Resource Manager: Best Practices

Oracle Open World, Session

Sue K. Lee, Director of Development
Akshay Shah, Principal Member of Technical Staff

Oracle Server Technologies
October 1, 2014
Safe Harbor Statement

The preceding 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.
Agenda

• Resource Manager Use Cases
  – Consolidation
  – Mixed Workloads

• Managing Resources
  – CPU
  – Exadata Disk and Flash
  – Standby Databases
  – Runaway Queries
Consolidation and Resource Manager
Why Consolidate?

• Efficient server and storage utilization
  – Each generation of servers and storage is more powerful
  – Typical database workload may not fully utilize hardware
  – Database workloads are often bursty, with long periods of low utilization
  – Lots of test, development, and non-critical databases

• Fewer systems to administer
  – Reduce effort for patching and maintenance
Consolidation Challenges

• Database users apprehensive about consolidation
  – Demand performance guarantees

• Workload surges from one application can affect others
  – Excessive CPU, PGA, or I/O usage
  – Surges can originate from heavy application usage or a single runaway query

• DBAs want to control resource usage
  – Fair access to resources
  – Hosted environments – “get what you pay for”
Approach #1: Server Consolidation

Multiple databases share a server

- Application isolation
- Each application is independently maintained and upgraded

But...

- Backgrounds and SGA are not shared – inefficient resource utilization
- Each application is independently maintained and upgraded
Approach #2: Schema Consolidation

Multiple applications share a database
- Backgrounds and all database resources are shared – efficient resource utilization
- One database to administer

But...
- Object name collisions due to shared dictionary
- May require application-level changes
Approach #3: Multi-Tenant Container Database

- **Container Database (CDB)**
  - Container for up to 252 Pluggable Databases
  - Shared backgrounds and SGA
  - One database to administer

- **Pluggable Database (PDB)**
  - Feels and operates identically to a non-CDB
  - Any database can be converted into a PDB
  - Root is a special PDB used to administer the CDB

New in 12c!
Mixed Workloads and Resource Manager
Mixed Workloads and Resource Management

• Every database runs multiple workloads with different priorities
  – OLTP database
    • OLTP
    • Real-time reports
    • Maintenance (backup, stats gathering, etc.)
  – Data warehouse
    • Critical, tuned reports
    • Batch jobs
    • ETL
    • Ad-hoc reports

• These workloads compete for resources
• Use Resource Manager to allocate resources to workloads
Configuring Workloads with Consumer Groups

• First step: group database sessions that comprise a workload into a Consumer Group

• Sample consumer groups
  – Critical
  – Batch
  – Maintenance
  – Other (default)
Configuring Workloads with Consumer Groups

If you use services, create a Consumer Group for each service

- **OLTP**
  - Connected to “OLTP” Service
  - Map session to “Critical” Consumer Group

- **Report**
  - Connected to “Report” Service
  - Map session to “Batch” Consumer Group

- **Maintenance Job**
  - Connected to “Maintenance” Service
  - Map session to “Maintenance” Consumer Group
Configuring Workloads with Consumer Groups

Use any of these session attributes to map sessions to Consumer Groups:

**Session Attributes:**
- Oracle user name
- Client O/S user name
- **Client program name**
- Client machine name
- Client id
- **Service name**
- Module name
- Action name

- **Function being performed**
  - “backup” (RMAN backup, defaults to BATCH_GROUP)
  - “copy” (RMAN image copy, defaults to BATCH_GROUP)
  - “dataload” (datapump, defaults to ETL_GROUP)
  - “inmemory” (in-memory population, defaults to ORA$AUTOTASK)

New in 12.1.0.2

```
dbms_resource_manager.set_consumer_group_mapping(
    attribute => 'ORACLE_FUNCTION', value => 'INMEMORY', consumer_group => 'BATCH_GROUP');
```
Managing CPU within a Database
Without Resource Manager, Consumer Groups with many active sessions, parallel execution, and CPU-intensive work hog the CPU!
Managing CPU

Without Resource Manager, Pluggable Databases with many active sessions, parallel execution, and CPU-intensive work hog the CPU!
Managing CPU

• CPU Resource Manager supports 3 common scenarios
  – Manage workloads within a database - Database Resource Plan
  – Manage PDBs within a CDB - CDB Resource Plan
  – Manage database instances sharing a server - Instance Caging

• All of these tools can be used together
Managing CPU
Consumer Groups within a Database

A Resource Plan uses “shares” to specify how CPU is distributed between Consumer Groups.

