An Oracle White Paper
September 2010

Dynamic SGA Tuning of Oracle Database on Oracle Solaris with DISM

Introduction

Oracle Database has matured over many years, and current releases support a rich set of features that greatly expand database functionality and improve performance. Fortunately, increased functionality has not led to greater complexity – each new major release of Oracle Database has featured fewer tunables thanks to automatic tuning of database instances.

Management of the System Global Area (SGA) illustrates these improvements. Recent releases of Oracle Database can automatically assign more memory, up to a prescribed limit, to both the buffer cache and the shared pool if additional memory will improve performance. The Oracle Solaris Operating System supports such operations with the Dynamic Intimate Shared Memory (DISM) feature. This whitepaper describes the purpose of DISM, its function, the steps necessary to ensure its effective behavior, and potential pitfalls that should be avoided.

It is important to note that significant performance degradation can occur if DISM is not configured correctly, and for that reason Oracle recommends that DISM be turned off by default on SPARC-based servers. For users who need the functionality provided by DISM, this document should help identify how to configure and monitor DISM to ensure its correct behavior.

Oracle recommends that DISM should always be turned off on x86-based systems running Oracle Solaris 10.

DISM is turned on by default for Oracle Database 11.2.0.1 on Oracle Solaris, which makes it important for Database Administrators (DBAs) to understand its capabilities and behavior. The MEMORY_TARGET parameter and its implications are discussed later in this paper.

.

THE ROLE OF SHARED MEMORY IN ORACLE DATABASE

There is usually considerable commonality in the tasks undertaken by users connected to the same database instance, and users running transaction-based workloads in particular frequently access many of the same data blocks. For this reason, Oracle Database keeps frequently used data blocks in a cache in memory called the Buffer Cache, and shares other frequently accessed information, such as table metadata and parsed (processed) SQL statements, in a second memory cache called the Shared Pool. These memory caches are held in a single shared memory to allow multiple users to access them concurrently. Shared memory also facilitates interprocess communication.

Introduction to ISM and DISM

Since shared memory is very heavily used in Oracle Database environments, it is important to optimize access to it and to minimize the amount of CPU consumed while referring to it. With this in mind, a specially-tuned variant of System V Shared Memory was introduced in Oracle Solaris many years ago, called Intimate Shared Memory (ISM). ISM has been widely used for database shared memory since.

Intimate Shared Memory (ISM)

Intimate Shared Memory (ISM) is functionally equivalent to System V shared memory. Oracle Database invokes ISM automatically on your behalf by adding an optional flag – the `SHM_SHARE_MMU` flag – when attaching to Oracle Database's SGA shared memory segment (via the `shmat(2)` system call).

When Oracle Database attaches to an ISM segment, all memory pages are instantiated and locked in memory. The result is that shared memory is always instantly available. If, for some reason, the requested amount of shared memory cannot be locked in memory (for example, not enough physical memory is available, or the user does not have permission to allocate that much locked memory), the database instance startup will abort and an error message will be posted in the alert log.

Performance Benefits of ISM

ISM offers a number of performance benefits over standard System V shared memory:

- ISM shared memory is automatically locked by the Oracle Solaris kernel when the segment is created. This not only ensures that the memory cannot be paged out, it also allows the kernel to use a fast locking mechanism when doing I/O into or out of the shared memory segment, avoiding the need to use mutual exclusion locks (mutexes) and thereby saving significant CPU time.
- Kernel virtual to physical memory address translation structures are shared between processes that attach to the shared memory, saving kernel memory and CPU time.
- Large pages, supported by the system’s Memory Management Unit (MMU), are automatically allocated for ISM segments. Large pages can reduce the number of memory pointers by more than 500 times. This reduction in complexity translates into noticeable performance improvements, especially on systems with very large amounts of memory.

Other Benefits of ISM

Since ISM memory is locked, no swap space is needed to back it, thereby saving disk space. ISM memory can also be locked by the oracle user without the need for superuser privileges, thereby simplifying system administration.

ISM Limitations

The one significant limitation of ISM is that ISM segments cannot be resized. To resize an ISM-based database buffer cache, the database must be shutdown and restarted, affecting application availability. If the need arises to remove significant amounts of memory by Dynamic Reconfiguration, for example, database instances may need to be shutdown. Further, dynamic resizing of the buffer cache and shared pool will not be possible.

