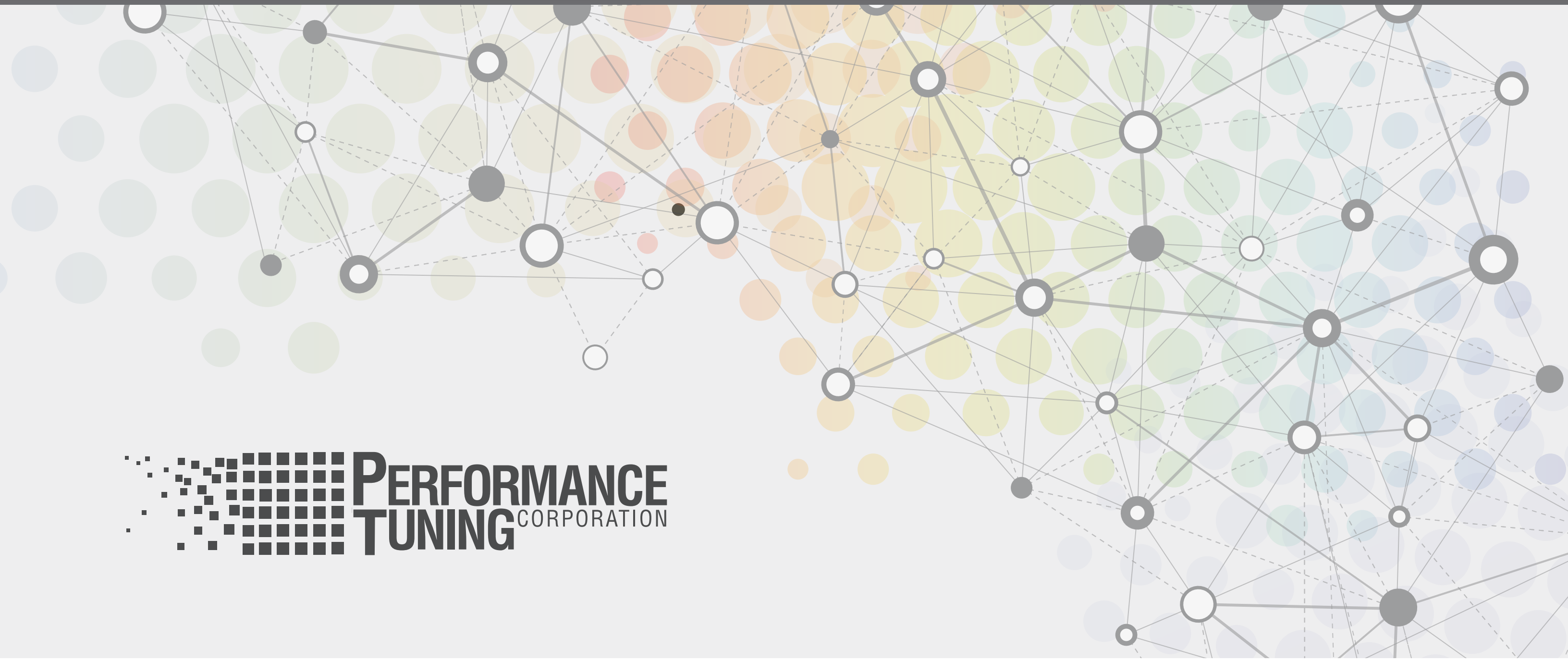


PERFORMANCE TUNING ORACLE RAC ON LINUX

By: Edward Whalen

Performance Tuning Corporation

INTRODUCTION

Performance tuning is an integral part of the maintenance and administration of the Oracle database server. Performance tuning includes modifications to the application and SQL statements, index creation and maintenance, and Oracle instance tuning. The tuning process and the areas optimized are the same regardless of the operating system, and therefore, will not be covered in detail in this paper. Instead, this paper will focus on the aspects of performance tuning that apply to the Linux operating system. In addition, storage sizing will also be covered here.

Performance tuning is as much an art as it is a skill. It involves patience, the ability to think outside the box, and a positive attitude. With each version of Oracle, the monitoring and troubleshooting tools improve. In addition, Oracle now provides more tools to monitor the OS as well. These include Grid Control, Lightweight Onboard Monitor (LTOM) and the Oracle OS Watcher (OSW). This paper will also explore these tools and how to best apply them.

