

Real Life RAC Performance Tuning

Arup Nanda

Lead DBA

Starwood Hotels



Who am I

- Oracle DBA for 13 years and counting
- Working on OPS from 1999
- Implemented and supported 10g RAC in 83 sites since 2004
- Speak at conferences, write papers, books

Why This Session

- I get emails like this:
 - *We are facing performance issues in RAC. What should I do next?*
- Real Life Advice
 - Common Issues (with Wait Events)
 - Dispelling Myths
 - Formulate a Plan of Attack
 - Real Life Case Study
- prolignence.com/downloads.html

Our RAC Implementation

- Oracle 10g RAC in March 2004
 - Itanium Platform running HP/UX
 - Oracle 10.1.0.2
- Result: Failed ☹️
- Second Attempt: Dec 2004 – 10.1.0.3
- Result: Failed Again ☹️
- Third Attempt: March 2005 – 10.1.0.4
- Result: Success! 😊

Challenges

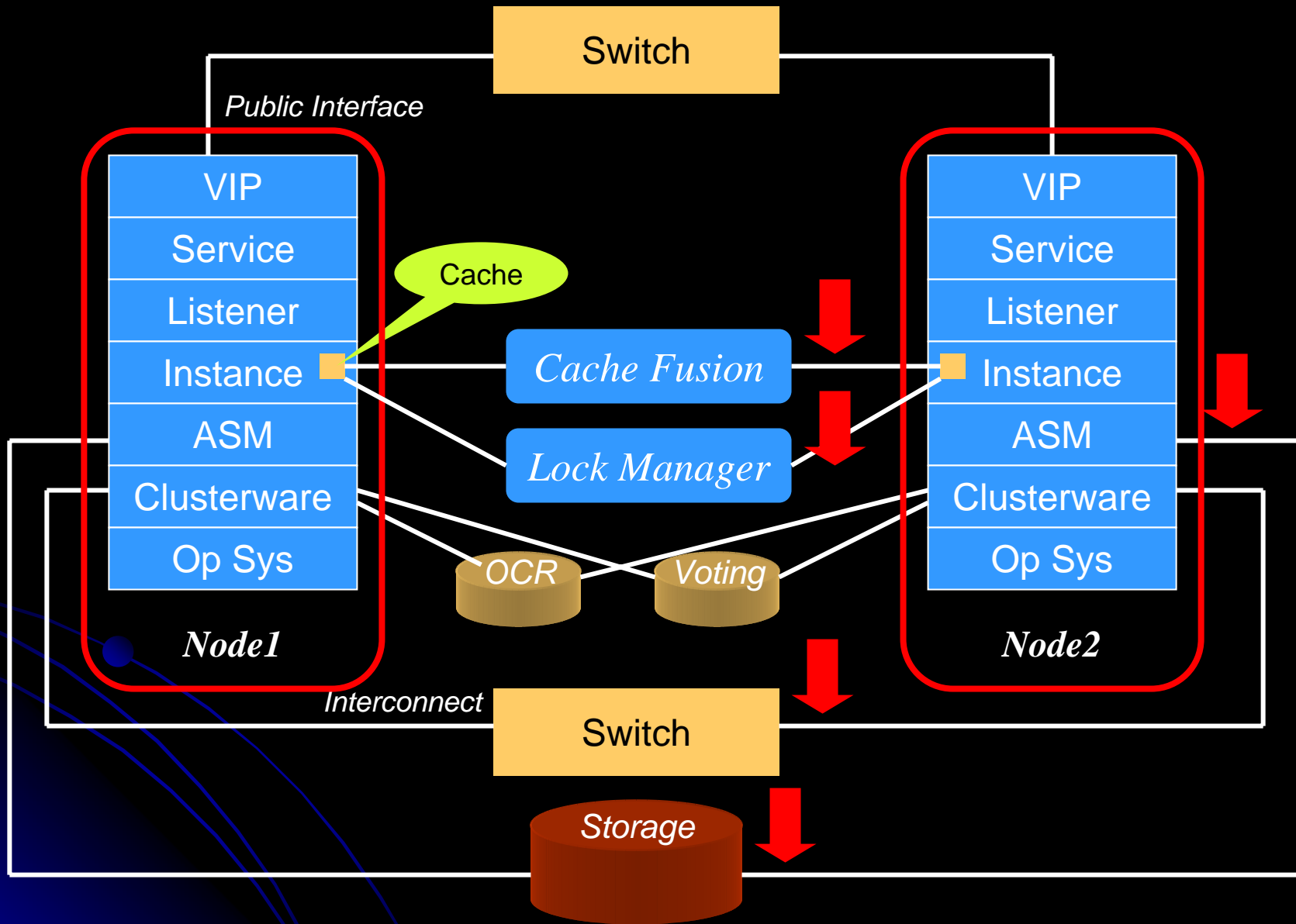
- Technology
 - Lone ranger
 - A lot of “mystery” and disconnected “facts”!
- People
 - Building a team that could not only deliver; but also sustain the delivered parts
 - Each day we learned something new
- In today’s session: real performance issues we faced and how we resolved them, along with wait events.

