

Performance Tuning of Oracle for NetWare®

White Paper
July 1999

ORACLE®

Hitachi

Hitachi PC

Four 400Mhz CPUs
4GB RAM
Qlogic and Emulex fiber channel controllers
Two Hitachi 5800 storage cabinets
each cabinet contains 28 18GB disks

Oracle and this author appreciate the cooperation extended by these hardware vendors.

MEMORY

When configuring memory for Oracle on NetWare, take all precautions necessary to keep from running out of memory. Keep in mind that there is a finite amount of memory. More memory will increase performance if tuned properly as explained below. NetWare 5 can handle up to 4GB of memory. The database buffer cache, the shared pool, redo log buffers and sort area size determine the amount of memory used by Oracle. The System Global Area (SGA), made up of the database buffer cache, the shared pool and the redo log buffers, is allocated when the instance is started. Setting the SGA too large will cause cache memory allocation errors because there will not be enough memory available for the PGA. The Program Global Area (PGA) is memory used by Oracle outside of the SGA. The memory used for the PGA is allocated as it is needed for every Oracle process. Each user session uses a minimum of 250KB. Additional memory is required by each query slave for sorting. Sorting is described in more detail below. Data warehouse/data mart applications, or any operations which call for excessive sorting, require additional memory for the PGA. It is possible to use more memory for the PGA than for the SGA.

Starting NetWare With The -U Option

NetWare 5 reserves 512MB of OS address space for the user address space. If the system has 4GB of memory, starting NetWare with the -u 0 option will allow access to 512MB of additional memory that is ordinarily reserved for the user address space.

```
c:>\nwserver\Server.exe -u 0
```

For performance reasons, Oracle does not ordinarily run in the user address space. Starting NetWare with this parameter will allow Oracle access to an additional 512MB of memory.

Database buffer cache

The database buffer cache is an area in the SGA that is used to store the most recently used data blocks. The size of each buffer in the buffer cache is equal to the size of a data block and is specified in bytes by the DB_BLOCK_SIZE parameter. The number of buffers equal to the value of the DB_BLOCK_BUFFERS parameter. The database buffer cache is determined by the DB_BLOCK_BUFFERS x the DB_BLOCK_SIZE. Use the cache hit ratio to determine if this is sized properly. For OLTP applications, a 90% hit ratio is optimal. A much smaller hit ratio is acceptable for data warehouse/data mart applications or applications that need to do full table scans. If the cache hit ratio is below 90% check to see if the application is doing unnecessary full table scans. The cache hit ratio can be determined querying the dynamic performance table V\$SYSSTAT. V\$SYSSTAT contains the following three values: db block gets, consistent gets and physical reads. These three values are used to calculate the hit ratio with this formula: Hit Ratio = 1 - (physical reads/(db block gets + consistent gets))

```
SELECT name,value  
FROM v$sysstat  
WHERE name IN ('physical reads', 'db block gets', 'consistent gets');
```

Shared pool

The shared pool contains two main structures, the library cache and the data dictionary cache. The library cache contains SQL® and PL/SQL™ areas and stores statement text, parsed code and execution plans. The data dictionary cache contains table, column definitions and privileges from the data dictionary tables. The shared pool size is more important for OLTP applications than for data warehouse/data mart applications. For OLTP applications, the hit ratio should be above 90%. Data warehouse/data mart applications may have a much smaller hit ratio. The following SQL statement to determine the get hit ratio:

```
SELECT namespace, gethitratio
FROM v$librarycache;
```

The following SQL statement displays whether statements that have already been parsed have been aged out of the cache. The number of reloads should not be more than 1% of the number of pins:

```
SELECT sum(pins) "PINS", sum(reloads) "RELOADS",
       100*sum(pins)/(sum(pins)+sum(reloads))
FROM v$librarycache;
```

The miss ratio for the data dictionary should be less than 15% for all types of applications. If it is higher than this, consider increasing the SHARED_POOL_SIZE parameter. The ratio of the sum of all GETMISSES to the sum of all GETS should be less than 15%:

```
SELECT parameter, gets, getmisses
FROM v$rowcache;
```

Redo log buffer

The Oracle server maintains online redo log files to minimize the loss of data in the database. The redo log files record all changes made to data in the database buffer cache with some exceptions; for example, in the case of direct writes when sorting. Redo log files are used in a situation such as an instance failure to recover committed data that has not been written to the datafiles.

