Managing a Large OLTP Database

Paresh Patel

Database Engineer

11/19/2014

Agenda

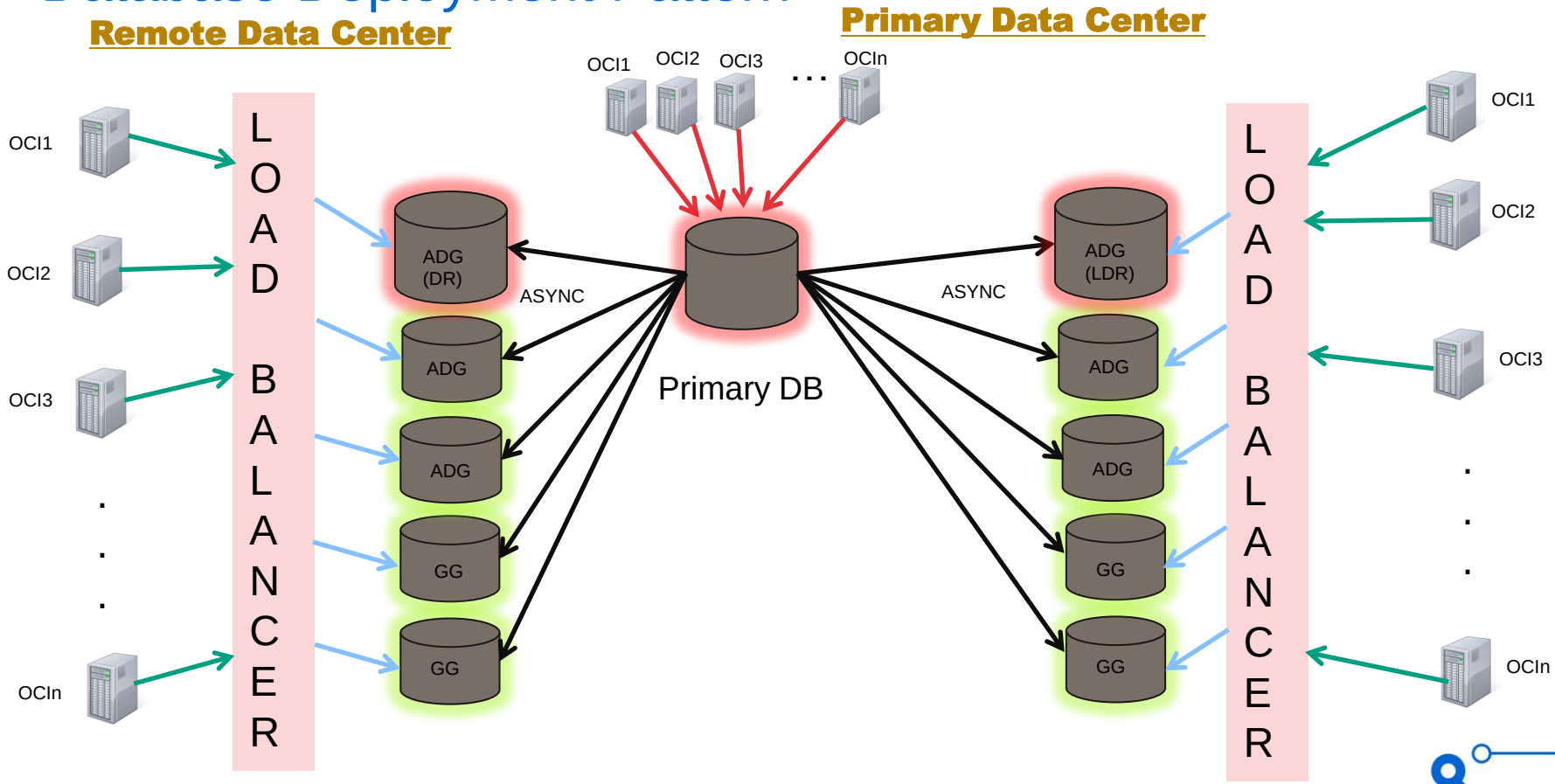
- Introduction to PayPal
- Who am I
- Overview of PayPal Database infrastructure
- Capacity management
- Planned maintenances
- Performance management
- Troubleshooting
- Summary
- Q & A

Disclaimer: Some of the observations here may not be applicable to your environment so test them out or contact Oracle before implementing.