Why “RAC” Performance?

- All tuning concepts in single instance applied to RAC as well
- RAC has other complexities
 - More than 1 buffer cache
 - Multiple caches – library cache, row cache
 - Interconnect
 - Pinging
 - Global Locking

Why RAC Perf Tuning

- We want to make sure we identify the right problem and go after it
- not just a problem



Areas of Concern in RAC

- More than 1 buffer cache
- Multiple caches – library cache, row cache
- Interconnect
- Global Locking

Cache Issues

- Two Caches, requires synchronization
- What that means:
 - A changed block in one instance, when requested by another, should be sent across via a “bridge”
 - This bridge is the Interconnect

Interconnect Performance

- Interconnect must be on a private LAN
- Port aggregation to increase throughput
 - APA on HPUX
- If using Gigabit over Ethernet, use Jumbo Frames

Checking Interconnect Used

- Identify the interconnect used

```
$ oifcfg getif  
lan902 172.17.1.0 global  
cluster_interconnect  
lan901 10.28.188.0 global public
```

- Is lan902 the bonded interface? If not, then set it

```
$ oifcfg setif ...
```

Pop Quiz

- If I have a very fast interconnect, I can perform the same work in multiple node RAC as a single server with faster CPUs. True/False?
- Since cache fusion is now write-write, a fast interconnect will compensate for a slower IO subsystem. True/False?

Cache Coherence Times

- The time is a sum of time for:
 - Finding the block in the cache
 - Identifying the master
 - Get the block in the interconnect
 - Transfer speed of the interconnect
 - Latency of the interconnect
 - Receive the block by the remote instance
 - Create the consistent image for the user
-
- The diagram uses curly braces to group the steps into three categories:
- CPU:** Finding the block in the cache, Identifying the master, and Get the block in the interconnect.
 - Interconnect:** Transfer speed of the interconnect and Latency of the interconnect.
 - CPU:** Receive the block by the remote instance and Create the consistent image for the user.

So it all boils down to:

- Block Access Cost
 - more blocks -> more the time
 - Parallel Query
- Lock Management Cost
 - More coordination -> more time
 - Implicit Cache Checks – Sequence Numbers
- Interconnect Cost
 - Latency
 - Speed
 - more data to transfer -> more the time

Hard Lessons

- In RAC, problem symptoms may not indicate the correct problem!
- Example:
 - When the CPU is too busy to receive or send packets via UDP, the packets fails and the Clusterware thinks the node is down and evicts it.

OS Troubleshooting

- OS utilities to troubleshoot CPU issues
 - top
 - glance
- OS Utilities to troubleshoot process issues:
 - truss
 - strace
 - dbx
 - pstack

