

6

Oracle Exadata Storage Server Performance Monitoring and Optimization

ORACLE®

Copyright © 2009, Oracle. All rights reserved.

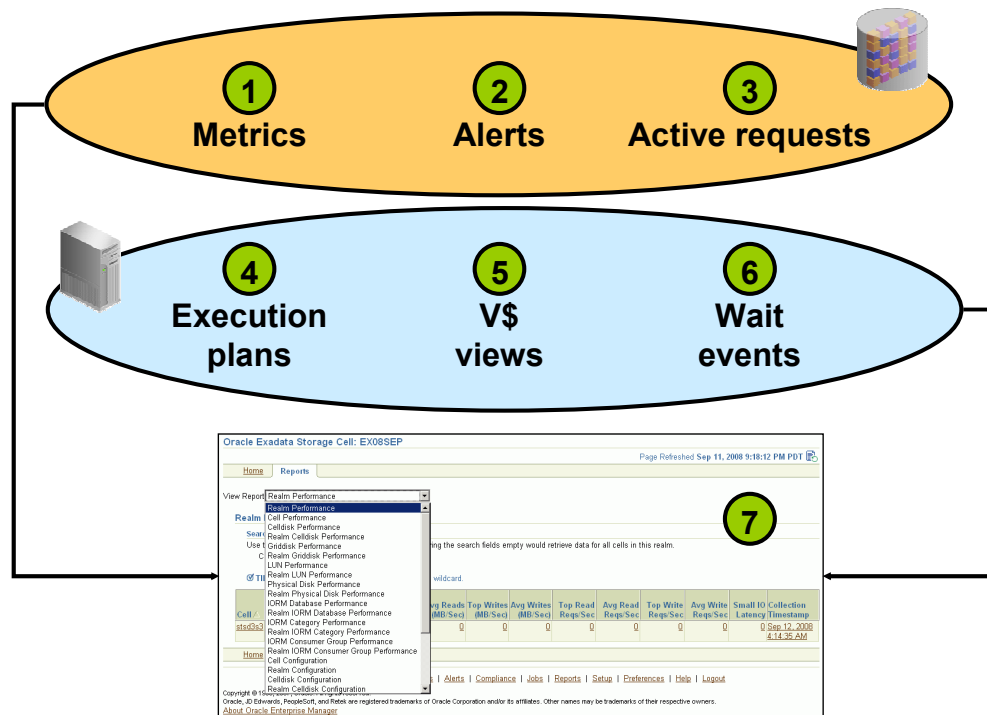
Objectives

After completing this lesson, you should be able to:

- Describe the various performance monitoring facilities available for Exadata
- Monitor Exadata from directly within a cell, from a database instance and through Enterprise Manager
- Interpret SQL execution plans that use smart scan
- Describe the recommendations for optimizing Exadata performance

ORACLE

Monitoring Overview

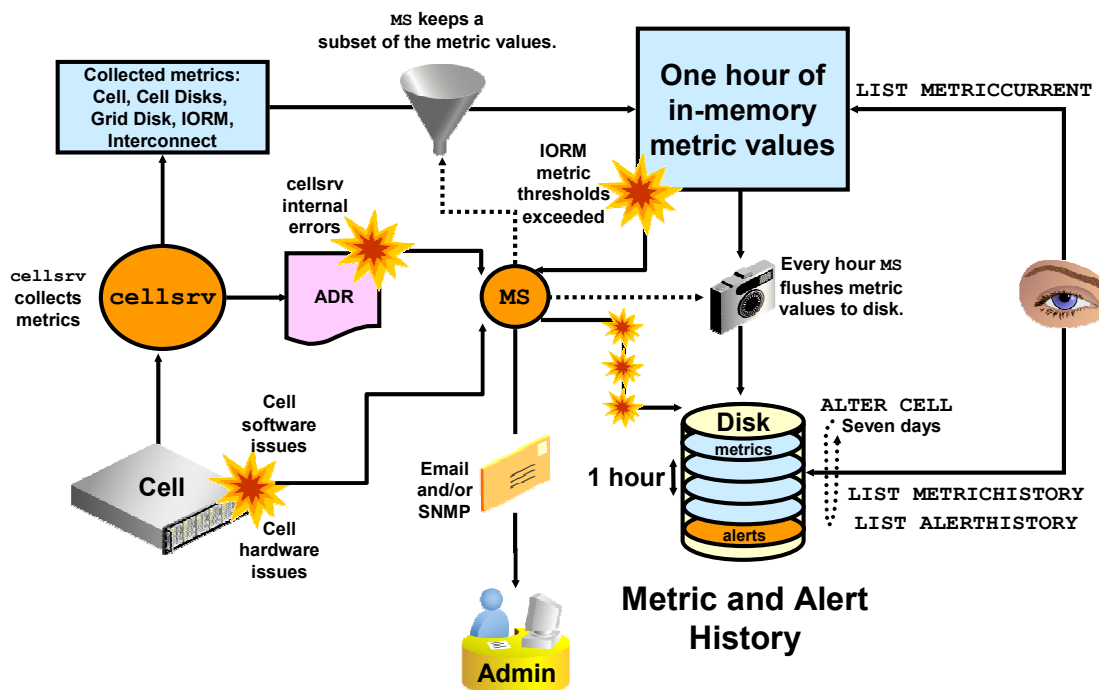


Monitoring Overview

To monitor an Oracle Exadata Cell, you can use the following tools and information:

1. Exadata cell metrics
2. Exadata cell alerts
3. Exadata active requests
4. Database SQL statement execution plans
5. Database V\$ views
6. Database wait events
7. Oracle Enterprise Manager Exadata monitoring plug-in

Exadata Metrics and Alerts Architecture



ORACLE

6 - 4

Copyright © 2009, Oracle. All rights reserved.

Exadata Metrics and Alerts Architecture

You can monitor each cell with Exadata cell metrics. `CELLSRV` periodically records important run-time properties called metrics for the cell's hardware components such as CPUs, cell disks, grid disks, flash cache, and IORM statistics. These metrics are recorded in memory.

Based on its own metric collection schedule, `MS` gets the set of metric data accumulated by `CELLSRV`. `MS` keeps a one hour subset of the metric values in its memory, and writes a history to the disk repository every hour.

The retention period for metric and alert history entries is specified by the `metricHistoryDays` cell attribute. You can modify this setting with the `CellCLI ALTER CELL` command. By default, it is seven days. This process is conceptually similar to database AWR snapshots.

You can get the metric value history by using the `CellCLI LIST METRICCHISTORY` command, and you can get the current metric values by using the `LIST METRICCURRENT` command.

At the Exadata cell level, you can define thresholds for IORM metrics. Using the Exadata Enterprise Manager plug-in, you can set separate EM thresholds for all the Exadata metrics supported by the plug-in.