Who am I

- Database Engineer, MTS2
- Oracle RAC Certified Professional with more than a decade’s experience starting with Oracle 9i
- Oracle RAC, ADG, performance tuning and GoldenGate expert
- Conversant with MongoDB, Cassandra and Couchbase

Database Deployment Pattern

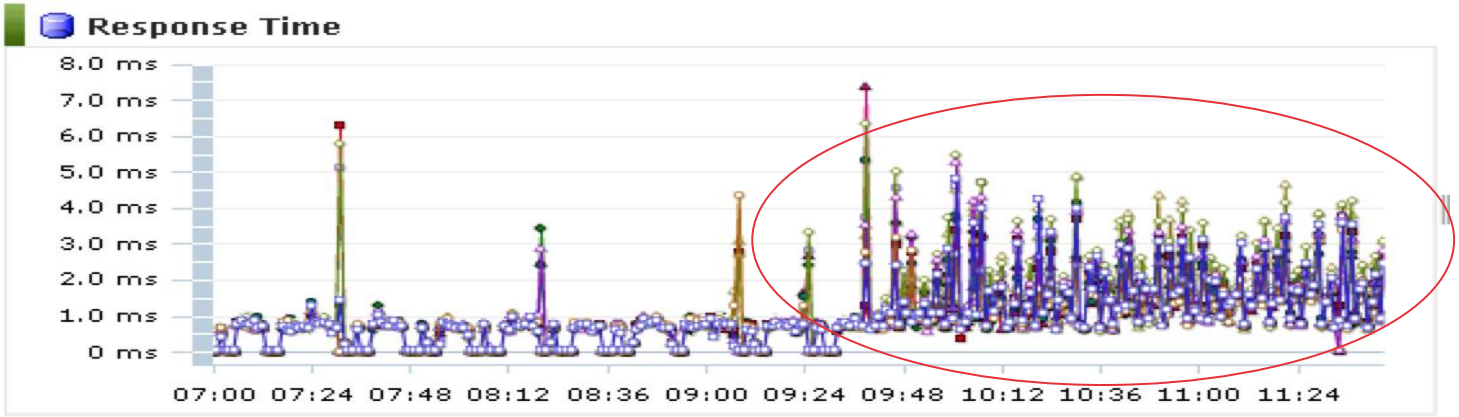


Note: Primary, all ADGs and GGs targets are RAC clusters.

Capacity Management

- To support defined business goals, capacitize database tier to provide uninterrupted service to end users
- Following KPI are used to determine how much business DB tier can support,
  - Storage Read/Write IOPS
    - Virtual Instruments

      Output from VI



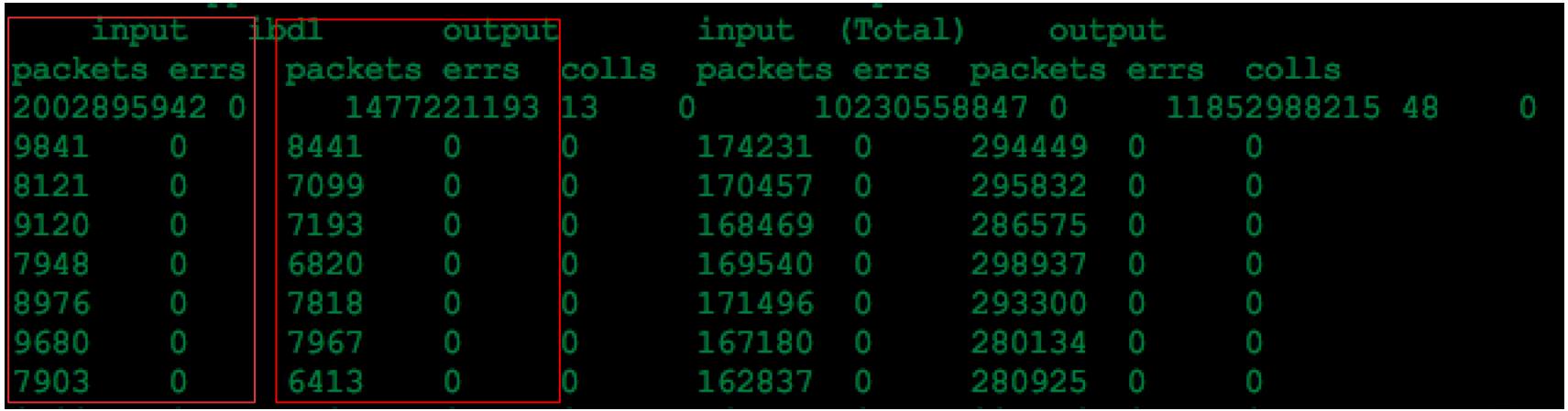
    - asmcmd iostat
  - CPU
    - vmstat

