Tuning HP Tru64 UNIX V5.1A and V5.1B for Oracle Environments

Introduction 3

Operating System Versions and Service PAKs 4
  Prior Version Support 4
  Patches Applicable to Systems Running Oracle 4
    Tru64 UNIX/TruCluster Server V5.1B-2 with PK4 Service PAK 4
    Tru64 UNIX/TruCluster Server V5.1B-3 with PK5 Service PAK 4
    Tru64 UNIX/TruCluster Server V5.1A with PK6 Service PAK 4
  Location of Patches and Service PAKs 4

Key Tunable Parameters 5
  VM Subsystem 5
    new_wire_method 5
    ubc_maxpercent 5
    ubc_minpercent 6
    ubc_borrowpercent 6
    vm_ubcseqpercent 6
    vm_ubcseqstartpercent 6
    vm_ubcdirtypercent 6
    vm_swap_eager 6
    ubc_overflow 7
    vm_overflow 7
  Shared Memory Allocation and Granularity Hints 7
  Generic Subsystem 9
    sched_distance 9
    cpus_in_rad 9
  AdvFS Subsystem 9
    AdvfsSyncMmapPages 9
    Obsolete AdvFS system parameters 10
  VFS Subsystem 10
    fifo_do_adaptive 10
    smoothsync 10
    bufcache and bufpages 11
  IPC Subsystem 11
    ssm_threshold 11

shm_max.......... 11
shm_min.......... 11
shm_mni.......... 12
shm_seg.......... 12
INET Subsystem.......... 12
udp_sendspace and tcp_sendspace.......... 12
udp_recvspace and tcp_recvspace.......... 12
ipport_userreserved.......... 12
PROC Subsystem.......... 13
per_proc_stack_size and max_per_proc_stack_size.......... 13
per_proc_data_size and max_per_proc_data_size.......... 13
per_proc_address_space and max_per_proc_address_space.......... 13
max_proc_per_user.......... 14
max_threads_per_user.......... 14
maxusers.......... 14
RT Subsystem.......... 14
RDG Subsystem.......... 14
max_objs.......... 14
msg_size.......... 14
max_async_req.......... 15
max_sessions.......... 15
rdg_max_auto_msg_wires.......... 15
RM Subsystem.......... 15
Improving the Performance of gettimeofday() in Oracle.......... 15
Selecting IPC Communication Protocols for Oracle RAC.......... 15
Selecting NUMA Support in Oracle.......... 16
For more information.......... 18
Acknowledgments.......... 19
Revision Information.......... 19

Introduction

This document provides some generic guidelines on how to set key tunable parameters applicable to Oracle® environments running HP Tru64 UNIX V5.1B. In addition, specific patches are recommended for use in Oracle environments.

Where appropriate, there are also recommendations for Tru64 UNIX V5.1A; since this release is still supported by HP under a Prior Version Support with Sustaining Engineering (PVS-SE) support agreement. Although Tru64 UNIX V4.0F and V4.0G are also still supported by HP under the special PVS-SE support agreement, no recommendations are made for this release. See the Prior Version Support section for more information on PVS-SE.

In the past there were several resources and documents available that provided recommendations on how to set certain tunable parameters in `/etc/sysconfigtab`, but there has always been a fair amount of confusion regarding which guidelines to follow.

For the purposes of this document, the assumption is that the Oracle version in use is 8.1.7 or later; this is important as, starting with version 8.1.7, Oracle uses the AdvFS direct I/O capabilities to bypass the unified buffer cache (UBC) of the file system layers.

Unless otherwise noted, the recommendations in this document are applicable to all systems used as an Oracle database server.

Unless stated otherwise, the recommendations for Oracle RAC described here will also apply to Oracle 8i OPS.

Operating System Versions and Service PAKs

The current supported release of HP Tru64 UNIX and TruCluster Server is V5.1B. This is the last release of Tru64 UNIX and TruCluster Server. HP will continue to support these products until at least 2011. As of March 2006, the supported patch kits for HP Tru64 UNIX V5.1B are V5.1B-2 (PK4) Service PAK and V5.1B-3 (PK5) Service PAK; these are the only supported and recommended Service PAKs to use for Oracle installations running HP Tru64 UNIX V5.1B.

Prior Version Support

The HP Tru64 UNIX V5.1A release is currently under Prior Version Support with Sustaining Engineering (PVS-SE). This means that this release will continue to be supported until 30th June 2007, provided that PVS-SE support is added to an HP Customer Services Support Contract.

For more information about PVS-SE, go to:

http://www.hp.com/hps/os/os_pvs_amap.html

Patches Applicable to Systems Running Oracle

The following list describes the patches related to systems running Oracle that HP recommends for the currently supported versions of Tru64 UNIX. Other patches, not specifically related to Oracle, may be relevant for the version of Tru64 UNIX that you are using. All of the patches are available for download from the location described in the Location of Patches and Service PAKs section.