HARDWARE TUNING

Several important tasks where performance tuning considerations must be taken into account include; hardware sizing, capacity planning and hardware design and configuration. All of these tasks involve some type or level of hardware tuning. When initially designing and configuring the system that will be hosting the Oracle database server, it is important to keep performance in mind. Too often price is the only consideration and performance is an afterthought. Some tuning problems can be resolved after the fact, but capacity issues are more difficult to address and might involve the addition or change of hardware or other resources.

CHOSE THE RIGHT HARDWARE

Choosing the right hardware is one of the most important steps one can take to optimize system performance. In most cases, an underpowered system will require the addition of new hardware later. By selecting the right equipment for the job you will avoid additional work later.

In order to choose the right hardware, one must determine the following:

- How much physical memory do you need? The amount of memory will determine how well you are able to cache objects and run large queries. Keep in mind the memory that might be needed for multiple Oracle caches as well as non-standard block size caches.
- What type and how many CPUs/Cores are required to support the expected system load?[1]
- What type of application(s) will be run?
- How much CPU power is needed?
- Will the servers be configured as a Real Application Cluster (RAC)? If so, how many nodes should there be? What type and speed are the networking components that will be used as the cluster interconnect?
- Is storage sized properly?

Answers to these questions and others will help you decide how to configure the system.

USE 64-BIT HARDWARE

As part of designing the system, 64-bit hardware and software should be used. Now that all new commodity hardware is 64-bit capable there is no need to run in 32-bit mode.

The disadvantages of 32-bit include:

[1] Oracle is currently licensing a core as ½ of a CPU, thus 2 cores require a 1 CPU license.

- Limited to 4 GB of virtual memory per process. This limitation is not usually problematic in Linux, but should be avoided nevertheless.
- Low memory issues. With 32-bit Linux there is special low memory below 840 MB reserved for kernel operations, page table entries and DMA. This memory can be taxed at times, thus causing process failures.
- Physical memory limitations. With the best 32-bit hardware, physical memory is limited to 64 GB. This is built into the architecture and cannot be extended.

These issues can easily be avoided by running 64-bit Linux and 64-bit Oracle software.

STORAGE SIZING AND TUNING

Perhaps the most important hardware component to be sized is the I/O subsystem. The I/O subsystem is made up of a number of components including the I/O bus in the server, the I/O controller(s), the I/O bus in the storage system and ultimately, the disk drive. The fundamental component, the disk drive, hasn't changed much in the last 20 years. It is slightly faster (unless you use SATA), but is essentially the same. No matter how much cache and hardware you put in between the disk drives and the system, ultimately the performance of the I/O subsystem will be constrained by throughout of the disk drive.

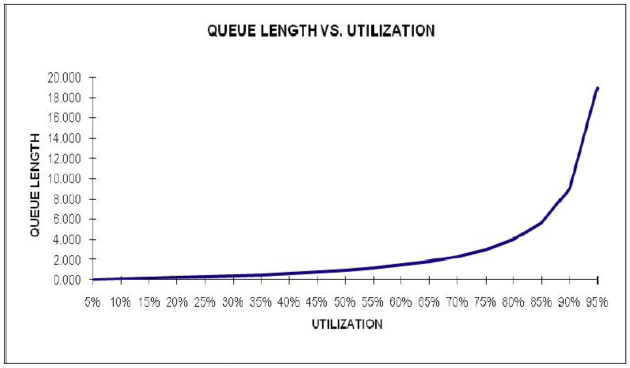
DISK DRIVE 101

The disk drive is still the fundamental component of the I/O subsystem and as such, often the limiting factor of its performance. A disk drive is a physical device that spins at a fixed rotational speed (usually 15,000 RPM or less). The disk head moves across the disk platter to the track where the desired data resides (seek time) and then waits for the disk drive to rotate under it in order to read the desired data (rotational latency). These two time spans - seek time and rotational latency - make up the bulk of the time it takes to read random data from a disk drive. For a typical SCSI or Fibre Channel drive, this time is on the order of 5 ms (seek time) plus 2 ms (rotational latency) = 7 ms service time.

