



Dedication | Experience | Results

White Paper:
Avoiding the Perfect Storm

Author: Randy Cunningham, February 2009



Overview

Memory starvation on a virtual memory system can be quite insidious; the virtual memory demand paging mechanism automatically handles the immediate issue – a shortage of real memory – but the consequences for overall system performance can be incredibly dire if the amount of paging is excessive.

Response times and job processing times can increase by orders of magnitude, overwhelming the help desk, violating service level agreements and reducing the effectiveness of all operations on the system.

When the virtual memory system is significantly overburdened – often called “thrashing” or a “swap storm,” – it might be difficult even for system administrators and database administrators to summon diagnostics and other tools to manage the system.

While the proximate cause might be simple – the system just doesn’t have enough real memory to service the workload – the root causes can be diverse and difficult to identify.

This paper examines six actual case histories of performance problems and the remediation for the problems. Additionally, practices for “staying ahead” of memory problems and preventing them are examined.

The Case Studies

These cases were all observed first-hand by the author in the course of performing triage at distressed sites:

- An Apache server would occasionally require a very long time to render pages, even for simple requests, and would return HTTP 503 and 408 errors from time to time.
- A data warehouse ETL cycle was getting ORA-00600 [2103] errors from various processes that were invoked by the ETL cycle. The aborts were not predictable as to which process would fail. These failures required restarting the specific ETL process that failed, delaying availability of the warehouse.
- In another data warehouse, attempts to connect to the database resulted in sporadic TNS-12170 errors.
- A database production server for a third-party application was observed to run slower and slower, the longer that the environment was up. Bouncing the entire server always restored normal service levels.
- An Oracle data warehouse ETL process evidenced very slow I/O wait times, although I/O wait times were measured to be normal during day-time query processing, or when the ETL step was run in isolation.
- An Oracle e-Business implementation on new Itanium hardware with 128GB of main memory was experiencing a debilitating swap storm, even though the SGA was sized at only 64GB.

For all of the above cases, the proximate cause was “excessive” virtual memory demand paging; however, the root causes were all different and the solution to each issue was different. In several of the cases, the problem was not deterministically reproducible.

A Brief Tutorial on Virtual Memory

On contemporary computing systems, all applications and even most system process access memory using a virtual memory address space, which is mapped to real memory (RAM) using a combination of hardware and operating system functions. This is illustrated in Figure 1.

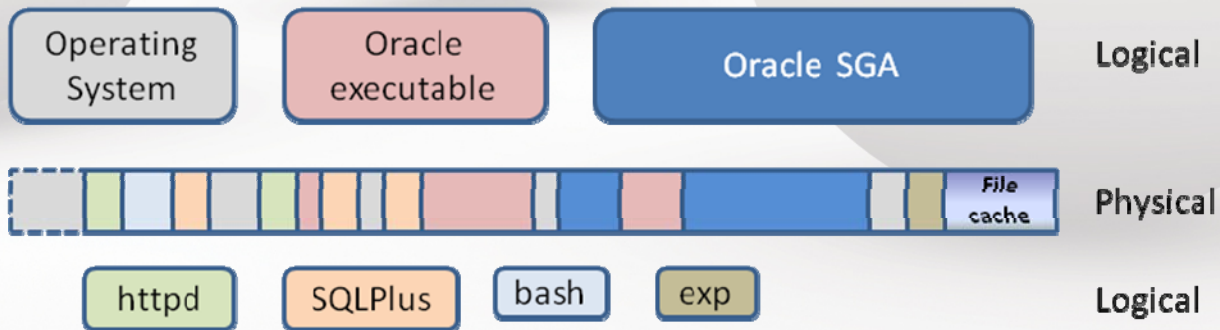


Figure 1. Logical address spaces and physical assignment

There are a number of advantages to this scheme:

- Problems of memory contiguity and honeycomb fragmentation that might exist on a multi-user system are automatically and reliably handled;
- Certain file operations, such as executing a program image stored on a file system, or the buffering of file system buffers for improved performance, are handled by the virtual memory mechanism;
- The address space available to each process is potentially larger than – and therefore, independent from – the actual amount of RAM available on the system;
- When RAM is oversubscribed, a second-stage storage mechanism, using the disk storage attached to the system, is utilized to transparently manage memory overlays between RAM and the second-level storage.

It is this last bullet point which can get out of control, resulting in “swap storms” that paralyze an entire computing system.

In Figure 2, we observe that a small portion of the operating system (outlined with a dashed line) is not demand paged at all.

