
Let me RAC your world..

By
Riyaj Shamsudeen



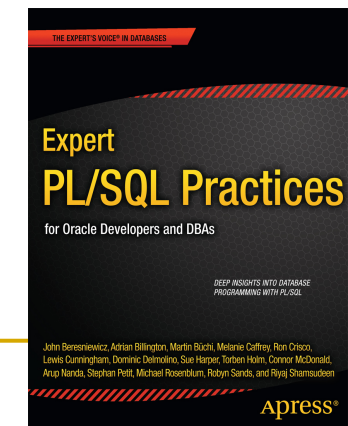
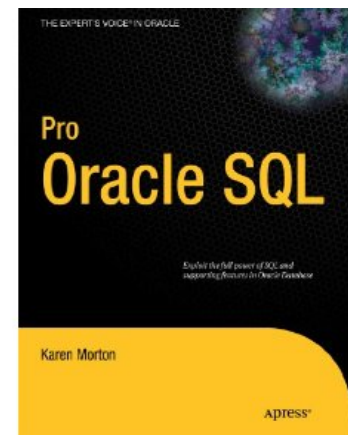
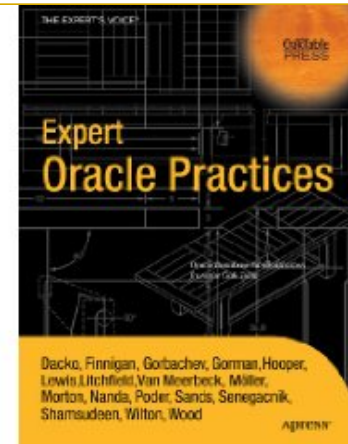
Who am I?



ORACLE
ACE Director

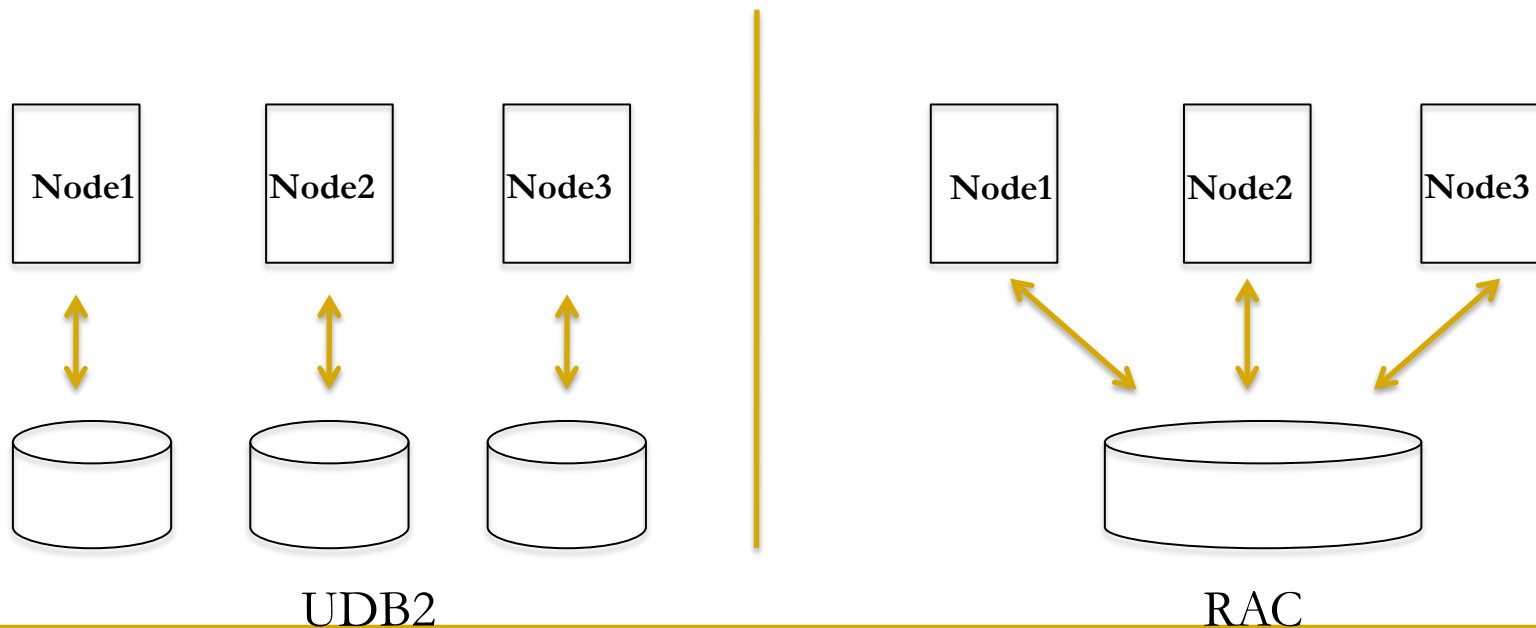


- 19 years using Oracle products/DBA
- OakTable member
- Oracle ACE Director
- Certified DBA versions 7.0,7.3,8,8i,9i &10g
- Specializes in RAC, performance tuning, Internals and E-business suite
- Chief DBA with OraInternals
- Co-author of “Expert Oracle Practices” ‘2009
- Co-author of “Pro Oracle SQL”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



Shared Nothing vs Shared Everything

- In a shared nothing architecture, obviously, nothing is shared. In a shared everything architecture everything is shared.
- Oracle RAC uses Shared everything architecture, where as DB2 uses shared nothing architecture.
- In RAC, loss of one node does not affect data availability.



Good reasons

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

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).

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

Clusterware

- Clusterware is an integral component of RAC architecture.
- Clusterware provides node membership, monitoring, and node management.
- Few daemons run with root ownership.
- Databases, listeners, ASM instances are simply resources for clusterware.
- These resources are managed by clusterware for high availability reasons.