Enabling ISM

From Oracle Solaris 10, it is not necessary to carry out any configuration changes to the operating system to allow a database to startup, unless the SGA size for a single instance is expected to exceed 25% of physical memory. For larger SGA sizes, the `project.max-shm-memory` parameter for the `oracle` user should be set to the maximum SGA size required. The following commands can be used to set this parameter (run these commands as the root user):

```
# projadd oracle
# echo oracle::::project=oracle>> /etc/user_attr
# prctl -n project.max-shm-memory -v 16gb -r -i project oracle
```

The prctl command can be run at any time, and takes effect immediately.

This dynamic approach to administration offers more flexibility than was available in earlier Oracle Solaris releases. Before Oracle Solaris 10, an Oracle database could not be started without changes to the /etc/system file. Examples of such changes are illustrated below:

```
set semsys:seminfo_semmni=100

set semsys:seminfo_semmsl=256

set shmsys:shminfo_shmmax=17179869184

set shmsys:shminfo_shmmni=100
```

The value of the `shmsys:shminfo_shmmax` parameter represented the maximum shared memory size (in bytes). Changes to these parameters did not take effect until the next system reboot.

ISM in Virtualized Environments

Oracle Solaris Containers can be used to offer an isolated environment for Oracle Database, offering all the benefits of a dedicated system without the cost and additional administrative overheads associated with separate systems. Oracle Database is fully supported when running in Oracle Solaris Containers, and the combination is also recognized when determining per-core license requirements.

The main tunable parameter of interest with Oracle Solaris Containers is the `zone.max-shm-memory` parameter (a "zone" is an Oracle Solaris Container). This parameter determines the maximum shared memory limit for an Oracle Solaris Container rather than for a project, but it otherwise has the same basic function as the `project.max-shm-memory` parameter described above under "Enabling ISM", and can be modified in the same way.

Dynamic Intimate Shared Memory (DISM)

Oracle Solaris Dynamic Intimate Shared Memory (DISM) provides shared memory with the same essential characteristics as ISM except that it is dynamically resizable. That means that DISM, if configured correctly, offers the performance benefits of ISM while allowing the shared memory segment to be dynamically resized, both for the sake of performance and to allow dynamic reconfiguration (for example, adding or removing memory from a system or a domain).

When Oracle Database decides to use DISM, it invokes it automatically on your behalf by adding an optional flag – the `SHM_PAGEABLE` flag – when attaching to Oracle Database's SGA shared memory segment (via the `shmat(2)` system call).

The Reason for Introducing DISM

DISM allows applications to respond to changes in memory availability by dynamically increasing or decreasing the size of optimized shared memory segments. Oracle Database has supported DISM since the Oracle Database 9i release, and was the first commercial application to do so.

The ability to dynamic change the amount of memory allocated to an Oracle Database instance has a number of benefits, including the ability to

- Allow Oracle Database to dynamically tune the database instance.
- Support Dynamic Reconfiguration operations, for example, adding or removing memory on Oracle's Sun SPARC Enterprise M-Series servers using the eXtended System Control Facility (XSCF).
- Enable memory to be dynamically moved between different database instances according to changing processing requirements.

A large DISM segment is created when the database boots, with sections of it selectively locked or unlocked as memory requirements change. Instead of the kernel automatically locking DISM memory when the segment is allocated, though, as happens with ISM, locking and unlocking is carried out as required by the application (Oracle Database).

Performance Benefits of DISM

DISM shares the same performance benefits as ISM. In particular:

- Memory can be locked, preventing paging and allowing I/O to use fast kernel locking mechanisms. Memory can also be unlocked and freed dynamically and returned to the operating system, allowing it to be used for other purposes.
- Kernel virtual to physical memory address translation structures are shared between processes that attach to the DISM segment, saving kernel memory and CPU time.
- Large pages, supported by the system's Memory Management Unit (MMU), are automatically allocated for DISM segments.

Note that while ISM can be used on any Oracle Solaris release from 2.6, DISM should not be used with any Oracle Solaris release before Oracle Solaris 9 Update 2 (the 8/03 release). Oracle Solaris 10 is, however, the earliest recommended release for use with DISM on SPARC-based systems. DISM is not recommended for use with Oracle Solaris on x86-based systems.