Exadata Metrics and Alerts Architecture (continued)

In addition to metrics, Exadata can trigger alerts. Alerts represent events of importance occurring within the cell, typically indicating that an Exadata cell function is compromised. MS triggers an alert when it discovers a:

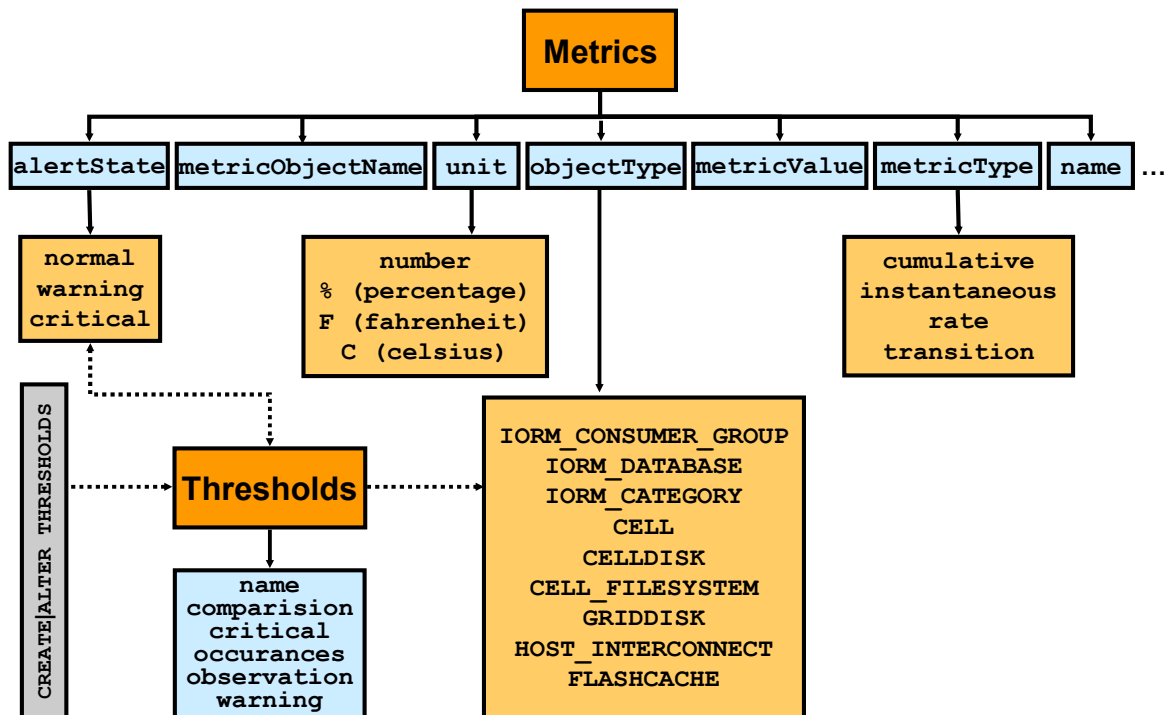
- cell hardware issue
- cell software or configuration issue
- CELLSRV internal error
- metric that has exceeded an alert threshold

You can view triggered alerts using the `LIST ALERTHISTORY` command. In addition, you can configure the cell to instruct MS to automatically send an email and/or SNMP messages to a designated set of storage administrators.



Monitoring Exadata Cells with Metrics

1



ORACLE

6 - 6

Copyright © 2009, Oracle. All rights reserved.

Monitoring Exadata Cells with Metrics

Metrics are recorded observations of important run-time properties or internal instrumentation values of the storage cell and its components, such as cell disks or grid disks. Metrics are a series of measurements that are computed and retained in memory for an interval of time, and stored on a disk for a more permanent history.

The above graphic describes some of the important metric attributes. Each metric:

- Has a name and description.
- Is associated with a **metricObjectName** that is the name of the object being measured, such as a specific cell disk, grid disk, or consumer group.
- Belongs to a group that is defined by its **objectType** attribute. The possible groups are shown in the slide.
- Has a **metricType** which is an indicator of how the statistic was created or defined. Possible values and their meanings are:
 - **cumulative**: Cumulative statistics since the metric was created
 - **instantaneous**: Value at the time that the metric is collected
 - **rate**: Rates computed by averaging statistics over observation periods
 - **transition**: Are collected at the time when the value of the metrics has changed, and typically captures important transitions in hardware status
- Has a **unit**. Possible units are shown in the slide.

Monitoring Exadata Cells with Metrics (continued)

Understanding the composition of the metric name provides a good insight into the meaning of the metric. The value of the name attribute is a composite of abbreviations. The attribute value starts with an abbreviation of the object type on which the metric is defined:

- CL_ (cell)
- CD_ (cell disk)
- GD_ (grid disk)
- FC_ (flash cache)
- DB_ (database-level)
- CG_ (consumer group, database-qualified)
- CT_ (category)
- N_ (interconnect network)

After the abbreviation of the object type, many metric names conclude with an abbreviation that relates to the description of the metric. For example, CL_FANS is the instantaneous number of working fans on the cell.

I/O related metric name attributes continue with one of the following combinations to identify the operation:

- IO_RQ (number of requests)
- IO_BY (number of MB)
- IO_TM (I/O latency)
- IO_WT (I/O wait time)

Next, in the name could be _R or _W for read or write. Following that in the name attribute value there might be _SM or _LG to identify small or large blocks, respectively. At the end of the name, there could be _SEC to signify per seconds or _RQ to signify per request. For example:

- CD_IO_RQ_R_SM is the number of requests to read small blocks on a cell disk.
- GD_IO_BY_W_LG_SEC is the rate which is the number of MB written in large blocks per second on a grid disk.

If a metric value crosses a user-defined threshold, an alert will be generated. Metrics can be associated with warning and critical thresholds. Thresholds relate to extreme values in the metric which might indicate a problem or other event of interest to an administrator. All cell metrics can have user-defined thresholds associated with them.

Users of Enterprise Manager Grid Control with the Exadata Plug-in can configure a separate set of thresholds and alerts in the Grid Control environment. These can be used in conjunction with metrics and alerts from across your systems to provide an enterprise-level view of system health and state.

Note: For a complete reference of metric and threshold attributes, refer to the *Oracle Exadata Storage Server Software User's Guide*. For more information on the Exadata Plug-in for Enterprise Manager Grid Control, refer to the *Oracle Exadata Storage Server Documentation* library.



Monitoring Exadata Cells with Metrics: Example

```
CellCLI> LIST METRICDEFINITION WHERE objectType = 'CELL' DETAIL
name: CL_CPU
description: "Cell CPU Utilization is the percentage of time over
             the previous minute that the system CPUs were not
             idle (from /proc/stat). "
metricType: Instantaneous objectType: CELL
unit: %
...
```

```
CellCLI> LIST METRICHISTORY WHERE name like 'CL_*' -
AND collectionTime > '2009-10-11T15:28:36-07:00'
CL_RUNQ    cell103_2    6.0      2009-10-11T15:28:37-07:00
CL_CPU     cell103_2    47.6 %   2009-10-11T15:29:36-07:00
CL_FANS     cell103_2    1      2009-10-11T15:29:36-07:00
CL_TEMP     cell103_2    0.0 C   2009-10-11T15:29:36-07:00
CL_RUNQ     cell103_2    5.2      2009-10-11T15:29:37-07:00
...
```

```
CellCLI> LIST METRICCURRENT WHERE objectType = 'CELDDISK'
CD_IO_TM_W_SM_RQ CD_1_cell103 205.5 us/request
CD_IO_TM_W_SM_RQ CD_2_cell103  93.3 us/request
CD_IO_TM_W_SM_RQ CD_3_cell103   0.0 us/request
...
```

ORACLE

6 - 8

Copyright © 2009, Oracle. All rights reserved.

