
RAC or Not, Here I come!

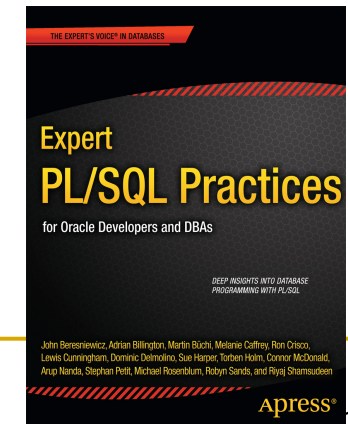
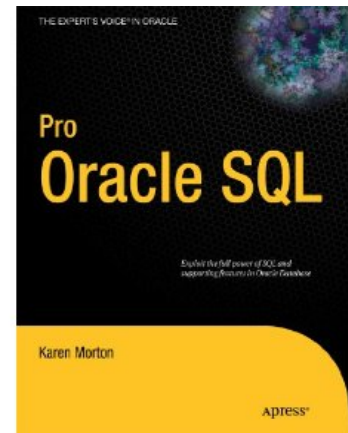
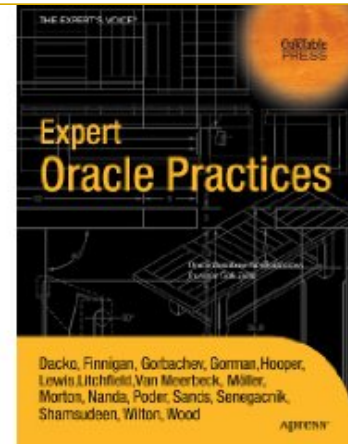
By
Riyaj Shamsudeen



Who am I?



- 18 years using Oracle products/DBA
- OakTable member
- Oracle ACE
- Certified DBA versions 7.0,7.3,8,8i,9i &10g
- Specializes in RAC, performance tuning, Internals and E-business suite
- Chief DBA with OraInternals
- Co-author of “Expert Oracle Practices” ‘2009
- Co-author of “Pro Oracle SQL” ‘2010
- Email: rshamsud@orainternals.com
- Blog : orainternals.wordpress.com
- URL: www.orainternals.com



Disclaimer

These slides and materials represent the work and opinions of the author and do not constitute official positions of my current or past employer or any other organization. This material has been peer reviewed, but author assume no responsibility whatsoever for the test cases.

If you corrupt your databases by running my scripts, you are solely responsible for that.

This material should not be reproduced or used without the authors' written permission.

Agenda

- **Considerations for conversion to RAC**
- Not so good reason to RAC
- Critical elements for successful conversion
 - Hardware setup
 - Network setup
 - ASM setup
 - Clusterware configuration
 - SGA configuration
- Extended RAC configuration
- Avoiding few pitfalls during conversion
- Performance management in RAC environment

