ORACLE
SOLARIS

An Oracle White Paper
June 2011

Effective Resource Management Using Oracle Database Resource Manager

ORACLE

Introduction

Application workloads on a server need to be balanced for the system efficiency. Granular resource management is a necessity to achieve the anticipated performance and service levels in any environment including virtual and consolidated environments. Without good resource management, faulty runaway workloads can bring progress to a halt causing unwanted delays to priority jobs.

In addition, efficient resource management helps organizations economize by consolidating servers. Server consolidation is one of the effective ways to maximize return on investment (ROI) by cutting unnecessary costs on underutilized servers in a datacenter. Resource management allows controlled allocation of resources to different workloads. An OS process and an active database session are examples of a basic unit of workload.

This paper is part 3 of a four-part series. It provides details about Oracle Database Resource Manager and how it can be used to manage system resources effectively. Although most of the topics in this paper are discussed from a consolidated point of view, resource management features and the content in this paper are equally applicable, with few exceptions, in consolidated and isolated environments running Oracle Solaris and Oracle Database.

For more information about Oracle Solaris Resource Manager and Oracle Database Resource Manager, plus a case study that provides examples that demonstrate their features, see the other parts of this series:

- Part 1: “Introduction to Resource Management in Oracle Solaris and Oracle Database”
- Part 2: “Effective Resource Management Using Oracle Solaris Resource Manager”
- Part 4: “Resource Management Case Study for Mixed Workloads and Server Sharing”

The target audience of this paper is Oracle Solaris System Administrators and Oracle Database Administrators. For the sake of simplicity, the acronym "CPU" was used in many places in reference to virtual processors and hardware threads.

Resource Management in Oracle Database

Oracle Database Resource Manager is a database resource management mechanism that controls how system resources are allocated in the database, which gives administrators more control over resource management decisions. It provides the ability to optimize resource allocation among concurrent database sessions. Using the Oracle Database Resource Manager, database operations can be prioritized by aligning the system resources to the business objectives and priorities of the enterprise.

As workloads are becoming complex and heterogeneous consisting of many different classes of users and activities, Oracle Database Resource Manager is extremely useful in managing mixed-mode workloads to consolidate business data management into fewer and larger database systems and in cloud computing services, such as application hosting.

In a busy environment with concurrent database user sessions that run jobs with differing priorities, all sessions should not be treated equally. With the help of Oracle Database Resource Manager, database sessions can be classified into different groups and allocated resources based on session attributes to optimize hardware utilization.

The following are some of the resource management tasks that can be performed with the Database Resource Manager:

- Guarantee certain sessions a minimum amount of resources regardless of the load on the system.
- Distribute available processing resources to different database users and applications. For example, in a data warehouse, a higher percentage can be allocated to online analytical processing than to batch processing.
- Partition and dedicate the CPU resources to any database instance in consolidated environments for predictable performance.
- Limit the degree of parallelism for any database operation.
- Create an active session pool consisting a specified maximum number of database sessions that are allowed to be concurrently active within a group of users.
- Manage runaway sessions by placing an absolute limit on the percentage of CPU that a group can consume and by detecting when a session consumes more than a specified amount of system resources, and then automatically terminating the session or switching it to a low-priority consumer group that has limited access to system resources.
- Prevent the execution of operations that the optimizer estimates will run for a longer time than a specified limit.
- Limit the amount of time that a session can be idle. This can further be refined to limit only those sessions that are blocking other sessions.

•Specify the maximum amount of undo space that can be generated by a group of users.

• Configure an instance to use a specific plan to allocate resources. Administrators can dynamically change the plan, for example, from a day-time setup to a night-time setup, without having to restart the instance.

Oracle Database Resource Manager is an evolving feature that was introduced in Oracle 8*i*. Starting with Oracle 11*g*, Oracle Database Resource Manager can be used to manage maintenance tasks with the default `default_maintenance_plan` in all editions of Oracle Database including Oracle Database, Standard Edition. However, customized maintenance plans require Oracle Database, Enterprise Edition.

Specifics of Oracle Database Resource Manager

Oracle Database Resource Manager is a module not a process. When enabled, each running Oracle process or thread calls in to the Oracle Resource Manager scheduling code periodically.

Oracle Database Resource Manager allows one Oracle process per CPU to run at a time. All other processes wait in an internal run queue. The database processes that are waiting to be scheduled can be identified by the Oracle wait event `resmgr:cpu quantum`.

Oracle Database Resource Manager allows an Oracle process to run for a small quantum of time, typically, 100 milliseconds or until it blocks for some resource, whichever occurs first. The running process then yields to a runnable process in the run queue. Oracle Database Resource Manager uses a round-robin algorithm to schedule all runnable processes so they all make progress.

Oracle Database Resource Manager monitors the number of runnable and running Oracle processes. As long as this number is less than the number of CPUs available on the system, Oracle Database Resource Manager lets the processes run with no restrictions. In addition, the Oracle Database Resource Manager scheduler attempts to maximize CPU utilization by redistributing allocated resources from idle consumer groups to other active consumer groups based on the resource plans.

Oracle Database Resource Manager does not manage the Oracle Database background processes unless they are non-critical and CPU-intensive. The background processes are usually either high-priority or not CPU-intensive.

The goal of Oracle Database Resource Manager is to control database instances' CPU usage but not to manage the CPUs on the physical server. Hence, it doesn't bind Oracle Database processes to CPUs.

