Oracle8*i* on Windows NT/2000: Architecture, Scalability, and Tuning

April, 2000

Introduction

As the adoption of Windows continues to progress rapidly, Oracle8*i* and Oracle8*i* Release 2 have become the market leading database for the Windows NT/2000 platform. From the outset, Oracle's goal has always been to provide the highest performing and most tightly integrated database on Windows. Oracle invested early to move its market leading UNIX database technology to the Windows platform. In 1993, Oracle was the first company to provide a database for Windows NT.

Initially, Oracle's development efforts were concentrated on improving the performance and optimizing the database architecture on Windows. Oracle7 on Windows NT was re-designed to take advantage of several features unique to the Windows platform including native thread support and integration with some of the Windows NT administrative tools such as Performance Monitor and the Event Viewer.

However, Oracle on Windows has evolved from the basic level of operating system integration to utilize some of the more advanced services in the Windows operating system, including Very Large Memory (VLM) support and 64-bit Windows 2000 support. Oracle is continuing to innovate and leverage new Windows technologies. The following paragraphs discuss the Oracle8*i* database architecture, scalability topics, and tuning procedures.

Oracle8*i* Architecture on Windows

When running on Windows, Oracle8*i* contains the same features and functionality as it does on the various UNIX platforms that Oracle supports. However, the interfaces between Oracle8*i* and the operating system have been substantially modified to take advantage of the unique services provided by Windows. As a result, Oracle8*i* on Windows is not a straightforward port of the UNIX code base. Significant engineering work has been done to make sure that Oracle8*i* exploits Windows to the fullest and also to guarantee that Oracle8*i* is a stable, reliable, and high performing system upon which to build applications.

Thread Model

By far, the most significant architectural change in Oracle8*i* on Windows is the conversion from a process-based server to a thread-based server. On UNIX, Oracle uses processes to implement background tasks such as database writer (DBWR), log writer (LGWR), MTS dispatchers, MTS shared servers and the like. In addition, each dedicated connection made to the database causes another operating system process to be spawned on behalf of that session. On Windows, however, all of these processes are implemented as threads inside a single, large process. What this means is that for each Oracle database instance or SID, there is only one process running on Windows for the Oracle8*i* server itself. Inside that process will be many running threads with each thread corresponding directly to a process in the UNIX architecture. So, if there were 100 Oracle processes running on UNIX for a particular instance, that same workload would be handled by 100 threads in one process on Windows.

Operationally, client applications connecting to the database are unaffected by this change in database architecture. Every effort has been made to ensure that the database operates in the same way on Windows as it does on other platforms, even though the internal process architecture has been converted to a thread-based approach.

The original motivation to move to a thread architecture had to do with performance issues with the first release of Windows NT when dealing with files shared among processes. By simply converting to a thread architecture and modifying no other code, performance was dramatically increased as the particular operating system bottleneck was avoided. No doubt that the original motivation for the change is no longer present, however the thread architecture for Oracle remains since it has been proven to be a very stable, maintainable one. In addition, there are other benefits that arise out of the thread architecture. These include faster operating system context switches among threads (as opposed to processes); a much simpler SGA allocation routine which does not require the use of shared memory; faster spawning of new connections since threads are more quickly created than processes; decreased memory usage since threads share more data structures than processes do; and a perception that a thread-based model is somehow more "NT-like" than a process-based one.

There are a few negatives that come with being a thread-based architecture. One is that it is harder to identify and manipulate the threads that are running in the database. Unlike a process based model, where terminating a process or finding statistics about a process can typically be done with operating system tools, a thread based model makes it more difficult to perform these tasks.

Another downside is the fact that all memory allocated for the SGA, PGAs, or for other purposes comes from the same 3GB address space. When large sort areas are in use (for instance), available memory can be an issue since all foreground threads get their memory from the same 3GB pool. In a process model, each foreground process gets its own address space and therefore has more memory available for it to use. This downside will be short-lived, however, as the 64-bit version of Windows 2000 promises to provide at least 1 terabyte of address space per process, thereby removing any memory issues for the foreseeable future.

Internally, the code to implement the thread model is compact and very isolated from the main body of Oracle code. Fewer than 20 modules provide the entire infrastructure needed to implement the thread model. In addition, robustness has been added to the architecture through the use of exception handlers and also through routines used to track and de-allocate resources. Both of these additions help allow for 24x7 operation with no downtime due to resource leaks or an ill-behaved program.