Demo: clusterware daemons, commands etc

Application Anti-patterns

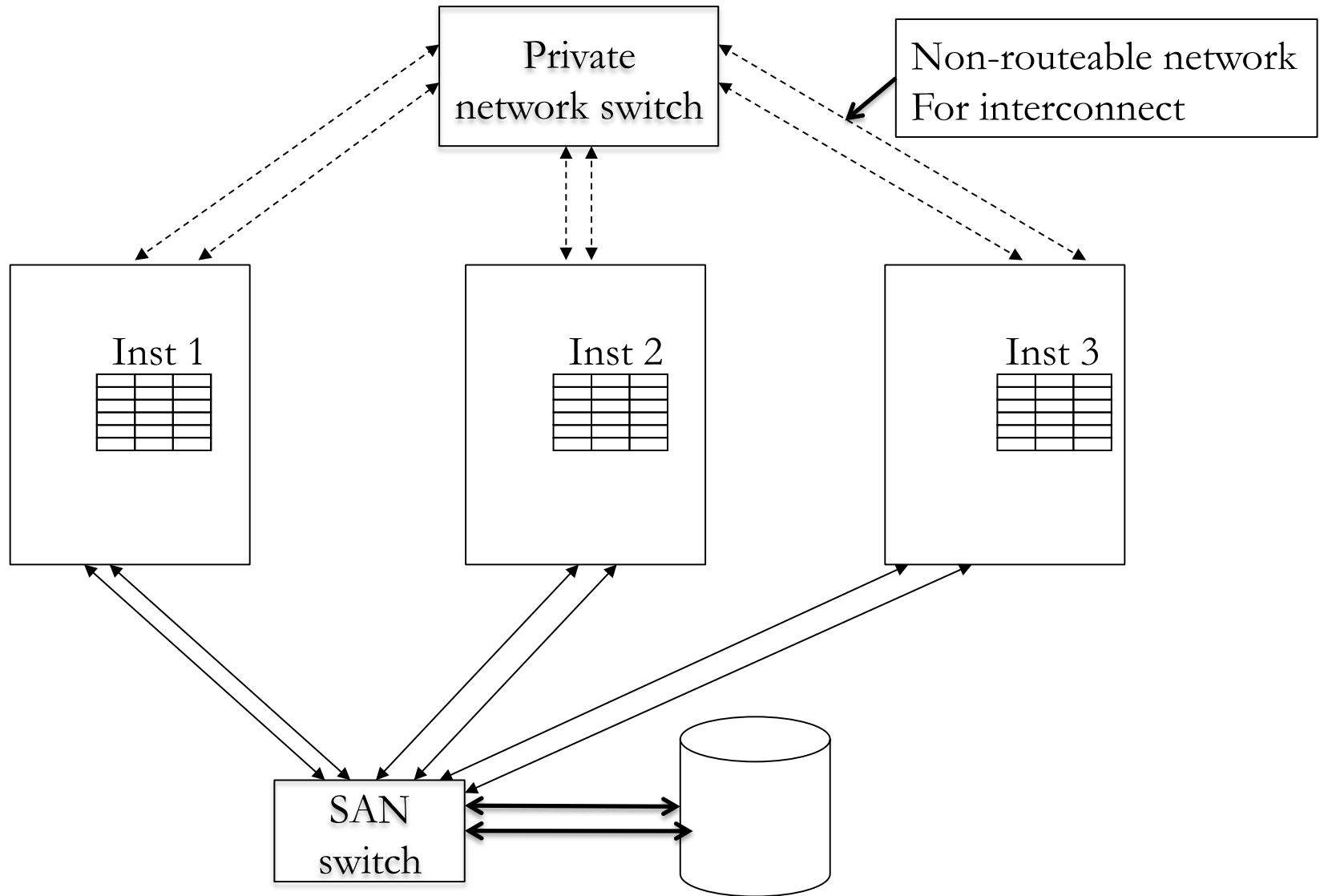
- Excessive truncates
- Excessive DDL
- Sequence Usage
- Excessive Parsing
- Pipes
- Excessive inserts with unique or localized keys

Instance vs Database

- Instance is defined as a set of memory structures (SGA) and processes.
- Database is defined as permanent structures such as data files.
- Instance is transient and the database is permanent.
- In a single instance database, there is a one-to-one relationship between an instance and a database.
- In RAC database, multiple instances can mount one database concurrently.
- Resources are globalized in RAC database.

Demo: instance, database etc.

Architecture



Redo, undo and temp

- Each instance has its own redo thread. You need to allocate at least a set of log file groups for a redo thread.
- Each instance uses its own undo tablespace. An undo tablespace must be assigned to each instance.
- TEMP tablespace is shared between the instances, but, we will see that sharing is at extent level later.
- Control files, data files are shared between the instances.
- Each instance has its own SGA.
- So, adding a new instance means that you need to add a redo thread, an undo tablespace, and possibly increase temporary tablespace etc.

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.

Initialization parameters

- There are two distinct class of parameters:
 - Parameters which must be the same in all instances
 - Parameters that can be different between the instances.
- `V$parameter.isinstance_modifiable` (11g) indicates whether the parameter must be alike or not.
- All parameters with `isinstance_modifiable=FALSE` must be alike in all instances. In 11gR2, 115 parameters must be alike and 230 parameters can be different.

Parameter changes

- To change parameter specific to an instance:

Alter system set <parameter>='value' scope=both sid='inst1';

- To change parameters in all instances:

Alter system set <parameter>='value' scope=both sid='*';

- If a parameter is specified with both * and instance_name, then the parameter with the instance_name has higher precedence.

- For example, with the parameter set below, inst1 will have 40G db_cache_size and other instances will have 38G db_cache_size:

```
inst1.db_cache_size = 40G;
```

```
*.db_cache_size = 38G;
```

Command syntax

`srvctl <action> <objects> <options>`

- *action* is the action to be taken, such as start, stop, enable, disable, relocate etc.
- *Object* is the type of object such as database, listener, vip etc.
- *Options* specifies the details about object itself such as database_name, instance_name, listener_name etc.
- *Options* also allows you to specify more options such as startup_mode etc.