Monitoring Exadata Cells with Metrics: Example

The slide shows you some basic commands that you could use to display metric information:

- Use the `LIST METRICDEFINITION` command to display the metric definitions for the cell. A metric definition describes the configuration of a metric. The example does not specify any particular metric, so all metrics corresponding to the `WHERE` clause are printed. In addition to the `WHERE` clause, you can also specify the metric attributes you want to print. If the `ATTRIBUTES` clause is not used, a default set of attributes is displayed. To list attributes, you can add the `DETAIL` keyword at the end of the command.
- Use the `LIST METRICHISTORY` command to display the metric history for the cell. A metric history describes a collection of past metric observations. Similar to the `LIST METRICDEFINITION` command, you can specify attribute filters, an attribute list, and the `DETAIL` keyword for the `LIST METRICHISTORY` command. The above example lists metrics having names that start with `CL_` that were collected after the specified time.
- Use the `LIST METRICCURRENT` command to display the current metric values for the cell. The above example lists all cell disk metrics. The metric values shown in the slide correspond to the average latency of writing small blocks per request to a cell disk. You have such a metric for every cell disk.



Using Database I/Os Metrics: Example

- Use metric with `objectType = IORM_DATABASE`.
- Separate metrics for small (≤ 128 KB) and large I/Os
- Which database has the heaviest load?
 - Look for highest `DB_IO_RQ_SM` + `DB_IO_RQ_LG` values.
- Which database was throttled the most?
 - Look for highest `DB_IO_WT_SM` + `DB_IO_WT_LG` values.

Name	Description
DB_IO_RQ_SM DB_IO_RQ_LG	Total number of I/O requests issued by this database since <i>any</i> resource plan was set
DB_IO_RQ_SM_SEC DB_IO_RQ_LG_SEC	I/O requests per second issued by this database in past minute
DB_IO_WT_SM DB_IO_WT_LG	Total number of seconds that I/O requests issued by this database waited to be scheduled

ORACLE

6 - 9

Copyright © 2009, Oracle. All rights reserved.

Using Database I/Os Metrics: Example

Exadata provides three groups of I/O metrics that correspond to the three types of IORM plans: category metrics, database metrics, and consumer group metrics.

This slide focuses on database metrics however the same principles apply for category metrics and consumer group metrics.

For database metrics the `objectType` attribute is set to `IORM_DATABASE`. For each database I/O metric, a distinction is made regarding small I/Os (SM) and large I/Os (LG). The above table gives you a quick description of some important metrics in the `IORM_DATABASE` group.

A separate set of metric observations is available for each database specified in the IORM plan. You can compare metrics between databases to determine which one has the heaviest load or which one was throttled most as illustrated in the slide.

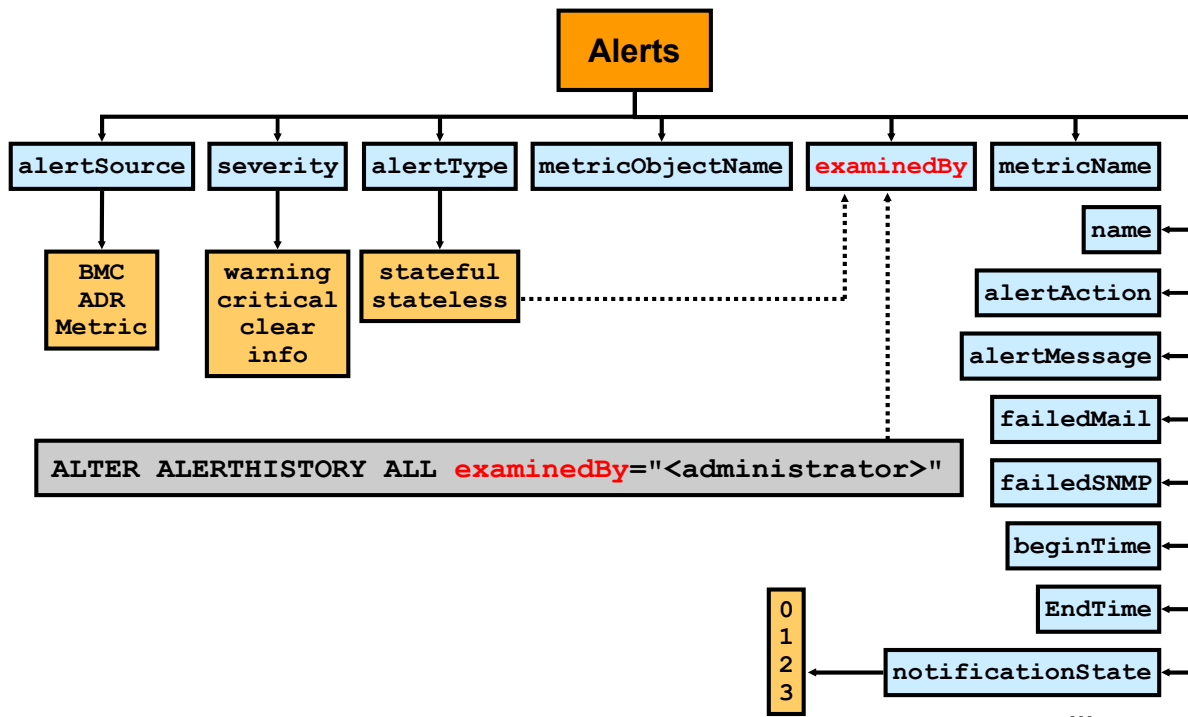
You can do the same type of comparisons for consumer groups and categories using different sets of I/O metrics.

Note: A special `metricObjectName` value of `_OTHER_DATABASE_` is used for database I/O metrics associated with ASM and for databases that are not explicitly mentioned in the interdatabase IORM plan.



Monitoring Exadata Cells with Alerts

2



6 - 10

Copyright © 2009, Oracle. All rights reserved.

ORACLE

Monitoring Exadata Cells with Alerts

Alerts represent events of importance occurring within the storage cell, typically indicating that storage cell functionality is either compromised or in danger of failure. An administrator should investigate alerts, because they might require urgent corrective or preventive action. Use the `ALTER CELL` command to configure email or SNMP notification for alerts.

Alerts are either stateful or stateless. Stateful alerts represent observable cell states that can be subsequently retested to detect whether the state has changed, so that a previously observed alert condition is no longer a problem. Stateless alerts represent point-in-time events that do not represent a persistent condition; they simply show that something has occurred.

Alerts can have one of the following severities: warning, critical, clear or info. Examples of possible alerts that can be used are physical disk failure, disk read/write errors, cell temperature exceeding recommended value, Exadata Cell software failure, and excessive I/O latency.

Metrics can be used to signal stateful alerts using warning or critical threshold values. When the metric value crosses the threshold value, an alert can be signaled. An alert with a clear severity indicates that a previous critical or warning condition has returned to normal. For threshold based alerts, a clear alert is generated when the measured value crosses back over the threshold value.

Alerts with an info severity are stateless and log conditions that might be informative to an administrator but for which no administrator action is required. Informational alerts are not distributed by email or SNMP notifications.

Monitoring Exadata Cells with Alerts (continued)

The slide illustrates some of the important alert attributes. Each alert has the following attributes:

- `name` provides an identifier for the alert.
- `alertSource` provides the source of the alert. Some possible sources are listed in the slide.
- `severity` determines the importance of the alert. Possible values are `warning`, `critical`, `clear` and `info`.
- `alertType` provides the type of the alert: `stateful` or `stateless`. Stateful alerts are automatically cleared on transition to normal. Stateless alerts are never cleared unless you change the alert by setting the `examinedBy` attribute. This attribute identifies the administrator who reviewed the alert and is the only alert attribute that can be modified by the administrator using the `ALTER ALERTHISTORY` command.
- `metricObjectName` is the object for which a metric threshold has caused an alert.
- `metricName` provides the metric name if the alert is based on a metric.
- `alertAction` is the recommended action to perform for this alert.
- `alertMessage` provides a brief explanation of the alert.
- `failedMail` is the intended email recipient when a notification failed.
- `failedSNMP` is the intended SNMP subscriber when a notification failed.
- `beginTime` provides the timestamp when an alert changes its state.
- `endTime` provides the timestamp for the end of the period when an alert changes its state.
- `notificationState` indicates progress in notifying subscribers to alert messages:
 - 0: never tried
 - 1: sent successfully
 - 2: retrying (up to 5 times)
 - 3: five failed retries

Note: Some I/O errors may result in an ASM disk going offline without generating an alert in Exadata. You should continue to perform I/O monitoring from your databases and ASM environments to identify and remedy these kinds of problems.