Enabling Oracle Database Resource Manager

Oracle Database Resource Manager can be enabled by setting the `RESOURCE_MANAGER_PLAN` initialization parameter. This parameter specifies the resource plan to be used for the current instance, for example:

```
RESOURCE_MANAGER_PLAN = some_plan
```

An error message is returned if the specified plan does not exist in the data dictionary.

Alternatively, Oracle Database Resource Manager can be activated or the resource plan can be changed dynamically using the `ALTER SYSTEM` or the `DBMS_RESOURCE_MANAGER.SWITCH_PLAN` package, for example:

```
SQL> ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = 'random_plan';
SQL> exec DBMS_RESOURCE_MANAGER.SWITCH_PLAN ('random_plan')
```

The SQL statements above set (or reset) the resource plan to `random_plan` and activate Oracle Database Resource Manager if it is not active.

Disabling Oracle Database Resource Manager

If no plan is specified with the database initialization parameter `RESOURCE_MANAGER_PLAN`, Oracle Database Resource Manager is not activated.

Oracle Database Resource Manager can be deactivated dynamically by setting the `RESOURCE_MANAGER_PLAN` to a NULL string, as shown below:

```
SQL> ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = '';
```

Although Oracle Database Resource Manager is not enabled or deactivated, it is automatically enabled for the duration of the scheduler maintenance windows.

Consider setting the parameter `_resource_manager_always_off=TRUE` to turn off Oracle Database Resource Manager completely. This is an undocumented parameter, so exercise caution when using it.

Automatic Plan Switches by Oracle Scheduler

Oracle Database Resource Manager allows only one resource plan to be active at a time. If a resource plan is activated, Oracle Database Resource Manager automatically switches the plan when an Oracle Scheduler maintenance window opens. When the Oracle Scheduler window closes, the resource plan associated with it is disabled and the resource plan that was running before the Oracle Scheduler window opened is re-enabled.

In an Oracle Real Application Clusters environment, an Oracle Scheduler window applies to all instances, so the window's resource plan is enabled on every instance.

In some cases, especially in time-critical situations, the automatic switching of resource plans is undesirable. To prevent the auto-switching, prepend `FORCE:` to the name of the resource plan, as shown in the following examples.

```
SQL> ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = 'FORCE:random_plan';
RESOURCE_MANAGER_PLAN = FORCE:random_plan // init.ora parameter
```

The prefix `FORCE:` indicates that the current resource plan can be changed only when the database administrator changes the value of the `RESOURCE_MANAGER_PLAN` initialization parameter. This restriction can be relaxed by rerunning the same command without the prefix `FORCE:`.

Minimal Resource Manager Configuration for a Healthy Database

If the objective is to keep the database healthy with minimal configuration changes, it can be achieved by setting the `resource_manager_plan` parameter to `DEFAULT_PLAN`.

The out-of-the-box `DEFAULT_PLAN` provides the following advantages:

- It protects critical background processes, such as PMON and LMS, from CPU starvation due to excessive load from foreground processes.
- SYSTEM and SYS database users are scheduled at the highest priority, allowing them to always be able to log in and debug database problems. Their response time will not be affected by runaway CPU activity on the system.
- Automated maintenance tasks, such as gathering optimizer statistics, are scheduled at the lowest priority. Therefore, these tasks do not compete with other sessions for CPU. However, when the database workload is minimal, these maintenance tasks are scheduled to consume any remaining unused CPU resources.

It is recommended that you devise your own resource plans and schedule according to your requirements to take full advantage of the flexibility that Oracle Database Resource Manager provides.

Building Blocks of Oracle Database Resource Manager

The basic building blocks of Oracle Database Resource Manager are resource consumer groups, resource plans, and resource plan directives.

A *resource consumer group* is a set of database sessions that are grouped together based on resource requirements. Oracle Database Resource Manager allocates resources to resource consumer groups, not to individual sessions.

A *resource plan* is a collection of Oracle Database Resource Manager directives or rules that specify how resources are allocated to resource consumer groups. Administrators specify resource allocations by activating a specific resource plan. Only one resource plan can be active per database instance.

A *resource plan directive* associates a resource consumer group with a particular resource plan and specifies how resources are to be allocated to that resource consumer group.

For example, in a data warehouse one of the challenges is to deliver the required I/O bandwidth and CPU for executing large-scale queries. Using Oracle Database Resource Manager, administrators can group end users into different resource consumer groups based on their roles and responsibilities in the organization, and set different policies to govern the amount of CPU and I/O resources that can be used by each consumer group to ensure that the data warehouse can meet the performance SLA for all users. Assuming executive management receives highest priority and employees at the bottom of an organization tree get the least priority in a data warehouse, a simple resource plan, such as the following, can be created.

TABLE 1. A SAMPLE RESOURCE PLAN

RESOURCE CONSUMER GROUP	CPU RESOURCE ALLOCATION	I/O RESOURCE ALLOCATION
Executive Managers	55%	550K IOPS
Sales Users	25%	250K IOPS
Marketing Users	15%	150K IOPS
Standard Users	5%	50K IOPS

Database Resource Manager in an Oracle Real Application Clusters (RAC) Environment

In an Oracle RAC environment, Oracle Database Resource Manager manages each database instance independently. Each member instance can be assigned different resource plans to tailor them to support different workloads. For example, in a two-node cluster, one instance can use a resource plan that allocates most of its resources to online users, while the other instance can use a different plan allocating most of its resources to batch operations. This allows different applications or users to use the same database without impacting each other.