Capacity Management

Continued…

- Interconnect(applicable to Oracle RAC)
  - nmon utility (AIX)
  - netstat -i -I ibd1 -P udp 1 (Solaris, AIX)
  - /usr/sbin/perfquery --extended <lid> <port> (Exadata)
  - » Ibstat command provides lid and port
  - DBA_HIST_IC_DEVICE_STATS (Populated only when UDP protocol is used)

Output from netstat command measuring in and out packets per second



input	ibd1	output		input	(Total)	output	
packets	errs	packets	errs	colls	packets	errs	packets	errs	colls
2002895942	0	1477221193	13	0	10230558847	0	11852988215	48	0
9841	0	8441	0	0	174231	0	294449	0	0
8121	0	7099	0	0	170457	0	295832	0	0
9120	0	7193	0	0	168469	0	286575	0	0
7948	0	6820	0	0	169540	0	298937	0	0
8976	0	7818	0	0	171496	0	293300	0	0
9680	0	7967	0	0	167180	0	280134	0	0
7903	0	6413	0	0	162837	0	280925	0	0

Capacity Management

Continued…

- Latency of cluster related wait events
  - V$EVENT_HISTOGRAM
  - DBA_HIST_EVENT_HISTOGRAM
  - Goal is to keep avg wait time for GC * grant wait events below a ms
  - Goal is to keep avg wait time for GC block transfer wait events below 1.5 ms

As you see in the AWR snippet below, it provides details about cluster related wait events

	Wait		Event		Wait Time			Summary Avg Wait Time (ms)				
I#	Class	Event	Waits	%Timeouts	Total(s)	Avg(ms)	%DB time	Avg	Min	Max	Std Dev	Cnt
*		DB CPU			267,443.99		63.81					4
	User I/O	db file sequential read	118,429,418	0.00	68,113.21	0.58	16.25	0.57	0.56	0.58	0.01	4
	Cluster	gc current block 3-way	41,226,297	0.00	56,461.66	1.37	13.47	1.37	1.25	1.43	0.08	4
	Cluster	gc current block 2-way	62,417,483	0.00	51,566.60	0.83	12.30	0.82	0.78	0.87	0.04	4
	Cluster	gc cr grant 2-way	77,397,293	0.00	46,600.30	0.60	11.12	0.61	0.57	0.63	0.03	4
	Cluster	gc current grant 2-way	12,361,092	0.00	7,356.98	0.60	1.76	0.63	0.59	0.65	0.03	4
	Cluster	gc buffer busy acquire	1,613,348	0.00	2,095.40	1.30	0.50	1.30	1.25	1.32	0.03	4
	Cluster	gc current block congested	683,784	0.00	1,716.05	2.51	0.41	2.42	1.68	2.82	0.51	4
	Cluster	gc cr multi block request	1,146,924	0.00	1,714.84	1.50	0.41	1.48	1.42	1.54	0.06	4
	Cluster	gc cr block busy	360,969	0.00	1,079.71	2.99	0.26	2.75	2.06	3.02	0.46	4

Database Monitoring

- Homegrown tools
  - Provides holistic view of all databases
    - Helps in detecting and mitigating a problem quickly
  - Provides detailed instance level view of all critical metrics
    - Executions, redo/sec, active sessions, physical reads, consistent gets, buffer gets, load, CPU etc.
    - Same stats are used to derive deviations for key metrics

