Tuning IBM AIX 5.3 and AIX 6.1 for Oracle Database

Dennis Massanari, Arun Sar
IBM eServer Solutions Enablement

Lilian Romero
IBM Power Systems Performance

January 2009

Table of contents

Abstract

This paper discusses performance analysis and tuning for Oracle workloads on IBM AIX 5.3 and IBM AIX 6.1 configurations. Oracle performance tuning is a vital part of the management and administration of successful database systems. The reader will learn how best to exploit the specific performance features of AIX 5.3 and AIX 6.1 on IBM POWER systems when running an Oracle database (including simultaneous multithreading [SMT] and IBM Micro-Partitioning technology). There is a discussion of many performance-tuning tools and configuration options to control memory usage, process execution and I/O.

Introduction

In this paper, the required operating-system tuning tasks are grouped into three sections: processor, memory and I/O tuning. Fine-grained tuning is important and valuable, although this level of tuning can lead to problems if not used correctly. Each of these is discussed in detail. The reader will come to understand that performance tuning involves trade-offs that might favor one environment, while hurting another. Therefore, the paper also explains the importance of applying tuning changes one at a time, while monitoring system throughput to determine if the alteration is beneficial for the environment.

This paper covers tuning on IBM® POWER5™, IBM POWER5+™ and IBM POWER6™ systems. For convenience, these will be collectively referred to as IBM POWER systems and their processors will be referred to as IBM POWER™ processors in this paper.

Except when needing to distinguish between IBM AIX® 5.3 and AIX 6.1 these AIX operating systems will be referred to collectively as AIX.

Note: Oracle Database generic-tuning concepts are widely available and are usually independent of the system and hardware. These concepts are not discussed in this paper.

Tuning processor performance

To gain the most out of IBM POWER processors, in a system that runs AIX, tune the environment to take advantage of simultaneous multithreading (SMT) and IBM Micro-Partitioning™ technology.

Simultaneous multithreading

SMT is a new feature on POWER5 (or later) systems that run AIX 5.3 or IBM i5/OS® Version 5, Release 3 or newer releases of these operating systems, including AIX 6.1. (**Note:** This paper addresses SMT characteristics in the AIX environment.) SMT allows two instruction streams to share access to the execution units on every clock cycle. Instructions are fetched and dispatched based on the available resources at execution time, allowing for the interleaving of instruction streams to improve concurrent instruction execution and for better use of hardware resources. The operating system abstracts each instruction path as a separate processor (also referred to as a *SMT thread of execution*). The two instruction paths appear as two logical processors, one for each instruction stream. For example, a server with four physical processors and SMT enabled has eight logical processors as far as the operating system or Oracle Database is concerned. Oracle processor licensing is based on the number of physical processors used, which in this case, is four.

A key SMT feature is that no application modification or tuning is needed to benefit from it. SMT is available on POWER processor-based systems with AIX 5.3 (or newer) and is dynamically enabled or disabled. This enablement is specific to a partition, including a partition where the Micro-Partitioning features is enabled) and allows multiple concurrent partitions to run in different modes (single threading and SMT). To control the SMT modes, AIX 5.3 provides the **smtctl** command with the following syntax:

smtctl [–m off | on [–w boot –w now]], where:

- **-m off** : This option disables SMT mode.
- **-m on** : This option enables SMT mode.
- **-w boot**: This option makes the SMT mode change effective on next and subsequent reboots.
- **-w now** : This option makes the SMT mode change immediately, but does not persist across reboot.

Note: If neither the **-w boot** nor the **-w now** options are specified, then the mode change is made immediately and persists across subsequent boots.

A user with root authority can use the smtctl command with no options to verify that the hardware is SMT-capable and to check the current SMT state (see Figure 1).

```
# smtctl
This system is SMT capable.
SMT is currently enabled.
SMT boot mode is not set.
SMT threads are bound to the same physical processor.

proc0 has 2 SMT threads.
Bind processor 0 is bound with proc0
Bind processor 1 is bound with proc0

proc2 has 2 SMT threads.
Bind processor 2 is bound with proc2
Bind processor 3 is bound with proc2
```

Figure 1. Using the smtctl command with no options to verify that the hardwre is SMT-capable

Users without root authority can see if SMT is enabled by using the AIX **lparstat** command, as in the following example (see Figure 2).

```
$ lparstat
System configuration: type=Dedicated mode=Capped smt=On lcpu=12 mem=17408
%user  %sys  %wait  %idle
----- ----- ------ ------
  2.1   1.5    0.1   96.3
```

Figure 2. Using the lparstat command to see of SMT is enabled

The benefits of SMT vary from workload to workload. Workloads with a high number of runnable processes or threads receive the most benefit. The benefit of SMT is improved system-throughput capacity, which results from better instruction-execution concurrency and better utilization of the execution-hardware resources. The response time of individual threads can increase, but the total system throughput also increases. It is recommended that, by default, you enable SMT if the system allows it.

Process scheduling and priority

In most situations, the user should not change the priority of Oracle processes. On the AIX operating system, the default scheduling policy is fair round robin, also referred to as *SCHED_OTHER*. This policy implements a classic priority-queue round-robin algorithm with one major difference: the priority is no longer fixed. For example, this means that, if a task uses a relatively large amount of processor capacity compared to other processes at the same priority level, its priority level is slowly made less favorable to give other jobs an opportunity to gain access to the processor.

Process priority and RAC

In Oracle RAC 10gR2 and 11gR1, several critical processes run with a special scheduling policy and priority. You should not change the priority of these processes, but it is useful to understand which processes have special scheduling requirements so that you can monitor them to make sure the system is properly configured to allow them to run with the proper settings.

In Oracle Clusterware, the **oprocd** and **ocssd.bin** processes monitor the health of the nodes and cluster that interconnect in a Oracle Real Application Clusters (RAC) system. These processes need to quickly detect and respond to potential cluster problems, so they are given a special fixed scheduling policy and more favorable priority to ensure that they are scheduled promptly when they become runnable.

In the RAC database, the Lock Manager Server (LMS) processes serve cache data blocks to processes on other nodes in the RAC cluster. If the LMS process does not get the processor resources it requires, it can slow down processing in the entire cluster, so it runs at a slightly more favorable fixed priority. On a RAC system, there can be times when lots of data blocks need to be passed from the buffer cache on one node to the buffer cache on another node. During this time, the LMS processes can consume a significant amount of processor capacity, and it is important that their priority is not downgraded during this time, as might be the case if they were running with the default AIX scheduling policy. For this reason, the LMS processes run with a fixed and slightly more favorable priority than most other user processes.

VKTM is a new process in Oracle 11g. It provides timekeeping functions for other Oracle processesc; therefore, it is also given special priority so that it does not hold up other processes that are waiting on timing functions.

Table 1 indicates the correct values for Oracle 10.2.0.4 and 11.1.0.6 (or newer) releases.

Process	Priority	Scheduling policy	Scheduling notation in ps command output
Oprocd	0 (highest priority)	sched_rr (fixed priority)	2
ocssd.bin	1	sched_rr (fixed priority)	-
LMS processes	39	sched_rr (fixed priority)	2
VKTM process	39	sched_rr (fixed priority)	2

Table 1. Correct values for Oracle 10.2.0.4 and 11.1.0.6 (or newer) releases

You can verify the priority and scheduling policy of these processes with the following commands (see Figure 3).

```
$ ps -efo "pid,user,pri,nice,sched,command"|egrep "PRI|ocssd.bin|oprocd"
    PID     USER PRI NI SCH COMMAND
 475154     root   0 --   2 oprocd
 503882   oracle   1 --   - ocssd.bin

$ ps -efo "pid,user,pri,nice,sched,args"|egrep "PRI|--"
151722   oracle  39 --   2 ora_vktm_tpch6
155832   oracle  39 --   2 ora_lms1_tpch6
112044   oracle  39 --   2 ora_lms0_tpch6
```

Figure 3. Using the ps command to verify the priority and scheduling policy of a process

If the priority and scheduling policy of the oprocd and ocssd.bin process is set incorrectly, verify that the Oracle Clusterware user ID has the proper capabilities set (CAP_NUMA_ATTACH, CAP_BYPASS_RAC_VMM and CAP_PROPAGATE). The CAP_NUMA_ATTACH capability is required because it gives authority to a non-root process to increase its priority (CAP_NUMA_ATTACH also provides other capabilities, but, because oprocd or ocssd.bin do not require these, this paper does not discuss them). The CAP_PROPAGATE capability propagates the parent process's capabilities to the child processes. The CAP_BYPASS_RAC_VMM capability is not related to scheduling, but is required; therefore, the oprocd and ocssd.bin processes can pin memory (CAP_BYPASS_RAC_VMM also authorizes other capabilities that this section does not discuss).

To check existing capabilities for a user whose user ID is *oracle*, see the following example:

```
# lsuser -a capabilities oracle
```

To set the capabilities for that same user, see the following example:

```
# chuser capabilities=CAP_NUMA_ATTACH,CAP_BYPASS_RAC_VMM,CAP_PROPAGATE oracle
```

If the priority and scheduling policy of the LMS and VKTM processes are not correct, verify that the ${ORACLE_HOME}/bin/oradism executable process is owned by root and has the owner **S** bit set.

Processor affinity

On a symmetric multiprocessor (SMP) system, threads are usually scheduled to run on the next available processor. If the threads are always scheduled on the same processor, the thread is said to have processor affinity. AIX favors running threads on the same processor each time they are scheduled. This increases performance because the processor level 1 and level 2 caches already contain code and data for that process, resulting in fewer requests to main memory for data. The disadvantage of processor affinity is that, if the threads with work to do become unbalanced across the processors, then some processors become overused while other processors have idle time. The AIX scheduler automatically adjusts and rebalances the threads across the processors periodically. Scheduling in this way, AIX can optimize for performance, yet balance the workload when necessary. Other affinity alternatives to the default AIX scheduling behavior are processor binding and resource sets.