Differences between DISM and ISM

While performance benefits are equivalent between DISM and ISM, other differences are worth noting:

- Unlike ISM memory, DISM memory is not automatically locked by the Oracle Solaris Operating System. The application (in this case Oracle Database) is responsible for locking the memory. Significant performance degradation will result if memory should fail to be locked for any reason, or if the specific application process that established the locks should die thereby releasing the locks. Possible failure scenarios and remedies are discussed below.
- Since DISM memory is not automatically locked, swap space must be allocated for the whole segment. The capacity of modern disk drives means that this difference need not be a significant issue. But it could become a problem if system administrators are unaware of the need to provide swap space for DISM. This difference is mostly likely to affect users that have exclusively used ISM in the past, but now find themselves using DISM.

Enabling DISM

The steps required to configure Oracle Solaris to enable DISM are exactly the same as for ISM. See the “Enabling ISM” discussion above for more information.

DISM in Virtualized Environments

Running DISM in an Oracle Solaris Containers has much the same implications as running ISM. Refer to “ISM in Virtualized Environments” above for more information.

How Oracle Database Decides Between ISM and DISM

Oracle Database 9i introduced a new init.ora parameter, SGA_MAX_SIZE, to activate DISM. This variable establishes the maximum size to which the SGA can grow; it can only be modified statically (in other words an Oracle Database reboot is required before any change to SGA_MAX_SIZE takes effect). Oracle Database will use DISM instead of ISM if SGA_MAX_SIZE is set larger than the total of the database buffers (in particular, DB_CACHE_SIZE dynamic SGA resizing is not supported with the older db_block_buffers parameter), the shared pool, the redo buffers, the large pool, the Java pool, and the SGA fixed size (representing Oracle's internal requirements).

Once DISM has been invoked, Oracle Database automatically locks an amount of memory determined by the total of the elements described above (database cache, shared pool, etc). Subsequently, the DBA can alter the size of the database buffers (DB_CACHE_SIZE) and the shared pool (SHARED_POOL_SIZE) with Oracle Database's alter system command. For example:

```
alter system set db_cache_size = <new size in bytes>
```

From Oracle Database 10g, the size of the SGA can also be controlled by the SGA_TARGET parameter rather tuning the buffer cache and shared pool individually.

Depending on the command, Oracle Database then locks additional memory, subject to the upper limit imposed by SGA_MAX_SIZE, or releases memory for use elsewhere by the operating system. The results of `alter system` actions are shown in the alert log file, typically located in `$ORACLE_HOME/rdbms/log/alert_$ORACLE_SID.log`.

From Oracle Database 11g, the choice between ISM and DISM is based on the following criteria:

- From the 11.2.0.1 release Oracle Database uses DISM if it is available on the system and MEMORY_TARGET or MEMORY_MAX_TARGET is set by the user. Note that, by default, from Oracle Database 11.2.0.1 the installer sets MEMORY_TARGET, thereby invoking DISM. When MEMORY_TARGET is in use, the default is to allocate 60% of the available memory to the SGA, which also means that 60% of memory used by Oracle Database will be locked.
- For earlier releases, from Oracle Database 9i, DISM will be used if it is available on the system, and if the value of the SGA_MAX_SIZE initialization parameter is larger than the size required for all SGA components combined. This enables Oracle Database to lock only the amount of physical memory that is used.
- Oracle Database uses ISM if the entire shared memory segment is in use at startup or if the value of the SGA_MAX_SIZE parameter is equal to or smaller than the size required for all SGA components combined.

Regardless of whether Oracle Database uses ISM or DISM, it can always exchange the memory between dynamically sizable components such as the buffer cache, the shared pool, and the large pool after it starts an instance. Oracle Database can relinquish memory from one dynamic SGA component and allocate it to another component.

The Implications of the MEMORY_TARGET Parameter

MEMORY_TARGET was introduced in Oracle Database 11g to allow automatic memory tuning. The amount of memory used is determined by the SGA_TARGET parameter. By default 40% of physical memory will be used. Further, by default 60% of the memory addressed by MEMORY_TARGET is allocated to the System Global Area (SGA) and the remaining 40% to the Program Global Area (PGA). The SGA/PGA mix can be changed dynamically.

Consequently the portion of SGA_TARGET which is expected to be locked (the SGA component) by default is therefore 60% of 40% of physical memory. That represents 24% of physical memory, just below the 25% that can be allocated without changing `project.max-shm-memory` parameter (or `zone.max-shm-memory` if running in an Oracle Solaris Container).

