

SERVER MANAGEABILITY: A DBA'S MANTRA FOR A GOOD NIGHT'S SLEEP

Sushil Kumar, Oracle Corporation

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 life blood 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 Oracle9i Release 9.0 has been to enhance Oracle Server 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. Management of Oracle networking (Net8) configuration has also been significantly improved with an industry standard LDAP compliant directory server. This paper discusses some of the features of Oracle9i Database Server that have been provided to simplify server management and ease database administrative tasks.

SPACE MANAGEMENT

AUTOMATIC SEGMENT FREE SPACE MANAGEMENT

Oracle9i introduces a new scheme of managing free space inside a database object. Currently data structures called the FREELISTS keep a track of blocks within an object that can be used to insert new rows. In addition to the FREELISTS, Oracle9i will allow free space tracking within the extents allocated to an object using bit maps. The new implementation will provide the following benefits:

- Eliminate the necessity to tune space management related knobs such as PCTUSED, FREELISTS, FREELIST GROUPS.
- Better space utilization, especially for objects with rows of highly varying size.
- Improved performance of concurrent INSERT operations.
- Better multi-instance behavior in terms of performance/space utilization.

ORACLE MANAGED FILES

Continuing in its quest to make life simpler for DBAs, Oracle9i "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 will allow for specifying operations in terms of database objects. Oracle will internally use 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 will then ensure creation of a file with a unique name and delete it when the corresponding object is dropped. OMF will reduce errors caused by administrators specifying incorrect file names, reduce disk space wasted in obsolete files, and simplify creation of test and development databases. It will also make development of portable third party applications easier since it eliminates 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 neither of the last two 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, will be 100 MB in size and will be auto extensible with unlimited maximum size. The default size of Oracle Managed online logs will also be 100MB.

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 very inefficient. With the “Default Temporary Tablespace” feature in Oracle9i, DBAs will be able to 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. The default temporary tablespace created with the `CREATE DATABASE` command will be of locally managed type and all database users who are not explicitly assigned a temporary tablespace will default to it. Administrators migrating from earlier versions of Oracle will be able to assign any temporary tablespace, either dictionary or locally managed, as the default temporary tablespace.

DELETE DATAFILES

Oracle9i simplifies the maintenance of Oracle datafiles by allowing DBAs to automatically delete them after dropping a tablespace. With Oracle9i, the administrators will be able to specify `INCLUDING CONTENTS AND DATAFILES` clause in the `DROP TABLESPACE` command which will use OS services to delete the appropriate datafiles automatically. Oracle9i will also ensure that failed DDL operations, which create OS files, such as `ALTER TABLESPACE ADD DATAFILE`, `CREATE TABLESPACE` etc., will automatically remove any partially created underlying files.

FILE MAP AND I/O TOPOLOGY FOR INTELLIGENT STORAGE ARRAY PRODUCTS

In an environment where datafiles are simply file system files or are created directly on a raw device, it is relatively straight forward 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.

The `V$FILEMAP` view in Oracle9i will provide a complete mapping of Oracle datafiles to intermediate layers of logical volumes and actual physical devices on supported storage sub-systems. Using this view, a DBA will be able to locate the exact disk on which any block of a file resides . This information will help immensely in diagnosing and correcting problems related to storage layout.

ORACLE DISK MANAGER

The Oracle Disk Manager (ODM) is a new disk management interface defined by Oracle. The ODM interface provides enhancements in the areas of file management and file I/O performance. This interface is designed for OS vendors to create a file management system optimal for Oracle performance.

MEMORY MANAGEMENT

DYNAMIC SHARED MEMORY MANAGEMENT

Oracle System Global Area (SGA) is a shared memory region, accessible to all threads of execution. Oracle9i makes it simple to add or remove memory to or from an Oracle instance by allowing administrators to change the

SGA configuration without shutting the instance down. Consequently, all initialization parameters determining the size of SGA components, such as `SHARED_POOL_SIZE`, `DB_CACHE_SIZE` & `DB_<n>K_CACHE_SIZE` and `LARGE_POOL`, have been made dynamic in Oracle9i. On supported OS platforms, DBAs will also be able to modify Oracle's virtual address space to respond to the operating system's use of physical memory. Dynamic SGA will allow 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.