If a disk drive takes 7 ms to do a typical random I/O, then you can do approximately 142 random I/Os per second as the theoretical maximum. However, because of queuing and collisions, as you near that theoretical maximum, the 7 ms I/O latency begins to increase exponentially. As you surpass 142 I/Os per second, queuing occurs and the I/O latency or response time is equal to:

Response time = service time + queue time

As the number of items in the queue increases, the length of the queue increases exponentially as shown in the following graph:



Utilization vs. Queue Length.

As the queue length increases, so does the response time. In a poorly sized I/O subsystem it is not uncommon to see latencies in the 100's of milliseconds. This can be avoided somewhat by properly sizing the I/O subsystem. The problem may also be reduced by properly tuning queries in order to reduce unnecessary I/Os.
There are a number of I/O subsystems available on the market today. Typically we find larger databases either using SAN or NAS subsystems. In the past, IT Managers avoided NAS to store data, but improvements in the technology make these subsystems faster and more reliable and an ideal choice for Oracle databases. For example, with Oracle 11g, an NFS client is built into the Oracle database server itself, thus reducing NAS overhead further.

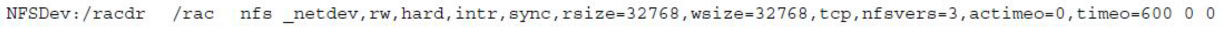
SAN

The Storage Area Network or SAN has been the most common I/O subsystem for large Oracle databases. The SAN is an actual network of computer systems, fibre channel switches and storage subsystems. These systems form a network where multiple systems can access one or more storage subsystems, which can easily be shared, thus providing shared storage for RAC clusters. In addition, SAN storage systems typically support a large amount of cache and performance features. However, no matter what features and cache is available, the fundamental building block of the storage subsystem is still the disk drive.

NAS

In the past, Network Attached Storage or NAS systems typically did not provide enough performance for the Oracle database server, however, with gigabit and faster networks, TCP Offload Engine (TOE) adapters and faster network gear, NAS is now a viable option for the Oracle database server. One reason that NAS has become more popular is that it tends to be less expensive than SAN because it uses the existing network infrastructure.

NAS performance can be enhanced via the use of TOE network adapters and by optimally configuring the NAS device in /etc/fstab. Here is an example of a NAS device:



```
NFSDev:/racdr   /rac   nfs _netdev,rw,hard,intr,sync,rsize=32768,wsize=32768,tcp,nfsvers=3,actimeo=0,timeo=600 0 0
```

Note: For file systems that only contain Oracle data files (not Oracle program files) you can use forcedirectio as well. For RAC systems, include the parameter noac.[2]

By properly configuring the NAS device, performance can be enhanced. With NFS it is particularly important to use the proper mount parameters. In addition, if possible, use a dedicated network for storage processing.

RAID

Both SAN and NAS systems support a number of RAID levels. Like most configurable components, there are tradeoffs. Some RAID levels are slower and more economical, while other RAID levels are faster and more expensive. For most SAN and NAS systems it is recommended that RAID 10 be used rather than RAID 5. RAID 5 is more economical, but the write performance is slower than RAID 10. In addition, if the RAID volume is in a degraded state (i.e. a drive has failed) RAID 5 has a much higher rebuild penalty and performance is greatly degraded.

TUNING THE NETWORK

Tuning the network involves two components - tuning the network from the hardware standpoint, covered in this section, and tuning from the OS standpoint, covered in the next section. Tuning the network hardware includes purchasing and enabling the features of both the network hardware in the system as well as the network infrastructure. Hardware tuning can include TOE adapters, Jumbo Frames and VLANs.

TOE ADAPTERS

TOE or TCP Offload Engine Network Interface Cards (NICs) allow much of the processing associated with the TCP portion of TCP/IP to be performed by the network card hardware, rather than having Linux do it. By processing TCP in hardware, it is faster and frees up the CPUs in your system to do more important work.

[2] See Oracle Metalink note 551159.1.

JUMBO FRAMES

Jumbo Frames allows Ethernet packets to be sent in a larger size. Ethernet frames typically use a 1500 byte size. Jumbo Frames allow this to be extended to a 9000 byte frame. The 9000 byte frame allows 8K of data to actually be sent in one frame, rather than having it split into multiple frames, thus improving performance where large amounts of data are being transferred. The Jumbo Frame is configured for both the network card and the switch, thus creating a hardware tuning issue. Jumbo frames are less efficient when small amounts of data are transmitted, and therefore, not recommended for all cases. We recommend that jumbo frames be used in the following cases:

- An Oracle RAC interconnect, where the RAC interconnect experiences extensive traffic and most packets are large.
- Networks dedicated for storage, where NAS is used via a private network.
- Networks dedicated for backup, where a dedicated network is used for backups.