If memory is dynamically diverted from the PGA to the SGA, the 25% threshold is likely to be exceeded. It is very important to set the `project.max-shm-memory` parameter (or `zone.max-shm-memory`) high enough to ensure that all SGA memory can be locked under all circumstances.

How to Check if ISM and DISM Are In Use

Examining shared memory allocations can be used as a simple way of checking whether ISM or DISM are in use. The following example comes from a server running Oracle:

```
% ipcs -im
IPC status from <system> as of Thu Aug 19 01:09:30 PDT 2010
T     ID      KEY        MODE        OWNER    GROUP  ISMATTCH
Shared Memory:
m     11   0xacea3900 --rw-rw----  dbbench      dba      160
```

The ISMATTCH field shows 160 processes attached to this shared memory segment. ISMATTCH does not distinguish between ISM and DISM – the 160 processes could be using either (note, though, that a single shared memory segment cannot use both ISM and DISM simultaneously).

From Oracle Solaris 10, the best way of determining whether ISM or DISM is in use is to use pmap –xs on an Oracle Database process. An example is shown below using the SMON process (which in this case is process id 12524):

```
% ps -aef | grep ora | grep smon
oracle 12524   1   0 05:40:13 ?           0:25 ora_smon_prod
% pmap –xs 12524 | grep ism
0000000380000000  38010880  38010880  - 38010880 256M rwxsR    [ ism shmid=0xb ]
0000000C90000000    131072    131072  -   131072   4M rwxsR    [ ism shmid=0xb ]
0000000C98000000        16        16  -       16   8K rwxsR    [ ism shmid=0xb ]
```

The output from pmap –xs shows three address ranges associated with shmid=0xb (that is, id=11, since "b" is 11 in hexadecimal): one with 256 Mbyte pages, one with 4 Mbyte pages, and one with 8 Kbyte pages. Each of them is clearly labeled "ism". If DISM had been in use, the output would have shown "dism" instead.

NUMA and Shared Memory

Memory Placement Optimization

Many Sun systems, including the Sun SPARC Enterprise M-Series servers, for example, are designed with Non Uniform Memory Architecture (NUMA) characteristics. Put simply, a processor on a NUMA system does not access all physical memory at the same speed. Memory that is local to the processor – usually because it is located on the same system board – can be reached faster (that is, with lower latency) than memory that is local to a processor located on a different system board.

To allow more effective access to memory on Sun's NUMA platforms, the Oracle Solaris 9 9/02 release introduced a Memory Placement Optimization (MPO) feature. Thanks to an associated set of APIs, these MPO features can be leveraged directly by applications like Oracle Database.

Oracle Solaris MPO is based on the concept of a locality group (lgroup), which contains resources like CPUs and memory that are co-located (on the same system board, for example). A process is able to be assigned to an lgroup, with the result that the process will be scheduled to run on one of the CPUs in that lgroup, and will access memory local to that lgroup. The end result is that a much higher proportion of memory accesses can be carried out locally, and memory latency costs can therefore be minimized.

NUMA and Oracle Database

Since Oracle Solaris 9 and later releases include MPO, Oracle Database's NUMA features can be used with any Sun system designed with NUMA characteristics. Enabling NUMA support for Oracle Database on such platforms can improve database performance.

Oracle Database NUMA support must be enabled manually from 11.2.0.1 since NUMA support is disabled by default. To enable NUMA support with Oracle Server Version 11.2.0.1, the following underscore init.ora parameter needs to be set:

```
_enable_NUMA_support=TRUE
```

Note that earlier Oracle Database releases used the init.ora parameter _enable_NUMA_optimization, which is deprecated from 11.2.0.1.

Once the init.ora parameter _enable_NUMA_support is set to TRUE on NUMA-capable Sun platforms, the alert log of the database instance should show that NUMA support has been enabled and indicate the NUMA configuration that was detected.

It is strongly recommended to evaluate performance before and after enabling NUMA in a test environment before going into production.

NUMA lgroups and the Implications for ISM and DISM

When NUMA is turned on for Oracle Database on Sun systems that support MPO, the following ISM and DISM behavior applies:

- ISM-based shared memory segments will be automatically spread across all lgroups.
- DISM-based shared memory segments will be automatically partitioned into NUMA pools, each with its own default, keep, and recycle pools. Dynamic memory allocation within the SGA will also be lgroup aware.

