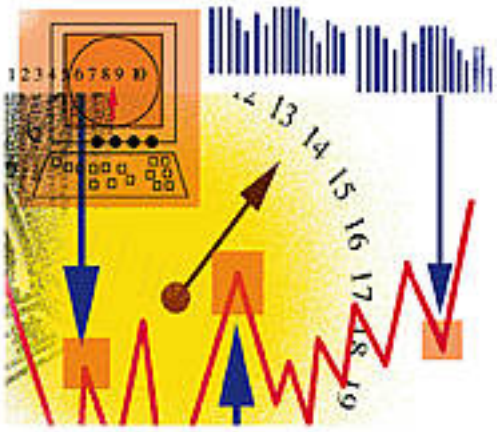
Sys Admin
the journal for UNIX systems administrators

Optimizing Solaris for Oracle 8.x



Leonardo Orellano and Timothy Young

Systems administrators are often called upon to configure or "optimize" an operating system for a given application. Although often unfamiliar with the application itself, administrators are challenged to ensure the optimal performance of the product that is implemented on the platform constituting the area of their expertise. Although much documentation may have been written to assist in the configuration and maintenance of the application from within its own interface and configuration files, clear, concise information for tuning OS-specific variables is often needed. One common example of this scenario is the optimization of Sun Solaris for Oracle 8.x.

About 70% of all Solaris environments contain Oracle implementations; not surprisingly, the majority of Oracle installations occur on the Sun platform. The systems administrator plays an integral role in the implementation process, and is ultimately responsible for the assurance that the database administrator has delivered a finely-tuned environment with which to work. The initial achievement as well as the continuing success of the implementation depend upon effective teamwork and informed communication between the Solaris systems administrator and the Oracle DBA. The sharing of information and understanding of the needs of this closely related (and often symbiotic) area of expertise also provide the systems administrator with an easily accessible path toward the augmentation of skills and increased experience.

As she works with developers and DBAs, the systems administrator will attempt to identify quantifiable architectural and performance-related issues, as well as the methods by which such concepts as high-availability and fault-tolerance can be realized. The general strategy is to facilitate the tuning process by focusing on four major areas: operating system, application, database, and network.

The Solaris operating environment is viewed as encompassing key ingredients

essential for the proper functionality of the Oracle database. The dynamic kernel must be modified to support Oracle's requirements, I/O must be facilitated to avoid bottlenecks and areas of contention, the appropriate filesystem type must be chosen, and careful consideration must be given to memory allocation.

Configuring the Solaris Kernel

As the Solaris kernel is automatically rebuilt at boot-time, modifying it involves adding parameters to the initialization process. Oracle requires that certain assignments be made in the `/etc/system` file to customize the way in which Solaris manages semaphores and shared-memory segments. The Oracle installation guide provides for minimum values for these parameters, although they may need adjustment depending upon the specific configuration of the server (n-tier architecture, business processes, etc.). The following kernel parameters must be specified (remember, any modifications to `/etc/system` require a reboot to take effect):

- `shminfo_shmmax` -- Maximum size in bytes of a shared memory segment. This parameter should be set to around 60% of the physical memory size.
- `shminfo_shmmin` -- The smallest size in bytes of a shared memory segment.
- `shminfo_shmseg` -- Maximum number of shared memory segments to which a process can attach.
- `shminfo_shmlba` -- Controls alignment of shared memory segments. All segments must be attached at multiples of this value.

The next parameters deal with semaphores, which are counters used by Oracle to monitor and control the availability of shared memory segments. Although they are typically used to synchronize access to shared resources, Oracle utilizes them to control timing throughout a variety of processes. For this reason, the traditional Solaris configuration is insufficient for the needs of the Oracle instance and should be modified.

- `seminfo_semmns` -- Maximum number of semaphores for the entire system.
- `seminfo_semmni` -- Maximum number of semaphore sets.
- `seminfo_semmsl` -- Maximum number of semaphores in a semaphore set.

Although the values supplied by the Oracle documentation should be sufficient for a vanilla Oracle install, semaphores can be calculated by performing the following operations:

1. Record the value for the "processes" parameter in `$ORACLE_HOME/dbs/init<SID>.ora` (where SID refers to the

name of the Oracle instance). This is the Oracle initialization file, and each instance of Oracle will have its own file. If multiple instances are present on the same server, the "process" values from all `init<SID>.ora` files should be added together and the sum recorded.