<table>
<thead>
<tr>
<th>Consumer Group</th>
<th>Shares*</th>
<th>Utilization Limit</th>
<th>Guaranteed CPU</th>
<th>Maximum CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>5</td>
<td></td>
<td>50%</td>
<td>100%</td>
</tr>
<tr>
<td>Batch</td>
<td>2</td>
<td></td>
<td>20%</td>
<td>100%</td>
</tr>
<tr>
<td>Maintenance</td>
<td>2</td>
<td>90%</td>
<td>20%</td>
<td>90%</td>
</tr>
<tr>
<td>Other</td>
<td>1</td>
<td></td>
<td>10%</td>
<td>100%</td>
</tr>
</tbody>
</table>

*In 10g and 11g, use “mgmt_p1” instead of shares. They work the same way.
## Managing CPU

### Consumer Groups within a Database

<table>
<thead>
<tr>
<th>Consumer Group</th>
<th>Shares*</th>
<th>Utilization Limit</th>
<th>Guaranteed CPU</th>
<th>Maximum CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>5</td>
<td></td>
<td>5/(5+2+2+1) = 50%</td>
<td>100%</td>
</tr>
<tr>
<td>Batch</td>
<td>2</td>
<td>20%</td>
<td>20%</td>
<td>100%</td>
</tr>
<tr>
<td>Maintenance</td>
<td>2</td>
<td>90%</td>
<td>20%</td>
<td>90%</td>
</tr>
<tr>
<td>Other</td>
<td>1</td>
<td>10%</td>
<td>10%</td>
<td>100%</td>
</tr>
</tbody>
</table>

*In 10g and 11g, use “mgmt_p1” instead of shares. They work the same way.

“Critical” is guaranteed 50% of the CPU. If it doesn’t use it, someone else can.
Managing CPU

CRITICAL is guaranteed \( \frac{50}{50+20} = 71\% \) of the CPU.

BATCH can use the unutilized portion of CRITICAL’s allocation.

CRITICAL maintains its performance, regardless of the queries running in BATCH!

With Resource Manager, your resource plan controls how the CPU is used!
### Managing CPU

**Pluggable Databases**

A CDB Resource Plan uses “shares” to specify how CPU is distributed between PDBs.

“Sales” is guaranteed 50% of the CPU. If it doesn't use it, someone else can.

<table>
<thead>
<tr>
<th>CDB Resource Plan</th>
<th>Shares</th>
<th>Utilization Limit</th>
<th>Guaranteed CPU</th>
<th>Maximum CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pluggable Database</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales</td>
<td>2</td>
<td></td>
<td>(\frac{2}{(2+1+1)} = 50%)</td>
<td>100%</td>
</tr>
<tr>
<td>Marketing</td>
<td>1</td>
<td>75%</td>
<td>25%</td>
<td>75%</td>
</tr>
<tr>
<td>Support</td>
<td>1</td>
<td>75%</td>
<td>25%</td>
<td>75%</td>
</tr>
</tbody>
</table>

PDBs are managed just like Consumer Groups.
## Managing CPU

### Pluggable Databases

<table>
<thead>
<tr>
<th>Pluggable Database</th>
<th>Shares</th>
<th>Utilization Limit</th>
<th>Guaranteed CPU</th>
<th>Maximum CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales</td>
<td>2</td>
<td></td>
<td>$\frac{2}{2+1+1} = \frac{1}{2}$ = 50%</td>
<td>100%</td>
</tr>
<tr>
<td>Marketing</td>
<td>default (1)</td>
<td>default (75%)</td>
<td>25%</td>
<td>75%</td>
</tr>
<tr>
<td>Support</td>
<td>default (1)</td>
<td>default (75%)</td>
<td>25%</td>
<td>75%</td>
</tr>
<tr>
<td>(Default directive)</td>
<td>1</td>
<td>75%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Configure a “default directive”: the default shares and utilization limit for a PDB.
How Do CDB and PDB Resource Plans Work Together?

CDB resource plan controls how CPU is utilized by PDBs

PDB resource plan controls how CPU is utilized by Consumer Groups in a PDB
Utilization Limits for Pluggable Databases

Managing CPU

Utilization Limits are typically used in Cloud Consolidations to enforce “pay for performance”.