To take maximum advantage of Oracle Database's NUMA features and optimizations, the SGA needs to be allocated using DISM.

Potential DISM Issues AND remedies

The issues that might arise with DISM are outlined below, along with best practice recommendations.

oradism permissions are not setup correctly

The correct way to install Oracle Database is to use the installer. Unfortunately, though, System administrators sometimes take shortcuts, and with potentially disastrous consequences. Suppose, for example, that an existing installation is copied to another system using `tar`, `cpio` or a similar utility. If the file transfer is not carried out with superuser privileges, the Oracle Database oradism (`ora_dism_<$ORACLE_SID>)` program will not run with the correct permissions.

Oracle Database makes the `oradism` program setuid root, and it is important that any transfer of the Oracle binaries to another system preserves these permissions. Similarly, if binaries are located on another system and mounted on the local system with NFS, it is possible that the mount options required to permit root access have not been set, with the result that the `oradism` setuid permissions will not be honored.

Recommendations:

- Always use the Oracle Database installer to create new installations
- If instance migration is carried out, ensure that it is done with superuser privileges
- If using NFS to mount Oracle Database binaries, ensure that NFS mounts allow root access.

The Oradism process terminates

If the Oracle Database oradism process dies for some reason (which is unlikely unless a system administrator with superuser privileges kills it), all locked memory will be automatically unlocked. Performance will suffer accordingly.

From the Oracle Database 10g release, oradism will be restarted if it dies. Unfortunately, though, the SGA locks will not be reestablished. In this case it is necessary to restart the database instance to ensure that SGA memory is correctly locked.

Recommendations:

- Avoid any situation that might lead to the oradism process being accidentally killed
- If performance is very poor, yet oradism is running, and running with root privileges, check to see if SGA memory is not locked. If SGA memory is not locked, it will be necessary to restart the database. (Refer to “Diagnosing and Troubleshooting DISM Issues” below to determine if SGA memory is not locked.)

SGA memory is only partially locked

As noted in the “Enabling ISM” section above, by default Solaris 10 allows 25% of physical memory to be locked by a single instance. If more SGA memory is required by a single instance, system administrators are able to increase the appropriate limit. It is possible, though, for the DBA to start a database instance with the total size of database buffers, shared pool etc greater than the current upper limit of `project.max-shm-memory` (or `zone.max-shm-memory` if the database is running in an Oracle Solaris Container).

The DBA might also later use an `alter system` command to increase the total size of the buffer pool and shared pool beyond the maximum shared memory limit. In this case Oracle Database will not be able to successfully lock the whole of the requested memory. When Oracle Database uses the portion of the memory that is not locked, significantly degraded performance will result. Note that Oracle Database locks and unlocks in chunks, or memory 'granules', typically 16Mbytes in size for Oracle Database 10g and 64 to 512 Mbytes for Oracle Database 11g.

Diagnosing and Troubleshooting DISM Issues

Symptoms

SGA not locked

Systems with active use of unlocked SGA memory can be identified with the lockstat utility. The lockstat utility ships with Oracle Solaris 8 and later releases.

The lockstat utility shows mutex contention. As the superuser (root user), look for mutex contention in the segspt_softunlock or spt_anon_getpages functions. Two examples are included below showing the first few lines of a lockstat profile from a system running an active Oracle Database instance with unlocked SGA memory. The following command can be used to generate these profiles.

```
serv1# /usr/sbin/lockstat -A -n 200000 sleep 10
```

Here are the two sample profiles:

```
Count indv cuml rcnt     spin Lock                   Caller
---------------------------------------------------------------------------
 4896  21%  21% 1.00    27532 0x3001314ec88          spt_anon_getpages+0x54
   81   0%  22% 1.00       11 0x300001a1e98          sc_flush+0x1c4
   78   0%  22% 1.00        2 pse_mutex+0xb0         page_unlock+0x1c
   77   0%  22% 1.00        2 pse_mutex+0x368        page_unlock+0x1c
Adaptive mutex spin: 561050 events in 10.017 seconds (56011 events/sec)
Count indv cuml rcnt     spin Lock                   Caller
---------------------------------------------------------------------------
549729 98%  98% 1.00      146 0x300085a2580          segspt_softunlock+0xc4
  834   0%  98% 1.00    10685 0x300085a2578          spt_anon_getpages+0x54
  715   0%  98% 1.00       12 0x300085a2580          spt_anon_getpages+0x64
```