Reducing Latency

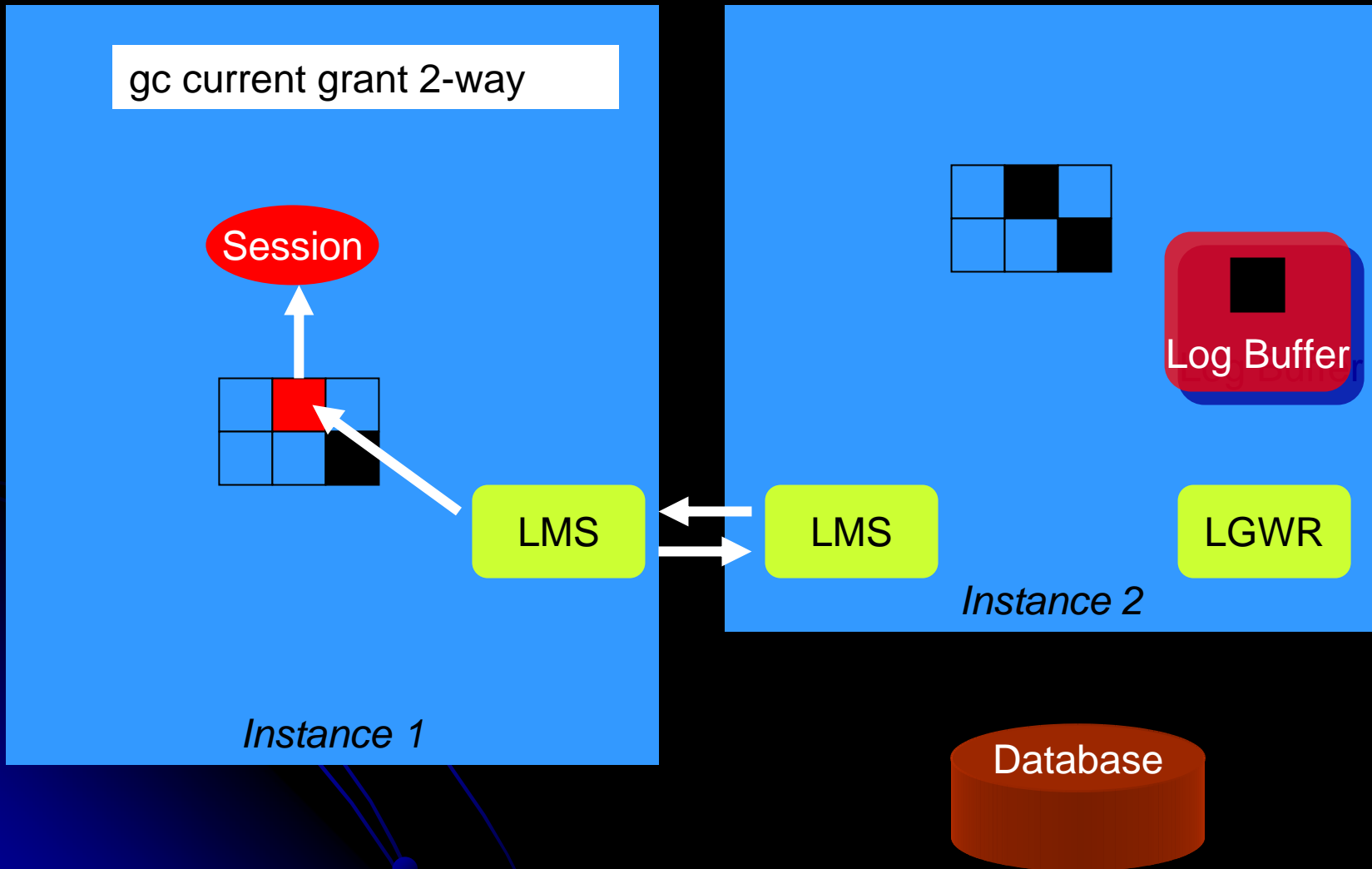
- A factor of technology
- TCP is the most latent
- UDP is better (over Ethernet)
- Proprietary protocols are usually better
 - HyperFabric by HP
 - Reliable Datagram (RDP)
 - Direct Memory Channel
- Infiniband
 - UDP over Infiniband
 - RDP over Infiniband