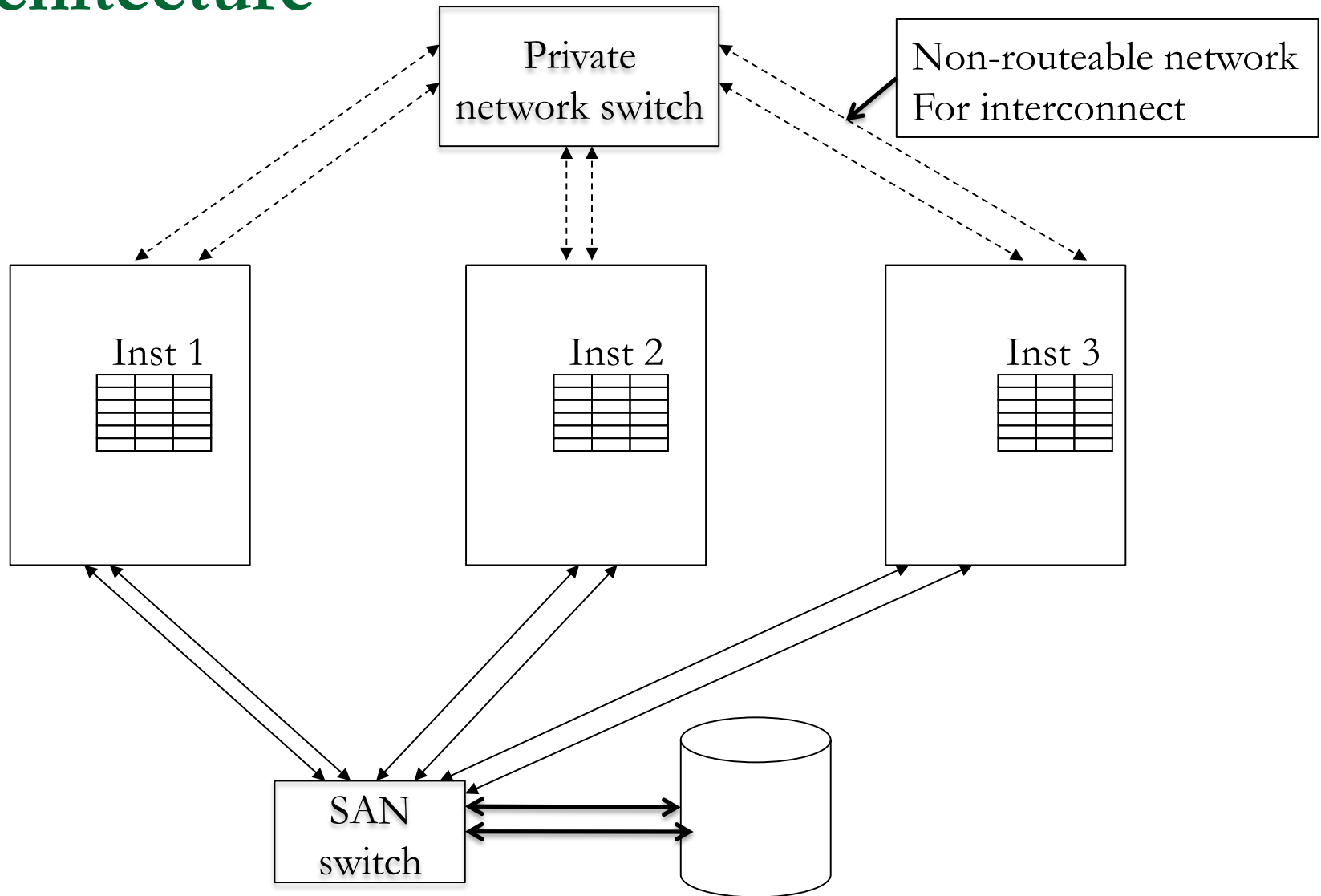
Good reasons

- Hardware fault tolerance
- Workload segregation
- Application affinity
- To manage excessive redo generation
- To avoid SMP bottlenecks

Hardware faults

- RAC protects from non-shared hardware failures.
- For example, CPU board failure in a node doesn't affect other node availability.
- But, failure in interconnect hardware can still cause cluster wide failures, unless interconnect path is fault-tolerant.
- Path to Storage also must be fault-tolerant.

Architecture



Workload segregation

- If you are planning to (or able to) segregate the workload to different nodes, then RAC may be a good option.
- Long running reports generating huge amount of I/O (with higher percent of single block I/O) can pollute the buffer cache, causing performance issues to critical parts of the application.
- For example, separation of OLTP and RPT to separate instances.
- Of course, you should consider Active Data Guard for off-loading Reporting activity.

Application affinity

- Application node affinity is a great way to improve performance in RAC databases.
- For example, if you have three applications, say PO, FIN, and SC, running in the same database, then consider node affinity.
- Node affinity should also translate in to segment level affinity.
- Say, if the application PO is accessing mostly *PO* tables and the application SC accessing mostly *SC* tables, then the node affinity might be helpful.

RAC tax

- SGA between the RAC instances is shared among the instances.
- Access and Changes to database blocks needs global co-ordination.
- GCS messages and GES messages need to be exchanged between the instances before blocks can be accessed.
- This global co-ordination is RAC tax, overhead due to RAC.
- Dynamic ReMastering (DRM), Application node affinity are some ways this problem can be minimized.
- Shared cache (and so effective reduction in I/O) compensates little for the RAC tax.

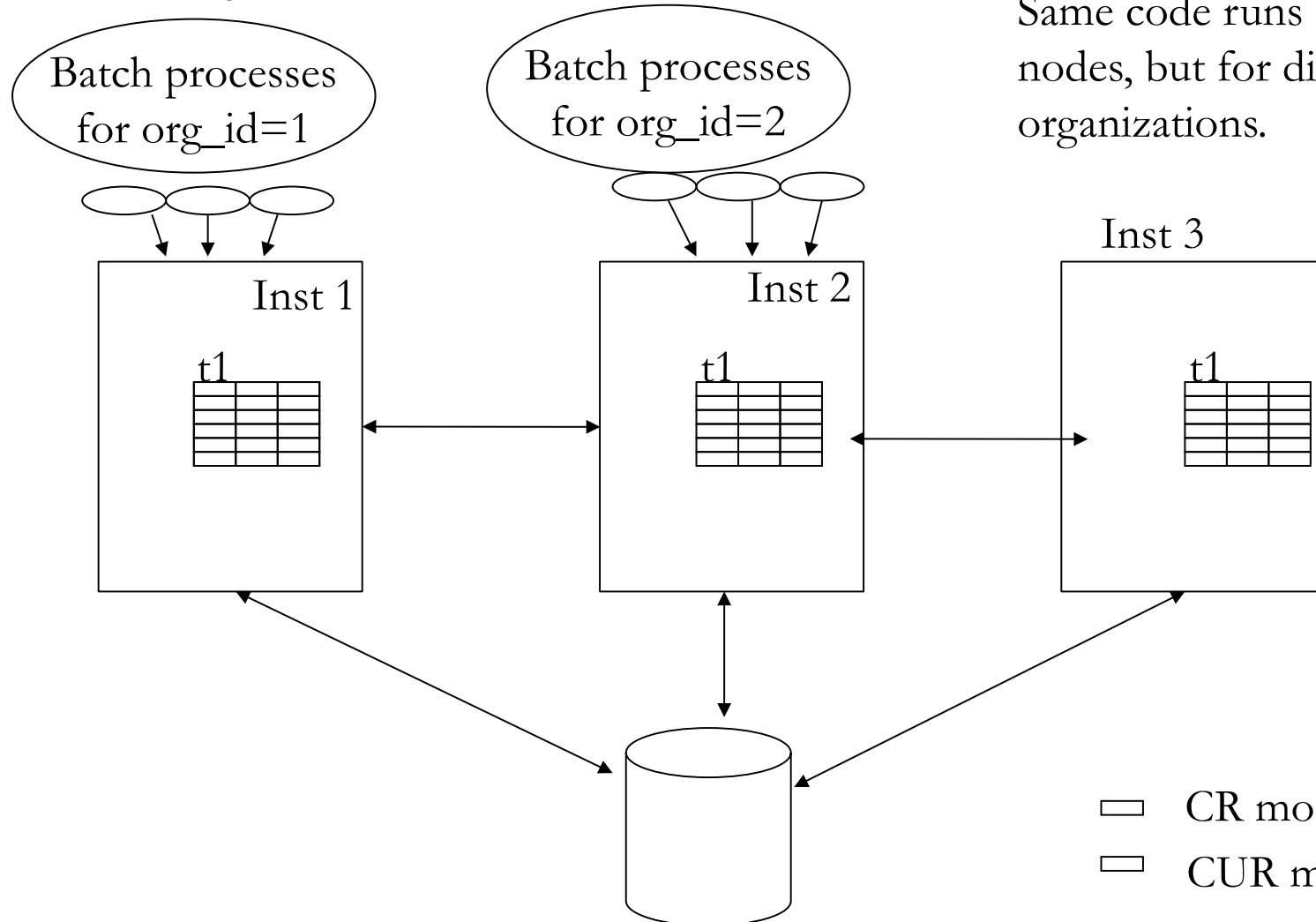
Are GCS waits cheap?

