Linux Virtual Memory in Red Hat Linux Advanced Server 2.1 and Oracle's Memory Usage Characteristics

An Oracle White Paper
June 2002

Linux Virtual Memory in Redhat 2.1 Advanced Server and Oracle's Memory usage characteristics

Linux Virtual Memory in Redhat 2.1 Advanced Server and Oracle's Memory usage characteristics

1. EXECUTIVE OVERVIEW

Oracle database server is traditionally run on large servers that are expensive in price and maintenance. But with huge technological strides in the commodity server area and enhancements in Oracle database architecture, Intel-based servers have become an attractive market for Oracle. Price to performance has always been a strong point for the Intel servers. If the scalability and fault tolerance capabilities of Oracle9i RAC are added to the low cost servers, there is a strong case for lowering total cost of ownership without sacrificing the high availability and stability of the systems.

One of the key components of this solution is the operating system. In the x86 architecture space, Linux is a very popular operating system. Although based on UNIX, until now it had several missing pieces for it to be truly enterprise ready. One major area where the OS was lacking stability and scalability was the Virtual Memory Management. The work described here has solved most of those problems from the point of view of being able to run Oracle software extremely reliably.

2. INTRODUCTION

In order to get above 4GB virtual memory on IA-32 architecture a technique known as PAE (Page Address Extensions) is used. It is a method that translates 32-bit linear addresses to 36-bit physical addresses. In the linux kernel, the support is provided through a compile time option that produces two separate kernels - the SMP kernel which supports only upto 4GB VM and the enterprise kernel which can go up to 64GB VM (also called VLM capable). This means applications like oracle can make use of the large memory and scale up to a large number of users without loss of performance or reliability.

Oracle also makes extensive use of shared memory. A generic 2.4.x kernel would only allow up to 1.8GB of SGA. But enhancements to the Advanced Server release of the Linux kernel would allow larger SGA sizes. A large SGA and a big database instance would produce a lot of page table entries in the kernel that would severely limit the number of users supported and may lead to extremely slow performance and even crash. There has been considerable work done to overcome those limitations and show a graceful degradation under heavy stress.

The intention here is to look at the Linux VM architecture and the enhancements that have gone into the Advanced Server kernel that would enable oracle database to scale much better in a lot more reliable manner.

3. TERMINOLOGY

Name	Explanation
VLM	Very large Memory (> 4GB memory)
shmfs	Shared memory file system on /dev/shm
lowmem	Lower memory addresses directly mapped by kernel
highmem	Higher memory addresses
mapped_base	Per process value for setting address to search memory chunk during mmap
VLM window	Virtual memory range to dynamically map in the buffer cache
pte	Page table entry
highpte	Kernel patch which allows putting pte-s into highmem
enterprise kernel	Kernel that can support VLM
SMP kernel	Kernel that can support only up to 4GB VM
SGA	Shared Global Area for oracle instance
TLB	Translation Lookahead Buffer
bigpages	Kernel feature to enable large page size for shared memory pages

Table 1: Technical terms used in this document

4. VM ARCHITECTURES IN LINUX

Parallel hardware systems utilizing inexpensive commodity components have the potential to provide excellent price/performance advantages over traditional mainframe systems in data-intensive decision support applications. Tightly coupled Symmetric Multi-processor systems (SMP) have been the most widely used parallel hardware systems. These systems utilize multiple processors that share common memory and disk resources and hence are also known as ‘shared everything’ systems. Primary advantages of SMP systems include simplicity of application development and ease of administration. These systems, however, do not provide any inherent fault-tolerance—the failure of a single critical component such as a CPU could bring the entire system down. Further, they are currently somewhat limited in terms of scalability and growth due to limitations in available system bus bandwidth and operating system software scalability.

5. ENHANCEMENTS AND FIXES

5.1 VLM support:

Ability to use up to 64GB pagecache on 32-bit system.

5.1.1 Problem:

IA-32 can address only up to 4GB memory by default. This poses a severe limitation on high end enterprise class applications like oracle which depends on availability of large memory space for use of its SGA and to scale to a large number of users. Typically, the kernel would reserve some amount of memory and the remaining would have to be shared by the SGA and the oracle processes. This means that the number of users and the size of SGA would be limited to using only up to 4GB of virtual memory.

5.1.2 Solution:

The PAE(page address extensions) mechanism allows addressing using 36bits on IA-32 systems. Linux makes use of this technology to implement the VLM feature which extends the pagecache to 64 GB in the enterprise kernel. This immediately increases the scalability of oracle database in the number of concurrent users that can be supported.