Resource Allocation Methods

Resource allocation methods determine what method or policy Oracle Database Resource Manager uses when allocating a particular resource. Oracle Database Resource Manager provides several policies for allocating resources among resource consumer groups. These capabilities allow administrators to proactively limit the resources consumed by different database operations preventing them from negatively impacting other users.

CPU Allocations

The CPU allocation method allows administrators to specify how CPU resources are to be allocated among consumer groups or subplans when the CPU resources are saturated. By allocating a minimum amount of CPU to sessions in each consumer group, Oracle Database Resource Manager can lower the CPU consumption of low-priority sessions, thus providing more resources to sessions of higher priority. By default, the resource allocation method uses percentages and can be used with multilevel plans. An alternative method available only for single-level plans uses ratios.

This method supports CPU allocation at eight different levels. The multiple levels of CPU allocation provide a means of further prioritizing CPU usage within a resource plan. For example, level 2 receives resources only after level 1 is unable to use all of its allocated resources. Multiple levels not only provide a way of prioritizing but they also provide a way of explicitly specifying how primary and leftover resources are to be used.

Per-Session I/O Limit

Starting with Oracle Database 11*g* Release 1, I/O can be limited per session. I/O usage per session can be controlled with the help of the `SWITCH_IO_REQS`, `SWITCH_IO_MEGABYTES`, and `SWITCH_TIME` resource directives. The `SWITCH_IO_REQS` and `SWITCH_IO_MEGABYTES` directives specify the number of I/O requests that a session can perform and the amount of I/O bandwidth in megabytes that a session can consume, respectively. Once the session reaches the specified thresholds, it will either be switched to another consumer group or killed. Per-session I/O limits are enforced irrespective of the availability of the resources on the system.

Active Session Pool with Queuing

Oracle Database Resource Manager allows controlling the maximum number of concurrent database calls at any time within a consumer group. This maximum designates the active session pool. When a call cannot be initiated because the pool is full, the session is placed into a queue. When an active call is completed, the first session in the queue can then be scheduled for execution. It is also possible to specify a timeout period after which a session waiting for execution in the queue will time out, causing it to terminate with an error. The active session pool limits the total number of sessions actively competing for resources, thereby enabling active sessions to make faster progress.

Limiting Degree of Parallelism

Administrators can control the maximum number of parallel execution servers associated with any single operation within a consumer group by specifying the maximum degree of parallelism. This method is extremely useful in data warehousing environments because it ensures that no single parallel operation consumes all the processing resources on the system.

Starting with Oracle Database 11*g* Release 2, automatic degree of parallelism, statement queuing, and in-memory parallel execution can be enabled with the `PARALLEL_DEGREE_POLICY` parameter to fully utilize resources.

Automatic Consumer Group Switching

This method lets administrators control resources by specifying criteria that can cause the automatic switching of sessions to another consumer group that usually has lower priority. The following criteria can be used to determine the switching of the session:

- The length of time that the session executes before it is switched. Oracle Database Resource Manager can be directed to switch an active session permanently or temporarily for the duration of the top call. The session is restored to its original consumer group once the top call finishes.
- The estimated time the operation might take to complete. If the database estimate is longer than the specified switch time, the database switches the session even before execution starts.

Canceling SQL Queries and Terminating Sessions

Consumer group switching directives can be used to cancel long-running SQL queries or to terminate long-running sessions.

Execution Time Limit

Oracle Database Resource Manager allows specifying the maximum execution time allowed for a database operation. If the database estimates that the operation might run longer than the specified maximum execution time, the operation is not started. Database administrators can use this method to prevent unacceptably long, resource-intensive operations from starting.

Undo Pool

Administrators can specify an undo pool for each consumer group to control the total number of rollback segments that can be generated by the sessions of that consumer group. When the total undo generated exceeds its limit, the corresponding SQL statement is terminated. No other members of the consumer group can perform further data manipulation until undo space is freed from the pool. This method prevents runaway operations from consuming excessive undo space.

Idle Time Limit

Using Oracle Database Resource Manager, administrators can specify an amount of time that a session can be idle, after which it is terminated. It is possible to restrict such termination only to sessions that are idle and blocking other sessions from acquiring resources.

Managing Multiple Workloads Within a Database Instance

By default, Oracle Database Resource Manager treats all sessions alike. However, Oracle Database Resource Manager can be configured to manage workloads differently by creating consumer groups and resource plans.

A consumer group is a collection of database sessions that are managed as a unit. Each application in a database can be defined as a consumer group, or consumer groups can be created for each type of workload, such as OLTP, batch, reports, and maintenance.

Database sessions can be automatically mapped to a consumer group by defining consumer group mapping rules. For example, a session from the OLTP service can be mapped to the interactive consumer group. The attributes that can be used in mapping rules include the session's service name, module name, action, database user name, client user name, and program name.

A resource plan specifies how system resources, such as CPU and I/O, should be shared among the consumer groups. A resource plan contains a resource plan directive for each consumer group that specifies its system resource allocation.

