
RAC: Administering Parallel Execution

By
Riyaj Shamsudeen



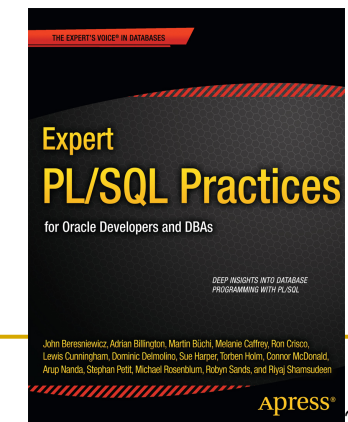
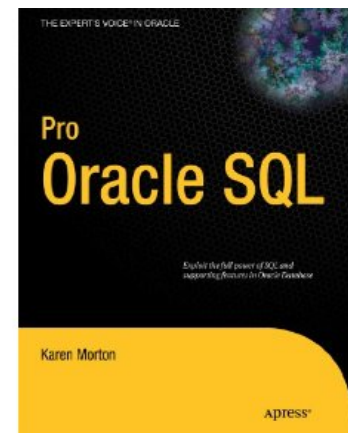
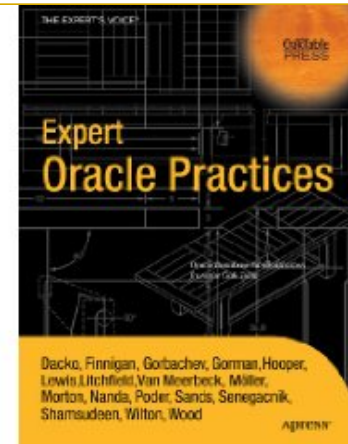
Who am I?



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



Parallel Execution

- Parallel query uses PX slave processes to perform work.
- A slave set perform specific task at a point in execution plan. After the completion of a task, slave set can be reassigned to perform a different task.
- Proper configuration of private interconnect and optimal execution plan is an essential step in scaling the PX operation.
- Placement of PX slaves can be controlled by Services or `parallel_instance_group` configuration.

Example

```
create index mut_t1 on mut (transaction_id)
parallel (degree 8) nologging;
```

| Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time | TQ | IN-OUT | PQ Distrib |
|-----|------------------------|----------|------|-------|-------------|----------|-------|--------|------------|
| 0 | CREATE INDEX STATEMENT | | | | 117K(100) | | | | |
| 1 | PX COORDINATOR | | | | | | | | |
| 2 | PX SEND QC (ORDER) | :TQ10001 | 13M | 166M | | | Q1,01 | P->S | QC (ORDER) |
| 3 | INDEX BUILD NON UNIQUE | MUT_T1 | | | | | Q1,01 | PCWP | |
| 4 | SORT CREATE INDEX | | 13M | 166M | | | Q1,01 | PCWP | |
| 5 | PX RECEIVE | | 13M | 166M | 116K (1) | 00:05:50 | Q1,01 | PCWP | |
| 6 | PX SEND RANGE | :TQ10000 | 13M | 166M | 116K (1) | 00:05:50 | Q1,00 | P->P | RANGE |
| 7 | PX BLOCK ITERATOR | | 13M | 166M | 116K (1) | 00:05:50 | Q1,00 | PCWP | |
| * 8 | TABLE ACCESS FULL | MUT | 13M | 166M | 116K (1) | 00:05:50 | Q1,00 | PCWP | |

One set of server processes reading the data

Another set of server processes, sorting and creating index segments.