Snippet from homegrown tool for monitoring Database instance

	CGet	BGet	PhyR	Sent	Execs	FBReq	BuIns	Parse	Writes	BrBW	Load	Log	CPU	LF	DBF	PQ	ACT	REDO	Scan	EnQ
11-17 17:10:45	0K	0K	0K	37K	56	296	0	0	0	0	26.89	0	314	0	12K	0	2	1777	0	0
11-17 17:10:56	0K	0K	0K	37K	60	137	0	0	0	0	23.08	0	215	0	5K	0	2	1745	0	0
11-17 17:11:05	0K	0K	0K	37K	53	291	0	0	0	0	20.45	0	4354	0	2K	0	2	1609	0	0
11-17 17:11:15	0K	0K	0K	37K	58	242	0	0	0	0	17.95	0	227	0	6K	0	2	1695	0	0
11-17 17:11:25	0K	0K	0K	38K	61	297	0	0	0	0	15.88	0	231	0	5K	0	2	1753	0	0
11-17 17:11:36	0K	0K	0K	40K	68	303	0	0	2	0	13.93	0	4686	0	10K	0	2	1727	0	0
11-17 17:11:45	0K	0K	0K	39K	65	291	0	0	1	0	12.59	0	233	0	1K	0	2	1560	0	0
11-17 17:11:55	0K	0K	0K	40K	67	311	0	0	0	0	11.36	0	223	0	7K	0	2	1742	0	0
11-17 17:12:06	0K	0K	0K	39K	67	254	0	0	0	0	10.19	0	5104	0	15K	0	2	1754	0	0
11-17 17:12:15	0K	0K	0K	39K	65	411	0	0	0	0	9.36	0	240	0	23K	0	2	1564	0	0

  - AWR warehouse
    - Helps us monitor key metrics across Read Replicas
    - Keep historical AWR data
    - SQL profiling for W-o-W/M-o-M deviation in execs, lio, pio, cpu, elapsed time

Database Monitoring

Continued…

- AWR
  - @?/rdbms/admin/awrgrpt.sql (Global report of RAC cluster)
  - @?/rdbms/admin/awrgdrpt.sql (Global diff report of RAC cluster)
  - @?/rdbms/admin/awrddrpt.sql (Instance diff report)

NOTE: Generate reports from physical standbys rather than getting them from live

Load Profile

	1st per sec	2nd per sec	%Diff	1st per txn	2nd per txn	%Diff
DB time:	78.3	127.6	63.0	16.0	28.5	77.8
CPU time:	51.0	65.1	27.6	10.4	14.5	39.1
Redo size (bytes):	3,372,438.0	3,052,861.2	-9.5	690,214.9	681,057.0	-1.3
Logical read (blocks):	809,166.7	753,215.4	-6.9	165,606.9	168,033.4	1.5
Block changes:	17,079.1	15,163.4	-11.2	3,495.5	3,382.8	-3.2
Physical read (blocks):	31,107.4	33,542.6	7.8	6,366.6	7,483.0	17.5
Physical write (blocks):	3,722.8	3,318.3	-10.9	761.9	740.3	-2.8
Read IO requests:	31,104.2	33,539.6	7.8	6,365.9	7,482.3	17.5
Write IO requests:	3,383.7	3,028.1	-10.5	692.5	675.5	-2.5
Read IO (MB):	243.0	262.1	7.8	49.7	58.5	17.5
Write IO (MB):	29.1	25.9	-10.8	6.0	5.8	-2.9
User calls:	61,802.5	52,118.2	-15.7	12,648.7	11,626.9	-8.1
Parses (SQL):	170.0	143.0	-15.9	34.8	31.9	-8.3
Hard parses (SQL):	0.0	0.0	-25.0	0.0	0.0	0.0
SQL Work Area (MB):	2.5	1.3	-45.9	0.5	0.3	-45.9
Logons:	0.4	0.4	16.7	0.1	0.1	28.6
Executes (SQL):	31,928.1	27,759.0	-13.1	6,534.5	6,192.7	-5.2
Transactions:	4.9	4.5	-8.4			