Displaying Alert Examples

```
CellCLI> LIST ALERTDEFINITION ATTRIBUTES name, metricName, description
ADRAAlert "CELL Incident Error"
HardwareAlert "Hardware Alert"
StatefulAlert_CG_IO_RQ_LG CG_IO_RQ_LG "Threshold Based Stateful Alert"
StatefulAlert_CG_IO_RQ_LG_SEC CG_IO_RQ_LG_SEC "Threshold Based ..Alert"
StatefulAlert_CG_IO_RQ_SM CG_IO_RQ_SM "Threshold Based Stateful Alert"
...
```

```
CellCLI> LIST ALERTHISTORY WHERE severity = 'critical' -
AND examinedBy = '' DETAIL
```

```
CellCLI> ALTER ALERTHISTORY 1671443814 examinedBy="JFV"
```

```
CellCLI> CREATE THRESHOLD ct_io_wt_lg_rq.interactive -
warning=1000, critical=2000, comparison='>', -
occurrences=2, observation=5
```

ORACLE

Displaying Alert Examples

The slide shows you some examples of commands that display alert information. The commands for displaying alerts are very similar to the ones used for displaying metric information:

- Use the `LIST ALERTDEFINITION` command to display the alert definitions for the cell. An alert definition provides a definition for every alert that can be produced on the cell. The above example displays the alert name, metric name, and description. The metric name identifies the metric on which the alert is based. `ADRAAlert` and `HardwareAlert` are not based on any metric and, therefore, do not have metric names.
- Use the `LIST ALERTHISTORY` command to display the alert history that has occurred on a cell. The above example lists in detail all critical alerts that have not been reviewed by an administrator.
- Use the `ALTER ALERTHISTORY` command to update the alert history for the cell. The above example shows how to set the `examinedBy` attribute to the user ID of the administrator that examined the alert. The `examinedBy` attribute is the only `ALERTHISTORY` attribute that can be modified. The example uses the alert sequence ID to identify the alert. `alertSequenceID` provides a unique sequence ID number for the alert. When an alert changes its state, another occurrence of the alert is created with the same sequence number but with a different timestamp.

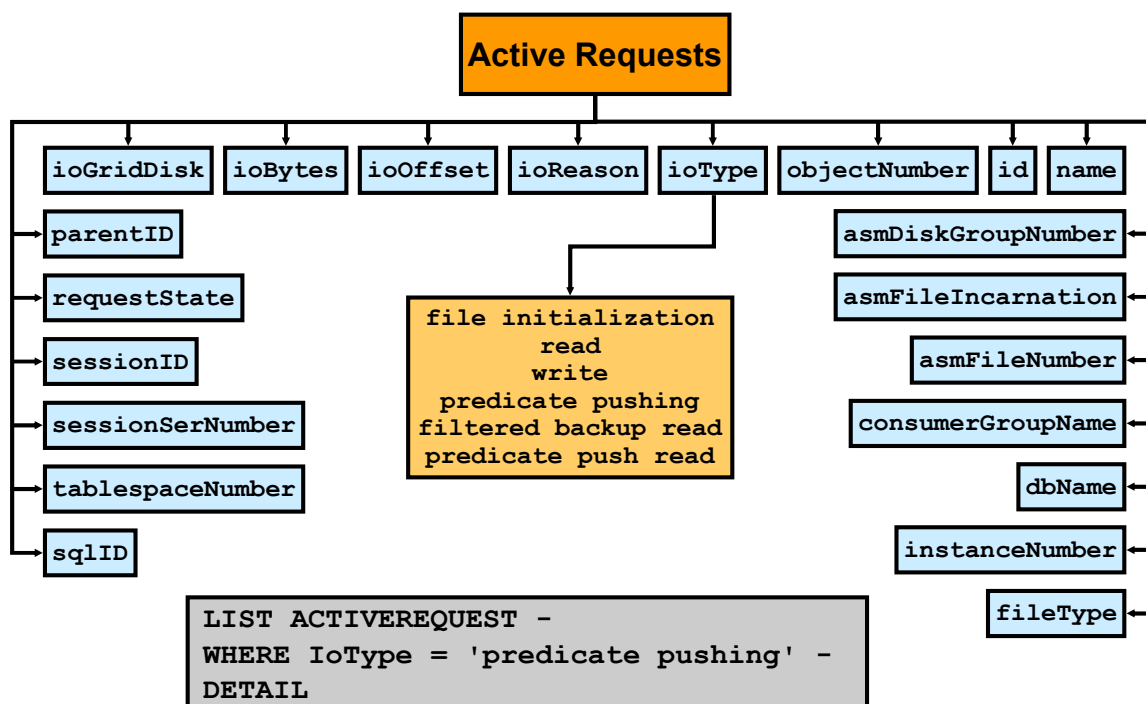
Displaying Alert Examples (continued)

- The `CREATE THRESHOLD` command creates a threshold that specifies the conditions for generation of an IORM metric alert. The example creates a threshold for the `CT_IO_WT_LG_RQ` metric associated with the `INTERACTIVE` category. This metric specifies the average number of milliseconds that large I/O requests issued by the category have waited to be scheduled by IORM in the past minute. A large value indicates that the I/O workload from this category is exceeding the allocation specified for it in the category plan. The alert is triggered by two consecutive measurements over the threshold values; one second for a warning alert and two seconds for a critical alert. The observation attribute is the number of measurements over which measured values are averaged.



Monitoring Exadata Cells with Active Requests

3



ORACLE

6 - 14

Copyright © 2009, Oracle. All rights reserved.

Monitoring Exadata Cells with Active Requests

An active request provides a client-centric or application-centric view of client I/O requests that are currently being processed by a cell.

The slide shows the most important attributes of an active request. You can see that an active request is characterized at all levels: instance, database, ASM, and cell. Most of the attributes have self-explanatory names. Here is a brief explanation of some of the attributes:

- `ioReason` is the reason for the I/O activity, such as a control-file read.
- `ioType` identifies the type of active request. Possible values are listed on the slide.
- `requestState` identifies the state of the active request. Possible values include:
 - Accessing Disk
 - Computing Result
 - Network Receive
 - Network Send
 - Queued Extent
 - Queued for Disk
 - Queued for File Initialization
 - Queued for Filtered Backup Read
 - Queued for Network Send
 - Queued for Predicate Pushing
 - Queued for Read
 - Queued for Write
 - Queued in Resource Manager

Use the `LIST ACTIVEREQUEST` command to display active request details for the cell. The syntax is very similar to other `LIST` commands. You can specify which attributes to display or you can display them all using the `DETAIL` clause. You can also filter the output using a `WHERE` clause.



Monitoring SQL Execution Plans

4

Relevant Initialization Parameters:

- `CELL_OFFLOAD_PROCESSING`
 - `TRUE` | `FALSE`
 - Enables or disables smart scan and other smart storage capabilities
 - Dynamically modifiable at the session or system level using `ALTER SESSION` or `ALTER SYSTEM`
 - Specifiable at the statement level using the `OPT_PARAM` hint
- `CELL_OFFLOAD_PLAN_DISPLAY`
 - `NEVER` | `AUTO` | `ALWAYS`
 - Allows execution plan to show offloaded predicates
 - Dynamically modifiable at the session or system level using `ALTER SESSION` or `ALTER SYSTEM`

ORACLE

6 - 15

Copyright © 2009, Oracle. All rights reserved.