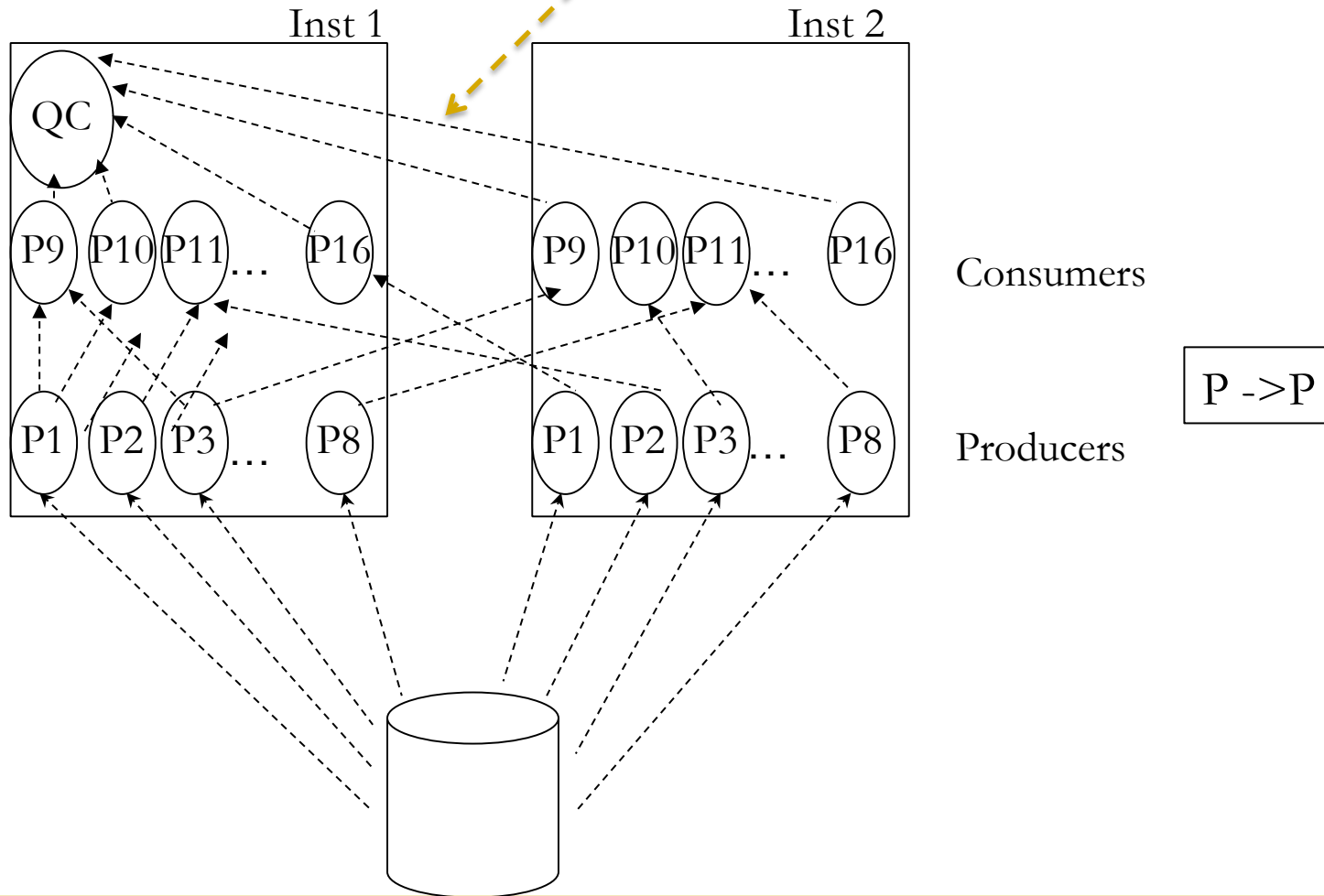
P -> P transfer between these two slave sets.

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.

Architecture

Distribution method of rows between producers and consumers are key to avoid flooding interconnect.



