                       Write Cache if Bad BBU"
         status:                 normal
```

To list the physical disk details, such as disk name, status, and so forth, on the local cell, run the following command:

```
CellCLI> list physicaldisk detail
         name:                   20:0
         deviceId:               8
         diskType:               HardDisk
         enclosureDeviceId:      20
         errMediaCount:          0
         errOtherCount:          0
         luns:                   0_0
         makeModel:              "HGST    H101212SESUN1.2T"
         physicalFirmware:       A690
         physicalInsertTime:     2014-05-21T04:24:40+03:00
         physicalInterface:      sas
         physicalSerial:         DEAT5F
         physicalSize:           1117.8140487670898G
         slotNumber:             0
         status:                 normal

         name:                   20:1
         deviceId:               9
         diskType:               HardDisk
         enclosureDeviceId:      20
         errMediaCount:          0
         errOtherCount:          0
         luns:                   0_1
         makeModel:              "HGST    H101212SESUN1.2T"
         physicalFirmware:       A690
         physicalInsertTime:     2014-05-21T04:24:40+03:00
         physicalInterface:      sas
         physicalSerial:         DE7ZWF
         physicalSize:           1117.8140487670898G
         slotNumber:             1
         status:                 normal
```

To list the cell disk details, such as device name, creation time, size, and so on, run the following command:

```
CellCLI> list celldisk detail
         name:                   CD_00_usdwilo18
         comment:
         creationTime:           2014-09-24T16:14:52+03:00
         deviceName:             /dev/sda
         devicePartition:        /dev/sda3
         diskType:               HardDisk
         errorCount:             0
```

```
         freeSpace:              0
         id:                     ac757133-886d-465c-b449-8fe35f05519c
         interleaving:           none
         lun:                    0_0
         physicalDisk:           DEAT5F
         raidLevel:              0
         size:                   1082.84375G
         status:                 normal

         name:                   CD_01_usdwilo18
         comment:
         creationTime:           2014-09-24T16:14:53+03:00
         deviceName:             /dev/sdb
         devicePartition:        /dev/sdb3
         diskType:               HardDisk
         errorCount:             0
         freeSpace:              0
         id:                     af978555-022a-4440-9c6c-2c05f776b6cc
         interleaving:           none
         lun:                    0_1
         physicalDisk:           DE7ZWF
         raidLevel:              0
         size:                   1082.84375G
         status:                 normal
```

To list the grid disk details, such as cell disks mapped to the physical disk, size, status, and so on, run the following command:

```
CellCLI> list griddisk detail
         name:                   DG_DBFS_CD_02_usdwilo18
         asmDiskgroupName:       DG_DBFS
         asmDiskName:            DG_DBFS_CD_02_USDWILO18
         asmFailGroupName:       USDWILO18
         availableTo:
         cachingPolicy:          default
         cellDisk:               CD_02_usdwilo18
         comment:
         creationTime:           2014-09-24T16:19:02+03:00
         diskType:               HardDisk
         errorCount:             0
         id:                     7e2d7848-cf81-4918-bb01-d27ef3da3950
         offset:                 1082.84375G
         size:                   33.796875G
         status:                 active

         name:                   DG_DBFS_CD_03_usdwilo18
         asmDiskgroupName:       DG_DBFS
         asmDiskName:            DG_DBFS_CD_03_USDWILO18
         asmFailGroupName:       USDWILO18
         availableTo:
         cachingPolicy:          default
         cellDisk:               CD_03_usdwilo18
         comment:
         creationTime:           2014-09-24T16:19:02+03:00
         diskType:               HardDisk
         errorCount:             0
         id:                     972be19d-5614-4b98-8806-7bdc2faf7630
         offset:                 1082.84375G
         size:                   33.796875G
         status:                 active
```

Configuring Cell Disks

The following command will configure 12 cell disks, one for each LUN, with the default naming convention. This is usually run as part of the initial deployment.