Monitoring SQL Execution Plans

The `CELL_OFFLOAD_PROCESSING` initialization parameter enables SQL processing offload to Exadata. The default value of the parameter is `TRUE` which means that predicate evaluation can be offloaded to Exadata. If set to `FALSE`, then the database performs all the predicate evaluation with cells serving blocks like traditional storage. To enable offloading for a particular SQL statement, use the `OPT_PARAM` hint as shown in the following example:

```
SELECT /*+ OPT_PARAM('cell_offload_processing' 'true') */ ...
```

The `CELL_OFFLOAD_PLAN_DISPLAY` initialization parameter determines whether the `SQL EXPLAIN PLAN` statement displays the predicates that can be evaluated by Exadata as `STORAGE` predicates for a given SQL statement. The possible values are:

- `AUTO` instructs the `SQL EXPLAIN PLAN` statement to display the predicates that can be evaluated as `STORAGE` only if a cell is present and if a table is on the cell.
- `ALWAYS` produces changes to the `SQL EXPLAIN PLAN` statement whether or not Exadata is present or the table is on the cell. You can use this setting to identify statements that are candidates for offloading before migrating to Exadata.
- `NEVER` produces no changes to the `SQL EXPLAIN PLAN` statement due to Exadata. This may be desirable, for example, if you wrote tools that process execution plan output and these tools have not been updated to deal with new syntax or when comparing plans for two systems: one with Exadata, and one without.



Smart Scan Execution Plan Example

```
SQL> alter session set CELL_OFFLOAD_PROCESSING = TRUE;
Session altered.
SQL> alter session set CELL_OFFLOAD_PLAN_DISPLAY = ALWAYS;
Session altered.
SQL> explain plan for select * from customers where c_customer_sk < 10;
Explained.
SQL> select * from table(dbms_xplan.display);
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)
0	SELECT STATEMENT		1	196	326 (1)
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10000	1	196	326 (1)
3	PX BLOCK ITERATOR		1	196	326 (1)
* 4	TABLE ACCESS STORAGE FULL	CUSTOMER	1	196	326 (1)

Predicate Information (identified by operation id):

```
4 - storage("C_CUSTOMER_SK"<10) filter("C_CUSTOMER_SK"<10)
```

ORACLE

6 - 16

Copyright © 2009, Oracle. All rights reserved.

Smart Scan Execution Plan Example

Smart scan is enabled by default for direct read operations. So Exadata only uses direct reads, not the buffer cache, to process queries that can be offloaded.

Exadata optimization is a run-time decision, and it is not integrated with the Oracle optimizer. So offloading is only possible for full scans and is only available with segments stored on disk groups that are completely hosted on Exadata storage.

If Exadata is not sure that the a block is current, it transfers the read of that block to the traditional buffer cache/read consistency path. So if you run updates at the same time as queries, you will benefit less from smart scan than if you were executing a read-only workload. This is also true for indirect rows.

The slide shows an example of SQL processing offload manifested in a query plan. The first command enables offloading for the session. The second command enables storage predicates to be showed in the SQL execution plans of the session, even if Exadata is not present.

At the bottom of the plan output, you can see the STORAGE operation indicating the predicate ("C_CUSTOMER_SK"<10) being offloaded to Exadata storage.

Note: Smart scan is also available for index fast full scans, not just table scans. Also, you cannot see column projection in a query plan.



Predicate Offloading Considerations

- Predicate evaluation is not offloaded when:
 - `CELL_OFFLOAD_PROCESSING` is set to `FALSE`
 - The table or partition being scanned is small
 - The optimizer decides not to use direct path reads
 - A scan is performed on a clustered table
 - The table has row dependencies enabled or the `rowscn` is being fetched
 - The optimizer wants the scan to return rows in `ROWID` order
 - The command is `CREATE INDEX` using `NOSORT`
 - A `LOB` or `LONG` column is being selected or queried
 - A scan is performed on a flashback table
 - The data is encrypted and cell-based decryption is disabled
 - The tablespace is not completely stored on Exadata storage
 - A statement involves columns outside the first 255 columns
 - The predicate evaluation is on a virtual column

ORACLE

6 - 17

Copyright © 2009, Oracle. All rights reserved.

Predicate Offloading Considerations

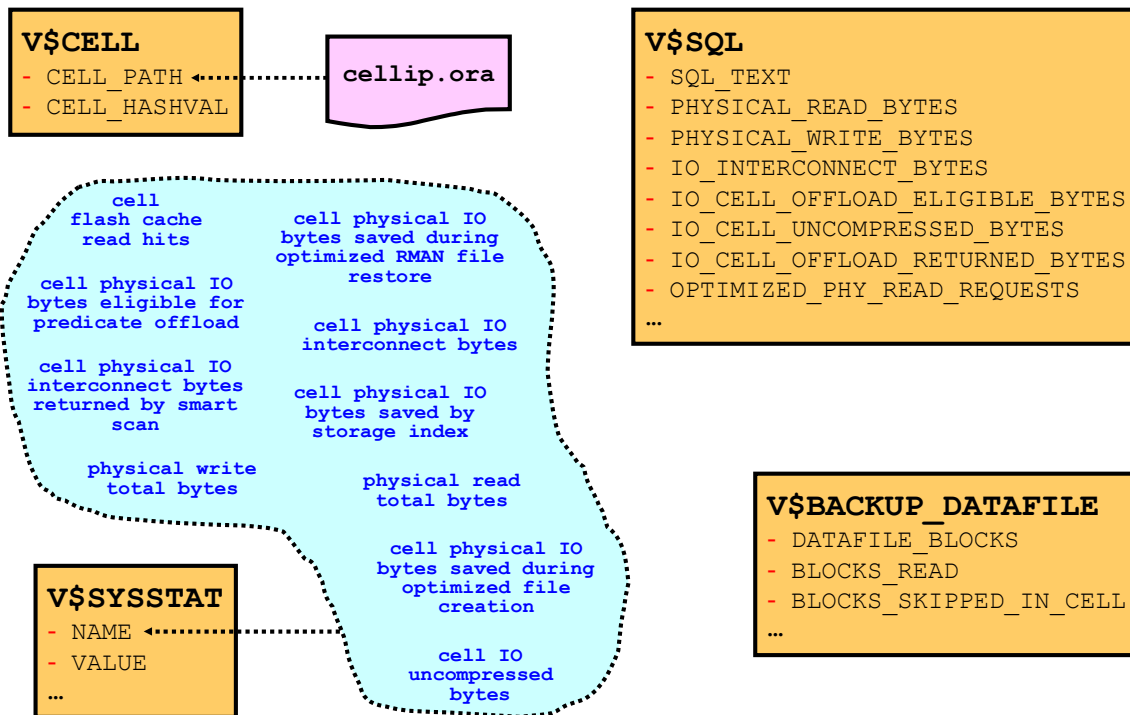
The slide lists the cases where predicate evaluation is not offloaded to Exadata. The following provides additional information for some of these cases:

- The optimizer decides not to use direct path reads: Direct path reads are mostly used by parallel operations. Serial operations can do direct reads too, depending on factors such as the table size and the state of the buffer cache. Direct path reads can also be forced for serial access by setting `_serial_direct_read` to `TRUE`.
- The data is encrypted and cell-based decryption is disabled: In order for Exadata to perform decryption, Oracle Database needs to send the decryption key to the Exadata cell. If there are security concerns about keys being shipped across the network, then you can disable cell-based decryption by setting the `CELL_OFFLOAD_DECRYPTION` parameter to `FALSE`.



Monitoring Exadata Cells from Your Database

5



Monitoring Exadata Cells from Your Database

You can use the above V\$ views and corresponding statistics to monitor Exadata from a database instance:

- V\$CELL provides the cell IP address extracted from the cellip.ora file. It also contains a numeric hash value for the cell which is used as an identifier for the cell in other views, such as V\$SESSION_WAIT and V\$ACTIVE_SESSION_HISTORY.
- V\$BACKUP_DATAFILE contains columns relevant to Exadata during RMAN incremental backups. The BLOCKS_SKIPPED_IN_CELL indicates the number of blocks that were read and filtered at the Exadata cell to optimize the RMAN incremental backup.
- V\$SYSSTAT and V\$SESSTAT contain key statistics that can be used to compute Exadata cell effectiveness at both the system and session level. Statistics in these views can be used to monitor the effectiveness of flash cache, Exadata Hybrid Columnar Compression, SQL offloading, storage indexes, fast file creation, and optimized incremental backups. In addition, the views provide the total volume of I/O exchanged over the interconnect and the total volume of physical disk reads and writes.
- V\$SQL lists statistics on shared SQL areas. It contains statement-level statistics for the volume of physical I/O (reads and writes), the volume of I/O exchanged over the interconnect, along with information relating to the effectiveness of flash cache, Exadata Hybrid Columnar Compression, and SQL offloading.

Note: For more information refer to the *Oracle Exadata Storage Server Software User's Guide*.



Assessing Smart Scan Efficiency: Example

