Database Administration Made Easy With Oracle9*i*

An Oracle White Paper
May 2002

ORACLE

INTRODUCTION

The Oracle server has evolved from a traditional RDBMS to a broad based Internet platform. Revolutionary growth of the Internet and emergence of the Oracle server as the lifeblood of any eBusiness has underlined the need for better management of Oracle databases to ensure round the clock availability and high performance. One of the key focus areas of Oracle9*i* has been to enhance Oracle Database manageability by automating routine DBA tasks, reducing complexity of administration and making it more self-tuning. A number of new features have been added to streamline space, memory, and resource management as well as other day-to-day database administrative tasks. This paper discusses the key features of Oracle9*i* that have been provided to simplify server management and ease database administrative tasks.

SPACE MANAGEMENT

Database space management has always been an important part of any database administrator's job. Administrators spend a significant amount of their time in planning and monitoring the space utilization in order to ensure uninterrupted database operations. New features introduced in Oracle9*i* simplify the space administration tasks, enforce best practices and eliminate much of the space management related performance tuning.

Automatic Segment Space Management

Oracle9*i* introduces a new scheme of managing free space inside a database segment such as tables or indexes. Currently data structures called the FREELISTS keep track of blocks within an object that can be used to insert new rows. In addition to the FREELLISTS, Oracle9*i* allows the free space within a segment to be tracked using bit maps. The new mechanism makes the task of managing space within an object completely transparent to the administrators by using bitmaps to track the space utilization of each data block allocated to the object. The state of the bitmap indicates how much free space exists in a given data block (i.e. > 75%, between 50 and 75%, between 25 to 50% or < 25%) as well as whether it its formatted or not. The new implementation eliminates the necessity to tune space management related controls (such as FREELISTS, FREELIST GROUPS and PCTUSED) thereby freeing database administrators from manually managing the space within a database object. At the same time, it improves the space utilization

since the database now has a more accurate knowledge of how free a data block is. This enables better reuse of the available free space especially for objects with rows of highly varying size. Additionally, the Automatic Segment Space Management feature improves the performance of concurrent DML operations significantly since different parts of the bitmap can be used simultaneously eliminating serialization for free space lookups.

The Automatic Segment Space Management feature is available only with locally managed tablespaces. A new clause SEGMENT SPACE MANAGEMENT in the CREATE TABLESPACE command allows administrators to choose between automatic and manual modes. A tablespace created with MANUAL segment space management continues to use FREELISTS for managing free space within the objects located in it. Any specification of PCTUSED, FREELISTS and FREELIST GROUPS parameters for objects created in this tablespace will be ignored. A new column called SEGMENT_SPACE_MANAGEMENT has been added to the DBA_TABLESPACES view to indicate the segment space management mode used by a tablespace.

Oracle Managed Files

Continuing in its quest to make life simpler for DBAs, Oracle9*i*'s "Oracle Managed File" (OMF) feature simplifies database administration by eliminating the need for administrators to directly manage the files of an Oracle database. This feature allows for specifying operations in terms of database objects. Oracle internally uses the standard operating system (OS) file system interfaces to create and delete files as needed for tablespaces, online logs and controlfiles. DBAs merely need to specify the location of these files using new initialization parameters. Oracle then ensures creation of a file with a unique name and deletes it when the corresponding object is dropped.

OMF eliminates errors caused by administrators specifying incorrect file names, reduces disk space wasted in obsolete files, and simplifies creation of test and development databases. It also makes development of portable third party applications easier by eliminating the need to put OS-specific file names in SQL scripts.

While the parameter DB_CREATE_FILE_DEST specifies the default location of datafiles, DB_CREATE_ONLINE_LOG_DEST_<n>, where n is any integer between 1 and 5, decide the default location for copies of online logs and controlfiles. If no DB_CREATE_ONLINE_LOG_DEST parameters are set, all the files (datafiles, controlfiles and online logs) will be created at the destination specified by the DB_CREATE_FILE_DEST parameter.

Oracle Managed datafiles, created by default, are 100 MB in size and are auto extensible with unlimited maximum size. The default size of Oracle Managed online logs is also 100MB.

Fully Locally Managed Database

Locally Managed Tablespaces, introduced in Oracle8*i*, liberated DBAs from having to manage the space within a tablespace manually. As opposed to the administrator deciding how the space is to be allocated and reused, this feature enables the Oracle Database to automatically use the available disk space in the most optimal manner. As a result, DBAs no longer need to worry about tablespace de-fragmentation issues and spend a significant amount of their time reorganizing database objects just to coalesce fragmented free space within a tablespace.

Beginning with Oracle9*i* Release 2, the SYSTEM tablespace can also of locally managed type. This allows creation of a fully locally managed database consisting of locally managed tablespaces only. Such a database uses the available space more efficiently, significantly improves the performance of DML (INSERT, UPDATE, DELETE) and DDL operations (DROP) and, frees administrators from some of the most time-consuming routine space management tasks.

A fully locally managed database can be created either by using the Database Configuration Assistant (DBCA) predefined templates or by specifying the EXTENT MANAGEMENT LOCAL clause in the CREATE DATABASE command. It is also possible to migrate an existing dictionary managed SYSTEM tablespace to locally managed type using either Oracle Enterprise Manager or the DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL procedure.

File Map and IO Topology for Intelligent Storage Arrays

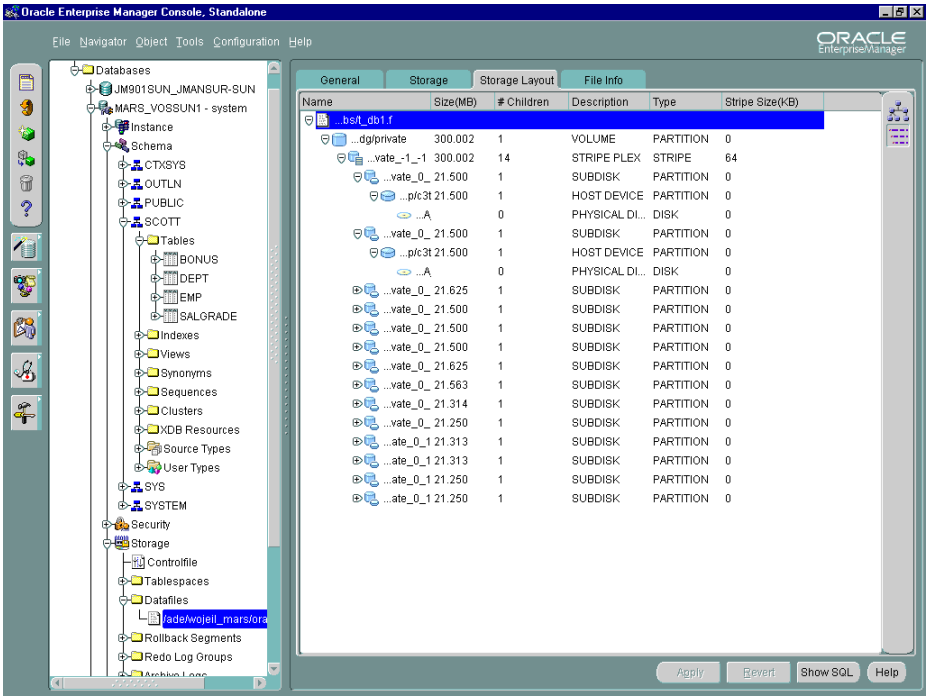
In an environment where datafiles are simply file system files or are created directly on a raw device, it is relatively straightforward to see the association between a tablespace and the underlying device. The mapping of files onto devices can be used together with device statistics to determine I/O performance.