Processor binding

A process can be forced to run on only one particular processor, which is called *binding a process to a processor*. The benefit is usually an increase in the cache hit ratio (the ratio of the number of times referenced data is found in cache compared to the total number of references). The main disadvantage is that the process then cannot be dispatched to an unused processor. The AIX scheduler does a good job of optimizing between running threads on the same processor and rebalancing the load across processors; therefore, it is best not to use processor binding.

Resource sets

Since AIX 5.2, you can also use the Resource Set API to restrict the resources used by a process to a resource set. A resource set is a bundle of system resources that can consist of one or more processors, one or more memory pools, or a mixture of memory pools and processors. Just as with processor binding, the AIX scheduler does a good job of optimizing between running threads on the same processor and rebalancing the load across processors; therefore, you should not directly use resource sets to restrict a process to a particular processor or group of processors.

Many improvements have been made regarding processor affinity to improve performance, and workload management has been introduced. Therefore, binding processes to processes or resource sets is hardly of benefit any more in the AIX 5 and AIX 6.1 operating systems.

Logical partitioning and Micro-Partitioning

Logical partitioning (LPAR) is a feature that is available on POWER5 (or newer) processor-based systems that run the AIX 5 (or newer) operating systems. This capability was introduced to create a partition of a computer's processors, memory and hardware-resources environment such that each partition can be operated not only independently but also with its own operating system and applications.

Dynamic logical partitioning (dynamic LPAR or DLPAR) extends the capability of LPAR by providing the ability to attach and detach a managed system's resources logically to and from a logical hardware partition's operating system without rebooting. You can release processor, memory and I/O adapters into a free pool that has been acquired from the same pool, or you can move them to another partition dynamically.

From a performance perspective, LPARs and dynamic LPARs behave as if they are hardware partitioning, yet they rquire minimal overhead. However, it is a good practice to try to create an LPAR or dynamic LPAR partition based on processors belonging to the same dual-chip module (DCM) or multichip module (MCM) — because it can increase the data locality of the Oracle data in shared cache and minimize the memory-access latency. For more information on affinity and migration statistics, consult two books that are listed under the IBM Redbooks™ section of the **Resources** section of this paper (*AIX 5L Differences Guide Version 5.3 Edition* and *Partitioning Implementations for IBM eServer pSeries Servers*) regarding LPAR and dynamic LPAR configuration with MCM affinity.

Micro-Partitioning technology is a feature of the advanced IBM PowerVM™ (formerly know as IBM Virtualization Engine™) and is only available on POWER5 and POWER6 processor-based systems. This feature allows multiple partitions to share the processing power of a set of physical processors. A partition that uses the Micro-Partitioning feature can be assigned as little as 1/10th of a physical processor's resource. In most cases, a shared-processor pool that contains multiple physical processors is shared

among the partitions, and you can configure a partition to cede its processor resource for a short period of time when idle. In this way, spare capacity can be reallocated among the partitions (with a granularity of 1/100th of a processor, within 10 milliseconds). This can be beneficial for the performance of a big high-priority workload that shares a server with small lower-priority jobs that run in different partitions.

PowerVM requires only 1/10th of a processor entitlement to configure a partition, allowing up to 10 virtual processors (these could even be from 10 different partitions) on a single physical processor. However, when 10 partitions share a single physical processor, an increased latency results from dispatching the 10 virtual processors on a single physical processor. If the workloads on these 10 partitions do not have sustained high load levels and do not have strict response-time criteria, then those workloads can share a single processor by using a small fraction of a processor. However, if the load increases, response time goes up. Therefore, it is better to use proper sizing tools such as IBM Workload Estimator (WLE) to size processor capacity accurately for partitions. For workloads such as Oracle, depending on application performance requirements such as response time and load levels, it is recommended that you use a higher fraction processor entitlement, which increases the entitlement if response time needs to improve. (Refer to the IBM PowerVM Redbook documents that are listed in the **Resources** section of this paper for details on how to select the proper number of virtual processors, processor folding and SMT snooze-delay tuning values.) In cases where you need to allocate a whole processor to a partition, there are still benefits in using shared-processor LPAR (SPLPAR) instead of a dedicated processor partition, because sharing processors maximizes processor utilization in a consolidated environment. You need to study each case carefully to decide if the Micro-Partitioning feature, SPLPAR or dedicated LPAR best suit your workload-performance requirements. Using SPLPAR uncapped mode creates benefits by giving up cycles when not needed and from getting more cycles than entitled cycles if free cycles are available in the shared-processor pool.

Micro-Partitioning configuration and tuning, as well as related features (such as virtual I/O and virtual local area network [virtual LAN]), can be complex and are covered in the *Advanced POWER Virtualization on IBM eServer p5 Servers: Introduction and Basic Configuration* and *Advanced POWER Virtualization on IBM eServer p5 Servers Architecture and Performance Considerations* Redbook documents that are listed in the **Resources** section of this paper.

Oracle Database 10*g* and 11g are capable of detecting a change in the number of processors that are dynamically added to or removed from an LPAR, and will change the CPU_COUNT variable accordingly. Subsequent operations, for example parallel query, can take advantage of the additional processor resources after increasing the number of processors. This works well in a Micro-Partitioning environment, as long as the capacity that is available to the LPARs is not excessively limited.

For more information on LPARs and the Micro-Partitioning feature, consult the IBM Redbook documents that are listed in the **Resources** section of this paper (*AIX 5L Differences Guide Version 5.3 Edition* and *Partitioning Implementations for IBM eServer pSeries Servers*) regarding LPAR and dynamic LPAR configuration.

Tools to monitor processor performance

The AIX operating system provides commands to help investigate and solve problems where the system is processor-bound or suffers from abnormal processor usage. These commands include: **vmstat**, **iostat** and **sar**.

The vmstat command

The first tool to use when monitoring processor performance is the **vmstat** command, which quickly provides compact information about various system resources and their related performance problems. The vmstat command reports statistics about kernel threads in the run and wait queue, memory, paging, disks, interrupts, system calls, context switches and processor activity.

The reported processor activity is a percentage breakdown of user mode, system mode, idle time and waits for disk I/O operations. A new vmstat command option -I displays the threads that are waiting on raw I/O to complete, as well as the number of file pages that are paged in and out per second.

Figure 4 shows the output of the **vmstat -I 2** command:

```
# vmstat -I 2

  kthr     memory              page              faults        cpu
-------- ----------- ------------------------ ------------ -----------
 r  b  p   avm   fre  fi  fo  pi  po  fr sr  in  sy  cs   us sy id wa
 1  0  0 1160603 235   4   8   0   0   0  0 351 1032 1034 23  8 68  1
 0  0  0 1160599 235   0   0   0   0   0  0 275 1436 674   6  3 91  0
 0  0  0 1160618 235   0   6   0   0   0  0 337 1091 739   4  3 93  0
 0  0  0 1160599 235   0   6   0   0   0  0 285  287 644   1  2 97  0
 0  0  0 1160602 235   0   0   0   0   0  0 310  554 689   2  8 90  0
 0  0  0 1160627 235   0   7   0   0   0  0 341  561 748   2  3 94  1
```

Figure 4. Output of the vmstat -1 2 command

Descriptions of the relevant columns are as follows:

- **kthr**: These columns represents kernel-thread state changes per second for the sampling interval.
  - **r**: This represents the average number of kernel threads that are runnable over the sampling period, which includes threads that are already running and threads that are waiting to run in the run queue. This is zero on an idle or a lightly loaded system. The higher the number, the busier the processor. A number that is consistently higher than the number of logical processors (two times the number of virtual processors, or physical processors when SMT is enabled) in the LPAR might indicate a processor bottleneck.
  - **b**: This represents the average number of kernel threads in the Virtual Memory Manager (VMM) wait queue. For example, threads might be waiting for journal file system (JFS) or enhanced journal file system (JFS2) I/O operations to complete.
  - **p**: This represents the number of threads waiting on raw I/O operations (bypassing JFS or JFS2) to complete.
- **page**: This set of columns represents information about page faults and paging activity. These are averaged over the interval and given in units per second.
  - **fi and fo**: These columns show the number of file pages that were paged in and out per second.
- **cpu**: This set of columns represents a breakdown of the processor-time percentage. For multiprocessor systems, processor values are global averages among all active processors. Also, the I/O wait state is defined system-wide, not per processor.
  - **us**: This represents the average percentage of processor time running in user mode.
  - **sy**: This represents the average percentage of processor time running in system mode.
  - **id**: This represents the average percentage of time that processors were idle and the system did not have an outstanding disk I/O request.

- **wa**: This represents the average percentage of processor time that the processors were idle while the system had an outstanding disk I/O request. This value might be inflated if the actual number of I/O requesting threads is less than the number of idling processors.