Managing CPU Resources

Oracle Database Resource Manager manages CPU resources by controlling the database load at a very granular level. By default, this level is set to the number of processors, as reported by the operating system. In case of hyperthreaded CPUs (such as Intel Xeon, AMD Opteron, or SPARC64-VII) and multithreaded processors (such as SPARC T3), Oracle Solaris treats each hardware thread as a virtual CPU and reports a larger number of CPUs than the actual number of physical CPUs installed on the server. In such cases, Oracle Database Resource Manager manages the database load based on the total number of threads or virtual CPUs on the server. For example, on Oracle's two-socket SPARC T3-2 server, Oracle Database Resource Manager ensures that no more than 256 Oracle Database processes are running at a time. This behavior allows the database instance to fully utilize the server CPU resources while balancing the workload. By controlling the database load, critical background processes are able to run in a timely manner and the load on the server is regulated.

Managing Multiple Database Instances on a Single Server

To utilize available hardware resources efficiently, you might decide to consolidate multiple Oracle Database instances on a single multi-CPU, multicore server. However, when running multiple instances on a single server, the instances compete for CPU resources on the system. One CPU-intensive database instance could significantly degrade the performance of the other instances. For example, on a 16-CPU server with two database instances running, one instance could use the majority of the CPUs during a period of heavy load. This could degrade the performance of the other instance. To handle situations that arise from a CPU resource shortage, Oracle Database provides a feature called *instance caging* for managing CPU allocations on a multi-CPU server running multiple database instances. Instance caging was introduced in Oracle Database 11*g* Release 2 (11.2.0.1).

Instance Caging

Instance caging provides a simple way to limit CPU consumption for each database instance. Instance caging uses an initialization parameter to limit the number of CPUs that an instance can use simultaneously. On a 16-CPU server with two database instances running, if the instance caging feature is used to limit the number of CPUs to eight for each of the two instances, there is less likelihood that one instance can interfere with the other. When constrained to eight CPUs, an instance might become CPU-bound. This is when Oracle Database Resource Manager can be used to allocate CPU resources according to the resource plan that was created for the instance. Thus, instance caging and Oracle Database Resource Manager together provide a simple, effective way to manage multiple instances on a single server.

In short, the instance caging feature relies on Oracle Database Resource Manager and the `CPU_COUNT` parameter to limit the amount of CPU an Oracle Database instance can consume.

There are two typical approaches:

- Overprovisioning: In this approach, the sum of the CPU limits for each instance exceeds the actual number of CPUs on the system. For example, on a 16-CPU server with two database instances, an administrator might limit each instance to 10 CPUs. When a server is overprovisioned in this way, the instances can impact each other. Instance caging limits the performance impact. On the other hand, if one of the instances experiences high load, the CPUs are available to handle it. This is a reasonable approach for low-load non-critical production systems or non-critical systems, such as development and test systems, because one or more of the instances might frequently be at a low load or idle.
- Partitioning: In this approach, the sum of all allocations is equal to the number of CPUs on the server. For example, on a 16-CPU server with two database instances, an administrator might limit each instance to eight CPUs or might allow one instance to use 10 CPUs leaving six CPUs for the other instance. By dedicating CPU resources to each database instance, the load on one instance has no impact on the other and each instance performs in a predictable manner. This approach is ideal for critical production systems. However, when the database is idle, the allocated CPUs remain idle and cannot be used in other instances, even when other instances are short on CPU resources.

With instance caging, it is not possible to specify which CPUs Oracle Database should use for a particular database instance. Desired isolation can be achieved by using Oracle Solaris Resource Manager features such as processor sets and resource pools. Refer to part 2 of this series, "Effective Resource Management Using Oracle Solaris Resource Manager," for instructions.

Enabling Instance Caging

To enable instance caging, perform the following for each instance on the server:

1. Enable Oracle Database Resource Manager by assigning any resource plan (including the `default_plan`), and ensure that the resource plan has CPU directives, such as `MGMT_P1` through `MGMT_P8` or `MAX_UTILIZATION_LIMIT`.

2.Set the `CPU_COUNT` dynamic initialization parameter to the desired number of CPUs, for example:

```
% uname -X | grep CPU
NumCPU = 32

% sqlplus / as sysdba
SQL> ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = 'RANDOM_PLAN';
SQL> ALTER SYSTEM SET CPU_COUNT = 8 SCOPE=BOTH;
SQL>
```

To check whether instance caging is enabled, execute the following queries:

```
SQL> SHOW PARAMETER CPU_COUNT
SQL> SHOW PARAMETER RESOURCE_MANAGER_PLAN
SQL> SELECT NAME FROM V$RSRC_PLAN WHERE IS_TOP_PLAN='TRUE'
  2> AND CPU_MANAGED='ON';
```

If the last query returns a row, that is an indication that instance caging is in use.

Note that the parameter `CPU_COUNT` specifies the number of CPUs available to Oracle Database. By default, Oracle Database automatically initializes the value of `CPU_COUNT` to the number of CPUs reported by the operating system. Oracle Database uses the `CPU_COUNT` parameter to calculate the default values for many other parameters including minimum SGA size, parallel maximum servers, and degree of parallelism. On servers with a large number of virtual CPUs, such as Oracle's Sun SPARC Enterprise T-series servers, certain parameters are set to unexpectedly high values during the database startup. As a result, the database might exhibit aberrant behavior occasionally while performing certain database operations.