However, with more common use of host based Logical Volume Managers (LVM), and sophisticated storage subsystems that provide RAID features, it is not always easy to determine the file to device mapping. In fact, to get an understanding of I/O performance, one must have detailed knowledge of storage hierarchy on which the data file resides.

Oracle9*i* Release 2 provides a number of dynamic performance views (i.e. v$ views) to map Oracle datafiles to intermediate layers of logical volumes and actual physical devices on supported storage sub-systems. Using these views, a DBA can locate the exact disk on which any block of a file resides. Oracle Enterprise Manager Storage Management makes it easy to view the files mapping, their logical grouping and the properties of each storage device. This information will help immensely in diagnosing and correcting problems related to storage layout.

This feature is currently available only for EMC disk subsystems.

Using Oracle Enterprise Manager, it is now possible to easily traverse through the storage mapping tree starting from an individual table all the way up to the physical disk block where the corresponding data is stored.



Default Temporary Tablespace

Storing temporary data in a permanent tablespace incurs the same space management overhead as that of managing permanent data and therefore can be inefficient. With the "Default Temporary Tablespace" feature in Oracle9*i*, DBAs can specify a database-wide default temporary tablespace at the time of database creation and eliminate the possibility of using inappropriate tablespaces for storing temporary data. There are several advantages to using temporary tablespaces: 1. The space management overhead for temporary tablespace is considerable less, compared to that of permanent tablespaces, 2. During recovery, the database avoids doing work in temporary segments, reducing recovery time, 3. The SYSTEM tablespace (a frequent default choice for temporary objects) is protected from space contention caused by creation of large or numerous user space allocations.

It is also possible to assign or change the default temporary table after creating the database. The default temporary tablespace created with Database Configuration Assistant (DBCA) or the CREATE DATABASE command is of locally managed type and all database users who are not explicitly assigned a temporary tablespace default to it. Administrators migrating from earlier versions of Oracle can assign any temporary tablespace, either dictionary or locally managed, as the default temporary tablespace.

Delete Datafiles

Oracle9*i* simplifies the maintenance of Oracle datafiles by allowing DBAs to automatically delete them after dropping a tablespace. With Oracle9*i*,

administrators can specify INCLUDING CONTENTS AND DATAFILES clause in the DROP TABLESPACE command, which uses OS services to delete the appropriate datafiles automatically. Oracle9*i* also ensures that failed DDL operations, which create OS files, such as ALTER TABLESPACE ADD DATAFILE, CREATE TABLESPACE etc., automatically remove any partially created underlying files. Oracle Enterprise Manager fully supports these options in the administration of tablespaces.

MEMORY MANAGEMENT

Memory is a critical system resource. Since memory access is many times faster than accessing data from disk, effective utilization of memory is necessary for optimal system performance. Administrators, therefore, continuously strive to tune memory related parameters to maximize system performance and ensure the most efficient use of system memory. Oracle9*i* seeks to automate much of the tuning process and allows administrators to alter the instance memory configuration dynamically. These features provide improved system performance, optimal memory utilization and reduced maintenance downtime.

Dynamic Shared Memory Management

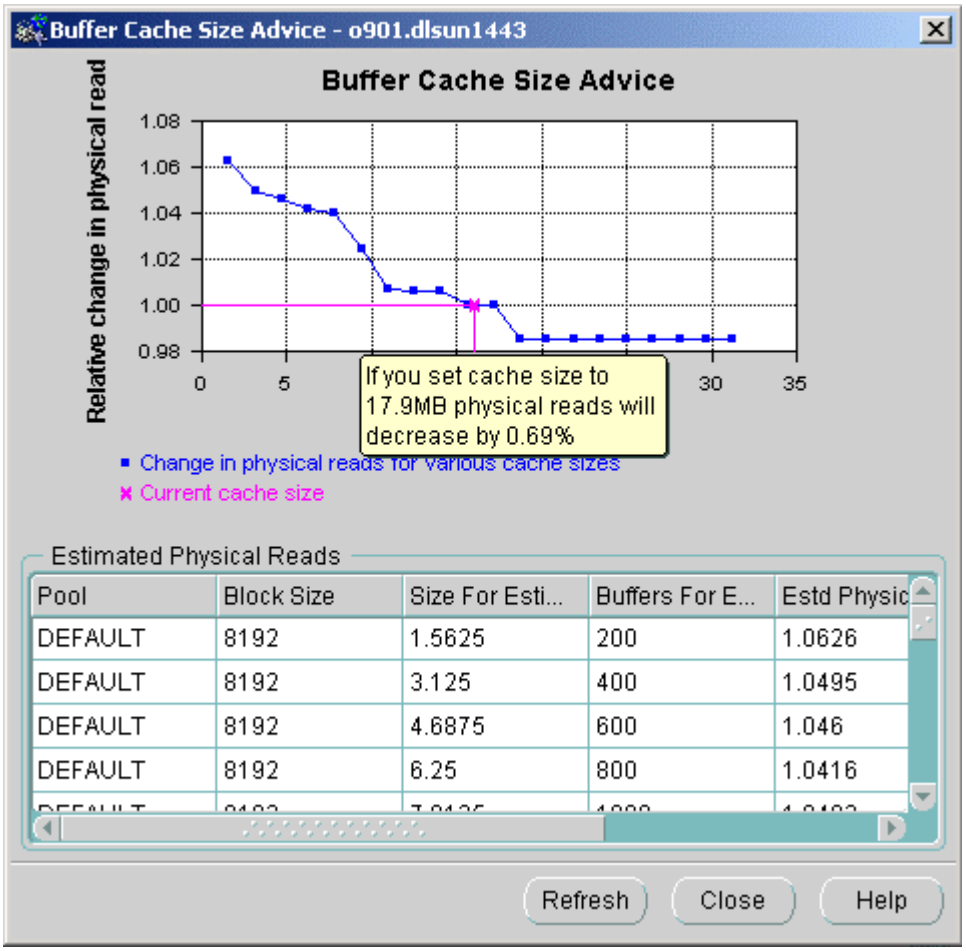
Oracle System Global Area (SGA) is a shared memory region, accessible to all threads of execution. Oracle9*i* makes it simple to add to or remove memory from an Oracle instance by allowing administrators to change the SGA configuration without shutting the instance down. To achieve this, all initialization parameters determining the size of SGA components, such as SHARED_POOL_SIZE, DB_CACHE_SIZE and LARGE_POOL_SIZE, have been made dynamic in Oracle9*i*. On supported OS platforms, DBAs can also modify Oracle's virtual address space to respond to the operating system's use of physical memory.