```
# CellCLI> CREATE CELLDISK ALL HARDDISK
```

Alternatively, use the following command to create the cell disks to enable interleaving:

```
# CellCLI> CREATE CELLDISK ALL HARDDISK INTERLEAVING='normal_redundancy'
```

Creating Grid Disks

The following command will create a grid disk at the outermost track layer of a physical disk for high performance:

```
# CellCLI> create griddisk ALL HARDDISK prefix=data, size 500G
```

The next command will create a grid disk at the inner track layer of a physical disk for less I/O-intensive applications:

```
# CellCLI> CREATE GRIDDISK ALL PREFIX=FRA
```

Configuring Flash Grid Disks

The following procedure is used to drop the current Flash Cache and rebuild with the nondefault size:

```
# CellCLI> DROP FLASHCACHE
# CellCLI> CREATE FLASHCACHE ALL SIZE =200G
# CellCLI> CREATE GRIDDISK ALL FLASHDISK
```

Once the Exadata storage configuration is done, the next step is to configure the database hosts to access the grid disks. The `cellinit.ora` and `cellip.ora` files must be configured at the Compute Nodes in order to access the grid disk from the cell. The following example shows the contents of each file:

```
#/etc/oracle/cell/network-config/cellinit.ora
Ipaddress=192.168.0.13/24
```

The `cellinit.ora` file contains the database server IP address. Each database server will have its own IP address recorded in the `cellinit.ora` file:

```
/etc/oracle/cell/network-config/cellip.ora
cell="192.168.0.11"
cell="192.168.0.12"
cell="192.168.0.13"
```

The `cellip.ora` file contains the IP addresses of all cells, and all Compute Nodes should have the same entries in order to access storage on the cell servers.

Creating an ASM Disk Group

To show how to create an ASM disk group for your database, let's take the grid disks from cell01 and cell02 to create a data disk group with high-redundancy capabilities:

```
SQL> CREATE DISKGROUP DG_DATA HIGH REDUNDANCY DISK 'o/*/
 DATA_EX01_CD_00_ex01cel01', 'o/*/ DATA_EX01_CD_01_ex01cel01'
, 'o/*/DATA_EX01_CD_02_ex01cel01', 'o/*/ DATA_EX01_CD_00_ex01cel02'
, 'o/*/DATA_EX01_CD_01_ex01cel02', 'o/*/DATA_EX01_CD_02_ex01cel02'
'compatible.asm'='11.2.0.3',     'compatinle.rdbms'='11.2.0.2',
'cell_smart_scan'='TRUE';
```

Managing the Cell Server

Sometimes it becomes necessary, especially before and after patch deployment on the cell as well as on the Compute Nodes, to know the current cell software version and the previous version to which the cell can potentially roll back. In this context, Oracle provides two utilities in /usr/local/bin: `imageinfo` and `imagehistory`.

When the `imageinfo` utility is executed as the root user on the cell server as follows, it will help you get the active cell software details, such as cell kernel version, OS version, active cell image details, cell boot partitions, and so on:

```
# imageinfo

Kernel version: 2.6.32-400.11.1.el5uek #1 SMP Thu Nov 22 03:29:09 PST 2012 x86_64
Cell version: OSS_11.2.3.2.1_LINUX.X64_130109
Cell rpm version: cell-11.2.3.2.1_LINUX.X64_130109-1

Active image version: 11.2.3.2.1.130109
Active image activated: 2013-01-30 19:14:40 +0300
Active image status: success
Active system partition on device: /dev/md5
Active software partition on device: /dev/md7

In partition rollback: Impossible
```

```
Cell boot usb partition: /dev/sdm1
Cell boot usb version: 11.2.3.2.1.130109

Inactive image version: 11.2.3.2.0.120713
Inactive image activated: 2012-12-10 11:59:57 +0300
Inactive image status: success
Inactive system partition on device: /dev/md6
Inactive software partition on device: /dev/md8

Boot area has rollback archive for the version: 11.2.3.2.0.120713
Rollback to the inactive partitions: Possible
```

The `imagehistory` utility helps you get all the previous software versions installed on the particular cell:

```
#imagehistory

Version                             : 11.2.3.1.1.120607
Image activation date               : 2012-07-25 01:25:34 +0300
Imaging mode                        : fresh
Imaging status                      : success

Version                             : 11.2.3.2.0.120713
Image activation date               : 2012-12-10 11:59:57 +0300
Imaging mode                        : out of partition upgrade
Imaging status                      : success

