The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Implement Best Practices for Extreme Performance with Oracle Data Warehousing

Maria Colgan
Principal Product Manager
Agenda

- The three P's of Data Warehousing
  - Power
  - Partitioning
  - Parallel Execution
- Data Loading
- Workload Management
  - Statistics management
  - Initialization Parameters
  - Workload Monitoring
The Three Ps
3 Ps - Power, Partitioning, Parallelism

- Balanced Hardware Configuration
  - Weakest link defines the throughput
- larger tables or fact tables should be partitioned
  - Facilitates data load, data elimination and join performance
  - Enables easier Information Lifecycle Management
- Parallel Execution should be used
  - Instead of one process doing all the work multiple processes working concurrently on smaller units
  - Parallel degree should be power of 2
Balanced Configuration

“The weakest link” defines the throughput

- CPU Quantity and Speed dictate number of HBAs
- HBA Quantity and Speed dictate number of Disk Controllers
- Controllers Quantity and Speed dictate number of Disks
- Disk Quantity and Speed

CPU Quantity and Speed dictate capacity of interconnect

HBA Quantity and Speed dictate Speed and quantity of switches

Controllers Quantity and Speed dictate Speed and quantity of switches

Disk Quantity and Speed
Sun Oracle Database Machine
A Balance Hardware Configuration

Extreme Performance

RAC Database Server Grid
- 8 High-performance low-cost compute servers
- 2 Intel quad-core Xeons each

Exadata Storage Server Grid
- 14 High-performance low-cost storage servers
- 100 TB raw SAS disk storage
- 5TB of Flash storage
- 21 GB/sec disk bandwidth
- 50 GB/sec flash bandwidth
- 100 GB/sec memory bandwidth

InfiniBand Network
- 3 36-port Infiniband
- 880 Gb/sec aggregate throughput
Partitioning

- Range partition large fact tables typically on date column
  - Consider data loading frequency
    - Is an incremental load required?
    - How much data is involved, a day, a week, a month?
  - Partition pruning for queries
    - What range of data do the queries touch - a quarter, a year?
- Subpartition by hash to improve join performance between fact tables and / or dimension tables
  - Pick the common join column
  - If all dimension have different join columns use join column for the largest dimension or most common join in the queries
Partition Pruning

Q: What was the total sales for the weekend of May 20 - 22 2008?

Select sum(sales_amount)
From SALES
Where sales_date between
to_date('05/20/2008','MM/DD/YYYY')
And
to_date('05/23/2008','MM/DD/YYYY');

Only the 3 relevant partitions are accessed
Partition Wise join

Both tables have the same degree of parallelism and are partitioned the same way on the join column (cust_id)

A large join is divided into multiple smaller joins, each joins a pair of partitions in parallel

```
Select sum(sales_amount) 
From 
SALES s, CUSTOMER c 
Where s.cust_id = c.cust_id;
```
### Execution plan for partition-wise join

<table>
<thead>
<tr>
<th>ID</th>
<th>Operation</th>
<th>Name</th>
<th>Pstart</th>
<th>Pstop</th>
<th>TQ</th>
<th>PQ Distrib</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PX COORDINATOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>QC (RAND)</td>
</tr>
<tr>
<td>2</td>
<td>PX SEND QC (RANDOM)</td>
<td>:TQ10001</td>
<td></td>
<td></td>
<td>Q1,01</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SORT GROUP BY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PX RECEIVE</td>
<td></td>
<td></td>
<td></td>
<td>Q1,01</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PX SEND HASH</td>
<td>:TQ10000</td>
<td></td>
<td></td>
<td>Q1,00</td>
<td>HASH</td>
</tr>
<tr>
<td>6</td>
<td>SORT GROUP BY</td>
<td></td>
<td></td>
<td></td>
<td>Q1,00</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>PX PARTITION HASH ALL</td>
<td></td>
<td>1</td>
<td>128</td>
<td>Q1,00</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>HASH JOIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TABLE ACCESS FULL</td>
<td>Customers</td>
<td>1</td>
<td>128</td>
<td>Q1,00</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>TABLE ACCESS FULL</td>
<td>Sales</td>
<td>1</td>
<td>128</td>
<td>Q1,00</td>
<td></td>
</tr>
</tbody>
</table>
How Parallel Execution works

User connects to the database

Background process is spawned

When user issues a parallel SQL statement the background process becomes the Query Coordinator