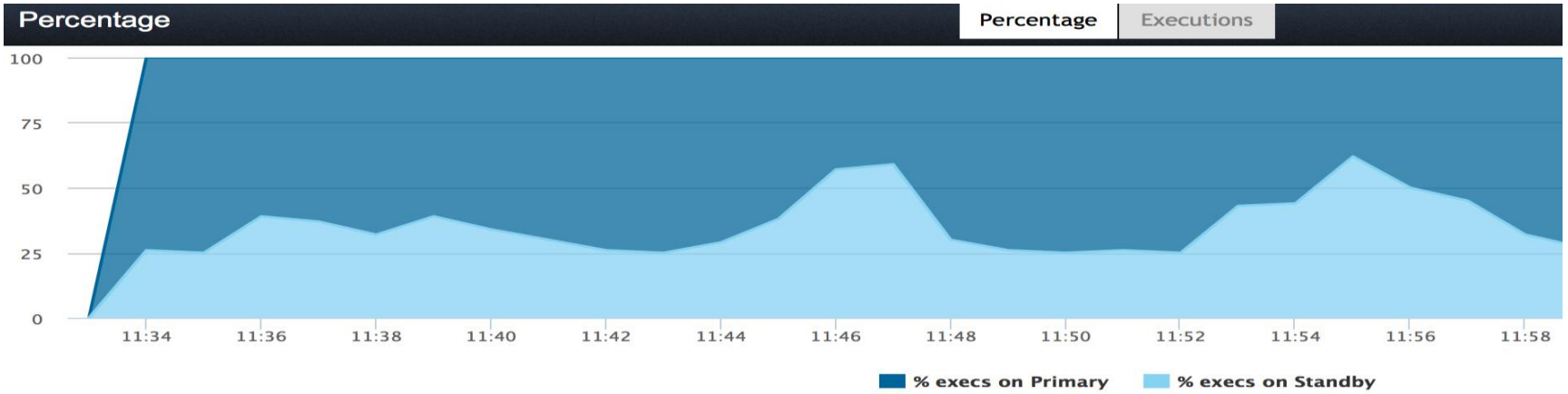
	1st	2nd	Diff
% Blocks changed per Read:	2.1	2.0	-0.1
Recursive Call %:	0.2	0.4	0.2
Rollback per transaction %:	0.0	0.0	0.0
Rows per Sort:	15.5	22.7	7.2
Avg DB time per Call (sec):	0.0	0.0	0.0

Database Monitoring

Continued…

- Monitor ADG
  - Using STATSPACK to monitor ADGs
    NOTE: Please follow MOS: Doc ID 454848.1 to install Statspack on ADG
  - Using homegrown tools

snippet from homegrown tool Executions RO vs Live



- System Metrics
  - Home grown utilities
  - Oracle OSWatcher

Planned Maintenance

- Performing DDL operations on busy tables
  - Set DDL_LOCK_TIMEOUT to 10 sec
    - » If lock is not acquired in specified seconds then DDL will error out
  - Make sure to clear DML batch job prior to issuing this if there is one running
  - Any new DMLs will queue behind this DDL
  - Expect hard parses
- Creating/Dropping Indexes
  - Always create in invisible mode to avoid adverse effects such as plan changes, cursor invalidations, etc.
  - Decide to make it visible after verifying explain plan of the SQLs by setting OPTIMIZER_USE_INVISIBLE_INDEXES=TRUE at session level
  - To avoid impact to production databases, make indexes invisible before dropping them
- Leverage physical standby for testing
  - Convert standby to snapshot standby for testing after taking a GRP

Planned Maintenance

Continued…

- Patching process
  - Build ORACLE_HOME and GRID_HOME Gold Images after performing extensive tests with production like workload and concurrency
  - Make sure all patches can be applied in rolling fashion
  - Client connectivity is tested and verified
  - Patching ramp up planning (start off with patching tier 3 databases)
  - Copy important files from old home to new home
    - » GRID_HOME: gpnp, crs, dbs, cdata, network/admin etc.
    - » ORACLE_HOME: network/admin, dbs etc.
  - In the case of RAC, compile binaries with the protocol other nodes of a cluster using
    - » /usr/ccs/bin/make -f ins_rdbms.mk ipc_rds ioracle (to compile using RDS)
    - » /usr/ccs/bin/make -f ins_rdbms.mk ipc_g ioracle (to compile using UDP)
    - » Use skgxpinfo after setting correct environment variables or nm on $HOME/lib/libskgxp11.so