The total of the last four columns (us, sys, id and wa) must equal to 100 percent, or very close. For dedicated processor LPARs, if the sum of user and system (us and sy) processor-utilization percentages consistently approaches 100 percent, there might be a processor bottleneck. For SPLPARs, high us plus sys values do not necessarily indicate an existing processor constraint because these values are based on active processors and do not take into account any processors that might have been folded by AIX because of low overall processor demand. Beginning in AIX 5.3, additional columns are included in the vmstat report to provide additional information for SPLPARs (see Figure 5).

```
$ vmstat 2 2

System configuration: lcpu=8 mem=8191MB ent=2.00

kthr     memory             page               faults           cpu
----- ------------- ------------------ -------------- ----------------------
r   b    avm     fre re pi po fr sr cy   in     sy   cs us sy id wa  pc    ec
1   0 444719 1721454  0  0  0  0  0  0  324  37751 4597  2  3 95  0  0.13  6.5
2   0 444728 1721445  0  0  0  0  0  0  211 141717 4298 11  9 79  0  0.48 23.9
```

Figure 5. AIX 5.3 vmstat listing that includes additional information for shared-processor partitions

Descriptions of the additional columns are as follows:

- **pc:** Amount of physical processors that is consumed. This is displayed only if the partition runs with shared processor. In this example, for the last interval, 48 percent of the capacity of one physical processor is consumed.
- **ec:** The percentage of entitled capacity that is consumed. This is displayed only if the partition runs with shared processor. In this example, for the last interval, 23.9 percent of the entitled capacity is being consumed. Note, that in this example the entitled capacity (ent) is 2.00; 0.48 of a processor (pc) equals 23.9 percent (ec) of 2.00 (ent).

The iostat command

The **iostat** command (see Figure 6) is used primarily to monitor I/O devices, but also provides processor utilization. As with vmstat, iostat can display the breakdown of processor usage.

```
#iostat 2

tty:      tin         tout   avg-cpu:  % user  % sys  % idle  % iowait
          0.0         20.9                0.6   10.3   88.8       0.3

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.5       4.1       1.0          0         8
hdisk1           0.0       0.0       0.0          0         0
```

Figure 6. Results from an iostat listing

The four columns of the average processor usage must total about 100 percent. If the sum of user and system (us and sy) processor usagen consistently approaches 100 percent, there might be a processor bottleneck. For SPLPARs, high us plus sys values do not necessarily indicate a processor

constraint, because these values are based on active processors and do not take into account any processors that might have been folded by AIX because of low overall processor demand.

The sar command

The **sar** command generates two kinds of reports. The first report type will display global-system statistics over time. The second type will display process statistics. The sar command can provide queue and processor statistics just as vmstat and iostat do. However, it has two additional features:

- Each sample has a leading time stamp. Thus, an overall average appears after the samples.
- The -P option generates per-processor statistics and global averages for all processors.

Figure 7 shows a typical report from a four-way POWER processor-based system with SMT enabled and per-processor statistics (in this case, two samples with one-second intervals are taken).

```
 #sar -P ALL 1 2

AIX stsf 3 5 00CD242F4C00    9/16/04

System configuration: lcpu=4

19:50:18 cpu    %usr    %sys    %wio   %idle   physc
19:50:19  0        1       2       1      96    0.59
          1        0       0       0     100    0.35
          2        9      19       0      72    0.70
          3        0       0       0     100    0.31
          -        4       7       0      89    1.95
19:50:20  0        4       3       0      92    0.66
          1        0       0       0     100    0.35
          2       22      15       1      62    0.72
          3        0       0       0     100    0.28
          -        9       7       0      83    2.00

Average   0        3       3       0      94    0.62
          1        0       0       0     100    0.35
          2       15      17       0      67    0.71
          3        0       0       0     100    0.29
          -        7       7       0      86    1.98
```

Figure 7. Sar report for a four-way POWER processor-based system with SMT enabled and per-processor statistics

Notice that the sar command reports the logical processor statistics, not physical processor statistics, because SMT was enabled.

The sar command does not report cumulative activity since the last system boot.

The lparstat command

The lparstat command is used to display the resources that are allocated to a partition (see Figure 8).

```
$ lparstat -i
Node Name                                  : oranode
Partition Name                             : oranode
Partition Number                           : 8
Type                                       : Dedicated-SMT
Mode                                       : Capped
Entitled Capacity                          : 6.00
Partition Group-ID                         : 32776
Shared Pool ID                             : -
Online Virtual CPUs                        : 6
Maximum Virtual CPUs                       : 6
Minimum Virtual CPUs                       : 1
Online Memory                              : 17408 MB
Maximum Memory                             : 17408 MB
Minimum Memory                             : 1024 MB
Variable Capacity Weight                   : -
Minimum Capacity                           : 1.00
Maximum Capacity                           : 6.00
Capacity Increment                         : 1.00
Maximum Physical CPUs in system            : 64
Active Physical CPUs in system             : 32
Active CPUs in Pool                        : -
Shared Physical CPUs in system             : -

Maximum Capacity of Pool                   : -
Entitled Capacity of Pool                  : -
Unallocated Capacity                       : -
Physical CPU Percentage                    : 100.00%
Unallocated Weight                         : -
```

Figure 8. An lparstat listing of the resources that are allocated to a partition

Notice that the lparstat command indicates the physical resources (processor and memory) that are allocated to the partition and other attributes of the partition. The meaning of tests attributes is covered in the *Advanced POWER Virtualization on IBM eServer p5 Servers: Introduction and Basic Configuration* and *Advanced POWER Virtualization on IBM eServer p5 Servers Architecture and Performance Considerations* Redbooks document that is listed in the **Resources** section of this paper.

Tuning memory

This section addresses the Oracle process memory footprint, recommended VMM tuning, 16 MB pages and 64 KB pages, as well as tools to monitor memory usage.

Oracle process memory footprint

The AIX operating system provides a robust environment for running processes with a wide variety of execution characteristics. By taking advantage of the many variables and features to specifically tune the process environment for the Oracle application, memory usage of each Oracle process is reduced.

You can use the AIXTHREAD_SCOPE environment variable for control if a process runs with process-wide contention scope (the default) or with system-wide contention scope. When using system-wide contention scope, there is a one-to-one mapping between the user thread and a kernel thread. On UNIX® systems, Oracle is primarily a multiprocess, single-threaded application. One of the mechanisms that Oracle uses to enable this multiprocess system to operate effectively is the AIX post and wait mechanism: thread_post(), thread_post_many() and thread_wait(). This post and wait mechanism operates most effectively with the Oracle application when using system-wide contention scope (AIXTHREAD_SCOPE=S). As of AIX V5.2, system-wide contention scope also significantly reduces the memory that is required for each Oracle process. For these reasons, always export AIXTHREAD_SCOPE=S before starting all Oracle processes. In Oracle Database 10*g* a newer, system-wide contention scope is set internally when the instance and user processes are started. It is best to export AIXTHREAD_SCOPE=S before starting all Oracle processes.

Process memory is divided into different groups based on how it is accessed. The **.text** section contains the program text and read-only (constant) data. The **.data** section contains read and write data. Because the .text section is read-only, it can be shared across all the processes that run the same binary. The writeable .data has to be unique to each process. AIX and the IBM XL C/C++ compiler allow constant pointers to be placed in the .text section. When the –qroptr compiler option is in effect, the compiler places constant pointers, virtual function tables and virtual type tables in read-only storage (.text section). This can improve runtime performance, save storage and provide shared access, but code that attempts to modify a read-only constant value generates a memory error. Oracle 10gR2 (10.2.0.2) or later takes advantage of this feature, which is transparent to Oracle users and can reduce the memory that each Oracle process requires by approximately one megabyte.

Note: If you use Oracle 10gR2 (10.2.0.1) or earlier, Oracle provides a script and procedure for relinking the Oracle binary to take advantage of this new feature. Refer to Oracle MetaLink note 259983.1 provided through the Oracle MetaLink Web site that is listed in the **Resources** section of this paper.

VMM tuning

Paging is one of the most common reasons for performance problems. Paging means that the VMM page-replacement algorithm takes computational pages away to free memory and then writes those pages into the system paging space. You can attain better performance when page references are found in the real memory. One way to do this is to pin the application's pages so that the kernel cannot swap them out. To pin Oracle SGA, set the vmo v_pinshm option to **1**. However, the potential risk that is involved by enabling v_pinshm is that the system, when running short on memory, can either hang or crash. The more appropriate way to assure that database memory is not paged out is to apply the recommended VMM page-replacement tuning that is mentioned in the following table, instead of pinning the SGA.

The VMM page-replacement tunable settings shown in Table 2 allow the system to use up to 90 percent of its real memory for file caching, but favors computational pages instead of file pages. Page replacement might steal computational pages after the percentage of computational pages in memory exceeds 97 percent; that is, 100 percent minus the minperm% value.

AIX 5.3 defaults	**Recommended values**
Minperm% = 20	minperm% = 3
Maxperm% = 80	maxperm% = 90
maxclient% = 80	maxclient% = 90
strict_maxperm = 0	strict_maxperm = 0
strict_maxclient = 1	strict_maxclient = 1
lru_file_repage = 1	lru_file_repage = 0
page_steal_method = 0	page_steal_method = 1*

Table 2. Recommended values for VMM tunable parameters

* Changing the page_steal_method is an optional tuning method for systems that use only a small percentage of their memory for file caching. However, for systems that run Oracle RAC, when using the VMM settings shown in Table 2, it is highly recommended that you also set page_steal_method=1, to reduce the likelihood of Oracle oprocd node evictions. Enabling the page_steal_method requires a system reboot and, therefore, it is recommended that you enable it during scheduled maintenance.

Note**:** The tunable values shown in Table 2 are the defaults on AIX 6.1.

Multiple-page size support

Beginning with AIX 5.1, when running on IBM POWER4™ or POWER5 processors, POWER processor-based systems support two virtual page sizes: 4 KB (small or standard page) and 16 MB (large page). The POWER5+ processor and AIX Version 5.3 Maintenance Level 4 introduce support for two new virtual memory page sizes – 64 KB (medium page) and 16 GB (huge page). The POWER6 processor and AIX 6.1 also have support for 64 KB and 16 GB pages. However, the page sizes that a particular system supports varies, based on processor type. (See Table 3.)