Permissions problems

Examine the alert log for each instance. The following entry indicates that Oracle was unable to start the oradism process with superuser privileges:

```
Wed Jul 16 14:03:39 2003
WARNING: -------------------------------
WARNING: oradism not set up correctly.
  Dynamic ISM can not be locked. Please
  setup oradism, or unset sga_max_size.
  [diagnostic 0, 5, 0]
----------------------------------------
```

Performance could be very significantly degraded for this instance, since SGA memory is not locked.

Check if the setuid bit has been set on the oradism program. For example:

```
serv1% ls -l $ORACLE_HOME/bin/oradism
-rwsr-sr-x   1 root     dba         9280 Apr  9  2002 /export/home/oracle/bin/oradism*
```

In the above example the oradism program is owned by root, and the setuid bit is set (as indicated by the first s in rws). The program should therefore be able to start with appropriate permissions.

Oradism dies

The only likely reason why oradism might die is if a system or database administrator accidentally kills it. Check for the presence of the oradism process with the `ps` program.

Unfortunately, in the unlikely event that this process dies, in Oracle Database 9i it will not be noted in the alert log file until the next time alter system is used to grow or shrink memory.

Oracle Database10g and later releases have the ability to restart oradism if it dies. Unfortunately, though, all SGA memory locks will have been lost, and these missing locks will not be replaced until the database is restarted. Performance could be extremely poor in the meantime. When diagnosing the problem, be aware that it is therefore possible to have a running oradism with root privileges, but to have no locks (and therefore very poor performance).

SGA memory only partially locked

From Oracle Database 10g, a new table (x$ksmge) can be examined to determine the state of locks for each granule of SGA memory (each granule is typically 16Mbytes in size for Oracle Database 10g and 64 to 512 Mbytes for Oracle Database 11g).

How to fix it

Permissions problems

The simplest way of ensuring appropriate permissions is to make the `oradism` program setuid root, obviating the need for RBAC. This can be achieved as follows:

```
serv1% chown root $ORACLE_HOME/bin/oradism
serv1% chmod 4755 $ORACLE_HOME/bin/oradism
serv1% ls -l $ORACLE_HOME/bin/oradism
-rwsr-xr-x   1 root  dba  9280 Apr 9 2002 /export/home/oracle/bin/oradism*
```

Oradism dies

If the oradism process dies in Oracle Database 10g and later releases, Oracle Database is able restart it. As with Oracle Database 9i, though, it will be necessary to shutdown and restart the instance to achieve optimal performance.

SGA memory only partially locked

The main solution to this problem is to ensure that it doesn't happen. Avoid attempts to lock more memory than the user has permission to lock, or more memory than the system has available.

General Guidelines

The following guidelines are offered in conclusion:

- Do not use DISM on SPARC-based systems unless you need to dynamically resize the SGA.
- Avoid DISM on x86-based systems.
- If you don't need to resize the SGA dynamically, you probably don't need to set SGA_MAX_SIZE. If you use SGA_MAX_SIZE, though, check that oradism is working correctly as described above.
- If using DISM, ensure that you are using an appropriate release of Oracle Solaris (refer to the next section of this document).
- Ensure that the oradism binary is owned by root, and setuid root.
- Ensure that the oracle user has a large enough setting for the `project.max-shm-memory` parameter (or `zone.max-shm-memory` if running in an Oracle Solaris Container), so that **all** memory used by the SGA can be locked.
- Avoid using DISM with releases of Oracle Solaris prior to Oracle Solaris 9 Update 2. On Oracle Solaris 9 from Update 2, the performance of DISM is equivalent to ISM.

Conclusion

DISM can offer significant benefits to users of Oracle Database on the Oracle Solaris Operating System by allowing the SGA to be dynamically resized, both to boost performance by adding more memory to the buffer cache or shared pool, and also to support dynamic reconfiguration of the server by addition or removal of physical memory. Provided DISM is correctly installed and the oracle user is able to lock the necessary amount of memory, DISM will perform as expected. Should performance degrade due to configuration issues, it should be possible to readily diagnose the cause of the problems and apply appropriate remedies.