Parallel servers communicate among themselves & the QC using messages that are passed via memory buffers in the shared pool

Parallel servers - individual sessions that perform work in parallel. Allocated from a pool of globally available parallel server processes & assigned to a given operation.
### Parallel Execution Plan

```
SELECT c.cust_name, s.purchase_date, s.amount
FROM sales s, customers c
WHERE s.cust_id = c.cust_id;
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Operation</th>
<th>Name</th>
<th>TQ</th>
<th>IN-OUT</th>
<th>PQ Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PX COORDINATOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PX SEND QC (RANDOM)</td>
<td></td>
<td>Q1,01</td>
<td>P-&gt;S</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>HASH JOIN</td>
<td></td>
<td>Q1,01</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PX RECEIVE</td>
<td></td>
<td>Q1,01</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PX SEND BROADCAST</td>
<td></td>
<td>Q1,01</td>
<td>P-&gt;P</td>
<td>BROADCAST</td>
</tr>
<tr>
<td>6</td>
<td>PX BLOCK ITERATOR</td>
<td></td>
<td>Q1,01</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS FULL</td>
<td>CUSTOMERS</td>
<td>Q1,01</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>PX BLOCK ITERATOR</td>
<td></td>
<td>Q1,01</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TABLE ACCESS FULL</td>
<td>SALES</td>
<td>Q1,01</td>
<td>PCWP</td>
<td></td>
</tr>
</tbody>
</table>

**Query Coordinator**

**Parallel Servers**
do majority of the work
Parallel Execution of a Query