One way to get around this behavior is to manually set the CPU_COUNT to the number of cores available to Oracle Database or to any other value based on experimental data. When instance caging is not enabled, initializing CPU_COUNT to a non-default value does not prevent an Oracle Database instance from using all the available CPUs on the server to perform various database operations. It merely reduces the default values for many database parameters to more realistic values during the database instance startup.

Database Resource Allocation Management

Listed below are the typical steps involved in allocating system resources using Oracle Database Resource Manager to manage multiple workloads within a database.

- Grant privileges to administer Oracle Database Resource Manager
- Decide between a simple or complex resource plan
- Create a resource plan
- Assign database sessions to resource consumer groups
- Grant switch privilege for resource consumer groups
- Enable the resource plan

Granting Privileges to Administer Oracle Database Resource Manager

An administrator with the ADMIN option may grant the ADMINISTER_RESOURCE_MANAGER system privilege to other users or roles so they can administer Oracle Database Resource Manager.

```
SQL> exec DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SYSTEM_PRIVILEGE (
        GRANTEE_NAME   => 'SCOTT',
        PRIVILEGE_NAME => 'ADMINISTER_RESOURCE_MANAGER',
        ADMIN_OPTION   => FALSE);
```

In this example, the required privilege is granted to database user SCOTT. Due to the lack of the ADMIN option, SCOTT would not be able to grant administrative privilege to other users.

This privilege can be revoked using REVOKE_SYSTEM_PRVILEGE in a similar manner.

Deciding Between a Simple or Complex Resource Plan

Review the predefined resource plans that are available in the database by default. See "Predefined Resource Plans and Consumer Groups" in the "Oracle Database Resource Manager Reference" section of Chapter 27, "Managing Resource Allocation with Oracle Database Resource Manager" in the Oracle Database Administrator's Guide 11g Release 2 (11.2). If none of the predefined plans meets the requirement, decide between a simple or a complex resource plan.

Creating a Simple Resource Plan

Resource plans that are suitable for many situations can be quickly created using the `CREATE_SIMPLE_PLAN` procedure. This procedure enables you to create consumer groups and allocate resources to them in a single procedure call. Up to eight consumer groups can be specified with this procedure. CPU is the only resource allocation method supported by any simple resource plan. By default each consumer group uses the `ROUND_ROBIN` scheduling policy.

```
SQL> exec DBMS_RESOURCE_MANAGER.CREATE_SIMPLE_PLAN
         (SIMPLE_PLAN => 'SALES_SIMPLE_PLAN',
        CONSUMER_GROUP1 => 'SLSGRP1', GROUP1_PERCENT => 25,
        CONSUMER_GROUP2 => 'SLSGRP2', GROUP2_PERCENT => 30,
        CONSUMER_GROUP3 => 'SLSGRP3', GROUP3_PERCENT => 40,
        CONSUMER_GROUP4 => 'SLSGRP4', GROUP4_PERCENT =>  5);
```

Creating a Complex Resource Plan

If the situation demands a not-so-simple resource plan, creating such a plan requires performing multiple tasks, such as the following.

1. Create a pending area.
2. Create resource consumer groups.
3. Create the resource plan.
4. Create resource plan directives.
5. Assign sessions to resource consumer groups.
6. Validate the pending area.
7. Submit the pending area.

Step 1. Create a Pending Area

A temporary staging area called the *pending area* must be created to hold the resource management configuration when creating a new resource plan, updating an existing plan, or removing an existing plan.

```
SQL> exec DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
```

Any attempt to create, update, or delete a plan without creating the pending area results in error. The changes in the pending area are not visible until the changes are validated and submitted. Upon submission, all pending changes are applied to the data dictionary and the pending area is cleared and deactivated. Only one pending area per database can be activated at a time.

The changes in the pending area can be abandoned at any time by clearing the pending area.

```
SQL> exec DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA();
```

Step 2. Create Resource Consumer Groups

If the predefined resource consumer groups do not meet the requirements, a new resource consumer group can be created using the `CREATE_CONSUMER_GROUP` procedure. See "Predefined Resource Plans and Consumer Groups" in the "Oracle Database Resource Manager Reference" section of Chapter 27, "Managing Resource Allocation with Oracle Database Resource Manager" in the Oracle Database Administrator's Guide 11g Release 2 (11.2).

The following PL/SQL code creates a consumer group called `PLATINUM_CG` with the `RUN-TO-COMPLETION` method of allocating resources to sessions in the group.

```
SQL> exec DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
     CONSUMER_GROUP => 'PLATINUM_CG',
     COMMENT        => 'Consumer Group for Platinum Service Level Customers',
     MGMT_MTH       => 'RUN-TO-COMPLETION');
```

There can be no more than 31 resource consumer groups in any active plan.

It is possible to create orphan consumer groups that have no plan directives referring to them.

Step 3. Create the Resource Plan

If the predefined resource plans do not meet the requirements, a new resource plan can be created with the `CREATE_PLAN` procedure.

```
SQL> exec DBMS_RESOURCE_MANAGER.CREATE_PLAN (
        PLAN     => 'SERVICE_LEVEL_PLAN',
        COMMENT  => 'Plan that supports two service levels',
        MGMT_MTH => 'RATIO');
```

By default, the resource allocation method uses percentages and can be used with multilevel plans. An alternative method available only for single-level plans uses ratios. The `RATIO CPU` allocation method specifies numbers corresponding to the ratio of CPU that is allocated to each consumer group.