Page size	Processor type	User configuration	Kernel
4 KB (small)	ALL	No	64 and 32
64 KB (medium)	POWER5+ (or later)	No	64 only
16 MB (large)	POWER4 (or later)	Yes	64 and 32
16 GB (huge)	POWER5+ (or later)	Yes	64 only

Table 3. Virtual memory page sizes

The new 64 KB pages are preferred over 16 MB pages, and it is recommended that you use them instead of 16 MB pages on systems that support both page sizes. This is because 64 KB pages provide most of the benefit of 16 MB pages, without the special tuning requirements of 16 MB pages. If the memory is not properly tuned for 16 MB pages, the performance can be worse. On systems where 64 KB pages are not supported, you can use 16 MB pages, but carefully monitor the memory usage on these systems to make sure that memory is used properly. Users who want to consider 16 MB pages should discuss this with AIX support or IBM consultant.

Note: Discussion of 16 GB pages is beyond the scope of this paper.

16 MB (large) pages

When using large pages to map virtual memory, the translation lookaside buffer (TLB) is able to map more virtual memory with a given number of entries, resulting in a lower TLB miss rate for applications that use a large amount of virtual memory. Additionally, when using large pages, there are fewer page boundaries, which improve the performance of prefetching. Both online transaction processing (OLTP) and data-warehouse environments can benefit from using large pages.

For applications such as Oracle, which typically exploits a large amount of virtual memory, using large page memory generally results in improved performance. Oracle can use large pages with three types of memory: shared memory (SGA), process data (**.data**) and the instruction text (**.text**). In most Oracle applications, SGA dominates virtual-memory usage and, consequently, most of the benefit of using large pages is achieved by using them for SGA. In some special Oracle applications, using large pages for .data and .text can provide some additional performance benefit. However, because of the granularity of allocation with large pages, each Oracle process might have a larger memory footprint. On systems where 64 KB pages are not supported and if you choose to use 16 MB pages, use them only for SGA, not for .text or .data.

In Oracle Database 10*g* (10.2.0.3 or earlier), when the Oracle initialization parameter LOCK_SGA is set to TRUE, Oracle requests large pages when allocating shared memory (that is, when the shmget() call has the SHM_LGPAGE flag set). If some or all of requested large pages are not available, Oracle pins regular 4 KB pages. For the AIX operating system to use large pages when allocating shared memory, the Oracle user ID must have CAP_BYPASS_RAC_VMM and CAP_PROPAGATE capabilities. Also, the AIX large page pool must be configured (as shown in the example that follows). When using large pages on an Oracle RAC database, where the srvctl command is used to start and stop the RAC instances, it is also necessary to set the CAP_BYPASS_RAC_VMM and CAP_PROPAGATE capabilities for the root user ID.

Large pages are always pinned and can not be used for standard memory. If the AIX operating system is configured to have a pool of large page memory configured, then this memory is unusable for allocation of standard memory, even if no other application currently uses large pages.

Note: Oracle 10gR2 (10.2.0.4) and Oracle 11gR1 (11.1.0.6) introduced support for 64 KB pages for SGA. That required optimization around Oracle initialization parameter LOCK_SGA. When LOCK_SGA is set to TRUE, pinning memory is no longer a request to the AIX kernel. If the locking-memory operation fails, Oracle does not start. Refer to the 64 KB (medium) pages section of this paper for detailed information. Using large page for SGA requires you to set the vmo option v_pinshm to **1**.

Example configuration for using large pages for the Oracle SGA

Give the Oracle user ID the CAP_BYPASS_RAC_VMM and CAP_PROPAGATE capabilities by following these steps:

1. Check the current capabilities by issuing the following command:

   ```
   # lsuser –a capabilities oracle
   ```

   Note: Only the root user can display the capabilities attribute.

2. Add the CAP_BYPASS_RAC_VMM and CAP_PROPAGATE capabilities to the list of capabilities that are already assigned to this user ID, if any:

   ```
   # chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE oracle
   ```

3. Configure the AIX large-page pool by calculating the number of large pages that are required for the SGA:

   ```
   num_of_large_pages = INT((total_SGA_size-1)/16MB)+1
   ```

4. Configure the number and size of large pages:

   ```
   # vmo -p -o lgpg_regions=num_of_large_pages -o lgpg_size=16777216
   ```

Example configuration for using large pages for .text and .data

These configuration steps are in addition to the steps that are required for using large page SGA:

1. Calculate the additional large pages required for .text and .data.
2. Start the Oracle instance and run the following command as root (see Figure 9):

```
# svmon –P <PID an Oracle process>
for example:
# svmon -P 552974
------------------------------------------------------------------------------
     Pid Command          Inuse      Pin    Pgsp  Virtual 64-bit Mthrd LPage
  552974 oracle           95447     4012       0    86224      Y     N     N

    Vsid      Esid Type Description            LPage  Inuse   Pin Pgsp Virtual
  181958  70000000 work default shmat/mmap         -  58243     0    0 58243
  1e197e  70000001 work default shmat/mmap         -  12663     0    0 12663
  1a17da        10 clnt text data BSS heap        -   9212     0    -     -
       0         0 work kernel segment            -   7821  3990    0  7821
   d08ad  90000000 work loader segment            -   4776     0    0  4776
  1f1a7f        11 work text data BSS heap        -   1924     0    0  1924
  121272  90020014 work shared library text       -    375     0    0   375
   21a82  8001000a work private load              -    138     0    0   138
    1a40  9001000a work shared library text       -    113     0    0   113
   d1a6d  80020014 work private load              -    107     0    0   107
       :
       :
```

Figure 9. Example of svmon usage

The row with an Esid of 10 corresponds to the .text segment, and the row with an Esid of 11 corresponds to the .data segment. When checking the .text segment for all the Oracle processes, notice that they each have the same Vsid. This is because they share this read-only segment. The .data section is private to each Oracle process so that the total large page requirement is dependant on the number of foreground and background Oracle processes when using large page data. The column labeled **Inuse** contains the number of 4 KB pages in that segment.

3. Calculate the number of additional pages for large page .text and .data:

```
num_of_additional_pages =
       INT((.text_pages+4095)/4096) + N*(INT(.data_pages+4095)/4096)
```

4. Using the **svmon** example output and assuming 25 foreground and background processes (N=25), then:

```
num_of_additional_pages =
       INT((9212+4095)/4096) + 25*(INT((1924+4095)/4096)) = 28
```

5. Add this number to the large pages needed for the SGA and execute vmo:

```
# vmo -p -o lgpg_regions=num_of_large_pages+num_of_additional_pages -o
lgpg_size=16777216
```

6. Edit the XCOFF file header in the oracle binary to enable it to use large page data:

```
# ldedit –b lpdata
```

7. Prior to starting the Oracle instance and listener, export LDR_CNTRL in the Oracle user ID that was used to start the instance and listener:

```
# export LDR_CNTRL=LARGE_PAGE_TEXT=Y@LARGE_PAGE_DATA=M
```

Note: This must be set in addition to editing the XCOFF file header to allow both large page text and large page data. Setting the LARGE_PAGE_DATA=M option allocates only enough large pages for the data segment up to the brk value. The @ character is used to separate the definition of multiple values to the LDR_CNTRL variable.

64 KB (medium) pages

Unlike 16 MB pages, 64 KB pages are fully capable of being paged and very easy to use. 64 KB pages also reduce TLB miss rate. It is expected that many applications can see performance benefits when using 64 KB pages rather than 4 KB pages. No system configuration is necessary to enable a system to use 64 KB pages. On systems that support 64 KB pages, the AIX kernel automatically configures 64 KB pages for the system and also dynamically and fully manages the size of the pool of 64 KB page frames on a system. AIX varies the number of 4 KB and 64 KB page frames on a system to meet demand on the different page sizes. You can use 64 KB pages for text, data, stack and shared memory regions. Extensive testing of 64 KB pages with Oracle Database was done during several benchmarks. They concluded that the performance improvement is in the range of one and four percent for text, data and stack, and is up to eight percent for shared memory.

Note: results can vary, depending on the workload; and the numbers provided reflect an OLTP workload.

64 KB page-size support for shared memory requires source code modification. Oracle Database 10gR2 (10.2.0.4) and 11gR1 (11.1.0.6) introduced support of 64 KB pages for SGA. Oracle Database users do not have to do any additional configuration to have SGA backed by 64 KB pages. Prior database releases cannot use 64 KB pages for SGA.

Because Oracle does not have a separate initialization parameter to specify the desired page size for SGA, the LOCK_SGA initialization parameter is optimized enough to take advantage of the best available pages to achieve optimal performance.

When LOCK_SGA is set to TRUE:

1. If sufficient 16 MB pages are available to meet the desired SGA size, SGA is backed by 16 MB pages. (**Note:** 16 MB pages are always pinned memory.)
2. Otherwise, if the system supports 64 KB pages, SGA is backed by 64 KB pages and the memory region is pinned. If the pinning memory operation fails, Oracle Database does not start. Check the alert log for a detailed error message.
3. Otherwise, SGA is backed by 4 KB pages and the memory region is pinned. If the pinning memory operation fails, Oracle does not start. Check the alert log for a detailed error message.

When LOCK_SGA is set to FALSE (default case):

1. If the system supports 64 KB pages, SGA is backed by 64 KB pages.
2. Otherwise, SGA is backed by 4 KB pages.

Because of the complex nature of pinning, if handled incorrectly, general purpose pages (such as 4 KB and 64 KB) can lead to severe problems such as system hang or system crash. Weighing between the performance gain and the potential risk of an unstable system, it is no longer recommended that you pin SGA if the SGA is backed by 64 KB pages or 4 KB pages. To prevent paging out of Oracle processes' pages, consider the recommendations mentioned in the **VMM tuning** section of this paper.