Srvctl - database

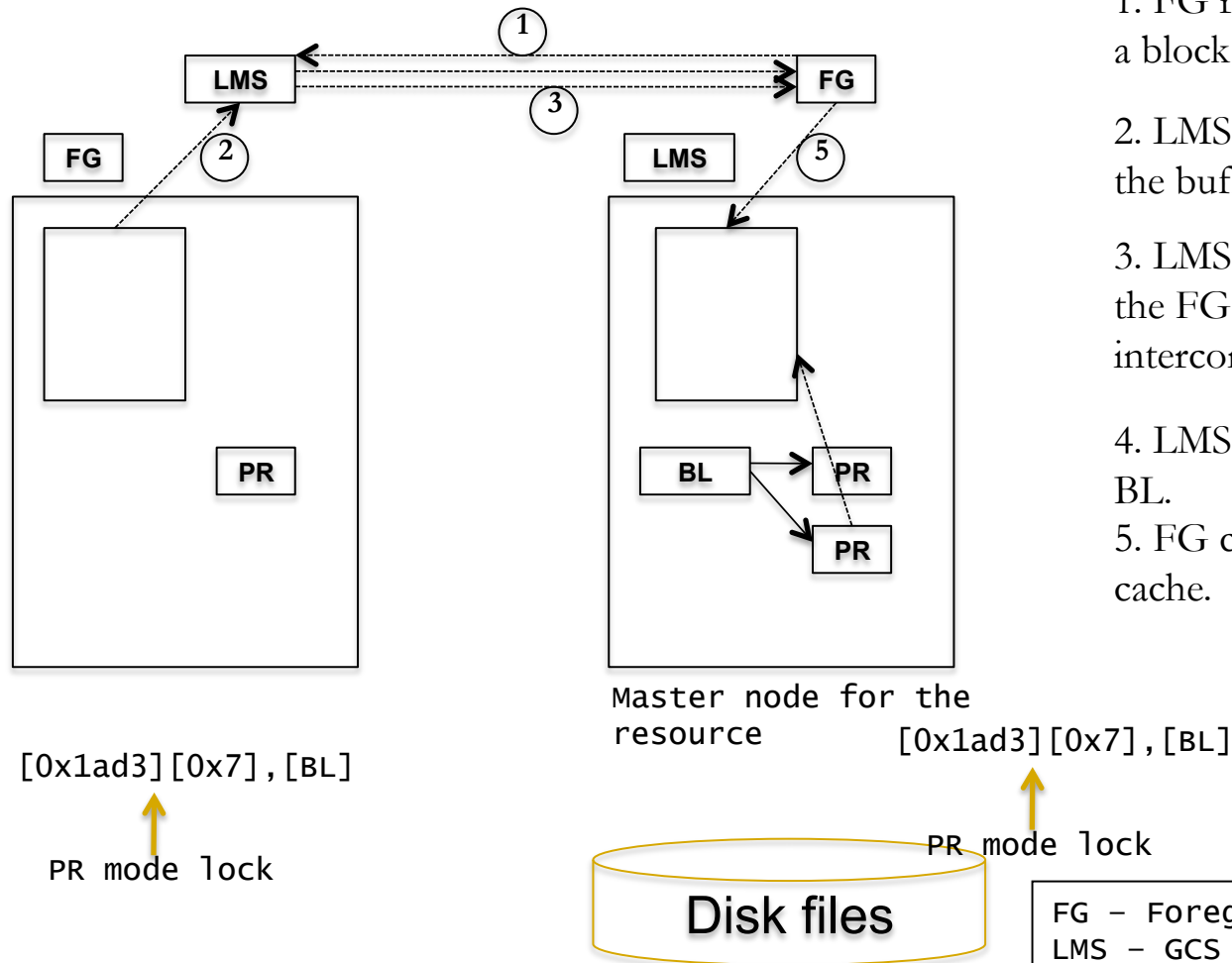
```
srvctl start database -d solrac -o open,pfile=initso1rac.ora
```

- Above command starts all instances configured with that database.
- You can specify various modes such as restrict, mount etc as option.
- Open mode is optional and is default. Database will use configured spfile if no pfile specified in the command line.

Cache coherency

- There are multiple buffer caches in an instance and Oracle RAC uses shared everything architecture.
- Cache coherency is the method by which consistency of the database is maintained.
- Only one instance can hold a block in current mode in exclusive mode and a block can be modified only if the block is held in exclusive current mode.
- There can be two pending transactions modifying the same block, but a block can only be held in exclusive mode in an instance.