A resource plan allows multiple levels of CPU resource allocation to prioritize CPU usage within a plan. As of Oracle Database 11*g* Release 2, up to eight levels of CPU resource allocation are supported in a resource plan. In a multilevel plan, resources are rolled over to the next level only if the current level is unable to use all the allocated resources.

For example, in a two-level plan, level 2 receives resources only after level 1 is unable to use all of its allocated resources. Multiple levels not only provide a way of prioritizing but they also provide a way of explicitly specifying how all primary and leftover resources are to be used. When there is only one level, unused allocation by any consumer group or subplan can be used by other consumer groups or subplans in the same level.

Resource plans and resource consumer groups cannot have the same name. Make sure you provide unique names to the resource plans and the resource consumer groups.

Step 4. Create Resource Plan Directives

A resource plan directive specifies how much CPU should be allocated to the consumer group or when a session is switched to another consumer group based on specified I/O limits. A new directive can be created by calling the `CREATE_PLAN_DIRECTIVE` procedure, and an existing directive can be updated or deleted by calling the `DELETE_PLAN_DIRECTIVE` procedure.

The arguments `MGMT_P1` to `MGMT_P8` specify the percentage of the CPU to allocate for the consumer group at respective levels. For example, `MGMT_P1` specifies the percentage or weight of the CPU to allocate for the consumer group at the first level. Since the `RATIO` CPU allocation is supported only in a single-level resource plan, weights or ratios cannot be specified using `MGMT_P2` to `MGMT_P8` arguments.

With the release of Oracle Database 11*g*, automatic consumer group switching based on specified I/O thresholds is allowed. The threshold limit can be imposed on the maximum amount of data in megabytes a session can transfer and/or on the number of I/O requests that a session can execute. The argument `SWITCH_IO_MEGABYTES` specifies the number of megabytes of I/O that a session can transfer (read and write) before an action is taken, and the argument `SWITCH_IO_REQS` specifies the number of I/O requests that a session can execute before an action is taken. `UNLIMITED` is the default for both of these I/O related arguments.

The argument `MAX_UTILIZATION_LIMIT` specifies the absolute maximum CPU utilization percentage permitted for the consumer group. This value overrides any level allocations for CPU (`MGMT_P1` through `MGMT_P8`) and also imposes a limit on total CPU utilization when unused allocations are redistributed. If this functionality is needed, an administrator can specify this attribute and leave `MGMT_P1` through `MGMT_P8` to the default value of `NULL`. This attribute cannot be used in a subplan.

For example, if an application offers three service levels to clients, platinum, gold, and silver, create three consumer groups named `PLATINUM_CG`, `GOLD_CG`, and `SILVER_CG`, and then create the resource plan directives for plan `SERVICE_LEVEL_PLAN`, as shown below. The following PL/SQL code assumes that the `SERVICE_LEVEL_PLAN` plan and the `PLATINUM_CG`, `GOLD_CG`, and `SILVER_CG` consumer groups are already created in the pending area.

```
SQL> exec DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
        PLAN             => 'SERVICE_LEVEL_PLAN',
        GROUP_OR_SUBPLAN => 'PLATINUM_CG',
        COMMENT          => 'Platinum Service Level Customers',
        SWITCH_GROUP     => 'GOLD_CG',
        SWITCH_IO_MEGABYTES => 2048,
        MGMT_P1          => 25);

SQL> exec DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
        PLAN             => 'SERVICE_LEVEL_PLAN',
        GROUP_OR_SUBPLAN => 'GOLD_CG',
        COMMENT          => 'Gold Service Level Customers',
        MGMT_P1          => 15,
        ACTIVE_SESS_POOL_P1 => 32);

SQL> exec DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
        PLAN             => 'SERVICE_LEVEL_PLAN',
        GROUP_OR_SUBPLAN => 'SILVER_CG',
        COMMENT          => 'Silver Service Level Customers',
        MGMT_P1          =>  5,
        SWITCH_GROUP     => 'CANCEL_SQL',
        SWITCH_IO_REQS   => 2500);

SQL> exec DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
        PLAN             => 'SERVICE_LEVEL_PLAN',
        GROUP_OR_SUBPLAN => 'OTHER_GROUPS',
        COMMENT          => 'Other Customers',
        MGMT_P1          => 1,
        PARALLEL_DEGREE_LIMIT_P1 => 4);
```

The ratio of CPU allocation is 25:15:5:1 for the `PLATINUM_CG`, `GOLD_CG`, `SILVER_CG`, and `OTHER_GROUPS` consumer groups, respectively. Note that you must provide a directive for `OTHER_GROUPS`, the default consumer group. This ensures that a session that is not part of any of the consumer groups included in the currently active plan is allocated resources (as specified by the directive for `OTHER_GROUPS`).

If any CPU is not being used by one or more consumer groups, Oracle Database Resource Manager redistributes the CPU to the consumer groups that need it. If sessions exist only in the `PLATINUM_CG` and `GOLD_CG` consumer groups, the ratio of CPU allocation is 25:15 between the two groups. Therefore, `MGMT_P1` specifies the amount of CPU that the consumer group is guaranteed to get. The maximum amount of CPU that the consumer group can consume is 100%.

In the example, consumer group OTHER_GROUPS is limited to a maximum degree of parallelism of 4 for any operation, whereas none of the other consumer groups' degree of parallelism was limited. The consumer group GOLD_CG has a maximum of 32 concurrent active sessions. PLATINUM_CG consumer group will be automatically switched to GOLD_CG consumer group when the session consumes more than 2048 MB in I/O reads and writes. In the case of the SILVER_CG consumer group, the query will be canceled if the number of I/O requests exceed 2500.