<table>
<thead>
<tr>
<th>CDB Resource Plan</th>
<th>Guaranteed CPU</th>
<th>Maximum CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pluggable Database</strong></td>
<td><strong>Shares</strong></td>
<td><strong>Utilization Limit</strong></td>
</tr>
<tr>
<td>Sales</td>
<td>2</td>
<td>75%</td>
</tr>
<tr>
<td>Marketing</td>
<td>1</td>
<td>75%</td>
</tr>
<tr>
<td>Support</td>
<td>1</td>
<td>75%</td>
</tr>
</tbody>
</table>
Managing CPU

Utilization Limits for Pluggable Databases

With a utilization limit of 75%, SUPPORT is throttled, even though CPU is available.

Utilization Limits provide clients consistent performance.
They also restrict their resource usage, based on what the client paid.
Configuring CPU Resource Manager

• Resource Manager can be configured with SQL or Enterprise Manager
• `dbms_resource_manager`
  – PL/SQL package
  – Create consumer group mapping rules
  – Create resource plans
  – Modify resource plans
Monitoring and Tuning CPU Resource Manager

• Configuring a resource plan is an iterative process
  – Create a resource plan
  – Monitor application performance and Resource Manager metrics
  – Adjust resource allocations and re-monitor

• Any changes to the resource plan are enforced immediately
  – Instance restart NOT required
Monitoring and Tuning CPU Resource Manager

Monitor how CPU is actually being distributed

Avg Number of Running Sessions

cpu_count

Avg Number of Sessions Waiting for CPU

ETL (10 shares)
AD-HOC (10 shares)
BATCH (20 shares)
CRITICAL (50 shares)

Monitor using v$rsrcmrgmetric_history or Enterprise Manager
Instance Caging
The Need for Instance Caging

Your database starts with enough CPU, resulting in good performance!

And then another database hogs the CPU, leaving you with not enough CPU and bad performance…
"Cage" a database by limiting the amount of CPU it can use.

Step 1: Set “cpu_count” to the max CPU threads the instance can use at any time
```
alter system set cpu_count = 8;
```

Step 2: Set “resource_manager_plan” to enable CPU Resource Manager
```
alter system set resource_manager_plan = 'default_plan';
```
Partition Approach for Critical Databases

- Partition CPUs among the database instances
  - $\text{sum(cpu\_counts) } \leq \# \text{ cpu threads}$
- Partitioning provides maximum isolation
  - No CPU contention between instances
- Best for performance-critical databases

But if Instance A is idle, its CPU allocation is unused…

This server has a total of 24 CPU threads

Instance B’s cpu_count

Instance A: 8 CPUs
Instance B: 8 CPUs
Instance C: 4 CPUs
Instance D: 4 CPUs
Over-Subscribe Approach for Non-Critical Databases

- Over-subscribe the CPUs among the database instances
  - \( \text{sum(cpu\_counts)} \leq 3 \times \# \text{cpu threads} \)
- Over-subscribing provides efficient CPU utilization
  - Some contention for CPU if databases are sufficiently loaded
- Best for non-critical databases
Over-Subscribe Approach for Non-Critical Databases

If Instance A is idle, there is no CPU contention! The databases can fully utilize the server.

This server has a total of 24 CPU threads.

Instance B: 8 CPUs
Instance C: 8 CPUs
Instance D: 8 CPUs
Over-Subscribe Approach for Non-Critical Databases

If all databases are active, there is some contention. This is your maximum excess load. Without Instance Caging, there is **no** limit on your maximum load!

This server has a total of 24 CPU threads.
Binding Critical Database to Specific CPUs

- With CPU binding, databases don’t share CPU cores or caches.
- Additional isolation only useful for platinum databases...

See MOS note 1585184.1, 1928328.1 for step-by-step guide and best practices
Binding Databases to Specific NUMA CPUs

- Bind database instance to specific NUMA nodes (Exadata X4-8)
- Or bind to specific CPU sockets (SPARC T-Series)
- Reduce expensive remote memory accesses for superior performance

See MOS note 1585184.1, 1928328.1 for step-by-step guide and best practices
Binding Databases to Specific CPUs

To bind a database instance to CPUs or NUMA nodes

1) Specify the CPUs or NUMA nodes by creating a “processor group”
   - Linux cGroups
   - Solaris resource pools
   “pg1” = CPUs 0-3

2) Set the Oracle parameter “processor_group_name” to the name of this processor group
   - All Oracle backgrounds and foregrounds run on these CPUs
   - All SGA and PGA is allocated from these NUMA nodes
   processor_group_name = pg1

See MOS note 1585184.1 for step-by-step guide and best practices
Instance Caging

Best Practices

• Keep the strategy simple
  – Initial cpu_count settings can be a guess

• Monitor actual CPU usage and then tweak
  – cpu_count can be adjusted dynamically – no database bounce needed!
  – If over-subscribing, monitor the server to make sure it’s not over-loaded
  – Avoid large changes to cpu_count

• cpu_count corresponds to CPU threads, not cores!