Jumbo frames should not be used on your general public network, because it can degrade performance for small Oracle net packets.

VLANS

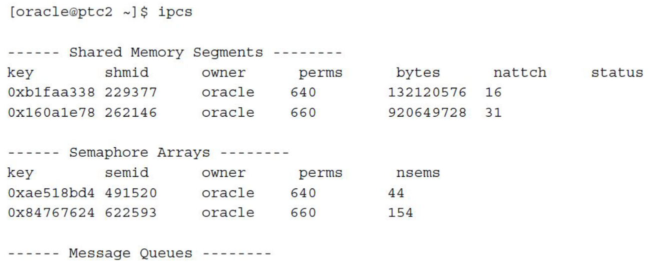
The Virtual Local Area Network (VLAN) is a private network carved out of your network infrastructure. In a VLAN, only the traffic destined for systems within the VLAN is permitted, thus general network broadcasts are blocked out. A VLAN can be effective for a RAC interconnect a private backup network, or a private storage network. Each individual system's purpose and requirements must be taken into account to determine whether a VLAN will be effective.

OS TUNING

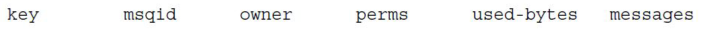
There are a number of areas that can be tuned in the Operating System (OS), however, not all of them will directly provide better performance. For example, allocating more shared memory in the OS does not in itself provide greater performance. However, by allocating more OS shared memory; a larger SGA can be created by Oracle, which will potentially provide greater performance.

SHARED MEMORY AND SEMAPHORES

Oracle uses shared memory for the Oracle System Global Area (SGA). The SGA is used to store database block buffers, the shared pool, the large pool, the log buffer and other Oracle data structures. Semaphores are used by Oracle for synchronization between Oracle server processes, the Oracle background processes, and SGA. Both shared memory and semaphores can be tuned within Linux – the values equate to the maximum amount allowed to be allocated by Oracle. Tuning them higher than Oracle actually needs will add no additional performance, these are simply limits. Increasing the shared memory size allows Oracle to create larger SGAs. Increasing the number of semaphores allows more Oracle sessions to be created. In order to view the shared memory and semaphore usage in Linux use the command ipcs. The output of ipcs is as follows:



```
[oracle@ptc2 ~]$ ipcs

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0xb1faa338 229377     oracle     640        132120576  16
0x160a1e78 262146     oracle     660        920649728  31

------ Semaphore Arrays --------
key        semid      owner      perms      nsems
0xae518bd4 491520     oracle     640        44
0x84767624 622593     oracle     660        154

------ Message Queues --------
```

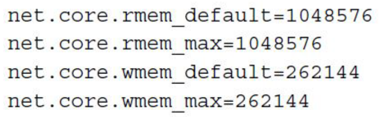


```
key        msqid      owner      perms      used-bytes   messages
```

This output shows Oracle allocating two shared memory segments. The 129 MB segment is used by the ASM instance, the 920 MB shared memory segment is used by the Oracle database instance. The information under nattch shows how many processes are connected to the shared memory segment.

NETWORK TUNING

Linux networking parameters can be tuned, thus improving the performance of the Oracle RAC interconnect. This will provide significant improvement in RAC operations. There are several parameters that should be tuned in the Linux system for Oracle 10g. Below are the recommended OS maximum send and receive buffer sizes for Oracle 10g. These settings should be set in the /etc/sysctl.conf file by adding the following lines to increase the default and max receive read and write buffer sizes:

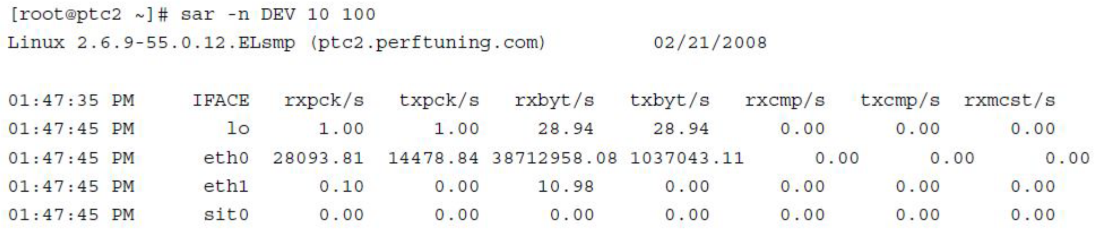


```
net.core.rmem_default=1048576
net.core.rmem_max=1048576
net.core.wmem_default=262144
net.core.wmem_max=262144
```

These parameters tell Linux how much buffer space to reserve for each socket.