Version                             : 11.2.3.2.1.130109
Image activation date               : 2013-01-30 19:14:40 +0300
Imaging mode                        : out of partition upgrade
Imaging status                      : success
```

The `-h` option can be used to list all parameters that are associated with the `imageinfo` and `imagehistory` utilities.

To remove the old `alerthistory` on the cell, you can use the following commands:

```
#CellCLI> drop alerthistory all   -- will drop the complete alerthistory info
#CellCLI> drop alerthistory <9_1> -- will drop a particular incident history
```

Troubleshooting the Cell Server

The following sections discuss and demonstrate some of very important tools and utilities provided on Exadata to collect the diagnostic information on a cell. Most of the diagnostic tools and utilities reside under the /opt/oracle.SupportTool folder.

SunDiag

The `sundiag.sh` diagnostic collection script exists on each Compute Node and Storage Cell under /opt/oracle.SupportTools. The script can also be downloaded

from support.oracle.com. The script helps you gather the required diagnostic information related to problematic disks or any other hardware issues on the cell.

You have to execute the script as root user, as follows:

```
/opt/oracle.SupportTools/sundiag.sh
```

If you would like to gather similar diagnostic information across all cell servers, you will have to execute the script through the dcli utility.

This script generates a timestamped .tar file under /tmp/sundiag_Filesystem which can be uploaded to Oracle Support for analysis.

ExaWatcher

The new ExaWatcher utility located under /opt/oracle.ExaWatcher replaces the traditional OSWatcher utility in Exadata Storage Software 11.2.3.3 and is used for system data collection. The utility is up and running upon system reboot. It collects the statistics for the following components on the cell and keeps the log files under /opt/oracle.ExaWatcher/archive:

- `Diskinfo`
- `IBCardino`
- `Iostat`
- `Netstat`
- `Ps`
- `Top`
- `Vmstat`

In order to produce or extract the reports from the logs generated by the ExaWatcher utility, you will have to use the `GetExaWatcherResults.sh` script. You can collect input at various levels:

- `FromTime` until `ToTime` extracts range reports.
- `ToTime` extracts on or before time reports.
- `AtTime` extracts around the time reports.
- `Hours` extracts time in range reports.

Following are some examples:

```
# ./GetExaWatcherResults.sh --from <time frame> to <time frame>
# ./GetExaWatcherResults.sh --at <time frame> --range 2
```

The second example extracts 2 hours starting with the time defined with the `at` parameter.

The `ExaWatcherCleanup` module is used to automatically manage the file system space used by ExaWatcher. Based on the limits set for space management, the module is responsible for cleaning up the old log files by removing them.

To get more help on how to use the utility, use the following commands:

```
# ExaWatcher.sh --help
# ExaWatcherResults.sh --help
```

Exachk

Exachk is an Oracle Exadata diagnostic tool that comes with different levels of verification and collects hardware, software, firmware, and configuration data on Exadata systems. It is strongly recommended that you include this script as part of your periodic maintenance operation tasks. Also, run the script before any migration, upgrade, or any other major change operations take place.

This script doesn't come with the Exadata machine; you will have to download it (exachk_225_bundle.zip file) from Oracle Support Note ID 1070954.1, which requires login credentials. The Note and the .zip files contain all the information required to use the tool.

CheckHWnFWProfile

The CheckHWnFWProfile utility verifies any hardware and firmware component details and reports if there are any recommended items missing. It is also used to validate the current configuration on the servers. This utility is used without passing any parameters, as shown in the following example:

```
# /opt/oracle.cellos/CheckHWnFWProfile
```

If the current hardware and firmware are to the correct version, it will give the `SUCCESS` output.

To obtain more information on the utilization of this utility use the `-d` option with the command.

Storage Cell Startup and Shutdown

When rebooting or shutting down the Exadata Storage Cell for maintenance or any other valid reason, ensure that you adhere to proper stop/start cell procedure

to guarantee a graceful cell shutdown. This section emphasizes the significance of complying with an appropriate cell stop and start procedure and demonstrates the steps in depth.

One of the key responsibilities of a DMA includes graceful shutdown of the cell, be it for a quick reboot or for maintenance. The shutdown shouldn't impact the underlying ASM instance and the active database that is running. That being said, shutting down the cell and its services without affecting the ASM availability largely depends on the current ASM redundancy level. Under any circumstances, it is a best practice to follow this procedure for the graceful shutdown of a cell:

1. Verify that ASM doesn't have any impact by taking the grid disks offline on the cell. Use the following command to verify the result:

```
# CellCLI> list griddisk attributes name,asmdeactivationoutcome
```

2. If the result of `asmdeactivationoutcome` is yes for all the grid disks listed, it is an indication that ASM will not have any impact and it is safe to deactivate all the grid disks, using the next command:

```
# CellCLI> alter griddisk all inactive
```

3. Once you turn off all the grid disks on the cell, run the first command to verify the `asmdeactivationoutcome` output and verify that all the grid disks are now offline, using this command:

```
# CellCLI> list griddisk
```

4. Now it is safe to power off/reboot/shut down the cell. As root, shut down the cell using the following command:

```
$ shutdown -h now  -- OS command to shut down the cell server
```

Note

If you intend to take the cell down for a very long period of time, you will have to adjust the ASM disk's `DISK_REPAIR_ATTRIBUTE` default value to prevent ASM from dropping the disks automatically upon taking them offline. The default value is set to 3.6 hours; therefore, if you are taking the cell down for 5 hours, for example, use the following command to set the value through an ASM instance:

```
SQL> ALTER DISKGROUP DG_DATA SET ATTRIBUTE 'DISK_REPAIR_TIME'='5H';
```

You will have to adjust all the required disks on the cell.

Once the cell is rebooted or comes online, follow these instructions to bring the services and grid disks back to action:

1. First step of the procedure:

```
# CellCLI > alter griddisk all active
```

2. Second step of the procedure:

```
# CellCLI> list griddisk attributes name,asmmodestatus
```

3. Last step of the procedure:

```
# CellCLI> list cell detail
```

If you have worked on Oracle Database and cluster technologies previously, you probably know that each of them maintains an alert log file where all important events and sequences are written. Similarly, an alert file is maintain by each cell server to record all the important events of the cell, such as when the services started or stopped, disk warning messages, cell and grid disk creation, and so on. It is highly recommended that you review the logs frequently. You can also refer to the OS file to find out when the cell restarted.

Following are some of the most commonly referenced log/trace files and their locations:

- /log/diag/asm/cell/{cell name}/trace
- **MS log**—/opt/oracle/cell/log/diag/asm/cell/{cell name}/trace/ms-odl.log
- **OSWatcher logs**—/opt/oracle.oswatcher/osw/archive
- **OS messages**—/var/log/messages
- **Cell-patching-related logs**—/var/log/cellos

Solving Disk Problems

Yet another major responsibility of a DMA includes determining when and how a faulty (dead, predictive failure, poor performance) disk is identified and getting it replaced on an Exadata Storage Server. Although most of the procedure is automated by Oracle, including identification and notification of an underperforming, faulty, or damaged disk, it is equally important for you to understand the factors and procedures involved in troubleshooting and replacing the disk when it becomes necessary to do so.

When a disk confronts performance issues or any sort of hard failure, an alert is generated by the MS background process on the cell server, and it notifies the CELLSRV background service about the alert. At the same time, if OEM is configured, the message is also pushed to Grid Control, through which you can receive an email or SMS message.

Initially, a set of performance tests is carried out by the MS service on the disk on which the performance degradation has been identified to determine whether the behavior is a temporary glitch or a permanent one. If the disk passes the tests successfully, it is brought back to the active configuration; if not, it is marked as performing poorly and an Auto Service Request (ASR), if configured, is opened for disk replacement.

Whenever a disk failure occurs or a disk goes into predictive status, ASM automatically drops the related grid disks of the failed disk either normally or forcefully. After the disk issues are addressed and the disks are ready to go active, ASM automatically brings related grid disks online as part of the Exadata auto disk management, which is controlled by the `_AUTO_MANAGE_EXADATA_DISKS` parameter.

Typically the following actions are performed when disk issues are identified on the cell server:

1. When poor performance is detected, the cell disk and physical disk statuses are changed.
2. All grid disks of the particular cell disk are taken offline.
3. The MS service notifies the CELLSRV service about the findings, and in turn, CELLSRV notifies ASM instances to take the grid disk offline.
4. The MS service on the cell then performs a set of confinement checks to determine if the disk needs to be dropped.
5. If the disk passes the performance tests, the MS service notifies the CELLSRV service to turn all the cell disks and all its grid disks online.
6. If the disk fails the performance tests, the cell disk and physical disk statuses are modified, and the disk is removed from the active configuration.
7. The MS service notifies the CELLSRV service about the disk issues. In turn, the CELLSRV service informs ASM instances to drop all the grid disks of the cell.
8. If ASR is configured, a service request is logged to Oracle Support about disk replacement.
9. You will have to either use the spare disk to replace the faulty disk or request a replacement disk from Oracle.

Disk problems can be categorized in two levels: hard failure and predictive failure. A predictive failure is when a disk is flagged as predictive or in a poor performance state. A hard failure is when a disk goes into critical state.

When you have been notified about the critical failure state of a physical disk, through either email or SMS message, your first priority is to identify the damaged physical disk's exact name, position, location, and slot number through the

cell alert history or by reviewing the cell logs. Also refer to the ASM alert logs to ensure that ASM turned the damaged disk offline (dropped the disk) and ASM rebalancing is completed before replacing the disk.

To view the disk-related alert history, use the following command on the cell:

```
CellCLI> list alerthistory

         7_1     2014-10-17T13:51:44+03:00       info    "The HDD disk controller battery
is performing a learn cycle. Battery Serial Number : 591  Battery Type          : ibbu08
Battery Temperature   : 29 C  Full Charge Capacity  : 1405 mAh  Relative Charge       :
100 %  Ambient Temperature   : 24 C"

         7_2     2014-10-17T13:51:47+03:00       clear   "All disk drives are in WriteBack
caching mode.  Battery Serial Number : 591  Battery Type          : ibbu08  Battery
Temperature   : 29 C  Full Charge Capactiy  : 1405 mAh  Relative Charge       : 100 %
Ambient Temperature   : 24 C"
```

```
# CellCLI> list physicaldisk WHERE diskType=HardDisk AND status=critical detail
# CellCLI> list physicaldisk WHERE diskType=HardDisk AND status like ".*failure.*" detail
# CellCLI> alter physicaldisk disk_name:disk_id drop for replacement
```

Verify that the grid disks of the cell disk are dropped and the rebalancing operations are completed on the ASM instances:

```
SQL> SELECT name,state from v$asm_diskgroup;
SQL> SELECT * FROM v$asm_operation;
```

Three minutes after replacing the faulty physical disk on the cell, all the grid disks and cell disks are automatically re-created, added subsequently to the respective disk group, and then rebalanced.

Enforcing Cell Security

Exadata offers many layers of security setups to meet your business needs. An Exadata cell server by default comes with open storage security, where there are no restrictions applied on accessing grid disks from ASM or database clients. Apart from open security, Oracle Exadata supports two levels of security: ASM-scoped security and database-scoped security. These control the storage access from ASM cluster or database clients.

Security can control which ASM or database clients can access a specific grid disk or pools of grid disks on the cell. With ASM-scoped security, all database clients of that particular ASM cluster can access the grid disks. You can go further and configure database-scoped security to restrict the storage access at the database level.

Even if you intend to deploy database-scoped storage-level security, you will have to first configure ASM-scoped security. The following sections describe step-by-step procedures for how to enforce ASM-scoped and database-scoped security on the cell.

Configuring ASM-Scoped Security

In order to enforce ASM-scoped security on Exadata, follow this procedure:

1. Shut down ASM and all database instances on the Compute Node.
2. Generate a security key using the `CREATE KEY` command on a cell `CellCLI` prompt which will be used or copied across all cell servers to enforce the security:

```
CellCLI> CREATE KEY
```

3. Create a cellkey.ora file for ASM under /etc/oracle/network-config on the Compute Node, assign the security key against the ASM instance name, and change the permission and ownership as shown:

```
# cellkey.ora
    key=<key generated in the above command>
    asm=<asm db_unique_name>
    realm=<xyz> -- optional
# chown oracle:dba /etc/oracle/network-config/cellkey.ora
# chmod 640 /etc/oracle/network-config/cellkey.ora
```

4. If you want to change the realm name on the cell, use the following:

```
CellCLI> alter cell realmName=prod_realm
```

5. Assign the security key to the ASM instances across all the cell servers where you want to enforce the security:

```
CellCLI> ASSIGN KEY FOR '+ASM'='<security key>'
```

6. Add or modify the grid disk's `availableTo` attribute to add the ASM instance, as follows:

```
CellCLI> list griddisk
```

7. After getting the grid disks' names, change the attribute for each grid disk:

```
CellCLI> alter griddisk <grid disk list> availableTo ='+ASM'
CellCLI> alter griddisk ALL availableTo='+ASM'
```

8. Restart the ASM and database instances.

This type of security enables grid disk access to all the database clients of that ASM instance.

Configuring Database-Scoped Security

In order to enforce database-scoped security on Exadata, follow the next procedure:

1. Stop ASM and all database instances on the Compute Node.
2. Generate a separate new security key for each database connecting to the ASM instance with the `CREATE KEY` command using the `CellCLI` utility on the cell:

```
CellCLI> CREATE KEY
```

3. Create a cellkey.ora key file under $ORACLE_HOME/admin/<db_name>/pfile for each database on the Compute Node, assign the key against the database name, and change the permission and ownership as follows:

```
# cellkey.ora
    key=<key generated in the above command>
    asm=<asm db_unique_name>
    realm=<xyz> -- optional
# chmod 640 $ORACLE_HOME/admin/<db_name>/pfile/cellkey.ora
```

4. If you want to change the realm name on the cell, use the following:

```
CellCLI> alter cell realmName=xyz
```

5. Assign the security key to the database on all the cell servers:

```
CellCLI> ASSIGN KEY FOR <DB_NAME1>='<security key1>',
<DB_NAME2>='<security key2>'
```

6. Change the grid disk's `availableTo` attribute, as follows:

```
CellCLI> list griddisk
```

7. After getting the grid disks' names, change the attribute for each grid disk:

```
CellCLI> alter griddisk <griddisk list> availableTo ='+ASM,
<DB_NAME>,<DB_NAME2>'
CellCLI> alter griddisk ALL availableTo='+ASM,<DB_NAME>,<DB_NAME2>'
```

8. Restart the ASM and database instances.

When you want to further restrict individual databases of that ASM instance to access a different pool of grid disks, you will have to enforce database-scoped security as explained previously.

You can list or view the existing security keys using the following command:

```
# CellCLI> list key
```

Exempting Cell Security

At any given time, you can exempt the cell security that has been imposed earlier. You can bring down the database-scoped security to ASM-scoped security and ASM-scoped security to the default open security. Keep in mind that any change and enforcement require ASM and database downtime.

The following procedure can be used to remove the cell security:

1. Stop ASM and databases.
2. Get the list of grid disk attributes assigned.
3. Exempt the databases from the security and remove Access Control List (ACL) setup using the following commands:

```
CellCLI> alter griddisk griddiskName availableTo='+ASM'
CellCLI> assign key for <DB_NAME>=''
```

4. Remove the cellkey.ora file from the respective database Home location.
5. Finally, verify the grid disks to ensure that the databases are exempted from the security list:

```
CellCLI> list griddisk attributes name,availableTo
```

6. The previous steps exempt the databases from database-scoped security. If you want to also remove ASM-scoped security, first use the following commands:

```
CellCLI> alter griddisk all availableTo=''
CellCLI> alter griddisk griddiskName availableTo=''
CellCLI> assign key for +ASM=''
```

7. Finally, remove the cellkey.ora file from /etc/oracle/cell/network-config.

Summary

To manage any environment effectively, it is always important for a DBA/DMA to understand the underlying architecture and the total functionality of the system. This chapter explained all the essentials and the importance of Exadata Storage Cells in detail. You have learned how to gracefully shut down Storage Cell background services, how to configure cell security, and how to create and manage storage on the cell.

This page intentionally left blank

Index