Performance Diagnostics With AWR Reports

Richard Foote Consulting

Richard Foote

richardfoote

- Working in IT for over 30+ years , 20+ years with Oracle Database
- 19 years employed in Australian Federal Government in various IT roles
- Responsible for many large scale, mission critical, “life-dependant” classified Oracle systems, tuned numerous databases often with 10x performance improvements
- Worked for Oracle Corporation between 1996 and 2002 and between 2011 and 2017
- In September 2017, started my own independent company Richard Foote Consulting
- Oracle OakTable Member since 2002 and awarded Oracle ACE Director in 2008
- Regular speaker at user group meetings and conferences such as Oracle OpenWorld, IOUG Collaborate, Hotsos Symposium, AUSOUG InSync, ODTUG Kscope, UKOUG Tech Conference, E4 Enkitec Extreme Exadata Expo, …
- Richard Foote's Oracle Blog: https://richardfoote.wordpress.com
- Richard Foote Consulting: https//richardfooteconsulting.com
- Spend as much free time as possible listening to the music of David Bowie !!

richardfoote.wordpress.com

Oracle Indexing Internals and Best Practices 5 Day Webinars

8-12 October, 6-10 November 2018 (4 Hours Daily)

Of benefit to DBAs, Developers, Solution Architects and anyone else interested in designing, developing or maintaining high performance Oracle-based applications/databases.

Examines most available index structures/options & discusses in considerable detail how indexes function, how/when they should be used & how they should be maintained. Also how indexes are costed & evaluated by the Cost Based Optimizer (CBO) & how appropriate data management practices are vital for an effective indexing strategy. Covers many useful tips and strategies to maximise the benefits of indexes on application/database performance & scalability.

Running between 10am-2pm Zurich Time 8th – 12th October 2018

richardfooteconsulting.com/indexing-webinar/

DB Running Slow, Where To Start ??

WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
BOWIE	1234567890	BOWIE	1	25-Jun-99 12:06	12.1.0.2.0	NO

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
bowie011	Linux x86 64-bit	8	8	8	31.36

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	4724	30-Jul-99 13:00:20	278	6.7
End Snap:	4725	30-Jul-99 14:00:31	333	7.0
Elapsed:		60.17 (mins)		
DB Time:		773.53 (mins)		

Report Summary

Top ADDM Findings by Average Active Sessions

Finding Name	Avg active sessions of the task	Percent active sessions of finding	Task Name	Begin Snap Time	End Snap Time
Top SQL Statements	12.85	79.67	ADDM:1234567890_1_4725	30-Jul-99 13:00	30-Jul-99 14:00
Row Lock Waits	12.85	56.91	ADDM:1234567890_1_4725	30-Jul-99 13:00	30-Jul-99 14:00
Unusual "Network" Wait Event	12.85	6.58	ADDM:1234567890_1_4725	30-Jul-99 13:00	30-Jul-99 14:00
Hard Parse Due to Literal Usage	12.85	6.45	ADDM:1234567890_1_4725	30-Jul-99 13:00	30-Jul-99 14:00
Unusual "Network" Wait Event	12.85	3.75	ADDM:1234567890_1_4725	30-Jul-99 13:00	30-Jul-99 14:00

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	12.9	0.7	0.00	0.00
DB CPU(s):	4.0	0.2	0.00	0.00
Background CPU(s):	0.1	0.0	0.00	0.00
Redo size (bytes):	218,162.4	12,514.4		
Logical read (blocks):	245,114.3	14,060.4		
Block changes:	1,102.0	63.2		
Physical read (blocks):	24,546.8	1,408.1		
Physical write (blocks):	52.5	3.0		
Read IO requests:	1,474.2	84.6		
Write IO requests:	23.0	1.3		
Read IO (MB):	191.8	11.0		
Write IO (MB):	0.4	0.0		
IM scan rows:	0.0	0.0		
Session Logical Read IM:				
User calls:	7,235.3	415.0		
Parses (SQL):	802.8	46.1		
Hard parses (SQL):	34.1	2.0		
SQL Work Area (MB):	30.7	1.8		
Logons:	0.6	0.0		
Executes (SQL):	4,933.3	283.0		
Rollbacks:	0.3	0.0		
Transactions:	17.4			

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.21	In-memory Sort %:	100.00
Library Hit %:	97.66	Soft Parse %:	95.75
Execute to Parse %:	83.73	Latch Hit %:	99.39
Parse CPU to Parse Elapsd %:	75.58	% Non-Parse CPU:	83.72
Flash Cache Hit %:	0.00		

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
enq: TX - row lock contention	346	26.4K	76340.32	56.9	Application
DB CPU		14.5K		31.3	
SQL*Net message from dblink	17,886,859	3055.2	0.17	6.6	Network
SQL*Net more data from dblink	1,126,300	1738.4	1.54	3.7	Network
latch free	26,770	623.3	23.28	1.3	Other
direct path read	668,179	170.6	0.26	.4	User I/O
cursor: pin S wait on X	1,732	168.9	97.53	.4	Concurrency
log file sync	62,198	105.5	1.70	.2	Commit
library cache: mutex X	35,239	67.7	1.92	.1	Concurrency
db file sequential read	4,224,329	63.5	0.02	.1	User I/O

Wait Classes by Total Wait Time

Where's The Milk ?