The server process records changes made by an instance in the redo log buffer which is part of the SGA. It stores redo records which record changes. It is a circular buffer that is reused after it is filled up, but only after all the old redo entries are recorded in the redo log files.

The size of the redo log buffer is determined by the LOG_BUFFER parameter in the init.ora file. The log buffer size is more important for OLTP applications than for data warehouse/data mart applications. If experiencing many redo buffer allocation retries, increase the LOG_BUFFER parameter. Excess redo buffer allocation retries can be determined by querying the redo buffer allocation retries and redo entries values in the V\$SYSSTAT table. The redo buffer allocation entries value should be near 0 and less than 1% of the redo entries:

```
SELECT name, value
FROM v$sysstat
WHERE name IN ('redo buffer allocation retries', 'redo entries');
```

Data warehouse/data mart applications often use SQL*Loader to insert data into the database. Reduce the redo operations by using SQL*Loader direct path loading without archiving or use SQL*Loader direct path loading with archiving and using NOLOGGING mode. SQL*Loader direct path loading requires running SQL*Loader on the NetWare server, and the files containing the data should be loaded on a NetWare volume because direct path loading cannot be done across the network.

Sorts

The amount of memory required for sorts increases with parallel query because each query server needs `SORT_AREA_SIZE` amount of memory. The `SORT_AREA_RETAINED_SIZE` is the size to which Oracle reduces the sort area if its data is not expected to be referenced soon. For example, a query may do a JOIN and needs to sort both parts of the join individually. A smaller retained sort area size reduces memory usage, but causes additional I/O to write and read data to and from temporary segments on disk.

For OLTP applications, the `SORT_AREA_SIZE` should equal the `SORT_AREA_RETAINED_SIZE`. Data warehouse/data mart applications may benefit from a smaller sort area retained size. Large queries may require sorts bigger than the amount of memory available. If the sort space required is greater than the `SORT_AREA_SIZE`, the data is split into smaller pieces, called sort runs; each piece is sorted individually. The server process writes pieces to temporary segments on disk. The sorted pieces are merged to produce the final result. A larger `SORT_AREA_SIZE` will cause fewer sort runs. The ratio of disk sorts to memory sorts should be less than 5% for OLTP but can be much higher for data warehouse/data marts. The ratio can be determined from the `V$SYSSTAT` table by querying the name column for sorts (memory) and sorts (disk):

```
SELECT name, value
FROM v$sysstat
WHERE name IN ('sorts (memory)', 'sorts (disk)');
```

For parallel query, Oracle uses a producer/consumer model when executing its queries. Parent operations can begin consuming rows as soon as the child operations have produced rows allowing the parent and child operations to run simultaneously. Therefore, as many as two query slaves may be working for each degree of parallelism during a parallel query. One query may require multiple sorts. For example, a sort-merge join of two tables may be followed by a sort for an ORDER BY clause. This gives three sorts all together

CPU

Oracle is SMP-aware and fully takes advantage of multiple CPU systems. Oracle8™ v.8.0.4 has been enhanced to provide better performance on systems with multiple CPUs.

Parallelism

Upgrading to Oracle8 v.8.0.4 can increase performance; however, to fully take advantage of Symetric MultiProcessing (SMP), multiple threads need to be running at the same time. A single thread can only run on one CPU at a time. Parallel query makes effective use of SMP because it parallelizes a query using several query slaves. Other operations which will have significant performance boosts on multiple CPU systems are parallel index creation, several SQL*Loaders running in parallel, parallel inserts and any operation which requires a full table scan.

Both OLTP and data warehouse/data mart applications should consider parallel index creation. Parallel index creation will allow an index to be created with query slaves running in parallel based on the degree of parallelism of the index being created. For more information on parallel index creation see [Oracle8 Server SQL Reference](#).

Parallel query can greatly affect the performance of a database. This is especially true in data warehouse/data mart environments. For example, reducing the degree of parallelism of an index for a query changed the execution time of the query from 76 seconds to 709 seconds. OLTP applications that occasionally do large queries, such as end of month reporting, should take advantage of parallel query. Parallel query is available with the enterprise versions of Oracle7™ and Oracle8.