Services

In addition to being thread-based, the Oracle8*i* database is also not a typical Windows process. It is a Windows *service*, which is basically a background process that's registered with the operating system, started by Windows at boot time, and which runs under a particular security context. The conversion of Oracle into a service was necessary to allow the database to come up automatically upon system reboot, since services require no user interaction to start. When the Oracle database service starts, there are no typical Oracle threads running in the process. Instead, the process basically waits for the first connection from Server Manager which will cause a foreground thread to start and which will eventually cause the creation of the background threads and of the SGA. When the database is shutdown, all the threads that were created will terminate, but the process itself will continue to run and will wait for the next connection request and startup command. In addition to the Oracle database service, a second service is created, which spawns Server Manager and opens up the database for client use. Finally, the Oracle Net8 Listener is a service since it too needs to be running before users can connect to the database. Again, all of this it basically an implementation detail that does not affect how clients connect to or otherwise use the database.

File I/O Enhancements

One other area in which much work has been done in the Oracle8*i* code concerns support for large files and for raw files. In an effort to guarantee that all features of Windows are fully exploited by Oracle8*i*, the database supports 64-bit file I/O to allow the use of files larger than 4GB in size. In addition, physical and logical raw files are supported as data, log, and control files in order to enable Oracle Parallel Server on Windows and also for those cases where performance needs to be maximized.

64-bit File I/O

Internally, all Oracle8*i* file I/O routines support 64-bit file offsets, meaning that there are no 2GB or 4GB file size limitations when it comes to data, log, or control files as is the case on some other platforms. In fact, the limitations that are in place are generic Oracle

limitations across all ports. These limits include 4 million database blocks per file, 16KB maximum block size, and 64K files per database. If these values are multiplied, the maximum file size for a database file on Windows is calculated to be 64GB while the maximum total database size supported (with 16KB database blocks) is 4 petabytes.

Raw File Support

Like UNIX, Windows supports the concept of raw files, which are basically unformatted disk partitions that can be used as one large file. Raw files have the benefit of no file system overhead, since they are unformatted partitions. As a result, using raw files for database or log files can have a slight performance gain. However, the downside to using raw files is manageability since standard Windows commands do not support manipulating or backing up raw files. As a result, raw files are generally used only by very high-end installations and by Oracle Parallel Server, where they are required.

To use a raw file, all Oracle needs to be told is the filename specifying which drive letter or partition to use for the file. For instance, the filename \\.\PhysicalDrive3 tells Oracle to use the 3rd physical drive as a physical raw file as part of the database. Likewise, \\.\G: tells Oracle to use the logical raw file that has been assigned drive letter G. In addition, a file such as \\.\log_file_1 is an example of a raw file that has been assigned an alias for ease of understanding. Aliases can be assigned with the SETLINKS utility provided with Oracle8*i*. When specifying raw filenames to Oracle, care must be taken to choose the right partition number or drive letter, as Oracle will simply overwrite anything on the drive specified when it adds the file to the database, even if it's already an NTFS or FAT formatted drive.

To Oracle, raw files are really no different from other Oracle database files. They are treated in the same way by Oracle and can be backed up and restored via Recovery Manager as any other file can be.

Oracle8*i* Scalability on Windows

One of the key goals of the Oracle8*i* product on Windows is to fully exploit any technologies that can help increase scalability, throughput, and database capacity. The following section describes a few of these technologies, how they affect Oracle, and the benefits that can be derived from them.

4GB RAM Tuning (4GT) Support

Windows NT Server v4.0 Enterprise Edition includes a feature called 4GB RAM Tuning (4GT). This feature allows memory-intensive applications running on Windows NT Server Enterprise Edition to access up to 3GB of memory as opposed to the standard 2GB in previous versions of the operating system. The obvious benefit to Oracle8*i* is that 50% more memory becomes available for database use, which can increase SGA sizes or connection counts. All Oracle database server releases since 7.3.4 have supported this feature with no modifications necessary to a standard Oracle installation. The only configuration change required is to ensure that the /3GB flag is used in Windows' boot.ini file. This feature is also supported in Windows 2000.

For more information on the 4GT feature, follow the following links:
http://www.microsoft.com/NTServer/ntserverenterprise/exec/feature/4GBT.asp
http://www.microsoft.com/ntserver/NTServerEnterprise/exec/overview/WindowsNTEnterpriseFAQ.asp#4gig

Very Large Memory (VLM) Support

One of the key Windows-specific additions introduced in Oracle8*i* was support for Very Large Memory (VLM) configurations. Specifically, Oracle8*i* breaks through the 3GB address space limit imposed by Windows and allows a single database instance access to up to 16GB of database buffers. By configuring a database with z large amount of buffers, disk I/O activity can be diminished since more data is cached in memory. This leads to a corresponding increase in throughput and performance. The follow sections will describe the different VLM implementations on Windows NT v4.0 and Windows 2000.