Start with AWR

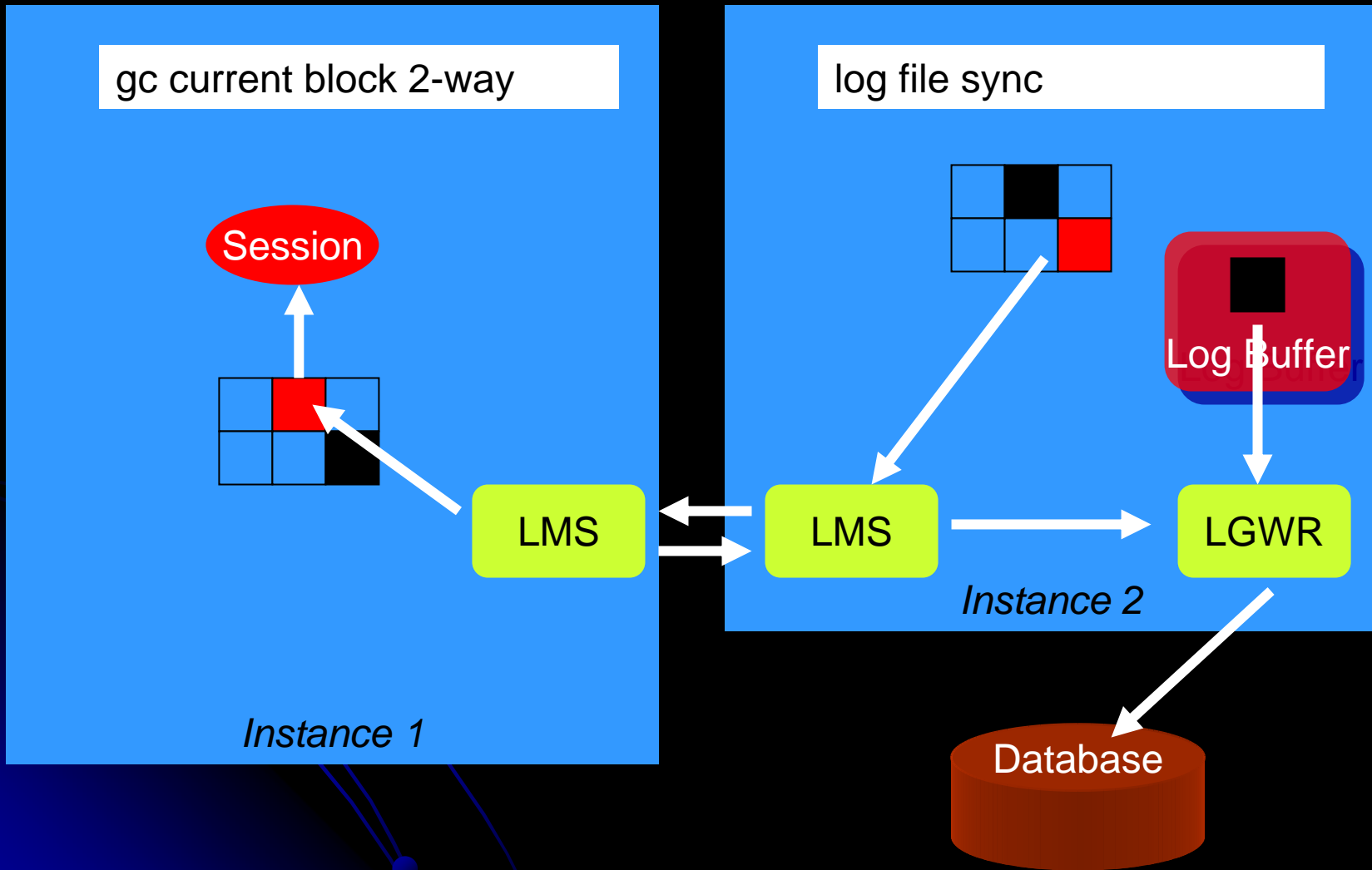
Top 5 Timed Events

Event	Waits	Time(s)	Percent Total DB Time	Wait Class
db file sequential read	3,754,273	30,966	70.67	User I/O
CPU time		8,320	18.99	
db file parallel read	64,468	1,456	3.32	User I/O
gc cr grant 2-way	1,470,759	984	2.25	Cluster
read by other session	79,807	486	1.11	User I/O

gc current|cr grant 2-way



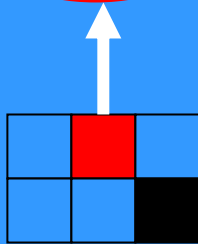
gc current|cr block 2-way



gc current|cr block 3-way

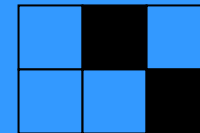
gc current block 3-way

Session



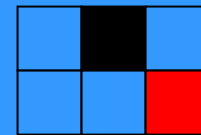
Instance 1

Requestor



Instance 2

Master



Instance 3

Holder

RAC related Stats

RAC Statistics

	Begin	End
Number of Instances:	2	2

Global Cache Load Profile

	Per Second	Per Transaction
Global Cache blocks received:	293.67	11.19
Global Cache blocks served:	271.61	10.35
GCS/GES messages received:	2,655.12	101.17
GCS/GES messages sent:	2,515.61	95.86
DBWR Fusion writes:	11.10	0.42

Global Cache Efficiency Percentages (Target local+remote 100%)

Buffer access - local cache %:	95.89
Buffer access - remote cache %:	0.73
Buffer access - disk %:	3.38

RAC Stats contd.