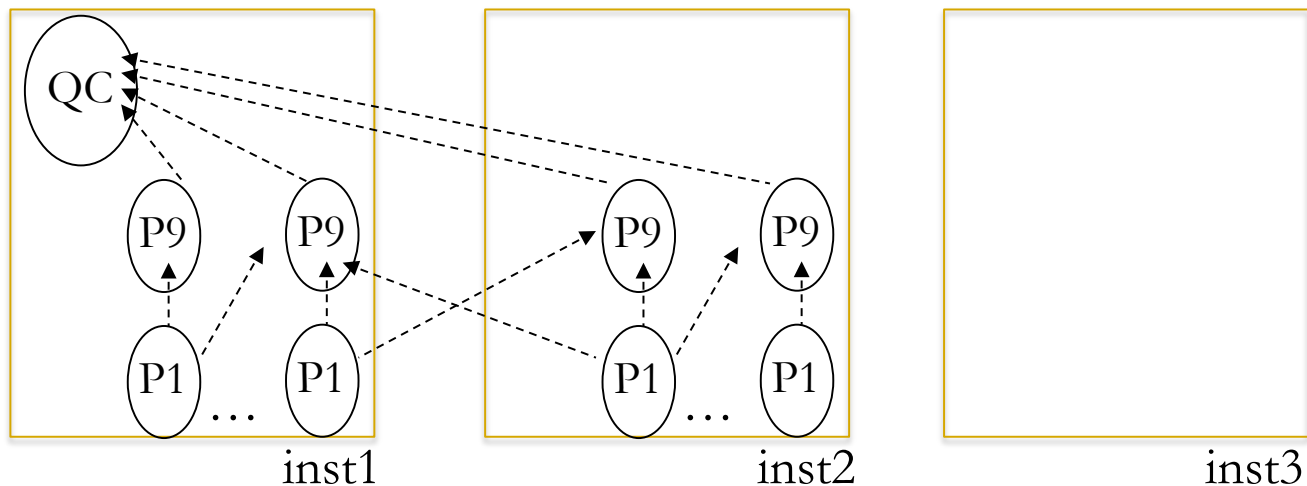
Controlling placement

- Allocation of PX slaves can be controlled using few techniques:
 - 1) Services
 - 2) Instance groups/parallel_instance_group
 - 3) Combination of Services and parallel_instance_group
- Instance_groups based setup is widely used in the database versions 10g and below. By default, all instances can participate in a PX operation.
- From 10g onwards, you should control placement of slaves using Services.
- Placement control using services is much more elegant and dynamic. Failover of services automatically handles PX placement.

Placement: Two instances

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

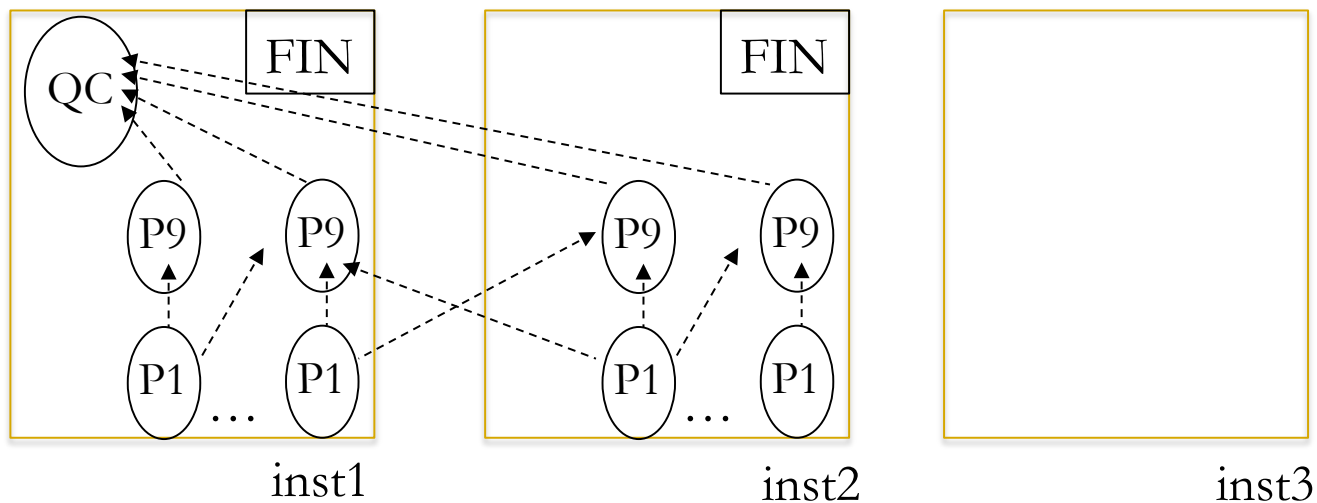
```
inst1.instance_groups='inst1','inst12'  
inst2.instance_groups='inst2','inst12'  
inst3.instance_groups='inst3'  
inst1.parallel_instance_group= 'inst12'
```