Windows NT V4.0 and ESMA

When running on Windows NT v4.0 Enterprise Edition, Oracle8*i* version 8.1.4 and higher have been enhanced to support Intel's Extended Server Memory Architecture (ESMA) via the use of Intel's PSE36 device driver. This device driver allows applications to access up to 16GB of RAM when running on Intel Xeon processors. Specifically, Oracle8*i* can now make calls to PSE36 in order to read from and write to memory not normally accessible to Windows' memory manager. Oracle8*i* uses these calls to allow the use of large numbers of database buffers. By merely increasing the value of the db_block_buffers initialization parameter and setting one other initialization parameter and a registry value, a database instance gets instant access to much more memory than was previously possible. No database file changes are required, nor are any database operations affected other than the increase in available database buffers.

For further information about this support and about the PSE36 device driver, follow the following links:

http://www.intel.com/procs/servers/Xeon/downloads/esma.pdf
http://developer.intel.com/vtune/pse36/index.htm

Windows 2000 and AWE

With Windows 2000, Microsoft has enabled an even faster implementation of VLM support than PSE36. Called the Address Windowing Extensions (AWE), this support is a set of API calls that allow applications to access more than the traditional 3GB of RAM normally accessible to Windows NT applications. As opposed to PSE36, which was a read/write interface to the extended memory, the AWE interface takes advantage of the Intel Xeon architecture and provides a fast map/unmap interface that avoids the expensive memory copying done by PSE36. It is expected that the AWE interface will prove to be a faster implementation of VLM support for Oracle than PSE36 was. The Oracle8*i* Release 2 (8.1.6) and higher supports the AWE interface.

In exactly the same way that PSE36 allows Oracle to access many database buffers, so too will AWE allow a large increase in database buffer usage up to 16GB of buffers total. Again, as with PSE36, this support is purely an in-memory change with no changes or modifications made to the database files.

On Windows 2000 Datacenter Server, Oracle supports up to 64GB of memory. However, Windows 2000 restricts the database address space to 2GB if you more than 16GB of RAM exists on the machine. When running Windows 2000 on a machine with more than 16GB of RAM installed, there is a conflict between the 4GT feature and the use of AWE calls. Windows 2000 on machines with 16GB or more of RAM only allows one of these features to be used at a time. If the 4GT feature is turned in via the /3GB flag in boot.ini, then Windows 2000 will ignore any memory over and above 16GB, making it not accessible to Oracle. If the /3GB flag is not used, then Windows 2000 can and will use all the memory in the machine, but Oracle will get a small address space of only 2GB.

For more information on AWE, see the following:
http://www.microsoft.com/Windows/server/News/fromMS/intelpae.asp
http://www.microsoft.com/HWDEV/NTDRIVERS/AWE.htm

Turning on VLM

Since the PSE36 and the AWE implementations are so similar, they will both be described here. To enable the use of many database buffers for Oracle, perform the following steps:

1. Windows NT v4.0 Enterprise Edition, Service Pack 3 or later (for PSE36) or Windows 2000 Datacenter Server (for AWE) must be installed.

2. For PSE36, the Intel PSE36 driver must be installed and operational.

3. For AWE, verify that the user account under which Oracle runs (typically the local SYSTEM account) has the "Lock memory pages" NT privilege.

4. use_indirect_data_buffers=true must be present in the init.ora for the database instance that will use VLM. If this parameter is not set, then Oracle8*i* behaves in exactly the same way as previous releases.

5. Set db_block_buffers and db_block_size as desired for the database. Note that the total number of bytes of database buffers (that is, db_block_buffers multiplied by db_block_size) is no longer limited to 3GB as was the case in previous releases.

6. The VLM_BUFFER_MEMORY (for PSE36) or AWE_WINDOW_MEMORY (for AWE) registry parameter must be created and set in the appropriate key for the Oracle home in the Windows NT registry. This parameter is specified in bytes and has a default value of 1GB. When using PSE36, this parameter tells Oracle8*i* how much normal, non-PSE36 memory to use for database buffers. When using AWE, this parameter tells Oracle8*i* how much of its 3GB address space to reserve for mapping in database buffers. For both implementations, the value set comes from Oracle8*i*'s 3GB virtual address space, so its value must be less than 3GB. Setting this parameter to a large value has the effect of using more of Oracle8*i*'s address space for buffers and using less PSE36 or AWE memory for buffers. However, since accessing PSE36 or AWE buffers is somewhat slower than accessing virtual address space buffers, tune this parameter to be as large as possible without adversely limiting database operations.