2. The general formula for calculating semaphore requirements for Oracle 8 is: `seminfo_semmns` >2 X (("processes" value from step 1) + 10 X (number of Oracle instances)) + 15%.
3. Note that the sum of the Oracle background processes and the user processes may not exceed this limit.
4. `Semmsl` is an arbitrary value that is best set to a round figure no smaller than the smallest "processes" value for any instance on the system.
5. Semaphores are allocated by Solaris in "sets" of up to `semmsl` semaphores per set. One can have a MAXIMUM of `semmni` sets on the system at any time. To determine Oracle's requirement for `semmni`, use the following formula: sets required for an instance = (value of "processes")/`semmsl`. This value should be rounded up to the nearest integer and given an additional 15% for system overhead.

Here are some common commands for diagnosing Oracle memory usage:

- `Ipcs -b` -- Look at the SEGSZ field for shared-memory size in use.
- `ps -elf` -- Remember that the SZ field is in 4K pages for Oracle.
- `/usr/ucb/ps alx` -- SZ field shows amount of swap-space in kb used by a process.
- `crash` -- Can be used to display kernel values. Read the man pages.

How Oracle Uses Memory

To understand the mechanism by which Oracle utilizes memory, the systems administrator should first have a basic understanding of the instance configuration file. As mentioned previously, each instance of Oracle relies upon a file labeled `init<SID>.ora` in the Oracle home directory, where SID is simply the name of the database instance as determined at the time of installation. This file contains configuration parameters for the instance, the most important to the systems administrator being:

- `db_block_size` -- The size in bytes of an Oracle block. (Note that this value can only be changed by recreating the entire database.)
- `db_block_buffers` -- The number of blocks assigned to the instance.
- `log_buffer` -- The number of blocks allocated to the Redo log buffer.
- `shared_ pool_size` -- The size, in Oracle blocks, of the shared pool area.
- `processes` -- The number of simultaneous users in the system.

Upon startup, the Oracle instance performs the following operations:

- Read `init<SID>.ora`.
- Start the background processes.
- Allocate required shared memory and semaphores.

The main Oracle memory structure is the System Global Area (SGA). The size of the SGA will be calculated from various `init<SID>.ora` parameters. The sum of these parameters yields the minimum amount of shared memory required. The SGA is broken up into four sections -- the fixed section, which is constant in size; the variable section, which varies in size depending on some `init<SID>.ora` parameters; the redo block buffer, which has its size controlled by the `init<SID>.ora` parameter `log_buffers`; and the db block buffer, which has its size controlled by `db_block_buffers`.

The size of the SGA is the sum of the sizes of the four sections. There is no simple formula for determining the size of the variable section. Generally, the shared pool dominates all other parts of the variable section. So, as a rule of thumb, one can estimate the size as the value of `shared_pool_size`, which is determined by the DBA through tuning the Dictionary cache hit ratio and the Library cache hit ratio.

Oracle has three different possible models for the SGA: one-segment, contiguous multi-segment, and non-contiguous multi-segment. When attempting to allocate and attach shared memory for the SGA, it will attempt each one in the given order. The entire SGA must fit into shared memory, so the total amount of shared memory allocated under any model will be equal to the size of the SGA.

The one-segment model is the simplest and first model tried. In this model, the SGA resides in only one shared memory segment. Oracle attempts to allocate and attach one shared memory segment of size equal to total size of the SGA. The size of the SGA must be smaller than the configured SHMMAX in `/etc/system`; otherwise, the SGA will need to be placed in multiple shared memory segments, and Oracle will proceed to the next memory model for the SGA. With multiple segments there are two possibilities. The segments can be attached contiguously, so that it appears to be one large shared memory segment, or non-contiguously, with gaps between the segments. At this point, Oracle calculates SHMMAX using a binary search algorithm to determine how many segments will be required. In the contiguous segment model, Oracle simply divides the SGA into SGA/SHMMAX (rounded down) segments. It then allocates and attaches one segment at a time, so that all segments are contiguous in memory. If an error occurs, Oracle tries the next model for SGA allocation, non-contiguous segments.