Planned Maintenance

Continued…

Snippet from nm command output

```
oracle@TESTBOX > /usr/ccs/bin/nm /x/home/oracle/product/11.2.0.2/lib/libskgxp11.so | grep rds
[39]    |              992660|                   4|OBJT |LOCL |2    |13     |skgxp_rds_disabled
[83]    |              992664|                   4|OBJT |LOCL |2    |13     |skgxp_rds_hint
[215]   |               59168|                 384|FUNC |LOCL |0    |10     |skgxpcini_load_rds_param
[511]   |                   0|                   0|FILE |LOCL |0    |ABS    |skgxpg_rds_off.c
[42]    |              965184|                  16|FUNC |LOCL |2    |10     |sskgxp_load_rds_interface
[509]   |              985152|                 208|FUNC |LOCL |0    |10     |ssskgxp_setup_rds
```

– Minimize Brown out during RAC reconfiguration
  - Take instance out of traffic
  - Shutting down instance(s) in RAC cluster has direct impact to ongoing DMLs
  - Shrink DB_CACHE_SIZE, DB_KEEP_CACHE_SIZE and DB_RECYCLE_CACHE_SIZE pools gradually
    - » alter system set DB_CACHE_SIZE=5GB scope=memory sid='A_1';

Planned Maintenance

Continued...

– Database Switchover to Physical Standby

- To minimize the downtime
  - Set below parameter on current primary,
    - » alter system set "_SWITCHOVER_TO_STANDBY_OPTION"="OPEN_ONE_IGNORE_SESSIONS"; (applicable from 11.2.0.2 onwards)

      NOTE: Killing session before Database switchover could take mins. To avoid that, we set this parameter which essentially ignore the sessions. In Oracle Database 12c, this parameter is default.
  - Defer all archive destinations except new primary target
  - Enable flashback and take a Guaranteed Restore Point
  - Mount all instances of new primary target before switchover to avoid brown out during RAC reconfiguration
  - Create Online Redo Log files on new primary target
  - Set following parameters on new primary target to avoid high Physical reads and load
    - » _DB_BLOCK_PREFETCH_QUOTA = 0
    - » _DB_BLOCK_PREFETCH_LIMIT = 0
    - » _DB_FILE_NONCONTIG_MBLOCK_READ_COUNT= 0
    - » _DB_CACHE_PRE_WARM = FALSE

      NOTE: These parameters help disabling read ahead right after switchover. Only Index full scan operations get benefited by this.
  - Once switchover to Physical standby command completed successfully on current primary and after MRP detects the End-Of-Redo indicator, issue switchover to primary on Physical standby. Old primary can be shutdown after new primary up and running

Planned Maintenance

Continued…

- Database Switchover to Physical Standby
  - In the failover situation, to avoid rebuilding all standbys, flash them back to activation SCN and apply redo from new target

    NOTE: Enable flashback on standbys before applying any redo If they didn't have it enabled.
  - How to switchover/failover GoldenGate:
    - » Copy over dirprm, dirchk directories to new target
    - » Make necessary changes in configuration parameters like RMTTRAIL, CACHEDIRECTORY etc.
    - » Use TRANLOGOPTIONS ARCHIVEDLOGONLY to recover data from archive logs in failover situation
- Plan stability is the key
  - Explain plan stays stable during various growth phases of a segment
  - Avoid plan invalidation when stats are published
  - Disable _OPTIM_PEEK_USER_BINDS parameter
  - Less Data skewness
  - Set stats manually to derive the explain plan

Planned Maintenance

Continued…

- Plan stability is the key
  - STATS we set manually
    - » New Table: # of row set to 1,000,000 and # blocks to 100,000
    - » PK based Index stats: # of blocks set to 1,000 # distinct values to 1,000,000 clustering factor to 100,000
    - » Non-unique Index stats: # of blocks set to 1,500 # distinct values to 500,000 clustering factor to 150,000
    - » Column stats: Density set to 1/no of distinct rows

    Use DBMS_STATS to set stats manually

    dbms_stats.set_table_stats

    dbms_stats.set_index_stats

    dbms_stats.set_column_stats
  - To avoid overhead of auto stats collection job
  - Investigating SQL Plan baselines for next upgrade cycle