Note: For Oracle 9i, the recommendation is for both rmem and wmem to be set to 256K.
Note: For Oracle 11g, the recommendation is to set both rmem and wmem to 4194304.

The amount of traffic being sent across the network can be monitored by using the Linux command sar. Using the command sar –n DEV the number of packets and throughput can be seen as shown here:



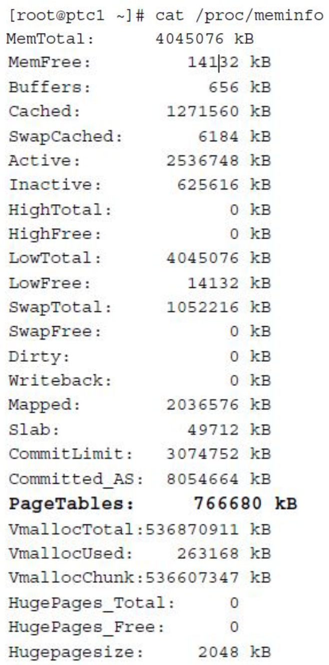
```
[root@ptc2 ~]# sar -n DEV 10 100
Linux 2.6.9-55.0.12.ELsmp (ptc2.perftuning.com)         02/21/2008

01:47:35 PM     IFACE   rxpck/s   txpck/s   rxbyt/s   txbyt/s   rxcmp/s   txcmp/s  rxmcst/s
01:47:45 PM        lo      1.00      1.00     28.94     28.94      0.00      0.00      0.00
01:47:45 PM      eth0  28093.81  14478.84 38712958.08 1037043.11      0.00      0.00      0.00
01:47:45 PM      eth1      0.10      0.00     10.98      0.00      0.00      0.00      0.00
01:47:45 PM      sit0      0.00      0.00      0.00      0.00      0.00      0.00      0.00
```

By monitoring the networks you can determine if the interconnect is being overutilized. Some interconnect statistics are available in the Oracle AWR reports as well.

HUGEPAGES

When using large amounts of memory for Oracle, the OS consumes significant resources to manage the pagetable. The pagetable is a table that keeps track of the virtual memory used by all of the processes in the system. The pagetable has a cache called the TLB (Translation Lookaside Buffers). The TLB speeds up lookups by caching the pagetable entries, just like any other cache. Unfortunately, the TLB is fairly small and will hold on the order of 512 to 4096 entries.

When using 64-bit Linux and lots of memory (above 8GB), the number of pagetable entries (and the associated memory) can be very large. The following example of meminfo output for a large SGA shows the TLB pagetable entries taking up 766 MB of RAM.



```
[root@ptc1 ~]# cat /proc/meminfo
MemTotal:      4045076 kB
MemFree:         14132 kB
Buffers:           656 kB
Cached:        1271560 kB
SwapCached:       6184 kB
Active:        2536748 kB
Inactive:       625616 kB
HighTotal:           0 kB
HighFree:            0 kB
LowTotal:      4045076 kB
LowFree:         14132 kB
SwapTotal:     1052216 kB
SwapFree:            0 kB
Dirty:               0 kB
Writeback:           0 kB
Mapped:        2036576 kB
Slab:            49712 kB
CommitLimit:   3074752 kB
Committed_AS:  8054664 kB
PageTables:     766680 kB
VmallocTotal:536870911 kB
VmallocUsed:    263168 kB
VmallocChunk:536607347 kB
HugePages_Total:     0
HugePages_Free:      0
Hugepagesize:     2048 kB
```

This can be a significant overhead on the system. By enabling hugepages (by setting a Linux parameter) or large page allocations for database buffers, the number of TLB entries can be reduced by increasing the page size. In Linux, when hugepages are used, the page size is changed from the default of 4 KB to 2 MB. In the Linux 2.6 kernel, hugepages are configured by setting the Linux initialization parameter vm.nr_hugepages to the number of 2M pages that you want. This reduces the total number of pages by making each page larger.[3]

Note: Hugepages are only used for the Oracle SGA. The OS will run with a combination of 4 KB and 2 MB pages when hugepages is enabled.