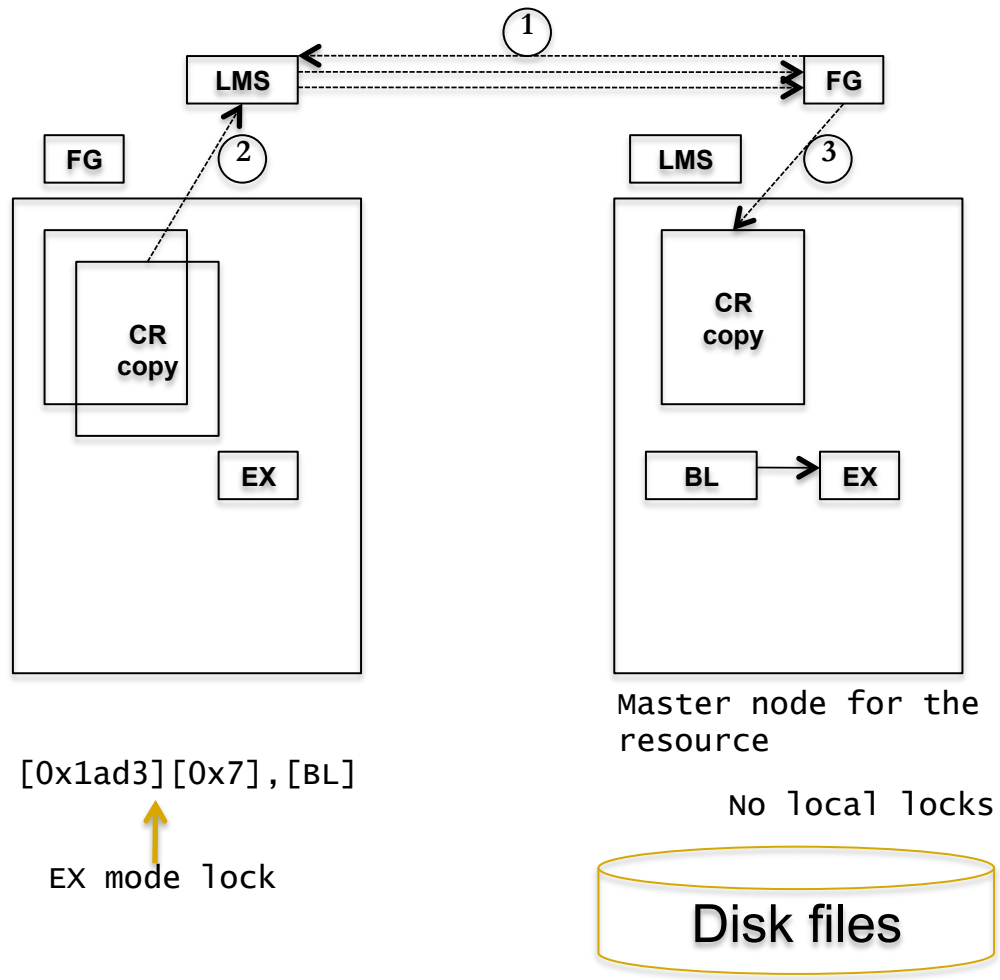
Cache fusion: Read-Read



1. FG requests LMS process for a block.
2. LMS process finds a block in the buffer cache.
3. LMS process sends a block to the FG process through interconnect.
4. LMS also sets up GRD for the BL.
5. FG copies the buffer to buffer cache.

FG - Foreground Process
 LMS - GCS Server process
 GRD - Global Resource Directory
 BL - Buffer Lock
 PR - Protected Read

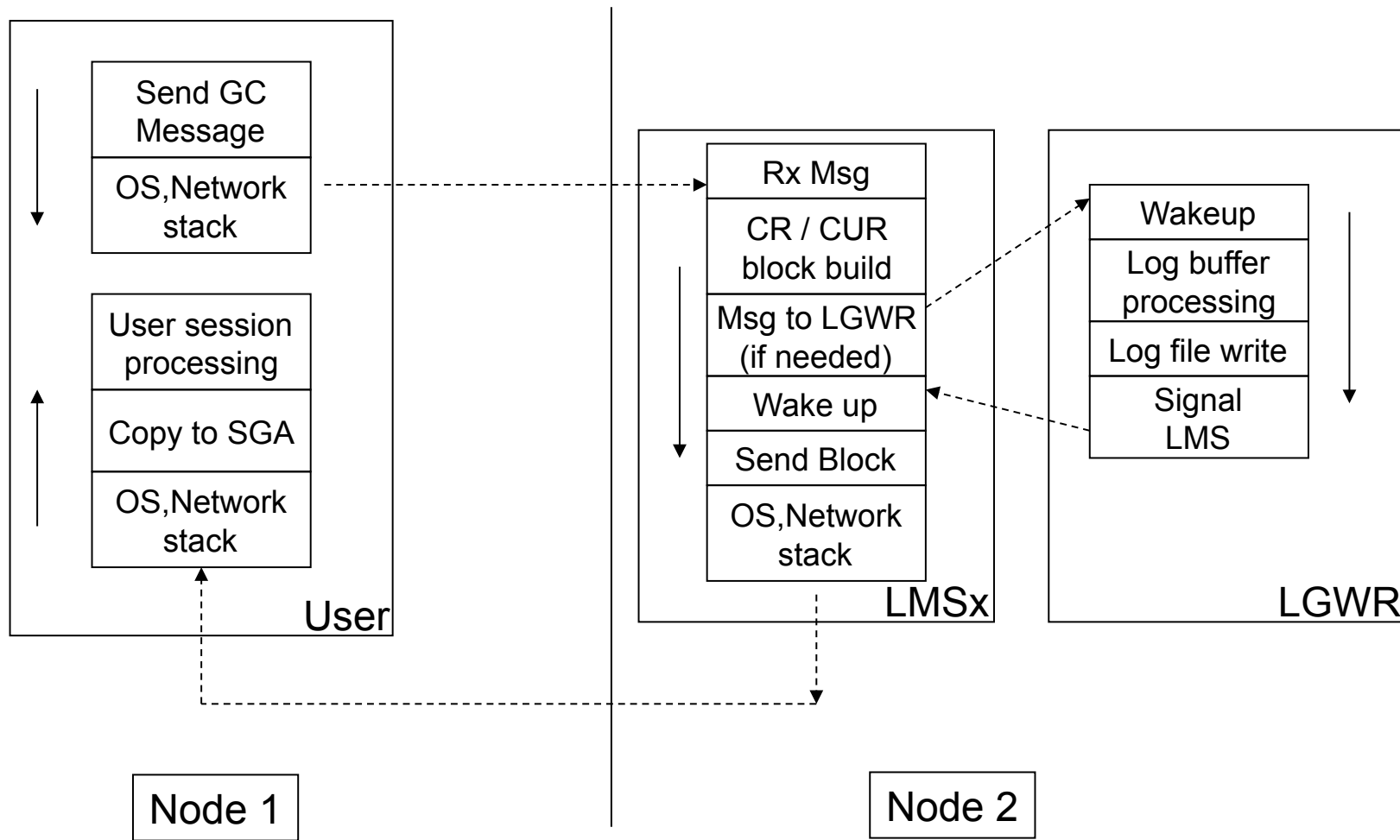
Cache fusion: Write-Read



In this case, FG asks for a Specific version of the block.