Global Cache and Enqueue Services - Workload Characteristics

Avg global enqueue get time (ms):	0.3
Avg global cache cr block receive time (ms):	1.1
Avg global cache current block receive time (ms):	1.3
Avg global cache cr block build time (ms):	0.0
Avg global cache cr block send time (ms):	0.1
Global cache log flushes for cr blocks served %:	1.5
Avg global cache cr block flush time (ms):	3.6
Avg global cache current block pin time (ms):	0.0
Avg global cache current block send time (ms):	0.1
Global cache log flushes for current blocks served %:	0.1
Avg global cache current block flush time (ms):	5.7

Global Cache and Enqueue Services - Messaging Statistics

Avg message sent queue time (ms):	0.1
Avg message sent queue time on ksxp (ms):	0.6
Avg message received queue time (ms):	0.0
Avg GCS message process time (ms):	0.0
Avg GES message process time (ms):	0.0
% of direct sent messages:	48.38
% of indirect sent messages:	49.81
% of flow controlled messages:	1.81

Event	Waits	Timeouts	Total Wait Time (s)	Avg wait (ms)	Waits /txn
db file sequential read	3,754,273	0	30,966	8	39.72
db file parallel read	64,468	0	1,456	23	0.68
gc cr grant 2-way	1,470,759	625	984	1	15.56
read by other session	79,807	30,142	486	6	0.84
db file parallel write	216,065	0	433	2	2.29
gc buffer busy	87,088	29,032	410	5	0.92
enq: US - contention	1,280,682	23	392	0	13.55
gc current block 2-way	432,631	276	391	1	4.58
RFS dispatch	8,192	0	376	46	0.09
RFS write	8,190	0	376	46	0.09
gc cr block 2-way	392,748	328	348	1	4.16
gc current grant 2-way	418,405	353	334	1	4.43
db file scattered read	49,224	0	302	6	0.52
log file sync	85,247	143	274	3	0.90
SQL*Net more data from client	69,137	0	222	3	0.73
log file sequential read	8,199	0	205	25	0.09
log file parallel write	179,193	0	203	1	1.90
PX Deq Credit: send blkd	9,213	98	129	14	0.10
gc cr multi block request	261,317	38	125	0	2.77
DFS lock handle	245,617	8	90	0	2.60
gc cr block busy	15,005	39	78	5	0.16
control file sequential read	61,821	0	57	1	0.65
control file parallel write	27,898	0	56	2	0.30
latch: KCL gc element parent latch	2,680	2,231	29	11	0.03
enq: TX - index contention	2,057	23	28	14	0.02
gc current grant busy	39,048	102	27	1	0.41
latch: cache buffers chains	2,486	2,466	20	8	0.03
row cache lock	12,780	63	19	2	0.14
gc current block busy	1,137	5	14	12	0.01
ges inquiry response	19,650	73	13	1	0.21
gc cr grant congested	1,614	1	13	8	0.02
latch: cache buffers lru chain	1,275	0	10	8	0.01
CGS wait for IPC msg	467,085	461,916	10	0	4.94
gcs log flush sync	7,317	1	10	1	0.08

Other GC Block Waits

- gc current/cr block lost
 - Lost blocks due to Interconnect or CPU
- gc curent/cr block busy
 - The consistent read request was delayed, most likely an I/O bottleneck
- gc current/cr block congested
 - Long run queues and/or paging due to memory deficiency.

Hung or Slow?

- Check `v$SESSION` for `WAIT_TIME`
 - If 0, then it's not waiting; it's hung
- When hung:
 - Take a systemstate dump from all nodes
 - Wait some time
 - Take another systemstate dump
 - Check change in values. If unchanged, then system is hung

Chart a Plan

- Rule out the obvious
- Start with AWR Report
- Start with Top-5 Waits
- See if they have any significant waits
- ... especially RAC related
- Go on to RAC Statistics
- Base your solution based on the wait event

Rule out the obvious

- Is interconnect private?
- Is interconnect on UDP?
- Do you see high CPU?
- Do you see a lot of IO bottleneck?
- How about memory?
- Are the apps spread over evenly?
- Do you see lost blocks?

Make Simple Fixes

- Strongly consider RAID 0+1
- Highest possible number of I/O paths
- Use fastest interconnect possible
- Use private collision free domain for I/C
- Cache and NOORDER sequences

Enterprise Manager



Cluster: ndsstg

Latest Data Collected From Target **Mar 21, 2007 9:42:09 PM EST** [Refresh](#)

[Home](#)[Performance](#)[Targets](#)[Interconnects](#)[Topology](#)

The interconnect configuration and internode communication will influence the performance of cluster databases. The tables below show network interfaces on all hosts and network interfaces currently in use by cluster databases. It is important that cluster databases are configured to use a private interconnect for message and block transfers.