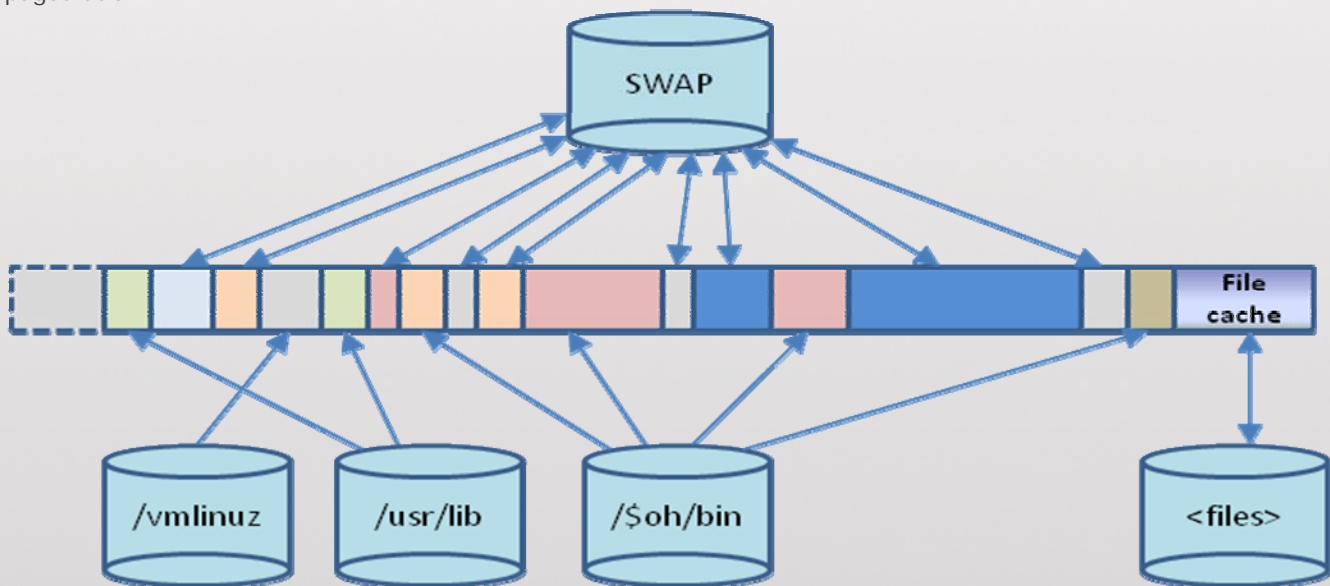


Figure 2. Source and direction for demand paging operations

Some of the executable code, such as SQL*Plus, httpd, the Oracle export utility and even the Oracle kernel itself can be paged in (only) from the executable image on disk into memory, as required. While this continues to require I/O operations, it does not result in quite the penalty required when “dirty” (modified) pages must be paged out prior to bringing in the newly demanded pages.

For the rest of memory, though, including user data, stacks and the Oracle SGA and PGA memory areas, the swap area¹ is utilized to manage the second stage memory. Typically this swapping disk area will utilize disk storage infrastructure shared with the operating system and even the database itself, including HDAs, storage array caches, controllers and physical connectors (cables) between the system and the storage array (or disk drive).

While it might be tempting to isolate the swap area to avoid this contention, it is usually more worthwhile simply to avoid extensive utilization of the swap area.

How Much Is Too Much?

No hard and fast rule or threshold can be offered for when demand paging has become excessive. It is entirely dependent upon the organization's IT history, culture, urgency and service orientation. In general, demand paging is out of control when system response and throughput slow to a point that normal work functions are measurably impaired.

How Bad Can It Get?

The degree of impact that a swap storm can have on a system is commonly expressed in orders of magnitude. This is illustrated in a simple exhibit below. In this test, a RedHat Linux 5.2 virtual machine hosted on a 2GB system was run in 4 different scenarios, with only the real memory available to the virtual machine changed in each test. Here are the particulars of the test scenarios:

Host System	Hewlett-Packard, AMD Turion 64bit, 3GHz, 2GB running Oracle Enterprise Linux 5.2, VMware server
Guest (virtualized) System	RedHat Linux 5.2, with varying memory sizes.
Oracle Database on the Guest System	Oracle 11.1.0.6
SGA	700M, with PRE_PAGE_SGA=TRUE specified
PGA	128M
DB_CACHE_SIZE	40,960,000
DB_BLOCK_SIZE	8,192
Test steps	Startup database instance Drop test table Create test table consisting of ~500,000 rows Build an index on the test table Take measurements and shutdown instance

Here are the elapsed time results with five different memory sizes:

Main Memory (MB)	Total Time	DB Startup	Index Creation	Swapped	si	so
1280MB	0:01:50	0:01:26	0:00:11	0	0	0
1120MB	0:03:11	0:02:23	0:00:20	0	0	0
960MB	0:03:05	0:02:25	0:00:15	82496	0	73
800MB	0:07:10	0:05:47	0:00:12	178752	103	169
640MB	0:37:53	0:29:46	0:04:20	267020	629	614

¹ The term "swap" is uniform terminology among both Linux and proprietary UNIX implementations. On Windows, it is called pagefile.sys.