- Typical Global cache waits for 1 block transfer is 1-3ms for a decent hardware setup.
- Typical Single block reads are 2-4ms for decent hardware.
- But typical access to the buffer in the local SGA is in nano seconds (10-100ns).
- So, higher global cache waits still introduces latency comparable to I/O based latencies.

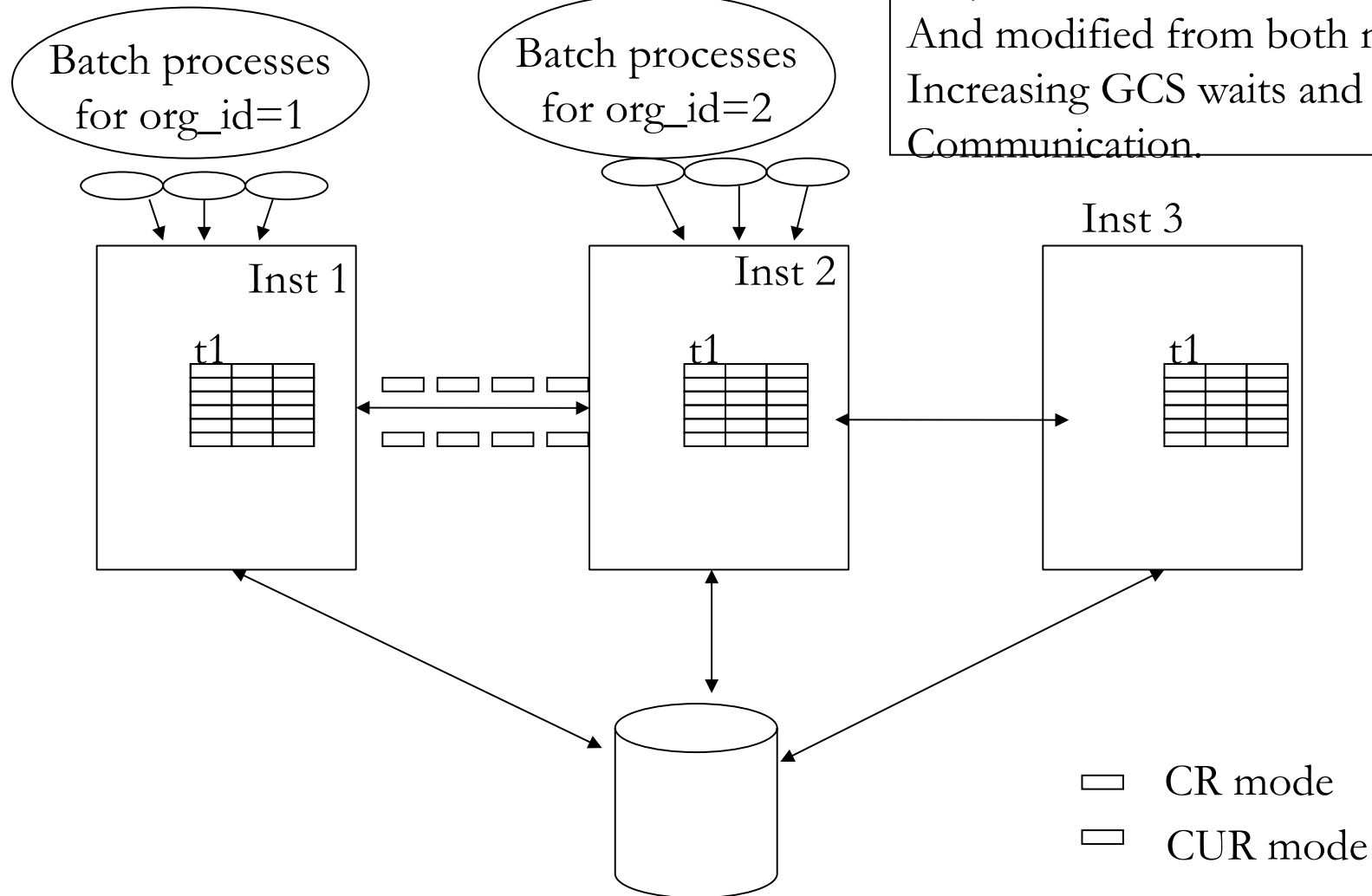
Example of bad affinity design.

- Workload segregation should be at the physical segment level, not just logical level.
- A batch process was designed to accept `organization_id` as an argument, and spawns multiple threads for each organization.
- This is multi-node RAC cluster and so
 - Batch process was started in node 1 for `org_id=1`.
 - Same batch process was started in node 2 for `org_id=2`.
- But batch code accesses just single set of tables and they are not partitioned.

Affinity?



Affinity?



What happened?

- Enormous increase in Global cache waits!