Tru64 UNIX/TruCluster Server V5.1B-2 with PK4 Service PAK

T64KIT0023584-V51BB25-E-20040913

Tru64 UNIX/TruCluster Server V5.1B-3 with PK5 Service PAK

Fixes for `vm:vm_overflow` and `generic:cpus_in_rad` as updated in T64KIT1000910-V51BB26-E-20060928

Tru64 UNIX/TruCluster Server V5.1A with PK6 Service PAK

PAKT64KIT0023583-V51AB24-E-20040913

Location of Patches and Service PAKs

All of the Tru64 UNIX service PAKs and patches are available for download from the IT Resource Center (ITRC) at the following URL:

http://www2.itrc.hp.com/service/patch/search.do?BC=patch.breadcrumb.main|&pageContextName=tru::

Note that you may have to register and log in to use this site if you have not already done so.

Key Tunable Parameters

This section describes the various kernel parameter changes that HP recommends for use in an Oracle environment.

HP recommends that only those parameters that are changed from the default value be included in the `/etc/sysconfigtab` file. HP further recommends that changes to the `/etc/sysconfigtab` file are made using the `sysconfigdb` command.

When the version of the release to which a parameter applies is not explicitly mentioned, then the recommendation is valid for both Tru64 UNIX V5.1A and Tru64 UNIX V5.1B systems.

VM Subsystem

The following sections describe the tunable parameters to modify in the VM subsystem section of `/etc/sysconfigtab`.

new_wire_method

Since the introduction of the light-weight wiring (`vm:new_wire_method=1`) algorithm as a means to wire pages in memory, there have been several issues that have prevented its successful use. Some examples of such problems were:

- An interaction between asynchronous I/O (AIO) and light-weight wiring resulting in high system times
- The combination of granularity hints memory (`gh_regions > 0` or `rad_gh_regions > 0`) with light-weight wiring, preventing the successful start-up of Oracle

Since there was no negative impact to the use of the regular wiring algorithm, the recommendation has, for many years, been to disable light-weight wiring; that is, `new_wire_method = 0`.

With the introduction of Tru64 UNIX V5.1B, all known problems with light-weight wiring have been fixed and light-weight wiring has become the preferred page-wiring algorithm for HP. In addition, Oracle has also qualified Oracle 9i (9.2) and later releases with light weight wiring.

In the Tru64 UNIX V5.1B PK3 Service PAK, additional fixes were introduced to the AIO mechanism that may cause asynchronous I/Os to be slow in combination with regular style page wiring. Therefore, Oracle now requires that systems running Tru64 UNIX V5.1B PK3 Service PAK or later use light-weight page wiring; that is, `new_wire_method = 1`.

In summary, HP recommends the following values for `new_wire_method`:

- `new_wire_method = 1` when running Oracle on Tru64 UNIX V5.1B PK3 Service PAK or later
- `new_wire_method = 0` when running Oracle on Tru64 UNIX V5.1B PK2 or earlier (including previous releases of Tru64 UNIX)

Note: In Tru64 UNIX V5.1B-3 (PK5) and later, the `vm:new_wire_method` kernel parameter is deprecated. Tuning of this parameter is no longer possible and light-weight wiring is always enabled.

ubc_maxpercent

As of Oracle 8.1.7, AdvFS direct I/O was enabled by default, resulting in there no longer being a need to artificially restrict the UBC. Before the availability of direct I/O, HP recommended that the amount of memory used by the UBC should be reduced to prevent double-caching of Oracle data by the file system. Now that the AdvFS file system is fully integrated with the regular UBC mechanisms, double-caching is no longer an issue for Tru64 UNIX and Oracle based database environments when using the direct I/O settings recommended by Oracle. However, setting this tunable parameter to a low value may cause contention in the kernel and have a negative impact on performance.

The recommended value for `ubc_maxpercent` should be at least 70%, but less than 100%, and should not be set to a value smaller than 35%.

ubc_minpercent

The value for this parameter represents the minimum amount of memory available for the UBC. When this parameter is set to a very low value, it may cause severe performance problems on buffered file system operations such as file copies. This is especially true on NUMA systems where the UBC is distributed over the Resource Affinity Domains (RAD)s.

The recommended value for `ubc_minpercent` is to leave it at its default value of 10%.

ubc_borrowpercent

The value for this parameter represents a percentage of memory above which the UBC is only borrowing memory from the virtual memory subsystem. Paging does not occur until the UBC has returned all its borrowed pages. The default value of 20% is usually a good fit for most systems.

vm_ubcseqpercent

This parameter controls the maximum percentage of UBC memory that can be used to cache a single file.

The default value of 10% is usually a good fit for most systems.

vm_ubcseqstartpercent

This parameter is a percentage of the UBC in terms of its current size that determines the threshold at which the UBC starts to check whether the percentage of UBC pages cached for each file object has been exceeded (as specified by `vm_ubcseqpercent`). If the cached page percentage for any file exceeds the value of `vm_ubcseqpercent`, the UBC returns that file's UBC Least Recently Used (LRU) pages to virtual memory.

Prior to HP Tru64 UNIX V5.x, the `vm_ubcseqpercent` and `vm_ubcseqstartpercent` parameters were a percentage of total memory.