If the database is not tuned properly, there may be times when a task could be parallelized, but is not. There are several things which determine the number of query slaves running in parallel. The `PARALLEL_MAX_SERVERS` and `PARALLEL_MIN_SERVERS` parameters need to be set in the `init.ora` file. These two parameters determine the maximum and minimum number of parallel query slaves to be used for the instance.

If the `PARALLEL_MAX_SERVERS` is set too high, more memory may be required because each time a query is run, the optimizer may choose a different set of query slaves causing an additional amount of memory to be allocated. Setting the `PARALLEL_MAX_SERVERS` too low will negatively impact performance because there may not be enough parallel query slaves available to efficiently accomplish a task.

There may be some benefit to increasing the `PARALLEL_MIN_SERVERS` because the threads will be available when the query is run; they will not have to be created. Memory from the PGA is freed when the query servers are no longer needed and their threads exit. If `PARALLEL_MIN_SERVERS` is set too high, memory needed may not be freed because the query servers' threads never exit.

The degree of parallelism on tables and indexes is a key factor. It sets the maximum number of parallel query slaves which can be used for a table or index. On NetWare, the best performance is achieved by setting the degree of parallelism to two to four times the number of CPUs. The degree of parallelism helps the optimizer determine which index or table to use. Be careful not to set the degree of parallelism too high. More query slaves will take up more memory and other resources. Using the example above, increasing the degree of parallelism on a different index for the same query caused the execution time to change from 76 seconds to 249 seconds. OLTP environments can use hints within SQL statements to change the degree of parallelism on a particular table or index. If hints are not possible, for example if the SQL statements cannot be changed because they are within compiled code, the degree of parallelism can be changed dynamically on a per session basis by using the `ALTER TABLE <table_name> PARALLEL DEGREE(integer)` and `ALTER INDEX <index_name> PARALLEL DEGREE(integer)` clauses. For more information on setting the degree of parallelism, see [Oracle8 Server SQL Reference](#).

Oracle Optimizer

The optimizer determines the execution plans for SQL statements based on the values of init.ora parameters as well as the data collected from tables and indexes which have been analyzed. After editing the init.ora file, it is a good idea to regenerate the execution plans to see how they may have been affected. For example, the values set for SORT_AREA_SIZE, HASH_AREA_SIZE, and DB_FILE_MULTIBLOCK_READ_COUNT will influence the execution plan of an SQL statement. Index range scans cannot be parallelized; fast full index scans can. The DB_FILE_MULTIBLOCK_READ_COUNT determines the size of each I/O and is used for table scans and fast full index scans. Changing the DB_FILE_MULTIBLOCK_READ_COUNT may influence the optimizer to choose a table scan over an index range scan. See the section on I/O for more information on multiblock I/O.

The storage characteristics of tables, indexes and clusters can be analyzed to gather statistics which are then stored in the data dictionary. The optimizer uses these statistics in a cost-based approach to determine the most efficient execution plan for the SQL statements issued. It is a good idea to schedule a regular routine to analyze tables and indexes based on the amount of DML activity occurring against the database. Databases which require large amounts inserts, updates and deletes need to have analyzing done more frequently. Analyzing uses considerable amounts of resources, so it is best to schedule analyzing during off-peak hours. Analyzing is done using the ANALYZE TABLE <table_name> and the ANALYZE index <index name> commands. See [Oracle8 SQL Reference](#) for more information.

Each parallel query slave uses memory in the PGA while executing its part of a query or index creation. The amount of memory is determined by the SORT_AREA_SIZE and the HASH_AREA_SIZE init.ora parameters. Changing the values of these parameters may cause the optimizer to choose a different execution plan. For example, increasing the HASH_AREA_SIZE may cause the optimizer to choose a hash join rather than a merge sort.

Another init.ora parameter which affects parallelism is FAST_FULL_SCAN_ENABLED. Set this parameter to TRUE to take advantage of fast full index scans.

For more information on execution plans see [Oracle8 Server Concepts](#).