- My wife kindly pops down to the local shop to buy some milk. A full *<u>60 minutes</u>* later, she finally returns and plops the milk on the kitchen bench.
- This "response time" is clearly unacceptable, I needed the milk within 5 minutes…
- So how do I improve the response time ?

Note: In real-life, my wife would of course say go get the milk yourself !!

Response Time

Response times can simplistically be broken up into two basic components:

Time it takes doing something

+

Time it takes waiting on something

Response Time = Doing Time + Waiting Time

To tune effectively and reduce response times, we need to focus on where most time is being spent !!

Database Time (DB Time)

- Total time in database calls by foreground sessions
- Includes CPU time, IO time and non-idle wait time

Database time is total time spent by user processes either actively working or actively waiting in a database call

✓ Always encouraged when a DBA can state typical DB Time and Active Sessions of their databases

Fundamental Concepts Summary

Database Time (DB Time) =

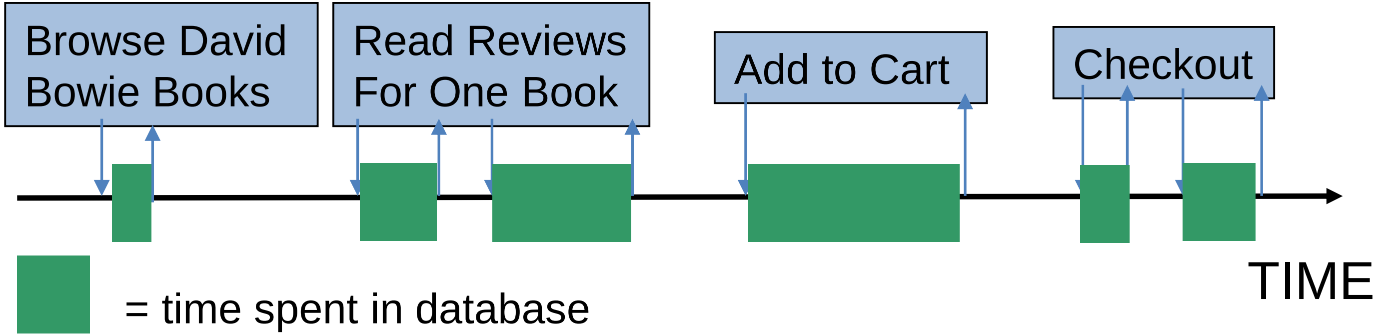
Total time foreground sessions spend in all database calls