The recommended value is to keep `vm_ubcseqstartpercent` at the default of 50%.

vm_ubcdirtypercent

This parameter specifies the percentage of pages that can be dirty (modified) before the UBC must flush them to disk. The default value of 10% usually works very well for most systems. However, on systems with lots of file system/UBC activity, increasing the value for this parameter may increase the ability to re-read from the file system cache.

The recommended value for `vm_ubcdirtypercent` is to keep the default value of 10%, but on systems that could benefit from keeping file system pages in the UBC you could increase it to 90%.

vm_swap_eager

This parameter controls how the system will use the available swap space. For a database server environment the swap allocation mode could be changed from eager (the default) to deferred (lazy) swap mode.

The minimal size of swap space that should be allocated is equal to the size of physical memory plus 10%.

Setting `vm_swap_eager = 1` (the default) indicates that the system is in `eager swap' allocation mode. On systems with an undetermined workload, you may want to allocate a swap space with a size between two and three times the size of physical memory. Use eager swap allocation mode for highly reliable systems that overcommit memory resources.

Setting `vm_swap_eager = 0` will cause the system to enter a deferred swap allocation mode. Use lazy swap allocation mode for any system that does not overcommit memory resources. If `vm_swap_eager` has been set to 0 and the system runs out of allocatable swap space, any process that tries to allocate additional swap space will be terminated by the kernel. There are no mechanisms available to protect processes from being terminated in this situation.

ubc_overflow

A new kernel parameter was introduced in Tru64 UNIX V5.1B PK3 Service PAK that allows UBC memory to be allocated across RADs on certain types of systems (EV7 based systems such as the AlphaServer ES47, ES80, and GS1280).

Unfortunately, this parameter was incorrectly initialized on some EV68 based systems such as the AlphaServer GS80, GS160, and GS320. For these types of systems running Tru64 UNIX V5.1B-1 (PK3) Service PAK only, you can manually correct this parameter by setting `ubc_overflow = 0`. Initialization of the parameter is correct with Tru64 UNIX V5.1B-2 (PK4) and later.

vm_overflow

On NUMA systems (such as the AlphaServer ES47, ES80, GS160, GS320, and GS1280 systems), HP recommends the setting `vm_overflow = 1`. This is only possible with Tru64 UNIX V5.1B-3 (PK5 Service PAK) or later and should only be done when the patches described in the Tru64 UNIX/TruCluster Server V5.1B-3 with PK5 Service PAK section are applied.

With this patch enabled, the system can more easily allocate memory in other RADs when the local RAD is low on memory. This prevents systems from paging and/or processes stalled for minutes when allocated more VM data structures, even though sufficient free memory is available in other RADs.

Shared Memory Allocation and Granularity Hints

HP Tru64 UNIX supports several different options to allocate shared memory for Oracle, including:

- Default shared memory allocation
  The traditional option is to use Segmented Shared Memory (SSM), which is controlled with the `ssm_threshold` tunable parameter in the `ipc` subsystem. The default for Tru64 UNIX V5.x is to use SSM type shared memory (with `ipc:ssm_threshold` set to its default value of 8838608). HP recommends this type of shared memory allocation for all systems, except those that need to get the maximum performance out of the system.

- Big pages
  Another option available is to use granularity hints memory. This can either be done by statically reserving an area of memory specifically for this purpose at boot time, or by using a new option in Tru64 UNIX V5.1B called big pages. Although the big pages option offers the best performance in addition to a significant management advantage over statically allocated granularity hints memory, Oracle has not been certified for use on a system with big pages enabled. As a result, Oracle does not support the use of big pages.

  The recommended value for use with Oracle is to leave the parameter `vm_bigpg_enabled` at its default value of 0, which is to not use big pages".

  HP supports the use of big pages in general, provided that segmentation is disabled, that is, `vm:vm_bigpg_seg=0`.

- Statically allocated granularity hints memory
  Statically allocated granularity hints memory is reserved at boot time and cannot be used for any other purpose, nor can it be returned to the system or reclaimed when not in use. It will be solely

used by the database for its System Global Area (SGA) or any other application using `shmget()` or `nshmget()` to allocate shared memory.

Using granularity hints memory offers the best performance and is ideally suited for large GS class AlphaServer systems, especially those with applications that have many processes attached to a large shared memory region (such as large Oracle databases). On smaller systems the overall performance gain using granularity hints memory averages 7%; on those systems this may not be an interesting option, due to its complexity in implementation and management and its relatively small performance benefit. HP recommends this type of shared memory (statically allocated granularity hints memory) for the most demanding systems using large shared memory segments.

It is important to ensure that all shared memory segments are allocated in the memory area reserved for granularity hints memory. When the shared memory segments cannot be allocated in the reserved memory, then messages will be displayed on the console and through `syslog` (for example, `/var/adm/messages`). System performance will be negatively impacted, see the `ssm_threshold` parameter for more information. You can prevent overallocation of this memory area by setting the parameter `gh_fail_if_no_mem = 1`.

`ipc:ssm_threshold` must be disabled by setting `ssm_threshold = 0` when using statically allocated granularity hints memory (`rad_gh_regions` or `gh_chunks`) is used.