To summarize 16 MB and 64 KB page size support for Oracle SGA, set LOCK_SGA to TRUE only when the intention is to use 16 MB pages for SGA. Ensure that you have sufficient free 16 MB pages before Oracle starts. (You can use **svmon –G** or **vmstat –P ALL** to see available free memory.) SGA that is backed by 16 MB pages requires an explicit locking operation; therefore, the Oracle user ID must have CAP_BYPASS_RAC_VMM capabilities. When LOCK_SGA is FALSE (which is the default), then, 64 KB pages (if supported) is used for Oracle SGA. The performance improvement in both cases is significant, compared to 4 KB pages, but it is recommended that you use 64 KB pages (if supported) for SGA because this does not require any configuration and is transparent to Oracle users. Hence, do not set LOCK_SGA to TRUE if 64 KB pages are supported.

> **Note:** Starting with AIX 6.1, running on POWER6 processor-based systems, VMM can dynamically promote pages to a larger page size. By default, VMM does not do page promotion, but you can enable this by setting the **vmo pspa** parameter. This page promotion is completely transparent to the application and is done without the need of user intervention.

Oracle Database users can specify page sizes to use for three regions (text, data and stack) of the Oracle binary. These page sizes are configurable with an environment variable or by setting the XCOFF header of the Oracle binary by using the AIX **ld** or **ldedit** command.

Region	ld or ldedit option	LDR_CNTRL environment variable	Description
Text	-btextpsize	TEXTPSIZE	Oracle binary text
Data	-bdatapsize	DATAPSIZE	Initialized data, bss, and heap (Oracle PGA)
Stack	-bstackpsize	STACKPSIZE	Initial thread stack

Table 4.

Users can specify different page sizes for each of the three regions. If an unsupported page size is specified, the kernel uses the next smallest supported page size. If there is no page size smaller than the specified page size, the kernel uses the 4 KB page size.

- To use 64 KB pages for Oracle text, data and stack, edit the Oracle XCOFF header file
  ```
  $ ldedit -btextpsize=64K -bdatapsize=64K -bstackpsize=64K oracle
  ```
- Or, use LDR_CNTRL before starting Oracle:
  ```
  $ export LDR_CNTRL=DATAPSIZE=64K@TEXTPSIZE=64K@STACKPSIZE=64K
  ```

Tools to monitor memory usage

This section discusses tools to monitor memory usage, such as the **vmstat** and **svmon** commands.

vmstat

In addition to monitoring processor usage, the **vmstat** command also helps monitor memory usage.

Use the **vmstat –l** command to monitor the allocated and free large pages (see Figure 10).

```
$ vmstat -l

System configuration: lcpu=2 mem=7424MB

kthr    memory              page              faults        cpu     large-page
----- ----------- ------------------------ ------------ ----------- -----------
 r  b    avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa   alp   flp
 1  1 433591 338295   0   0   0   0    0   0  85  382 340  0  0 99  0     0     0
```

Figure 10. Monitor the allocated and free large pages

In this vmstat output, the alp column indicates the number of allocated large pages, and the flp column is the number of free large pages.

Use the **vmstat –P** or **vmstat –p** command to display VMM statistics for a specified page size or for all supported page sizes (see Figure 11).

```
$ vmstat -P ALL

System configuration: mem=20480MB

pgsz            memory                           page
----- -------------------------- ------------------------------------
           siz      avm      fre    re    pi    po    fr     sr    cy
   4K  2545696   523134  1961650     0     0     0     0      0     0
  64K   142974     3198   139776     0     0     0     0      0     0
  16M      100        0      100     0     0     0     0      0     0
$

$ vmstat -p ALL

System Configuration: lcpu=8 mem=20480MB

kthr    memory              page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b    avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 1  1 574302 4198066   0   0   0   0    0   0  56  305 255  0  0 99  0

  psz   avm   fre  re  pi  po  fr   sr  cy     siz
   4K 523135 1961649   0   0   0   0    0   0 2545696
  64K  3198 139776   0   0   0   0    0   0  142974
  16M     0   100   0   0   0   0    0   0     100
$
```

Figure 11. Displaying VMM statistics for a specified page size or for all supported page sizes

svmon

The **svmon** command is useful for looking at the memory that is used by a process or group of processes. The **–P <pid>** option that is used in the **16 MB (large) pages** section of this paper shows the memory usage for a specific process. The **–C oracle** option shows the memory usage for all Oracle processes that are running on the system. Only root authority can use the **svmon** command.

You can use the **svmon** command to display page-size usage across the system. Most **svmon** commands have been enhanced to provide a per-page size breakdown of statistics. For example, to display global statistics about each page size, you can use the **svmon -G** command (see Figure 12):

```
# svmon -G
               size      inuse       free        pin    virtual
memory      5242880    1045204    4197676     810112     574629
pg space     393216       2074

               work       pers       clnt
pin          400512          0          0
in use       574629        178      60797

PageSize   PoolSize      inuse       pgsp        pin    virtual
s    4 KB          -     584404       2074     367392     523429
m   64 KB          -       3200          0       2070       3200
L   16 MB        100          0          0        100          0
#
```

Figure 12. Displaying global statistics about each page size

ps

Use the **ps** command to monitor the page sizes that are used for a process's data, stack and text. Three new columns are displayed when the –Z option is specified to ps command (see Figure 13):

- *DPGSZ* - process's data page size
- *SPGSZ* - process's stack page size
- *TPGSZ* - process's text page size

```
$ ps -Z
    PID    TTY  TIME DPGSZ SPGSZ TPGSZ CMD
 241822 pts/22  0:00    4K    4K    4K ksh
 405808 pts/22  0:00    4K    4K    4K ps
 516590 pts/22  0:06    4K    4K    4K ksh
$
```

Figure 13. Monitoring the page sizes that are used for a process's data, stack and text

pagesize

Use the **pagesize** command to determine the system's supported page sizes (see Figure 14).

```
$ pagesize -af
4K
64K
16M
16G
$
```

Figure 14. Determining the system's supported page sizes

Tuning I/O performance

The processors and memory can be tuned to peak efficiency for Oracle, but if the I/O processes are not tuned properly, the Oracle database can still show performance problems. For example, if the POWER processor is set to run SMT, but the I/O subsystem only allows one process at a time, the system will run as though it only has a single-threaded processor. The solution to this problem is to enable asynchronous I/O (AIO), which allows several I/O processes to run concurrently. This and other preferred I/O tuning practices are discussed in this section.

AIX V6.1

Starting with AIX 6.1, the list of system tunable parameters and details on how to use them is no longer available in the AIX documentation or man pages. For the tunable description of **vmo**, **ioo**, **schedo**, **raso**, **no** and **nfso** usage, there is a new –h <tunable> option. For example, to display the documentation for aio_maxservers execute ioo –h aio maxservers.

In AIX 6.1, some of the tunable options are restricted and can only be displayed with the –F option of vmo, ioo, schedo, raso, no and nfso. It is advisable not to change the restricted tunables unless recommended by AIX development or support.

Asynchronous I/O

The asynchronous I/O feature allows a program to initiate I/O and to continue running useful work, while other I/O operations are carried out in parallel by the operating system. Because Oracle often requires multiple server and user processes at the same time, it takes advantage of asynchronous I/O feature to overlap program execution with I/O operations. The asynchronous I/O feature is used with Oracle on the AIX operating system to improve system performance.

In AIX Version 5 (and later), two asynchronous I/O subsystems are supported, the original asynchronous I/O subsystem, called *legacy asynchronous I/O*, and the Portable Operating System Interface (POSIX) compliant AIO called POSIX AIO. The two types of asynchronous I/O subsystems differ in the way the asynchronous I/O subsystem API is defined. Their performance characteristics are the same. All Oracle releases up to, and including, the current release, Oracle Database 11*g*, use legacy asynchronous I/O.

Asynchronous I/O servers

In the AIX operating system, kernel processes are used for asynchronous I/O to a file system (JFS, JFS2 or General Parallel File System [GPFS]). Referred to as *asynchronous I/O servers*, these kernel processes handle each asynchronous I/O request. When performing asynchronous I/O to raw devices with the asynchronous I/O fast path enabled, asynchronous I/O servers are not used. Therefore, the following discussion on tuning minservers and maxservers does not apply when the asynchronous I/O fast path is enabled and being used for raw devices. The asynchronous I/O fast path is enabled by default.

When doing asynchronous I/O to a file system, each asynchronous I/O operation is tied to an asynchronous I/O server. Thus, the number of asynchronous I/O servers limits the number of concurrent asynchronous I/O operations in the system. The initial number of servers that are started at boot time is determined by the minservers parameter. As more concurrent asynchronous I/O

operations occur, additional asynchronous I/O servers are started, up to the maxservers value,. If using Oracle with data files on a file system, the default values for minservers and maxservers in AIX 5.3 are too small and need to be increased. The values that are provided for minservers and maxservers are per processor. On AIX Version 5, asynchronous I/O must be activated to make it available, where as on AIX 6.1, asynchronous I/O is enabled by default and is not shown active until used.

To fine-tune the number of asynchronous I/O servers, adjust the initial value of maxservers (limit the maximum number of servers to 10 times the number of disks that are to be used concurrently, but no more than 80) and monitor the performance effects on the system during periods of high I/O activity. The downside of too many asynchronous I/O servers is the increased memory and processor overhead of additional processes.

For details on tuning maxservers for Oracle and GPFS, refer to the *GPFS for AIX FAQs* found in the **Resources** section of this paper.