Placement: Services: two instances

- Service FIN is located in two instance.
- Slaves allocated from inst1 and inst2.

```
srvctl add service -d solrac -s FIN -r inst1,inst2 -a inst3
```

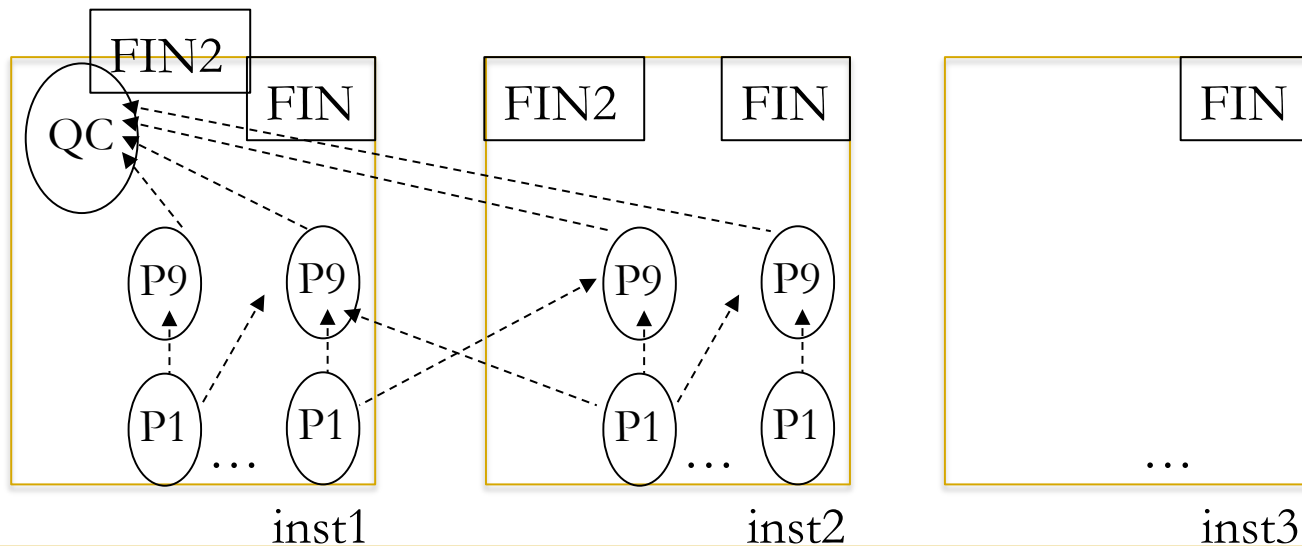


demo:pq_query_nopart.sql with po service

Placement: Combination: PIG

- Service alone also can be used to control the placement.
- Slaves will be allocated from two instances: inst1 and inst2.

```
srvctl add service -d solrac -s FIN -r inst1,inst2,inst3  
srvctl add service -d solrac -s FIN2 -r inst1,inst2  
Alter session set parallel_instance_group='FIN2';
```



Recommendations

- Both instance_groups/parallel_instance_groups and Services method can be concurrently used in a database.
- That might complicate debugging, **Use services from 11g onwards**. Use instance_groups for 10g and below.
- For more granular control, use parallel_instance_groups and point to a more appropriate service, as an exception.