DBAs are provided with the capability to monitor the progress of an operation involving change in the SGA configuration and abort it if needed.

Oracle9i will also include an advisory mechanism that can be used to determine an optimal size for the buffer cache. A new fixed view `V$DB_CACHE_ADVICE` will contain "miss" rate predictions for twenty cache sizes ranging from 10% up to 200% of the current cache size. This view can be used to determine whether the cache needs to be shrunk or grown for the present workload.

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

SELF TUNING SQL EXECUTION MEMORY

Oracle stores its data and control information for each server process in a memory region called Program Global Area (PGA). It is a non shared memory region created by Oracle when a server process is started. For environments with large number of complex queries, such as DSS systems, a big portion of the PGA is dedicated to working areas used by memory intensive row source operations such as sort, hash-join, group by, bitmap merge and bitmap index creation. In Oracle8i, the sizes of these working areas can be controlled and tuned using "init.ora" parameters `SORT_AREA_SIZE`, `HASH_AREA_SIZE`, `BITMAP_MERGE_AREA_SIZE` and `CREATE_BITMAP_AREA_SIZE`. For DSS or mixed workloads, these are some of the key parameters used by DBAs to control and limit PGA memory as well as to tune the system for better performance.

Oracle9i can automatically tune these parameters for the most efficient use of memory and optimal system performance. The goal of the tuning process will be to adapt to the given circumstances thus 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 to maximize system performance. The auto tuning mode is activated using newly introduced "init.ora" parameters `PGA_AGGREGATE_TARGET` and `WORKAREA_SIZE_POLICY`. While the parameter `PGA_AGGREGATE_TARGET` allows a DBA to advise an Oracle instance to limit its overall private memory consumption to the specified value in the auto tuning mode, `WORKAREA_SIZE_POLICY` can be set to "auto" or "manual", either at the system or at the individual session level, to enable or disable auto tuning mode respectively.

While tuning working area sizes, Oracle9i's self tuning capability is not limited to just determining optimal values for the "init.ora" parameters mentioned above. In Oracle9i, memory consuming algorithm (such as sort, hash join) have been modified to dynamically alter their memory profile at run time to ensure the best possible use of system memory and maximize system performance.

Oracle9i maintains backward compatibility by allowing DBAs to deactivate the auto tuning option. If the parameter `PGA_AGGREGATE_TARGET` is not set, auto tuning mode is deactivated automatically.

This feature will help customers reduce the time required to tune the memory usage for their DSS applications. Scripts may no longer need to include an "ALTER SESSION...." command to modify the memory configuration

for individual queries. The saving gained from improved use of memory should translate to better throughput where large number of users are on the system, as well as improved response time for queries.

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 PGA 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 a I/O and caching mechanism for reading and writing disk blocks directly from a process's private memory without going through the 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 the data layer clients need to access a block from disk, it would have already been read and cached in the process's private memory. Currently, the read ahead is done one extent at a time. Oracle9i can prefetch multiple extents which will be very useful when the extent sizes are small. It can also dynamically adjust direct read algorithm based on the workload to ensure optimal performance.

RESOURCE MANAGEMENT

Oracle Database Resource Manager, introduced in Oracle8i, provides a powerful mechanism for DBAs to manage system workload. The Database Resource Manager allows administrators to group users into resource consumer groups and assign CPU resources to these groups in a manner consistent with enterprise business needs. Oracle9i features a significantly enhanced Database Resource Manager with new features such as Automatic Consumer Group Switching, Query Queuing, Maximum Estimated Execution Time and Rollback Quota.