```
SELECT c.cust_name, s.date, s.amount
FROM sales s, customers c
WHERE s.cust_id = c.cust_id;
```
### Producers and Consumer in the execution plan

<table>
<thead>
<tr>
<th>ID</th>
<th>Operation</th>
<th>Name</th>
<th>TQ</th>
<th>IN-OUT</th>
<th>PQ Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PX COORDINATOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PX SEND QC (RANDOM)</td>
<td></td>
<td>Q1.02</td>
<td>P-&gt;S</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>HASH JOIN</td>
<td></td>
<td>Q1.02</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PX RECEIVE</td>
<td></td>
<td>Q1.02</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PX SEND HASH</td>
<td></td>
<td>Q1.00</td>
<td>P-&gt;P</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>PX BLOCK ITERATOR</td>
<td></td>
<td>Q1.00</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS FULL</td>
<td>CUSTOMERS</td>
<td>Q1.00</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>PX RECEIVE</td>
<td></td>
<td>Q1.02</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PX SEND HASH</td>
<td></td>
<td>Q1.01</td>
<td>P-&gt;P</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>PX BLOCK ITERATOR</td>
<td></td>
<td>Q1.01</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>TABLE ACCESS FULL</td>
<td>SALES</td>
<td>Q1.01</td>
<td>PCWP</td>
<td></td>
</tr>
</tbody>
</table>
Parallel Execution of a Scan

- Data is divided into Granules
  - block range or partition
- Each Parallel Server is assigned one or more Granules
- No two Parallel Servers ever contend for the same Granule
- Granules are assigned so that the load is balanced across all Parallel Servers
- **Dynamic Granules chosen by the optimizer**
- Granule decision is visible in execution plan
Identifying Granules of Parallelism during scans in the plan

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
<th>Pstart</th>
<th>Pstop</th>
<th>TQ</th>
<th>IN-OUT</th>
<th>PQ Distrib</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>17</td>
<td>153</td>
<td>565 (100)</td>
<td>00:00:07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PX COORDINATOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PX SEND QC (RANDOM)</td>
<td>:TQ10001</td>
<td>17</td>
<td>153</td>
<td>565 (100)</td>
<td>00:00:07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>HASH GROUP BY</td>
<td></td>
<td>17</td>
<td>153</td>
<td>565 (100)</td>
<td>00:00:07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PX RECEIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PX SEND HASH</td>
<td>:TQ10000</td>
<td>17</td>
<td>153</td>
<td>565 (100)</td>
<td>00:00:07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>HASH GROUP BY</td>
<td></td>
<td>17</td>
<td>153</td>
<td>565 (100)</td>
<td>00:00:07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>PX BLOCK ITERATOR</td>
<td></td>
<td>10Mi</td>
<td>85Mi</td>
<td>60 (97)</td>
<td>00:00:01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>TABLE ACCESS FULL</td>
<td>SALES</td>
<td>10Mi</td>
<td>85Mi</td>
<td>60 (97)</td>
<td>00:00:01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

redicate Information (identified by operation id):

8 - filter("CUST_ID"<22810 AND "CUST_ID">=22300)

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
<th>Pstart</th>
<th>Pstop</th>
<th>TQ</th>
<th>IN-OUT</th>
<th>PQ Distrib</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>17</td>
<td>153</td>
<td>2 (50)</td>
<td>00:00:01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PX COORDINATOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PX SEND QC (RANDOM)</td>
<td>:TQ10001</td>
<td>17</td>
<td>153</td>
<td>2 (50)</td>
<td>00:00:01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>HASH GROUP BY</td>
<td></td>
<td>17</td>
<td>153</td>
<td>2 (50)</td>
<td>00:00:01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PX RECEIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PX SEND HASH</td>
<td>:TQ10000</td>
<td>26</td>
<td>234</td>
<td>1 (0)</td>
<td>00:00:01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>PX PARTITION RANGE ALL</td>
<td></td>
<td>26</td>
<td>234</td>
<td>1 (0)</td>
<td>00:00:01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS BY LOCAL INDEX ROWIDs</td>
<td>SALES</td>
<td>26</td>
<td>234</td>
<td>1 (0)</td>
<td>00:00:01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>INDEX RANGE SCAN</td>
<td>SALES_CUST</td>
<td>26</td>
<td></td>
<td>0 (0)</td>
<td>00:00:01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

redicate Information (identified by operation id):

8 - access("F1ST_III"<22810 AND "F1ST_III">=22300)
1. Use RAC Services

Create two services

Srvctl add service –d database_name
   -s ETL
   -r sid1, sid2

Srvctl add service –d database_name
   -s AHOC
   -r sid3, sid4

2. PARALLEL_FORCE_LOCAL - New Parameter forces parallel statement to run on just node it was issued on
   Default is FALSE
Use Parallel Execution with common sense

- Parallel execution provides performance boost but requires more resources
- General rules of thumb for determining the appropriate DOP
  - objects smaller than 200 MB should not use any parallelism
  - objects between 200 MB and 5GB should use a DOP of 4
  - objects beyond 5GB use a DOP of 32

Mileage may vary depending on concurrent workload and hardware configuration
Data Loading
Access Methods

Data Pump

TTS

Flat Files

XML Files

Web Services

DBLinks

Bulk Performance

Heterogeneous
Data Loading Best Practices

• **External Tables**
  – Allows flat file to be accessed via SQL PL/SQL as if it was a table
  – Enables complex data transformations & data cleansing to occur “on the fly”
  – Avoids space wastage

• **Pre-processing**
  – Ability to specify a program that the access driver will execute to read the data
  – Specify gunzip to decompress a .gzip file “on the fly” while its being

• **Direct Path in parallel**
  – Bypasses buffer cache and writes data directly to disk via multi-block async IO
  – Use parallel to speed up load
  – Remember to use `ALTER SESSION ENABLE PARALLEL DML`

• **Range Partitioning**
  – Enables partition exchange loads

• **Data Compression**
SQL Loader or External Tables

• And the winner is => External Tables

• Why:
  – Full usage of SQL capabilities directly on the data
  – Automatic use of parallel capabilities (just like a table)
  – No need to stage the data again
  – Better allocation of space when storing data
    • High watermark brokering
    • Autoallocate tablespace will trim extents after the load
  – Interesting capabilities like
    • The usage of data pump
    • The usage of pre-processing
Tips for External Tables

- **File locations and size**
  - When using multiple files the file size should be similar
  - List largest to smallest in LOCATION clause if not similar in size
- **File Formats**
  - Use a format allowing position-able and seek-able scans
  - Delimitate clearly and use well known record termination to allow for automatic Granulation
  - Always specify the character set if it's different to the database
- Consider compressing data files and uncompressing during loading
- Run all queries before the data load to populate column usage for histogram creation during statistics gathering
Pre-Processing in an External Table

- New functionality in 11.1.0.7 and 10.2.0.5
- Allows flat files to be processed automatically during load
  - Decompression of large file zipped files
- Pre-processing doesn’t support automatic granulation
  - Need to supply multiple data files - # of files will determine DOP
- Need to Grant read, execute privileges directories