Dynamic SGA allows administrators to use the ALTER SYSTEM command to

- Grow the size of SGA components (Buffer Cache, Shared Pool, LARGE POOL).
- Shrink SGA by reducing the size of SGA components to an Oracle prescribed minimum.

Oracle9*i* also includes an advisory mechanism that can be used to determine the optimal sizes for the buffer cache and the shared pool. A new fixed view V$DB_CACHE_ADVICE contains "miss" rate predictions for twenty cache sizes ranging from 10% to 200% of the current cache size. This view can be used to determine whether the cache should be shrunk or grown for the present workload. Similarly, the view V$SHARED_POOL_ADVICE displays information about estimated change in the total parse time for different sizes of the shared pool ranging from 50% to 200% of the current size. The Enterprise Manager graphical

It is easy to find the optimal size of the buffer cache with Oracle9i. The Buffer Cache advisory can predict the number of "misses" for different size of buffer cache.



interface for these advisories make it very simple to view, interpret and use the system provided advice to size the SGA components optimally.

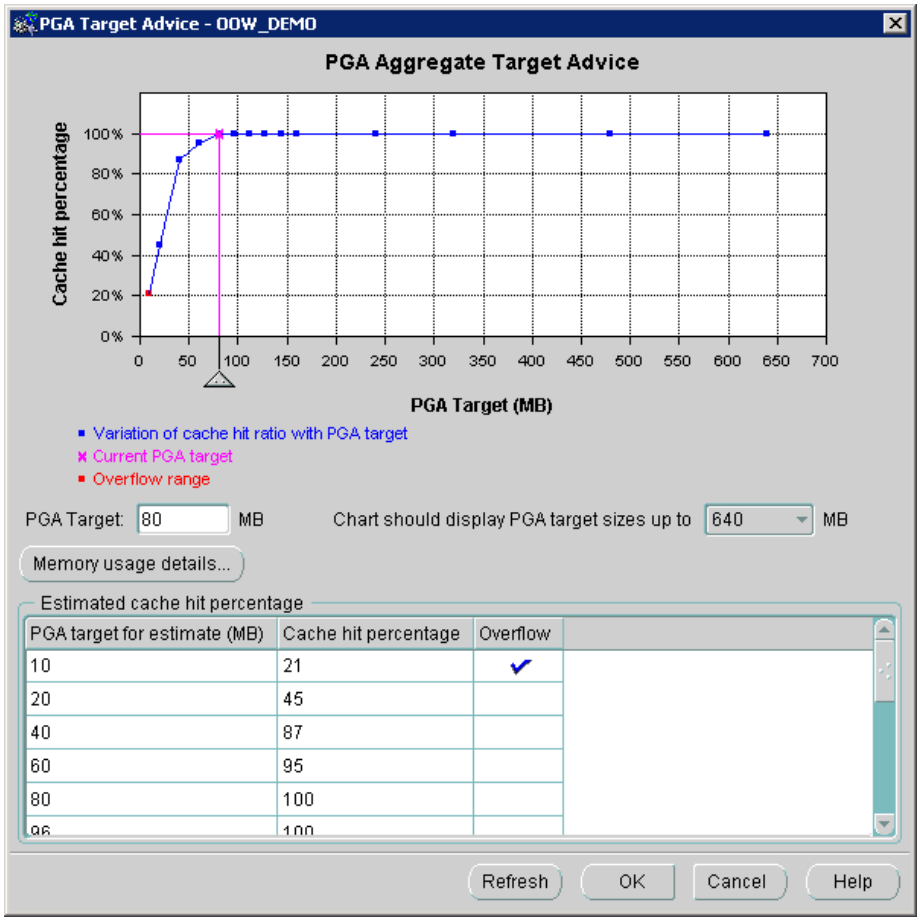
There are numerous advantages to dynamic SGA. For instance, it allows the buffer cache to relinquish memory to other SGA components (such as shared pool) if the memory requirements of these components increase. Conversely, it allows the size of buffer cache to increase at the expense of other components such as large pool and shared pool, if the buffer cache hit ratio is low. It is also possible to accommodate changes in the memory available to Oracle resulting either from changes in system hardware or changes resulting from OS resource manager allocations.

Self-Tuning SQL Execution Memory

Queries performing complex join or sort operations, typical in DSS environments, consume a large amount of memory for storing the "in-process" data. Oracle9*i* can automatically tune itself for the most efficient use of such SQL execution memory and optimal system performance. The goal of the tuning process is to adapt to the current circumstances, utilizing resources efficiently regardless of the load on the system. In this mode, all work areas allocated by the session are automatically tuned by Oracle for maximum system performance and administrators no longer have to manually adjust the value of parameters such as SORT_AREA_SIZE, HASH_AREA_SIZE, BITMAP_MERGE_AREA_SIZE and CREATE_BITMAP_AREA_SIZE.

While tuning working area sizes, Oracle9*i*'s self-tuning capability is not limited to just determining optimal values for the initialization parameters mentioned above. In Oracle9*i*, memory consuming algorithms (such as sort, hash join) have been modified to dynamically alter their memory usage at run time to ensure the best possible use of system memory and maximize system performance. At the same time, Oracle9*i* can also help administrators decide how large should the overall PGA size be in order to support the current workload. The view V$PGA_TARGET_ADVICE contains simulated prediction of the effect of increasing or decreasing the value of the parameter PGA_AGGREGATE_TARGET on the performance of long running operations. These predictions are generated by using the workload history to simulate the

With automatic SQL execution memory management, DBAs just need to specify the maximum PGA memory available for an instance. Oracle9i automatically distributes this memory among various active sessions in a manner which results in maximum performance gains



system performance for different settings of PGA_AGRREGATE_TARGET.

The auto-tuning mode is activated using two newly introduced initialization parameters PGA_AGGREGATE_TARGET and WORKAREA_SIZE_POLICY. While the PGA_AGGREGATE_TARGET parameter allows a DBA to advise an Oracle instance to limit its overall private memory consumption to the specified value, the WORKAREA_SIZE_POLICY parameter can be set to "auto" or "manual" to enable or disable the auto-tuning mode.

Although, the performance improvement will be most noticeable in heavy DSS workload environment, this feature will be very useful for OLTP environments as well. Most of the OLTP applications require complex reports to be run periodically whose performance can be improved due to self-tuning of SQL execution memory working areas. Users are encouraged to enable auto-tuning mode irrespective of the nature of workload.