View Data

Private Interconnect Transfer Rate (MB/Sec) 0.419

Transfer rate on the private network in the last 5 minutes.

Interfaces by Hosts

View

[Expand All](#) | [Collapse All](#)

Name	Type	Subnet	Interface Type	Total I/O Rate (MB/Sec) (Last 5 Minutes)	Total Error Rate (%) (Last 5 Minutes)
▼ ndsstg	Cluster				
▼ hndssdb1	Host				
lan902	Interface	172.17.1.0	Private	<u>.11</u>	<u>0</u>
▼ hndssdb2	Host				
lan902	Interface	172.17.1.0	Private	<u>.29</u>	<u>0</u>
▼ hndssdb3	Host				
lan902	Interface	172.17.1.0	Private	<u>.39</u>	<u>0</u>

Interfaces in Use by Cluster Databases

Buffer Busy

- Cause

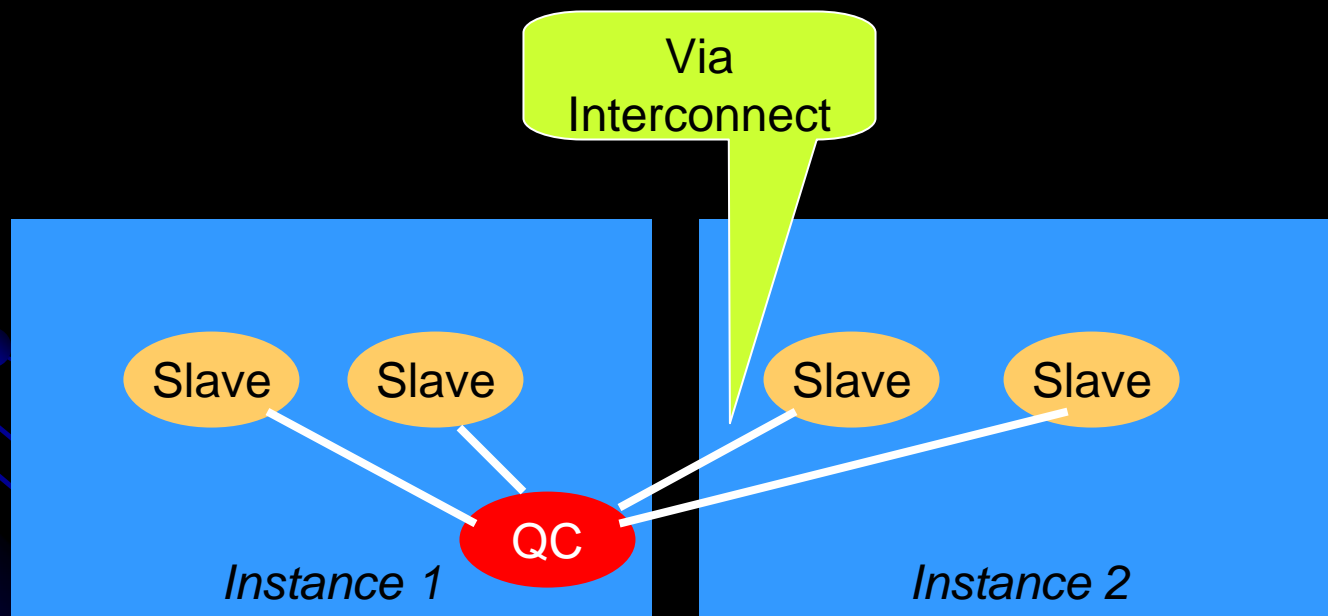
- Instance wants to bring something from disk to the buffer cache
- Delay, due to space not available
- Delay, b'coz the source buffer is not ready
- Delay, I/O is slow
- Delay, b'coz redo log is being flushed

- In summary

- Log buffer flush -> gc buffer busy

Parallel Query

- One major issue in RAC is parallel query that goes across many nodes



Restricting PQ

- Define Instance Groups

Specify in init.ora

```
prodb1.instance_groups='pqgroup1'  
prodb2.instance_groups='pqgroup2'
```