TUNING IO

There are only a few things that you can actually do from the OS perspective to tune the I/O subsystem. One is to change parameters in the device driver, which is not recommended. The other is to change the I/O scheduler. The Linux operating system has replaced the elevator sorting algorithms (tuned with elvtune) in the 2.4 kernel with multiple I/O schedulers.

I/O SCHEDULERS

The Linux 2.6 kernel provides four I/O schedulers to choose from, each with its own characteristics and designed to perform optimally for different types of environments, as described in the following sections. The I/O scheduler is specified in the /boot/grub/grub.conf configuration file, at the end of the kernel selection line, by adding the following:

```
set elevator=<scheduler>
```

[3] See Metalink note 401749.1.

The four options for the <scheduler> are:

- cfq - Completely Fair Queuing.
- deadline – the Deadline scheduler.
- noop – The NOOP or No Operation scheduler.
- as - Anticipatory

THE CFQ SCHEDULER

The cfq scheduler is the default scheduler for Linux and is usually the best general purpose scheduler. This scheduler maintains a per-process I/O queue. It attempts to distribute available bandwidth equally among all I/O requests. It is well suited for multi-processor systems and for storage subsystems with multiple LUNs and controllers. This is the default scheduler because it seems to perform well among a number of diverse workloads.

THE DEADLINE SCHEDULER

The deadline scheduler uses a deadline algorithm to minimize latency. It is said to provide near real-time behavior by using a round robin policy. In addition it will aggressively re-order requests. There is some speculation that this might be a better scheduler under a heavy Oracle workload.

THE NOOP SCHEDULER

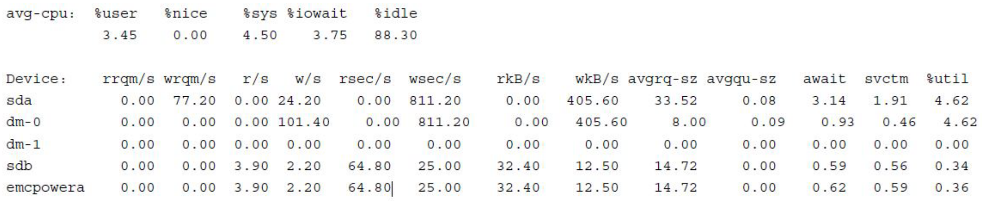
The noop or no op scheduler is a simple FIFO queue. It uses minimal CPU and assumes I/O will be optimized by the HBA or external disk controller. In theory, this should be an ideal scheduler for high powered SAN systems, since the idea of the noop scheduler is to simply get the request down to the I/O subsystem as quickly as possible. No processing or re-ordering is done on the I/O request.

THE AS SCHEDULER

The as or anticipatory elevator scheduler is similar to what was done in the 2.4 kernel. It uses controlled delay to allow time for requests to be re-ordered by elevator algorithms. This scheduler is designed for small or slow disk subsystems. It works by delaying the I/O and then re-ordering the I/Os in the queue. Since the underlying storage is probably a RAID array, the requests will have to be re-ordered on a per-disk basis later anyway. This scheduler should not be used with database systems.

MONITORING I/O

Since the I/O subsystem performance is extremely important to the overall performance of Oracle, it should be closely monitored. Monitoring the I/O subsystem can be done via the iostat command. By running iostat –x it is possible to get more detailed information on the performance of the I/O subsystem, such as service time (svctm) and average wait time (await). The output from this command appears as follows:



```
avg-cpu:  %user   %nice    %sys %iowait   %idle
           3.45    0.00    4.50    3.75   88.30

Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await  svctm  %util
sda          0.00  77.20  0.00 24.20    0.00  811.20     0.00   405.60    33.52     0.08    3.14   1.91   4.62
dm-0         0.00   0.00  0.00 101.40    0.00  811.20     0.00   405.60     8.00     0.09    0.93   0.46   4.62
dm-1         0.00   0.00  0.00  0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
sdb          0.00   0.00  3.90  2.20   64.80   25.00    32.40    12.50    14.72     0.00    0.59   0.56   0.34
emcpowera    0.00   0.00  3.90  2.20   64.80   25.00    32.40    12.50    14.72     0.00    0.62   0.59   0.36
```

ORACLE TUNING

Tuning Oracle on Linux is no different than tuning Oracle on other platforms. The basic goals of tuning are still in place; reduce the amount of resources being used to process SQL statements, optimize the use of CPU and memory, and attempt to reduce the time spent blocking on other processes.