```sql
CREATE TABLE sales_external
  (...) ORGANIZATION EXTERNAL
  ( TYPE ORACLE_LOADER
    DEFAULT DIRECTORY data_dir1
    ACCESS PARAMETERS
    (RECORDS DELIMITED BY NEWLINE
      PREPROCESSOR exec_dir: 'gunzip'
      FIELDS TERMINATED BY '|
    )
  )
  LOCATION (...));
```
Direct Path Load

- Data is written directly to the database storage using multiple blocks per I/O request using asynchronous writes
- A CTAS command always uses direct path
- An Insert As Select needs an APPEND hint to go direct

```
Insert /*+ APPEND */ into Sales partition(p2)
Select * From ext_tab_for_sales_data;
```

- Only one direct path operation can occur on an object
  - By specifying a specific partition name in the table you can do multiple concurrent direct path loads into a partitioned table
Parallel Load

- Ensure direct path loads go parallel
  - Specify parallel degree either with hint or on both tables
  - Enable parallelism by issuing alter session command
- CTAS will go parallel automatically when DOP is specified
- IAS will not – it needs parallel DML to be enabled

ALTER SESSION ENABLE PARALLEL DML;
Partition Exchange Loading

1. Create external table for flat files
2. Use CTAS command to create non-partitioned table TMP_SALES
3. Create indexes
4. Alter table Sales exchange partition May_24_2008 with table tmp_sales
5. Gather Statistics
Data Compression

- Use if data being loaded will be read / used more than once
- Works by eliminating duplicate values within a database block
- Reduces disk and memory usage, often resulting in better scale-up performance for read-only operations
- Require additional CPU during the initial data load
- But what if workload requires conventional DML access to the data after it has been loaded?

Use the COMPRESS FOR ALL OPERATIONS
Workload Monitoring
Statistics gathering

- You must gather optimizer statistics
  - Using dynamic sampling is not an adequate solution
- Run all queries against empty tables to populate column usage
  - This helps identify which columns automatically get histograms created on them
- Optimizer statistics should be gathered after the data has been loaded but before any indexes are created
  - Oracle will automatically gather statistics for indexes as they are being created
Statistics Gathering

- By default DBMS_STATS gathers following stats for each table
  - global (table level)
  - partition level
  - Sub-partition
- Optimizer uses global stats if query touches two or more partitions
- Optimizer uses partition stats if queries do partition elimination and only one partition is necessary to answer the query
  - If queries touch two or more partitions the optimizer will use a combination of global and partition level statistics
- Optimizer uses sub-partition level statistics if your queries do partition elimination and only one sub-partition is necessary to answer query
Efficiency Statistics Management

• How do I gather accurate Statistics
  • “.. Compute statistics gives accurate results but takes too long ..”
  • “.. Sampling is fast but not always accurate ..”
  • “.. AUTO SAMPLE SIZE does not always work with data skew ..”

  • New groundbreaking implementation for AUTO SAMPLE SIZE
    • Faster than sampling
    • Accuracy comparable to compute statistics

• Gathering statistics on one partition (e.g. after a bulk load) causes a full scan of all partitions to gather global table statistics Extremely time and resource intensive

  • Use incremental statistics
    • Gather statistics for touched partition(s) ONLY
    • Table (global) statistics are built from partition statistics
Incremental Global Statistics

1. Partition level stats are gathered & synopsis created

2. Global stats generated by aggregating partition synopsis

Sales Table

- May 18th 2008
- May 19th 2008
- May 20th 2008
- May 21st 2008