I/O

The performance of many software applications is limited by disk I/O. Often, CPU activity must be suspended while I/O activity completes. Such an application is said to be "I/O bound". Oracle is designed so that performance need not be limited by I/O.

Tuning I/O can help performance if a disk containing database files is operating at its capacity. However, tuning I/O cannot help performance in "CPU bound" cases, or cases in which the computer's CPUs are operating at their capacity.

Load balancing is an important factor in reducing I/O contention. Make sure the datafiles are evenly distributed across the drives. Other factors include striping (hardware, NetWare and Oracle striping), hardware issues, datafile layout, multiblock count, database block size, and file system choice (CLIB vs. DFS). Each of these factors is described below in more detail.

Load balancing

File I/O should be spread evenly across multiple disk drives. The redo logs should be on different disks than the database files. Tables should be located in different tablespaces than their indexes. Rollback segments should be in a separate tablespace and spread across disks. The temporary tablespaces should be on their own disks and be defined as TEMPORARY.

Stripe large tables and indexes across many disks. This is very important in data warehouse/data mart applications. The datafiles for tables and indexes tend to be large. Spreading the datafiles across disks will reduce I/O contention when using parallel query.

Oracle striping

"Striping" is the dividing of a large table's data into small portions and storing these portions in separate datafiles on separate disks. This permits multiple processes to access different portions of the table concurrently without disk contention. Striping is particularly helpful in optimizing random access to tables with many rows. Striping can either be done manually, described in this section, or through NetWare or hardware striping utilities.

Size table extents to force distribution of data evenly across datafiles. To do this, create tablespaces so that they are made up of multiple datafiles. We have found that greater numbers of datafiles lead the optimizer to parallelize operations and reduce I/O contention. Oracle striping can be done by either creating objects with MINEXTENTS set to a value greater than 1 (one) where each extent is at least one block smaller than the datafiles or by allocating extents to a file explicitly with the ALTER TABLE tablename ALLOCATE EXTENT clause.

Extents are allocated differently for partitioned indexes than for non-partitioned indexes. For partitioned indexes, there is an initial extent created for each partition. For non-partitioned indexes, the number of initial extents is equal to the degree of parallelism of the index. One can use this information to force distribution of extents across datafiles when creating indexes. This is especially useful for very large indexes. For a large index, create it in its own tablespace. For a non-partitioned index, create the tablespace with the number of datafiles equal to the degree of parallelism for the index. When creating the index, set the initial extent to slightly smaller than the size of each datafile or at least more than half the size of each datafile, depending on the amount of free space needed for future index growth. This will force the index to be spread across all of the disks. Partitioned indexes can be created in one tablespace or can have a separate tablespace for each partition. If you use one tablespace, create a separate datafile for each partition. Set the initial extent to more than half the size of each datafile, up to one block less than the size of the datafile, leaving room for future index growth.

Extent sizing is important for the temporary tablespace. The initial and next should be the same for the temporary tablespace and should be a multiple of the SORT_AREA_SIZE. For a large database, i.e. data warehouse/data mart, extents of 5MB with a SORT_AREA_SIZE of 1MB works well.

Hardware and NetWare striping

NetWare striping is done when NetWare partitions and volumes are created. Volumes can be made up of more than one segment and therefore more than one disk. Hardware striping is most commonly done using RAID technology. RAID, Redundant Array of Independent(or Inexpensive) Disks, is a technology that provides several ways of coordinating multiple disks so they behave as a single unit.

We have found the following will lead the optimizer to increase parallelism without increasing I/O contention for large queries. Use a large hardware stripe (RAID 1 or RAID 5) or a large NetWare stripe(RAID 0). Split up datafiles into more smaller files. Distribute the data across the datafiles by manipulating the extent size. The optimizer treats the number of datafiles as an indication of the number of physical drives.

We recommend using some type of hardware striping for large databases. The type of striping depends on the manufacturer. We have found for NetWare, hardware RAID 0+1, (RAID 1 in Compaq terminology) works best for sequential reads on Compaq systems. Hitachi has an advanced algorithm for RAID 5 and recommends RAID 5 for sequential reads. Hardware striping appears to be better than NetWare striping, except for RAID 0. Our testing showed that NetWare striping outperformed hardware striping on RAID 0. This is true for databases that execute large queries, such as data warehouse/data mart applications. These types of applications do a lot of table scans and fast full index scans. Both of

these operations do sequential reads. RAID 5 is a poorer choice for writes, both random and sequential. RAID 0+1 is a good choice for writes. RAID 0 will give the best performance, but has no protection from failures.