Active Session =

Session currently spending time in a database call

Average Activity of the Session (% Activity) =

The ratio of time active to total wall clock time

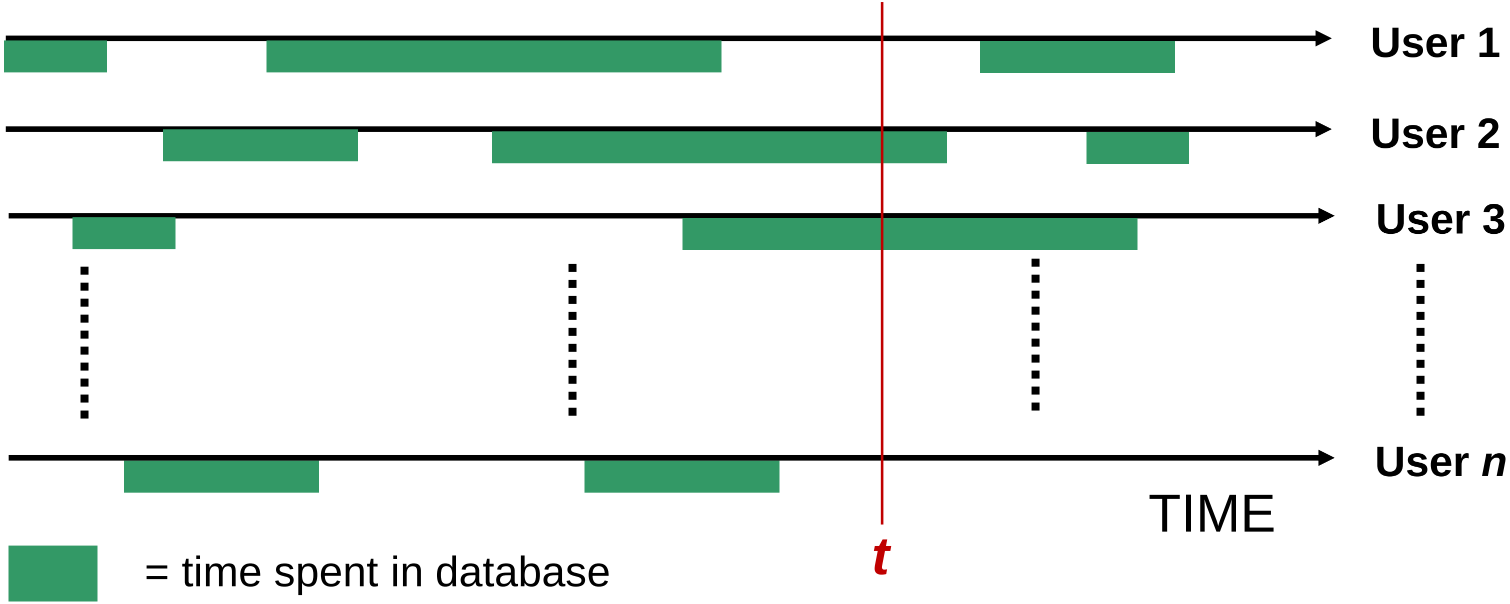


Active Sessions

DB Time = Sum of DB Time Over All Sessions

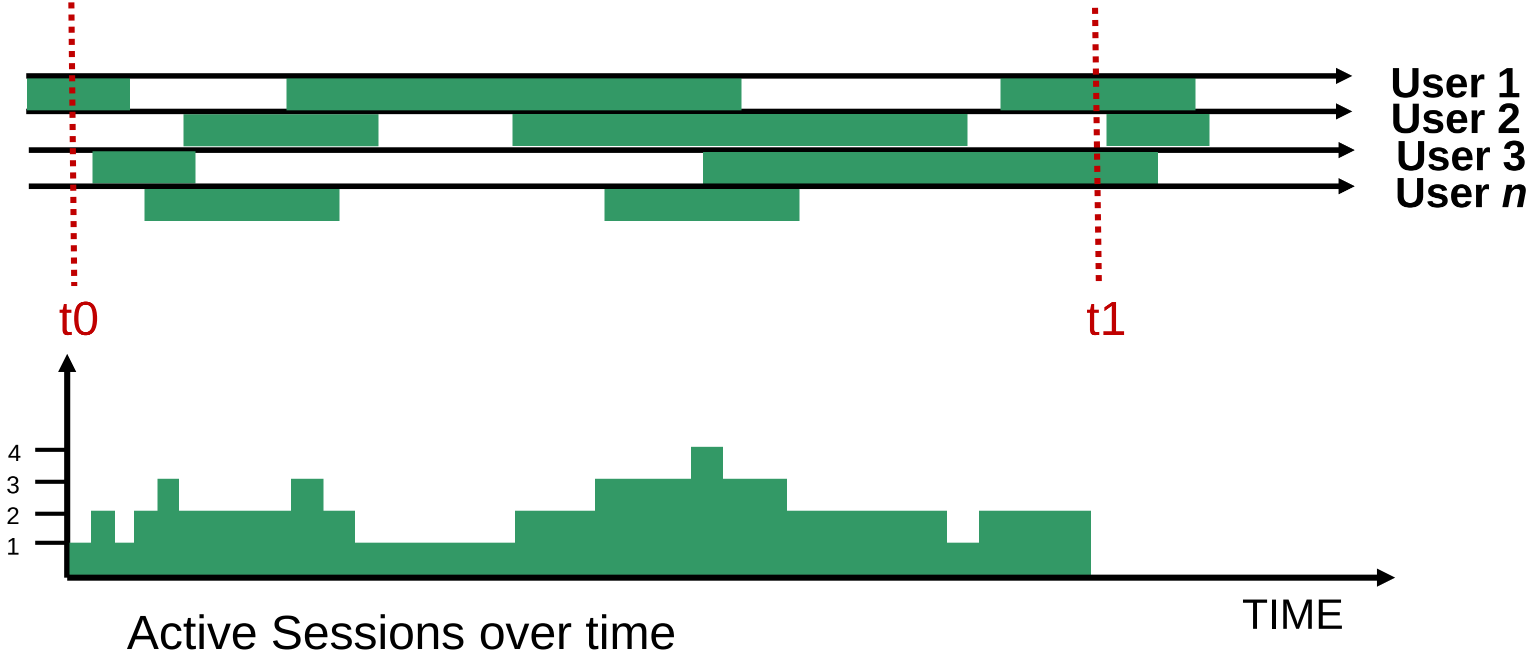
Avg. Active Sessions = Sum of Avg. Activity Over All Sessions