- May 22nd 2008
- May 23rd 2008

Syxaux Tablespace

Global Statistic

S1
S2
S3
S4
S5
S6
Incremental Global Statistics Cont’d

3. A new partition is added to the table & Data is Loaded

5. Retrieve synopsis for each of the other partitions from Sysaux

6. Global stats generated by aggregating the original partition synopsis with the new one

Sales Table

May 18th 2008
May 19th 2008
May 20th 2008
May 21st 2008
May 22nd 2008
May 23rd 2008
May 24th 2008

S1
S2
S3
S4
S5
S6
$7

Sysaux Tablespace

Global Statistic
Step necessary to gather accurate statistics

- Turn on incremental feature for the table
  
  ```sql
  EXEC
  DBMS_STATS.SET_TABLE_PREFS('SH','SALES','INCREMENTAL','TRUE');
  ```

- After load gather table statistics using `GATHER_TABLE_STATS` command don’t need to specify many parameter
  
  ```sql
  EXEC DBMS_STATS.GATHER_TABLE_STATS('SH','SALES');
  ```

- The command will collect statistics for partitions and update the global statistics based on the partition level statistics and synopsis

- Possible to set incremental to true for all tables using
  
  ```sql
  EXEC DBMS_STATS.SET_GLOBAL_PREFS('INCREMENTAL','TRUE');
  ```
## Initialization parameters

*Only set what you really need to*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>compatible</td>
<td>11.1.0.7.0</td>
<td>Needed for Exadata</td>
</tr>
<tr>
<td>db_block_size</td>
<td>8 KB</td>
<td>Larger size may help with compression ratio</td>
</tr>
<tr>
<td>db_cache_size</td>
<td>5 GB</td>
<td>Large enough to hold metadata</td>
</tr>
<tr>
<td>parallel_adaptive_multi_user</td>
<td>False</td>
<td>Can cause unpredictable response times as it is based on concurrency</td>
</tr>
<tr>
<td>parallel_execution_message_size</td>
<td>16 KB</td>
<td>Improves parallel server processes communication</td>
</tr>
<tr>
<td>parallel_min_servers</td>
<td>64</td>
<td>Avoids query startup costs</td>
</tr>
<tr>
<td>parallel_max_servers</td>
<td>128</td>
<td>Prevents systems from being flooded by parallel servers</td>
</tr>
<tr>
<td>pga_aggregate_target</td>
<td>18 GB</td>
<td>Tries to keep sorts in memory</td>
</tr>
<tr>
<td>shared_pool_size</td>
<td>4 GB</td>
<td>Large enough to for PX communicate and SQL Area</td>
</tr>
</tbody>
</table>
Using EM to monitor Parallel Query

Click on the performance tab
Parallel Execution screens

Additional Monitoring Links
Top Sessions and Top SQL data from ASH can be found on the Top Activity page.

- Top Activity
- Top Consumers
- Duplicate SQL
- Blocking Sessions
- Hang Analysis
- Instance Links
- Instance Activity
- Search Sessions
- Search SQL
- Snapshots

Click on the SQL Monitoring link

Oracle
Using EM to monitor Parallel Query

Click on a SQL ID to drill down to more details

Shows parallel degree used and number of nodes used in query
Only one set of parallel servers
Using EM to monitor Parallel Query
SQL Monitoring screens

Click on parallel tab to get more info on PQ

The green arrow indicates which line in the execution plan is currently being worked on
SQL Monitoring Screens

By clicking on the + tab you can get more detail about what each individual parallel server is doing. You want to check each slave is doing an equal amount of work.

<table>
<thead>
<tr>
<th>Parallel Server</th>
<th>Database Time</th>
<th>Wait Activity %</th>
<th>IO Count</th>
<th>Buffer Gets</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Parallel Servers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Coordinator</td>
<td>8.4s</td>
<td>2.04</td>
<td></td>
<td>264K</td>
</tr>
<tr>
<td>Parallel Set 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Server 1 (p001)</td>
<td>3.4s</td>
<td></td>
<td></td>
<td>8413</td>
</tr>
<tr>
<td>Parallel Server 2 (p008)</td>
<td>29.0s</td>
<td>2.04</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>Parallel Server 3 (p003)</td>
<td>3.4s</td>