On those systems that are not NUMA aware, that is, all systems that only have a single RAD, you can use the `gh_chunks` attribute to enable granularity hints memory. The `gh_chunks` parameter is specified in multiples of the largest possible granularity hints page, that is, it specifies the number of 4-MB pages to allocate for RAD 0. Both `rad_gh_regions` and `gh_chunks` are equally effective. This document only describes `rad_gh_regions`; however, you can use `gh_chunks` on single RAD systems as an alternative. To avoid confusion, HP recommends using the `rad_gh_regions` parameter rather than the `gh_chunks` parameter.

To view the amount of granularity hints memory configured on a system, use the `vmstat –P` command.

On the EV6x NUMA systems (AlphaServer GS80, GS160, and GS320) a RAD corresponds to a Quad Building Block (QBB). On EV7 CPU based systems (AlphaServer ES47, ES80, and GS1280) a RAD typically corresponds to a single CPU (unless it is changed through the `generic:cpus_in_rad` parameter). The amount of memory per RAD to statically allocate for granularity hints memory is specified in 1-MB increments through the corresponding `rad_gh_regions` parameter.

The naming convention for this tunable parameter is to specify `rad_gh_regions[<`*`RAD number`*`>]`. For example, to allocate 2 GB in RAD 0 you would specify 2048 for `rad_gh_regions[0]`, that is, `rad_gh_regions[0] = 2048`.

Take your planned/projected Oracle SGA size and divide by the number of RADs your system has configured. For each `rad_gh_regions[`*`x`*`]` (where *x* represents the RAD identifier, from 0-63, depending on the number of RADs in your system), you would specify the required value in megabytes (MB). Note that the number of the RAD is determined by the physical location in the system, that is, unpopulated RADs are still assigned a number.

The sum of `rad_gh_regions[*]` should be set to the size of the Oracle SGA; however, you may consider allocating a larger value for `rad_gh_regions` in order to be able to resize the Oracle SGA without having to reboot the system; `rad_gh_regions` is not a dynamic system tunable parameter so any changes will require a system reboot). Setting `rad_gh_regions` to a larger value than initially required or calculated can provide the benefit of being able to allocate a larger SGA without having to reboot the system.

Try to allocate shared memory in striped mode in order to distribute memory across all available RADs. Changing to a sequential allocation policy may adversely affect performance as this may cause hot spots in individual RADs. When NUMA mode is enabled in Oracle, it will itself select the correct attributes for the allocation policy.

Generic Subsystem

Modify the following tunable parameters in the respective section(s) of `/etc/sysconfigtab`.

sched_distance

On EV7 based systems such as the AlphaServer ES47, ES80, and GS1280, HP recommends allowing the system to have more freedom when scheduling processes. Setting the kernel parameter `sched_distance = 3` allows a process to be scheduled on a CPU that is up to two hops away from its home RAD.

cpus_in_rad

For Tru64 UNIX V5.1B-3 with PK5 Service PAK a new parameter is available that lets a system administrator change the software NUMA boundaries on EV7 based systems (AlphaServer ES47, ES80, or GS1280). If this parameter is changed from its default value, you must apply the patches described in the Tru64 UNIX/TruCluster Server V5.1B-3 with PK5 Service PAK section.

HP recommends that you keep this parameter at its default value of 0.

For more information about configuring the `cpus_in_rad` parameter, refer to the Configuring Tru64 UNIX for Large Memory Applications in a NUMA Environment white paper:

http://h30097.www3.hp.com/pdf/numa-2.pdf

AdvFS Subsystem

Modify the following tunable parameters in the `advfs` subsystem section of `/etc/sysconfigtab`.

AdvfsSyncMmapPages

This parameter controls how AdvFS flushes dirty `mmap()`'d pages using the `sync()` system call. Since most applications that use `mmap()` to map pages/files into memory are using their own synchronization through the `fsync()` call, there is rarely a need for AdvFS to perform the same operation again.

Setting `AdvfsSyncMmapPages = 0` will prevent AdvFS from trying to flush pages of files that have been `mmap()`'d and will also avoid AdvFS trying to flush pages that should be left memory resident; this is often seen in Oracle 9i Application Server environments.

Setting this parameter to the default value of 1 will cause memory mapped pages to be written asynchronously to disk during a `sync()` system call.

The recommended value for `AdvfsSyncMmapPages` is 0.

Obsolete AdvFS system parameters

The following AdvFS tunable parameters are obsolete since the integration of AdvFS with the regular UBC mechanisms in Tru64 UNIX V5.1:

- `AdvfsCacheMaxPercent`
- `AdvfsCacheHashSize`
- `AdvfsCleanupPercent`
- `AdvfsMaxFreeAccessPercent`
- `AdvfsMinFreeAccess`

Those parameters are no longer used to tune the AdvFS buffer cache and should be removed from `/etc/sysconfigtab`.

Use the UBC parameters `ubc_maxpercent` and `ubc_minpercent` to tune the file system cache.