FG - Foreground Process
 LMD - Lock Manager Daemon
 GRD - Global Resource Directory

LMS Processing (over simplified)



LGWR is important

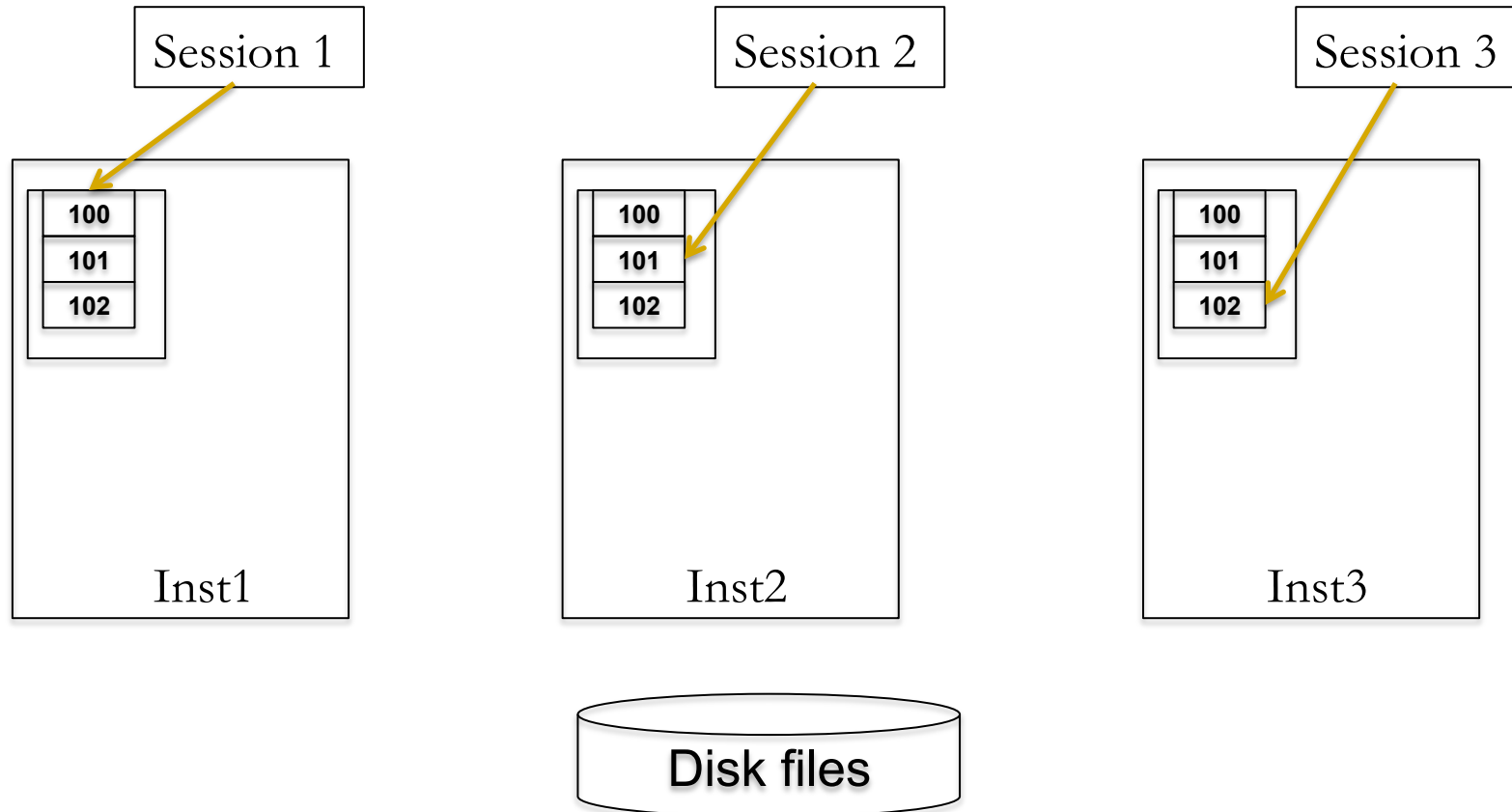
- So, if you think, LGWR performance is important in single instance, then it is ultra-important in RAC.
- If you have LGWR related performance issues, you can almost discard other waits as *symptoms*.
- It's a pity that LGWR does not run in RT mode (or even FX class).
- LMS processes runs in elevated priority, but LGWR does not run in elevated priority, classic priority-inversion!

Row level locking

- A key advantage of RAC is that it doesn't affect row level locking.
- We know that a row can not be modified by two sessions concurrently in a single instance database. This is true in RAC also.
- But, two sessions can modify rows in the same block, even from different instances.
- All Oracle features such as transaction integrity, transaction consistency, row level locking mechanism etc still maintained.

Row level locking

Three sessions are modifying three rows in *a* block concurrently.