<td>4.08</td>
<td></td>
<td>8063</td>
</tr>
<tr>
<td>Parallel Server 4 (p003)</td>
<td>6.6s</td>
<td>4.08</td>
<td>4769</td>
<td>19K</td>
</tr>
<tr>
<td>Parallel Server 5 (p004)</td>
<td>3.7s</td>
<td>2.04</td>
<td></td>
<td>9242</td>
</tr>
<tr>
<td>Parallel Server 6 (p012)</td>
<td>3.5s</td>
<td>4.08</td>
<td>9207</td>
<td>9016</td>
</tr>
<tr>
<td>Parallel Server 7 (p006)</td>
<td>6.4s</td>
<td></td>
<td>1062</td>
<td></td>
</tr>
<tr>
<td>Parallel Server 8 (p007)</td>
<td>5.0s</td>
<td>2.04</td>
<td>100</td>
<td>8016</td>
</tr>
<tr>
<td>Parallel Server 9 (p008)</td>
<td>15.3s</td>
<td></td>
<td>3985</td>
<td></td>
</tr>
<tr>
<td>Parallel Set 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Server 10 (p009)</td>
<td>3.4s</td>
<td>2.04</td>
<td>86</td>
<td>8296</td>
</tr>
<tr>
<td>Parallel Server 11 (p010)</td>
<td>16.6s</td>
<td></td>
<td>4733</td>
<td></td>
</tr>
<tr>
<td>Parallel Server 12 (p011)</td>
<td>3.5s</td>
<td>2.04</td>
<td></td>
<td>8069</td>
</tr>
<tr>
<td>Parallel Server 13 (p012)</td>
<td>6.5s</td>
<td>4.00</td>
<td>1107</td>
<td></td>
</tr>
<tr>
<td>Parallel Server 14 (p013)</td>
<td>3.6s</td>
<td></td>
<td>108</td>
<td>9923</td>
</tr>
<tr>
<td>Parallel Server 15 (p014)</td>
<td>3.3s</td>
<td>2.04</td>
<td></td>
<td>8016</td>
</tr>
<tr>
<td>Parallel Server 16 (p015)</td>
<td>6.4s</td>
<td>2.04</td>
<td>1062</td>
<td>10K</td>
</tr>
</tbody>
</table>
Disk Configuration with ASM

![Disk Configuration with ASM](image-url)
For More Information

search.oracle.com

Best Practices for Data Warehousing

or

# Exadata Sessions

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Room</th>
<th>Session Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon</td>
<td>5:30 PM</td>
<td>Moscone South 307</td>
<td><strong>S311436</strong> - Implement Best Practices for Extreme Performance with Oracle Data Warehouses.</td>
</tr>
<tr>
<td>Tue</td>
<td>11:30 AM</td>
<td>Moscone South 307</td>
<td><strong>S311385</strong> - Extreme Backup and Recovery on the Oracle Database Machine.</td>
</tr>
<tr>
<td>Tue</td>
<td>1:00 PM</td>
<td>Moscone South 307</td>
<td><strong>S311437</strong> - Achieve Extreme Performance with Oracle Exadata and Oracle Database Machine.</td>
</tr>
<tr>
<td>Tue</td>
<td>1:00 PM</td>
<td>Moscone South Room 102</td>
<td><strong>S311358</strong> - Oracle's Hybrid Columnar Compression: The Next-Generation Compression Technology</td>
</tr>
<tr>
<td>Tue</td>
<td>2:30 PM</td>
<td>Moscone South 102</td>
<td><strong>S311386</strong> - Customer Panel 1: Exadata Storage and Oracle Database Machine Deployments.</td>
</tr>
<tr>
<td>Tue</td>
<td>4:00 PM</td>
<td>Moscone South 102</td>
<td><strong>S311387</strong> - Top 10 Lessons Learned Implementing Oracle and Oracle Database Machine.</td>
</tr>
<tr>
<td>Tue</td>
<td>5:30 PM</td>
<td>Moscone South 308</td>
<td><strong>S311420</strong> - Extreme Performance with Oracle Database 11g and In-Memory Parallel Execution.</td>
</tr>
<tr>
<td>Tue</td>
<td>5:30 PM</td>
<td>Moscone South Room 104</td>
<td><strong>S311239</strong> - The Terabyte Hour with the Real-World Performance Group</td>
</tr>
<tr>
<td>Tue</td>
<td>5:30 PM</td>
<td>Moscone South 252</td>
<td><strong>S310048</strong> - Oracle Beehive and Oracle Exadata: The Perfect Match.</td>
</tr>
<tr>
<td>Wed</td>
<td>4:00 PM</td>
<td>Moscone South 102</td>
<td><strong>S311387</strong> - Top 10 Lessons Learned Implementing Oracle and Oracle Database Machine.</td>
</tr>
<tr>
<td>Wed</td>
<td>5:00 PM</td>
<td>Moscone South 104</td>
<td><strong>S311383</strong> - Next-Generation Oracle Exadata and Oracle Database Machine: The Future Is Now.</td>
</tr>
<tr>
<td>Thu</td>
<td>12:00 PM</td>
<td>Moscone South 307</td>
<td><strong>S311511</strong> - Technical Deep Dive: Next-Generation Oracle Exadata Storage Server and Oracle Database Machine</td>
</tr>
</tbody>
</table>