• Beware of over-subscribing on SPARC T-Series processors

• Don’t set cpu_count to 1
  – Makes database instance susceptible to hangs or poor performance
Managing Disk I/O – Exadata
Use Case for IORM: Scan Workloads

Without IORM

MARKETING database hogs the disk bandwidth. Without Exadata IORM, databases issuing the most load get the most bandwidth.

SALES starts at 50% disk utilization

SALES database sees a drop in throughput.

Disk Utilization by Database
Use Case for IORM: Scan Workloads

With IORM

IORM ensures predictable throughput for each database.

One database can use excess disk bandwidth without affecting the other!
Use Case for IORM: High Disk Latencies for OLTP

<table>
<thead>
<tr>
<th>Event</th>
<th>Total Waits</th>
<th>&lt;1ms</th>
<th>&lt;2ms</th>
<th>&lt;4ms</th>
<th>&lt;8ms</th>
<th>&lt;16ms</th>
<th>&lt;32ms</th>
<th>&lt;=1s</th>
<th>&gt;1s</th>
</tr>
</thead>
<tbody>
<tr>
<td>cell single block physical</td>
<td>213.7</td>
<td>90.1</td>
<td>3.3</td>
<td>.8</td>
<td>.1</td>
<td>.1</td>
<td>2.0</td>
<td>3.6</td>
<td></td>
</tr>
<tr>
<td>log file parallel write</td>
<td>951.5</td>
<td>93.9</td>
<td>2.6</td>
<td>1.3</td>
<td>1.1</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

• On Exadata, most OLTP I/Os are serviced from flash, except
  – Flash cache misses
  – Log writes when flash is busy
• During a smart scan, OLTP disk I/Os can be very slow
• Enable IORM to avoid outliers. Use the “objective” knob to enforce low disk latencies.
I/Os are processed in the order received. The O/S usually lets small OLTP I/Os cut towards the front of the queue. But waiting behind the I/Os in front can still take 100s of ms.
Use Case for IORM: High Disk Latencies for OLTP

Exadata Storage

IORM tags every IO:
- Database, PDB, Consumer Group id
- Background or foreground?
- Buffer cache or smart scan?

Exadata IORM can put a new IO anywhere in the queue!

Exadata IORM positions the I/O in the queue, based on the I/O’s properties and the Resource Plan

OLTP
Parallel Execution
Smart Scan
Exadata IORM IO Queue
Exadata Disk
Use Case for IORM: High Disk Latencies for OLTP

Exadata Storage

Exadata IORM can put redo log and control file writes at the very front of the queue!

Background and ASM IOs are prioritized, based on why they were issued.
DBWR IOs for checkpoint: high priority.
DBWR IOs when the buffer cache has plenty of free buffers: low priority.
Exadata IORM

• Exadata I/O Resource Manager
  – Disk I/O scheduler
  – Industry pioneer for disk I/O management
  – For Exadata only!

• Enables multiple workloads to share storage in a predictable way
  – Prioritizes critical workloads over non-critical workloads
  – Allows fair sharing for database consolidation

• Ensures low disk latency when OLTP is actively using disk

• Ensures high disk throughput for throughput-intensive workloads
Exadata IORM

To enable IORM

1) Set IORM objective from “basic” to “auto”, using CellCLI or EM Cloud Control 12c

   CellCLI> alter iormplan objective = auto

   – If workload is latency sensitive, ‘auto’ optimizes for low latency
   – If workload is throughput oriented, ‘auto’ optimizes for maximum throughput