7. Once these parameters are set, the Oracle8*i* database can be started up and function exactly the same as before except that more database buffers are available to the instance. In addition, disk I/O may be reduced since more Oracle data blocks can be cached in the SGA. If out of memory errors occur during the startup sequence, verify the following:

a. For PSE36, the PSE36 driver is installed and functional

b. db_block_buffers is not set too high for the amount of memory in the machine. Note that more memory than just the database buffers themselves is required when starting up the database. For each database buffer, a database buffer header is also allocated from Oracle8*i*'s virtual address space. When allocating 2,000,000 database buffers, the memory for these buffer headers amounts to several hundred megabytes. This must be considered when setting db_block_buffers and VLM_BUFFER_MEMORY or AWE_WINDOW_MEMORY.

c. VLM_BUFFER_MEMORY or AWE_WINDOW_MEMORY is not set too high for the amount of address space available to Oracle8*i*. In Windows NT's Performance Monitor, under the Process object, monitor the Virtual Bytes counter for the "ORACLE" process. If this counter approaches 3GB, then out of memory errors can occur. If this

happens, reduce db_block_buffers and/or VLM_BUFFER_MEMORY or AWE_WINDOW_MEMORY until the database is able to start up.

8. Currently, there is a limitation in Server Manager for NT whereby the amount of database buffers displayed during database startup is incorrect if more than 4GB of buffers are in use. For instance, if 5GB of buffers are used, Server Manager will incorrectly report that 1GB are being used. This limitation is fixed in Oracle8*i* Release 2.

Tuning Considerations

There are a few tuning considerations to deal with when using these VLM implementations. The first and most important is that Oracle is still limited to 3GB of address space in which it must put the buffer headers for the database buffers, the rest of the SGA, PGAs, stacks for all the threads, code, and other memory allocations. Setting db_block_size to a large value causes many database buffer headers to be allocated from Oracle8*i*'s address space. Likewise, setting VLM_BUFFER_MEMORY or AWE_WINDOW_MEMORY high again uses a lot of address space. Using too much address space for these structures has the effect of limiting how many connections can be made to the database or limiting other memory allocations needed to perform database operations. Lowering them will sacrifice some performance since fewer buffers are quickly accessible in the normal Oracle8*i* address space. If out of memory errors occur when running with VLM, one or the other of the above parameters will need to be reduced.

One final observation on the use of VLM is that it is useful even when running on a machine with only 4GB of RAM installed. On such a machine, Oracle is normally limited to 3GB of memory, while Windows and other applications on the system use the remaining 1GB. However, Windows does not need 1GB of physical RAM to perform its tasks. Instead, there are typically several hundred megabytes of RAM available and not being used in a typical installation. On Windows 2000, turning on the AWE support allows Oracle to access perhaps an extra 500MB of buffers just by bumping up db_block_buffers and setting AWE_WINDOW_MEMORY appropriately. Likewise, on Windows NT v4.0 Enterprise Edition, the /maxmem flag in boot.ini can limit NT's memory usage to 3.5 GB total and allow the PSE36 driver access to the remaining 500MB. Turning on the PSE36 support in Oracle will then allow the database access to the extra 500MB of buffers.

Large User Populations

One area in which much activity has been undertaken is an effort to support large numbers of connected database users on Windows. As far back as Oracle7 version 7.2, there have been customers in production with over 1000 concurrent connections to a

single database instance on Windows. As time has progressed, that number has increased to a point where a recent 3rd party benchmark connected over 2200 users concurrently to the database. When using the Oracle Multi-threaded Server architecture, which limits the number of threads running in the Oracle database process, over 10,000 simultaneous connections have been accomplished to a single database instance. In addition, the Net8 multiplexing and connection pooling features can also allow a large configuration to achieve more connected users to a single database instance. Finally, Oracle Parallel Server can be used to again increase connection counts dramatically by allowing multiple server machines access to the same database files, thereby increasing capacity to tens of thousands of user connections and at the same time increasing throughput as well.

64-bit Support

The next leap in Oracle8*i* performance and scalability on Windows will happen when a 64-bit version of Oracle8*i* is released on the upcoming Intel Itanium (Merced) processor and the corresponding 64-bit version of Windows 2000. The development teams at Oracle have been working closely with these technology vendors to guarantee that Oracle8*i* on Windows will be released in production form very shortly after the hardware and operating system are generally available. As with other Oracle 64-bit ports to different UNIX variants, a 64-bit port of Oracle8*i* to Windows 2000 will be able to handle more connections, allocate much more memory, and provide much better throughput than the current version of the database on Windows. In addition, the migration path from 32-bit to 64-bit Oracle will be very straightforward. There will be no need to recreate databases, nor will a full export and import be required. All that will be needed is to copy the current datafiles to the new system, install the 64-bit version of Oracle8*i*, and startup the database as normal.