5.1.3 Use:

The enterprise kernel is able to use up to 64GB pagecache without any modifications.

5.2 Large SGA size capability through changing of mapped_base

Changing of the memory address where the oracle executable and libraries will be loaded to increase the space for larger SGA to fit in memory. In the VLM case, changing the address where the kernel will search for a free chunk of VM space during mmap would also allow a larger size of the VLM window of the SGA

5.2.1 Problem:

In earlier kernels, by default the kernel parameter TASK_UNMAPPED_BASE used to be set to TASK_SIZE/3, where TASK_SIZE is the user space process size limit and is set to 3GB by default. This implies that the value of TASK_UNMAPPED_BASE used to be 1GB. The maximum possible SGA size on a machine with 4GB RAM was 1.7GB. Also, the maximum VLM window size that was possible with this setting was 512MB on a system with 4GB maximum VM size (non-enterprise kernels).

5.2.2 Solution:

The TASK_UNMAPPED_BASE parameter is now a value that can be set dynamically for individual processes in the /proc file system. Thus, oracle startup process can set a suitable value (see next section) in /proc/<pid>/mapped_base file and the kernel will use that value to look for mmap-ping a chunk of virtual memory for the SGA window. This feature needs to be used in conjunction with a relink of oracle with a particular value for generating ksms.o. The oracle libraries and executable will be loaded at a modified address, and the kernel would start looking for mmap-ping VLM window from the mapped_base address which results in a larger window size. For the non-VLM case, there would be a bigger part of memory available for fitting the SGA.

The default address where oracle is loaded from is 0x50000000 and the default mapped base is 0x40000000 (decimal 1073741824). The space between 0x40000000 to 0x5000000 is reserved for loading oracle libraries. If the mapped_base is lowered to 0x10000000 (decimal 268435456), then the space between 0x10000000 to 0x12000000 is used for loading the oracle libraries. Oracle executable will start getting loaded from 0x12000000 thus allowing a bigger SGA and a larger window size. The lowered values of mapped base specified above are examples, and the user should set these to values appropriate for their systems.

5.2.3 Use:

Relink oracle with the following options:

In $ORACLE_HOME/rdbms/lib, run

```
# genksms -s 0x12000000 > ksms.s
#  make -f ins_rdbms.mk ksms.o
#  make -f ins_rdbms.mk ioracle
```

Find out the pid of the process (shell) from where oracle will be started using *ps* command. Then execute the following command as root:

```
echo 268435456 >/proc/$pid/mapped_base
```

This will lower mapped base to 0x10000000 (decimal 268435456) so that oracle can be loaded from lowered mapped base. Start oracle from the shell whose mapped base is lowered**.** If there are errors in the startup of oracle with lowered mapped base, use a value higher than *0x12000000* in ksms.s and relink oracle.

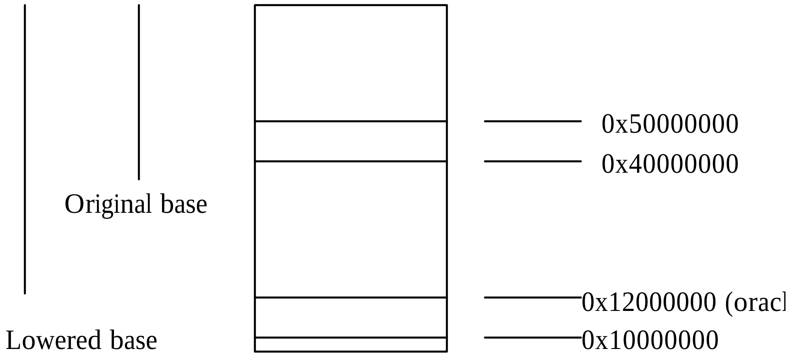


Figure 1: Schematic diagram for lowering mapped base in memory for oracle

5.3 Shared memory file-system(shmfs) support

Memory-based file system optimized for shared memory operations and for larger SGA size.

5.3.1 Problem:

Using regular shmget() and shmat() calls for SGA, the limit in the size of SGA for oracle in linux kernel is 1.7GB. For applications running on oracle that do a lot of I/O (OLTP type), a larger SGA size is desired.

5.3.2 Solution:

The shmfs (/dev/shm based) is used by oracle to memory map the dynamic portions of the SGA. This can theoretically allow an SGA up to the size of the shmfs file system that is created. Since shmfs is a memory file system, its size can be as high as the maximum allowable VM size which is 64GB.