At time *t* we have 2 active sessions



Visualizing DB Time

$$\text{Avg. Active Sessions} = \frac{\text{Total Database Time}}{\text{Wall Clock (Elapsed) Time}}$$



Host Performance and DB Time

- Host is CPU-bound

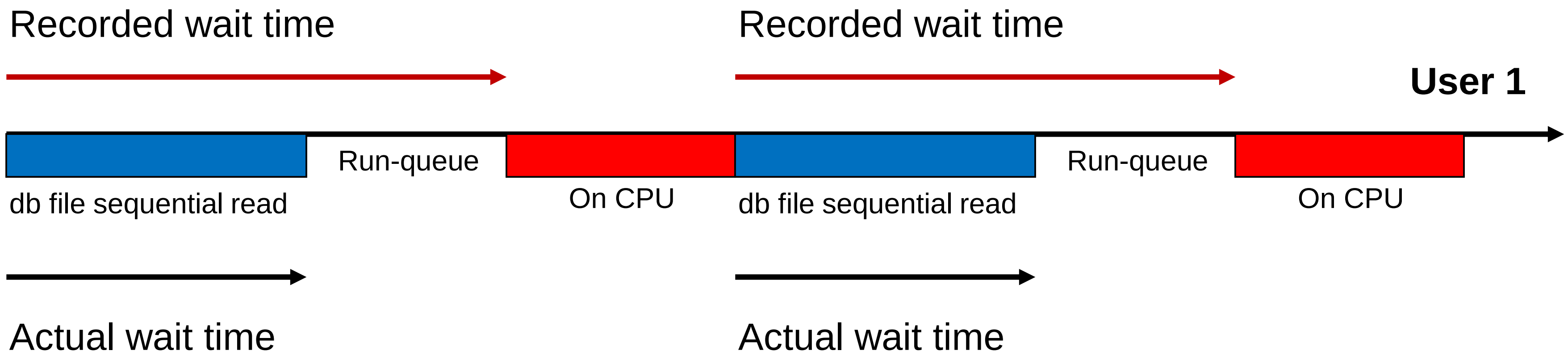
  => foregrounds accumulate active run-queue time

  => wait event times are artificially inflated

  => DB time increases

Tune for CPU before waits when CPU constrained

CPU Run-queue and DB Time



DB time can be inflated when host is CPU-bound

If Database Performance Is Poor Globally

- Everyone (or significant number) experiencing some form of performance degradation (response times have increased)
- Overall DB time can be expected to have increased
- Can potentially use session level analysis as specific sessions can be indicative of general issues
- But a system-wide view of the database can be extremely beneficial
- Enter Automatic Workload Repository (AWR) report
- Provides a detailed analysis of where/how all DB time is generated

Automatic Workload Repository (AWR)

- Built-in automatic in-memory performance statistics
- MMON background process writes performance statistics snapshots to disk every hour (default)
- Snapshots kept for 8 days (default)
- AWR Report provides a database wide report based on these metrics for given snapshot period
- Automatic Database Diagnostics Monitor (ADDM) finds top “problems”
- Requires Diagnostics Pack (or High Performance Oracle DB Cloud Service)

How To Generate AWR Report

Parameter: CONTROL_MANAGEMENT_PACK_ACCESS=diagnostic

@$ORACLE_HOME/rdbms/admin/awrrpt.sql (for local database)

Enter value for report_type: html (or text or active-html)

Enter value for num_days: 1

Enter value for begin_snap: 42

Enter value for end_snap: 43 (Note: Correct scoping is vital !!)

Enter value for report_name: bowie_report

How To Generate AWR Reports

@$ORACLE_HOME/rdbms/admin/awrrpti.sql (for specific database)

@$ORACLE_HOME/rdbms/admin/awrgrpt.sql (for local RAC database)

@$ORACLE_HOME/rdbms/admin/awrgrpti.sql (for specific RAC database)

@$ORACLE_HOME/rdbms/admin/awrsqrpt.sql (for specific SQL Id on local database)

@$ORACLE_HOME/rdbms/admin/awrsqrpi.sql (for specific SQL Id on specific database)

Having A Baseline Can Help

- Having a baseline of when the database is hunky dory helps when things go bad:

```
BEGIN DBMS_WORKLOAD_REPOSITORY.create_baseline (
      start_snap_id => 420,
       end_snap_id => 422,
       baseline_name => 'HUNKY DORY');
END; /
```