From an architectural perspective, the current, proven thread-based architecture will be used for the 64-bit port. As a result, creating the new 64-bit Oracle8*i* software basically entails re-compiling, re-linking, re-testing and re-releasing the new version. Very little new code will need to be written during the move to 64-bit since the underlying operating system APIs are expected to remain substantially the same. In addition, since Oracle8*i* has already been ported to other 64-bit ports, moving to 64-bit is a straightforward process that will produce a quality, stable product in a very short period of time.

Oracle8*i* Tuning on Windows

Included in the following sections are advice and guidelines for tuning Oracle8*i* on Windows from a hardware, operating system, and Oracle software perspective. Generic Oracle8*i* tuning advice (such as database structure layout and SQL statement tuning) which is applicable to all platforms on which Oracle8*i* runs is not included, since this is already covered extensively in the Oracle8*i* documentation library.

Hardware

As they say, "Bigger is better." This certainly applies when selecting hardware for an Oracle database server, since Oracle can typically use all resources provided to it, as needed by the particular application in question. However, not all applications require an extremely large server to achieve reasonable response times and performance characteristics. Included in the following sections are some very general guidelines for choosing or configuring specific hardware components.

CPU

An Oracle database is a particularly memory-intensive application that heavily exercises the system bus with memory accesses, especially in the case where a large SGA is in use. As a result, the CPU to memory interface is a crucial part of system performance when dealing with Oracle8*i* and in some cases is more important than raw CPU speed. Choosing the largest available level 2 (L2) data cache and the fastest system bus available is typically a very important consideration when selecting a system. CPU speed obviously is also an important factor, but it is sometimes the case that extra CPU speed will not result in a proportional increase in throughput if the system bus is a limiting factor in the configuration. In an SMP configuration in which multiple CPUs are used, this effect is amplified since there can be 2, 4 or 8 CPUs performing memory operations on the same system bus simultaneously. Having a large L2 cache can reduce the number of bus transactions required and can help achieve better scalability and throughput.

Memory

With the introduction of VLM support as described earlier, Oracle8*i* on Windows NT v4.0 Server Enterprise Edition and Windows 2000 Datacenter Server can now utilize up to 16GB of RAM for database buffers. (For Windows 2000 Server and Advanced Server, Oracle can support 4GB and 8GB of RAM for database buffers, respectively, the maximum amount of memory allowed on these systems.) As a result, memory sizing is no longer limited to 4GB, even though this number still implies a very high end system. At this point, the number of applications running on Windows that need 16GB of database buffers are very few indeed as most applications that need to scale this high typically are hosted on UNIX servers. With the release of 8-way SMP servers that run Windows, however, the number of large applications moving to Windows will increase and 16GB of RAM will become a more common configuration.

For smaller installations, memory should obviously be sized such that no paging (or minimal paging) occurs during normal database operations. Determining the right amount

of memory ahead of time is extremely difficult since an application often needs to be up and running before it's possible to determine the minimum required memory usage and SGA settings needed to achieve acceptable performance.

Disk Subsystem

Besides the CPU and memory, the disk subsystem is another crucial component of an Oracle8*i* server machine. This is particularly true for transaction-intensive applications that post many changes to the database. The following tips come from a relevant article in Microsoft's Knowledge Base (http://support.microsoft.com/support/kb/articles/Q199/1/60.ASP):

When selecting a disk subsystem, these general rules should be followed:

- SCSI is preferable to IDE.
- Fast wide SCSI is preferable to narrow/standard SCSI.
- RAID level 5 is preferable to a single disk.
- A hardware-based RAID controller is preferable to software RAID.
- More spindles are preferred over fewer spindles, to distribute writes.
- A caching disk controller is preferable to a non-caching controller, but a caching controller must have a battery backup to ensure data integrity.

Operating System Tuning

Compared to UNIX, Windows offers considerably fewer user-configurable parameters that can be adjusted to tune the operating system. This reduces the ability of system administrators to optimize Windows performance, but helps to make Windows easier to use than some operating systems.

There are still ways, however, to make Windows a better application server environment for the Oracle8*i* database. Most of these Windows-specific procedures have the effect of reserving more system resources for the Oracle8*i* database (for example, CPU, memory, and I/O bandwidth). These procedures are described in the following sections.

Use a Dedicated Server for Oracle

In general, the Windows computer that is running your Oracle8*i* database should not serve as any of the following:

- primary or backup domain controller
- file or print server
- remote access server

- router

These services consume network, memory, and CPU resources. In addition, the Windows computer that is running your Oracle8*i* database should not be locally accessed in high frequency or intensively used for local user processing, unless there exist significant resources to accommodate all this activity.