- Specify Instance Groups in Session

```
SQL> alter session set  
parallel_instance_group =  
'pqgroup1';
```

Forcing PQ on both Nodes

- Define a common Instance Group

```
prodb1.instance_groups='pqgroup1'  
prodb1.instance_groups='pq2nodes'  
prodb2.instance_groups='pqgroup2'  
prodb2.instance_groups='pq2nodes'
```

- Specify Instance Groups in Session

```
SQL> alter session set  
parallel_instance_group =  
'pq2nodes';
```

Vital Cache Fusion Views

- `gv$cache_transfer`: Monitor blocks transferred by object
- `gv$class_cache_transfer`: Monitor block transfer by class
- `gv$file_cache_transfer`: Monitor the blocks transferred per file
- `gv$temp_cache_transfer`: Monitor the transfer of temporary tablespace blocks

“Hot” Tables

- Tables, e.g. Rate Plans
 - Small
 - Compact blocks
 - High updates
 - High reads
- Symptoms
 - gc buffer busy waits
- Solution
 - Less rows per block
 - High PCTFREE, INITRANS,
 - ALTER TABLE ... MINIMIZE RECORDS_PER_BLOCK

Hot Sequences

- Symptoms:
 - High waits on Sequence Number latch
 - High waits on SEQ\$ table
- Solution:
 - Increase the cache
 - Make it NOORDER
- Especially AUDSESS\$ sequence in SYS, used in Auditing

Read Only? Say So.

- Reading table data from other instances create “gc *” contentions
- Suggestion:
 - Move Read Only tables to a single tablespace
 - Make this tablespace Read Only

```
SQL> alter tablespace ROD read  
only;
```

Partitioning

- Partitioning creates several segments for the same table (or index)
- => more resources
- => less contention

Monotonically Increasing Index

- Problem:

- “Reservation ID”, a sequence generated key
- Index is heavy on one side

- Symptoms

- Buffer busy waits
- Index block splits

- Solutions:

- Reverse key indexes
- Hash partitioned index (even if the table is not partitioned) 10gR2