AIX 6.1 has a new implementation of the asynchronous I/O kernel extension and, in most cases, does not require any additional tuning. The asynchronous I/O dynamic tunables are now set up by using the ioo command. The new tunables are aio_maxservers, aio_minservers and aio_server_inactivity for the legacy asynchronous I/O subsystem.

For AIX Version 5, use the following command shown in Figure 15 to determine the current minservers and maxservers settings:

```
#lsattr –El aio0

Sample output

autoconfig available   STATE to be configured at system restart   True
fastpath   enable      State of fast path                          True
kprocprio  39          Server PRIORITY                             True
maxreqs    40960       MAXIMUM number of REQUESTS                  True
maxservers 256         MAXIMUM number of servers per cpu           True
minservers  20         MINIMUM number of servers                   True
```

Figure 15. Using the lsattr command to determine the current minservers and maxservers settings

On AIX Version 5 to change the minservers and maxservers, use the **smit** system management interface or run the following command from the root user:

```
#chdev -P -l aio0 -a maxservers='m' -a minservers='n'
```

Note: Use the –P option to cause the change to remain in effect after a reboot. If the asynchronous I/O subsystem is already in the available state, then the –P option is required, and the system must be rebooted for this change to take effect. A reboot is not required if using AIX V5.3 TL05 or later. With AIX V5.3 TL05, the **aioo** command is used to change these tunables on a running system; it does not make the changes persistent across boots.

AIX V5.3 TL05 introduced a new fsfastpath tunable, which specifies how the I/O requests for files that are opened with concurrent I/O mode in aJFS2 are managed. If enabled, it initiates asynchronous I/O requests directly to LVM or disks. This tunable is not enabled by default. To enable fsfastpath in AIX V5.3 TL05, run the following command as root user:

```
# aioo –o fsfastpath=1
```

In AIX V5.3 TL05, use the following command to determine the value of minservers, maxservers, maxreqs and fsfastpath:

```
# aioo -a
```

The fsfastpath tunable is nonpersistent across boots in AIX 5L. In AIX 6, this tunable is restricted and is enabled by default.

On AIX 6.1, use **ioo –a | grep aio** to determine the current values of the following asynchronous I/O parameters:

```
aio_active = 0
aio_maxreqs = 65536
aio_maxservers = 30
aio_minservers = 3
aio_server_inactivity = 300
posix_aio_active = 0
posix_aio_maxreqs = 65536
posix_aio_maxservers = 30
posix_aio_minservers = 3
posix_aio_server_inactivity = 300
ioo -a -F | grep path
aio_fastpath = 1
aio_fsfastpath = 1
posix_aio_fastpath = 1
posix_aio_fsfastpath = 1
```

In AIX 6.1, it is recommended that you use the default values for the asynchronous I/O tunables, unless it is recommended by AIX development or support to modify their values.

The initialized asynchronous I/O server processes appear in the **ps** output command with the name *aioserver* for legacy asynchronous I/O servers and *posix_aioserver* for POSIX asynchronous I/O servers.

AIX Version 5 provides a –A option to the **iostat** command, which gives useful information for tuning the number of asynchronous I/O servers and maxreqs. The asynchronous I/O report has the following columns:

- **avgc**: The average global asynchronous I/O request count per second for the specified interval
- **avfc**: The average fastpath request count per second for the specified interval
- **maxgc**: The maximum global asynchronous I/O request count since the last time this value was fetched
- **maxfc**: The maximum fastpath request count since the last time this value was fetched
- **maxreqs**: The maximum asynchronous I/O requests allowed

Use the avfc and maxfc columns for fastpath asynchronous I/O requests. These requests do not use an asynchronous I/O server.

maxreqs

As well as tuning minservers and maxservers for asynchronous I/O, maxreqs also needs to be tuned. The maxreqs value specifies the maximum number of asynchronous I/O requests that are waiting for processing in the system at one time. This value includes the asynchronous I/O operations that are in progress in asynchronous I/O servers, as well as the requests that are queued but not in progress.

If the maxreqs value is too low, then the following Oracle warning message might be seen:

```
Warning: lio_listio returned EAGAIN
```

This warning message indicates a need to increase the maxreqs value. When performing file-system I/O, a maxservers value that is too low can also indirectly cause this warning message. A low maxservers value can limit the rate at which asynchronous I/O requests are completed, resulting in an increase in the outstanding I/O requests during a period of heavy I/O activity. The same is true for a performance problem in the physical I/O subsystem. For example, if the data is poorly distributed across the disks, data access can be slow.

File system and raw I/O

Oracle data files can be located in file systems or raw devices. The AIX operating system has special features to enhance the performance of file system I/O for general-purpose file access. These features include read ahead, write behind, and I/O buffering, which can provide huge performance benefits for many applications. However, Oracle employs its own I/O optimizations and buffering that, in most cases, are redundant to those provided by the AIX file systems. Oracle uses buffer-cache management that involves data blocks in shared memory. The AIX operating system uses virtual memory management (VMM), which involves data that is buffered in virtual memory. If both separately try to manage data caching, it results in wasted memory, processor overhead and suboptimal performance.

It is generally better to allow Oracle to manage I/O buffering because it has information regarding the context in which the data is referenced and, therefore, it can better optimize memory usage for I/O buffering. Oracle manages buffered data in the buffer cache and disk access, based on the type of SQL query that accesses the data. Examples of this include the Oracle use of the recycle pool when accessing data that it expects not to be referenced again, and its use of the db_file_multiblock_read_count parameter to increase the read block size when performing large table or index scans.

When the file-system I/O buffer cache is used, I/O data is copied into the buffer cache, and then a second copy is written into the application I/O buffer. This increases the path length for each I/O operation to a file system as compared to raw I/O.

For data integrity, Oracle opens all data files with the O_DSYNC flag. When the O_DSYNC flag is used, it guarantees that all the data and metadata that are required to access the data have been safely written to nonvolatile storage when a write operation completes. This ensures that the data can be accessed even after a system crash. The O_DSYNC flag also disables the file-system write operation behind algorithms.

For some special database operations, the I/O operation bypasses the Oracle buffer cache and can benefit from using the file-system I/O buffer cache. These special operations include: reads and writes to the TEMPORARY table space, data in NOCACHE large objects (LOBs), and parallel-query slaves that read data.

Raw I/O

When accessing Oracle data files on raw devices, fewer AIX parameters are available to tune than when using data files on file systems. This is because, with raw I/O, the file system and VMM layers are bypassed, resulting in less path length per physical I/O operation. If Oracle can efficiently buffer I/O data for the workload, such that the number of physical I/O operations does not increase, this improves performance. When using raw devices, contention on inode locks is also avoided.

When using raw devices with Oracle on the AIX operating system, the devices are either raw logical volumes or raw disks. When using raw disks for I/O operations, the logical volume manager (LVM) layer is bypassed. When using raw hdisks and AIX5.3 TL05+, you must apply APAR IY92037. The LVM allows the grouping of disks into volume groups (VG), and then defines logical volumes (LV) where the size and number of LVs is independent of the size and number of disks, within the limitation of the total space in the VG. Because of this capability, the use of raw LVs is recommended for Oracle data files, unless the Oracle Automatic Storage Management (ASM) is used. Oracle ASM includes the capability to create Oracle data files, which do not need to be mapped directly to the size and number of disks. With Oracle ASM, using raw disks is preferred.

When using raw devices with Oracle RAC, which requires shared access to the data from all instances, the devices must be raw logical volumes in concurrent volume groups or raw shared disks. The use of concurrent volume groups requires High-Availability Cluster Multi-Processing for the AIX operating system (IBM HACMP® for AIX). For configurations without HACMP clustering, which requires shared disks, use Oracle ASM to manage the mapping of data files to the disk group.

pbufs

When accessing LVs in a buffer in pinned memory, use **pbuf** to contain information that is associated with each pending disk I/O request. If an I/O request is made to an LV and there are no free pbufs, then the I/O request is delayed. Use the following command, from the root user, to determine blocked I/O requests:

```
AIX V5.3 and newer:          /usr/bin/vmstat –v|grep “blocked with no pbuf”
```

If this command returns a value other than **0**, there might be a benefit from increasing the number of pbufs.

Use the following command, from the root user, to determine the current number of pbufs:

```
AIX 5.3:                     /usr/sbin/ioo –a |grep pv_min_pbuf
AIX 6:                       /usr/sbin/ioo –a –F | grep pv_min_pbuf
```

Use the following command, from the root user, to change the number of pbufs:

```
AIX V5.3 and newer:          /usr/sbin/ioo –p –o pv_min_pbuf=<number of pbufs
per PV>
```

Note: The –p flag makes the change stay in effect after a reboot.

AIX 5L provides a command, **lvmo**, which you can use to fine-tune the usage of pbufs. Use the lvmo command to tune the number of LVM pbufs on a per-volume-group basis. The **lvmo –a –v <vgname>** command shows pervg_block_iocount, which is, the number of I/O requests that were

blocked because of a lack of free pbufs on a per-volume-group basis. The lvmo command allows the following parameters:

- **pv_pbuf_count**: The number of pbufs that are added when a physical volume is added to the volume group.
- **max_vg_pbuf_count**: The maximum number of pbufs that can be allocated for the volume group. For this value to take effect, the volume group must be varied off and varied on again.

Monitor the system performance before and after increasing the pbufs to make sure that there are no adverse effects. The pbufs are in pinned memory; thus, making them unnecessarily large reduces the availability of memory for other purposes.