Stripe factor on controllers

Use a bigger stripe factor for the disk controllers when available. We have determined that a bigger stripe factor generally increases I/O performance. Disk drive speed can also affect performance. The 10,000 rpm drives will give as much as a 40% performance increase in throughput over 7,200 rpm drives for example, especially when the number of disk drives is increased.

Oracle block size

Applications that perform many sequential reads can reduce the number of reads by increasing the size of each read. We've found that the optimal DB_BLOCK_SIZE for data warehouse/data mart (DSS, batch) type operations is 8KB to 16KB, smaller block sizes are better for OLTP operations (2KB - 8KB). OLTP applications do more random reads and fewer sequential reads than DSS applications.

Multiblock I/O

The DB_FILE_MULTIBLOCK_READ_COUNT tells Oracle how many blocks to read for a single read request. DB_FILE_MULTIBLOCK_READ_COUNT should be set higher for data warehouse/data mart applications. We ran some sequential read tests on a Hitachi fiber storage system and determined that the optimal read size for sequential reads is between 32KB and 128KB depending on the hardware configuration. For single drives, the optimal read size is 32KB, for RAID 5, 128KB is best and for RAID 0+1, 64KB gives the best performance for sequential reads. The amount of data read is equal to the DB_FILE_MULTIBLOCK_READ_COUNT x the DB_BLOCK_SIZE. For example, if the DB_BLOCK_SIZE is 16KB, the DB_FILE_MULTIBLOCK_READ_COUNT should be between 2 and 8. If the DB_BLOCK_SIZE is 8KB, the DB_FILE_MULTIBLOCK_READ_COUNT should be between 4 and 16.

NetWare block size

The NetWare volumes containing Oracle datafiles should be created with the default 64KB block size. They should be created with file compression and block suballocation turned off. File compression causes extra overhead and can cause problems because Oracle datafiles tend to be large; when uncompressed (due to a database open) they may exceed the amount of physical disk space available. Block suballocation needs to be turned off when using the DFS file system. DFS gives approximately a 5% performance gain over CLIB.

Storage clause

Data warehouse/data mart applications usually do not require many updates, so the PCTFREE can be set to a small value. For OLTP applications, set this value as low as possible without causing row migration. For more information on chaining and row migration see [Oracle8 Concepts Guide](#), Chapter 2. Similarly, PCTUSED can be set high in data warehouse/data mart applications because these applications usually do not require many deletes. In general, set PCTUSED high enough to keep the blocks full. This will keep the blocks off of the free list and will reduce the number of I/Os for queries. The sum of PCTFREE and PCTUSED should not equal 100%. Doing so will cause blocks to continuously go off and on the free list. For more information on storage clauses see [Oracle8 Concepts Guide](#), Chapter 2, [Oracle8 Administrators Guide](#), Chapter 10 and the STORAGE CLAUSE section of [Oracle8 SQL Reference](#).

NETWARE SETTINGS

set enable file compression = off

Do not use file compression on the volumes where the Oracle datafiles reside. The datafiles are often large and if compressed they may be larger than the amount of available disk space when later decompressed.

Disable file compression enabled volumes containing Oracle datafiles by setting this parameter to off, if the server is dedicated to Oracle. Note: Setting this parameter to off will disable compression on all volumes.

set minimum service processes = <integer value>

Use MONITOR.NLM to determine if there is a need to adjust this setting. If monitor is displaying a value higher than the default, increase this parameter appropriately. The default is 100. *Set maximum service processes* may also need to be changed. The default is 500.

set minimum packet receive buffers = <integer value>

Bumping this parameter up will reserve more packet receive buffers for Oracle network communications. This is especially true for a lot of OLTP type transactions with many users. Determine if there is a need to adjust this setting by looking at the MONITOR.NLM screen. The default is 128. If the packet receive buffers are greater than the current setting, consider increasing the setting in the STARTUP.NCF file. *Set maximum packet receive buffers* may also need to be changed. The default is 500.