Self-Tuning Direct I/O

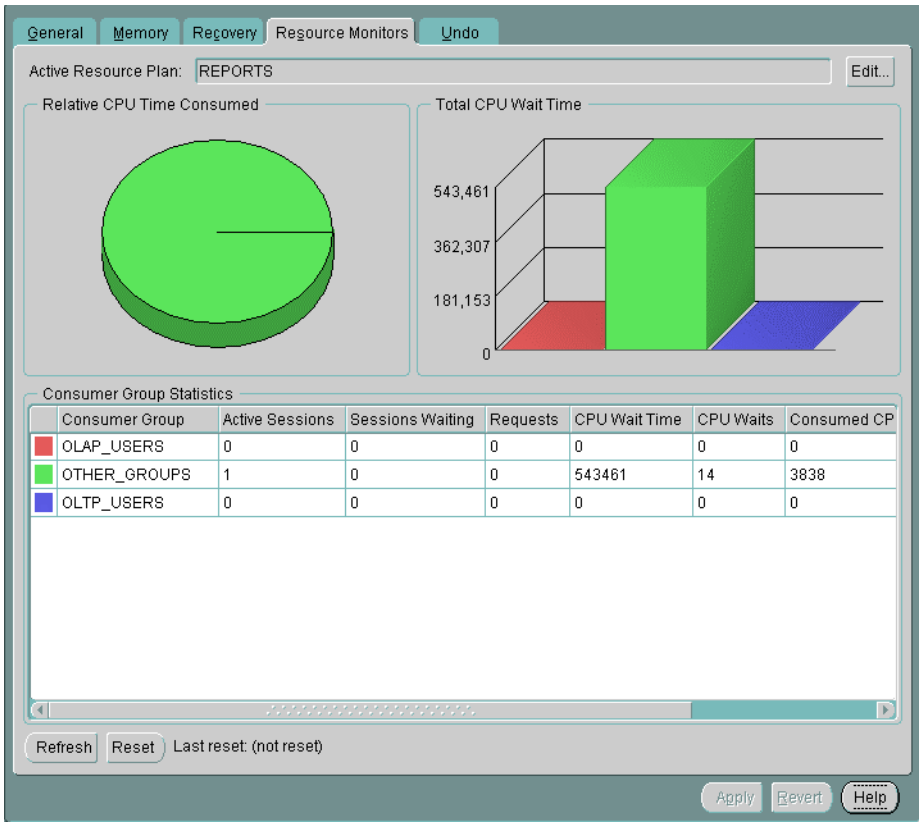
Direct path I/O is an I/O and caching mechanism for reading and writing disk blocks directly from a process's private memory without going through the SGA buffer cache. It improves the performance of operations such as table, index and LOB scans significantly.

Direct read uses a read ahead algorithm to prefetch extents so that when clients need to access a block from disk, it would have already been read and cached in the process's private memory. Prior to Oracle9*i*, the read-ahead is done one extent at a time. Oracle9*i* can prefetch multiple extents, which is very useful when the extent sizes are small. It can also dynamically adjust its direct read algorithm based on the workload to ensure optimal performance.

RESOURCE MANAGEMENT

Database Resource Manager was introduced in Oracle8*i* to enable database administrators to distribute available system resources among various users/application in a manner consistent with enterprise business needs. In absence of such a tool, it was not possible to differentiate one database session from another and one long resource intensive operation could slow the entire database. With Database Resource Manager, administrators can group database users into resource consumer groups and allocate percentages of CPU resources. Using the ability to dynamically move a session from a high priority to low priority group, DBAs can limit the impact of long operations on overall database performance. Oracle9*i* enhances this tool further with significant task automation and proactive resource management capabilities. Database Resource Manager in Oracle9i can automatically move a long running operation to a low priority consumer group, limit the number of such operations running concurrently and prevent their execution if they exceed the administrator defined execution time limit.

All Resource Manager features are fully supported by Oracle Enterprise Manager,



which makes setup and monitoring of resources much simplified.

Database Resource Manager in Oracle9i can automatically move a long running operation to a low priority consumer group, limit the number of such operations running concurrently and prevent their execution if they exceed the administrator defined execution time limit

Automatic Consumer Group Switching

With Oracle8i Database Resource Manager, a DBA can manually switch the consumer group for any running session. This capability can be used to switch the consumer group of a long running resource intensive session from a high priority to a low priority group. Oracle9*i* further enhances this feature by allowing administrators to specify criteria, which, if met, will cause the Database Resource Manager to automatically switch a session's consumer group.

Each plan directive referring to a resource consumer group in Oracle9*i* has three new parameters related to automatic change of consumer groups:

- SWITCH_TIME
- SWITCH_GROUP
- SWITCH_ESTIMATE

The Database Resource Manager switches a running session to *SWITCH_GROUP* if a session is active for more than *SWITCH_TIME* seconds. Once the session finishes its operation and becomes idle, it is automatically switched back to its

original group. If the value of the parameter *SWITCH_ESTIMATE* is set to TRUE, the Database Resource Manager uses its estimate of execution time to decide whether to switch the session before an operation even starts running. It is also possible to set up a cascading switching plan where a switched session can be switched further to a different group if its exceeds the switch time of its current group, provided there is no looping in the plan (i.e. a process can not be automatically switched back to its original group for the duration of its current active operation). Administrators can use this feature to manage the workload better by segregating long-running batch jobs from short OLTP transactions. Batch jobs can be automatically assigned to consumer groups with lower resource allocation during prime time to ensure good response time for OLTP users.

Operation Queuing

A heavily utilized system may experience severe performance degradation due to newly arriving operations unless the concurrent workload is restricted. The Database Resource Manager in Oracle9*i* provides a mechanism to allow administrators to set an *active session pool* per resource consumer group. An *active session* is defined as a session that is currently a part of an active transaction or query. Once this pool is filled with active sessions, all subsequent sessions attempting to become active are queued until other active sessions complete or become inactive.

Since the Database Resource Manager never blocks a running operation, an active switched session gets to run regardless of whether the "switch" group's active session pool is full or not. This is done in order to avoid any possible deadlocks that might arise from queuing sessions that still hold shared resources. Also, it avoids possible resource depletion (memory, temporary space) that might arise from indefinitely queuing sessions holding these resources. Because of this, the active session pool of a consumer group may be temporarily exceeded. However, once the running session becomes inactive and attempts to start another operation, it will be treated normally (i.e. may be queued if required). If the *SWITCH_ESTIMATE* parameter is set to TRUE and the operation is a part of an active transaction that holds no shared resources, then it will be queued when it is switched prior to execution.

Administrators can specify an optional time-out period (in seconds) as a part of resource plan directive. The time-out parameter indicates how long any session will wait on the queue. If a session waits in the queue longer than the time-out period, it will abort with an error. Users can then either ignore the error or trap it and resubmit the operation at a later time.