(5 minutes statspack)

Top 5 Timed Events

~~~~~

| Event                   | waits   | Time (s) | % Total<br>Ela Time |
|-------------------------|---------|----------|---------------------|
| -----                   | -----   | -----    | -----               |
| CPU time                |         | 3,901    | 46.17               |
| PL/SQL lock timer       | 80      | 2,316    | 27.42               |
| global cache cr request | 123,725 | 670      | 7.93                |
| global cache null to x  | 13,551  | 446      | 5.28                |
| buffer busy global CR   | 12,383  | 215      | 2.54                |

- If this has been a non-RAC, access to buffers would be in terms of nano seconds instead of milli-seconds.
- Point is that since these batch processes are accessing same physical blocks, they need to be grouped at physical level and scheduled to run from one node.
- Proper partitioning will help. But pay attention to Global indexes.

# What happened?

- Trace file analysis also shows excessive global cache waits.

Elapsed times include waiting on following events:

| Event waited on                     | Times<br>Waited | Max. wait | Total waited |
|-------------------------------------|-----------------|-----------|--------------|
| log file sync                       | 13              | 0.08      | 0.28         |
| latch free                          | 2185            | 0.17      | 32.76        |
| global cache cr request             | 21719           | 0.31      | 71.18        |
| db file sequential read             | 50              | 0.04      | 0.33         |
| global cache s to x                 | 791             | 0.06      | 5.31         |
| row cache lock                      | 113             | 0.02      | 0.41         |
| global cache null to x              | 4881            | 0.30      | 97.41        |
| buffer busy global CR               | 3306            | 0.23      | 15.26        |
| global cache open x                 | 903             | 0.06      | 5.83         |
| buffer busy waits                   | 1462            | 0.98      | 17.90        |
| global cache busy                   | 644             | 0.98      | 10.36        |
| buffer deadlock                     | 224             | 0.02      | 0.07         |
| global cache null to s              | 89              | 0.15      | 0.77         |
| buffer busy global cache            | 1066            | 0.31      | 27.62        |
| enqueue                             | 302             | 0.17      | 2.61         |
| KJC: wait for msg sends to complete | 102             | 0.00      | 0.00         |
| global cache open s                 | 62              | 0.01      | 0.13         |
| cr request retry                    | 4               | 0.00      | 0.00         |

---

# Redo

- Each instance has its own redo thread and LGWR process.
- If your application generates huge amount of redo, and if single instance LGWR can not handle the load, RAC might be a solution to effectively scale up LGWR throughput.
- For example, by converting to a 3 node cluster and balancing application workload, you can increase LGWR throughput, approximately by 3.
- Still, this doesn't solve the problem completely, if excessive redo generation is in conjunction with excessive commits.

---

# SMP bottleneck

- Access to memory and System bus becomes a bottleneck, in SMP architecture.
- Increasing number of CPUs in an SMP architecture doesn't scale linearly.
- Big Iron machines now uses NUMA architecture to alleviate this problem.
- If you can't afford big iron machines to increase scalability, RAC is a good option to consider.

---

# Agenda

- Considerations for conversion to RAC
- **Not so good reasons to RAC**
- Critical elements for successful conversion
  - Hardware setup
  - Network setup
  - ASM setup
  - Clusterware configuration
  - SGA configuration
- Extended RAC configuration
- Avoiding few pitfalls during conversion
- Performance management in RAC environment

---

## Not-So-Good reasons

- General Performance improvements
- To combat poor application design and coding practices
- RAC as a stand-alone Disaster Recovery solution
- To maximize use of hardware
- Stretch cluster to enhance hardware usage

---

# General performance improvements

- Generally, you shouldn't expect performance improvements, by just conversion to RAC.
- RAC tax kicks in. While Dynamic Remastering feature reduces the RAC tax, it doesn't completely eliminate the problem.
- While increased shared cache can improve the performance of the application, it comes with the cost of global co-ordination.
- Global cache latencies are in the same scale of Single Block I/O. Remote buffer access is still costly.

---

# Application design

- If the application doesn't scale well in a Single instance, then it probably perform poorly in RAC environment.
- You can't solve bad application design with RAC.
- Yes, it has been tried and failed ☹️
- Application patterns not performing well in RAC
  - Excessive DDL statements and Truncate statements.
  - Excessive Parsing.
  - Inefficiently written SQL statements and programs.
  - Uncached sequences etc.

---

# Availability

- RAC (non-Stretch ) provides hardware level fault tolerance.
- RAC + Data Guard is a good disaster recovery solution, but RAC alone is not a good DR solution.
- RAC as a DR solution does not protect from site disaster, human errors, or corruption (Both hardware and Software corruption).

---

# Application Anti-patterns

- Excessive truncates
- Excessive DDL
- Sequence Usage
- Excessive Parsing
- Pipes
- Excessive inserts with unique or localized keys

---

# Excessive Truncates

- Truncate of a table will need an object checkpoint from all instances.
- While the object checkpoint have greatly improved the performance of checkpoints, still it can cause latencies.
- Use Global Temporary tables for truly temporary data.
- Data in Global temporary tables are local to a session and so does not suffer from ill effects.

---

# DDL

- If your application generates enormous amount of DDL statements, application performance worsens after RAC conversion.
- DDLs also update 'row cache' and can lead to excessive 'DFS Lock Handle' waits.
- Library cache locks and Library cache pins must be broken during DDL execution.
- Library cache locks and pins are also global resources and can lead to massive waits in the cluster during excessive DDL execution.

---

# Sequence usage

- Application logic requiring sequences with no gap, can cause issues after conversion to RAC.
- If you require ordered values, it can be achieved by using sequences defined as ORDER and CACHE.
- Sequences defined as nocache can lead to major performance issues, even hung database.

# Cache and Order

- If ordering is a must, then use ORDER and cache attributes for sequences:

```
create sequence test_seq order cache 1000;
```

- With 'cache' and 'order', RAC is using GES layer to synchronize the sequence values (at least from 10g onwards).

Elapsed times include waiting on following events:

| Event waited on        | Times<br>waited | Max. wait | Total waited |
|------------------------|-----------------|-----------|--------------|
| -----                  | -----           | -----     | -----        |
| row cache lock         | 24              | 0.00      | 0.02         |
| DFS lock handle        | 3430            | 0.03      | 2.94         |
| gc current block 2-way | 13              | 0.21      | 0.24         |
| gc cr block 2-way      | 4               | 0.00      | 0.00         |

Demo: demo\_sequences\_01.sql

---

# Pipes

- Application extensively using pipes might need code changes to support RAC.
- Pipes are instance specific and can not be accessed remotely.
- If the sender and receivers of a pipe connect to different instance, then that might break application functionality.
- Use Queues instead of pipes or programs must be pinned to an instance.
- Pinning a program to an instance is a concern during failover as the program might not correctly failover.

---

# Agenda

- Considerations for conversion to RAC
- Fallacies – RAC conversion
- **Critical elements for successful conversion**
  - Hardware setup
  - Network setup
  - ASM setup
  - Clusterware configuration
  - SGA configuration
- Extended RAC configuration
- Avoiding few pitfalls during conversion
- Performance management in RAC environment

---

# Hardware

- Fault tolerant hardware is a key to successful RAC implementation.
- Use a multipathing solution to avoid loss of access to disks rebooting the server.
- Multiple voting disks is a must. Odd number of voting disks and 3 is a good place to start.
- Remember that a node must have access to at least 50% of visibility to the voting disks to survive.
- LGWR performance is critical for Global cache performance. Global cache transmission requires a log flush sync for Current blocks and “busy” CR blocks.

---

# Private Interconnect

- Use teaming, bonding or aggregation to bond network cards in to one logical entity [ or 11gR2 haip feature].
- Loss of one network cards should not affect heart beats between the nodes.
- Use Jumbo frame only if complete path (routers and other network hardware) supports Jumbo frames.
- Use MTU of 9000 for jumbo frames and other MTU sizes are not really tested in most cases.
- Jumbo frames decrease CPU usage as the need for assembly and disassembly decreases. But, network hardware must be designed to support Jumbo frames.
- Keep the IP addresses private and non-routable.

---

# ASM Setup

- Use ASM for storage. Avoid third party file systems, if you can.
- Even specialized file systems doesn't deliver enough performance improvement to complicate the tech stack.
- Just two disk groups, one for FRA and other for the database files are usually good enough.
- Even if there are multiple databases in the same ASM, just two disk groups is a good idea.
- Only if you have tiered storage within the same ASM, consider using multiple disk groups to match storage attributes.
- Keep voting disks and OCR in ASM avoiding the need for cluster file system or raw devices.

---

# Clusterware

- Keep odd number of voting disks.
- CSSD daemon should run in RT priority. You wouldn't want your cluster to restart in case of high CPU usage.
- Monitor alert log associated with the clusterware.
- If you are running third party clusterware also, make sure to integrate third party clusterware with CRS clusterware.