5.3.3 Use:

Mount the shmfs file system as root using command:

```
mount -t shm shmfs -o nr_blocks=8388608 /dev/shm
```

Set the shmmax parameter to half of RAM size

```
echo 3000000000 >/proc/sys/kernel/shmmax
```

Set the init.ora parameter **use_indirect_data_buffers=true**

Startup oracle.

5.4 Highpte fix

Enabling the VM to write page table entries in the highmem area.

5.4.1 Problem:

Traditionally in linux kernel, the VM allocated page tables in the lowmem pool which has a size limitation of 1GB. With support for VLM and because oracle is an enterprise application with lot of SGA and user memory requirements, the size of the lowmem pool becomes a bottleneck. The number of page table entries generated with a sizable number of connected oracle users soon starts choking the lowmem pool. Even though there is lot of free physical memory and swap space still available for more users to connect, the system runs out of space for pte's and hence scalability becomes limited.

Chart 1 below shows a typical memory usage pattern with light to moderate workload on a Linux system before the application of the highpte patch. The workload used was oracle tpcc with 200 warehouses and 200 concurrent users on a 4-way system with 8GB RAM and 2.7GB SGA. Since the system has relatively lower workload, even when it runs out of lowmem, it is able to recover and continue. The point to note here is that the lowfree line(dashed, bottom-most) is very low, even though the memfree line (topmost) and the highfree line (dotted, the one following memfree) indicate sufficient available space.

Chart 1: Memory usage pattern for successful run before highpte patch

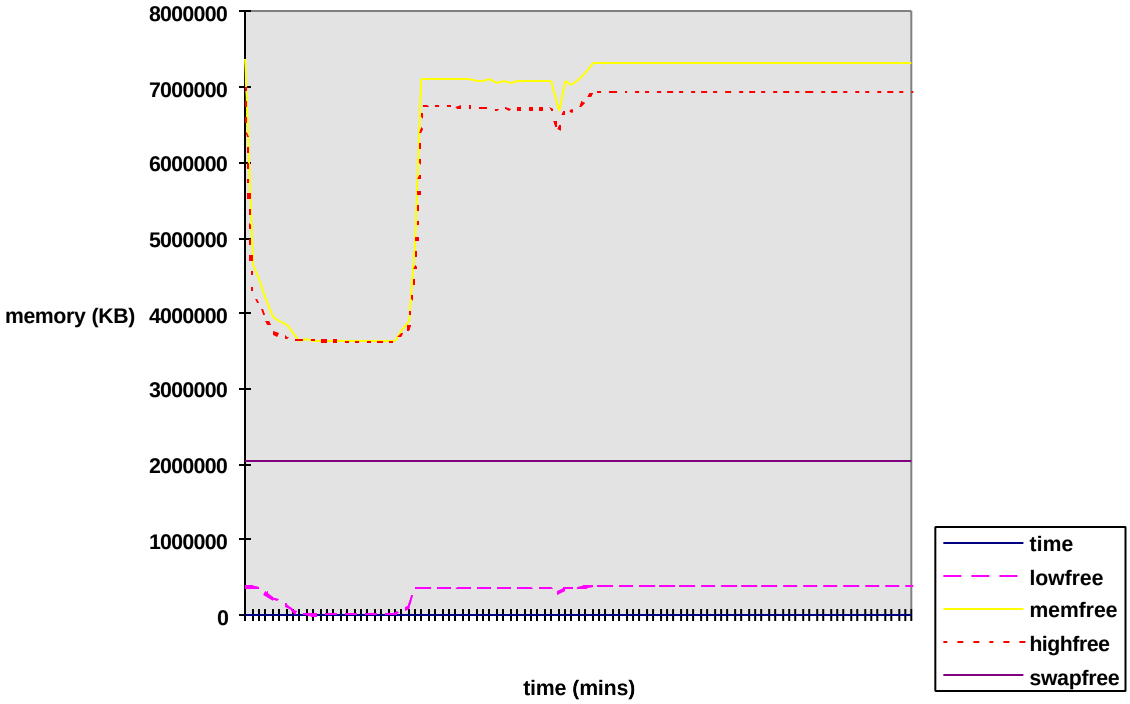
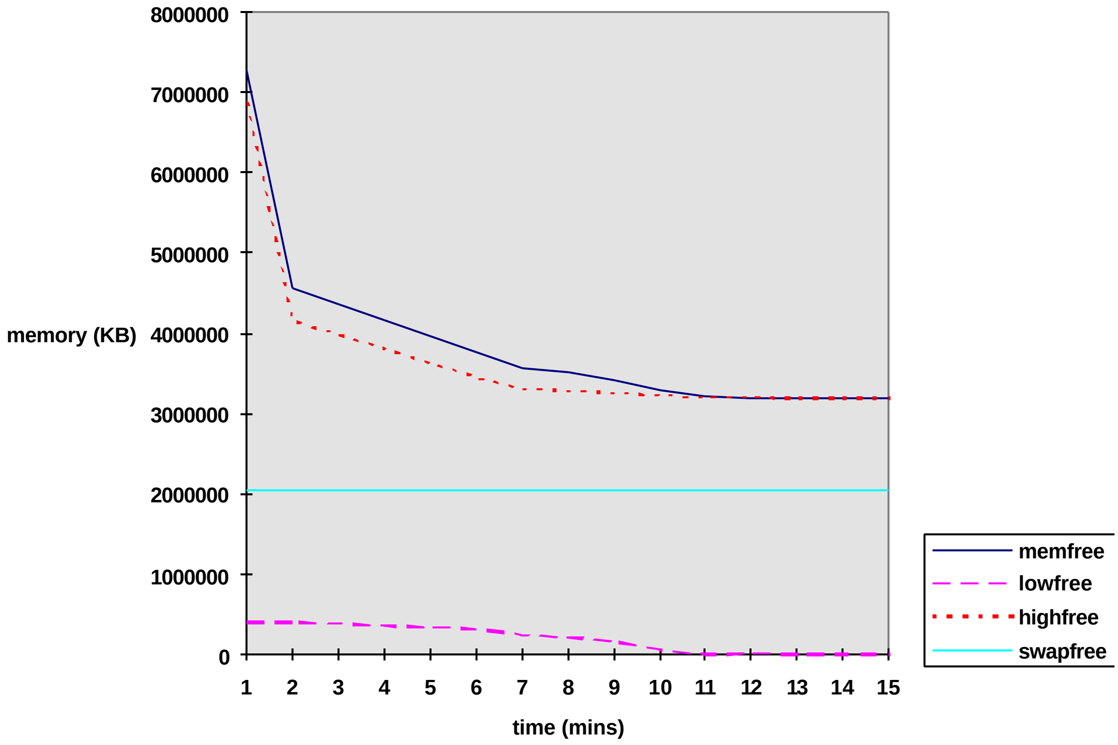