VFS Subsystem

Modify the following tunable parameters in the `vfs` subsystem section of `/etc/sysconfigtab`.

fifo_do_adaptive

This is one of the tunable parameters where the default value may not have been chosen perfectly if the system is running as a database server.

The default setting of the `fifo_do_adaptive` parameter enables an alternate algorithm in the FIFO routines (used for pipes). The original intent was to come up with an optimal working set size and then attempt to perform fewer data transfer operations (but of a larger size).

This works reasonably well for applications that perform data transfers of a uniform or near-uniform size. This does not work so well for some applications that perform data transfers of a random size (for example, those applications that started out performing transfers so that the FIFO code determined an optimal transfer size), and works poorly for some applications in which the peer processes operate in sync (meaning process A transfers data to process B and then waits for process B's response).

By disabling the `fifo_do_adaptive` parameter, the performance for some applications will degrade and improve for others. The performance change depends on how the pipes get used.

For Oracle environments, HP recommends the `fifo_do_adaptive = 0` setting.

smoothsync

The `smoothsync_age` attribute is currently configured at the default setting of 30 seconds. The `smoothsync_age` attribute is used to determine how a buffered dirty page is allowed to age before it is written to disk. With Oracle, buffered I/O is utilized when accessing redo logs and archive logs. Write requests to the redo and archive logs flow into UBC where they are eventually transferred, usually asynchronously, to physical storage by some synchronization mechanism (`smoothsync`, `fsync`, and so on). `smoothsync` provides the behavior that each written page will age for `smoothsync_age` seconds in UBC before being flushed to disk.

Reducing the `smoothsync_age` time allows dirty buffer writes to occur sooner after they arrive, which has the effect of keeping the I/O pipe more evenly loaded at all times. The downside is that frequently updated buffers may be written multiple times; however, this is unlikely to happen on an Oracle database server. A reasonable starting point for `smoothsync_age` on most modern systems is 4 (seconds). Faster CPUs and networks and slower storage systems all call for smaller values of `smoothsync_age`.

To change the time of `smoothsync_age` attribute, make the following change in `/etc/inittab`:

```
smsync:23:wait:/sbin/sysconfig -r vfs
smoothsync_age=4 > /dev/null 2>&1
```

bufcache and bufpages

If none of the file systems are UFS based, then some memory can be freed up by setting the following parameters. On systems with less than 2 GB of memory, set `vfs:bufcache=1`. On systems with more than 2 GB of memory, set `vfs:bufpages=2048` and remove any setting of `vfs:bufcache` from `/etc/sysconfigtab`.

IPC Subsystem

Modify the following tunable parameters in the `ipc` subsystem section of `/etc/sysconfigtab`.

ssm_threshold

This tunable parameter controls which type of shared memory implementation is used. The default in Tru64 UNIX V5.x is to use Segmented Shared Memory (SSM) type shared memory. The only two reasons to change this parameter are the use of statically allocated granularity hints memory or the use of an application on the same system that prevents the use of SSM type shared memory (for example, Sybase); in either case `ssm_threshold` should be set to 0. Note that not using SSM type shared memory while not using granularity hints memory (which includes those cases where shared memory allocations exceed the amount of memory reserved for granularity hints) can have a severe negative performance impact, especially when many processes attach and detach from large shared memory segments.

HP recommends leaving the value of `ssm_threshold` at its default value of 8388608, unless statically allocated granularity hints memory (`rad_gh_regions` or `gh_chunks`) is used; in which case `ssm_threshold` should be set to 0.

shm_max

This tunable parameter controls the maximum size for a single shared memory segment. Oracle will automatically concatenate multiple shared memory regions if the SGA is larger then the value configured for `shm_max`. As of Tru65 UNIX V5.1, the size for a single shared memory segment could be larger than 2 GB. However, other applications using shared memory on the same system may have problems with shared memory segments larger than 2GB. In order to avoid compatibility issues HP recommends setting the maximum size for an individual shared memory segment to 2 GB.

The recommended (maximum) value for `shm_max` is 2 GB minus 8 MB, that is:

2147483648 - 8388608 = 0x7f800000 or 2139095040

If only Oracle is being used on the system, you can increase the size of `shm_max` to 4 GB minus 16 MB, that is:

0xff000000 or 4278190080

shm_min

This tunable parameter specifies the smallest amount of memory allocated for a shared memory request. The Oracle 9i (9.0.1) System Administration guide incorrectly states a value of 1024 should be specified; the correct value is the default value of 1.

The error in the Oracle documentation will be corrected in a future release and the typo corrected (the tunable parameter that should be changed is `shm_mni`).

The recommended value for `shm_min` is the default of 1.

shm_mni

This tunable parameter specifies the maximum number of shared memory regions that can be used on the system at one time. When NUMA is enabled in Oracle, Oracle will allocate several shared memory segments for each RAD. Oracle will allocate these segments for each instance running on the system. This value should be large enough to cover all instances. This parameter is a limit, which does not have performance implications.

The recommended value for `shm_mni` is 256.

shm_seg

This tunable parameter specifies the maximum number of System V shared memory regions that can be attached to a single process at one time. The same guidelines as for `shm_mni` (see the previous parameter) apply, but for one Oracle instance only.

