

# **Tuning Oracle on Windows for Maximum Performance on PowerEdge Servers**

a White Paper sponsored by Dell, Oracle and Microsoft

By Edward Whalen, Performance Tuning Corporation, December 2004

## **Executive Summary**

Microsoft® Windows® is an excellent platform for running the Oracle® Database server. The Windows platform provides an easy to use graphical environment for managing your Oracle Database server. In addition, the Windows platform provides support for not only the Oracle Database server, but other Oracle add-ons and products as well, such as:

- Oracle Real Application Clusters (RAC)
- Oracle Fail Safe (using Microsoft Cluster Services)
- Oracle Application Server (AS)
- Oracle E-Business Suite
- Both 32-bit and 64-bit versions

In general the Windows platform is tuned in much the same manner as any other Oracle Database server; however, there are some key differences. These differences are mainly in the use of memory (32-bit Oracle on Windows) and in the process model. Since Oracle on Windows uses a threaded model, there are some additional concerns that you will not have on most UNIX platforms. As long as you are aware of these issues and have tuned appropriately, the Windows platform is a high-performing and stable platform for your Oracle Database server.

This paper covers some of the general Oracle tuning practices as well as specific tuning for the Windows platform.

# Table of Contents

Title: Tuning Oracle on Windows for Maximum Performance on PowerEdge Servers ....	1
Executive Summary .....	1
Table of Contents .....	2
Introduction.....	4
Introduction to Tuning .....	4
Tuning methodology.....	4
Create a Performance Plan.....	5
Windows Tuning on Dell PowerEdge Servers .....	5
Oracle Architecture on Windows.....	6
Tuning Oracle Memory.....	7
Tuning the Dell Storage Subsystem.....	9
Perfmon.....	10
Windows Configuration for Oracle Performance .....	12
What should not be run on the database server.....	13
What should be run on the database server.....	13
Oracle Tuning .....	14
Using Statspack.....	14
SGA sizing.....	16
PGA sizing.....	16
New features of Oracle10g that affect tuning .....	17
Automatic Workload Repository (AWR) .....	17
Formatting Trace Files Using trcesess.....	19
Tuning for backups .....	20
Database Backups .....	20
Using Snapshots for Backups .....	20
Miscellaneous Oracle Configuration Topics .....	20
Oracle Services .....	21
Error and Log file locations .....	21
Oracle connectivity.....	21
Summary .....	22

Appendix A. Editing the Windows Registry .....	24
Appendix B. Analyzing Statspack Output.....	25
Statspack Overview .....	25
Summary Information.....	25
RAC statistics (if applicable).....	27
Wait events.....	30
Top resource consumers .....	32
Instance activity .....	32
I/O Statistics.....	32
Cache statistics.....	32
Initialization parameters.....	34
Statspack Summary.....	34

## **Introduction**

Tuning is a very important component of the overall health and maintenance of your Oracle Database server. However, what to tune, when to tune, and if you even really need to tune are questions that are sometimes not easily answered. The tendency is to throw hardware at a performance problem. This usually works, but not forever. Eventually you cannot purchase a system large enough, or enough disk drives to satisfy your performance requirements. This is not to say that many systems do not have performance problems because they are undersized. Often performance problems are caused by too little memory or I/O capacity. Choosing the correctly sized hardware is only part of an overall performance plan.

Tuning the Oracle Database server involves several components that depend on each other, but are tuned separately. These components are the application, the hardware and the Oracle instance as you will learn in this paper.

## **Introduction to Tuning**

Tuning is the art and science of modifying and reconfiguring your system in order to achieve better performance. Tuning and sizing are closely related, in that tuning hardware might require the addition of more hardware. Tuning is done on a live system that is in use or in test, whereas sizing is a theoretical exercise that is done without actually modifying the system.

Tuning is not an exact science. It requires deductive reasoning, patience and sometimes intuition. This is why some people describe tuning as an art, as well as a science. In addition, tuning requires patience and dedication. When tuning you must also be willing to document your work as well as just making ad hoc changes. Performance tuning requires a methodology that you should stick to.

### ***Tuning methodology***

A tuning methodology is a discipline, a process and procedures that you consistently follow in order to tune your system. By following a tuning methodology you will be able to achieve better, more consistent and repeatable results. An example of a tuning methodology is as follows:

1. Perform initial assessment
2. Monitor the system
3. Analyze results
4. Create a hypothesis
5. Propose a solution
6. Implement the solution
7. Test the solution
8. Go to step 2

Your tuning methodology does not need to be the same as this one; however, it will probably be similar. It is important that your methodology is written down and followed on a regular basis.

The tuning methodology covers the mechanical aspects of tuning a system. In addition to the methodology, an overall performance plan should be developed, utilized and evaluated.

### ***Create a Performance Plan***

Once it has been determined that a performance problem exists and a tuning exercise is being started you should put together a performance plan. The performance plan is used to set goals for the desired performance levels. The performance plan should be evaluated regularly and the current state of the system compared against those goals.

Without a performance plan a tuning engagement could go on forever. It is almost always possible to achieve better performance, but at a cost of time and money. At some point you must determine that you have either achieved your goals, or exhausted all possibilities in trying to achieve that goal. Without this type of a plan your job will never be over.

Part of your tuning goals includes the desired performance. The desired improvement in performance will vary based on what your goals are. In some cases your desired performance is a reduction in response time. In other cases your desired performance is an increase in throughput. In other cases it might involve other improvements.

### **Tuning for response time**

Tuning for response time involves modifying the system components in order to reduce the latencies on transactions and making transactions go faster. Typically tuning for response time is done by tuning the application, installing faster hardware or tuning the Oracle instance. Response time can also be improved by tuning the Oracle Database schema with index tuning, partitioning, etc...

### **Tuning for throughput**

Tuning for throughput involves modifying the system to allow more work to be done without necessarily making any individual process or query run faster. Throughput increases might allow multiple queries to get more work done without making any of them run faster. For example, if a query takes 10 seconds to run and by tuning the system you can now have two queries run in 10 seconds you have increase the throughput of the system without improving the response time of the individual query.

This paper will describe methods and procedures to tune both for throughput and for response time. It includes features that can be enabled as well as configuration changes that can improve performance.

## **Windows Tuning on Dell PowerEdge Servers**

The main focus of this paper is to explore methods of improving the performance of the Oracle® Database server running on Microsoft® Windows® on Dell™ PowerEdge™ servers. The paper will look at various topics such as configuring Oracle on Windows for

the use of large memory, monitoring the OS and Oracle and tuning the I/O subsystem. In addition, some Oracle instance tuning will be covered as well.

It is difficult to tune the Oracle Database server on Windows without understanding how Oracle works on Windows. Those of you who have worked with the Oracle Database server on other platforms will find that memory and processing is handled slightly differently on Windows.

### ***Oracle Architecture on Windows***

The Oracle architecture on the Microsoft Windows platform is different from other platforms that Oracle supports. There are several differences, but our focus will be on just a few main differences. These differences are:

- Oracle on Windows runs as a service on the Windows platform.
- All of the Oracle server and background processes run as threads under the oracle.exe process.
- Oracle on 32-bit Windows uses PAE memory for memory above 4 GB.

There are also some other minor differences, however, this paper will focus on the key differences.

### **The Oracle Services**

The Windows architecture does not support the concept of background processes in the same way that UNIX operating systems do. Processes that have been started as a particular user will be terminated whenever the user logs out of Windows. This would not allow the database administrator to start the Oracle instance and then disconnect from Windows without the instance being terminated.

In order to overcome this issue, each Oracle instance must be started from a service. This service is called the bootstrap instance. The bootstrap instance is a small program that runs as a windows service and is used to control the real Oracle instance. Since the actual Oracle instance is started from a service, the instance is not affected by the database administrator logging out.