AUTOMATIC CONSUMER GROUP SWITCHING

With the existing 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. Oracle9i 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 Oracle9i will have three new parameters related to automatic change of consumer groups:

- *SWITCH_TIME*
- *SWITCH_GROUP*
- *SWITCH_ESTIMATE*

The Database Resource Manager would switch 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 will be automatically switched back to its original group. If the value of the parameter *SWITCH_ESTIMATE* is set to TRUE, the Database Resource Manager will use 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 it 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 day time to ensure good response time for OLTP users.

OPERATION QUEUING

A heavily utilized system could experience severe performance degradation due to newly arriving operations unless the concurrent workload is restricted to allow for minimum acceptable performance level. The Database Resource Manager in Oracle9i 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 will be 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 will be used to reduce the degree of parallelism based on the load of the instance. This will be calculated before the PQ/PDML attempts execution. Once the degree of parallelism is calculated, the Query Coordinator (QC) session will queue itself if the active session pool of its consumer group is full. Once the QC becomes active, it will request a certain number of PQ slaves. These slaves will not count towards the number of active sessions for QC’s resource consumer group. The entire parallel operation will be counted as one active session.

MAXIMUM ESTIMATED EXECUTION TIME

In Oracle9i, the Database Resource Manager will allow 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 will estimate the execution time 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 even accept any job that is exceptionally large and would use too much system resources.

UNDO (ROLLBACK) QUOTA

Finally, in Oracle9i, administrators will be 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 System Managed Undo as well as user defined rollback segments.

OTHER DAY TO DAY DATABASE ADMINISTRATIVE TASKS

AUTOMATIC UNDO MANAGEMENT

In order to simplify management of rollback segments, Oracle9i introduces Automatic Undo Management which enables the database server to automatically manage allocation and management of Undo (Rollback) space among various active sessions. Administrators merely need to create an UNDO tablespace (Using the CREATE UNDO TABLESPACE... SQL command) with sufficient disk space instead of manually creating a number of rollback segments and strategically assigning transactions to rollback segments 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 can have more than one undo tablespace, each instance will only use one of them at a time. However undo blocks can be read by any instance (in Real Application Cluster environments) for “consistent read” purposes. Also, any instance can update an undo tablespace during transaction recovery as long as it 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 decides to create a different undo tablespace. The process of switching the undo tablespace is an online operation and happens totally behind the scene without stopping the database or impacting users.

In a database using Automatic Undo Management, all transactions execute in common undo space and any executing transaction can consume free space in this tablespace. Undo space will be 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 the 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 made off-line and the space used by them reclaimed whenever they are no longer needed due to reduced workload concurrency. All these operations occur behind the scene with absolutely no intervention from the administrator.

Automatic Undo Management feature also provides a way for administrators to exert control on the undo retention. A DBA can specify the duration for which the undo data should 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 persistent and is remembered across system shutdowns. It can be specified using a new 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, Oracle9i 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 view 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 Oracle9i to show various undo/transaction statistics. For example, the amount of undo consumed in the instance is presented in the view.

In Oracle9i, DBAs have an option of managing undo either manually (Rollback Undo Mode) – for backward compatibility -- or Automatically. The Automatic Undo Management mode can be activated by setting the

initialization parameter `UNDO_MODE` to `AUTO`. This mode requires availability of at least one undo tablespace which can be specified using the initialization parameter `UNDO_TABLESPACE`. The specification of `UNDO_TABLESPACE` parameter is optional since the server can automatically activate an undo tablespace. If the database contains only one undo tablespace, there should be no need to explicitly set this parameter. This parameter has been provided primarily to allow administrators to assign an undo tablespace with a particular instance in the Real Application Clusters environment.