set maximum concurrent disk cache writes = <integer value>

This parameter could dramatically affect the performance of datafile creations and expansions along with setting *dirty disk cache delay time* (see next parameter). Before creating or expanding datafiles, increase this setting to the most the controller can handle. Most modern disk controllers can handle a value of 4000. Setting it too high can cause undesirable results if the disk controller cannot handle a high value. If the server is a dedicated to Oracle, it does not hurt to leave this parameter set high. If, however, the server is also being used for file and print, this parameter needs to come back down after creating or expanding the datafiles. The default for this parameter is 50.

set dirty disk cache delay time = <time value>

This parameter could dramatically affect the performance of datafile creations along with setting *maximum concurrent disk cache writes*. Set this value as low as possible (e.g. 0.1 seconds) before creating or expanding datafiles. The default for this parameter is 3.3 seconds (see previous parameter for more information).

set system threshold <integer value>

This parameter affects load balancing of CPUs. Adjusting this parameter may help performance, depending on the configuration; however, throughput measurements determine whether this would be helpful. We've seen that setting this value to 0 increased our performance, but each configuration should be tested before setting this value in a production environment. The default for this parameter is 1536. (Set parameter for NetWare 5)

set immediate purge of deleted files = on

Because Oracle datafiles tend to be large, set this when deleting and recreating datafiles.

set garbage collection interval = <time value>

Garbage collection is CPU intensive. This parameter changes the amount of time between garbage collections. Shortening or lengthening the interval may help, depending on the application. We have seen that setting this parameter to the minimum setting has improved performance for some operations.

set maximum file locks per connection = <integer value>

This parameter may need to be increased for a large database with many datafiles. The default is 250. The maximum is 1000. If an Oracle process needs more file handles than there are locks available, Oracle will hang or return errors. We have seen this happen while doing large loading jobs in parallel with SQL*Loader over the network.

set maximum file locks = <integer value>

This parameter may need to be increased for a large database with many files. The default is 10,000. The maximum is 100,000. If more files are opened by all Oracle processes combined than there are locks available, file opens will fail and Oracle will hang or return errors (e.g. *Unable to open CONFIG.ORA after 600 tries*).

CONFIG.ORA SETTINGS

The CONFIG.ORA file is found in the %ORACLE_HOME%\NLM directory.

NW_CFFSIZ=5000

When creating a datafile, Oracle will expand the datafile in chunks up to NW_CFFSIZ blocks. Setting this parameter high will help speed up database and datafile creation. The disk controller must be able to handle writes of this size when using this parameter or the server may become sluggish. The default for this parameter is 100.

NW_NUM_TO_SWITCH=5

This parameter determines how often Oracle yields control of the CPU. The higher the value, less thread switching will occur. The lower the value, the more often Oracle yields control. Setting this parameter too high can result in network timeouts or CPU errors. The default is 1.

TCP_KEEPAALIVE=5

This parameter determines the amount of time (in minutes) it takes TCP/IP to detect a dropped connection. The default is 120.

NW_FSTYPE=CLIB

This parameter is used to specify which file system interface Oracle should use. The default is CLIB. It uses the NetWare cache. We have found a 5% decrease in performance using this file system over DFS. This file system must be used when block suballocation is enabled on volumes containing Oracle datafiles.

NW_FSTYPE=DFS

This parameter is used to specify DFS (Direct File System). We have found that DFS increases performance by approximately 5%.

Oracle datafiles must reside on volumes which have block suballocation disabled when using a file system type of DFS. There have been cases where Oracle's datafiles or log files have been corrupted when NetWare's block suballocation feature is enabled.

A workaround is to run Oracle using the CLIB file system interface. Set the parameter NW_FSTYPE=CLIB in CONFIG.ORA.

While this issue is being addressed by Novell, Oracle datafiles should be placed on NetWare volumes that have block suballocation turned off. Since Oracle datafiles are typically large the space savings that block suballocation achieves are not significant.