2) Enable a resource plan
# Exadata IORM

## Types of Resource Plans

<table>
<thead>
<tr>
<th>Consolidation Scenario</th>
<th>CPU Resource Manager</th>
<th>I/O Resource Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consumer Group workloads sharing a database</td>
<td>Database Resource Plan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(configured on database)</td>
<td></td>
</tr>
<tr>
<td><strong>New in 12c</strong> PDBs sharing a CDB</td>
<td>CDB Resource Plan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(configured on database)</td>
<td></td>
</tr>
<tr>
<td>Database Instances sharing a server or storage</td>
<td>Instance Caging</td>
<td>Inter-Database IORM Plan</td>
</tr>
<tr>
<td></td>
<td>(configured on database)</td>
<td>(configured on storage cell)</td>
</tr>
</tbody>
</table>
# Exadata IORM

## Types of Resource Plans

<table>
<thead>
<tr>
<th>Consolidation Scenario</th>
<th>CPU Resource Manager</th>
<th>I/O Resource Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consumer Group workloads</td>
<td>Database Resource Plan (configured on database)</td>
<td></td>
</tr>
<tr>
<td>sharing a database</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>New in 12c</strong></td>
<td><strong>Database Resource Plan (configured on database)</strong></td>
<td><strong>CDB Resource Plan (configured on database)</strong></td>
</tr>
<tr>
<td>PDBs sharing a CDB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database Instances sharing a</td>
<td>Instance Caging (configured on database)</td>
<td>Inter-Database IORM Plan (configured on storage cell)</td>
</tr>
<tr>
<td>server or storage</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Inter-Database Plans are only for IORM. They are configured on the storage cell.
IORM for Workloads within a Database

The Database Resource Plan is used to manage both CPU and Disk I/O

<table>
<thead>
<tr>
<th>Consumer Group</th>
<th>Resource Allocation (shares or mgmt_p1)</th>
<th>Utilization Limit</th>
<th>Guaranteed Disk Utilization</th>
<th>Maximum Disk Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>4</td>
<td></td>
<td>4/(4+1+1+1+1)=50%</td>
<td>100%</td>
</tr>
<tr>
<td>Batch</td>
<td>1</td>
<td></td>
<td>12%</td>
<td>100%</td>
</tr>
<tr>
<td>AdHoc</td>
<td>1</td>
<td>50%</td>
<td>12%</td>
<td>50%</td>
</tr>
<tr>
<td>ETL</td>
<td>1</td>
<td></td>
<td>12%</td>
<td>100%</td>
</tr>
<tr>
<td>Other</td>
<td>1</td>
<td></td>
<td>12%</td>
<td>100%</td>
</tr>
</tbody>
</table>
IORM for Workloads within a Database
Configure with Database Resource Plan

Non-critical batch operations use remaining disk resources

CRITICAL is guaranteed 4/(4+1)=80% disk utilization
IORM for Workloads within a Database
Configure with Database Resource Plan

To change this distribution, change the resource plan.

CRITICAL is guaranteed $4/(4+1)=80\%$ disk utilization
## IORM for Multiple Databases

Without an inter-database plan, all databases are scheduled equally.

<table>
<thead>
<tr>
<th>Database</th>
<th>Shares</th>
<th>Utilization Limit</th>
<th>Guaranteed Disk Utilization</th>
<th>Maximum Disk Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales</td>
<td>2</td>
<td></td>
<td>$\frac{2}{2+1+1}=50%$</td>
<td>100%</td>
</tr>
<tr>
<td>Support</td>
<td>1</td>
<td></td>
<td>25%</td>
<td>100%</td>
</tr>
<tr>
<td>Marketing</td>
<td>1</td>
<td>50%</td>
<td>25%</td>
<td>50%</td>
</tr>
<tr>
<td>Default</td>
<td>1</td>
<td>25%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Databases not listed in the plan use the ‘default’ values.
IORM Utilization Limits

*IORM can limit the disk utilization of a consumer group, PDB, or database. Useful for enforcing “pay for performance” in cloud environments.*

<table>
<thead>
<tr>
<th>Inter-Database IORM Plan</th>
<th>CDB Resource Plan</th>
<th>Database Resource Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td><strong>Shares</strong></td>
<td><strong>Utilization Limit</strong></td>
</tr>
<tr>
<td>Sales</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Marketing</td>
<td>1</td>
<td>50%</td>
</tr>
<tr>
<td>Support</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Monitoring and Tuning IORM

Average throttle time by database

Disk I/O utilization by database

Average small I/O latency

IORM objective
Managing Exadata Flash
Exadata Smart Flash Cache

- Exadata Flash provides impressive capacity and bandwidth*
  - 45 TB space
  - 100 GB/s of scan throughput
  - ~2.5 M IOPS

- Use Exadata Flash to
  - Cache frequently-used OLTP data
  - Accelerate scans, using excess flash resources
  - Accelerate log writes

* For a X4-2/8 full rack
Exadata Smart Flash Cache

Why Exadata?