In addition to the bootstrap instance, there are several other services that Oracle on Windows utilizes. These include the Oracle listeners, support processes and with Oracle10g, the Cluster Ready Services.

### **Oracle Threads**

Oracle in a Windows environment runs as one process. All Oracle background processes and server processes are actually Windows threads. The server processes and background processes are all running oracle.exe. Within the Windows architecture all threads within a process share the same virtual memory space, therefore all of the background processes and server processes are limited to 4 GB of virtual memory address space. However, this virtual memory is split into 2 GB for user memory and 2 GB for kernel memory. With the use of the /3GB flag in the boot.ini file the split is modified to 3 GB for user memory and 1 GB for kernel memory. This topic is covered in more detail later in this paper.

This can sometimes limit the number of connections if some server processes take up too much memory. Oracle9i introduced the parameter `PGA_AGGREGATE_TARGET`. This parameter sets a target memory allocation for the collection of all server processes and should be set in order to limit the amount of PGA memory being used.

### **PAE Memory (32-bit architectures)**

The 32-bit architecture allows for 4 GB of physical memory to be addressed. In order to overcome this limitation, Intel came up with a method of addressing more than 4 GB of RAM. This is done using the Physical Address Extension (PAE). With PAE, the page directories and page tables are extended from 4 byte to 8 byte formats, thus allowing for the base address of page tables to be increased from 20 bits to 24 bits. The net affect is that with PAE up to 64GB of RAM can be addressed.

### **Memory (64-bit architectures)**

With 64-bit systems the limitations of virtual and physical addressing has been increased to Terabytes, thus are not an issue (yet). With 64-bit systems enormous SGAs can be created and the amount of memory used by the Oracle process is not an issue.

### ***Tuning Oracle Memory***

Oracle memory is divided into the SGA (System Global Area) memory and user memory (for each server thread). The server PGA (Program Global Area) memory is the memory used by the individual connections that are using the Oracle database server. The SGA is the shared memory that contains the Oracle data buffers, the shared pool and the log buffer. The SGA is fairly static in size and is set with the Oracle initialization parameters.

The Program Global Area is memory that is used by each individual Oracle server process. In addition, the size of each individual PGA is dynamic and variable based on how much data is being processed. Maximums for sort areas and hash areas can be set, but since the user's memory is used for processing data, the more data that is accessed, the larger the user's PGA memory can get.

The amount of memory used for sorting can be set with the initialization parameter `SORT_AREA_SIZE`, and the amount of memory used for hash joins can be set with the initialization parameter `HASH_AREA_SIZE`. Since this is not all of what the PGA memory is used for, the PGA can get quite large. I have seen PGA memory exceed 500 MB of RAM. Since all of the server processes combined and the SGA has to share a 2 GB or 3 GB address space, this can become an issue.

There are several things that can be done to increase the amount of memory that is used by Oracle as well as increasing the amount of memory that it can access. In the next few sections, these topics will be addressed.

### **Using the 3 GB Flag**

During the later years of Windows NT 4.0, Microsoft offered a method to modify the user/kernel split from a 2 GB / 2 GB split to a 3 GB / 1 GB split. This allows for more memory to be used by the Oracle process. The `/3GB` flag has been available since

Windows NT 4.0 SP3 and is available in Windows 2000 and Windows Server 2003 as well.

The /3GB flag is recommended for use in systems with up to 4 GB of RAM. In systems with 16GB or more of physical RAM, the use of /3GB and /PAE are mutually exclusive. When running on such a system, the /3GB flag should be used to optimize Oracle for large or many PGAs, while the /PAE flag should be used when many database buffers are required for optimal performance.

Note: The /3GB flag is used to control virtual memory address space, the /PAE flag is used to control physical memory.

### **Using the /PAE flag and Upper Memory**

In order to use memory over 4 GB the /PAE flag must be set and Oracle must be tuned accordingly. As mentioned above, for using memory over 4 GB and less than 16 GB, the /3GB flag can be set as well. Oracle uses memory above 4 GB by taking advantage of the AWE (Address Windowing Extension) feature built into Windows 2000, Windows 2003 and Windows XP.

In order to use this memory, the Oracle administrator account (either LocalSystem or the domain user that installed Oracle) must have the “Lock memory pages” privilege set. In addition the AWE\_WINDOW\_MEMORY parameter must also be created and set. This parameter specifies how much of the 3 GB virtual memory space allocated to Oracle should be used for mapping database buffers. The default value of 1GB should be sufficient and increasing this value could be detrimental, since you only have 3 GB of virtual memory to use. The higher the value of AWE\_WINDOW\_MEMORY that you have set, the more likely it is that you will experience user memory issues. Start with the default value of 1 GB and tune upwards only if you need to.

Once these steps have been completed, larger memory can be set by configuring DB\_BLOCK\_SIZE and DB\_BLOCK\_BUFFERS initialization parameters. When using Very Large Memory (VLM) you should use DB\_BLOCK\_BUFFERS rather than DB\_CACHE\_SIZE in order to configure memory.

Note: The previous sections do not apply if you are running Windows on a 64-bit platform in 64-bit mode.

### **Configuring PGA\_AGGREGATE\_TARGET**

Introduced in Oracle9i the Oracle initialization parameter PGA\_AGGREGATE\_TARGET allows you to tell Oracle how much memory that you would like the sum of all server process memory to be. Most people are skeptical that the PGA\_AGGREGATE\_TARGET actually works, but it does. By setting the PGA aggregate target to limit the cumulative memory that is used by the user processes, out of memory conditions can be avoided.

There is always a delicate balance between allocating memory to the SGA or to the user processes. This is something that must be tuned and adjusted as necessary based on your configuration and your user community. The PGA\_AGGREGATE\_TARGET is set to the amount of KB, MB, or GB of memory to be allocated to the PGAs.

## ***Tuning the Dell Storage Subsystem***

The I/O performance of your system is critical to the overall performance of the Oracle Database server. Whether you are running on a PERC array controller or on a Dell/ EMC SAN, many of the performance issues are the same. I/O performance is usually influenced by capacity, but there are some configuration parameters that can be tuned.

The following is a brief overview of I/O and RAID performance issues. Sizing Oracle on Dell servers is covered in a separate paper.

### **I/O Performance Review**

I/O performance depends on several factors. A single disk drive is a physical entity and can only perform a finite number of I/O Operations per Second (IOPS). Different RAID levels have different performance characteristics. The following is a set of suggestions that will improve I/O performance.

- A typical 15K RPM disk drive can service approximately 125 IOPS at reasonable disk latencies. Running disk drives at a higher rate will cause latencies to increase to levels that affect Oracle performance.
- RAID 5 has good read performance but each write to a RAID 5 volume causes 4 physical I/Os. Even though you will not see them in perfmon, these I/Os are occurring and can cause the physical disk drives to be overdriven.
- RAID 1 has good read performance and write performance, but keep in mind that each logical write incurs two physical writes.
- In a shared I/O environment such as a SAN or shared SCSI system, I/Os from another system could be affecting your performance.

The next sections provide information on how to specifically tune Dell storage subsystems for optimal performance.

### **Tuning the PERC Array Controller**

The PowerEdge RAID Controller is a high performance SCSI array that can handle very large I/O loads. However, a poorly configured PERC controller will perform poorly. The following are tips on how to optimally configure your PERC controller for Oracle performance:

- Use a sufficient number of disk drives for your I/O load. As mentioned above, a single disk drive should not be run at more than 125 IOPS. If more disk performance capacity is needed, add more drives, preferably in a RAID array.
- Configure the RAID stripe size at 64 KB. Internal performance tests have shown that a 64 KB stripe size is optimal for Oracle performance.
- Monitor the I/O subsystem. This is covered in the next section.