- With multiple cluster daemons in the play, problem analysis becomes quite cumbersome.

---

## SGA setup

- Surprisingly, many SGA sizes are too small for the workload.
- In RAC, if the buffers are in the local cache in a proper mode, GCS messages are completely avoided. So, keeping SGA big enough is a favorable action.
- If SGA is bigger, chances are that you can also avoid disk reads from other nodes too.
- But, bigger SGA can increase Global Cache traffic. Basically, moving the resource usage from disks to Network.
- Shared pool also has special sections for GCS/GES. Keep shared pool to couple of GBs to avoid unintended consequences.
- It is preferable to provide a lower bound for `shared_pool_size` and `db_cache_size` to avoid ill effects of ASMM.

---

# Agenda

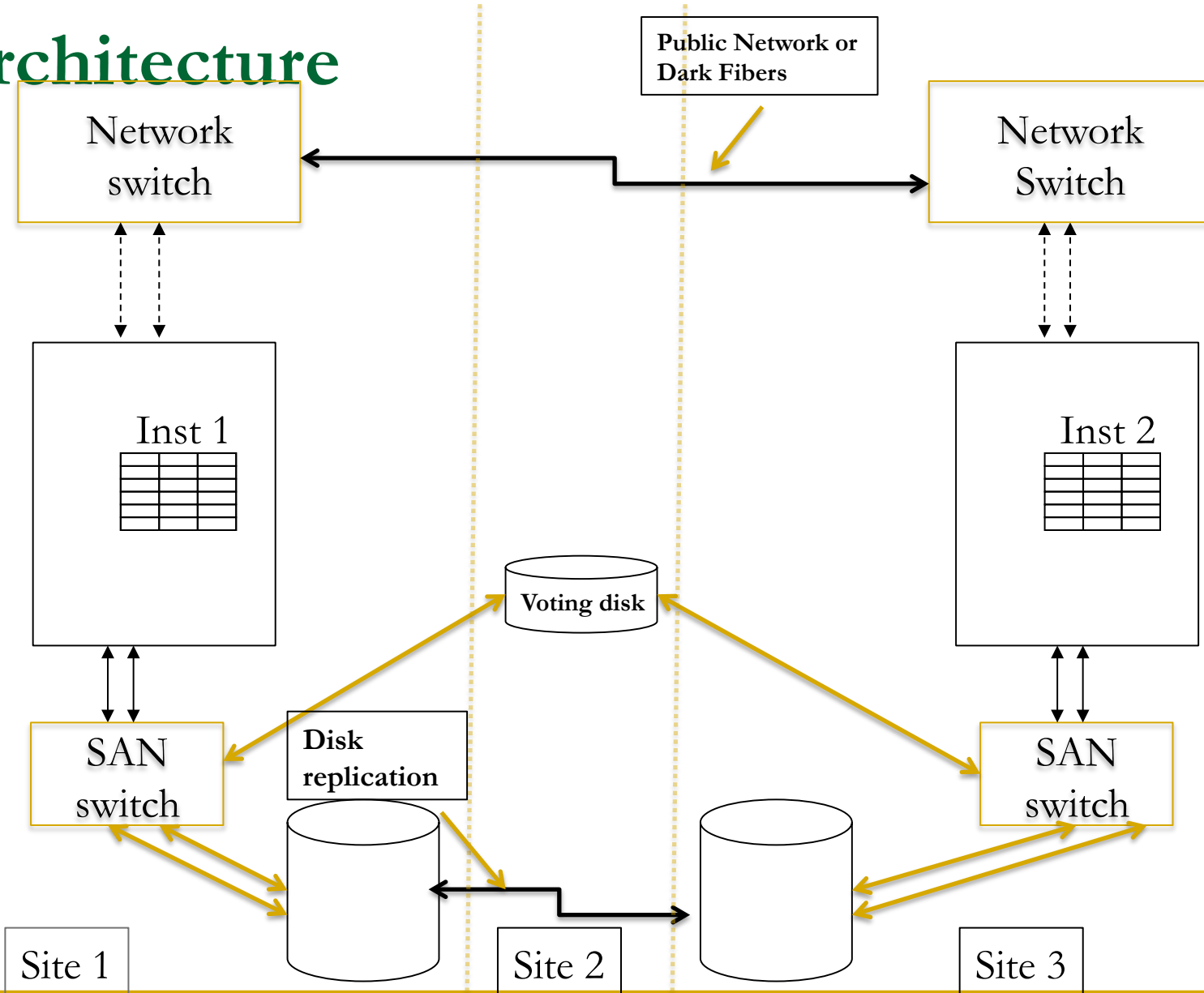
- Considerations for conversion to RAC
- Fallacies – RAC conversion
- Critical elements for successful conversion
  - Hardware setup
  - Network setup
  - ASM setup
  - Clusterware configuration
  - SGA configuration
- **Stretch RAC configuration**
- Avoiding few pitfalls during conversion
- Performance management in RAC environment

---

# Stretch RAC

- Stretch RAC means that nodes are physically separated by many miles.
- Both GES/GCS traffic latency can lead to application performance latency issues.
- Beware that as the distance between the nodes increases, latency increases and can lead to disastrous performance consequences.
- Dark fibers are must if you exceed few miles.
- Use third site for voting disks to detect split brain and communication failures between the sites.

# Architecture



---

## Preferred mirror read

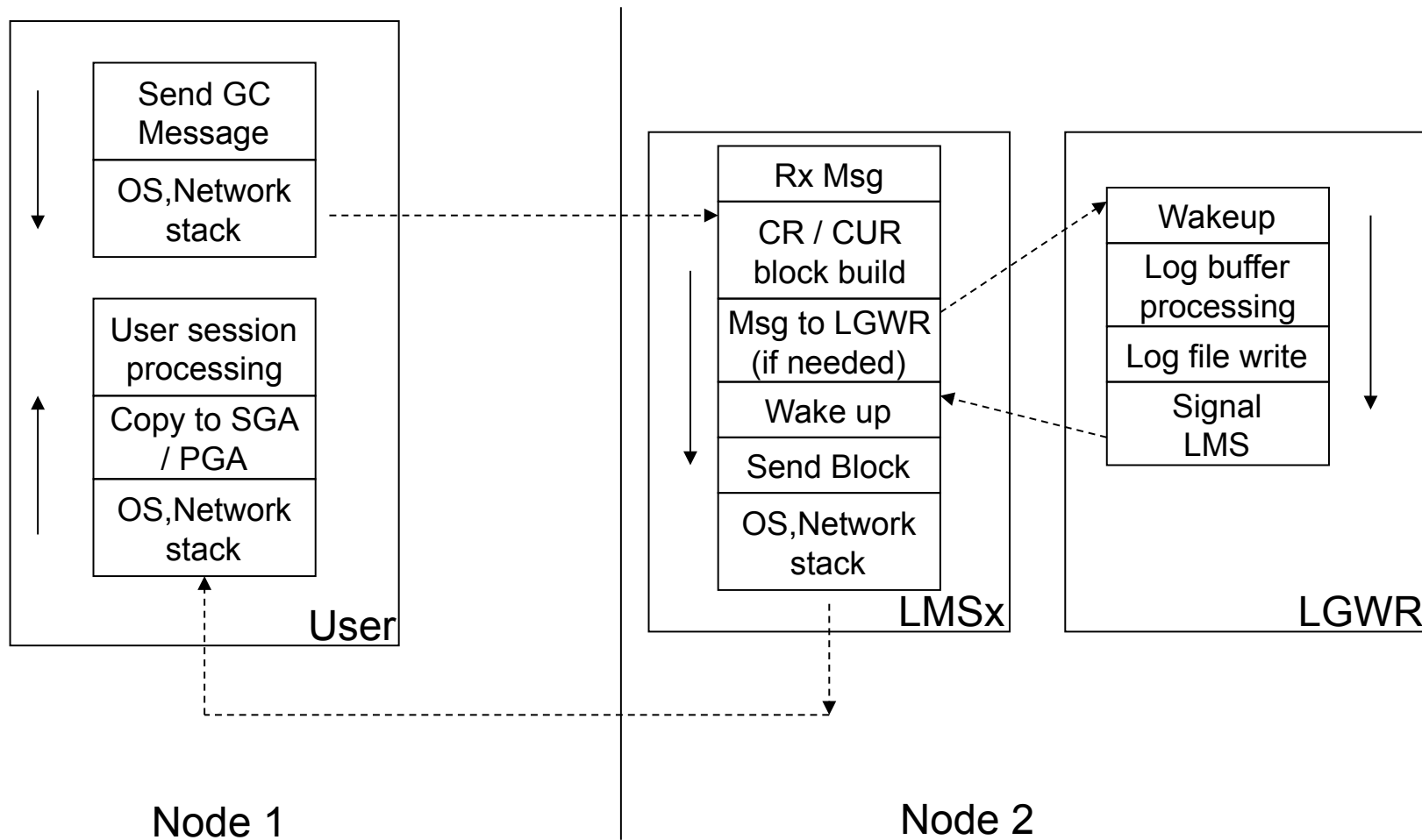
- Oracle 11g introduced preferred mirror read.
- ASM will try to read primary group extents.
- In extended RAC, since one of the failover group is local, it is preferable to read the extents from that local failover group.
- Parameter `asm_preferred_read_failure_groups` control the failover group to use for reading the extents.
- DB instance will read from local extents avoiding latency.

---

# Agenda

- Considerations for conversion to RAC
- Fallacies – RAC conversion
- Critical elements for successful conversion
  - Hardware setup
  - Network setup
  - ASM setup
  - Clusterware configuration
  - SGA configuration
- Stretch RAC configuration
- Performance analysis

# LMS Processing (over simplified)



---

# GC CR latency

- GC CR latency  $\approx$

Time spent in sending message to LMS +  
LMS processing (building blocks etc) +  
LGWR latency ( if any) +  
LMS send time +  
Wire latency

Averages can be misleading. Always review both total time and average to understand the issue.

# Breakdown latency

In this case, LGWR flush time  
Need to be reduced to tune latency.

Avg global cache cr block receive time (ms): 6.2

| Wait time              | Node 1 | Node 2 | Node 3 | Node 4 | Total |
|------------------------|--------|--------|--------|--------|-------|
| gc cr block build time | 402    | 199    | 100    | 227    | 1679  |
| Gc cr block flush time | 3016   | 870    | 978    | 2247   | 7111  |
| Gc cr block send time  | 375    | 188    | 87     | 265    | 1290  |

---

# GC CURRENT latency

- GC CUR latency  $\sim =$

Time spent in sending message to LMS +  
LMS processing : (Pin and build block) +  
LGWR latency: Log flush +  
Wire latency

Statistics : gc current block flush time  
gc current block pin time  
gc current block send time

---

# Caution

- Don't use gv\$views to find averages. Use AWR reports or custom scripts.
- gv\$views are aggregated data and persistent from the instance restart.
- For example this query can be misleading:

```
select b1.inst_id, b2.value "RECEIVED",  
       b1.value "RECEIVE TIME",  
       ((b1.value / b2.value) * 10) "AVG RECEIVE TIME (ms)"  
from gv$sysstat b1, gv$sysstat b2  
where b1.name = 'gc cr block receive time' and  
       b2.name = 'gc cr blocks received' and b1.inst_id = b2.inst_id
```

# gc\_traffic\_print.sql

- You can use my script to print global cache performance data for the past minute. Download from scripts archive:

[http://www.orainternals.com/scripts\\_rac1.php](http://www.orainternals.com/scripts_rac1.php)

| Inst | CR blocks Rx | CR time | CUR blocks Rx | CUR time | CR blocks Tx | CUR blocks Tx | Tot blocks |
|------|--------------|---------|---------------|----------|--------------|---------------|------------|
| 1    | 40999        | 13.82   | 7827          | 4.82     | 25070        | 17855         | 91751      |
| 2    | 12471        | 5.85    | 8389          | 5.28     | 31269        | 9772          | 61901      |
| 3    | 28795        | 4.11    | 18065         | 3.97     | 28946        | 4248          | 80054      |
| 4    | 33105        | 4.54    | 12136         | 4.68     | 29517        | 13645         | 88403      |

- During the same time frame, output of the script from prior slide:

| INST_ID | RECEIVED  | RECEIVE TIME | AVG RECEIVE TIME (ms) |
|---------|-----------|--------------|-----------------------|
| 4       | 165602481 | 104243160    | 6.2947825             |
| 2       | 123971820 | 82993393     | 6.69453695            |
| 3       | 215681074 | 103170166    | 4.7834594             |
| 1       | 134814176 | 66663093     | 4.9448133             |

Very misleading!

## Review all nodes.

- It is important to review performance data from all the nodes.
- It is easy to create AWR reports from all nodes using my script:  
Refer awrrpt\_all\_gen.sql.
  - [ Don't forget that access to AWR report needs license ]
- Or use my script gc\_traffic\_processing.sql from my script archive.

Default collection period is 60 seconds.... Please wait for at least 60 seconds...

| Inst | CR blk TX | CR bld | CR fls tm | CR snd tm | CUR blk TX | CUR pin tm | CUR fls tm | CUR blk TX |
|------|-----------|--------|-----------|-----------|------------|------------|------------|------------|
| 2    | 67061     | .08    | .88       | .23       | 34909      | 1.62       | .2         | .23        |
| 3    | 38207     | .17    | 2.19      | .26       | 28303      | .61        | .08        | .26        |
| 4    | 72820     | .06    | 1.76      | .2        | 40578      | 1.76       | .24        | .19        |
| 5    | 84355     | .09    | 2.42      | .23       | 30717      | 2.69       | .44        | .25        |

---

# Place holder events

- Few events are place holder events such as:
  - gc cr request
  - gc cr multiblock request
  - gc current request
  - ...
- Sessions can be seen waiting for these wait events, but will not show up in AWR / ADDM reports.
- After sending the global cache block request, foreground process waits on these events.
- On receipt of the response, time is accounted for correct wait event.

---

## 2-way/3-way events

- As we saw in the prior section, there are 2-way and 3-way block transfer.
  - GC CR block 2-way
  - GC CR block 3-way
  - GC CURRENT block 2-way
  - GC CURRENT block 3-way
- Even if there are many instances, only three instances participate in a block transfer.
- But, flush messages can be sent to all instances in few cases.

---

# Gc grants

- Wait events 'gc cr grant 2-way' and 'gc current grant 2-way' indicates
  - Block is not in any cache
  - Permission granted to read from the disk.