For Parallel Queries (PQ) or Parallel DML (PDML), the adaptive degree of parallelism is used to reduce the degree of parallelism based on the load of the instance. This is calculated before the PQ/PDML attempts execution. Once the degree of parallelism is calculated, the Query Coordinator (QC) session queues itself if the active session pool of its consumer group is full. Once the QC becomes

active, it requests a certain number of PQ slaves. These slaves do not count towards the number of active sessions for QC's resource consumer group. The entire parallel operation is counted as one active session.

Maximum Estimated Execution Time

In Oracle9*i*, the Database Resource Manager allows the administrator to specify a maximum estimated execution time for an operation using a new resource plan directive, *MAX_ESTIMATED_EXEC_TIME*. If this parameter is set, the Database Resource Manager estimates the execution time of an operation before it starts. If this estimate is longer than the maximum estimated execution time specified by the administrator, the operation will abort with an error. This way the administrator can set up a plan that will not accept any job that is exceptionally large and would thus use too many system resources.

Undo Quota

Finally, in Oracle9*i*, administrators are able to manage the resources consumed by a long running transaction by limiting the use of rollback (undo) space. A new resource plan directive *UNDO_POOL* allows DBAs to assign a quota for undo space to each consumer group. Whenever the total undo space used by sessions belonging to a consumer group exceeds its quota, they will not be allowed to make any further INSERT, UPDATE or DELETE until some undo space is freed by another session in the same group. If the consumer group's undo quota is exceeded during the execution of a DML statement, the operation will abort with an error. As soon as a process aborts or completes, the consumer group will be credited with freed undo space. The default value for the UNDO_POOL directive is UNLIMITED allowing sessions to consume as much undo space as available. This feature can be used with Automatic Undo Management (discussed later in the paper) as well as user defined rollback segments.

BACKUP AND RECOVERY MANAGEABILITY ENHANCEMENTS

In Oracle9i, a database may be backed using single RMAN command i.e. BACKUP DATABSE

Oracle Recovery Manager (RMAN) is a powerful tool for managing backup and recovery of Oracle databases. RMAN provides DBAs a flexible and feature rich tool that allows them to manage centralized backup and recovery of enterprise wide databases. Oracle9*i* features an enhanced RMAN that is easier to use and is more self-managing. A wizards driven graphical interface provided with Enterprise Manager helps administrators setup appropriate backup options and recover as needed.

Persistent Parameter Configuration

Oracle Enterprise Manager provides a graphical interface to RMAN to simplify the setup of appropriate backup policies for the managed environment and aid in recovery. Pre-defined backup configurations provide recommendations for Backup

strategies based on your business needs. In addition, Oracle9*i*, administrators can customize RMAN setup to their environment so that many of its parameters need not be specified with each backup/recovery operation. Using the new CONFIGURE command, a DBA can specify a backup retention policy (discussed later), backup duplexing, any available non-disk device, and the number and types of channels (including the parameters to be passed on to the media manager) persistently. Having done so, they can execute backup/recovery using much simpler commands. The CONFIG parameters also has a default value which will help DBAs, new to Oracle environment, get started with backup and restore of databases easily and quickly.

Retention Policy

RMAN 9*i* allows DBAs to define a backup retention policy. The backup retention policy directs RMAN as to which backup should be preserved and for how long. DBAs can specify a retention policy by defining

- The number of backups that should be preserved. (Redundancy)
- A period of time in which the database should be recoverable to any desired point in time. (Recovery Window)

Once *Redundancy* and *Recovery Window* are defined by the DBA, RMAN will preserve all required backups to honor this policy and delete backups automatically when they are no longer required. The backup retention policy can be set persistently using the CONFIGURE command. By specifying a retention policy, DBAs no longer need to manually manage the number of backup copies and space used by backups. With Oracle 9.2, Enterprise Manager's Backup Wizard allows setting additional options, such as backup retention policy, deleting obsolete backups and specifying the archivelog deletion policy.

Restartable Backup and Restore

RMAN backup & recovery operations have been optimized in Oracle9*i* to allow resumption from a point of failure. In Oracle9*i*, RMAN backs up only those datafiles that have not already been backed up. Similarly during a restore operation, RMAN scans the datafile headers and determine if it is necessary to restore the file to complete a database recovery. This feature improves the performance of backup/restore operations by eliminating unnecessary activity. It will also allow DBAs to segment a large backup into smaller pieces, each backing up a portion of database, without having to specify files to be handled by each sub-process.

Archive Log Failover

It is a common practice among DBAs to maintain multiple copies of archived logs to ensure recoverability in case one of the archiving destinations encounters a failure. In Oracle9*i*, RMAN takes advantage of this multiplexing. RMAN now validates each archive log being backed up for corruption. If an invalid log is detected, it reads other multiplexed archive log destinations to find a good copy of

the archive log being backed up. Also, once an archived log has been backed up, its copies will be deleted from all archived log destinations thereby improving space management.

Self Describing Backup

RMAN requires the use of a repository to store information about backups and its components. RMAN always stores this information in the control file of the target database. Optionally, users can create a recovery catalog and propagate the metadata in the control file into the catalog. Loss of either of these repositories can complicate recovery.

RMAN 9*i* provides a mechanism by which the restore of the control file repository can be performed *without* depending on the existence of a control file or recovery catalog. The "auto backup" mechanism ensures that after any backup, a backup of the control file is also created. RMAN can restore this control file and then use the information in the control file to restore the database. Consequently, DBAs face fewer risks when using either the control file or the recovery catalog as the repository for RMAN metadata. Even if everything is lost but the backups, DBAs can restore the database with a minimum of effort.