RESUMABLE SPACE ALLOCATION

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

Oracle9i introduces a new feature called “Resumable Space Allocation” which allows users to suspend large operations that encounter this kind of failure, fix the problem, and then *automatically* resume execution from the point of interruption. This feature will enable application developers to write applications without worrying about running into space related errors. It will also help 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 be it a 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:

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

Once the operation is suspended, a warning to that effect will be written in the alert log file. A trigger can also be used on a new event called “AFTER SUSPEND” either to generate a notification or take corrective actions. Any transactions executed within the trigger will automatically be executed as an autonomous transaction and can therefore include operations such as inserts into a user table for error logging purposes. Users may also access the error data 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 which may disappear automatically after some time such as temporary tablespace shortage, no administrator intervention may be required to resume its execution. 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 the `ALTER SESSION ENABLE RESUMABLE TIMEOUT <time-out period in seconds>` command. A suspended operation will automatically be aborted if the error condition is not fixed within “time-out” period. An administrator can abort a suspended operation any time using `DBMS_RESUMABLE.ABORT()` procedure.

While running Parallel DML/Query, if one of the server processes encounters a correctable error, it will suspend 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.

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, which can be used to start the database such as Server Manager, SQL Plus or Enterprise Manager, could 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.

Oracle9i makes the management of the initialization parameter file simpler by introducing the “Server Parameter file” (SPFILE). This file will always reside 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 will be 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 will be 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”) option will now try to start the instance with default SPFILE settings. If the default SPFILE is not found, it will attempt to start the instance using the default “init.ora” file at the server side.

The SPFILE is automatically maintained by the server and any changes made to the values of parameters using the ALTER SYSTEM command can be recorded in this file. A DBA will have 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 the CREATE SPFILE command which can be executed 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 using a newly introduced parameter with the same name i.e., SPFILE.

In order to simplify the management of Oracle Parallel Server (OPS), a single SPFILE can be used to start the entire OPS cluster. Oracle9i still supports traditional PFILE (init.ora) files which can be different for different OPS 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 OPS environment, a parameter can be configured to have different values for each instance in the cluster i.e. parameter can be multi-valued. A multi-valued parameter has a default value and a value for each OPS instance that has modified the parameter to be different from its default value. The parameters which must be the same across all OPS instances are not multi-valued.

MULTIPLE BLOCK SIZE SUPPORT

Past versions of Oracle databases are composed of datafiles with single block size. Oracle9i supports creation of databases with multiple block sizes. An Oracle9i 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 size may be created (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 will always be of the default block size.

DBAs can use this feature to locate objects in tablespaces of appropriate block size in order to maximize I/O performance. Large summary tables, which are often fully scanned, can be created in tablespaces with larger block size while the tables used by OLTP applications can use those with smaller block size. This feature will also allow 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 also dynamic and hence 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

Oracle9i will facilitate better diagnosis of query performance problems by storing execution plan information for the cached cursors in a set of newly introduced dynamic performance views. Three new views namely, `V$SQL_PLAN`, `V$SQL_PLAN_ENV` and `V$SQL_PLAN_STATS`, have been provided in Oracle9i to display, for each cached cursor, information such as the execution plan, the compilation environment and the execution statistics for each operation in the execution plan (optional). DBAs and developers will now be able to 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.

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 which are non-uniform. This information helps the optimizer better estimate the selectivity of predicates which 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 Oracle9i, DBAs will be able to leave these decisions to the database itself. Using enhancements made to the `DBMS_STATS` package, they will be able to 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 will allow 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.

IDENTIFICATION OF UNUSED INDEXES

Unused indexes in a database not only waste valuable disk space but also reduce the performance of DML operations. Identification of such indexes has, therefore, been of great interest to DBAs. Oracle9i Server allows administrators to track index usage by providing a new “`MONITORING USAGE`” option. Monitoring can either be enabled for a particular index or all indexes on a given table. A new view `V$OBJECT_USAGE` contains the usage statistics for indexes being monitored.

TRANSACTION NAMING

Prior to Oracle9i, users could associate a comment to 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. Oracle9i 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.