Measuring PQ traffic

- Until 11g, measuring PQ traffic is not easy.
- In 10g, If AWR does not report PQ interconnect traffic, but if the device statistics are reporting high interconnect traffic, then it is possible that application might be generating high PQ load.
- IPQ statistics are visible in `x$ksxpclient` at a client level from 11g onwards..
- AWR snaps stores these statistics in `dba_hist_ic_client_stats` and prints PQ traffic in AWR section in a more readable format.

AWR report

- AWR report in 11g prints the stats as a per second rate

Interconnect Throughput by Client DB/Inst: SOLRAC/solrac1 Snaps: 1012-1013

-> Throughput of interconnect usage by major consumers

-> All throughput numbers are megabytes per second

| Used By | Send Mbytes/sec | Receive Mbytes/sec |
|----------------|--------------------|-----------------------|
| Global Cache | .02 | .04 |
| Parallel Query | .36 | .15 |
| DB Locks | .01 | .01 |
| DB Streams | .00 | .00 |
| Other | .00 | .00 |

PQ and cache fusion

- Parallel query slave processes read the block directly in to their PGA using direct reads (except 11gR2 new feature – in memory parallelism).
- With direct reads to PGA, there is no need for global cache grants.
- For intra-instance parallel operation, the performance difference between a single instance and RAC is minimal.
- Still, objects need to be checkpointed at the start of a PQ operation from *all* nodes in case of RAC.

Parallel_execution_message_size

- Size of buffer transferred between the slaves is determined by the parameter `parallel_execution_message_size` (PEMS).
- Default value is too small increasing chatty traffic.
- The default value is 2k or 4K depending upon the version.
- Increase the value of this parameter to at least 16K. Downside is that increase in shared pool size (PX Msg Buffers).
- Realizing the performance implications of this parameter, default value increased to 16K in 11gR2 (Compatible must be set to 11.2.0.2 and OS specific).
- Jumbo frames + PEMS, both set above 16K are good starting points for the inter-instance PQ intensive environments.

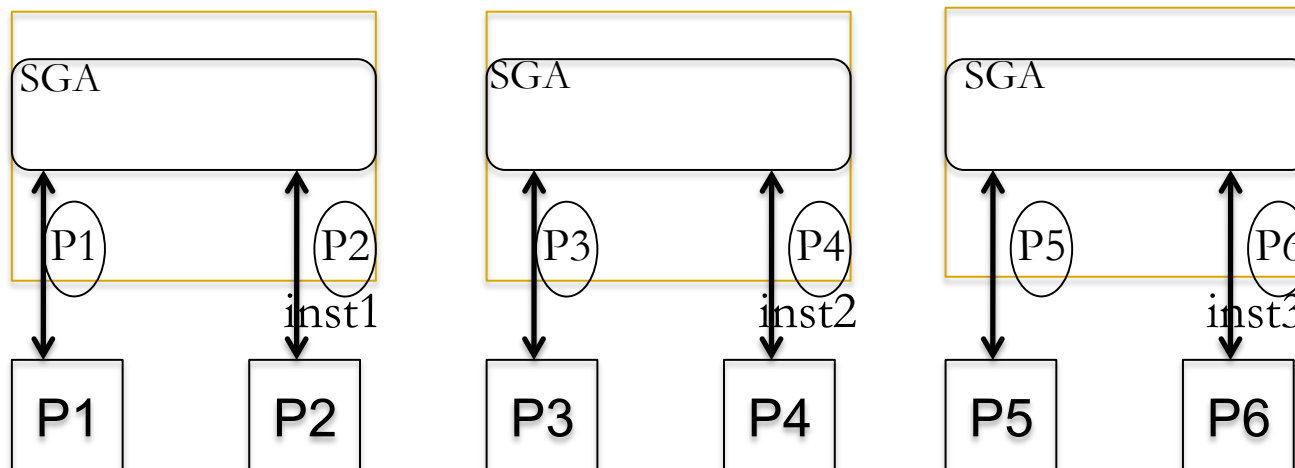
New features (11.2)

- Auto DOP: Optimizer chooses optimal parallelism even if you don't specify parallel hint or parallel degree at object level.
- Parallel Statement Queueing: Query will be queued until there is sufficient amount of PQ servers available.
- PQ queueing feature is useful in PQ intensive data warehousing environments.
- In memory parallelism: PQ can read blocks in to buffer cache.