### **Tuning the Dell/EMC Array**

The Dell/EMC SAN is a high performance Storage Area Network subsystem that can also handle very large I/O loads and is extremely expandable. However, as with the

PERC, a poorly configured SAN will perform poorly. The following are tips on how to optimally configure your SAN for Oracle performance:

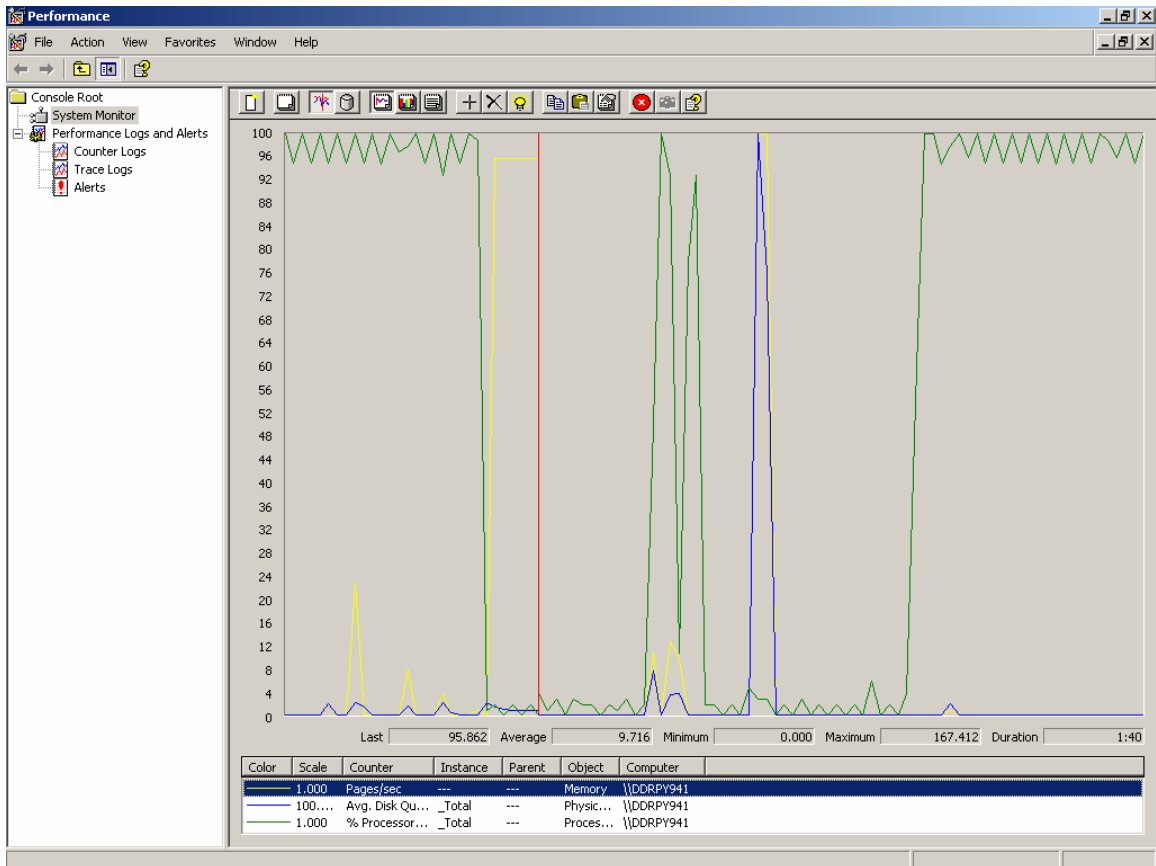
- Use a sufficient number of disk drives for your I/O load. As mentioned above, a single disk drive should not be run at more than 125 IOPS. If more disk performance capacity is needed, add more drives, preferably in a RAID array.
- Configure the RAID stripe size at 64 KB. Internal performance tests have shown that a 64 KB stripe size is optimal for Oracle performance.
- Set the SAN cache page size to 8KB or larger. Most Oracle systems use an 8 KB block size. By setting the SAN cache page size to a minimum of the Oracle block size, the SAN will not have to allocate more than one page per Oracle I/O.
- Set the read and write cache appropriately. Oracle user processes are more sensitive to read performance than to write performance. The Oracle redo log is more sensitive to write performance than read performance.
- Monitor the I/O subsystem from both the SAN and OS perspective. Since a SAN can service many systems, make sure that the storage processors are not overloaded. This can be done via the Navisphere Analyzer. Monitoring the I/O subsystem from an OS perspective is covered in the next section.

As mentioned before, the performance of your Oracle Database server relies on good I/O performance. If the I/O subsystem is performing poorly, the overall performance of the database server will suffer.

### ***Perfmon***

The Windows performance monitor (AKA perfmon) is an excellent tool for monitoring your Oracle Database server. Perfmon is a graphical tool that can display performance data both in chart or table format. It is recommended that both CPU and I/O activity be monitored.

CPU utilization is a very good metric for monitoring your system because it has a physical limit of 100%. The closer you are to 100%, the closer you are to the maximum capacity of the system. The following example shows the CPU utilization of an Oracle Database server. If your system is constantly above 60%, you might be experiencing performance problems.

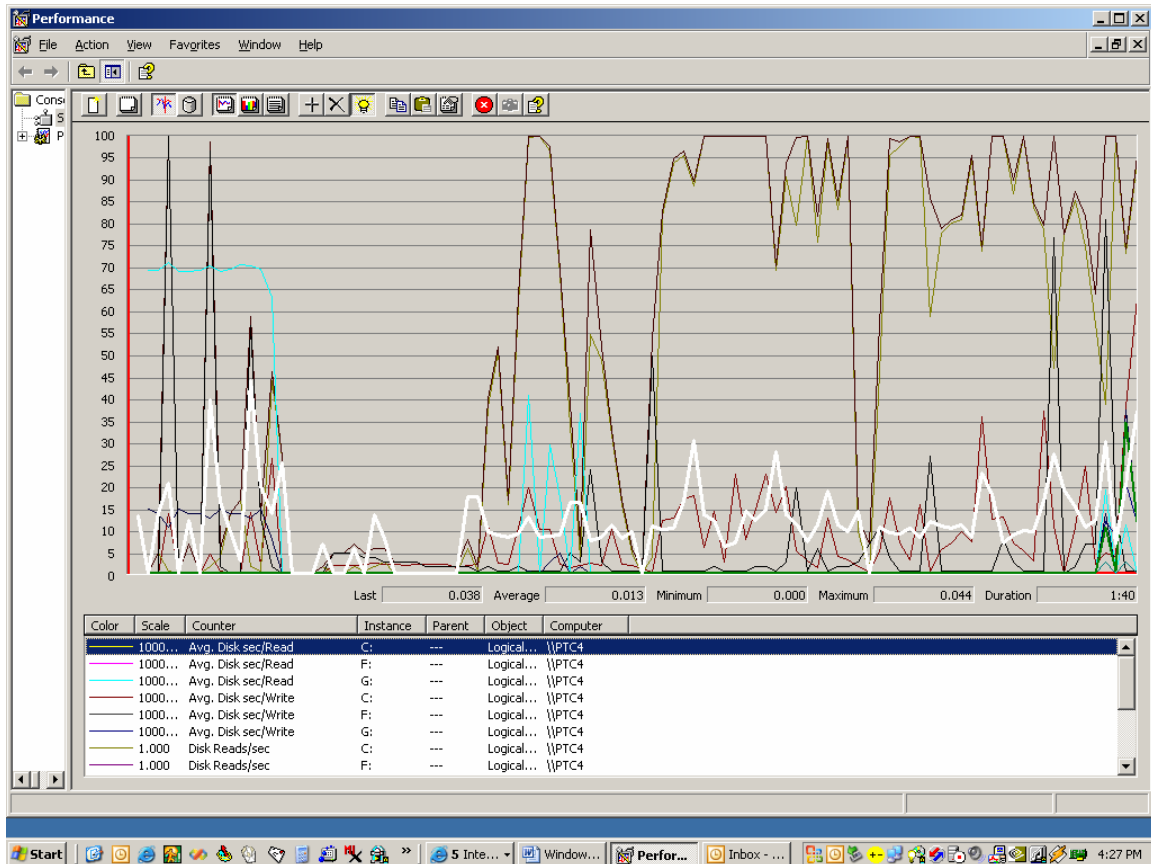


**Figure 1. CPU Monitoring with Perfmon**

I/O performance is critical to the performance of your database server. Therefore I/O performance should be monitored. Of particular interest is the LogicalDisk performance object. It is recommended that the following counters be monitored:

- Disk Reads/sec
- Disk Writes/sec
- Avg. Disk sec/Read
- Avg. Disk sec/Write
- Avg. Read Disk Queue Length
- Avg. Write Disk Queue Length
- Disk Read Bytes/sec
- Disk Write Bytes/sec

An example of I/O monitoring with perfmon is shown here:



**Figure 2: I/O Monitoring with Perfmon**

By monitoring the I/O subsystem you can tell if you are experiencing I/O performance problems, and perhaps understand what is causing those problems.

Note: On average, Disk sec/Read and Disk sec/Write (read and write latencies) should be in the 5 – 15 ms range. Sometimes spikes occur, but if you are experiencing very high spikes or long periods of time when disk latencies are high, then you are experiencing an I/O problem and it should be addressed immediately.

Monitoring the system with perfmon is an important part of an overall performance plan. In addition to perfmon, there are many fine third party tools for monitoring your system.

## Windows Configuration for Oracle Performance

Just as it is important to properly size and configure the hardware and Oracle Database server, it is also important to properly configure the Windows Operating System. Sometimes this involves modifying parameters, other times this involves enabling services and other times it involves disabling services. When tuning the Windows system what is not running is as important as what is running. The next sections describe what should and should not be running on the Oracle Database server.

### ***What should not be run on the database server***

To maintain optimal uptime, there are some common sense items that should never be run on a database server; these include:

- DNS server–If for any reason the server needs to be stopped for maintenance, domain naming will no longer work
- Domain Controller–For the same reasons as the DNS, except with greater negative enterprise consequences. The database server should not act as a primary or secondary domain controller
- Router–The database server should not act as a network router
- File and/or print server–The database server should not act as a file and/or print server since these functions can consume CPU and memory resources as well as network bandwidth
- Terminal Services/Citrix Services–These types of services require a large amount of memory, depending on user load
- Disable the following services–These services are unnecessary for the database server and utilize valuable system resources:
  - License Logging Service
  - Plug and Play
  - Remote Access Autodial Manager
  - Remote Access Connection Manager
  - Remote Access Server
  - Telephony Service
- Unused Network Protocols–You should remove all unused network protocols, thus only keeping the protocol that you are using with Oracle

### ***What should be run on the database server***

The following should be run on the database server:

- Any monitoring software that captures performance information. Oracle provides use of perfmon and Enterprise Manager
- Security and auditing software at the OS level
- The following services should not be disabled on the database server and are necessary for normal operations:
  - Alerter
  - Computer Browser
  - EventLog
  - Messenger
  - OracleServiceSID

- OracleHOME\_NAMETNSListener
- Remote Procedure Call (RPC) Service
- Server
- Spooler
- TCP/IP NetBIOS Helper
- Workstation

Note: Disabling any of these services could cause significant problems with the system.

Note: *SID* represents the Oracle SID. *HOME\_NAME* represents the ORACLE\_HOME name.

Depending on the version of Windows that you are running, this list might vary.

## **Oracle Tuning**

Tuning Oracle can be broken down into three major components. These areas are tuning the application (SQL Statements), tuning the Oracle instance and tuning the hardware. Tuning the hardware has already been covered in this paper. What remains is tuning the application and tuning the Oracle instance. Both tuning the Oracle application and tuning the Oracle instance can be aided via the use of statspack. Statspack is a tool that comes with the Oracle Database server that can be used in order to help you to identify long running or high resource using SQL statements, determine how well the instance is running and even how well the I/O subsystem is running.

### ***Using Statspack***

Statspack does not take very many system resources and is fairly easy to use; however, its scope is fairly limited. Statspack can be used to help you identify problem SQL statements and to identify issues with the Oracle instances, but it is not an alerting tool, nor is it an analysis tool. You have to do the analyzing yourself. Statspack is a statistics gathering and reporting tool. In the next sections you will see how to run statspack, and a little bit on how to interpret statspack data.

### **Running Statspack**

Statspack is an Oracle package that is provided as part of the Oracle Database server. There are several scripts that are used in order to create the statspack package and to report the statspack output. These scripts are %ORACLE\_HOME%\rdbms\admin\spcreate.sql and %ORACLE\_HOME%\rdbms\admin\spreport.sql. The spcreate.sql script is used to create the statspack package, and the spreport.sql script is used to create a statspack report.

In order to create the statspack package run the following SQL as the database administrator:

```
sqlplus “/ as sysdba” @?\rdbms\admin\spcreate
```

This procedure will prompt you for the password for the perfstat user as well as the default tablespace and temporary tablespace for that user. Once the statspack packages have been created a snapshot of the Oracle dynamic performance views can be taken by running the Oracle command:

```
SQL> exec statspack.snap
```

Once you have created several snapshots, a statspack report can be created by running the SQL script spreport. The spreport script will create a statspack report which can then be analyzed in order to determine the performance of the Oracle instance during the measurement period.

### **Analyzing Statspack Output**

Since statspack reports work on measurement periods, you can choose to analyze an entire day, or a specific time when an event, such as a specific batch job is running. Statspack allows for a great deal of flexibility in terms of measurement intervals. Statspack output is generated by running the spreport.sql SQL script. This SQL script generates the statspack output file which can then be interpreted.

The following statement can be used to invoke the statspack report utility:

```
sqlplus “/ as sysdba” @?\rdbms\admin\spreport
```

Once you have invoked spreport you will be prompted for beginning and ending snapshots as well as the output report name. Once you have generated this output file you are ready to analyze the results.

### **Analyzing Statspack Output**

Statspack output is written in the form of a text file. This file is named during the creation of the statspack report with the default being *sp\_start\_finish* where *start* and *finish* are the snapshot IDs. Statspack is divided into several sections including:

- Summary information
- RAC statistics (if applicable)
- Wait events
- Top resource consumers
- Instance activity
- Cache statistics
- Rollback and Undo statistics
- SGA and Shared Pool statistics
- Initialization parameters

It can be very time consuming to analyze statspack results because of the amount of data that is presented. A typical statspack output is over 20 pages in length. An overview of how to interpret statspack output is presented in Appendix B.

## ***SGA sizing***

When sizing the system there are two main components; the SGA memory and the PGA memory. Both of these must be tuned in order to maintain a well running system.

### **What is the SGA**

When describing an instance, one is describing the memory structure and background processes confined to a system memory structure defined a SID. Of these memory structures, the most important to tuning is the System Global Area or (SGA). The SGA contains the following structures that handle various parts of transaction processes.