BACKUP & RECOVERY MANAGEABILITY ENHANCEMENTS

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 which allows them to manage centralized backup and recovery of enterprise wide databases. Oracle9i features an enhanced RMAN which is easier to use and is more self managing.

PERSISTENT PARAMETER CONFIGURATION

In Oracle9i, administrators will be able to 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 will be able to 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 will be able to execute backup/recovery using much simpler commands. The `CONFIG` parameters will also have 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 9i allows DBAs to define a backup retention policy. The backup retention policy directs RMAN 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 will no longer need to manually manage the number of backup copies and space used by backups.

RESTARTABLE BACKUP AND RESTORE

RMAN backup & recovery operations have been optimized in Oracle9i to allow them to resume from the point of failure. In Oracle9i, RMAN will backup only those datafiles which have not already been backed up. Similarly during a restore operation, RMAN will scan the datafile headers and determine if it is necessary to restore the file to complete a database recovery. This feature will improve the performance of backup/restore operations by eliminating unnecessary backups and restores. It will also allow DBAs to segment a large backup into smaller pieces, each backing up a portion of database, without having to specify individual files to be backed up 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 destination encounters a failure. In Oracle9i, RMAN takes advantage of this multiplexing and backs up a corrupt archive log from another multiplexed destinations enhancing recoverability of Oracle databases. 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 make backup and recovery complicated.

RMAN 9i provides a mechanism in 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. If everything is lost but the backups, then users can restore the database with a minimum of effort.

FAST-START TIME-BASED RECOVERY LIMIT

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.

Oracle9i 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 will automatically determine appropriate values for the parameters that control recovery time, `FAST_START_IO_TARGET` and `LOG_CHECKPOINT_INTERVAL`. The algorithm will take into account such tasks as 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 will initially use internal estimates for these operations, but will substitute real world data as it becomes available. The estimate will 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 will greatly simplify and increase the accuracy of bounding database recovery time.

The `V$INSTANCE_RECOVERY` view is used to monitor checkpointing, and its impact on recovery time. Every 30 second, Oracle9i will calculate an estimate of the current MTTR and will display this value in `V$INSTANCE_RECOVERY`. This allows the DBA to monitor the current estimated MTTR, and compare it to the target specified by `FAST_START_MTTR_TARGET`.

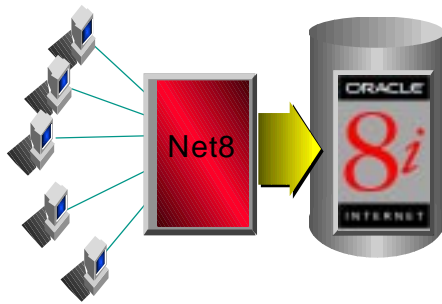
NET8 MANAGEABILITY ENHANCEMENTS

NET8 AND LDAP DIRECTORY SERVICES

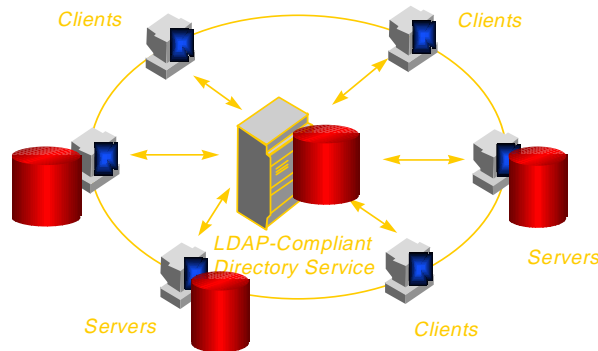
Today, network information is stored in multiple systems and in multiple directory formats. With new requirements of Internet computing and new e-business technologies, there is a growing need for a common infrastructure to serve as a foundation for management and configuration of all data and resources in the network. Such a common infrastructure will reduce the cost of managing and configuring resources in heterogeneous networks and, thereby, lower the total cost of ownership.