Chart 2 below shows the effect of slightly increasing the load on the same system (i.e. without highpte patch). With the same workload type of oracle and tpcc, but with about 300 users, the system quickly begins to run out of lowmem as there are lot of pte's generated as a result of SGA activity. Notice however, that there is plenty (about 3GB) of free memory and all of the swap free. After a while, the system completely freezes and becomes unresponsive because there is no more space to write the additional pte's. In the chart, the topmost line is for memfree, the line closely following it is highfree (dotted) and the bottom-most line which almost merges with the horizontal axis around the hang time is lowfree (dashed).

Chart 2: Memory usage pattern for crash run before highpte patch



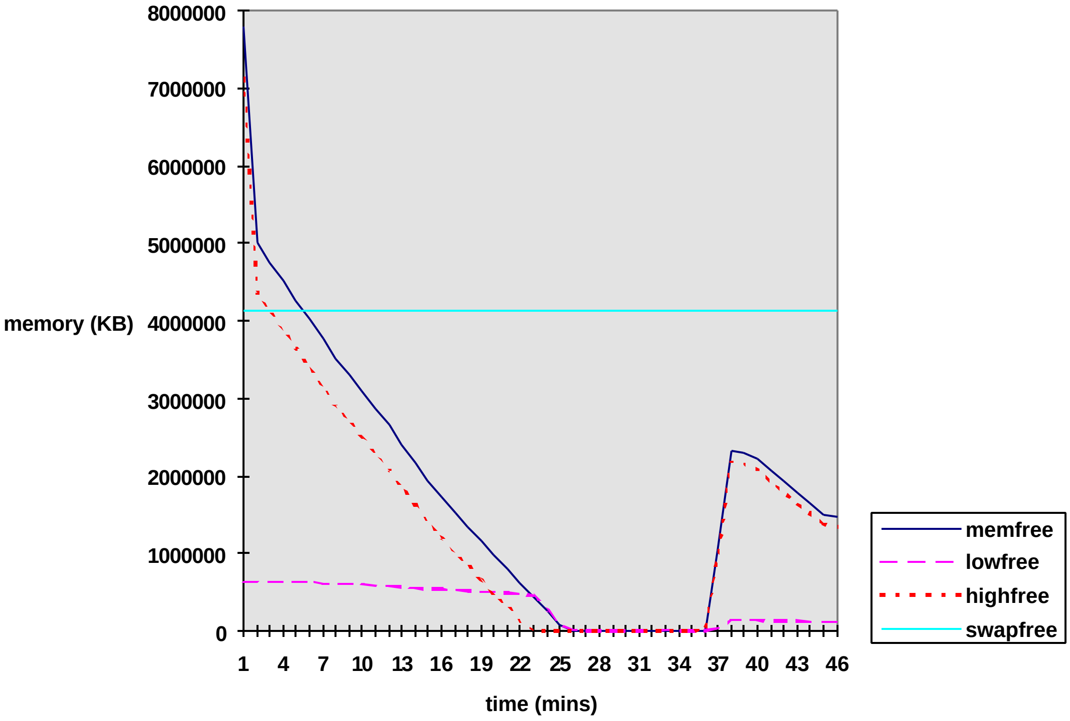
5.4.2 Solution:

The highpte fix allows the VM to put page table entries in the highmem pool. As a result, more simultaneous user connections can be handled until the system really runs out of memory with large number of users. The implementation does not use *kmap()* - this would prevent deadlocks in systems with load. The kernel-VM pte entries are not put in the highmem pool, only the user space pte entries overflow into highmem. Since the kernel pagetables are shared by every process in the system, there is tremendous overhead saving with direct mapping of those entries.

Chart 3 below shows the effect of the above fix on the oracle tpcc workloads mentioned in the Problems part of this section. Even with large number of concurrent oracle users (around 600), the system does not hang or crash. From the chart, it can be noted that at one point of time, there is very low available memory - all of lowmem, highmem and available memory are almost exhausted.

But since there is no restriction of writing pte's only to lowmem area anymore, the system can recover from the heavy stress situation much more elegantly. Also notable is the fact that during the period of heavy stress when the memfree (topmost), highfree (dotted, follows memfree) and lowfree (dashed, bottom-most) lines in the chart almost merge, all of the available memory is being used to serve the large number of processes.

Chart 3: Memory usage pattern for successful run after highpte patch



5.4.3 Use:

Automatic with AS kernels.

5.5 Bigpages feature

Page frame of size 4MB as opposed to the regular 4KB

5.5.1 Problem:

The regular page frame size in the Linux kernel is 4KB. This is fine for most applications, but with oracle and the SGA, there is room for optimization. Since oracle uses a large SGA which maps big chunks of memory, there would be a lot of page table entries generated for the SGA if the page size is 4KB. This means that the kernel overhead for storing the pte-s would be high for systems with large SGA. On enterprise kernels with shmfs, the SGA size can be very large and we would want the kernel overhead reduced if possible.

5.5.2 Solution:

Oracle uses a large contiguous area in the VM for mapping the VLM window. These are used for the dynamic part of the SGA the size of which is specified by the db_block_buffers parameter. The pages corresponding to this area in the VM can easily be of a larger size than the default 4KB and yet there would not be any of the problems like granularity associated with using large page size. A page size of 4MB for these pages would reduce the number of pte-s thus reducing the kernel overhead considerably. The number of TLBs used are also fewer thus reducing TLB thrashing. The result is better scalability in terms of the number of oracle users. Better performance is also achieved because the big pages are not swapped out which means the entire db_block_buffers are in physical memory. The system performance increases as a result of kswapd not having to 'think' about swapping out these pages. Since swap space is not pre-allocated for these pages, there is more swap area available and less pagecache complexity.

5.5.3 Use:

In the kernel boot options, add **"bigpages=<size>MB"** to the boot loader file (e.g. **/etc/lilo.conf**), where size is a value in MB appropriate for your system. Also set the /**proc/sys/kernel/shm-use-bigpages** file to contain the value **2**. The other possible values are **0** for no bigpages and **1** for bigpages using sysV shared memory (as opposed to shmfs).

Maximum bigpages value for system can be obtained by following formula:

Maximum value of Bigpages = HighTotal / 1024 * 0.8 MB

where HighTotal is value in Kbytes and obtained from /proc/meminfo.