- The database buffer cache–Stores hot datablocks in the cache, thus reducing I/O
- The redo log buffer–Buffers changed items before they are written to ONLINE\_REDO\_LOGS
- The Shared Pool–This contains shared memory objects, and is required to process unique SQL statements
- The Java Pool–Caches java memory objects, not used as much as J2EE standards do not allow for database installed JRM
- The Streams Pool–(10 g)
- The Data Dictionary Cache–When parsing statements, the data dictionary cache contains reference material concerning tables, structures, and users

### **What influences the size of the SGA**

The initial SGA size is determined by the values of the following initialization parameters:

- DB\_BLOCK\_SIZE Sets the size of the Oracle blocks.
- DB\_CACHE\_SIZE Sets the size of the Oracle buffer cache.
- SHARED\_POOL\_SIZE Sets the size of the Oracle Shared Pool

## ***PGA sizing***

In addition to the SGA, the PGA or Program Global Area is a large consumer of resources in the system. Because of the windows threading model, all of the PGA memory for all of the Oracle server processes must fit into the 3GB virtual memory space.

### **What is the Program Global Area**

The Program Global Area is an area of memory that holds program and control information for Oracle server processes. Each dedicated server process has its own PGA memory allocated to it. The PGA memory is allocated each time a server process is created. The sum of all individual PGAs attached to an Oracle instance is referred to as the Aggregate PGA memory.

The PGA memory is used for processes in order to retrieve and process data during sorts, joins, etc... Prior to Oracle9i the PGA memory was tuned individually via the Oracle initialization parameters SORT\_AREA\_SIZE and HASH\_AREA\_SIZE. With Oracle9i the Aggregate PGA memory can be tuned with the Oracle initialization parameter PGA\_AGGREGATE\_TARGET. This value allows Oracle to control the entire Aggregate PGA memory rather than the individual PGAs.

The PGA\_AGGREGATE\_TARGET parameter is especially important to Oracle running in a Microsoft Windows environment because of the constraint on virtual memory addressing. With Oracle on Windows, always set the PGA\_AGGREGATE\_TARGET value so that the sum of the PGA and SGA memory is less than 3GB.

## **New features of Oracle10g that affect tuning**

Most of the features mentioned in this paper are valid for both Oracle9i and Oracle10g, however there are a few new features introduced in Oracle10g that can benefit performance on the Windows platform. These features are the Automatic Workload Repository

### ***Automatic Workload Repository (AWR)***

Of all of the new performance related features that are included with Oracle 10g, the most encompassing is the Automatic Workload Repository (AWR). This feature is the next logical evolution of the statspack package. Database performance statistics are generated and stored in respective performance tables at regular intervals. The default setup has statistics gathered every hour, and stored for a week. From this datastore, metrics and performance related deltas can be calculated. It is also from this datastore that many of the self tuning features can derive predictive and diagnostic information, namely the Automatic Database Diagnostic Monitor (ADDM).

As with the predecessor Statspack, AWR stores system statistics, poorly performing SQL statements, and wait events. However, there are additional “sets” of statistics gathered from the AWR. Here are some examples:

- Time Model Statistics- Holistic set of statistics that are normalized based on time to reference to all performance statistics group. With this, a top level statistic can be taken that describes that total workload of a database since instance creation. This statistic is described as DB time and is found, amongst other time based statistics, in the V\$SESS\_TIME\_MODEL and V\$SYS\_TIME\_MODEL views. DB time is a cumulative calculation derived from non-idle db sessions. As a useful point of reference, a tuning goal can be to establish a downward trend on DB time.
- Active Session History (ASH)- As part of the AWR, samples of the Active Session History dynamic performance view V\$ACTIVE\_SESSION\_HISTORY are flushed to disk. This in combination with only active session being polled allows a clean look at how resources are being used, as they are being used. This data does not have any skew that accounts for idle sessions and waiting events.

The recommended interface with AWR is the Oracle Enterprise Manager. There are, of course, APIs to interface with ASF. To initiate snapshots, the CREATE\_SNAPSHOT package in the DBMS\_WORKLOAD\_REPOSITORY is run.

```
BEGIN
    DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT ();
END;
/
```

Snapshot records are stored in the DBA\_HIST\_SNAPSHOT view.

Functional analysis reports of the AWR data can be done by running one of two SQL scripts.

1. awrrpt.sql –similar to the STATSPACK sprpt.sql that reports performance data based on range of snapshot ID's
2. awrrpti.sql-same as awrrpt.sql, but instance specific

In addition, the STATISTICS\_LEVEL initialization parameter must be set to TYPICAL or ALL for AWR to fully function.

### **End to End Application Tracing**

While the AWR provided the method to gather statistics snapshots and compare items against time, End to End Application Tracing provides the means to capture and time specific events related to an application process. In addition the entire span of a specific request is traced from the client, through the middle tier to the database and then back. This is accomplished through the use of client identifiers. With the client and application being identified, a more focused attempt at heavy use SQL identification and modification can be performed. In addition to being able to uniquely identify a client, process tracing can be focused on:

Service - related applications with similar performance requirements

Module - defines a specific functional application, such as defined in 11i Applications, e.g. GL, AP, AR

Action - Specific DML run in a module

Once the desired tracing information has been collected, it can be processed for reporting purposes with the utility trcsess and formatted with TKPROF.

As with AWR, Oracle's intended interface with End to End Application Tracing is the Enterprise Manager Database Control. But as with AWR, there is a command line interface to enable the various tracing options. This is performed with the procedures contained in the DBMS\_MONITOR package.

To enable tracing information for a specific client, the CLIENT\_IDENTIFIER needs to be selected for the desired session from the V\$SESSION dynamic performance view. If the CLIENT\_IDENTIFIER turns out to be JDOE.JDOE for the desired session, then the command to enable tracing would be:

```
EXECUTE DBMS_MONITOR.CLIENT_ID_STAT_ENABLE(client_id =>
'JDOE.JDOE')
```

To disable tracing, the following would be run:

```
EXECUTE DBMS_MONITOR.CLIENT_ID_STAT_DISABLE(client_id =>
'JDOE.JDOE')
```

To further refine the scope of the trace, the `SERV_MOD_ACT_STAT_ENABLE` procedure can be run. This particular module can simultaneously gather the statistics of the service, module, and action. An example:

```
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(service_name =>
'Finance', module_name => 'AR', action_name => 'UPDATE')
```

Once the trace data has been collected, there is a table and several dynamic performance views that can be queried to analyze the statistics, depending on the collection method. These include:

## Tables

`DBA_ENABLED_AGGREGATIONS` - This table can be queried for aggregate information on currently running statistics from tables.

## Dynamic Performance Views

`V$CLIENT_STATS`- statistics gathered from some specified client identifier

`V$SERVICE_STATS`- statistics gathered from some specific service trace session

`V$SERV_MOD_ACT_STAT_ENABLE`- statistics gathered from `service_name/module_name/action_name` trace session enabled with the `DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE` procedure.

`V$SVCMETRIC`- elapsed time for both database calls and CPU can be queried from this view

## *Formatting Trace Files Using trcsess*

Once the trace files have been generated, there is the potential for a tremendous amount of information to wade through. Oracle provides a utility to filter out extraneous information. The scope of the formatted can be based upon:

- Module Name
- DML Action
- Service
- Client ID
- Session ID

The syntax for using trcess follows:

```
trcess [output=output_file_name]  
      [session=session_id]  
      [clientid=client_id]  
      [service=service_name]  
      [action=action_name]  
      [module=module_name]  
      [trace_files]
```

At least one of the identifying parameters (session, clientid, service, module, or action) must be declared. Once the trace files have been filtered, the information can then be formatted to a more readable format using the age-old Oracle utility TKPROF.

## **Tuning for backups**

Because of the importance of Backup and Restore operations it is necessary to spend time sizing and tuning the backup solution. Typical Backup schedules include a mixture of both offline (cold) backups and online backups. Since the improvements in RMAN (the Oracle Recovery Manger) backups in Oracle9i we always recommend the use of RMAN for hot backups.