Performance Management

– Oracle RAC

- Oracle RAC works great but there is certain amount of overhead on CPU depending on workload
- To reduce overhead on CPU, set workload isolation to subset of nodes of a cluster using Database services
- LMS processes directly impact system CPU utilization and interconnect traffic
- Starting/stopping instance causes RAC reconfiguration
  - » To reduce reconfiguration during planned maintenance, please follow the tips provided in “Planned Maintenance” section above
- UDP protocol over Ethernet and use RDS protocol over IB. Please check http://www.oracle.com/technetwork/database/clustering/tech-generic-unix-new-166583.html for certification Matrix
- RDS is low latency protocol compare to UDP but it doesn’t support Active-Active configuration unless bonding is done at OS level
- Use UDP to enhance the network throughput
- Always start LMS, LMD, LGWR and VKTM processes with RT priority,
  - » Set _HIGH_PRIORITY_PROCESSES to ‘LMS*|VKTM|LMD*|LGWR’
  - » chmod 4750 $ORACLE_HOME/bin/oradism; chown root:dba $ORACLE_HOME/bin/oradism

```
oracle@TESTBOX > ps -eo  class,pri,args | grep lms | grep -v grep
  FX   60 ora_lms2_TEST_2
  FX   60 asm_lms0_+ASM2
  FX   60 ora_lms0_TEST_2
  FX   60 ora_lms1_TEST_2
  FX   60 ora_lms3_TEST_2
  FX   60 ora_lms5_TEST_2
  FX   60 ora_lms4_TEST_2
  FX   60 ora_lms6_TEST_2
```

Performance Management

Continued…

- Oracle RAC
  - Disable DRM on critical databases as it brings on unacceptable and unpredictable freezes
    - » Disable it via setting _GC_POLICY_TIME parameter to 0
  - Monitor avg response time for cluster related wait events
  - Disable crs autostart and set “RESTART_ATTEMPS” to 0 for DB resource to avoid crs and database coming up after crash
    - » crsctl disable crs
    - » crsctl modify res ora.testdb.db –attr “RESTART_ATTEMPTS=0”
- ASSM vs MSSM
  - With a very high level of concurrency, ASSM may cause contention while MSSM allows you to set freelist and freelist groups with larger values
  - Use ASSM tablespace to create index online due to a bug which gets exposed only in MSSM
    - » Bug 18715233 (ORA-00600: internal error code, arguments: [kdifind:objdchk_kcbgcur_6], [1], [31226], [0], [0], [], [], [], [], [], [], [])
- Data Reorganization
  - Put data related to one logical entity in fewer data blocks periodically
  - If the rows of a table on disk are sorted in the same order as the index keys, the database will perform a minimum number of I/Os on the table to read rows via an index
  - Keep old and new tables in sync using Oracle GoldenGate and switch public synonym to new table

Performance Management

Continued…

- Active Data Guard
  - All the blocks are mastered on a node where media recovery is running
  - Starting/Stopping media recovery invokes RAC reconfiguration
  - Query response time on node where MRP is running is always higher than non-MRP node(s)
  - In primary database crash event, query response time on ADG goes up right after primary comes back online as ADG tries to apply redo fast to resolve apply lag
  - For critical read-mostly Databases, we maintain mix of ADG and Oracle GoldenGate reader farm
  - For quick session failover, set _ABORT_ON_MRP_CRASH to true to crash all instances of a cluster. Create a crs resource to introduce same behavior on GG based ROs

    NOTE: ADG Internals by Sai Devabhaktuni http://sai-oracle.blogspot.com/2012/11/internals-of-active-dataguard.html

Snippet of ADG monitoring from homegrown tool

Date	Time	DBUQNm	DBInst	Role	ACnt	TCnt	MXV	MRP	InTrf	RTA	LagS	TLagS	CkpT	AlyMB	R_IO	RFS_W	RFS_R	MStat	T:S:B
2014-11-18	12:02:21	TEST_DC_L1	TEST_1	PHY	131	2105	15000	Y	1856	Y	293	0	0	11.63	1	0	N	AL	3:1305:40901
2014-11-18	12:02:31	TEST_DC_L1	TEST_1	PHY	131	2105	15000	Y	1856	Y	297	0	0	11.63	1	0	N	AL	3:1305:43297
2014-11-18	12:02:41	TEST_DC_L1	TEST_1	PHY	142	2105	15000	Y	1856	Y	301	0	0	11.63	1	0	N	AL	3:1305:45115
2014-11-18	12:02:51	TEST_DC_L1	TEST_1	PHY	131	2105	15000	Y	1856	Y	305	0	0	11.63	0	0	N	AL	3:1305:46754
2014-11-18	12:03:01	TEST_DC_L1	TEST_1	PHY	130	2104	15000	Y	1855	Y	310	0	0	11.63	1	0	N	AL	3:1305:49406
2014-11-18	12:03:11	TEST_DC_L1	TEST_1	PHY	130	2104	15000	Y	1855	Y	314	0	0	11.63	1	0	N	AL	3:1305:51215
2014-11-18	12:03:21	TEST_DC_L1	TEST_1	PHY	130	2103	15000	Y	1859	Y	317	0	0	11.63	2	0	N	AL	3:1305:53823
2014-11-18	12:03:31	TEST_DC_L1	TEST_1	PHY	131	2103	15000	Y	1855	Y	320	0	0	11.63	1	0	N	AL	3:1305:56138