Demo:demo_upd_rows_a.sql, demo_upd_rows_b.sql, demo_upd_rows_c.sql

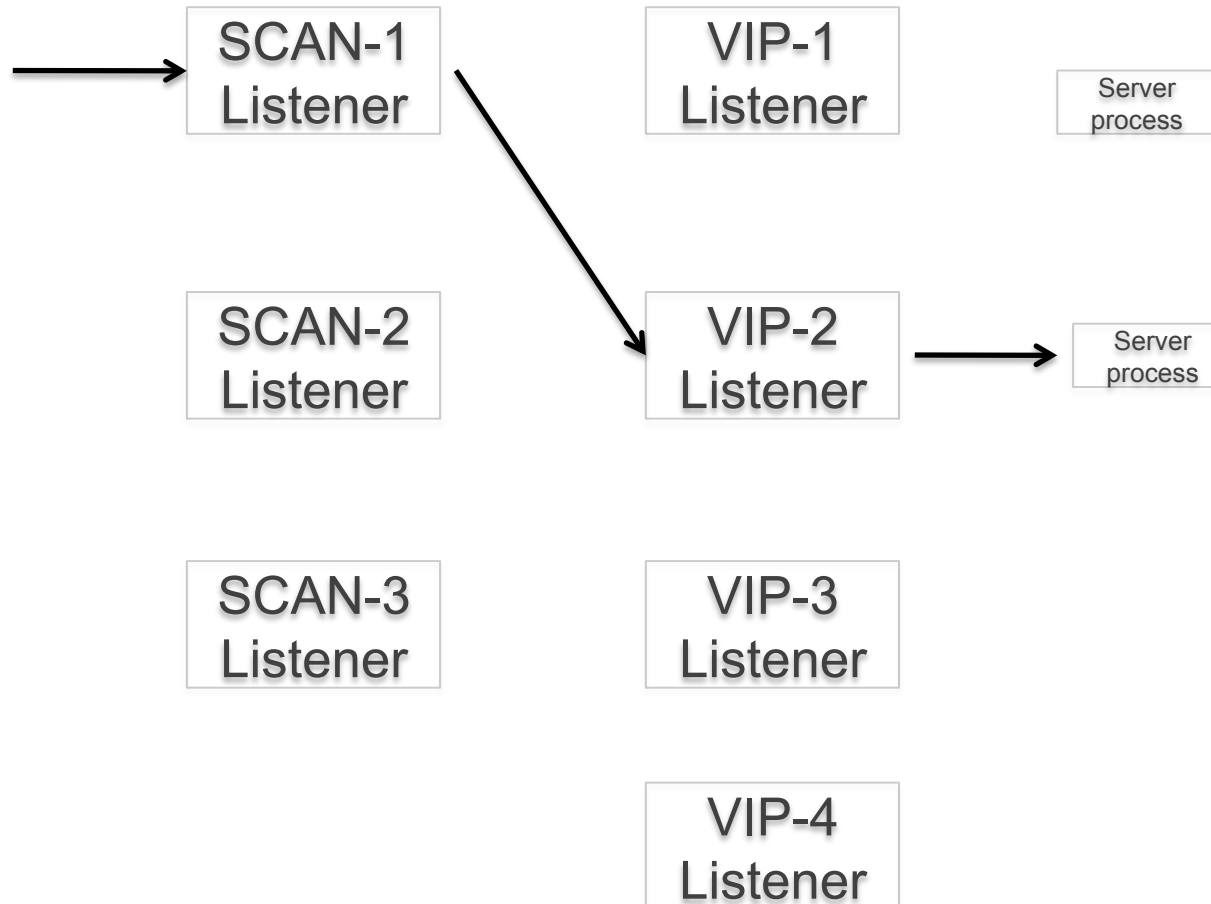
Load balancing & Failover

- Load balancing is a technique in which we try to utilize all nodes in a cluster equally.
- Failover is a technique in which session fails over to a different node at the onset of a node failure.
- Load balancing and Failover can be implemented together, or independent of each other.
- In most cases, both will be implemented together.

Overview

- There are many techniques available to implement failover and load balancing in various layers.
- Few techniques:
 - Client Side load balancing
 - Load balancing with SCAN IP
 - Server side load balancing
 - VIPs with client side load balancing
 - TAF – Transparent Application Failover
 - Services based load balancing and failover.
 - Combinations of above technique.