QUICK CHECK LIST

Database Size

Small: 100 - 200MB or about 10 users

Medium: up to 40GB or 50-100 users

Large: over 40GB or over 100 users

Memory Structure Sizing

SHARED_POOL_SIZE=bytes

Small: 5 - 20MB

Medium: 10-50MB

Large: 50-120MB

LOG_BUFFER=bytes

Small: 16-32KB

Medium: 32-128KB

Large: 128-512KB

DB_BLOCK_BUFFERS=integer

Small: 100

Medium: 550

Large: 3200

Online Redo Logfiles

Small: 500KB-2MB

Medium: 1MB - 10MB

Large: 5MB - 20MB

Very large and heavily updated: 50MB or more

Overall Database Design

Tablespaces (minimize disk contention and free space fragmentation in the datafiles)

Minimum 6

1 System

1 Temp

1 Rollback

1 Data (each application)

1 Indexes (each application)

1 Tools/Users

- Nothing in the System tablespace not owned by Sys or System
- One segment type per tablespace
- Make sure all users are sorting to Temp(not System)
- Make sure all users (except Sys) have a default tablespace that is not System
- Designate the Temp tablespace as being of type TEMPORARY.
- Initial and Next extent on Temp tablespace is a multiple of SORT_AREA_SIZE.

Memory

Shared Pool

Library Cache

Gethitratio > 90% (OLTP only) If not, increase the SHARED_POOL_SIZE parameter in the init.ora file

Dictionary Cache

Hits/Misses < 15% on sum of all parameters

Database Buffer Cache

Cache Hit Ratio > 90% (OLTP only) increase DB_BLOCK_BUFFERS if low

Redo Log Buffer

Redo Log Space Requests X 5000/Redo Entries < 1,(OLTP only) if not increase the size of LOG_BUFFER

Rollback Segments

1 rollback segment for every 4 "TRANSACTIONS"

1 rollback segment for every "batch job"

Waits/Gets from v\$rollstat < 5% if not add rollback segments

Storage

- PCTFREE as low as possible without causing row migration
- PCTUSED high enough to keep blocks full (about 3 deletes returns block to free list)
- No free list contention
- DB_FILE_MULTIBLOCK_READ_COUNT set to compute to 32KB to 128KB
- Extents a multiple of DB_FILE_MULTIBLOCK_READ_COUNT
- Analyze schema regularly if utilizing the cost based optimizer
- Avoid reaching MAX extents
- Avoid errors while allocating extents due to insufficient contiguous space in tablespace.
- Avoid large number of small extents (not a multiple of DB_FILE_MULTIBLOCK_READ_COUNT)
- Consider striping tables when using parallel query

Sorting

- Eliminate sorts where possible
- Sort in memory as often as possible for the application (bump up SORT_AREA_SIZE)
- For OLTP, set SORT_AREA_SIZE=SORT_AREA_RETAINED_SIZE
- For DSS, set SORT_AREA_RETAINED_SIZE = 0 (minimizes impact on shared pool)
- Use sort direct writes to minimize the impact of sorting on the buffer cache.

Consider bumping up SORT_AREA_SIZE dynamically when running large report jobs or creating indexes (Oracle8).

CONCLUSION

Oracle and NetWare provide a high performance solution to most database needs including OLTP and data warehouse/data mart applications. Oracle8, at installation, provides default configurations for small, medium and large configurations. These configurations provide much of the tuning necessary to run a robust database. In addition, SQL Expert, the Diagnostics Pack and the Tuning Pack in Oracle Enterprise Manager are effective tools to simplify and aid in tuning. Oracle will run well, in most cases, after the initial installation; however, each application and hardware configuration is unique. Memory, CPU and I/O are all important and need to be tuned properly; therefore, tuning is a necessary and ongoing process to provide the best possible environment.



Copyright © 1999. Oracle Corporation
All Rights Reserved
Printed in U.S.A.

Oracle Corporation
World Headquarters
500 Oracle parkway
Redwood Shores, CA 94065
USA

Worldwide Inquiries:
650.506.7000
Fax: 650.506.7200
<http://www.oracle.com>
<http://platforms.oracle.com/novell>

Oracle, SQL, and SQL*Loader are registered trademarks and Enabling the Information Age, Oracle7, Oracle8, PL/SQL are trademarks of Oracle Corporation.

All other product and company names are used for identification purposes only and may be trademarks of their respective company.