Configure NT to be an Application Server, not a File Server

The memory manager for Windows defines three different pools of memory that are allocated from available RAM. These pools include one pool for the operating system kernel and other system services, one pool for the file cache, and one pool for paged memory available to user applications. By default, a Windows server is set up to be a file and print server. As a result, the file cache component of system memory is large in anticipation of file server activity. However, when running an Oracle8*i* database, the file cache is not really needed at all since all Oracle file I/O operations bypass the file cache altogether and force data to be written directly to disk. This is required in order to ensure data integrity. In addition, the memory Oracle allocates for the SGA acts as Oracle's own private file cache, again making an operating system cache unnecessary.

To change the default behavior of Windows and reduce the size of the file cache, go to Start → Settings → Control Panel → Network → Services. Double-click the "Server" service and select "Maximize Throughput for Network Applications."

Reduce Priority of Foreground Applications on the Server Console

By default, the application currently in the foreground on Windows is given a higher priority than background processes. On a machine running an Oracle8*i* database, it is not desirable for a foreground application to have precedence over the database process. To change the default behavior such that the foreground application does not receive a performance boost, go to Start → Settings → Control Panel → System and hit the Performance tab. Then, select "None" as the setting for "Performance Boost for the Foreground Application."

Remove Unused Network Protocols

Since Windows supports many network protocols, it is common to have several enabled in a typical installation. However, very often not all of these protocols are required or used when running an Oracle8*i* database. If there are unnecessary protocols installed on a

system, it is recommended that they be removed so that the operating system does not devote processing time to these protocols.

To remove unnecessary network protocols, go to Start → Settings → Control Panel → Network. Click the Protocols tab. Select any unnecessary protocols not needed for the system's configuration and click Remove.

Reset the Network Protocol Bind Order

If there is a need to have several protocols installed on the server machine, the ones used most by Oracle should be given priority over those that are not.

To reset the network protocol bind order, go to Start → Settings → Control Panel → Network. Click the Bindings tab. Show bindings for all services, and then double-click Server to see the list of current protocols. Verify that the protocol used by Oracle is at the top of the list, and if it is not, make it so by selecting it and moving it up until it is at the top. If there are multiple network cards installed in the machine, then verify that the card used most frequently by Oracle is at the top of the list for each protocol as well. To do this, double click each protocol that Oracle uses, and move the appropriate network card to the top of the list for that protocol.

Page File Sizing

Although excessive paging is always discouraged when trying to achieve good performance from an application, it is recommended that the total combined size of all page files in the system be at least equal to the amount of RAM in the computer. Many installations go beyond this amount and have combined page file sizes that are two or more times as large as the amount of RAM in the machine. Sizing page files in this way provides enough cushion to avoid a situation in which Windows runs out of page file space and is unable to successfully perform the tasks required of it. In addition, since Windows balances page file activity across page files, it is important to place different page files on different physical disks in order to balance the I/O load on those disks.

Apply Latest Reliable Service Pack and Device Drivers

As Microsoft releases service packs for Windows, it is usually desirable to upgrade to these new releases, since they often fix critical bugs, increase operating system stability, contain performance enhancements, or even add new functionality or programmatic interfaces to the operating system. In general, however, it is best to wait a few weeks after a service pack is released before upgrading since there are frequently bugs and

incompatibilities that arise in certain instances. By waiting, these problems can be weighed against the benefits of the service pack to determine if upgrading is indicated. The latest Windows NT Service Packs may be downloaded from http://support.microsoft.com/Support/NTServer/Content/ServicePacks/Default.asp.

Oracle's current support policy with regard to Windows Service Packs is that Oracle does not specifically certify all products against specific Microsoft operating system service packs, but does support the use of its products on any service pack when that service pack becomes generally available. Depending upon the severity, quantity and impact of the service pack related issues found, Oracle may recommend that customers wait until relevant Oracle patches have been released before upgrading to a particular service pack. Oracle does not, and will not, recommend or discourage the installation of specific service packs unless the service packs will significantly affect the operation of Oracle software, either positively or negatively. If such a statement is deemed necessary, then Oracle will disseminate this statement in as timely a fashion as possible after the release of the service pack in question.

In addition to the occasional service pack, device drivers for a system's I/O controllers, network cards, and video subsystem should also be periodically updated as patches become available. From an Oracle perspective, the I/O device drivers are most critical and updates to these software pieces can increase stability and performance of an Oracle database. When running a RAID configuration, this is even more crucial since experience has shown that device drivers for this type of configuration tend to be more complex and more prone to problems that for other setups.

Disable Unnecessary Services

Additional memory can be provided to an Oracle8*i* database on Windows by disabling unnecessary Windows services. In a typical Windows installation, there are usually several services that are enabled by default that are not necessary for the current configuration. By going into Start → Settings → Control Panel → Services, the list of all running services can be found. By consulting the Windows documentation and the system administrator, a few services that are running typically prove to be extraneous. By disabling these services, more memory becomes available for Oracle.