The last model Oracle will try is the non-contiguous model. After calculating SHMMAX, Oracle first checks to see whether it can put the fixed and variable section into one shared memory segment just large enough to hold the two sections. If it cannot, it will put each into its own separate segment. Then Oracle will compute the number of redo block buffers and db block buffers it can fit in a segment. Oracle can then compute the total number of segments required for both the redo and database block buffers. These segments will be of a size just large enough to hold the buffers (so no space is wasted). The total number of segments allocated will then be the number needed for the fixed and variable sections (1 or 2) plus the number needed for the redo block buffers plus the number of segments needed for the database block buffers. Once the number of segments and their sizes is determined, Oracle will allocate and attach the segments one at a time, non-contiguously. The total size of segments attached will be exactly the size of the SGA with no space wasted. Once Oracle has the shared memory attached, Oracle will allocate the semaphores it requires.

A good rule of thumb is: SGA=60% physical RAM.

I/O Strategies

The best way to ensure balanced I/O throughput is through proper architecture of the storage system. This process includes allocating swap space, utilizing the appropriate filesystem type, and choosing a RAID configuration that provides for both availability and fault-tolerance.

Although the minimum amount of swap space created should be the size of the Oracle SGA plus the amount of system RAM, this may often prove insufficient. A better policy is to allocate twice the amount of system RAM to the swap partition, possibly even more in some instances.

The Oracle database can reside on a number of filesystem types, such as UFS, raw, and vxfs. Each offers certain advantages over the others, and all exploit Solaris's Asynchronous Input/Output (AIO) to enhance performance. Additionally, all of these filesystem types can be utilized in combination with a Logical Volume Manager(LVM) for maximum flexibility.

The Unix File System (UFS) is the standard filesystem type in Solaris, and will be very familiar to the systems administrator. It is fairly easy to manage, performs well, and can be backed-up through the use of many standard commands. It employs a read-ahead capability when performing large sequential read accesses, places data in the Solaris kernel buffer cache to accelerate subsequent scans on identical objects, and allows for the logical block size to be set to the same value as the Oracle database block size when the filesystem is created.

I've heard repeated claims that raw devices offer a better solution for Oracle

databases than UFS. I think these claims are generally based more upon marketing slogans than actual laboratory evidence. Although it is true that the use of raw devices can offer slightly better performance in certain situations (Oracle Parallel Server actually requires it), the virtual impossibility of moving data files around can have a serious pejorative effect on the ability of an administrator to tune the database. Moreover, the performance gain seen from migrating a database from a UFS partition to a raw device is seldom perceived for what it really is -- a result of the rebuilding of database objects (including indexes and tables), and the elimination of row migration and chaining. This initial benefit fades over time, and can, in fact, be gained just as easily by migrating a badly chained table from a raw device to a Unix filesystem. DBAs are often tempted to alleviate the struggles of an I/O-bound machine by adopting raw devices, but this is often an attempt to address a symptom rather than an underlying problem or set of problems. A systems administrator who observes a disk I/O performance bottleneck should suggest that the DBA perform SQL trace analysis and revisit the current indexing, joining, and data modeling configurations.

The Veritas File System (vxfs) is a commercial, extent-based filesystem. Because it is extent-based, a file is saved on disk as a single extent with the same size as the file, without the use of indirect blocks. This results in very quick sequential reads. Although the use of large extents causes disk fragmentation, Veritas provides tools for data de-fragmentation through the moving and merging of extents. Vxfs partitions bypass kernel buffers, enjoys high-performance direct I/O, and employs journaling features. All of the capabilities found in UFS are also included. Perhaps the most attractive aspect of vxfs is its ability to allow online resizing of partitions. With the addition of snapshot backup capabilities as a final bonus, the advantages to this filesystem type should be clear.

High availability and fault tolerance are primarily determined by the RAID configuration. Industry metrics have clearly demonstrated the superiority of RAID 0+1 and RAID 1+0 (termed Stripe Pro in Veritas Volume Manager) over RAID 5 when used in an Oracle environment. The stripe size should be set to a multiple of (db_block_size X db_block_multiread_count). Both of these values are found in the `init<SID>.ora` file.