RMAN includes the ability to tune the number of backup streams as well as the backup paths that are used to backup the Oracle Database. Because of the number of different backup options, only general guidelines are presented here.

### ***Database Backups***

Full database backups should be performed as often as feasible based on your needs. The database should always be running in archive log mode and archive log files should be backed up as soon as possible from the time of their creation. If possible, leave the most recent backups on a local disk in order to be quickly accessed in the event of a system failure. Do not backup to the same disks as the database, redo log files or archive log files.

### ***Using Snapshots for Backups***

With the Dell/EMC Fibre Channel RAID Arrays, SnapView™ can be used in order to facilitate the process of cold database backups. SnapView is not used in conjunction with RMAN, but rather as a supplement. SnapView can be used to take a snapshot of a shutdown Oracle Database in order to minimize downtime. Shut down the Oracle Database (all instances of a RAC cluster), perform the snapshot, start up the Oracle Database and then you can backup the snapshot image of the database from another server. This method allows for cold backups with very little downtime.

## **Miscellaneous Oracle Configuration Topics**

As mentioned earlier in this paper, with a few exceptions Oracle on Windows is very much like Oracle on any other platform; however, these exceptions are very important.

In this section some miscellaneous Oracle topics will be covered that are specific to the Windows platform.

### ***Oracle Services***

The following services are installed and configured as part of the Oracle installation:

Service	Description
OracleCSService	The Oracle Cluster Synchronization Service (10g only). This service is used for ASM communication.
OracleDBConsoleSID	The Oracle Enterprise Manager Console (10g only).
OracleJobSchedulerSID	Used to manage scheduled tasks.
OracleOraDB10g_home1iSQL*Plus	Used to manage iSQL*Plus.
OracleOraDB10g_home1SNMPPeerEncapsulator	SNMP service.
OracleOraDB10g_home1SNMPPeerMasterAgent	SNMP service.
OracleOraDB10g_home1TNSListener	The Oracle listener.
OracleServiceSID	The bootstrap instance.

Note: *SID* represents the Oracle SID. *ORADB10g\_home1* represents the ORACLE\_HOME name.

These services should not be modified manually.

### ***Error and Log file locations***

From time to time it becomes necessary to view the error logs. Many Oracle and generic service error messages are found in the standard Windows Error Log. In addition, an Oracle alert log is used in Oracle on Windows just as in any other Oracle platform. With Windows, this error log is found in %ORACLE\_BASE%\admin\SID\bdump; where %ORACLE\_BASE% is the base install directory and might look something like D:\oracle\product\10.1.0 and *SID* is the Oracle instance SID.

The Oracle error log can be very useful for tracking down Oracle problems as well as some system problems. Before calling technical support, please check the Oracle alert log.

### ***Oracle connectivity***

There are two ways to connect into the Oracle instance; the dedicated server process and the shared server process (MTS).

#### **Dedicated Server**

With the dedicated server process, each connection into the Oracle instance acquires one thread. As mentioned earlier, all of the Oracle threads share the same 3GB virtual memory limit. Although the dedicated server model requires more resources per user, it

also provides a higher degree of performance. It is recommended that dedicated server connections be used for user counts of less than 500. If the number of user connected into the Oracle instance exceeds 500, shared server processes should be used.

### **Shared Server (MTS) configuration**

Using the MultiThreaded Server or MTS connection, many users can connect into the Oracle instance and share the same server processes. These are known as shared server processes. Connections are made into the dispatcher which places the requests into a queue in the shared pool (or large pool). These requests are serviced by the shared server processes. When using MTS, both the number of dispatchers and the number of shared server processes should be tuned. Since there is additional overhead involved in using shared server processes it is recommended that dedicated server processes be used until such a time that the system cannot handle the large number of threads. This number will vary, but is usually around 500 users.

### **Summary**

As you have learned in this paper, tuning Oracle on Microsoft Windows is not significantly different from tuning the Oracle Database server on other platforms. Oracle tuning is broken down into three different types of tuning; application tuning, Oracle instance tuning and hardware tuning. Application tuning was briefly mentioned in this paper which focused on instance tuning and hardware tuning (sizing). The key differences between Oracle on Windows and Oracle on other platforms were covered in this paper. By being sensitive to these differences, you can more effectively tune the Oracle Database server on Windows for optimal performance.

THIS WHITE PAPER IS FOR INFORMATIONAL PURPOSES ONLY, AND MAY CONTAIN TYPOGRAPHICAL ERRORS AND TECHNICAL INACCURACIES. THE CONTENT IS PROVIDED AS IS, WITHOUT EXPRESS OR IMPLIED WARRANTIES OF ANY KIND.

© Copyright 2004 Dell Inc. All rights reserved. Reproduction in any manner whatsoever without the express written permission of Dell Inc. is strictly forbidden. For more information, contact Dell. December 2004

Dell is a trademark of Dell Inc. Other trademarks and trade names may be used in this document to refer to either the entities claiming the marks or the names of their products. Dell disclaims proprietary interest in the marks and names of others.

Copyright © 2004, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

© 2004 Microsoft Corporation. All rights reserved.

Microsoft, Microsoft Windows Server, and Microsoft Windows Storage Server 2003 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

## **Appendix A.**

### **Editing the Windows Registry**

Regedit is the executable used to edit the windows registry. The windows registry contains system level configuration parameters, of which some are of considerable tuning value. The registry is set into five general categories.

HKEY\_CLASSES\_ROOT = correlates application execution upon application selection in Windows Explorer

HKEY\_CURRENT\_USER=contains profile parameters for respective user.

HKEY\_LOCAL\_MACHINE=contains global parameters

HKEY\_USERS=contains root user profile information

HKEY\_CURRENT\_CONFIG=system startup parameters concerning hardware

The listed categories contain these datatypes

REG\_BINARY=binary application displayed in hexadecimal format

REG\_DWORD=generally used to store service and device information in 4 byte characters

REG\_EXPAND\_SZ=variable length strings that initialize upon application data access

REG\_MULTI\_SZ=readable parameters set up in multiple strings

REG\_SZ=static text strings

REG\_FULL\_RESOURCE\_DESCRIPTOR=nested arrays creating a resource list for driver or physical component.

A point to consider before doing any configuration to the registry, is to perform an export of the current settings. This is accomplished by selecting

File > Export

This will create a \*.reg file for you to save and restore the original settings if necessary.

# Appendix B.

## Analyzing Statspack Output

### Statspack Overview

Statspack output is written in the form of a text file. This file is named during the creation of the statspack report with the default being `sp_start_finish` where *start* and *finish* are the snapshot IDs. The entire statspack output is over 20 pages in length, so only pieces will be covered here:

Statspack is divided into several sections including:

- Summary information
- RAC statistics (if applicable)
- Wait events
- Top resource consumers
- Instance activity
- Cache statistics
- Rollback and Undo statistics
- SGA and Shared Pool statistics
- Initialization parameters

As you can see, there are quite a few statistics gathered by statspack. This paper can not cover all of these topics, but a few of the key statistics will be covered here.

### *Summary Information*

Since statspack is run as snapshots, it is important to know what time interval and when the snapshots were taken. This allows you to customize statspack to retrieve the information that you desire.