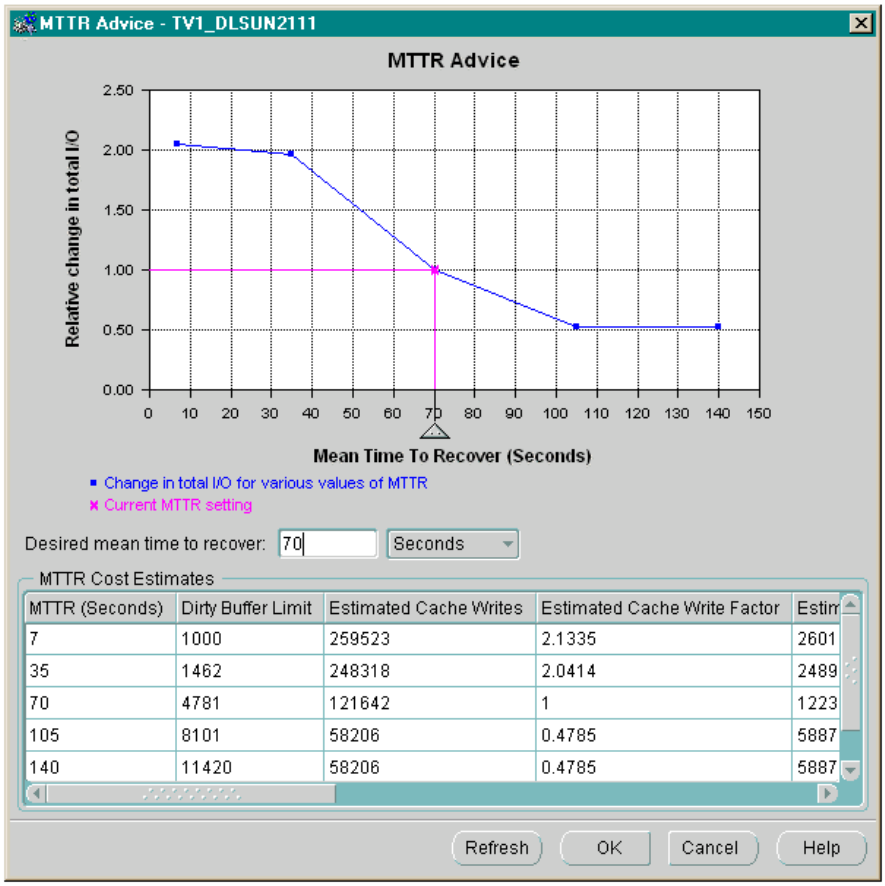
Administrator Bound Recovery Time

Many Service Level Agreements include a bound on the Mean Time To Recover (MTTR) after a failure. In order to meet these service levels, the database administrator must be able to reliably set a limit on the time it will take the database to recover from a crash or failure.

Oracle9*i* introduces Fast-Start Time-Based Recovery. This feature lets a DBA specify a target for recovery time in seconds, using the new parameter FAST_START_MTTR_TARGET, and the database server automatically determines appropriate values for the parameters that control recovery time, FAST_START_IO_TARGET and LOG_CHECKPOINT_INTERVAL. The algorithm takes into account such tasks as instance initialization, file open, reading the log, reading the data blocks from the data files, and writing the data blocks back to the data files. It initially uses defaults for these operations, but later substitutes actual statistics as they become available. The estimate therefore become more accurate over time, as the server learns more about its environment and expected I/O times. Because manually measuring the time it takes to complete these operations, and calculating values for parameters controlling recovery time is a complex task, this feature greatly simplifies and increases the accuracy of bounded database recovery time.

The V$INSTANCE_RECOVERY view can be used to monitor checkpointing, and it's impact on recovery time. Every 30 second, Oracle9*i* calculates an estimate of the current MTTR and displays this value in V$INSTANCE_RECOVERY. This allows the DBA to monitor the current estimated MTTR, and compare it to the target specified by FAST_START_MTTR_TARGET.

Since an aggressive setting of recovery time may increase the number of checkpoint writes significantly thereby degrading performance, Oracle9*i* Recovery Cost Estimator feature helps administrators set the value of the PAST_START_MTTR_TARGET parameter judiciously. A new view V$MTTR_TARGET_ADVICE can be used to determine the changes in checkpoint writes if the value of the recovery time parameter were to be altered.



This view displays the estimated number of physical I/O for different values of the FAST_START_MTTR_TARGET parameter raging from 10% to 200% of current setting. Data from this view is presented on an Enterprise Manager chart that shows the trade-offs between recovery time and run-time operation performance, as shown above.

TRANSACTION MANAGEMENT

Automatic Undo Management

In order to simplify management of rollback segments, Oracle9*i* introduces Automatic Undo Management where the database automatically manages allocation and management of Undo (Rollback) space among various active sessions. Administrators merely need to create an UNDO tablespace (using Enterprise Manager or the CREATE UNDO TABLESPACE…. SQL command). This

replaces the former process of creating a number of rollback segments and strategically assigning transactions to a rollback segment large enough to accommodate generated rollback data. This also frees DBAs from adjusting the attributes of rollback segments to avoid undo block and consistent read contention.

Undo tablespaces are special tablespaces used *solely* for storing undo information; creation of other database objects such as tables, indexes is not allowed in this tablespace. While a database may have more than one undo tablespace, each instance can use only one of them at a time. Undo blocks can be read by any instance (in Real Application Clusters environments) for "consistent read" purposes. Also, any instance can update an undo tablespace during transaction recovery as long as the tablespace is not already being used by any other instance either for undo generation by an active transaction or transaction recovery. It is possible to switch the undo tablespace being used by an instance in case the administrator wishes to create a different undo tablespace. The process of switching the undo tablespace is an online operation and happens without stopping the database or impacting users.

In a database using Automatic Undo Management, all transactions share a single undo tablespace. Any executing transaction can consume free space in this tablespace. Undo space is dynamically transferred from committed transactions to executing transactions in the event of space scarcity in the undo tablespace.

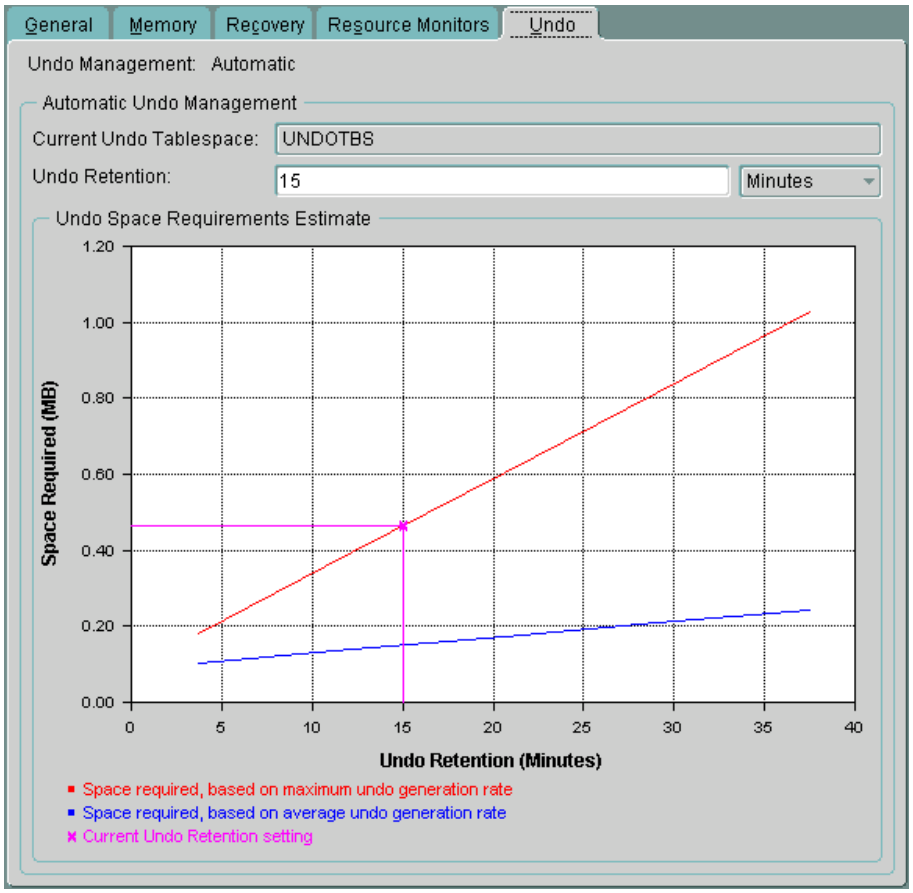
In order to avoid contention, the number of undo segments is dynamically adjusted to meet current workload requirements. An Oracle instance running in the Automatic Undo Management mode is capable of creating additional undo segments whenever required. Similarly some of the undo segments may be taken off-line and their space reclaimed whenever they are no longer needed. All these operations occur with no intervention by the administrator.