REDUCE RESOURCE USAGE

One of the main goals in performance tuning an Oracle system is to reduce the amount of resources needed by each SQL statement. This means optimizing the data access path (via indexes), the amount of data being accessed (via partitioning) and the amount of memory being used (via keep, recycle and default caches). Typically this includes tasks such as tracing SQL statements, viewing execution plans and developing better ones. Partitioning can help reduce resource usage by allowing table scans to operate on only the needed partitions. Indexes allow data to be retrieved with much fewer I/O operations. By reducing resource usage, the entire system will function more efficiently.

LOWER RAC INTERCONNECT TRAFFIC

In a RAC environment reducing interconnect traffic is an essential part of tuning the system. The RAC interconnect can be the limiting component to scalability. Since the typical interconnect hardware is gigabit Ethernet, there is not much that can be done to the physical hardware with the exception of the network tuning recommended earlier in this paper. However, interconnect traffic can be reduced via programming techniques and architecture. Services can be used to isolate specific functions to be performed on specific nodes in an effort to eliminate the occurrence of updates to the same tables and same rows on different nodes, thus reducing the amount of data blocks being bounced around the network, which should be avoided as much as possible.

NORMAL APPLICATION TUNING

Of course, all normal Oracle application tuning applies to Oracle on Linux just as with any other operating system. In fact, the first place to tune is usually the application. With RAC there are special concerns, such as using services to specify different workloads to different nodes. However, in general, tune Oracle on Linux as you would on any other operating system.

ONGOING MONITORING

Ongoing monitoring is an important part of performance tuning. There are a number of tools available for ongoing monitoring including Oracle AWR reports, Grid Control, LTOM, OSW as well as numerous third party tools. Which monitoring tool you use is important, however, what's more important, is that something be used. If you do not regularly monitor the system, you may only find out about performance problems from the user community. By that time it is often too late. Proactive tuning is much better in the long run than tuning only when a problem is reported.

AWR AND STATSPACK

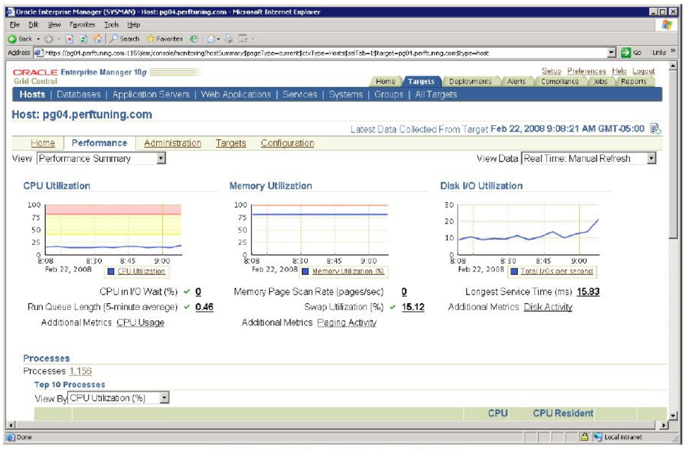
AWR and statspack reports are used to monitor the state of the Oracle instance. As such, there is little data in them that is OS specific, however, there are a few items that can help determine if there is an OS issue to be investigated.

- RAC statistics. If the RAC statistics are showing a high Remote Cache %, the application should be investigated. The application should be tuned so that the majority of the requests for data are local.
- RAC statistics. If the Avg global cache cr block receive time (ms) or the Avg global cache current block receive time (ms): is high, there is probably some sort of interconnect performance problem. Check the network.
- IO stats. Look for high Avg Rd(ms) and Avg Buf Wt(ms). This is an indication of high I/O latencies. This is a clue to investigate the I/O subsystem

Statspack and AWR reports offer a lot of good Oracle and OS information, but any OS performance data should always be followed-up and investigated using OS tools. Since statspack and AWR reports are generic to all platforms, they will not be covered here in detail.