The recommended value for `shm_seg` is 128.

INET Subsystem

Set the tunable parameters discussed in this section in the `inet` subsystem section of `/etc/sysconfigtab`.

Additionally, follow the Best Practice for tuning Internet servers tips available from:

http://h30097.www3.hp.com/docs/best_practices/BP_INTUNING/TITLE.HTM

You can find recommendations specific to Gigabit Ethernet (GbE) performance in this Best Practice:

http://h30097.www3.hp.com/docs/best_practices/BP_GIGABIT/TITLE.HTM

udp_sendspace and tcp_sendspace

These tunable parameters specify the send buffer size for UDP and TCP requests. In environments using Gigabit Ethernet (GbE) or very heavy network activity, you may want to set the value for these tunable parameters to an even higher value than the system defaults (61440-for `tcp_sendspace`, and -9216 for `udp_sendspace`).

These parameters should, at the very minimum, be set to twice the size of the largest Oracle data block in use for Oracle RAC systems. It is safe to increase these parameters to 128 K, 256 K, or even 512 K when data is transferred between applications that use large socket buffers.

The recommended value for `udp_sendspace` and `tcp_sendspace` is 65536 (64 KB), unless the application requires an even larger value.

udp_recvspace and tcp_recvspace

These tunable parameters specify the receive buffer size for UDP and TCP requests. In environments using Gigabit Ethernet (GbE), or very heavy network activity, you may want to set the value for these tunable parameters to an even higher default value than the suggest defaults.

These parameters should, at the very minimum, be set to twice the size of the largest Oracle data block in use for Oracle RAC systems. It safe to increase these parameters to 128 K, 256 K, or even 512 K when data is transferred between applications that use large socket buffers.

The recommended value for `udp_recvspace` and `tcp_recvspace` is 65536 (64 KB, unless the application requires an even larger value.

ipport_userreserved

This tunable parameter specifies the number of times a system can simultaneously make outgoing connections to other systems. The number of outgoing ports is the value of the `ipport_userreserved` attribute minus the value of the `ipport_userreserved_min` attribute.

The default value of the attribute is 5000; therefore, the maximum number of outgoing connections is, by default, 3976 (5000 – 1024).

On systems with a large number of connections, this value should be raised.

On TruCluster Server configurations that do not use Memory Channel as the cluster interconnect, this value should be raised to 65536.

The recommended value for `ipport_userreserved` on large-scale Oracle installations should be the maximum value of 65535.

PROC Subsystem

The following tunable parameters should be set in the `proc` subsystem section of `/etc/sysconfigtab`. Most of the parameters described in the following sections are limits that specify the amount of address space resources that processes are allowed to use. All of those parameter's names take the form of `per_proc` and `max_per_proc`. By default, all of these processes can grow their address space resources up to the limit specified by the `per_proc` parameter. It is possible to allow processes resource requirements to grow up to the size specified in the `max_per_proc` parameter by using the `limit` or `ulimit` shell commands. In order to allow the maximum resources for Oracle processes, you can specify the required command in the login script for the Oracle user. Alternatively, you can make the `per_proc` and `max_per_proc` parameter settings identical. This can safely be done on database servers where processes will never use an unexpected amount of address space resources.

per_proc_stack_size and max_per_proc_stack_size

This tunable parameter specifies the per process stack size in bytes. The default values of 8 MB and 32 MB for this tunable are usually large enough for most Oracle environments. However, in very large installations and data warehouse-type environments the value should be increased.

Depending on the Oracle environment, the maximum stack size can be increased to a maximum of 896 MB. This limit is due to the fact that Oracle has fixed the PGA and SGA to start at 0x38000000 and 0x58000000, respectively, for performance (this only applies to the Tru64 UNIX port). If you use a value larger than 896 MB for this parameter, Oracle could corrupt the fixed SGA and PGA.

The recommended value for `per_proc_stack_size` is 33554432 (32 MB).

The recommended value for `max_per_proc_stack_size` is 536870912 (512 MB).

per_proc_data_size and max_per_proc_data_size

This tunable parameter specifies the default per process data size in bytes. The following value will set this tunable parameter to the amount of physical memory installed on the system. It is possible to raise the value to a value larger than actual memory available. However, this would allow a single process to outgrow a system's main memory and cause extensive swapping and paging.

It is highly recommended to stay within the bounds of the available memory for this tunable parameter, and never raise the tunable parameter to a value larger than the physical memory available plus the configured swap space.

The recommended maximum value for `per_proc_data_size` is 0x40000000000.

The recommended value for `max_per_proc_data_size` is to set it to the same value as `per_proc_data_size` with a maximum of 0x40000000000.

per_proc_address_space and max_per_proc_address_space

This tunable parameter specifies the per process address size in bytes. This parameter needs to be large enough to account for the size of the virtual address space (including the dynamically allocated memory and the SGA) of the largest process in the system. It is possible to raise the value to a larger value than actual memory available. However this would allow a single process to outgrow a systems main memory and cause extensive swapping and paging.