Close All Unnecessary Foreground Applications

In addition to stopping unnecessary services, it is also advisable to close any non-needed foreground applications. Some of the more common places for optimization occur in the areas of the Startup folder, MS-DOS Command Prompts, and screen savers.

Startup Folder

The Startup folder of Windows frequently contains applications that are not vital to the operation of the server upon which Oracle8*i* resides. These applications use memory that otherwise could be provided to Oracle and can have an effect upon the operation of the database. In particular, applications which periodically index documents, scan files, or do other disk related activity have been shown to have a temporary detrimental effect upon database performance.

MS-DOS Command Prompts

MS-DOS Command Prompts can utilize a large portion of the CPU when the command prompt window is being repainted constantly due to scrolling of data or commands. A common situation in which this occurs is the running of a SQL script in Server Manager or SQL*Plus by a database administrator. These scripts can complete much more quickly and save CPU cycles just by minimizing the MS-DOS Command Prompt window in which they are running.

Screen Savers

As with MS-DOS Command Prompts, screen savers can also consume significant CPU resources. Instead of enabling a screen saver, either lock the workstation or turn off the monitor altogether. If a screen saver needs to be run, go to Start → Settings → Control Panel → Display, hit the Screen Saver tab, and select Blank Screen as the screen saver for the system.

Oracle Tuning

In addition to the hardware and the operating system, there are Windows-specific procedures that can be performed to tune the Oracle8*i* database software and its related files. These, of course, are in addition to generic Oracle tuning that is common to all platforms on which Oracle runs.

Database Files

The optimal placement of database, log and control files depends upon how many disk spindles are present in the system, which RAID levels are being run, and also performance characteristics of the I/O controllers and subsystem. Typical Oracle recommendations from the generic documentation certainly apply to Windows and there is little Windows-specific information necessary with regard to the placement of database files. However, two areas that should be mentioned are database file fragmentation and compression.

Fragmentation

Once created, Oracle database files do not increase or decrease in size, unless the expandable datafile feature is turned on. As a result, no fragmentation can occur to database files after they are created except for fragmentation that existed at creation time (assuming no expandable datafiles). So, to eliminate datafile fragmentation, do the following:

- shut down the database
- de-fragment the database files using a third-party de-fragmentation utility
- startup the database

By following these procedures, it is guaranteed that no database file fragmentation can occur unless the expandable datafiles feature is turned on. Further de-fragmentation is unnecessary, such that the database files can be excluded from subsequent de-fragmentation runs.

If expandable datafiles are in use, it is recommended that de-fragmentation only occur when the database is shut down since the inter-operation of Oracle software and de-fragmentation utilities is not guaranteed.

Compression

Compression of database files, log files, and control files is not supported by Oracle since compression of database files can cause a database instance to abort itself in the case of a write error to the compressed file. The technical reason for this behavior is as follows. When Oracle database files are first created, they are cleared out and filled with zeros. These new files compress extremely well since there is no data in them. As a result, the drive on which the database files reside continues to have a lot of free space available on it. If that drive were to fill up with other files before the Oracle database file became fully populated, the database would run into a situation where it tried to write to the database file but failed, since the compressed file could not be expanded due to lack of space. Any write or read error to an Oracle database file is treated as a fatal error by the database server, which causes the whole database instance to abort itself.

Affinity and Priority Settings

The Oracle8*i* database supports the modification of both priority and affinity settings for the database process and individual threads in that process when running on Windows.

By modifying the value of the ORACLE_PRIORITY registry setting, a database administrator can assign different priorities to the individual background threads and also

to the foreground threads as a whole. Likewise, the priority of the entire Oracle process can also be modified. In certain circumstances, this may improve performance slightly for some applications. For instance, if an application generates a great deal of log file activity, the priority of the LGWR thread can be increased to better handle the load put upon it. Likewise, if replication is heavily used, those threads that refresh data to and from remote databases can have their priority bumped up as well.

Much like the ORACLE_PRIORITY setting, the ORACLE_AFFINITY registry setting allows a database administrator to assign the entire Oracle process or individual threads in that process to particular CPUs or groups of CPUs in the system. Again, in certain cases, this can help performance. For instance, pinning DBWR to a single CPU such that it does not migrate from one CPU to another can in some cases provide a slight performance improvement. Also, if there are other applications running on the system, using ORACLE_AFFINITY can be a way to keep Oracle confined to a subset of the available CPUs in order to give the other applications time to run.