- If the database is suddenly problematic, then DB Times have increased. Question: Why ?
- Easily seeing what has "changed" can help pinpoint introduced issues
- Can generate AWR Diff report that details differences between 2 AWR reports:
    - @$ORACLE_HOME/rdbms/admin/awrddrpt.sql (difference report for local database)
    - @$ORACLE_HOME/rdbms/admin/awrgdrpt.sql (difference report for RAC database)

Useful AWR scripts

By default, AWR reports are generated every hour and retained for 8 days. To change:

```
BEGIN DBMS_WORKLOAD_REPOSITORY.modify_snapshot_settings(
   retention => 86400, -- minutes, 60 Days
   interval => 30, -- minutes
   topnsql => 50);
END;
```

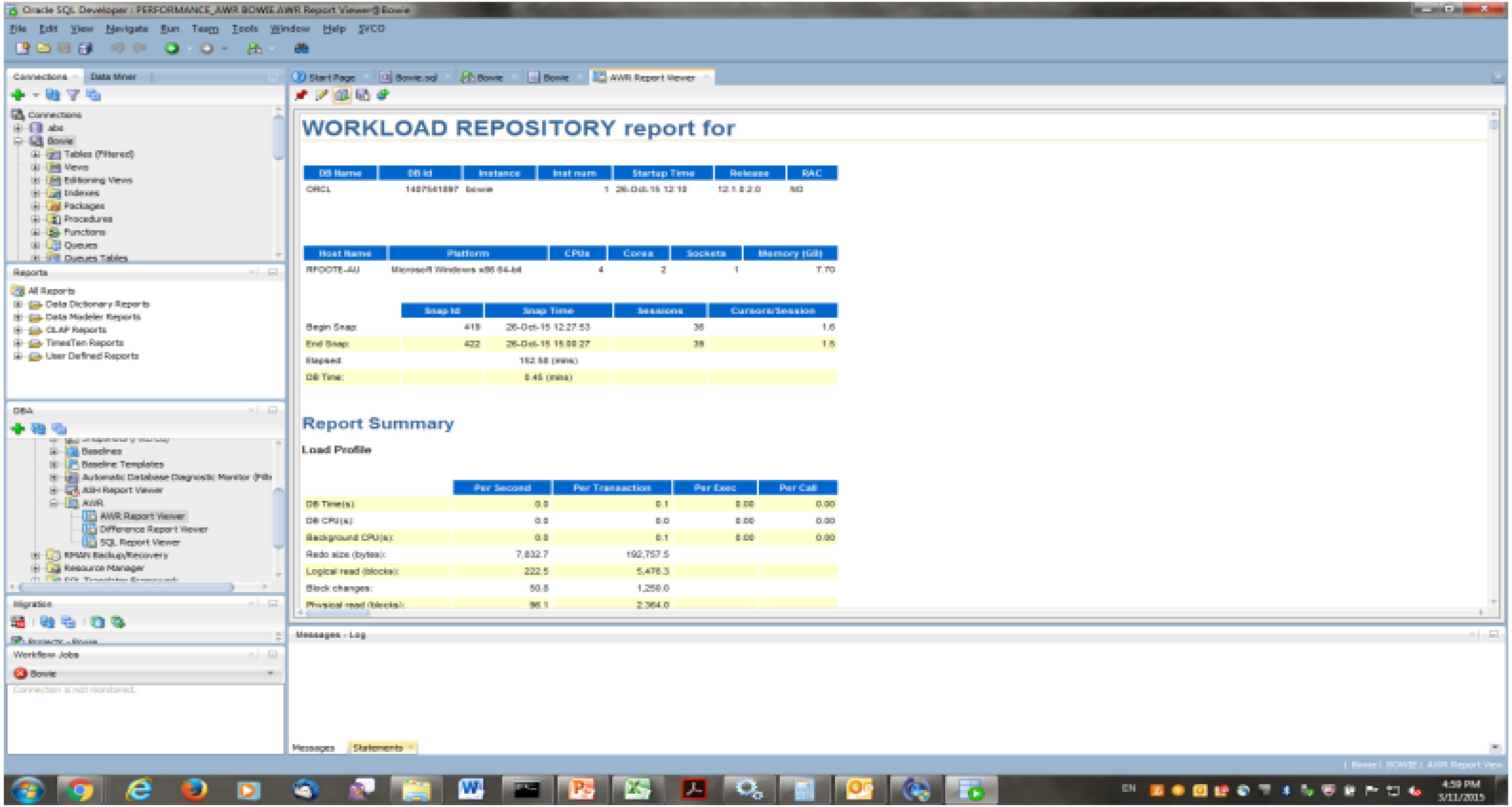
To manually create a new AWR snapshot:

```
EXEC DBMS_WORKLOAD_REPOSITORY.create_snapshot;
```