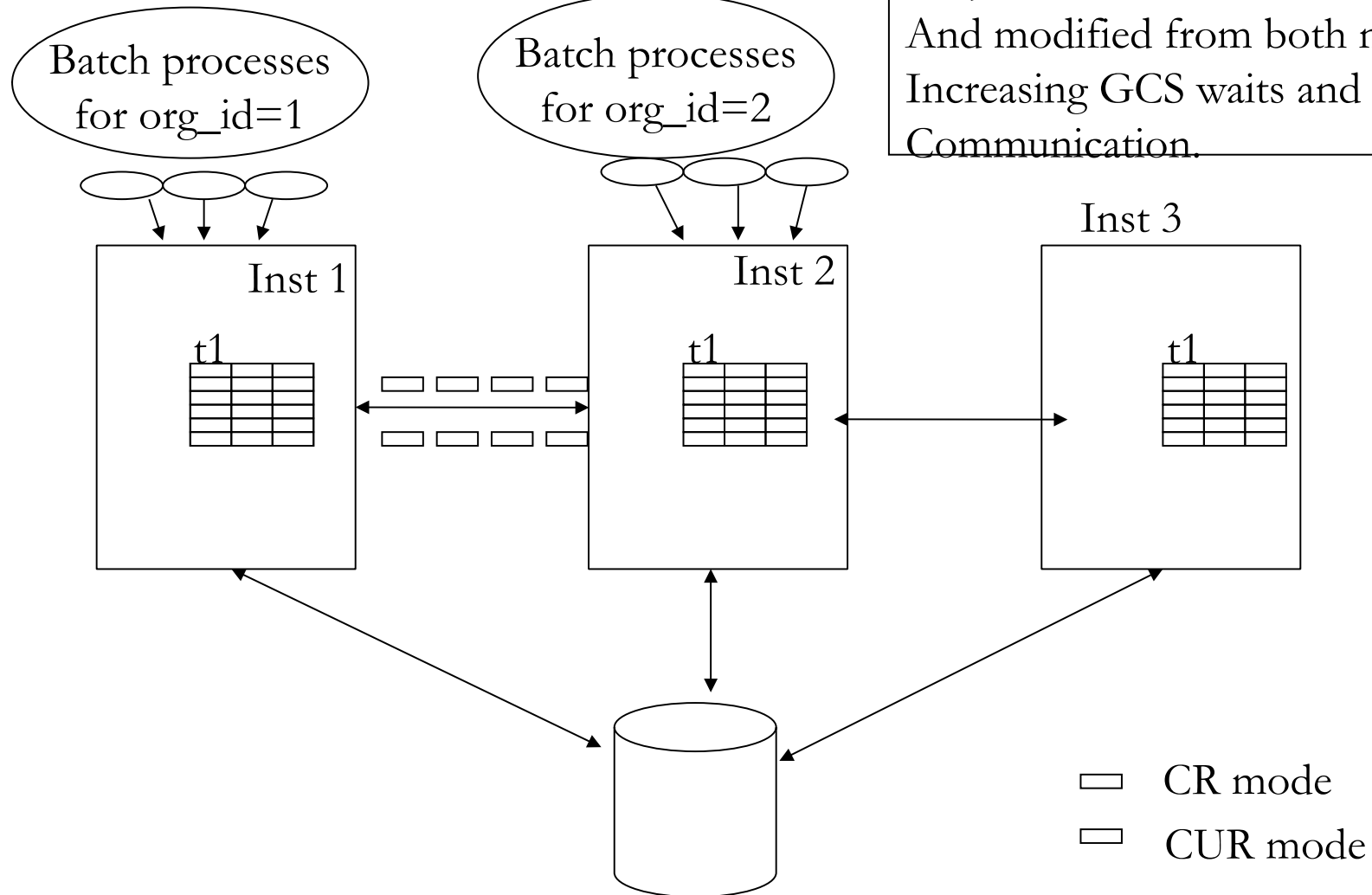
SCAN addresses



Affinity is still important!

- 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?



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.

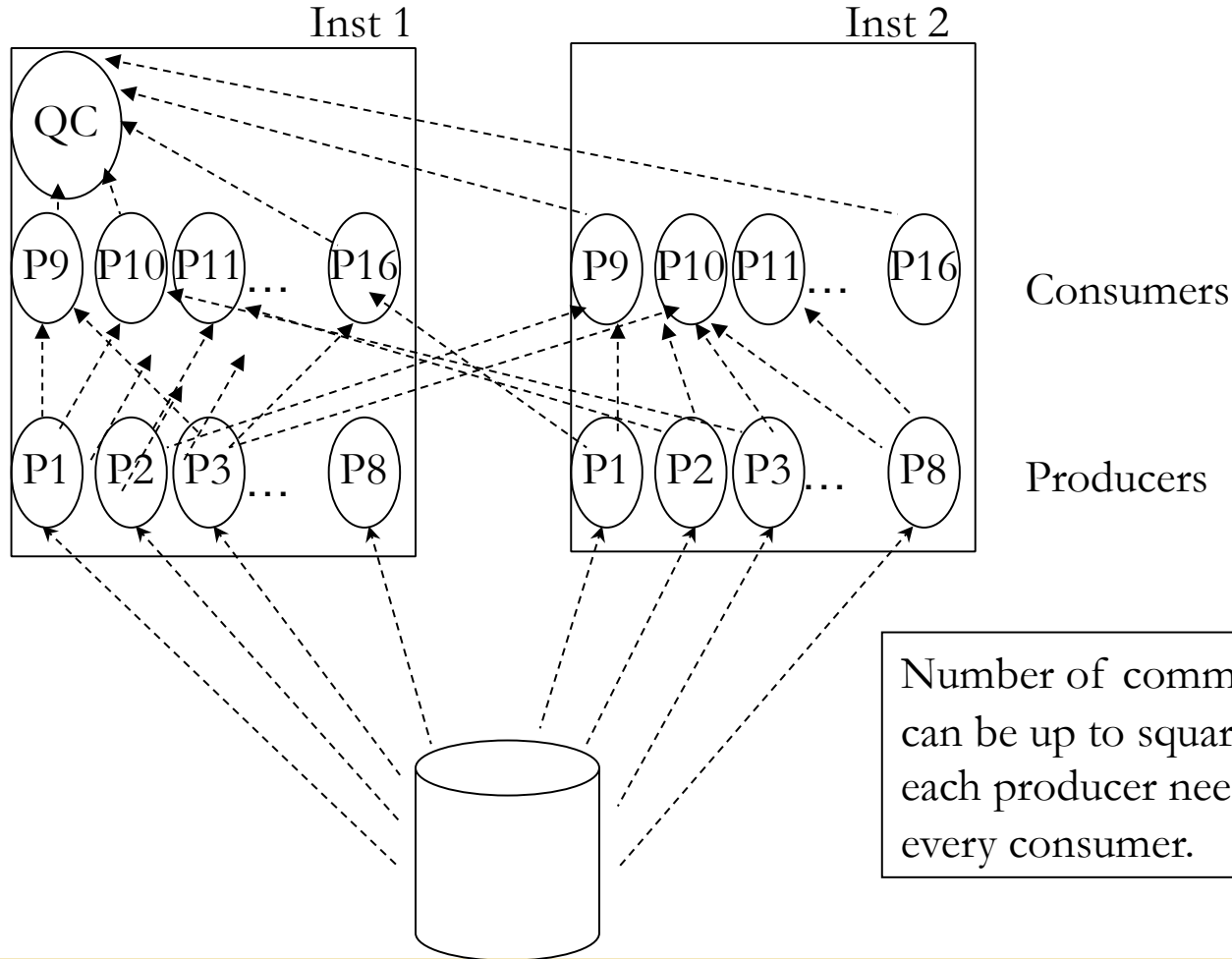
---

## PX: intra vs inter-instance

- *Intra*-instance PX operation: All slaves are allocated in the current instance.
- *Inter*-instance PX operation: Slaves are allocated from more than one instance.
- In intra-instance PX operation, Slaves communicate with each other, passing buffers between them and does not use interconnect.
- In inter-instance PX operation, slaves use interconnect to exchange buffers and messages, among the slave sets or Co-ordinator processes.

## Worst case scenario

A table data is broadcasted to every consumer using BROADCAST distribution method, during a join operation.