Both ORACLE_PRIORITY and ORACLE_AFFINITY are described in more detail in the Windows-specific documentation that accompanies Oracle8*i* on Windows.

Orastack

As described previously in the *Architecture* section, each Oracle instance is limited to 3GB (or 2GB if not running on NT v4.0 Enterprise Edition) of address space from which it must allocate the SGA, PGAs for all the threads, application code, and stacks for each thread that are used to store variables and thread state. By default, each thread is provided with 1MB of reserved address space for its stack. This reserved space is not backed by physical memory, but is just space in Oracle's address space that is reserved should the thread need to use 1MB of stack. In practice, 1MB is a very large, conservative number that is bigger than the database typically needs. However, it is set this way by default in order to make sure that for the few cases where it is needed, it is set to a sufficiently high value. The downside to this large setting is that 1MB of address space per thread adds up very quickly when 1000 or more threads are running in the Oracle process. For 1000 threads, 1GB of address space is used just for stacks and the threads do not need most of this space anyway. This large address space usage can limit how many connections can be made to the database or how big the SGA can be for a given number of connections.

To give administrators some tuning flexibility, Oracle provides a utility called ORASTACK that enables an administrator to lower the stack size for Oracle threads from 1MB down to a smaller number. This allows for either higher connection counts or a larger SGA in those cases where Oracle is bumping up against the 3GB address space limit. If an application does very little highly recursive SQL or not much in the way of nested triggers or stored procedures, then turning the stack size down to 300K (for instance) is a safe procedure that will save on address space usage. If the stack space is decreased too much, typical behavior will be "ORA-03113: end-of-file on

communication channel" errors returned to the client as its foreground thread terminates with a stack overflow error.

To use ORASTACK, run the following commands on the server machine:

```
C:\> orastack c:\oracle\ora81\bin\oracle.exe 300000
C:\> orastack c:\oracle\ora81\bin\tnslsnr.exe 300000
C:\> orastack c:\oracle\ora81\bin\svrmgrl.exe 300000
```

The first command will cause the stack for background threads to be 300000 bytes as opposed to 1MB for all instances running on the machine. The second command will cause the stack for all foreground threads running on behalf of network clients to be 300000 bytes. The third command causes the stack for all foreground threads running on behalf of Server Manager on the server machine to be 300000 bytes. If there are other executables that run on the server machine that connect to the database, ORASTACK can be run on those executables as well to modify the stack sizes of their corresponding foreground threads.

Operationally, all that ORASTACK does is modify part of the executable header in the .exe file to reflect the new stack specified by the user.

In general, ORASTACK is only called for in high-end installations where there are several hundred or more connections to the database or when the SGA is very large (over 2GB in size). To determine how much of the 3GB address space is in use by Oracle, run Windows Performance Monitor by going to Start → Programs → Administrative Tools → Performance Monitor. Once in Performance Monitor, choose Edit → Add to Chart... and select the Virtual Bytes counter of the Process object for the Oracle instance. If this value displayed is close to 3GB, then the amount of available address space for Oracle is running low. When there is no more address space free for Oracle, typical errors encountered are out of memory errors or connection spawning errors. It is in these circumstances that ORASTACK can be useful.

Optimal Flexible Architecture

While not exactly a tuning procedure, the Oracle Optimal Flexible Architecture (OFA) is a structured method for installing Oracle databases and applications in a way that promotes ease of maintenance through better file organization, increases reliability through multiple disk support, and enhances performance through decreased I/O contention for disks that contain data files and database files (log, trace, etc.).

The addition of the Optimal Flexible Architecture to the Oracle installation process provides the following functionality:

- Organizes files by type and usage. Binary files containing Oracle code are installed under one directory; control files, log files, and administrative files are

installed under a separate directory while database files are installed in yet another directory.
- Enables the separation of files onto different disk devices and into different directories.
- Promotes reliability in case of a disk failure.
- Promotes performance by decreasing disk contention for data files, binary files and administrative files that could now reside on separate disks.
- Simplifies backup and recovery by way of a more organized file layout.
- Preserves administrative and organizational advantages of OFA on UNIX.

For more information on the specifics regarding OFA, consult the Windows-specific documentation that comes with Oracle8*i* for Windows.

Summary

In summary, Oracle's database on Windows has evolved from a port of its UNIX database server to a well-integrated native application that takes full advantage of the services and features of the Windows operating system and underlying hardware. Oracle continues to improve the performance, scalability, and capability of its database server on Windows, while at the same time producing a stable, highly functional platform on which to build applications. Oracle is fully committed to providing the highest performing, most well integrated database on the Windows platform going forward as Windows 2000 and 64-bit versions of Windows become available.

For further information on Oracle's Windows products, please visit the following links:
http://otn.oracle.com/tech/windows/