```
WAIT #6: nam='gc cr grant 2-way' ela= 567 p1=295 p2=770871 p3=1  
obj#=5153800 tim=817052932927
```

```
WAIT #6: nam='db file sequential read' ela= 11003 file#=295 block#=770871  
blocks=1 obj#=5153800 tim=817052943998
```

---

# Congested..

- Congestion indicates that LMS processes were not able to service fast enough:
  - gc cr grant congested, gc current grant congested
  - gc cr block congested, gc current block congested
- Focus on LMS processes and usual culprits are load, SQL performance or longer CPU queue etc.

# Histogram

89.4% of these waits are Under 4ms.

- Averages can be misleading. Use `v$event_histogram` to understand true performance metrics.
- It is better to take snapshots of this data and compare the differences.

| INST_ID | EVENT             | WAIT_TIME_MILLI | WAIT_COUNT | THIS_PER | TOTAL_PER |
|---------|-------------------|-----------------|------------|----------|-----------|
| 1       | gc cr block 2-way | 1               | 466345     | .92      | .92       |
| 1       | gc cr block 2-way | 2               | 23863264   | 47.58    | 48.51     |
| 1       | gc cr block 2-way | 4               | 20543430   | 40.96    | 89.47     |
| 1       | gc cr block 2-way | 8               | 4921880    | 9.81     | 99.29     |
| 1       | gc cr block 2-way | 16              | 329769     | .65      | 99.95     |
| 1       | gc cr block 2-way | 32              | 17267      | .03      | 99.98     |
| 1       | gc cr block 2-way | 64              | 2876       | 0        | 99.99     |
| 1       | gc cr block 2-way | 128             | 1914       | 0        | 99.99     |
| 1       | gc cr block 2-way | 256             | 1483       | 0        | 99.99     |
| 1       | gc cr block 2-way | 512             | 618        | 0        | 99.99     |
| 1       | gc cr block 2-way | 1024            | 83         | 0        | 99.99     |
| 1       | gc cr block 2-way | 2048            | 4          | 0        | 99.99     |
| 1       | gc cr block 2-way | 4096            | 3          | 0        | 99.99     |
| 1       | gc cr block 2-way | 8192            | 5          | 0        | 99.99     |
| 1       | gc cr block 2-way | 16384           | 3          | 0        | 100       |

# GC event histograms

- Better yet, use my script `gc_event_histogram.sql` to understand current performance metrics.

Default collection period is sleep seconds. Please wait..

Enter value for event: gc cr block 2-way

Enter value for sleep: 60

| Inst id | Event             | wait time milli | wait cnt |
|---------|-------------------|-----------------|----------|
| 1       | gc cr block 2-way | 1               | 37       |
| 1       | gc cr block 2-way | 2               | 4277     |
| 1       | gc cr block 2-way | 4               | 5074     |
| 1       | gc cr block 2-way | 8               | 1410     |
| 1       | gc cr block 2-way | 16              | 89       |
| 1       | gc cr block 2-way | 32              | 1        |
| 1       | gc cr block 2-way | 64              | 0        |
| 1       | gc cr block 2-way | 128             | 0        |
| 1       | gc cr block 2-way | 256             | 0        |

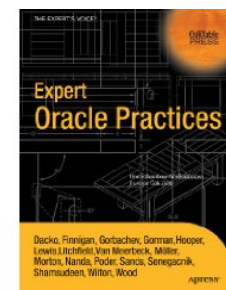
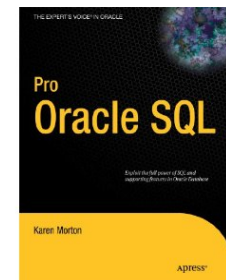
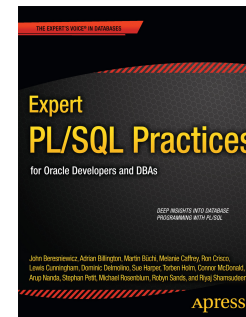
Thank you for attending!

If you like this presentation, you will love my  
upcoming intensive RAC webinar in June.

Watch for updates in:

[www.tanelpoder.com](http://www.tanelpoder.com)

[OraInternals.wordpress.com](http://OraInternals.wordpress.com)



---

# References

- Oracle support site. [Metalink.oracle.com](http://Metalink.oracle.com). Various documents
- Internal's guru Steve Adam's website  
[www.ixora.com.au](http://www.ixora.com.au)
- Jonathan Lewis' website  
[www.jlcomp.daemon.co.uk](http://www.jlcomp.daemon.co.uk)
- Julian Dyke's website  
[www.julian-dyke.com](http://www.julian-dyke.com)
- 'Oracle8i Internal Services for Waits, Latches, Locks, and Memory'  
by Steve Adams
- Tom Kyte's website  
[Asktom.oracle.com](http://Asktom.oracle.com)
- Blog: <http://orainternals.wordpress.com>