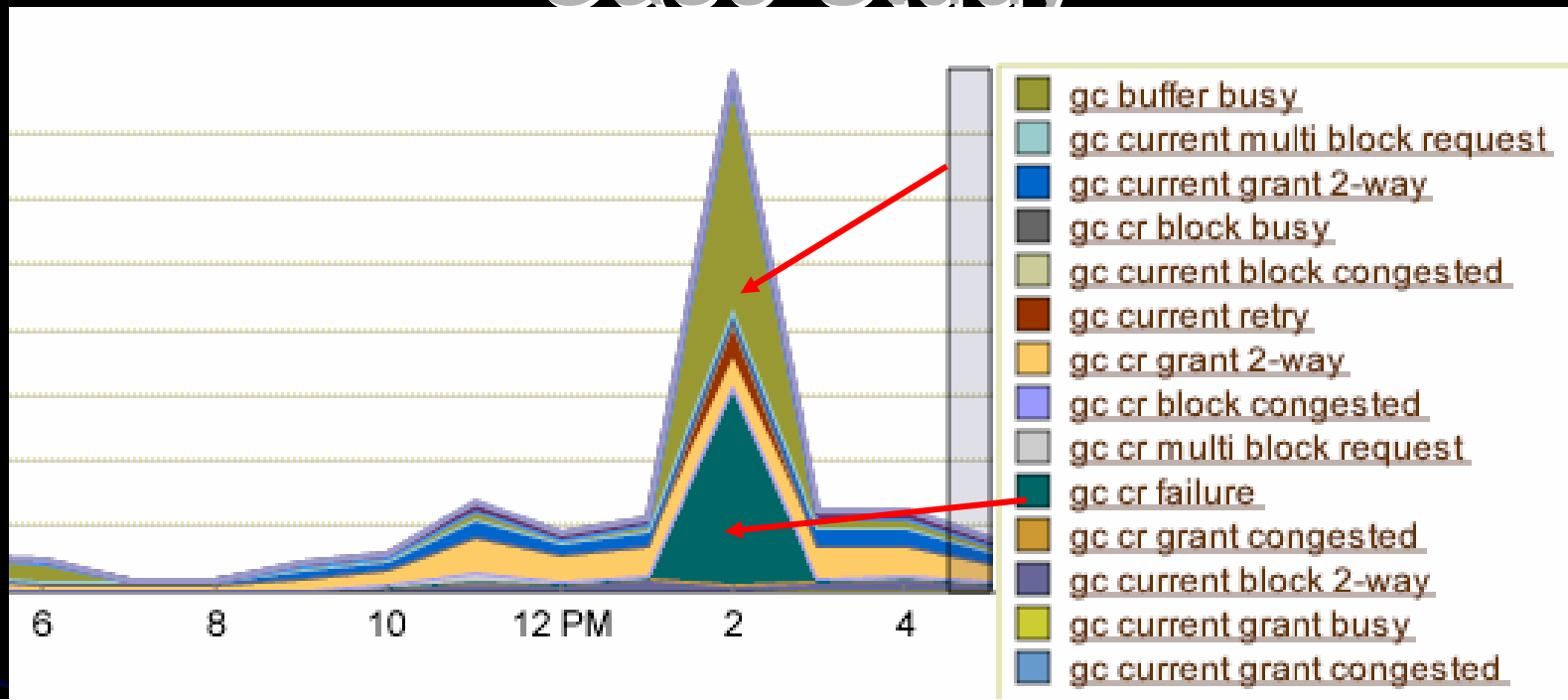
Library Cache

- In RAC, Library Cache is global
- So, parsing cost is worse than non-RAC
- Solutions:
 - Minimize table alters, drops, creates, truncates
 - Use PL/SQL stored programs, not unnamed blocks

Log Files

- In 10g R2, the log files are in a single location:
- `$CRS_HOME/log/<Host>/...`
 - ↳ `racg`
 - ↳ `crsd`
 - ↳ `cssd`
 - ↳ `evmd`
 - ↳ `client`
 - ↳ `cssd/oclsmon`
- `$ORACLE_HOME/racg/dump`

Case Study



Top 5 Timed Events

Event	Waits	Time(s)	Percent Total DB Time	Wait Class
db file sequential read	4,137,096	50,323	24.47	User I/O
gc domain validation	16,456	30,784	14.97	Cluster
gc buffer busy	148,267	26,707	12.99	Cluster
gc cr failure	18,799	22,914	11.14	Cluster
CPU time		17,609	8.56	

Diagnosis

- `ifconfig -a` shows no congestion or dropped packets
- Top shows 1% idle time on node 2
- Top processes
 - LMS and LMD
- And, several Netbackup processes

Further Diagnosis

- SQL:

```
select * from v$instance_cache_transfer
where class = 'data block'
and instance = 1;
```

- Output:

INSTANCE	CLASS	CR_BLOCK	CR_BUSY
-----	-----	-----	-----
CR_CONGESTED	CURRENT_BLOCK	CURRENT_BUSY	CURRENT_CONGESTED
-----	-----	-----	-----
1	data block	162478682	5097149
477721	347917908	2950144	16320267

See
increases

- After sometime:

INSTANCE	CLASS	CR_BLOCK	CR_BUSY
-----	-----	-----	-----
CR_CONGESTED	CURRENT_BLOCK	CURRENT_BUSY	CURRENT_CONGESTED
-----	-----	-----	-----
1	data block	162480580	5097185
477722	347923719	2950376	16320269

- **Diagnosis:**
 - CPU starvation by LMS/D processes caused GC waits.
- **Solution:**
 - Killed the Netbackup processes
 - LMD and LMS got the CPU

Increasing Interconnect Speed

- Faster Hardware
 - Gigabit Ethernet; not Fast
 - Infiniband, even if IP over IB
- NIC settings
 - Duplex Mode
 - Highest Top Bit Rate
- TCP Settings
 - Flow Control Settings
 - Network Interrupts for CPU
 - Socket Receive Buffer
- LAN Planning
 - Private LANs
 - Collision Domains

High Speed Interconnects

- Oracle will support RDS over Infiniband
- <http://oss.oracle.com/projects/rds/>
- On 10 Gig Ethernet as well

In summary: Planning

- Adequate CPU, Network, Memory
- Sequences – cache, noorder
- Tablespaces read only
- Un-compact small hot tables
- Keep undo and redo on fastest disks
- Avoid full table scans of large tables
- Avoid DDLs and unnamed PL/SQL blocks

In summary: Diagnosis

- Start with AWR
- Identify symptoms and assign causes
- Don't get fooled by "gc" waits as interconnect issues
- Find the correlation between "dropped" packets in network, CPU issues from sar and "gc buffer lost" in sysstat reports.

Thank You!

Download from:

prolignce.com/downloads.html