Step 5. Assign Sessions to Resource Consumer Groups

Before Oracle Database Resource Manager is enabled, specify how user sessions are assigned to resource consumer groups. Do this by creating mapping rules to automatically assign each session to a consumer group upon session startup. Mapping rules are used to automatically place sessions into a consumer group.

The following rule maps user SCOTT to the DEV_GROUP consumer group every time he logs in.

```
SQL> exec DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING (
        ATTRIBUTE       => DBMS_RESOURCE_MANAGER.ORACLE_USER,
        VALUE           => 'SCOTT',
        CONSUMER_GROUP  => 'DEV_GROUP');
```

Sessions can be explicitly mapped to a consumer group or configured to automatically switch from one consumer group to another based on the amount of CPU or I/O consumed or the expected execution time using the SWITCH_TIME, SWITCH_IO_MEGABYTES, SWITCH_IO_REQS, or SWITCH_ESTIMATE directives.

Step 6. Validate the Pending Area

When changes are being made in the pending area, the VALIDATE_PENDING_AREA procedure can be called any time to ensure that the pending area is valid thus far. This is an optional but useful step that makes debugging easier, especially when making large number of changes to plans.

```
SQL> exec DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
```

Step 7. Submit the Pending Area

After the changes in pending area are validated, call the SUBMIT_PENDING_AREA procedure to make the changes active. The submit procedure performs validation, and the changes are made active only if all the changes in the pending area pass the validation. Submitting the pending area does not activate any new plan that was created. The new or updated plan information is just stored in the data dictionary. If a plan that is currently active is modified, the plan is reactivated with the new plan definition.

```
SQL> exec DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
```

The SUBMIT_PENDING_AREA procedure clears (deactivates) the pending area after successfully validating and committing the changes.

Granting Switch Privilege for Consumer Groups

In order to switch into a consumer group, a user or role must have permission. The following example grants user SCOTT the privilege to switch to consumer group OLTP. User SCOTT is also granted permission to grant switch privileges for OLTP to others.

```
SQL> exec DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP (
        GRANTEE_NAME   => 'SCOTT',
        CONSUMER_GROUP => 'OLTP',
        GRANT_OPTION   =>  TRUE);
```

The following PL/SQL command allows any user to switch into the BATCH_GROUP consumer group.

```
SQL> exec DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP (
        GRANTEE_NAME   => 'PUBLIC',
        CONSUMER_GROUP => 'BATCH_GROUP',
        GRANT_OPTION   =>  FALSE);
```

Enabling the Resource Plan

Once the resource plan is defined, the plan can be enabled by setting the RESOURCE_MANAGER_PLAN parameter to the resource plan name.

The following line in a text initialization parameter file activates Oracle Database Resource Manager upon database startup and sets the top plan as RESOURCE_MANAGER_PLAN.

```
RESOURCE_MANAGER_PLAN = RESOURCE_MANAGER_PLAN
```

An error message is returned if the specified plan does not exist in the data dictionary.

Integrating Oracle Database Resource Manager with Other Database Components

Oracle Database Resource Manager is integrated with other components and features of Oracle Database, such as Oracle Job Scheduler, parallel execution, security system, and database services.

Oracle Job Scheduler enables scheduling the execution of a task or job at a particular date and time or when a particular event occurs in the database. Jobs that share common characteristics and behavior can be grouped into job classes, and each job class can specify a resource consumer group as an attribute. Administrators can create scheduler windows to allow different resource plans to be activated at different times to change resource allocation among jobs during specific periods.

In the following example, the SERVICE_LEVEL_PLAN is enabled when WEEKDAY_WINDOW opens and it is disabled when WEEKDAY_WINDOW closes.

```
SQL> exec DBMS_SCHEDULER.SET_ATTRIBUTE (
        NAME     => 'WEEKDAY_WINDOW',
        ATTRIBUTE => 'RESOURCE_PLAN',
        VALUE    => 'SERVICE_LEVEL_PLAN');
```

Oracle Database Resource Manager is fully integrated into the database *security system*. Administrators can create, update, or delete resource plans and resource consumer groups using either a PL/SQL interface or the Oracle Enterprise Manager console. The administrator assigns default consumer groups and required privileges to database users, and users can switch their session's resource consumer group to change the resources available if the users have been granted the necessary privileges. In addition, the administrator can change the database user's default consumer group or move any active session from one group to another dynamically.

The *Adaptive Degree of Parallelism* (ADOP) feature takes Oracle Database Resource Manager allocations into account while choosing the optimal degree of parallelism for a parallel operation. ADOP attempts to optimize system utilization by automatically adjusting the degree of parallelism for parallel queries and parallel Data Manipulation Language (DML) operations.

Database services are logical abstractions for managing workloads in Oracle Database. Services divide workloads into mutually disjoint groupings. Each service represents a workload with common attributes, service-level thresholds, and priorities. For example, the Oracle E-Business Suite defines a service for each responsibility, such as general ledger, accounts receivable, order entry, and so on. Each database service has a unique name. Oracle Database Resource Manager maps services to consumer groups and priorities. This lets services be managed in the database in the order of their importance. For instance, you can define gold, silver, and bronze services to prioritize the order in which requests are serviced for the same application. When planning the services for a system, include the priority of each service relative to the other services. In this way, Oracle Database Resource Manager can satisfy the highest priority services first, followed by the next priority services, and so on.