STATSPACK report for

DB Name	DB Id	Instance	Inst Num	Release	Cluster	Host
TEST	4041012933	TEST4		4 9.2.0.4.0	YES	RAC-NODE-4

	Snap Id	Snap Time	Sessions	Curs/Sess	Comment
Begin Snap:	31	18-Aug-04 09:44:52	16	11.0	
End Snap:	61	18-Aug-04 16:34:40	17	9.9	
Elapsed:		409.80 (mins)			

Cache Sizes (end)

~~~~~

Buffer Cache: 592M Std Block Size: 8K  
Shared Pool Size: 208M Log Buffer: 64K

Load Profile

~~~~~

	Per Second	Per Transaction
	-----	-----
Redo size:	39.23	28,367.29
Logical reads:	2.68	1,936.74
Block changes:	0.11	77.97
Physical reads:	0.25	182.09
Physical writes:	0.14	100.97
User calls:	0.22	156.09
Parses:	0.07	49.47
Hard parses:	0.00	3.56
Sorts:	0.03	18.47
Logons:	0.00	0.12
Executes:	0.10	74.85
Transactions:	0.00	

% Blocks changed per Read: 4.03 Recursive Call %: 70.50  
Rollback per transaction %: 0.00 Rows per Sort: 497.36

Instance Efficiency Percentages (Target 100%)

~~~~~

Buffer Nowait %: 100.00 Redo NoWait %: 100.00  
Buffer Hit %: 95.21 In-memory Sort %: 97.61  
Library Hit %: 92.99 Soft Parse %: 92.81  
Execute to Parse %: 33.91 Latch Hit %: 99.99  
Parse CPU to Parse Elapsd %: 59.65 % Non-Parse CPU: 90.56

Shared Pool Statistics Begin End

-----  
Memory Usage %: 25.93 26.73  
% SQL with executions>1: 64.01 70.27  
% Memory for SQL w/exec>1: 54.80 68.60

Top 5 Timed Events

~~~~~

Event	Waits	Time (s)	Ela Time	% Total
-------	-------	----------	----------	---------

-----

control file parallel write	8,065	29	25.33
ges enter server mode	7	12	10.47
wait for master scn	16,398	12	10.41
control file sequential read	33,737	11	9.38
db file sequential read	3,110	10	8.50

-----

It is important to note the following information:

- Run time of the statspack report; almost 7 hours.
- Cache sizes.
- Load profile; how much activity was going on during this run.
- Cache hit ratios. This is your first view of how well the SGA is tuned for your application. You would like to get your cache hit ratios as close as you can to 100%.
- Top wait events. One of the best features of Oracle statistics are the wait events. These events tell you what Oracle is waiting on in order to give you an idea of where to tune.

### ***RAC statistics (if applicable)***

If you are running in a RAC (Real Application Clusters) environment you will see RAC statistics following the summary information. These statistics will give you an idea of how the RAC cluster itself is doing. Some of the most important statistics that you should analyze are:

Cluster Statistics for DB: TEST Instance: TEST4 Snaps: 31 -61

#### Global Cache Service - Workload Characteristics

-----

Ave global cache get time (ms):	0.2
Ave global cache convert time (ms):	0.3
.	.
.	.
.	.
Global cache hit ratio:	6.1
Ratio of current block defers:	0.0
% of messages sent for buffer gets:	9.1
% of remote buffer gets:	0.8
Ratio of I/O for coherence:	0.5
Ratio of local vs remote work:	6.6
Ratio of fusion vs physical writes:	0.0

#### Global Enqueue Service Statistics

-----

Ave global lock get time (ms): 1.2  
 Ave global lock convert time (ms): 0.3  
 Ratio of global lock gets vs global lock releases: 2.0

GCS and GES Messaging statistics

-----  
 Ave message sent queue time (ms): 0.2  
 Ave message sent queue time on kxsp (ms): 0.7  
 Ave message received queue time (ms): 551.2  
 Ave GCS message process time (ms): 0.1  
 Ave GES message process time (ms): 0.1  
 % of direct sent messages: 87.3  
 % of indirect sent messages: 11.9  
 % of flow controlled messages: 0.8  
 -----

GES Statistics for DB: TEST Instance: TEST4 Snaps: 31 -61

Statistic	Total	per Second	per Trans
-----			
dynamically allocated gcs resourc	0	0.0	0.0
dynamically allocated gcs shadows	0	0.0	0.0
flow control messages received	12	0.0	0.4
flow control messages sent	12	0.0	0.4
gcs ast xid	0	0.0	0.0
gcs blocked converts	186	0.0	5.5
gcs blocked cr converts	252	0.0	7.4
gcs compatible bast	2	0.0	0.1
gcs compatible cr bast (global)	51	0.0	1.5
gcs compatible cr bast (local)	1,413	0.1	41.6
gcs cr bast to PIs	0	0.0	0.0
gcs cr serve without current lock	0	0.0	0.0
gcs error msgs	0	0.0	0.0
gcs flush pi msgs	172	0.0	5.1
gcs forward cr to pinged instance	0	0.0	0.0
gcs immediate (compatible) conver	40	0.0	1.2
gcs immediate (null) converts	48	0.0	1.4
gcs immediate cr (compatible) con	76	0.0	2.2
gcs immediate cr (null) converts	797	0.0	23.4
gcs msgs process time(ms)	479	0.0	14.1
gcs msgs received	4,952	0.2	145.6
gcs out-of-order msgs	0	0.0	0.0
gcs pings refused	0	0.0	0.0
gcs queued converts	0	0.0	0.0

gcs recovery claim msgs	2	0.0	0.1
gcs refuse xid	0	0.0	0.0
gcs retry convert request	0	0.0	0.0
gcs side channel msgs actual	138	0.0	4.1
gcs side channel msgs logical	3,181	0.1	93.6
gcs write notification msgs	2	0.0	0.1
gcs write request msgs	76	0.0	2.2
gcs writes refused	0	0.0	0.0
ges msgs process time(ms)	1,150	0.0	33.8
ges msgs received	18,357	0.7	539.9
global posts dropped	0	0.0	0.0
global posts queue time	0	0.0	0.0
global posts queued	0	0.0	0.0
global posts requested	0	0.0	0.0
global posts sent	0	0.0	0.0
implicit batch messages received	254	0.0	7.5
implicit batch messages sent	52	0.0	1.5
lmd msg send time(ms)	147	0.0	4.3
lms(s) msg send time(ms)	6	0.0	0.2
messages flow controlled	215	0.0	6.3
messages received actual	20,758	0.8	610.5
messages received logical	23,197	0.9	682.3
messages sent directly	22,892	0.9	673.3
messages sent indirectly	3,116	0.1	91.6
msgs causing lmd to send msgs	4,515	0.2	132.8
msgs causing lms(s) to send msgs	412	0.0	12.1
msgs received queue time (ms)	12,847,041	522.5	377,854.1
msgs received queued	23,309	0.9	685.6
msgs sent queue time (ms)	673	0.0	19.8
msgs sent queue time on kxsp (ms)	18,908	0.8	556.1
msgs sent queued	3,169	0.1	93.2
msgs sent queued on kxsp	26,119	1.1	768.2

GES Statistics for DB: TEST Instance: TEST4 Snaps: 31 -61

Statistic	Total	per Second	per Trans
process batch messages received	273	0.0	8.0
process batch messages sent	25	0.0	0.7

In addition, if you are experiencing interconnect performance problems you might also see the following wait statistics show up on the top wait events:

- global cache cr request
- buffer busy global CR
- buffer busy waits

If you see these waits in any significant numbers, further analysis is required.