Automatic Undo Management feature also provides a way for administrators to exert control on undo retention. A DBA can specify the amount of undo to be retained in terms of wall-clock time (number of seconds). For example, to configure a database to support queries that run for 30 minutes or less, the DBA can simply set the undo retention parameter to 30 minutes. With retention control, users can configure their systems to allow long running queries to execute successfully without encountering ORA-1555 (Snapshot too old) errors. The undo retention time is specified using a new persistent INIT.ORA parameter, UNDO_RETENTION. This parameter is dynamic and hence can be changed anytime during database operation using the ALTER SYSTEM command.

In order to prevent a rogue transaction from consuming excessive undo space and thus impacting system operation, Oracle9*i* allows assigning an undo quota at the resource consumer group level using a newly introduced resource manager plan directive UNDO_POOL. Whenever the total undo consumed by a group exceeds its limit, its sessions will not be allowed to execute any more DML statements until

some undo space is freed by other sessions of this group after their transactions commit or abort. (*See Resource Management section for more details on this feature)*

A number of new performance views have been added to ease the monitoring and configuring the system to ensure efficient use of undo space. The view V$UNDOSTAT has been added in Oracle9*i* to show various undo/transaction



statistics. For example, the amount of undo consumed in the instance is presented in the view.

Oracle9*i* Enterprise Manager allows creating and assigning a new undo tablespace for a database. In addition, it also helps administrators size the undo tablespace optimally to support a given undo retention time using its undo tablespace sizing advisor.

Resumable Space Allocation

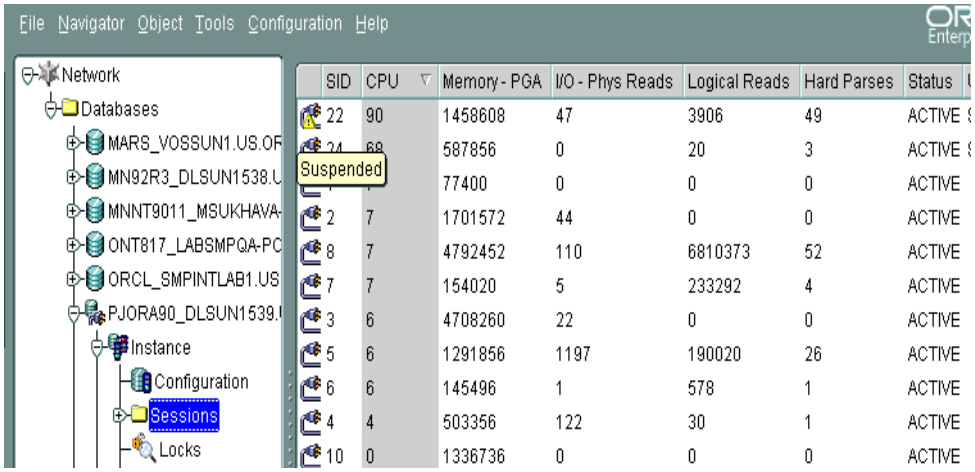
Large operations such as batch updates or data loads can encounter out-of-space errors after executing for a period of time, sometimes when they are just about to complete. Re-executing these processes could be a wasteful process, which could disrupt normal database operation.

Oracle9*i* introduces a new feature called "Resumable Space Allocation" which allows the system to suspend operations that encounter this kind of failure, fix the problem, and then *automatically* resume execution from the point of interruption. This feature enables application developers to write applications without worrying about running into space related errors. It also helps administrators avoid having to divide a large job into smaller sub-jobs and monitor progress of individual sub-jobs.

A statement can be executed in the "resumable" mode when explicitly specified by using the ALTER SESSION ENABLE RESUMABLE command. Virtually any kind of operation (e.g. PL/SQL stored procedure, Java stored procedure, queries, DML (*UPDATE, INSERT*) and DDL (*CREATE TABLE AS SELECT…., CREATE INDEX , INDEX REBUILD, ALTER TABLE MOVE PARTITION, ALTER TABLE SPLIT PARTITION, ALTER INDEX REBUILD PARTITION, ALTER INDEX SPLIT PARTITION, CREATE MATERIALIZED VIEW, CREATE MATERIALIZED VIEW LOG* etc.) can all be run as a "resumable" statement. A "resumable" operation will be suspended whenever it encounters one of the following types of failures:

- Out of space condition: The operation can not acquire any more space in a tablespace or when the tablespace itself can not extend by acquiring additional disk space from the OS.
- MAX Extents Reached: The number of extents in a table, index, temp segment, rollback segment, cluster, LOB, table partition, index partition already equals the maximum number of extents permitted by its DDL definition.
- User space quota is exceeded.

Once an operation is suspended, a warning to that effect is written in the alert log file. A notification is also sent using Oracle Enterprise Manager event sub-system alerting administrators about the suspended operation. More advanced corrective actions, such as extending a datafile or increasing the undo quota for the resource consumer group the suspended session belongs to, can be automatically performed using the new "AFTER SUSPEND" trigger. Any transactions executed within the trigger is automatically executed as an autonomous transaction and can therefore include operations such as inserts into a user table for error logging purposes. Oracle Enterprise Manager sessions screen highlights the suspended sessions to facilitate easy identification and displays all relevant information including the error encountered. The error data can also be accessed using the "DBMS_RESUMABLE" package and the DBA(USER)_RESUMABLE view.



When the problem that caused the failure is fixed, the suspended statement automatically resumes execution. If the operation encounters a transient problem no administrator intervention may be required to resume execution. For example, a resumable query running out of temporary space may resume automatically with absolutely no user or administrator intervention once other active queries complete. A resumable statement may be suspended and resumed multiple times during its execution.

Every resumable statement has a time-out period associated with it. The default value of time-out period is 2 hours but can be set to any value using Enterprise Manager or the ALTER SESSION ENABLE RESUMABLE TIMEOUT <time-out period in seconds> command. A suspended operation is automatically aborted if the error condition is not fixed within "time-out" period. An administrator can abort a suspended operation any time using Enterprise Manager or DBMS_RESUMABLE.ABORT() procedure.

While running Parallel DML/Query, if one of the server processes encounters a correctable error, it suspends its execution while other processes continue executing their respective tasks, until either they too encounter an error or are blocked by the suspended process. When the correctable error is resolved, the suspended process resumes execution but if it is aborted, the parallel operation aborts as well.