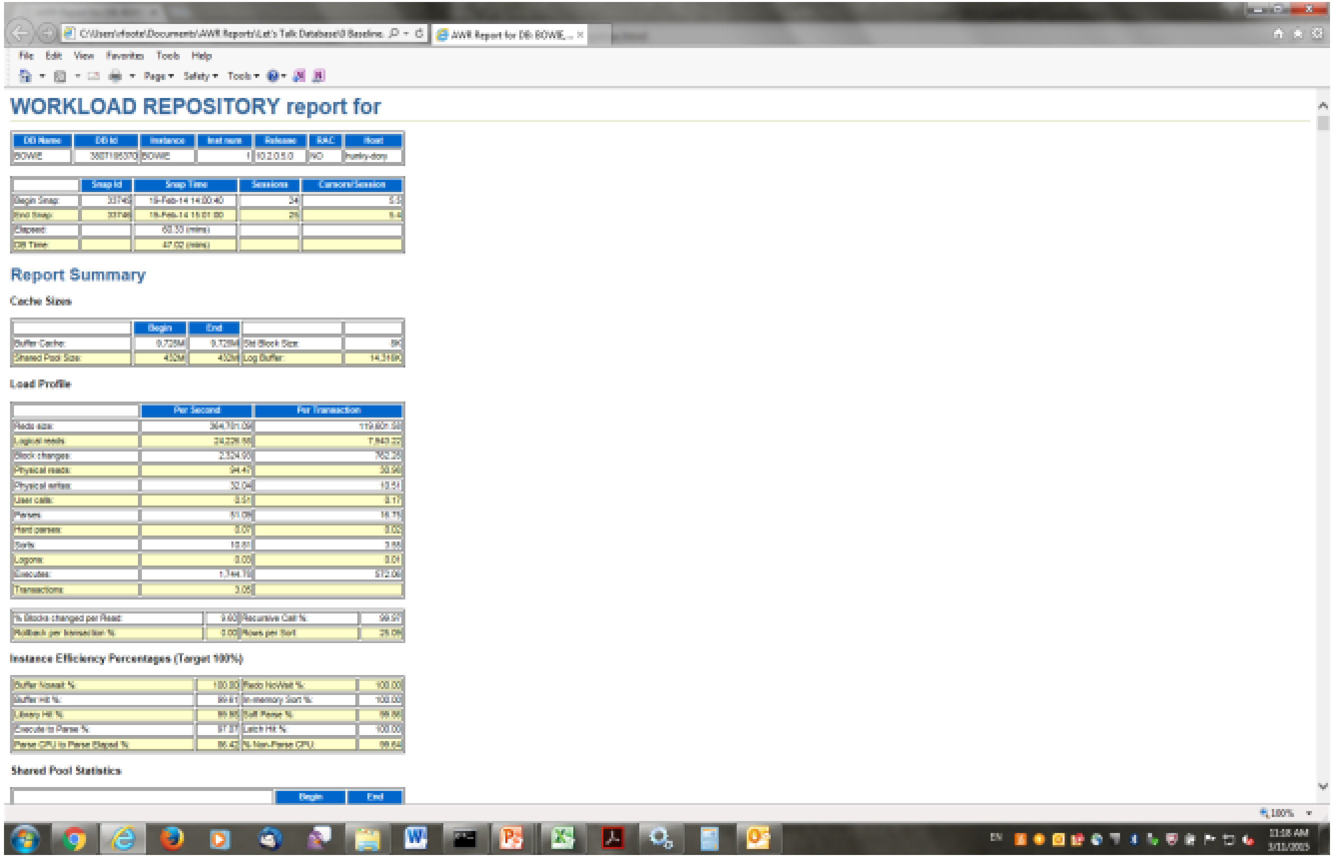
To drop a range of unwanted snapshots:

```
BEGIN DBMS_WORKLOAD_REPOSITORY.drop_snapshot_range (
     low_snap_id => 42, high_snap_id => 84);
END;
```

Oracle Enterprise Manager or SQL Developer



Focus Today On “How” To Best Interpret AWR Reports



Example: Log File Parallel Write vs. Log File Sync Times

Average log file parallel write times:

100 x 1ms, 20 x 9ms and 1 x 809ms = 100+180+809/111 = 1089/121 = 9ms

Average log file sync times (very very *simplistically*):

Assuming fewer log file syncs per log write over 1ms on average, waiting 1/2 the actual LFPW time:

(200 x 1 + 100 x 4.5 + 30 x 405)/(200 + 100 + 30) = 12800/330 = 39ms

Note: In reality, many more transactions on average likely impacted during slower LFPW periods…

What If Only Some Sessions Have Performance Issues?