It is highly recommended to stay within the bounds of the available memory for this tunable parameter, and never raise the tunable parameter to a value larger than the physical memory available plus the configured swap space.

The recommended maximum value for `per_proc_address_space` is `0x40000000000`.

The recommended value for `max_per_proc_address_space` is to set it to the same value as `per_proc_address_space` with a maximum of `0x40000000000`.

max_proc_per_user

This tunable parameter specifies the maximum number of processes a non-root user can create. To disable the limits for this tunable parameter it can be set to `0`. Disabling this limit has a slight performance advantage.

The recommended value for `max_proc_per_user` is `1024`. If the application environment requires more than `1024` processes per user the value can be increased accordingly.

max_threads_per_user

This tunable parameter specifies the maximum number of threads a (non-root) user can create. To disable the limits for this tunable parameter it can be set to `0`. Disabling this limit has a slight performance advantage.

The recommended value for `max_threads_per_user` is `4096`. If the application environment requires more than `4096` threads per user, the value can be increased accordingly.

maxusers

This tunable parameter is a generic base tunable parameter used to calculate the value of various data structures, tables and other tunable values.

The recommended value for `maxusers` on AlphaServer ES40 class systems and higher is `8192`. This value can be increased to the maximum of `16384`.

RT Subsystem

The `aio_task_max_num` parameter should be set in the `rt` subsystem section of `/etc/sysconfigtab`. This tunable parameter specifies the maximum number of concurrent active AIO tasks.

The current recommended value is `8193`.

RDG Subsystem

Set the following tunable parameters in the `rdg` subsystem section of `/etc/sysconfigtab`. Parameters in this section only apply to systems running Oracle RAC over RDG.

max_objs

HP recommends setting this tunable parameter to at least five times the number of Oracle processes per node, and up to `10240` or the number of Oracle processes multiplied by `70`, whichever is larger.

msg_size

HP recommends setting this tunable parameter equal to or greater than the maximum value of the `DB_BLOCK_SIZE` parameter for the database. Oracle recommends a value of `32768` because Oracle 9i and later support different block sizes for each table space.

max_async_req
HP recommends setting this tunable parameter to at least 256.

Note: As of Tru64 UNIX V5.1B-2 (PK4) Service PAK, the default value for this parameter has been raised to 1000.

max_sessions
HP recommends setting this tunable parameter to at least the number of Oracle processes plus 2.

rdg_max_auto_msg_wires
HP recommends setting this tunable parameter to 0.

RM Subsystem

Due to an oversight, the Oracle 9i RAC installation guide states that the value for the `rm_check_for_ipl` parameter must be set to 0. This is no longer true. `rm_check_for_ipl` should be left at its default value. Make sure that this parameter is not changed in the `/etc/sysconfigtab` file.

Improving the Performance of gettimeofday() in Oracle

The Oracle server times many functions as it executes; this is especially true if the `init.ora` parameter `timed_statistics` is set to TRUE.

These timing functions result in system calls into the operating system kernel, which can degrade Oracle performance because the calling process relinquishes the CPU. There is a feature in Tru64 UNIX that gives a process direct access to the operating system's real-time clock.

Using this feature will improve performance on a heavily used system; it will also improve performance on a lightly loaded system but it may not be as noticeable.

To enable this feature, enter the following command sequence with root privileges:

```
mknod /dev/timedev c 15 0
chmod 644 /dev/timedev
```

The previous commands only have to be performed once for each system or node in a cluster. The `/dev/timedev` device special file will be persistent across system reboots. In order to use this feature with Oracle the instance has to be restarted. The existence of `/dev/timedev` is checked only on instance startup.

It is recommended that all instances in a TruCluster Server environment (and hence all nodes) have this feature enabled. This feature is supported on Oracle version 7.3 and later.

Oracle has made further optimizations to this in the latest versions of Oracle 9 and Oracle 10 by using a newer version of the `times(3)` system call. These versions of Oracle require that an ERP be installed that implements the changed system-call. Seethe Tru64 UNIX/TruCluster Server V5.1B-2 with PK4 Service PAK for V5.1B-2 section and Tru64 UNIX/TruCluster Server V5.1A with PK6 Service PAK for V5.1A PK6 section for more information.

Selecting IPC Communication Protocols for Oracle RAC

Oracle can use either User Datagram Protocol (UDP) or Reliable Datagram (RDG) for DLM/IPQ inter-instance communication. Starting with Oracle 9i, HP recommends using RDG as the protocol for IPC instead of using UDP.

RDG communication is recommended for Oracle 9i Real Application Clusters (RAC) and later.

Although Oracle 8.1.7 Oracle Parallel Server (OPS) does support the use of RDG for communication it is not recommended to enable RDG and to continue to use UDP instead.

The following commands show how to enable/disable the different protocols for IPC.