The lvm_bufcnt command

The lvm_bufcnt command specifies the number of LVM buffers that are available for raw physical I/O operations. When the application-transfer size exceeds the LTG size, the I/O request is broken into a unit of up to lvm_bufcnt I/Os, each having a size of 128 KB. If the application-transfer size exceeds the size of one unit, then the I/O operation is done in multiple units, and the I/O operation for each unit must complete before the I/O operation for the next unit can begin. The default lvm_bufcnt value is nine and the maximum value is 64. For optimal performance, when the LTG size cannot be increased to the application transfer size because of limitations in the maximum transfer size in the disk, make sure the product of the lvm_bufcnt and the LTG size is greater than, or equal to, the application transfer size. This ensures that the I/O operation can complete in one unit.

Use the following command from the root user to display lvm_bufcnt:

```
/usr/sbin/ioo –a|grep lvm_bufcnt
```

Use the following command from the root user to set lvm_bufcnt.

```
/usr/sbin/ioo –p –o lvm_bufcnt=<number of lvm_bufcnt>
```

Note: The –p flag makes the change stay in effect after a reboot.

Concurrent I/O

The AIX operating system provides a feature for JFS2 called concurrent I/O. Concurrent I/O includes the performance benefits previously available with direct I/O, plus the additional performance benefit of eliminating contention on the inode lock. You can use concurrent I/O in applications such as Oracle that do their own data serialization. These applications do not require the functions of the AIX operating system inode lock.

As with direct I/O, concurrent I/O can be specified for the entire file system by specifying the **cio** mount option, or in the application for a particular file by using the O_CIO open flag. For example, to mount the file system /test with concurrent I/O, run the following command:

```
# mount –o cio /test
```

Concurrent I/O provides better performance than direct I/O, but has some additional restrictions. Concurrent I/O is only available for JFS2 file systems and namefs. Mixed open modes for a file are not supported with concurrent I/O. If a process attempts to open a file in concurrent I/O mode, and that file is already open by some other process in the system without concurrent I/O, then the

concurrent I/O open fails. If a file is opened in concurrent I/O mode, and another process tries to open that same file without concurrent I/O, then the nonconcurrent I/O open fails. In AIX 6.1, you can use the O_CIOR flag in conjunction with O_CIO to allow other applications to open the named file for read access without specifying O_CIO. This allows applications that are not coded to use concurrent I/O to run without modification. It should be noted that the usual guarantee regarding atomicity of the read and write operation does not apply in this case. O_CIOR is only available with the open64x() interface. (**Note:** This feature is not yet implemented in Oracle 11g Release 1 or earlier. Therefore, these applications do have to specify O_CIO to open Oracle data files when database is up and running and opened with O_CIO). The use of concurrent I/O is intended only for special applications, such as Oracle, which implement their own data serialization. When using the **cio** mount option on a file system, that file system can only contain Oracle data and log files. Never use the **cio** mount option on a file system that contains shared libraries or executable code, such as ORACLE_HOME.

Oracle Database 10*g* opens data files that are located on the JFS2 file system with the O_CIO option if the filesystemio_options initialization parameter is set to either **directIO** or **setall**. You can use the **setall** option, because asynchronous I/O is disabled when **directIO** is specified. Note that **directIO** is used for both direct I/O and concurrent I/O, and Oracle checks to see whether the data file is on JFS or JFS2 and uses direct I/O or concurrent I/O accordingly.

Enhanced journal file system (JFS2) and concurrent I/O

For non RAC users who prefer the flexibility of using a file system rather than raw devices, the best choice is the enhanced journal file system (JFS2) with concurrent I/O. When properly tuned, JFS2 with concurrent I/O provides performance that is close to that of raw devices. AIX V5.3 TL05 introduced a new tunable fsfastpath which specifies how to manage the I/O requests for files that are opened with Concurrent IO mode in a JFS2. If enabled, it initiates asynchronous I/O requests directly to LVM or disks. It is recommended that you enable this tunable for maximum performance with JFS2 file systems with concurrent I/O.

The I/O alignment restrictions that determine when the file-system buffer cache is bypassed with direct I/O or concurrent I/O are dependent on the JFS2 block size (agblksize). For optimal I/O performance, agblksize needs to be equal to the size of the smallest block size that is used when accessing the file system. This is the largest agblksize that meets the alignment restrictions for bypassing the file-system buffer cache. The smallest block size that is used by Oracle for data files is the database block size. For the redo log files and control files, the smallest block size that is used is 512 bytes. For this reason, always put the online redo log files and control files on a separate file system with an agblksize of 512, and put the data files on one or more file systems with the agblksize equal to the smallest database block size that is used for table spaces in data files on that file system, or 4096, whichever is smallest.

When the database has multiple database block sizes, it might be advantageous to group data files by the database block size that is specified in the table spaces they contain, and set the agblksize to this value. If a separate file system is not used for each database block size, the agblksize of the file system can be that of the smallest database block size that is used.

GPFS

For users who want to use a file system with Oracle RAC, GPFS is available. Beginning with Oracle Database 9i Release 2, Oracle automatically detects the presence of a GPFS file system and automatically opens all GPFS based data files in direct I/O mode.

For information on tuning GPFS for Oracle, refer to the GPFS for AIX FAQs link that is listed in the **Resources** section of this paper.

Automated storage management

Oracle release 10*g* introduced a new feature, the Oracle Automated Storage Management (ASM) component, that you can use to manage Oracle database files. You configure ASM by assigning raw devices to ASM disk groups. When creating a table space, you specify the disk group where the table space is to be located, and you specify a template, which describes the storage attributes of the data. Using this information, ASM allocates the table spaces with the desired properties. Among other attributes, ASM handles mirroring and striping, based on the template. A predefined set of templates provides for different database file types. ASM automatically rebalances and stripes data across all the raw devices that are specified in a disk group.

When using ASM, the database files are on raw devices. These raw devices are not in the AIX volume groups or logical volumes and, therefore, VG and LV AIX tuning parameters do not apply. The other AIX I/O tuning processes are the same as when using raw devices or raw logical volumes without the ASM.

Tools to monitor I/O performance

Several commands allow I/O performance to be monitored in the AIX operating system. Some of these commands also allow the monitoring of processor and other system performance. This section explains how to use these commands.

vmstat

The **vmstat** command contains a useful overview of system performance, including some parameters related to I/O processes. Refer to the **Tools to monitor processor performance** section of this paper.

iostat

The primary use of the **iostat** command is to report system I/O statistics. Typically, the command is used with an interval and count value, to show the I/O statistics over a number of fixed intervals.

The sample **iostat 10 1** output shown in Figure 16 illustrates the I/O statistics for one 10-second interval:

```
$ iostat 10 1
System configuration: lcpu=2 drives=21
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0          4.2               0.9      1.6       49.0      48.5
Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
dac0             0.0       9.2       3.8         82        10
dac0-utm         0.0       0.0       0.0          0         0
hdisk20          0.0       9.2       3.8         82        10
hdisk21          0.0       0.0       0.0          0         0
hdisk22          0.0       0.0       0.0          0         0
hdisk4           0.0       0.0       0.0          0         0
hdisk8          29.2    6553.6      51.2          0     65536
hdisk9          28.6    6553.6      51.2          0     65536
hdisk10          0.0       0.0       0.0          0         0
hdisk7           0.0       0.0       0.0          0         0
hdisk6          40.3    8704.0      68.0          0     87040
hdisk12          0.0       0.0       0.0          0         0
hdisk11          0.0       0.0       0.0          0         0
hdisk13          0.0       0.0       0.0          0         0
hdisk15          0.0       0.0       0.0          0         0
hdisk5           0.6       3.6       0.9          0        36
hdisk14          0.0       0.0       0.0          0         0
hdisk16          0.0       0.0       0.0          0         0
hdisk18          0.0       0.0       0.0          0         0
hdisk17          0.0       0.0       0.0          0         0
hdisk19          0.0       0.0       0.0          0         0
```

Figure 16. I/O statistics for one 10-second interval

The definitions for the columns in the sample output are as follows:

- **% tm_act**: Indicates the percentage of time the physical disk was active (bandwidth utilization for the drive).
- **Kbps**: Indicates the amount of data transferred (read or written) to the drive in kilobytes per second.
- **tps**: Indicates the number of transfers per second that were issued to the physical disk. A transfer is an I/O request to the physical disk. Multiple logical requests can be combined into a single I/O request to the disk. A transfer is of indeterminate size.
- **Kb_read**: Indicates the total number of kilobytes that were read.
- **Kb_wrtn**: Indicates the total number of kilobytes that were written.

Beginning with AIX V5.3, there is a new –A option for **iostat** that shows processor and asynchronous I/O statistics. By default, the –A option shows the asynchronous I/O statistics by disk, or the –Q option can be added to show the asynchronous I/O statistics by file system. When using the –A option, the same sets of statistics are shown, but only for asynchronous I/O operations.

The command **iostat –a** command shows disk-adapter throughput. For maximum aggregate performance, the total of the transfer rate must be below the disk-adapter throughput. For disk and adapter settings, refer to the storage vendor for its recommended values.