- In many scenarios, “overall” database performance appears fine
- However, a subset of users are complaining about performance
- Looking at database wide metrics are useless as useful data is drowned out by general system info
- Once data is aggregated, specific information is lost
- Averages can be very misleading (database/session)
- Database level stats don’t necessarily translate to specific performance issue
- Need to be able to drill down on the DB time of just the specific users
- Enter Active Session History (ASH)
- Can also use Extended SQL Trace (DBMS_MONITOR package)

TKPROF: Standard SQL Trace

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          6          0           0
Execute      1      0.06      62.46          0       7772          4           1
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.06      62.47          0       7778          4           1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         0          0          0  UPDATE  ZIGGY (cr=7772 pr=0 pw=0 time=62462379 us)
         1          1          1   TABLE ACCESS FULL ZIGGY (cr=7771 pr=0 pw=0 time=63686 us cost=2122 size=20 card=1)
```

Classic example: why the massive difference between CPU time and elapsed time ?

Let’s Check Database Level AWR Report

```
Top 5 Timed Events
~~~~~~~~~~~~~~~~~~                                                   % Total
Event                                         Waits      Time (s)   Ela Time
-------------------------------------------- ------------ ----------- -----------
db file sequential read                            943,457      18,678       46.33
db file scattered read                             381,532       6,059       15.03
CPU time                                                         5,627       13.96
direct path read (lob)                             326,048       5,550       13.77
SQL*Net more data to client                        204,957       3,051        7.57
```

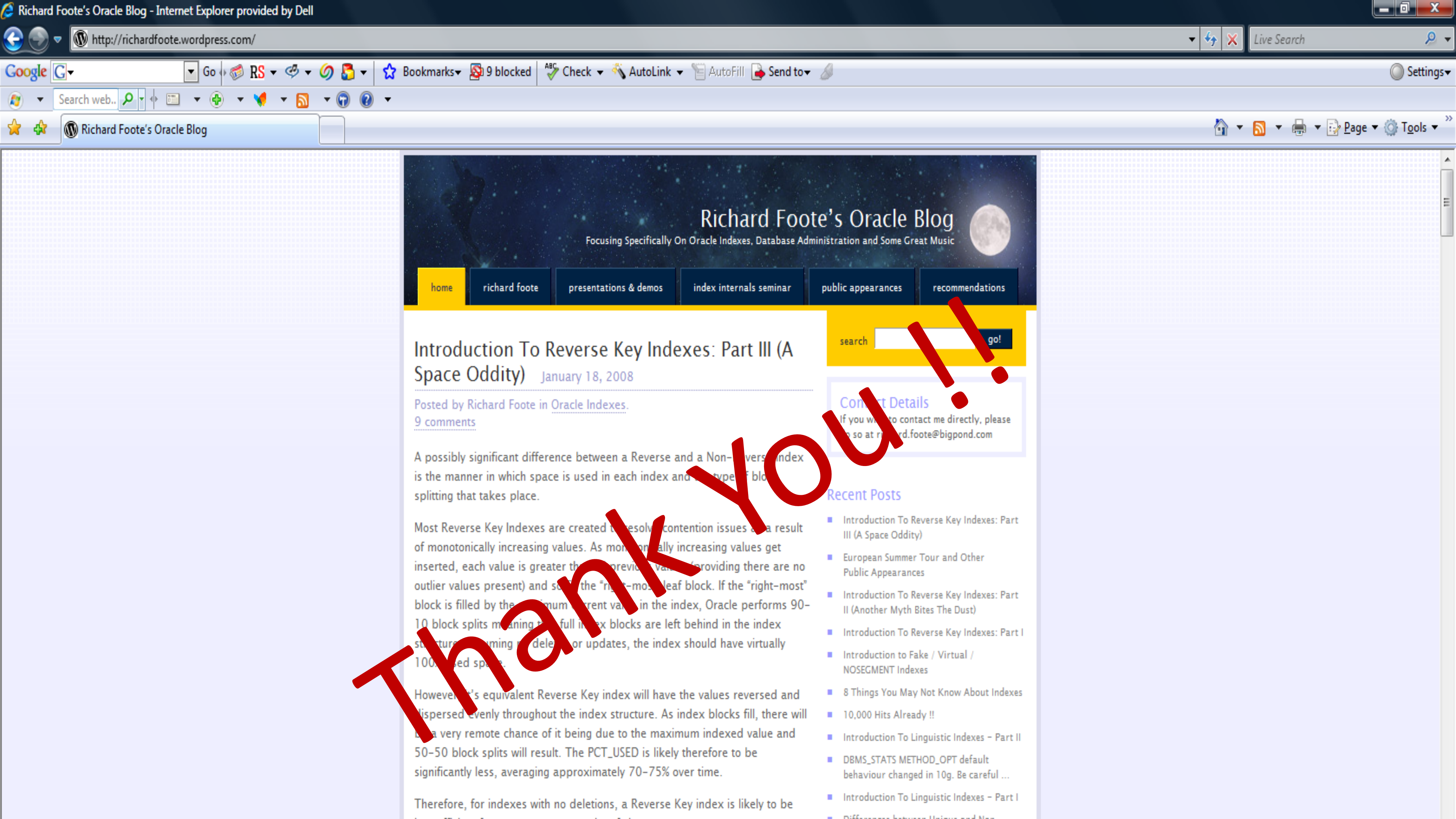
It certainly looks like we might have an I/O related problem here …