### ***Wait events***

Wait events are describes as statistics incurred by a process or thread having to wait on a specific event to continue. The wait events are often the consultant's first stop when looking for a performance bottleneck. The wait events specify what resources the Oracle instance and processes are waiting on most. They are sorted by the cumulative time spent waiting on a particular resource as shown here:

Wait Events for DB: TEST Instance: TEST4 Snaps: 31 -61

- > s - second
- > cs - centisecond - 100th of a second
- > ms - millisecond - 1000th of a second
- > us - microsecond - 1000000th of a second
- > ordered by wait time desc, waits desc (idle events last)

Event	Avg				
	Total Waits	Wait Timeouts	wait Time (s)	Waits (ms)	/txn
control file parallel write	8,065	0	29	4	237.2
ges enter server mode	7	6	12	1723	0.2
wait for master scn	16,398	19	12	1	482.3
control file sequential read	33,737	0	11	0	992.3
db file sequential read	3,110	0	10	3	91.5
ges global resource director	53	2	7	129	1.6
wait for membership change	160	160	5	31	4.7
wait for record update	146	141	4	30	4.3

```

gcs remote message      530,340  525,612  70,844  134 #####
ges remote message      546,793  528,170  35,422  65 #####
SQL*Net message from client    3,828    0  12,092  3159  112.6
PX Idle Wait            207    206    515  2488    6.1

```

Wait Events for DB: TEST Instance: TEST4 Snaps: 31 -61

-> s - second

-> cs - centisecond - 100th of a second

-> ms - millisecond - 1000th of a second

-> us - microsecond - 1000000th of a second

-> ordered by wait time desc, waits desc (idle events last)

These wait events include idle wait events as show in the lower part of the output. Waits such as gcs remote message, ges remote message and SQL\*Net message from client are wait events that indicate “waiting” on another system and can safely be ignored.

In addition, the background wait events are also provided within statspack as shown here

Background Wait Events for DB: TEST Instance: TEST4 Snaps: 31 -61

-> ordered by wait time desc, waits desc (idle events last)

Event	Avg				
	Total Waits	wait Timeouts	wait Time (s)	wait (ms)	Waits /txn
control file parallel write	8,065	0	29	4	237.2
ges enter server mode	7	6	12	1723	0.2
wait for master scn	16,398	19	12	1	482.3
control file sequential read	33,673	0	11	0	990.4
ges global resource director	53	2	7	129	1.6

```

global cache open x      4    1    0    0    0.1
global cache open s      1    0    0    0    0.0
rdbms ipc message       65,349  61,468  195,042  2985  1,922.0
gcs remote message      530,351  525,613  70,844  134 #####
ges remote message      546,772  528,170  35,422  65 #####
pmon timer              8,227  8,227  35,421  4305  242.0
smon timer              81    79  34,437 #####  2.4

```

As with the other wait events, the idle events are listed at the end of the report and can be safely ignored.

## ***Top resource consumers***

The top resource consumers are listed by Gets, by Reads and by Executions. This allows you to easily find the top SQL statements that are using the most resources in your system. By finding the top resource consumers, you can more easily focus your tuning activity on a certain set of processes that use the most resources in the system.

## ***Instance activity***

The instance activity section of the statspack report provides information on what had happened in the instance during the time period that was measured. This information consists of counters of various activities, such as sorts, checkpoints, executes, etc.. This information can be very useful in determining instance efficiency.

## ***I/O Statistics***

Both tablespace and data file I/O statistics provide information on the efficiency and the performance of the I/O subsystem. Information is provided on the number of reads and writes, the average read and write size and the average time per read and write. This might be your first indication of an imbalance in the I/O subsystem that is causing performance problems. Average I/O times in excess of 20ms are an indication of a problem. Because Oracle statspack provides this information both on a tablespace and data file level, pinpointing the problem is made easier.

## ***Cache statistics***

The cache statistics provide information on how well the SGA is performing. Included in the cache statistics are estimates of how well the caches would continue to perform in the event of an increase or decrease in cache size. This includes information such as the buffer pool statistics as shown here:

Buffer Pool Statistics for DB: TEST Instance: TEST4 Snaps: 31 -61

-> Standard block size Pools D: default, K: keep, R: recycle

-> Default Pools for other block sizes: 2k, 4k, 8k, 16k, 32k

			Free	Write	Buffer			
	Number of Cache	Buffer	Physical	Physical	Buffer	Complete	Busy	
P	Buffers Hit %	Gets	Reads	Writes	Waits	Waits	Waits	
D	72,076 95.3	66,514	3,156	370	0	0	0	

In addition, the buffer pool advisory shows you how well the buffer pool would perform if you had additional memory allocated to it, as shown here:

Buffer Pool Advisory for DB: TEST Instance: TEST4 End Snap: 61

-> Only rows with estimated physical reads >0 are displayed

-> ordered by Block Size, Buffers For Estimate

P	Size Estimate (M)	Size Factor	Buffers for Estimate	Est Physical Read Factor	Estimated Physical Reads
D	56	.1	6,818	3.62	176,818
D	112	.2	13,636	3.60	175,553
D	168	.3	20,454	3.59	175,430
D	224	.4	27,272	3.59	175,352
D	280	.5	34,090	3.59	175,198
D	336	.6	40,908	3.59	175,106
D	392	.7	47,726	1.00	48,975
D	448	.8	54,544	1.00	48,805
D	504	.9	61,362	1.00	48,805
D	560	.9	68,180	1.00	48,805
D	592	1.0	72,076	1.00	48,805
D	616	1.0	74,998	1.00	48,805
D	672	1.1	81,816	1.00	48,805
D	728	1.2	88,634	1.00	48,805
D	784	1.3	95,452	1.00	48,805
D	840	1.4	102,270	1.00	48,805
D	896	1.5	109,088	1.00	48,805
D	952	1.6	115,906	1.00	48,805
D	1,008	1.7	122,724	1.00	48,805
D	1,064	1.8	129,542	1.00	48,805
D	1,120	1.9	136,360	1.00	48,805

Note that with the cache size currently set at 592 MB, the buffer cache is very efficient. By adding more memory to the buffer cache, the number of physical reads would not be reduced at all, thus any additional memory would be wasted.

In addition, there is a shared pool advisory that provides the same estimates for the shared pool as shown here:

Shared Pool Advisory for DB: TEST Instance: TEST4 End Snap: 61

-> Note there is often a 1:Many correlation between a single logical object in the Library Cache, and the physical number of memory objects associated with it. Therefore comparing the number of Lib Cache objects (e.g. in v\$librarycache), with the number of Lib Cache Memory Objects is invalid

Shared Pool	SP	Estd	Estd	Estd	Lib LC Time
Size for Estim (M)	Size Lib Cache (M)	Lib Cache Mem Obj	Cache Time Saved (s)	Cache Time Saved (s)	Estd Lib Cache Mem Obj Hits
112	.5	15	4,090	91	1.0 18,117
136	.7	15	4,090	91	1.0 18,117

160	.8	15	4,090	91	1.0	18,117
184	.9	15	4,090	91	1.0	18,117
208	1.0	15	4,090	91	1.0	18,117
232	1.1	15	4,090	91	1.0	18,117
256	1.2	15	4,090	91	1.0	18,117
280	1.3	15	4,090	91	1.0	18,117
304	1.5	15	4,090	91	1.0	18,117
328	1.6	15	4,090	91	1.0	18,117
352	1.7	15	4,090	91	1.0	18,117
376	1.8	15	4,090	91	1.0	18,117
400	1.9	15	4,090	91	1.0	18,117
424	2.0	15	4,090	91	1.0	18,117

As you can see from this example, both the buffer pool and the shared pool are oversized in this example. Both of them can be reduced, thus freeing up memory, without any adverse side affects.

### ***Initialization parameters***

One nice feature of statspack is that it automatically documents the configuration parameters as they were at the time of the report. Statspack displays the non-default Oracle initialization parameters at the end of the report.

### ***Statspack Summary***

As you can see, statspack is a very powerful tool, but at times somewhat difficult to interpret. It is through experience and trial and error that you will become proficient in the interpretation of statspack output.