• Other market offerings
  – All-flash storage: expensive
  – Disks with flash to accelerate writes and cache recently-used data: doesn’t support mixed workloads

• Only Exadata guarantees that flash stores frequently-accessed OLTP data
  – Scans do not evict OLTP data from the cache
  – Only Exadata can identify OLTP from scan IOs!
  – All-flash storage guarantees this too, but at many times the price!
Exadata Smart Flash Cache

Flash Bandwidth

How can you capitalize on 100 GB/s of flash bandwidth?

• Use up to 20% for OLTP workloads
  – Maximum sustainable rate by OLTP

• Use the rest for scans!
  – Pre 11.2.2.3, mark tables with KEEP
  – 11.2.2.3+, scans automatically use flash if there’s space

• Scans use both flash and disks for maximum throughput
  – Using flash increases scan throughput by up to 5x!
Exadata Smart Flash Cache

Space Contention

For most workloads, Exadata already uses flash space optimally. To customize, use the Inter-Database IORM Plan.

<table>
<thead>
<tr>
<th>Inter-Database IORM Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Flash Log?</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>DB #1</td>
</tr>
<tr>
<td>DB #2</td>
</tr>
<tr>
<td>DB #3</td>
</tr>
</tbody>
</table>

Flash Logs use just 0.02% of total space. Don’t disable!

Disabling flash cache may cause a heavier disk I/O load.
Managing Standby Databases
Resource Manager for Standbys

1) Primary DB Workload can slowdown
2) Redo and Apply can slowdown
3) Read-Only Queries
4) DB X’s Workload can slowdown

Issue #1: If synchronous redo transport is enabled, the Primary DB will suffer from slow redo writes on the Standby

Issue #2: Heavy read-only queries and other database workloads can slow the Standby’s redo and apply
Resource Manager for Standbys

Prioritize Redo and Apply over Read-Only Queries

- Enable CPU and I/O resource management
  - Set a database resource plan on Standby
  - Use the primary’s resource plan or DEFAULT_PLAN (best practice - use the same resource plan on primary and standby in case a switch-over occurs!)
  - Enable I/O Resource Manager (alter iormplan set objective = auto)

- On physical standbys, all resource manager objects are replicated on the standby
  - New resource plans cannot be configured on the standby
  - The primary and standby can use different resource plans
  - See MOS note 1930540.1 for known issues for consumer group mappings

- On logical standbys, all resource manager objects must be recreated
Resource Manager for Standbys

Guarantee Resources for the Standby

• Enable Instance Caging for all database instances on the Standby’s server
  – CPU_COUNT must be sufficiently high to handle the load in the event of a switch-over
• Enable inter-database IORM for all databases on the Standby’s storage cells
  – Create the allocation using standby’s DB_UNIQUE_NAME
Managing Runaway Queries
Managing Runaway Queries

• Runaway queries can be caused by
  – Badly written SQL
  – Bad execution plans

• Severely impact performance of well-behaved queries

• Very hard to completely eradicate!
Managing Runaway Queries

Configure by Consumer Group

**Define** runaway query thresholds:

- ✔ Estimated execution time
- ✔ Elapsed time
- ✔ Amount of CPU time used
- ✔ Number of I/Os issued
- ✔ Bytes of I/O issued
- ✔ Number of logical I/Os issued

**Manage** runaway queries:

- ✔ Switch to a lower-priority consumer group
- ✔ Abort call
- ✔ Kill session
- ✔ Log to SQL Monitor

*New in 12c*
Managing Runaway Queries

Workload Management in a Cloud Database

SQL starts in HIGH group

HIGH allocation = 70

After 10 seconds, switched to MEDIUM group

MEDIUM allocation = 20

After 2 minutes, switched to LOW group

LOW allocation = 10

After 3 minutes, SQL aborted with ORA-40

For More Information
For More Information

• Master MOS document 1339769.1
  – Recommended bug fixes
  – Step-by-step configuration for common scenarios
  – Monitoring and tuning scripts

• Best Practices for Multi-Tenant Consolidation on Exadata

• Best Practices for Database Consolidation on Exadata

• Instance Caging white paper

• Resource Manager white paper
Hardware and Software
Engineered to Work Together