GRID CONTROL

Grid Control provides an excellent method of both alerting and monitoring. It provides excellent OS monitoring capabilities, as shown in this figure, including agents for specific operating systems such as Linux and Linux x86_64. These agents provide both the ability to monitor the OS as shown below, and the ability to monitor the Oracle instances on the system.



Oracle Enterprise Manager (SYSMAN) - Host: pg04.perftuning.com - Microsoft Internet Explorer

Address: https://pg04.perftuning.com:1159/em/console/monitoring/hostSummary$pageType=current$ctxType=Hosts$selTab=1$target=pg04.perftuning.com$type=host

Host: pg04.perftuning.com

Latest Data Collected From Target Feb 22, 2008 9:08:21 AM GMT-05:00

View: Performance Summary

View Data: Real Time: Manual Refresh

CPU in I/O Wait (%) ✓ 0
Run Queue Length (5-minute average) ✓ 0.46
Additional Metrics CPU Usage

Memory Page Scan Rate (pages/sec) 0
Swap Utilization (%) ✓ 15.12
Additional Metrics Paging Activity

Longest Service Time (ms) 15.83
Additional Metrics Disk Activity

Processes 1,156

View By: CPU Utilization (%)

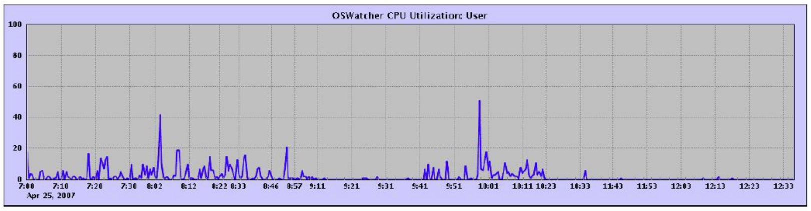
Done

Grid Control Linux Performance Monitoring.

Take specific care to monitor the disk latencies. A typical average I/O latency should not exceed 20 ms. If you are seeing much higher latencies, they should be investigated.

OSW AND LTOM

Oracle has recently begun offering a set of lightweight tools that can be used to monitor the system. The OS Watcher (OSW) tool is a collection of UNIX shell scripts intended to collect and archive OS and network metrics to aid in diagnosing performance issues. OSW operates as a set of background processes on the server and gathers OS data on a regular basis, invoking such Linux utilities as vmstat, netstat, and iostat. The tool is installed on each node where data is to be collected. OSW can be set up to automatically monitor the system during specified times and the data it collects can be turned into graphs that are created in the GIF format as shown here:



Sample OS Watcher Output.

The OS Watcher tool can be downloaded from Oracle Metalink[4].

The Oracle Light Onboard Monitor (LTOM) is another tool that is used for monitoring both the Oracle instance and the OS. It is a java program that is designed as a real-time diagnostic platform. LTOM provides some of the same data that is available with OSW, but provides Oracle session and wait event information as well. LTOM is tightly integrated with the host OS and provides an integrated solution for detecting and collecting trace files for system performance issues. LTOM is designed to be a proactive, rather than a reactive monitoring tool. LTOM delivers automatic problem detection and collects the necessary diagnostic information while the performance problem is occurring. It provides services for:

- Automatic Hang Detection
- System Profiler
- Automatic Session Tracing

Detailed information on the LTOM utility can also be downloaded from Oracle Metalink, along with the appropriate User Guide information.[5]

SUMMARY

Performance tuning is an integral part of the maintenance and administration of the Oracle database server. It involves patience, perseverance, and the ability to think outside of the box. It is as much an art as it is a skill. It takes constant monitoring and tuning in order to keep the Oracle database running optimally. With each version of Oracle, the monitoring and troubleshooting tools constantly improving. In addition, Oracle is now providing more tools to monitor the OS as well. This includes Grid Control, Lightweight Onboard Monitor (LTOM) and the Oracle OS Watcher (OSW). It is great to have these tools to help, but it still takes manual labor in order to keep things running smoothly and optimally.
This paper covered a number of aspects of performance tuning Oracle on Linux and some tips on what to look for and parameters to set. Do not consider this paper to be the final word in performance tuning Oracle on Linux. There will always be changes and new challenges to overcome.

[4] See Metalink note 301137.1.
[5] See Metalink note 352363.1