LDAP, standing for Lightweight Directory Access Protocol, is such a common infrastructure and an industry standard for Internet directory services. LDAP has rapidly emerged as a centralized vehicle for managing and configuring a distributed, heterogeneous network, a vital platform for enabling integration among applications and services on the network, and a critical infrastructure for network security.

Oracle's network services functionality, Net8, supplies the industry's most comprehensive, enterprise-wide data access solution for complex computing environments. Net8 enables Oracle databases to seamlessly communicate in client-server and server-server configurations, allowing databases and applications to reside on different computers.

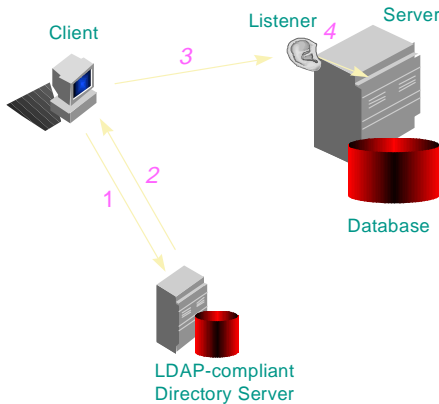


Net8's support of LDAP-compliant directory service provides a centralized vehicle for managing and configuring a complex enterprise network. The directory service becomes the central repository for all information on databases, network components, user and corporate policies and preferences, user authentication and security information, replacing client-side and server-side localized files. Administrators can centrally manage these objects and all systems on the heterogeneous network can refer to the directory for information.



NET8 DIRECTORY NAMING AND NEW FEATURES

Directory naming is the process of resolving an alias using an LDAP-compliant directory service. Net8 directory naming allows net service names to be stored in and retrieved from an LDAP-compliant directory service. Net service names stored in such a directory are accessible by any client machine in the network as long as the client has sufficient access privileges. An explanation of directory naming working procedure is shown as below:



- 1) A client initiates a connect request providing a connect identifier (e.g., sales.us.oracle.com).
- 2) The connect identifier is resolved to a connect descriptor (e.g., port number, host name, protocol, instance, ...) by a directory server, and sends back to the client.
- 3) The client makes the connect request to the address provided in the connect descriptor.
- 4) A listener receives the request and directs it to the server.

These centralized network names and addresses facilitate an easy administration of changes and updates, eliminate the need for an administrator to make changes to what potentially could be hundreds thousands of clients.

Net8 provides Net8 Manager and Net8 Configuration Assistant, two easy-to-use graphical tools, for configuration and management of the entire network and its components. Several new features in Net8 improve Net8 LDAP support significantly and make Oracle8i the most LDAP oriented database today. These new features are summarized as below:

- Improvements in Net8 Manger and Net8 Configuration Assistant ease LDAP-compliant directory configuration and management considerably. Net8 Configuration Assistant allows to create multiple entries of Oracle Context that stores Oracle databases and net service names. And Net8 Manager allows to change the default naming context so that all the information under different naming context can be viewed and edited. These improvements facilitate to manage a complex hierarchical naming structure easily.
- In addition to Oracle Internet Directory and Active Directory, Net8 supports Novell Directory Service. This new support extends Oracle9i's interoperability. Today, Oracle9i supports the most LDAP-compliant directory than any other competitor.
- Oracle Names Proxy enables customers to migrate seamlessly from Oracle Names to directory naming. The proxy allows Oracle Names client machines to connect to an LDAP-compliant directory server through Oracle Names server acting as a proxy, therefore only one system needs to be maintained during the migration period. This feature is especially useful to big customers' LDAP migration with thousands of Oracle Names client machines deployed, because it is almost impossible for so many client machines to convert to LDAP simultaneously.

CONCLUSION

Oracle databases have always been known for their scalability, availability and high performance. Oracle9 takes a giant step forward by providing a range of features and tools that lower the customer's total cost of ownership with simplified database management. Administrators can expect a significantly reduced amount of manual intervention to keep databases operational and enterprises should be able to maintain their sustained rate of grow without having to hire additional DBAs.