OTHER DAY-TO-DAY DATABASE ADMINISTRATIVE TASKS

Server Side Persistent Initialization Parameter File

Initialization parameters for an Oracle instance are currently specified using a client side parameter file popularly known as the INIT.ORA file. Although in most of the cases this file resides on the same machine as the Oracle instance, this is not a requirement since the front-end tools that are used to start the database (e.g. Server Manager, SQL Plus or Enterprise Manager) can be used from a remote machine as well. Since these tools need to read the parameter file and pass on the parameter values to the instance, it was necessary that this file should be accessible to the

front-end tools. This can lead to multiple parameter files for one instance and it may be difficult to keep them synchronized every time a parameter value is changed.

Oracle9*i* makes the management of the initialization parameter file simpler by introducing the persistent "Server Parameter file" (SPFILE). This file always resides on the server. Front-end tools merely need to specify the name of the SPFILE on the server to activate the instance. Similar to the current use of the INIT.ORA file, there is a default name and location for the SPFILE. Hence if the front-end tools do not explicitly specify the parameter file name, the default SPFILE is used if it exists. Introduction of the SPFILE slightly changes the behavior of the STARTUP command. A STARTUP command without an explicit PFILE (INIT.ORA) clause now tries to start the instance with default SPFILE settings. If the default SPFILE is not found, it attempts to start the instance using the default INIT.ORA file at the server side.

The SPFILE is automatically maintained by the server and any dynamic changes made to the values of parameters can be recorded in this file. A DBA has the ability to specify whether the change being made is temporary (i.e. the parameter will revert to its old value at next startup) or persistent. It is also possible to delete or reset a parameter from the SPFILE to allow an instance to revert to the default value of that parameter.

A SPFILE can be created from an INIT.ORA file using Enterprise Manager or the CREATE SPFILE command, before or after instance startup. It is possible to have multiple SPFILEs on a node, however only one of them can be used at a time. If multiple SPFILEs exist on a system, a DBA can direct an instance to use a particular SPFILE by setting the new SPFILE parameter in the INIT.ORA.

The SPFILE is automatically backed by RMAN during all backup operations. This eliminates the need to back up the parameter file separately and makes the database back up completely self-contained.

In order to simplify the management of Real Application Clusters (RAC), a single SPFILE is used by all member instances of a RAC cluster. Oracle9*i* still supports traditional PFILE (INIT.ORA) files that can be different for different RAC instances. However, if a SPFILE is used it must be the same on all instances. Since a SPFILE needs to be accessible from each instance node, it must reside on a shared device. In an RAC environment, a parameter can be configured to have different values for each instance in the cluster i.e. the parameter can be multi-valued. A multi-valued parameter has a default value and a value for each RAC instance that has modified the parameter to be different from its default value. The parameters that must be the same across all RAC instances are not multi-valued.

Multiple Block Size Support

Past versions of Oracle databases are composed of datafiles with a single block size. Oracle9*i* supports the creation of databases with multiple block sizes. An Oracle9*i*

database can be created with a default block size (specified by the initialization parameter DB_BLOCK_SIZE) and up to 5 alternate block sizes (2K, 4K, 8K, 16K and 32K). DBAs can configure "sub-caches" within the buffer cache for each alternate block size using new initialization parameters DB_<n>K_CACHE_SIZE , where n is the alternate block size. Tablespaces of any of the permitted block sizes may be created (using Enterprise Manager or CREATE TABLESPACE…….BLOCKSIZE <n>K) or plugged in (using the transportable tablespace feature) once the sub-caches for those block sizes have been configured. The SYSTEM tablespace in a database is always of the default block size.

This feature allows administrators to "transport" a tablespace from an OLTP database to a data warehouse for data archival and data mining purposes.

The initialization parameter DB_BLOCK_BUFFERS is deprecated in favor of DB_CACHE_SIZE, which defines the size of the standard block cache. Unlike DB_BLOCK_BUFFERS parameter, the value of DB_CACHE_SIZE as well as DB_<n>K_BLOCK_SIZE can be specified in terms of Bytes, Kilobytes (K), Megabytes (M) and Gigabytes (G). These parameters are dynamic; their values can be changed without shutting the instance down. *See memory management section for more details about dynamicity of these parameters.*

Cached Execution plans

Oracle9*i* facilitates better diagnosis of query performance problems by storing execution plan information for the cached cursors in a new dynamic performance view V$SQL_PLAN. DBAs and developers can now find out the actual execution plan used by queries at the time of reported performance problem and will be able to determine the cause of these problems better. This information is stored in memory as long as the cursor remains in the SGA.

Automatic Cost Based Optimizer (CBO) Statistics Gathering Enhancements

Oracle CBO requires statistics about data storage and distribution to generate accurate execution plans for queries. These statistics are generated using either the ANALYZE command or the DBMS_STATS package. Accuracy of these statistics depends largely on the judicious selection of the size of data sample being analyzed.

Also while using the CBO, histograms are used to store detailed information about data distributions that are non-uniform. This information helps the optimizer better estimate the selectivity of predicates that will result in more efficient execution plans. It is useful to create histograms when the application uses queries having:

- An equality predicate on a column which exhibits non-uniformity in repetition count (e.g. in a table having FIRST_NAME column, there are more "Mikes" than "Sushils").

- A range predicate on a column which exhibits non-uniformity in range (e.g. in EMPLOYEES table, 60% employees have been hired in the last two years out of 10 years since the company was established).

While creating histograms can result into significant performance improvements, it requires the knowledge of data distribution to decide when to create one.

In Oracle9*i*, DBAs can leave these decisions to the database itself. Using enhancements made to the DBMS_STATS package, they can direct the database to select the appropriate sample size to generate accurate statistics as well as identify the columns on which the histograms need to be created. This feature allows DBAs to ensure adequate optimizer statistics without being intimately familiar with either the data distribution or the structure of queries used by applications accessing the database.

Transaction Naming

Prior to Oracle9*i*, users could associate a comment with a transaction while committing using commit comment e.g. COMMIT COMMENT 'text'. This feature helps a DBA identify a specific transaction while resolving in-doubt transactions in a distributed computing environment. Oracle9*i* extends this capability by allowing users to name a transaction before it begins using the SET TRANSACTION NAME 'text' command. This transaction name can be used to monitor long running transactions or to search for a specific transaction from transaction auditing records in the log files using Log Miner.

CONCLUSION

Oracle databases have always been known for their scalability, availability and high performance. Oracle9*i* takes a giant step forward by providing a range of features and tools that lower the customer's total cost of ownership via simplified database management. Administrators can expect a significantly reduced burden keeping databases operational and enterprises should be able to maintain growth without having to hire as large an IT staff.

Database Administration Made Easy with Oracle9i
May 2002
Author: Sushil Kumar
Contributing Authors: Keith Lyon, Daniela Hansell

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
www.oracle.com

Oracle Corporation provides the software
that powers the internet.