Administrators can set-per user resource limits in Oracle Database using *profiles*. A profile is a named set of resource limits and password parameters that restrict database usage and instance resources for a user. User profiles set hard limits on resource consumption by various users in the database. Although Oracle Database continues to support the use of user profiles to implement hard resource limits, in recent releases of Oracle Database, server resource allocations and restrictions are primarily handled through Oracle Database Resource Manager. Oracle Database Resource Manager provides a more sophisticated way of managing database resources, since it can balance different requests for service against each other within the defined resource allocation plan and proactively control the resource consumption of long-running resource-intensive processes.

Monitoring Oracle Database Resource Manager

The following dynamic performance views help monitor the current status of Oracle Database Resource Manager, history of resource plan activations, current and historical statistics on resource consumption, and CPU waits by both resource consumer group and session.

- V$RSRC_PLAN
- V$RSRC_CONSUMER_GROUP
- V$RSRC_SESSION_INFO
- V$RSRC_PLAN_HISTORY
- V$RSRC_CONS_GROUP_HISTORY

Monitoring CPU Usage and Waits

The V$RSRC_CONSUMER_GROUP view provides the cumulative amount of CPU time consumed, cumulative amount of time waiting for CPU, and cumulative number of CPU waits by all sessions in each consumer group. It also provides a number of other measures that are helpful for tuning, such as CPU_WAIT_TIME. Not included in this measure are waits due to latch or enqueue contention, I/O waits, and so on.

Refer to the Using Oracle Database Resource Manager white paper for few examples.

About the resmgr: cpu quantum Wait Event

The resmgr: cpu quantum wait event occurs when a process is waiting to be scheduled by Oracle Database Resource Manager. The wait time is the actual time the session waited to acquire a CPU quantum.

```
Top 5 Timed Foreground Events

Event                          Waits           Time(s) Avg wait (ms)   % DB time       Wait Class
-------------------------------------------------------------------------------------------------
resmgr:cpu quantum             750,835         3,918   5               51.33           Scheduler
DB CPU                                         2,343                   30.70
latch: cache buffers chains    1,016,711       597     1               7.82            Concurrency
latch free                     347,811         369     1               4.83            Other
wait list latch free           224,914         231     1               3.03            Other
```

A high number of occurrences of this wait event does not necessarily indicate a performance issue. It is analogous to counting time spent in the OS queue. However, if performance degrades, identify the corresponding session and consider increasing the CPU allocation for the consumer group to which the session belongs.

Resources

Here are resources referenced earlier in this document:

- Part 1 of this series, "Introduction to Resource Management Using Oracle Solaris Resource Manager and Oracle Database Resource Manager": http://www.oracle.com/technetwork/articles/servers-storage-admin/o11-054-intro-rm-419298.pdf
- Part 2 of this series, "Effective Resource Management Using Oracle Solaris Resource Manager": http://www.oracle.com/technetwork/articles/servers-storage-admin/o11-055-solaris-rm-419384.pdf
- Part 4 of this series, "Resource Management Case Study for Mixed Workloads and Server Sharing": http://www.oracle.com/technetwork/articles/servers-storage-admin/o11-057-mixed-wl-rm-419381.pdf
- "Predefined Resource Plans and Consumer Groups" in the "Oracle Database Resource Manager Reference" section of Chapter 27, "Managing Resource Allocation with Oracle Database Resource Manager" in the *Oracle Database Administrator's Guide 11*g *Release 2 (11.2)*: http://download.oracle.com/docs/cd/E14072_01/index.htm
- "Using Oracle Database Resource Manager": http://www.oracle.com/technetwork/database/focus-areas/performance/resource-manager-twp-133705.pdf

And here are some additional resources:

- *Oracle Database Reference 11*g *Release 2 (11.2)*: http://download.oracle.com/docs/cd/E14072_01/index.htm
- Chapter 26, "Managing Resource Allocation with Oracle Database Resource Manager," in *Oracle Database Administrator's Guide 11*g *Release 2 (11.2)*: http://download.oracle.com/docs/cd/E14072_01/index.htm
- Chapter 27, "Oracle Scheduler Concepts," in *Oracle Database Administrator's Guide 11*g *Release 2 (11.2)*: http://download.oracle.com/docs/cd/E14072_01/index.htm
- "Database Instance Caging: A Simple Approach to Server Consolidation": http://www.oracle.com/technetwork/database/focus-areas/performance/instance-caging-wp-166854.pdf
- "The Oracle Database Resource Manager: Scheduling CPU Resources at the Application Level" by Ann Rhee, Sumanta Chatterjee, Tirthankar Lahiri: http://www.mvdirona.com/jrh/work/HPTS2001/Submissions/AnnRhee.pdf
- Oracle's note about server/hardware partitioning: http://www.oracle.com/us/corporate/pricing/partitioning-070609.pdf
- "Technical Overview of the Oracle Exadata Storage Server and Database Machine": http://www.oracle.com/us/solutions/datawarehousing/039572.pdf

Effective Resource Management Using Oracle Database Resource Manager

June 2011, Revision 1.0

Author: Giri Mandalika

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software
Engineered to Work Together