The **iostat –D** command shows the extended disk report. The sample **iostat –DT 10 1** output shown in Figure 17 illustrates the extended I/O statistics for one 10-second interval and the –T option specifies the time stamp. The –T option is very useful to correlate with other performance data.

```
System configuration: lcpu=8 drives=155 paths=6 vdisks=0

dac12          xfer:  %tm_act      bps      tps      bread      bwrtn      time
                        0.0     23.1M   7257.2       19.1M       4.0M  14:35:30
               read:      rps  avgserv  minserv  maxserv   timeouts      fails
                       6835.3      4.8      0.1     35.8           0          0
              write:      wps  avgserv  minserv  maxserv   timeouts      fails
                        421.9      0.9      0.1      9.7           0          0
              queue:  avgtime  mintime  maxtime  avgwqsz    avgsqsz     sqfull
                        0.0      0.0      4.6      0.0        0.0         0.0
hdisk7         xfer:  %tm_act      bps      tps      bread      bwrtn      time
                       95.0      1.9M   615.4        1.6M     319.9K  14:35:30
               read:      rps  avgserv  minserv  maxserv   timeouts      fails
                       582.1      5.0      0.1     29.9           0          0
              write:      wps  avgserv  minserv  maxserv   timeouts      fails
                        33.3      0.5      0.2      2.0           0          0
              queue:  avgtime  mintime  maxtime  avgwqsz    avgsqsz     sqfull
                        0.0      0.0      0.0      0.0        0.4         0.0
```

Figure 17.The extended disk report

The definitions for the columns in the sample output are as follows:

- The metrics related to transfers are as just described in regard to Figure 16.
- The metrics for read and write are the following:
  - **rps and wps**: Indicates the number of reads or writes per second
  - **avgserv**: Indicates the average service time per read or write transfer (default is in milliseconds)
  - **minserv**: Minimum read or write service time. Default is in milliseconds.
  - **maxserv**: Maximum read or write service time. Default is in milliseconds.
  - **timeouts**: Number of reads or writes timeouts per second
  - **fails**: Number of failed reads or writes requests per second
- The metrics related to queue service time are as follows:
  - **avgtime**: The average time spent by a transfer request in the wait queue.
  - **mintime**: Minimum time spent by a transfer request in the wait queue
  - **maxtime**: Maximum time spent by a transfer request in the wait queue. Default is in milliseconds.
  - **avgwqsz**: Indicates the average wait-queue size.
  - **avgsqsz**: Indicates the average service-queue size.
  - **sqfull**: Indicates the number of times the service queue becomes full (that is, the disk does not accept any more service requests) per second.

The filemon command

The *filemon* command reports the I/O activity for each of the layers in the I/O subsystem: logical files, virtual memory segments, logical volumes and physical volumes. For each layer, filemon reports a summary of the I/O characteristics.

Figure 18 shows a section of output from the filemon tool, with the I/O statistics at the logical-volume level for one of the logical volumes that was used by a file system that contains an Oracle data file:

```
------------------------------------------------------------------------
Detailed Logical Volume Stats   (512 byte blocks)
------------------------------------------------------------------------
VOLUME: /dev/fslv19  description: /j2r5A
reads:                  4       (0 errs)
  read sizes (blks):    avg    16.0 min      16 max      16 sdev     0.0
  read times (msec):    avg 371.190 min  66.367 max 479.576 sdev 176.161
  read sequences:       4
  read seq. lengths:    avg    16.0 min      16 max      16 sdev     0.0
writes:                 12923   (0 errs)
  write sizes (blks):   avg    16.0 min      16 max      16 sdev     0.0
  write times (msec):   avg 460.946 min   8.925 max 1390.970 sdev 158.716
  write sequences:      12923
  write seq. lengths:   avg    16.0 min      16 max      16 sdev     0.0
seeks:                  12927   (100.0%)
  seek dist (blks):     init 13318528,
                        avg 937146.4 min      16 max 14107768 sdev 2401696.8
time to next req(msec): avg   2.498 min   0.001 max 3126.228 sdev  34.707
throughput:             3183.6 KB/sec
utilization:            0.82
```

Figure 18. Section of output from filemon

In this sample output, during the filemon sample interval, there were three reads and 12 923 writes. All of the writes were the same size, 16 blocks, or 8 KB, which was the database block size.

Summary

Oracle performance tuning is a vital part of the management and administration of successful Oracle database systems, and requires tuning both the Oracle database and the operating system to work efficiently together. This paper presents how to tune the AIX 5.3 and AIX 6.1 operating system to achieve the best performance with the Oracle database, and to leverage some of the new hardware-based features that are provided by the latest IBM POWER processors (including simultaneous multithreading [SMT] and the Micro-Partitioning feature).

This paper does not cover performance tuning of the Oracle database as it relates to SQL code tuning, data modeling or any other aspect of the configuration of the Oracle database itself. Oracle Database generic-tuning concepts are widely available and are usually independent of the system and hardware.

To best exploit the AIX 5.3 and AIX 6.1 operating system, you need to tune it for the specific characteristics of the Oracle application. This is required to achieve the maximum performance and fully use the resources that are available on the server. Tuning tasks are grouped into three sections: processor, memory and I/O.

You can tune the processor for greater performance through the use of SMT, and scheduling and prioritizing Oracle database tasks. You can also specifically assign processors to certain tasks, and you can use LPARS can to improve Oracle throughput. Administrators and developers can also use the vmstat, iostat and sar command tools to analyze performance.

Further, you can tune the AIX memory subsystems, including the option of reducing the memory footprint of the Oracle processes, using pinned memory, and using large pages to achieve better performance.

You can do specific performance tuning of the I/O subsystem for different types of Oracle database deployments, depending on the use of raw devices (logical volumes or disks), file systems, Automated Storage Management (ASM) and General Parallel File System (GPFS).

In each section of this paper, the most useful tools to configure, monitor and evaluate the performance of the tuning aspects were covered.

Resources

These Web sites provide useful references to supplement the information contained in this document:

- IBM System p (POWER6, POWER5+ and POWER5)
  ibm.com/systems/p
- IBM POWER6 Microprocessor Technology
  www.research.ibm.com/journal/rd51-6.html
- IBM POWER5
  www.research.ibm.com/journal/rd49-45.html
- IBM AIX Information Center (AIX 6.1 and AIX 5.3 reference manual)
  http://publib.boulder.ibm.com/infocenter/systems/scope/aix/index.jsp
- IBM AIX (AIX 6.1 and AIX 5.3 offerings)
  ibm.com/servers/aix
- GPFS for AIX FAQs and the HACMP library
  http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfsbooks.html
- AIX 5L Performance Management Guide, Using POWER4-based Systems
  http://publib16.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/prftungd/prftungd03.htm
- Guide to Multiple Page Size Support on AIX 5L version 5.3
  ibm.com/systems/resources/systems_p_os_aix_whitepapers_multiple_page.pdf
- Oracle Database (11g and 10g)
  www.oracle.com/database/index.html
- Oracle Real Application Clusters (RAC) with Oracle Database
  www.oracle.com/database/rac_home.html
- Oracle MetaLink
  metalink.oracle.com
- Oracle Database 11g Release 1
  www.oracle.com/pls/db111/homepage
- Oracle Database Performance Tuning Guide, 10g Release 1 (10.1) (B10752-01)
  www.oracle.com/pls/db10g/portal.portal_demo3?selected=1
- Oracle Technology Network for current Oracle Database documentation
  www.oracle.com/technology/documentation/index.html

White papers

- Improving Database Performance With AIX Concurrent I/O
  ibm.com/servers/aix/whitepapers/db_perf_aix.pdf
- Oracle Architecture and Tuning on AIX
  ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100883
- Oracle DB & RAC 10gR2 on IBM AIX: Tips and Considerations
  ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101089
- Oracle DB & RAC 10gR2 on IBM AIX: Tips and Considerations
  ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101176
- Oracle9i & 10gR1 on IBM AIX5L: Tips & Considerations
  ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100556
- Configuring IBM System Storage DS4000 Series for Oracle Database Applications
  ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100780

IBM Redbooks (**ibm.com**/redbooks)

Advanced PowerVM Virtualization on IBM System p:

- Introduction and Basic Configuration (SG24-7940)
- Architecture and Performance Considerations (SG24-5768)
- PowerVM Virtualization Managing and Monitoring, SG24-7590)
- A Practical Guide for Resource Monitoring and Control (SG24-6615)
- Partitioning Implementations for IBM eServer p5 Servers, SG24-7039-02
- Partitioning Implementations for IBM eServer pSeries Servers (SG24-7039)
- Effective System Management Using the IBM HMC for pSeries (SG24-7038)
- Problem Solving and Troubleshooting in AIX 5L, SG24-5496-01
- Understanding IBM eServer pSeries Performance and Sizing, SG24-4810-01
- AIX Version 6.1 Differences Guide, SG24-7559-00)
- AIX Version 5.3 Différences Guide (SG24-5766)
- AIX 5L Performance Tools Handbook (SG24-6039)
- AIX 5L Practical Performance Tools and Tuning Guide, SG24-6478-00
- IBM AIX Version 6.1 Différences Guide (SG24-7559)

The technical sales library

How to find which AIX versions are certified with Oracle Database releases
ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS3369

About the authors

Dennis Massanari

IBM eServer Solution Enablement

Dennis Massanari is a senior software programmer in the IBM eServer Solutions Enablement organization, where he assists solution providers in enabling applications for the AIX operating system on the IBM POWER platforms. He has 26 years of experience in the area of performance, initially developing performance-measurement tools for IBM eServer zSeries® processors. For the past 10 years, he has worked on porting, performance and benchmarks with Oracle Database. He holds a Bachelor of Science in Computer Engineering from the University of Illinois. Dennis can be contacted at massanar@us.ibm.com.

Arun Sar

IBM eServer Solution Enablement

Arun Sar is a senior software engineer in the IBM eServer Solutions Enablement organization, where he assists solution providers in enabling their applications for AIX and Linux on the System p platform. His area of interest includes compiler, linker, file system and virtual-memory management. He holds a Masters degree in Computer Science and Application (MCA) from Utkal University, India. Arun can be contacted at asar@us.ibm.com.

Lilian Romero

IBM Power Systems Performance

Lilian Romero is a senior software engineer in the IBM Power Systems performance group. She has been involved in the publication of numerous OLTP industry benchmarks using Oracle Database. She evaluates the performance of new AIX releases and Power System hardware. She holds a Bachelor of Science degree in Computer Science. Lilian can be contacted at lilianr@us.ibm.com.

Trademarks and special notices

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.