```
SQL> SELECT b.name, a.value FROM v$mystat a, v$statname b
       WHERE a.STATISTIC# = b.STATISTIC# AND
       b.name = 'cell session smart scan efficiency';
```

NAME	VALUE
cell session smart scan efficiency	11.9

ORACLE

6 - 19

Copyright © 2009, Oracle. All rights reserved.

Assessing Smart Scan Efficiency: Example

You can determine smart scan efficiency by querying the `cell session smart scan efficiency` statistic from `V$SESSSTAT` (for all sessions) or `V$MYSTAT` (for the current session only). The efficiency is essentially the ratio between the amount of data referenced in the cell, and the amount of data returned. If there is no offloading, then the metric will be 1. The statistic is defined as follows:

$$\frac{(\text{cell IO uncompressed bytes} + \text{cell physical IO bytes saved by storage indexes})}{\text{cell physical IO interconnect bytes returned by smart scan}}$$

The slide shows an example of assessing smart scan efficiency for the current session. Essentially, the result shows that for all statements where smart scan was used, the amount of I/O returned from the cell was less than 12% of the total that would have been returned without smart scan. In other words, more than 88% of the I/O volume that would have been required without smart scan never needed to be transported to the database server.

Note: It is also possible to use SQL Performance Analyzer to assess smart scan efficiency using the `IO_INTERCONNECT_BYTES` statistics.



Monitoring Exadata Cells with Wait Events

6

```
SELECT w.event, c.cell_path, d.name, w.p3
FROM   V$SESSION_WAIT w, V$EVENT_NAME e, V$ASM_DISK d, V$CELL c
WHERE  e.name LIKE 'cell%' AND e.wait_class_id = w.wait_class_id
       AND w.p1 = c.cell_hashval AND w.p2 = d.hash_value;
```

Wait Event	Description
cell interconnect retransmit during physical read	Database wait during retransmission for an I/O of a single-block or multiblock read
cell single block physical read	Cell equivalent of db file sequential read
cell multiblock physical read	Cell equivalent of db file scattered read
cell smart table scan	Database wait for table scan to complete
cell smart index scan	Database wait for index or IOT fast full scan
cell smart file creation	Database wait for file creation operation
cell smart incremental backup	Database wait for incremental backup operation
cell smart restore from backup	Database wait during file initialization for restore
cell statistics gather	Wait during query of V\$CELL views

ORACLE®

6 - 20

Copyright © 2009, Oracle. All rights reserved.

Monitoring Exadata Cells with Wait Events

Oracle uses a specific set of wait events for disk I/O to Exadata cells that identifies the corresponding cell and grid disk being accessed. This information is more useful for performance and diagnostics purposes than the database file number and block number information that is provided by wait events for conventional storage.

Information about wait events is displayed in V\$ dynamic performance views such as V\$SESSION_WAIT, V\$SYSTEM_EVENT and V\$SESSION_EVENT.

The slide shows an example of a query used to display the cell IP address and disk name corresponding to cell wait events. A list of cell wait events with a brief description is also shown. Most of the cell wait events are self-explanatory.

The cell statistics gather event is a little different. It appears when a select is done on the V\$CELL_STATE, V\$CELL_THREAD_HISTORY, or V\$CELL_REQUEST_TOTALS views. During the select, data from the cells and any wait events are shown in this wait event. Normally, these V\$CELL views are only used by Oracle Support Services.

Note: For more information about these wait events, refer to the *Oracle Exadata Storage Server Software User's Guide*.

Monitoring Exadata Cells with Enterprise Manager

7

The screenshot displays the Oracle Enterprise Manager interface for monitoring an Exadata Storage Cell. The left-hand navigation pane shows the 'Configuration' tab selected, with 'View Configuration' and 'All Metrics' links highlighted. The right-hand pane shows the 'Reports' tab, where a dropdown menu is open, listing various performance metrics such as 'Realm Performance', 'Cell Performance', 'Cell Disk Performance', 'Griddisk Performance', 'LUN Performance', 'Physical Disk Performance', 'IORM Database Performance', 'IORM Category Performance', 'IORM Consumer Group Performance', 'Cell Configuration', and 'Cell Disk Configuration'. The main area of the interface displays a table of metrics for the 'std3s3test' cell, including CPU Busy, Database Small IO Requests, Run Queue Length, and CG Small IO Requests. The table has columns for Metric, Severity, Alert Triggered, and Last V.

Metric	Severity	Alert Triggered	Last V
IORM Boost for std3s3test	×	Sep 19, 2008 7:04:18 AM	
CPU Busy (%) for std3s3test	×	Sep 19, 2008 6:28:22 AM	2
Database Small IO Requests/Sec for _OTHER_DATABASE_/std3s3test	!	Sep 19, 2008 8:11:42 AM	
Run Queue Length for std3s3test	!	Sep 19, 2008 8:08:21 AM	
CG Small IO Requests/Sec for _OTHER_DATABASE_/_ORACLE_BACKGROUND_GROUP_/std3s3test	!	Sep 19, 2008 8:06:34 AM	

Monitoring Exadata Cells with Enterprise Manager

The System Monitoring Plug-In for Oracle Exadata Storage Server extends Grid Control to add support for managing Exadata storage server targets. By deploying the plug-in to your Grid Control environment, you gain the following management features:

- Monitor Exadata storage servers, individually or in groups, using a GUI environment.
- Gather storage configuration and performance information of various Exadata related storage components such as grid disks and cell disks.
- Raise alerts and violations based on thresholds set for monitoring and configuration data.
- Provide rich out-of-the-box metrics and reports based on the gathered data.

This plug-in supports the following versions of products:

- Oracle Exadata Storage Server 11g Release 2 (11.2) and later
- Enterprise Manager Grid Control 10g Release 3 (10.2.0.3) and later (OMS and Agent)

The plug-in requires SSH connectivity between the `celladmin` user on the Exadata cells and the Management Agent user on the computer running the Management Agent. Typically the agent on the OMS server is used to monitor Exadata, but another agent can be used.

Before you set up alerts in Grid Control, you must configure the Exadata cells to send SNMP alerts to the Management Agent that is monitoring them.

Note: Refer to the *Enterprise Manager System Monitoring Plug-In Installation Guide for Sun Oracle Exadata Storage Server Release 10 (1.1.4.0.0)* for more information about the plug-in.

Optimizing Performance

- Start with best practises for ASM and Oracle Database
- Areas for special consideration:
 - Flash memory usage
 - Compression usage
 - Index usage
 - ASM allocation unit size
 - Minimum extent size

ORACLE

6 - 22

Copyright © 2009, Oracle. All rights reserved.

Optimizing Performance

Optimizing the performance of your Exadata environment begins with following good practises for database design and application development. From an administration viewpoint, you should continue to follow the best practises for ASM and Oracle Database in conjunction with advise and statistics provided by tools such as SQL monitor and SQL Tuning Advisor.

In addition, there are a number of areas for special consideration listed on the slide. These are covered in the remainder of this lesson.

Flash Memory Usage

- Each Exadata storage server contains 384 GB of high performance flash memory.
- Flash memory can be configured two ways:
 - Exadata Smart Flash Cache:
 - Speeds up access to frequently accessed data
 - Uses all available flash memory by default
 - Managed automatically for maximum efficiency
 - Users can only provide optional hints to influence caching priorities
 - Beneficial for OLTP and Data Warehouse workloads
 - Flash-based permanent storage:
 - Uses flash memory as a premium storage device
 - Must be configured by an administrator
 - Requires deliberate planning to use efficiently
 - Additional benefits for specific applications



Flash Memory Usage

Each Exadata cell contains 384 GB of high performance flash memory.

By default, when an Exadata cell is first configured, this flash memory is configured as Exadata Smart Flash Cache. Exadata Smart Flash Cache provides a caching mechanism for frequently-accessed data on each Exadata cell. It is a write-through cache which is useful for absorbing repeated random reads, and very beneficial to online transaction processing (OLTP). Exadata Smart Flash Cache also benefits data warehousing applications by caching the most frequently accessed data.

Exadata Smart Flash Cache automatically focuses on caching frequently accessed data and index blocks, along with performance critical information such as control files and file headers. In addition, DBA's can influence caching priorities using the `CELL_FLASH_CACHE` storage attribute for specific database objects.

For example, to prioritize caching for the `CALLDETAIL` table use the following command:

```
ALTER TABLE calldetail STORAGE (CELL_FLASH_CACHE KEEP)
```

Because of this setting, Exadata will cache data for the `CALLDETAIL` table more aggressively than otherwise. Without this setting, Exadata caches individual table and index blocks. With this setting, Exadata attempts to cache scans on the `CALLDETAIL` table. This makes it more likely that subsequent scans on the `CALLDETAIL` table will also be satisfied using Exadata Smart Flash Cache.

Flash Memory Usage (continued)

This setting also causes Exadata clear out other tables from the cache before clearing out cached data for the `CALLDETAIL` table. This means that `CALLDETAIL` records will remain in the in Exadata Smart Flash Cache longer than cached data for other tables.

If the `CALLDETAIL` table is spread across several Sun Oracle Exadata Storage Servers, then each one will cache its part of the table in its own Exadata Smart Flash Cache. If the caches are sufficient in size, then the `CALLDETAIL` table is likely to be completely cached over time.

Alternatively, you can configure a smaller Exadata Smart Flash Cache and use the remaining flash memory as a flash disk drive. You can then create grid disks based on flash memory. These flash disks can be used to configure high performance ASM disk groups housing their own tablespaces. In this case the flash memory is not used to cache data, rather it becomes a high performance permanent store for some of your data. You should consider configuring flash-based permanent storage areas in the following cases:

- **Heavy write applications:** For applications that perform many updates, or in cases of heavy data ingest, you may consider storing specific tables, partitions or indexes on flash disks. In these cases the high volume of writes can consume a considerable portion of the physical disk bandwidth. Storing insert and update intensive tables, partitions and indexes on flash-based storage means that there are no physical disk I/Os associated with those objects. The physical disk bandwidth saved using this configuration can be used by other operations increasing the overall throughput of the system. Flash disks can also be used effectively in conjunction with partitioning. In many time based partitioning scenarios, the most current data can be quite volatile while older data is subject to few updates, if any. In these situations, you can host your most current partitions in a flash-based tablespace and routinely move these partitions to a disk based tablespace as their volatility diminishes due to age.
- **Star-schemas:** In a data warehouse based on a star-schema you may be able to store your entire fact table on flash disk. This effectively acts as a permanent cache for this key table. If your fact table is too large to store on flash disk you can partition it and store the most referenced partition on flash disk.

The flash memory on each cell can be divided between Exadata Smart Flash Cache and permanent flash-based storage. Both areas can be any size as long as the total does not exceed 384 GB on each cell. In practise, the effectiveness of Exadata Smart Flash Cache drops if the cache size is very small so in most cases you would allocate a substantial proportion of flash memory to Exadata Smart Flash Cache.

Note that Exadata Smart Flash Cache is largely automatic. Once it is configured you can only influence caching priorities for specific tables using the `CELL_FLASH_CACHE` storage attribute. On the other hand, configuring permanent flash-based storage requires accurate space planning to make the most effective use of this premium resource.

Compression Usage

Compression Method	Compression Ratio	CPU Overhead	Applications
BASIC	High for direct path inserts Conventional path inserts and updates are not compressed	Minimal: Compression and decompression occur inside the Oracle database	DSS
OLTP	High for all transaction types	Minimal: Compression and decompression occur inside the Oracle database, asynchronous compression for DML	OLTP and DSS
QUERY	Higher for direct path inserts High for conventional path inserts and updates	Higher: Decompression performed inside Exadata cell	DSS
ARCHIVE	Highest for direct path inserts High for conventional path inserts and updates	Highest: Decompression performed inside Exadata cell	Archiving

ORACLE

6 - 25

Copyright © 2009, Oracle. All rights reserved.

Compression Usage

Many people think of data compression as they approach the capacity limits of their storage. Commonly, data compression is seen as a performance overhead which must be tolerated in order to deliver extra storage capacity. However, in cases where I/O bandwidth is limited, data compression can be an effective tool to increase performance by using available CPU capacity to effectively increase the I/O throughput of a storage system.

Oracle Database provides the following modes of data compression:

- BASIC compression compresses data at the Oracle block level. It allows more data to be stored in each block by replacing duplicate values with a smaller symbolic representation. For example, if the numeric value 99999 was present 50 times in a block of data it could be replaced by 50 occurrences of the # symbol along with an entry in a symbol table (also stored in the block) defining the use of the symbol. The degree of compression depends on the level of duplication in each block. Higher compression ratios can be achieved by using larger block sizes or by sorting data to increase the coincidence of duplicate values. Data remains row-major format where the columns in each row are stored together. Compression occurs when data is loaded using a direct path load operation such as CREATE TABLE AS SELECT, or direct path SQL*Loader. Tables that use BASIC compression support DML operations however any data that is inserted or updated remains uncompressed. BASIC compression is useful in decision support systems (DSS) where the compressed data is subject to minimal change.

Compression Usage (continued)

- OLTP compression uses the same compression technique as BASIC compression but provides support for compressing transactional data. This makes it useful in DSS and online transaction processing (OLTP) environments. Instead of compressing changes as they are written, data blocks are periodically compressed by a background process. This means that compression can be implemented on OLTP systems without adversely impacting the transactional throughput of the system.

In addition to the BASIC and OLTP compression modes provide by the Oracle Database, Exadata storage server provides Exadata Hybrid Columnar Compression. Exadata Hybrid Columnar Compression technology uses a modified form of columnar storage instead of row-major storage. Sets of rows are stored in an internal structure called a compression unit which is typically 4 Oracle blocks in size. Within each compression unit, the values for each column are stored together along with metadata that maps the values to the rows. Compression is achieved by replacing repeating values with smaller symbolic references. Because a compression unit is much larger than an Oracle block, and because column organization brings similar values together, Exadata Hybrid Columnar Compression can deliver much better compression ratios than both BASIC and OLTP compression. The best rates of compression are achieved using direct path loads.

Exadata Hybrid Columnar Compression provides a choice of compression modes to achieve the proper trade-off between disk usage and CPU overhead:

- QUERY compression: This type of compression is optimized for query performance, and is intended for DSS and data warehouse applications.
- ARCHIVE compression: This type of compression is optimized for maximum compression ratios, and is intended for historical data and data that does not change.

Exadata Hybrid Columnar Compression supports DML operations on compressed data however such operations attract a considerable CPU overhead and yield a lower compression ratio than for direct path loads. In addition, DML on tables using Exadata Hybrid Columnar Compression requires the entire compression unit to be locked which may reduce concurrency. Finally, updates to rows using Exadata Hybrid Columnar Compression cause rowids to change. As a result, Exadata Hybrid Columnar Compression is recommended for situations where data changes are infrequent or where data sets are reloaded rather than substantially changed.

In conclusion, Exadata Hybrid Columnar Compression makes effective use of Exadata storage server hardware to deliver the highest levels of compression for data in an Oracle Database. It is best suited in cases where the data is not subject to substantial change. For transactional data sets you should consider OLTP compression instead of Exadata Hybrid Columnar Compression. In all cases you should be aware of the relative merits and overheads associated with each compression type in order to choose the best approach for your situation.

Index Usage

- With Exadata, scans might perform better than index lookups.
- Consider removing indexes where scans deliver acceptable performance.
- Removing unnecessary indexes will:
 - Improve DML performance.
 - Save storage space.
- Test the effect of removing indexes by making them invisible:

```
SQL> ALTER INDEX <index_name> INVISIBLE;
```

ORACLE

6 - 27

Copyright © 2009, Oracle. All rights reserved.

Index Usage

Some queries that require indexes when using conventional storage will perform as well, or better, without indexes using Exadata. Review your application execution plans that use indexes to determine if they would run faster with Exadata scans. To determine if queries would be faster without an index, you can make the index invisible to the optimizer. An invisible index is maintained by DML operations, but it is not be used by the optimizer for queries. To make an index invisible, use the following command:

```
ALTER INDEX <index_name> INVISIBLE
```

Removing unnecessary indexes will aid the performance of DML operations and will save storage space.

ASM Allocation Unit Size

- By default, ASM uses an allocation unit (AU) size of 1 MB
- For Exadata, the recommended AU size is 4 MB
 - AU size must be set when a disk group is created
 - AU size cannot be altered after a disk group is created
 - AU size is set using the `AU_SIZE` disk group attribute:

```
SQL> CREATE DISKGROUP data NORMAL REDUNDANCY
      DISK 'o/*/data_CD*'
      ATTRIBUTE 'compatible.rdbms' = '11.2.0.0.0',
                'compatible.asm' = '11.2.0.0.0',
                'cell.smart_scan_capable' = 'TRUE',
                'au_size' = '4M';
```

ORACLE

ASM Allocation Unit Size

To achieve fast disk scan rates, it is important to lay out segments with at least 4 MB of contiguous space. This allows disk scans to read 4 MB of data before performing another seek at a different location on disk. To ensure segments are laid out with 4 MB of contiguous space, set the Oracle ASM allocation unit size to 4 MB. The allocation unit can be set with the disk group attribute `AU_SIZE` when creating the disk group.

Minimum Extent Size

- For Exadata, large segments should initial extents of 8 MB:
 - Stops needless proliferation of small extents in the database.
 - Minimizes sub-optimal reads for small extents.
 - Compliments the 4 MB ASM AU size recommendation.
- Large initial extents can be specified using the:
 - `INITIAL` storage parameter for specific objects or for all objects in a locally managed tablespace.
 - `CELL_PARTITION_LARGE_EXTENTS` initialization parameter for all new segments in a partitioned object:
 - `TRUE`: Sets an 8 MB minimum extent size, but will trim last unloaded extents to 64 KB boundary for direct loads.
 - `ALWAYS`: Sets an 8 MB minimum extent size and also minimizes the creation of suboptimal extent sizes at the end of a load.
 - `DEFAULT`: Uses traditional extent allocation.

ORACLE

6 - 29

Copyright © 2009, Oracle. All rights reserved.

Minimum Extent Size

Extent size is managed automatically in locally-managed tablespaces. This option automatically increases the size of the extent depending on segment size, available free space in the tablespace, and other factors. By default, the extent size starts at 64 KB and increases in increments of 1 MB, 8 MB, or 64 MB.

Generally speaking it is recommended that large segments should be defined with larger than default initial extents to minimise the needless proliferation of small extents in the database. For Exadata, the recommendation is to have initial extents of 8MB for large segments. This setting compliments the recommendation to set the ASM allocation unit size to 4 MB. The following options can be used to hint to the database to allocate large extents:

- The `INITIAL` storage parameter will set the size of the first extent for a newly created object. It can be set specifically for individual objects or it can be set at the tablespace level for locally managed tablespaces. For objects that are large, set `INITIAL` to 8 MB.
- The `CELL_PARTITION_LARGE_EXTENTS` initialization parameter can be used to set the `INITIAL` storage parameter to 8 MB for all new segments in a partitioned object. This parameter is relevant only to locally-managed tablespaces created with the `AUTOALLOCATE` option, not the `UNIFORM` option. The parameter does not impact nonpartitioned objects. The parameter may be set to the following values: `TRUE`, `ALWAYS`, and `DEFAULT`. In general, Oracle recommends setting this parameter to `TRUE`.

Quiz

For which `objectTypes` can you specify Exadata cell metric thresholds?

1. CELL
2. GRIDDISK
3. CELLDISK
4. IORM_DATABASE
5. HOST_INTERCONNECT

ORACLE

6 - 30

Copyright © 2009, Oracle. All rights reserved.

Answers: 1, 2, 3, 4, 5

Exadata version 1 only supported thresholds for IORM metrics. Exadata V2 supports thresholds for all metrics.

Quiz

You enable SQL processing offload using the `CELL_OFFLOAD_PLAN_DISPLAY` initialization parameter.

1. TRUE
2. FALSE

ORACLE®

6 - 31

Copyright © 2009, Oracle. All rights reserved.

Answers: 2

The `CELL_OFFLOAD_PROCESSING` parameter is used to enable SQL processing offload.

Summary

In this lesson, you should have learned how to:

- Describe the various performance monitoring facilities available for Exadata
- Monitor Exadata from directly within a cell, from a database instance and through Enterprise Manager
- Interpret SQL execution plans that use offloading
- Describe the recommendations for optimizing Exadata performance

ORACLE

Demonstration Overview

The demonstrations associated with this lesson cover the following topics:

- A
- B
- C

ORACLE[®]