The assumption is that 20 % memory is reserved for kernel bookkeeping.

For example machine with 8 GB memory has HighTotal of 7208944 kB and therefore maximum bigpages value will come around 5631 MB. If the value for bigpages is set to a very high value, the memory available for user connections would be low. There is a trade-off between the number of users and the bigpages value. If we know approximately the maximun number of user connections expected and how much memory each will consume, then exact value bigpages can be calculated as follows:

Bigpages = (HighTotal - Memory required by maximum user connections in KB) / 1024 *0.8 MB

6. VALID COMBINATIONS

Some example cases are summarized in the following table -

If you want …	**Then use …**
Oracle on > 4GB intel server	Enterprise kernel
Above 1.7GB SGA, but not on an enterprise kernel	use_indirect_data_buffer=true in init.ora /dev/shm mounted with appropriate size OR use_indirect_data_buffer=false in init.ora /proc/<pid>/mapped_base set to 268435456 oracle relinked with ksms set to 0x12000000
Above 1.7GB SGA on an enterprise kernel	The settings 1 and 2 above and all settings for bigpages.
VLM window size > 512MB	/proc/<pid>/mapped_base set to 268435456 oracle relinked with ksms set to 0x12000000

Table 2: Some useful scenarios for Oracle SGA and how to achieve them

7. LIMITS

The limits on number of users and the size of SGA on stress tpcc runs without async I/O and on raw devices are compiled in the following table (Table 3). The size of the database in all cases was 50GB and the number of warehouses was 200. The database was running in dedicated server (non-MTS) mode. The kernel used in obtaining the results had all the VM enhancements and fixes applied to it. The system was dedicated to running only the workload described above.

RAM size /swap size	SGA size	db_block_buffers	shm-use-bigpages & bigpages	maximum number of concurrent users
4GB/4GB	1GB	0.6GB	0 & x	1200
4GB/4GB	1.7GB	1.3GB	0 & x	800
4GB/4GB	2.0GB	1.6GB	0 & x	700
4GB/4GB	2.6GB	2.2GB	0 & x	600
8GB/4GB	2GB	1.6GB	2 & 2100MB	1200
8GB/4GB	2.7GB	2.4GB	0 & x (LB, no shmfs)	900
8GB/4GB	2.7GB	2.4GB	2 & 3000MB	900
8GB/4GB	2.7GB	2.4GB	0 & x	900
8GB/4GB	4GB	3.5GB	0 & x	700
8GB/4GB	4GB	3.5GB	2 & 4100MB	700
16GB/8GB	1.7GB	1.3GB	0 & x	1200++
16GB/8GB	2.7GB	2.3GB	0 & x	1200+
16GB/8GB	8GB	7.2GB	0 & x	1200
16GB/8GB	8GB	7.2GB	2 & 8200MB	1200
16GB/8GB	14GB	12.5GB	0 & x (bigpages bug)	100

Table 3: Number of oracle concurrent users with varying memory and SGA size

The numbers in the last column denotes the maximum number of concurrent oracle database users that could be run on the system without it becoming unacceptably slow and causing timeouts to the tpcc clients. The general observation is that with increased size of SGA and the same RAM/swap sizes, the number of users that can be supported goes down as expected. But for most typical combinations (SGA size about half of the RAM size), the number of users permitted is a healthy figure after applying the enhancements to the original kernel. It should be noted that the tpcc clients were also running on the same machine as the database server - so the actual number of operating system users is double the figure shown on the last column.

The advantages of using bigpages may not be apparent from the data in the table. But what was noticed was that with bigpages turned on and the value tuned properly (see section 5.5.3), the average TPMC numbers (measuring performance) were better than those without bigpages.

8. CONCLUSION

The Linux Virtual Memory architecture lacked several important features required for Oracle software - in particular, the database server - to run optimally. In the Red Hat Linux Advanced Server kernel, most of these shortcomings were addressed so that oracle could scale better and run reliably. The large memory addressing capability, shared memory file system and mmapped SGA support on it, configurable process mapped base, the pte entries in highmem and the bigpages feature are all part of this new kernel that will allow oracle to use the system memory resource to the fullest.

Linux Virtual Memory in Redhat 2.1 Advanced Server and Oracle's Memory usage characteristics

ORACLE

Linux Virtual Memory in Red Hat LinuxAdvanced Server and Oracle's Memory Usage Characteristics
June 2002
Author:Niloy Banerjee
Contributing Authors: Deepak Patel

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
www.oracle.com