Demo: pq_queueing.sql if time permits

In-memory parallelism (11gR2)

- Due to the size of mammoth servers, now it is not uncommon to see SGA with a size of 100GB.
- 11gR2 introduced in-memory parallelism. Essentially, PQ servers can read buffers in to SGA.



Parallel_force_local (11.2)

- This parameter controls whether the parallelism to be forced to single instance.
- Default is FALSE and there is no reason to change it. Parallel Statement Queueing feature (11.2) interacts badly if this parameter is true.
- Use services if you want to keep Parallel executions to a single node, rather than adjusting this parameter.

Parallel DML

- Parallel DML (changes) also works in the same way of Parallel query.
- Parallel slaves can be allocated in an instance or multiple instances depending upon the configuration.
- In DML operation there are two distinct group of parallel operations:
 - Parallelism for scan only.
 - Parallelism for both scan and changes.

Parallel DML – scan only-execution plan

In the execution plan below, UPDATE step is above Co-ordinator. Only Scanning is done in parallel, changes are done in serial.

```
----- ...
| Id | Operation                               | Name           | Rows | Bytes | Cost (%CPU)| ...
----- ...
| 0  | UPDATE STATEMENT                       |                |    1 | 1028 | 2600 (1)| ...
| 1  | UPDATE                                 | HUGETABLE_HASH |    1 |      |          | ...
| 2  | PX COORDINATOR                         |                |    1 |      |          | ...
| 3  | PX SEND QC (RANDOM)                     | :TQ10003      |    1 | 1028 | 2600 (1)| ...
| 4  | NESTED LOOPS                           |                |    1 | 1028 | 2600 (1)| ...
| 5  | BUFFER SORT                             |                |    1 |      |          | ...
| 6  | PX RECEIVE                              |                |    1 |      |          | ...
| 7  | PX SEND BROADCAST                       | :TQ10002      |    1 |      |          | ...
| 8  | VIEW                                    | VW_NS0_1      | 32186 | 408K | 2445 (1)| ...
| 9  | HASH UNIQUE                             |                |    1 | 817K |          | ...
| 10 | PX RECEIVE                              |                |    1 | 817K |          | ...
| 11 | PX SEND HASH                            | :TQ10001      |    1 | 817K |          | ...
| 12 | HASH UNIQUE                             |                |    1 | 817K |          | ...
|* 13 | HASH JOIN                               |                | 32186 | 817K | 2445 (1)| ...
| 14 | PX BLOCK ITERATOR                       |                | 32186 | 408K | 1222 (0)| ...
| 15 | TABLE ACCESS FULL                      | HUGETABLE_HASH | 32186 | 408K | 1222 (0)| ...
-----
```

Should you use inter or intra-parallel operations?

- This is a thorny question in production environment.
- There is no correct answer to this question, it depends upon two factors:
 1. Node resources such as CPU, I/O bandwidth, IC bandwidth
 2. Size of the segment, whether it is partitioned or not etc.
- If the index creation can be completed with just one node, meaning the nodes are beefed up nodes, then you should use one node.
- If the nodes are not big enough, then it is okay to use all nodes, just realize that interconnect latency might affect performance.

Thank you for attending!

If you like this presentation, you will love my upcoming intensive **Advanced RAC Troubleshooting** class. Watch for updates in:

www.tanelpoder.com

Orainternals.wordpress.com

Contact info:

Email: rshamsud@gmail.com

Blog : orainternals.wordpress.com

URL : www.orainternals.com



MARK YOUR CALENDARS!

COLLABORATE 12
April 22-26, 2012
Mandalay Bay Convention Center
Las Vegas, Nevada

<http://events.ioug.org/p/cm/ld/fid=15>