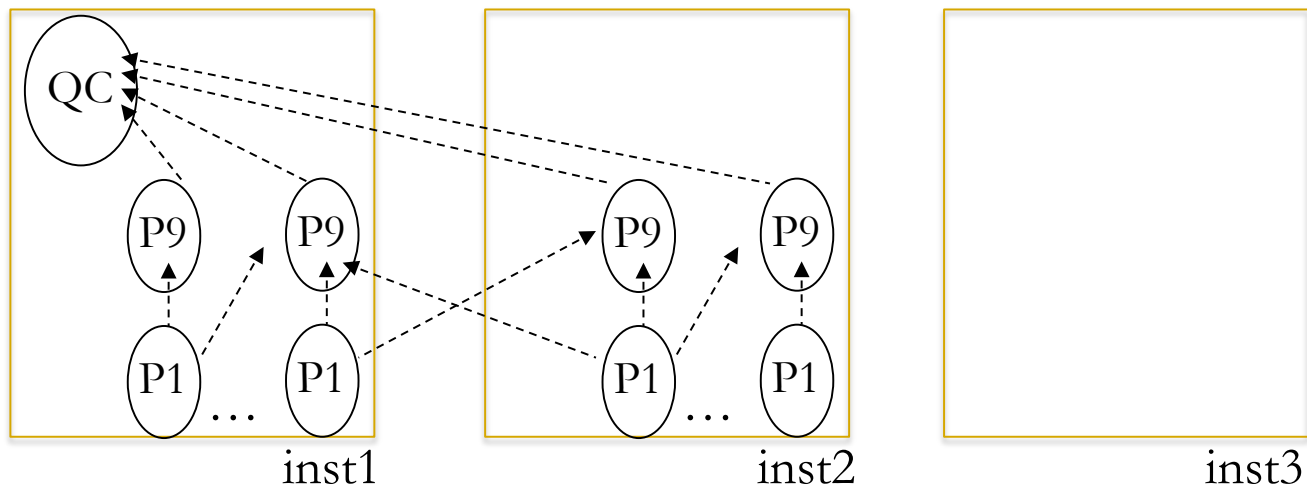


Number of communication channels can be up to square of parallelism, as each producer need to send buffers to every consumer.

## Placement: Two instances

- Sessions starting PX operations in inst1 can allocate slaves in both inst1 and inst2

```
inst1.instance_groups='inst1','inst12'  
inst2.instance_groups='inst2','inst12'  
inst3.instance_groups='inst3'  
inst1.parallel_instance_group= 'inst12'
```

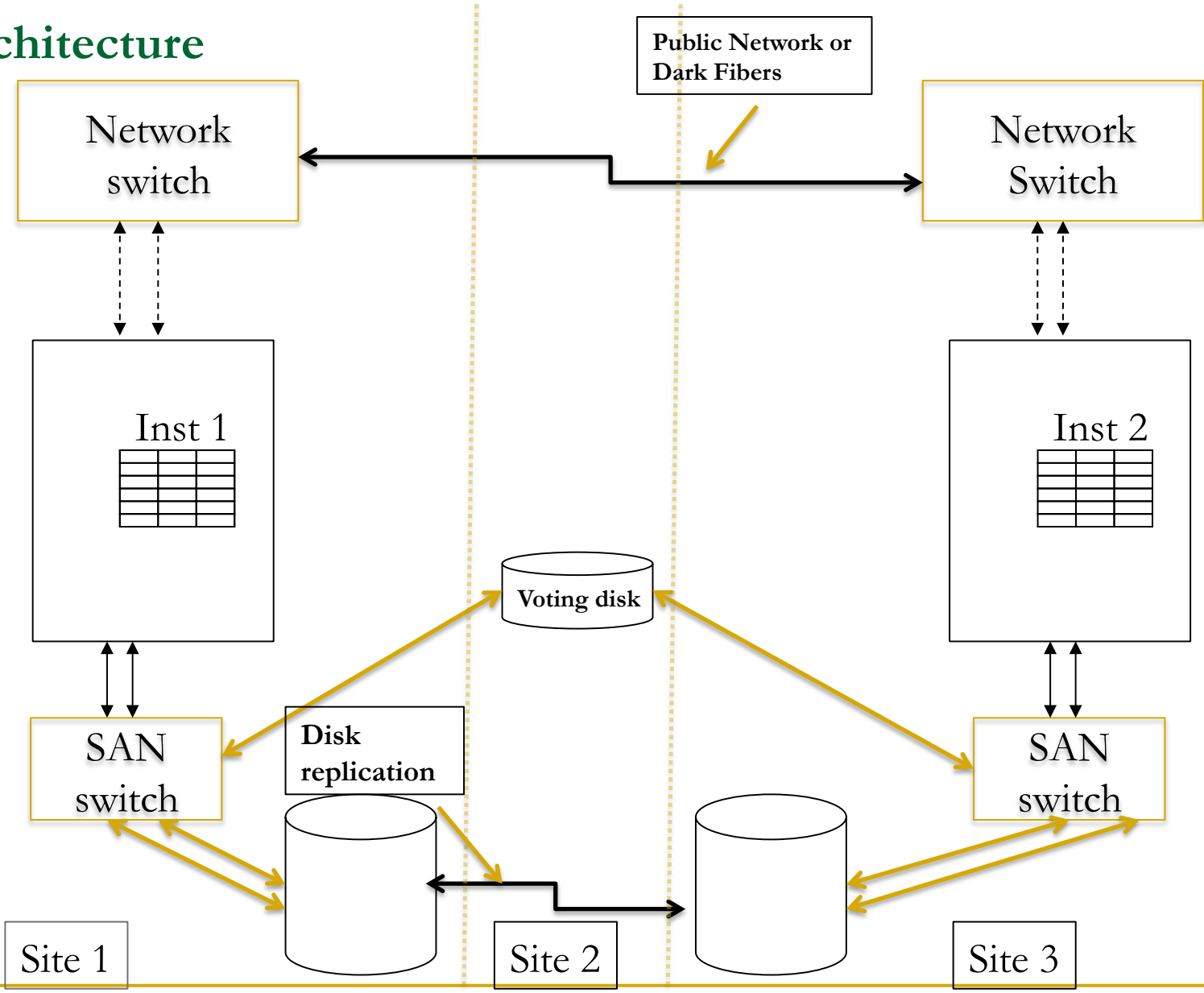


---

## 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.

Thank you for attending!

If you like this presentation, you will love my  
upcoming intensive RAC webinar  
in Aug/Sep 2011.

Watch for updates in:

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

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