To enable Oracle RAC:

```
cd $ORACLE_HOME/rdbms/lib
```

For Oracle 8i:

```
make -f ins_rdbms.mk ops_on
make –f ins_rdbms.mk ioracle
```

For Oracle 10g/9i:

```
make -f ins_rdbms.mk rac_on
make -f ins_rdbms.mk ioracle
```

To use the UDP Protocol for IPC (the default for Oracle 8i):

```
make -f ins_rdbms.mk ipc_udp
make -f ins_rdbms.mk ioracle
```

To use RDG Protocol for IPC (the default for Oracle 9i):

```
make -f ins_rdbms.mk rac_on
make -f ins_rdbms.mk ioracle
```

Selecting NUMA Support in Oracle

When a single Oracle instance is used in more than 2 QBBs on large EV6 based NUMA systems, such as the GS160 and the GS320, it may be advantageous to enable NUMA mode in Oracle as well. This reduces the amount of references to memory in other QBBs and can substantially improve performance. However, there is additional management overhead associated with it as well, which consists of at least some of the following items:

- Oracle must be rebuilt to include NUMA support (see below).
- Oracle will split the fixed, variable, and block buffer areas in the SGA into different shared memory segments. Oracle will also distribute these shared memory segments over each QBB in the system and allocate separate shared memory segments to do so. This will significantly increase the number of shared memory segments that need to be allocated for the instance.
- The Oracle `db_writers` parameter should be set to the number of QBBs in the system, and each QBB should have a storage adapter that has connectivity to all the necessary storage for Oracle. Not all of the QBBs may have storage connectivity, so `db_writers` should be set to the number of QBBs that do have storage connectivity. Oracle will allocate the specified number of DB writer processes evenly over the QBBs, for example, if `db_writers` is set to 2 in a 4-QBB system, then Oracle will allocate 2 DB writer processes and bind one to QBB 0 and one to QBB 2. Make sure that QBBs with storage connectivity match the QBBs to which Oracle has bound the DB writer processes as this may require hardware reconfiguration if they do not match.

On large EV7 based systems (ES80 and GS1280) it is also possible to achieve a very small performance benefit from enabling NUMA mode in a large single Oracle instance. HP does not recommend enabling Oracle NUMA mode on EV7 based systems.

To enable NUMA support in Oracle 10g, Oracle 9i, or Oracle 8i, enter the following commands:

```
cd $ORACLE_HOME/rdbms/lib
make -f ins_rdbms.mk numa_on
make -f ins_rdbms.mk ioracle
```

To disable NUMA support in Oracle 10g, Oracle 9i, or Oracle 8i, enter the following commands:

```
cd $ORACLE_HOME/rdbms/lib
make -f ins_rdbms.mk numa_off
make -f ins_rdbms.mk ioracle
```

Note: In the Oracle Database Reference, enabling NUMA options is referred to as “Enabling Oracle Database Directed Placement Optimizations”.

Enabling NUMA support in Oracle is currently not supported before Oracle 9iR2 in combination with RAC. If you are planning to use RAC you must have NUMA disabled. Starting with Oracle 9.2 (Oracle 9iR2) the use of NUMA enablers in an Oracle 9iR2 RAC environment has been certified and is now fully supported.

For more information

Tru64 UNIX NUMA Overview
http://h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51_HTML/NUMA/TITLE.HTM

Tru64 UNIX V5.1B Base Operating System Documentation
http://h30097.www3.hp.com/docs/pub_page/V51B_DOCS/V51B_DOCLIST.HTM

TruCluster V5.1B Documentation
http://h30097.www3.hp.com/docs/pub_page/cluster51B_list.html

Tru64 UNIX and TruCluster Server best practices documentation
http://h30097.www3.hp.com/docs/best_practices/

Cluster File System (CFS) Direct I/O Technical Description
http://h30097.www3.hp.com/docs/cluster_doc/cluster_51A/HTML/ARHGVDTE/TITLE.HTM

Tru64 and Oracle 9i white paper
http://www.oracle.com/technology/products/oracle9i/pdf/Oracle_9i_on_Tru64_UNIX.pdf

Tru64 to HP-UX 11i transition modules
http://h30097.www3.hp.com/transition/modules.html?jumpid=reg_R1002_USEN

Oracle Web sites and resources:

ORACLE 10g
http://www.oracle.com/technology/products/database/oracle10g/index.html

ORACLE 9i
http://www.oracle.com/technology/products/oracle9i/index.html

Oracle Products & Services
http://oraclepartnernetwork.oracle.com/portal/page?_pageid=0,16144&_dad=moc&_schema=MOC

Oracle Events
http://www.oracle.com/webapps/events/Events.jsp

Technical Documents (certify, white papers, fact sheets, benchmarks)

http://docs.oracle.com/

http://www.oracle.com/technology//index.html

http://metalink.oracle.com/

Marketing, Press Releases, Success Stories, Collateral
http://www.oracle.com/corporate/press/home/index.html

Acknowledgments

Thanks are due to everyone who assisted with the creation of this document, particularly the following:

Han Pilmeyer, primary contributor
Mario Broodbakker
Geoff Gardner
Hein van den Heuvel
Jan Mark Holzer
Bill Kirk
Euan McMaster
John Shakshober
Thomas Sjolshagen

Revision Information

This is a new document.

4AA0-9949ENW, December 2006