Performance Management

Continued…

- Outliers
  - ASH
  - V$EVENT_HISTOGRAM
- Top SQLs
  - Maintain inventory of TOP SQLs (by cluster wait time, executions, buffer gets, CPU etc.)
  - Check AWR diff report or DBA_HIST_SQLSTAT
  - Generate reports for comparing various metric data across ROs from AWR warehousing
- Bigger SGA
  - Turn off Automatic SGA management
  - Set appropriate values _LM_TICKETS and GCS_SERVER_PROCESSES
    - » Follow MOS note: Best Practices and Recommendations for RAC databases using SGA larger than 300GB (Doc ID 1619155.1)
  - Consider configuring DB_KEEP_CACHE_SIZE and DB_RECYCLE_CACHE_SIZE pools and put appropriate segments in them
- Managing Sequences
  - Ordered sequences present scalability challenges due to high GC message activity
  - Try to keep sequence no-ordered and route write workload to designated node
  - Watch out for the large gaps in sequence values if write traffic is routed to set of nodes
  - Create logon trigger to handle sequence order in failover scenario

Troubleshooting

- V$SESSION
  - Active session count is an indicator of user activities in Database
  - Action, module and client_identifier can reveal most important information about application requests
    - » OCI client can set bind variables' value, client application name etc. using APIs

    NOTE: We use this workaround as _optim_peek_user_binds parameter is set to FALSE
  - WAIT_TIME_MICRO provides how long the session is waiting or waited if it's not waiting
  - EVENT provides why session is waiting
  - Most of the time, query on v$session can provide enough clues to diagnose the issue further
- V$ACTIVE_SESSION_HISTORY
  - Provides
    - » Timing and duration of the issue
    - » session details
    - » wait event information
    - » Blocking session information
    - » Wait time information
    - » IN_XXXX columns provide session's execution state information
  - Check last X mins of data to get clues on where the problem could be
  - ASH data can be inconsistent due to lack of read consistency in underlying X$ fixed tables
  - Always take a copy of v$active_session_history right after an incident

    NOTE: Deep dive into ASH by Sai Devabhaktuni http://sai-oracle.blogspot.com/2012/11/deep-dive-into-ash.html

Troubleshooting

Continued…

- Homegrown tools
  - Provides us the various database metrics from V$SESSION, V$SYSSTAT, V$SYSTEM_EVENT every 10 seconds
  - Executions, redo/sec, active sessions, physical reads, consistent gets, buffer gets, load, CPU etc.

  NOTE: Doesn't use any GV$ query as Oracle spawns new processes on all instances of RAC
- Reproducing issue in test environment
  - Some of the issues happen in production don't produce enough diagnostic data for Oracle to provide RCA and possible fix
  - Identify the workload and concurrency at the time of problem occurrence
  - Set identical environment and run workload with same concurrency
- Log files
  - RDBMS and ASM Alert log files
  - agent, crsd log files
  - gipcd, cssd log files
  - System log files under /var/adm/

Summary

- Always perform scale up tests with 5x workload for new feature, patches and Oracle version upgrade
- Drive the database stack to failure to test capacity limits
- Master important views such as v$session and v$active_session_history
- Take advantage of Snapshot Standby for testing
- Stable Execution plans is the key for stable performance
- Measure capacity by various dimensions including Interconnect
- Monitor databases using complementary set of tools to fully understand the database profile
- Right tools will help troubleshooting the issue quicker

Q & A

Thank You!