TKPROF: Extended SQL Trace

```
Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  enq: TX - row lock contention                   1       60.86         60.86
  SQL*Net message to client                       1        0.00          0.00
  SQL*Net message from client                     1        5.90          5.90
********************************************************************************
```

With extended SQL wait data, we're lead to the actual cause of the problem

Note: Targeted ASH data is generally sufficient to also determine actual problem



Richard Foote's Oracle Blog - Internet Explorer provided by Dell

http://richardfoote.wordpress.com/

Richard Foote's Oracle Blog

Focusing Specifically On Oracle Indexes, Database Administration and Some Great Music

home | richard foote | presentations & demos | index internals seminar | public appearances | recommendations

Introduction To Reverse Key Indexes: Part III (A Space Oddity) January 18, 2008

Posted by Richard Foote in Oracle Indexes.
9 comments

A possibly significant difference between a Reverse and a Non-[illegible]vers[illegible] index is the manner in which space is used in each index and [illegible] type [illegible] block splitting that takes place.

Most Reverse Key Indexes are created t[illegible] resolve contention issues [illegible] a result of monotonically increasing values. As mon[illegible]ally increasing values get inserted, each value is greater th[illegible] previo[illegible] val[illegible] (providing there are no outlier values present) and s[illegible] the "ri[illegible]t-mos[illegible] leaf block. If the "right-most" block is filled by the [illegible]mum [illegible]rent va[illegible] in the index, Oracle performs 90-10 block splits m[illegible]aning t[illegible] full i[illegible]x blocks are left behind in the index st[illegible]ture[illegible]ming [illegible] dele[illegible] or updates, the index should have virtually 100[illegible]ed sp[illegible]

However[illegible]'s equivalent Reverse Key index will have the values reversed and [illegible]ispersed evenly throughout the index structure. As index blocks fill, there will [illegible] a very remote chance of it being due to the maximum indexed value and 50-50 block splits will result. The PCT_USED is likely therefore to be significantly less, averaging approximately 70-75% over time.

Therefore, for indexes with no deletions, a Reverse Key index is likely to be

search go!

Con[illegible]t Details

If you w[illegible] to contact me directly, please [illegible] so at r[illegible]rd.foote@bigpond.com

Recent Posts

- Introduction To Reverse Key Indexes: Part III (A Space Oddity)
- European Summer Tour and Other Public Appearances
- Introduction To Reverse Key Indexes: Part II (Another Myth Bites The Dust)
- Introduction To Reverse Key Indexes: Part I
- Introduction to Fake / Virtual / NOSEGMENT Indexes
- 8 Things You May Not Know About Indexes
- 10,000 Hits Already !!
- Introduction To Linguistic Indexes – Part II
- DBMS_STATS METHOD_OPT default behaviour changed in 10g. Be careful ...
- Introduction To Linguistic Indexes – Part I

Oracle Indexing Internals and Best Practices 5 Day Webinars

8-12 October, 6-10 November 2018 (4 Hours Daily)

Of benefit to DBAs, Developers, Solution Architects and anyone else interested in designing, developing or maintaining high performance Oracle-based applications/databases.

Examines most available index structures/options & discusses in considerable detail how indexes function, how/when they should be used & how they should be maintained. Also how indexes are costed & evaluated by the Cost Based Optimizer (CBO) & how appropriate data management practices are vital for an effective indexing strategy. Covers many useful tips and strategies to maximise the benefits of indexes on application/database performance & scalability.

Running between 10am-2pm Zurich Time 8th – 12th October 2018

richardfooteconsulting.com/indexing-webinar/