Appendix A Brief Technical Overview of the Implementation of ISM and DISM in Oracle Solaris

Oracle Database took advantage of DISM in Oracle Database 9i and subsequent releases to implement dynamic SGA resizing. Dynamic SGA resizing allows a database administrator to respond to changing needs by increasing or decreasing SGA memory without a database reboot. The DISM feature provides this capability on Oracle Solaris.

Since DISM requires the application to lock memory, and since memory locking can only be carried out by applications with superuser privileges, Oracle Database implemented a daemon that runs with root privileges. Since Oracle does not normally run with superuser privileges, the Role Based Access

Control (RBAC) features introduced in Solaris 8 were used to provide that access in the first release of Oracle Database 9i. RBAC allows a nominated binary to run with an effective uid of root. During the Oracle Database install procedure, the installer is asked to run a script as root; this script established the necessary RBAC permissions. In particular, entries were added to the `/etc/user_attr` and /etc/security/exec_attr files. Later Oracle Database releases simply made the $ORACLE_HOME/bin/oradism binary setuid root to achieve the same end.

The `$ORACLE_HOME/bin/oradism` binary implemented the root daemon. Look for it in ps with the description `ora_dism_$ORACLE_SID`. For example:

```
serv1% ps -aef | grep dism

    root   747     1   0 13:57:26 ?          19:42 ora_dism_custdb

 oradba1 18456 18391   0 23:37:08 pts/6       0:00 grep dism
```

Oracle Database 9i introduced a new init.ora parameter, SGA_MAX_SIZE, to activate DISM. This variable establishes the maximum size to which the SGA can grow; it can only be modified statically (in other words an Oracle Database reboot is required before any change to SGA_MAX_SIZE takes effect). Oracle Database will use DISM instead of ISM if SGA_MAX_SIZE is set larger than the total of the database buffers (in particular, DB_CACHE_SIZE dynamic SGA resizing is not supported with the older db_block_buffers parameter), the shared pool, the redo buffers, the large pool, the Java pool, and the SGA fixed size (representing Oracle's internal requirements).

Once DISM has been invoked, Oracle Database automatically locks an amount of memory determined by the total of the elements described above (database cache, shared pool, etc). Subsequently, the DBA can alter the size of the database buffers (DB_CACHE_SIZE) and the shared pool (SHARED_POOL_SIZE) with Oracle's alter system command. For example:

```
alter system set db_cache_size = <new size in bytes>
```

Depending on the command, Oracle Database then locks additional memory, subject to the upper limit imposed by SGA_MAX_SIZE, or releases memory for use elsewhere by the operating system. The results of alter system actions are shown in the alert log file, typically located in $ORACLE_HOME/rdbms/log/alert_$ORACLE_SID.log. An example is shown below:

```
CKPT: Begin resize of buffer pool 3 (DEFAULT for block size 2048)

CKPT: Current size = 37904 MB, Target size = 3760 MB

Fri Jun 21 21:29:50 2002

Completed checkpoint up to RBA [0x7e.2.10], SCN: 0x0000.25c3f387

CKPT: Resize completed for buffer pool DEFAULT for blocksize 2048

Fri Jun 21 21:29:54 2002

ALTER SYSTEM SET db_cache_size='3932160000' SCOPE=MEMORY;
```

Appendix B Software Release Details

Oracle Solaris 9

DISM should only be used on Solaris 9 Update 2 or later. Earlier releases of Solaris 9 do not contain all necessary patches.

Oracle Solaris 8

For correct operation of DISM, Solaris Patch 117000-05 must be installed. The minimum additional requirement will be Solaris 8 Update 3 (1/01) with Patch 108528-16 (or later). It is not recommended, however, to use DISM with Oracle Solaris 8.

Oracle Database

Oracle Database supports DISM from the Oracle Database 9i release. For the first release only (Oracle9.0.1.0), an Oracle patch is required.

References

Dynamic Reconfiguration and Oracle 9i Dynamically Resizable SGA - http://www.sun.com/blueprints/0104/817-5209.pdf

ORACLE®

Oracle is committed to developing practices and products that help protect the environment

Dynamic SGA Tuning of Oracle Database on
Oracle Solaris with DISM
September 2010
Author: Allan Packer

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

SOFTWARE. HARDWARE. COMPLETE.