Tuning NET8 for Increased Network Performance

Optimizing network performance essentially consists of two designs: increasing bandwidth and reducing network traffic. Because available bandwidth is a factor usually outside the control of the systems administrator, the best that she can do is to ensure that the server is configured to maximize the connection it has been given. She should confirm the transmission speed and duplex status of the interface. This can be accomplished by momentarily unplugging and replacing

the network cable itself and watching for a system message displaying the connection properties, or sometimes by the following:

```
grep duplex /var/adm/messages
```

It is not unusual for an Ethernet adapter to fail in negotiations with certain switches, resulting in a half-duplex connection. In these instances, the switch should be set to full-duplex, and the interface should likewise be forced into full-duplex mode by the addition of the following lines into `/etc/system` (assuming a 100-Mbps Sun Ethernet interface):

```
set hme:hme_adv_autoneg_cap=0
set hme:hme_adv_100fdx_cap=1
set hme:hme_adv_100hdx_cap=0
set hme:hme_adv_10fdx_cap=0
set hme:hme_adv_10hdx_cap=0
```

The remaining avenues for improving network performance rely upon reducing the amount of traffic. Certain external factors should be considered in order to ensure the proper configuration and placement of the server in the networked environment, such as the use of a domain name service, network topology, number of hops between clients and server, and heterogeneous protocols. The RDBMS (or SQL*Plus on a client) resides at layer 7 of the OSI model, with NET8 occupying layers 5 and 6, and the underlying protocol (assumed here to be TCP/IP) operating at layers 3 and 4. (More information on tuning Solaris TCP/IP parameters can be found at: http://www.sunhelpdesk.com.)

Although primarily concerned with the configuration of Solaris, the systems administrator will benefit tremendously if she is familiar with the factors that a database administrator considers when tuning NET8. Topics of discussion between both individuals should include pre-spawn processes, multi-threaded server vs. dedicated connections, status of NET8 tracing, status of security features, and size of the `tnsnames.ora` file. Although pre-spawn server processes and multi-threaded server connections offer improved performance, the enabling of trace files and encryption/decryption algorithms increase connect time and processing requirements. Because each client-initiated transaction causes the entire `tnsnames.ora` file to be read before data can be retrieved from the server, the size of this file can significantly impact connect time as well.

DBAs also attempt to reduce query time by effective indexing. A very important factor here is the array size, which may range from 1 to 5000. This value represents the number of rows that Oracle will fetch before it passes them to the NET8 server, and then on to the client. Although a large value requires more host memory, it increases the efficiency of queries that fetch many rows.

Because this affects NET8 packet size, the impact on the data stream is obvious. The size is specified in SQL*Plus with the following command:

```
SQL> set array_size value
```

The value of the Oracle Session Data Unit is specified in the `tnsnames.ora` file and determines the size of the NET8 buffer. Although the default value of 2K is generally sufficient, it can be increased up to 32K. Ideally, it should be the same as the MTU to avoid either under-utilization or fragmentation; however, as each network layer adds header information and presents buffer incompatibilities, this rarely produces the actual desired effect. The variable nature of packet size, array size, and row size further skew the relationship. In any event, the SDU must be negotiated between the client and the server, with the lesser value prevailing. It must also be remembered that the MTU itself may not always be the standard 1500, as in Sun's Gigabit Ethernet.

Advanced Concepts

We encourage systems administrators working with Oracle to explore advanced concepts, such as Oracle Parallel Server, Sun Cluster, database roles (OLTP, DSS, etc.), and Veritas Database Edition. They may also wish to explore binding Oracle processes to specific SPARC processors, using pbind. Remember not to bind the database writer (dbwr) process to the same CPU as any other Oracle background processes, unless using raw devices. Constant updates and improvements in Oracle, Solaris, LVM's, and filesystems make the art of OS tuning a moving target, so administrators should stay aware of recent developments and encourage employers to facilitate formal instruction when appropriate. After all, a successful Oracle implementation depends every bit as much upon the systems administrator as upon the DBA.

Leonardo Orellano is an Oracle Certified Database Administrator, as well as a Sun Certified System Administrator. He has more than seven years experience administering Oracle in various environments, and is a Senior Consultant for Tech Data corporation. He can be reached at: lorella@techdata.com.

Timothy Young is a Sun Certified Enterprise Systems Engineer, with ten years experience architecting Sun environments and implementing applications such as Oracle. He is a Senior Systems Engineer for Bay Data Consultants. He can be reached at: tyoung@bay.com.

The authors thank Dr. Adam Butera for his continuing guidance in the area of neural networks and Aron Elston for his valuable input in enterprise planning.

Copyright © 2001 Sys Admin, Sys Admin's Privacy Policy. Comments about theWeb site: webmaster@sysadminmag.com