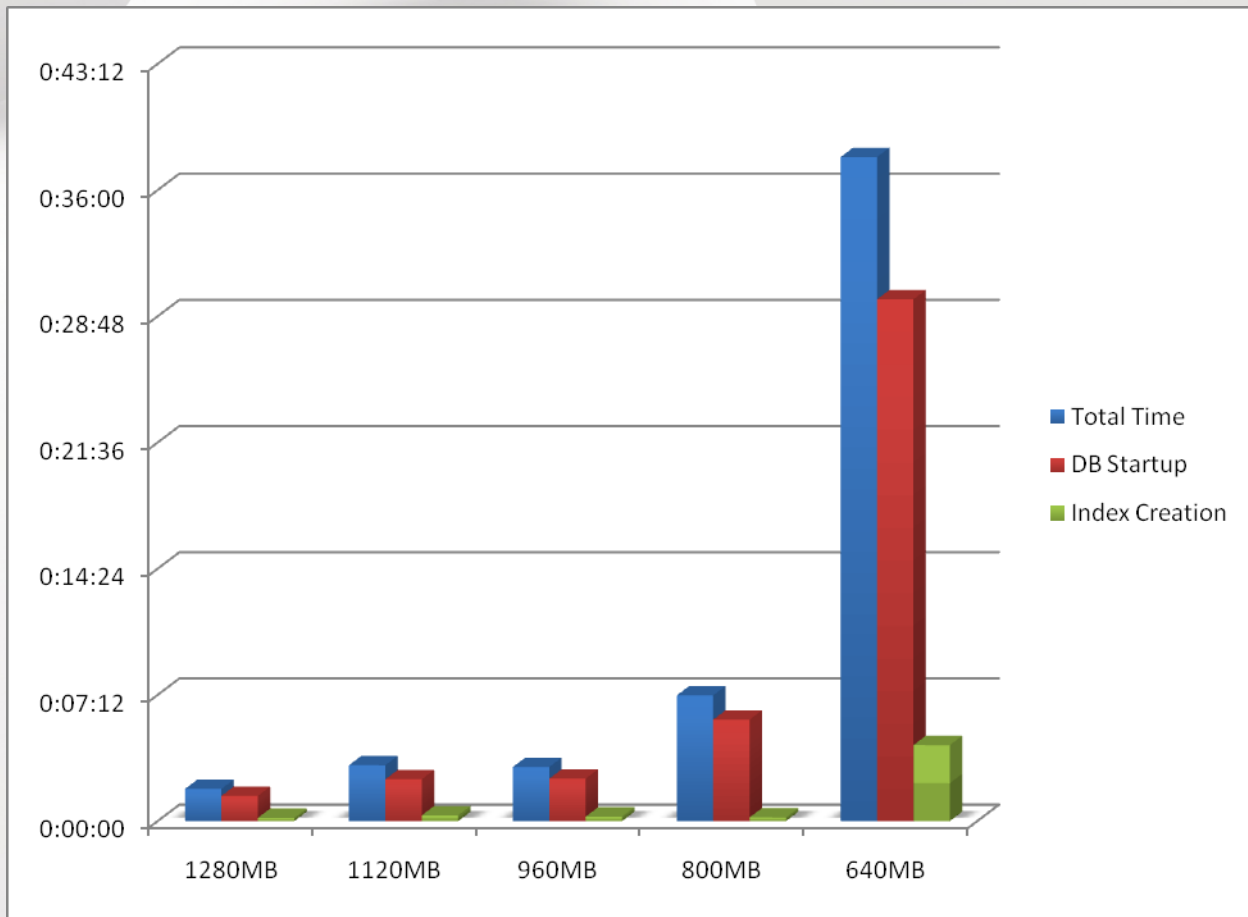


Figure 3. Actual Test results: Oracle under varying memory pressure

How Can I Measure and Monitor Paging?

It is a relatively easy matter to discern and quantify paging/swap activity that is taking place in the present moment (assuming that the system is not in a severe swap storm, such that it is totally non-responsive!)

Unfortunately, there is not a standard way to measure and correlate these events that might have occurred in the past; the sar utility for UNIX and Linux systems provides some capabilities in this regard, as does any system monitored by Oracle Grid Control. Of course, the total swap utilization on a UNIX or Linux system will provide a proxy of recent paging activity, but it is not an accurate tool, as swap utilization can continue to be high long after a problem has abated, or swap utilization can drop precipitously as the total system load profile changes.

On Windows

The best way to obtain a profile of paging activity on a Windows system is using the Windows Measurement Instrumentation (WMI) through the Microsoft Management Console (MMC). This interface includes such tools as the performance utility, which provides some historical context.

For an instantaneous snapshot of paging activity, utilize the Windows Task Manager, invoked by right-clicking the Windows taskbar, or by submitting CTL-Alt-DEL.

The Windows Task Manager will also allow you to drill down to the culprit(s), by adding additional columns (Memory Size and Virtual Memory Size) to the default process display.

On Linux and UNIX

Here are some of the informational tools available on Linux (and most UNIX) systems; these tools and utilities can help you to identify suspected paging activity:

- vmstat – the si and so columns of this report quantify the amount of paging activity taking place on the system. The swpd column shows the current amount of swap (paging) disk space that is currently in use.
- sar – Use sar –b to provide paging statistics.
- free – this is probably the simplest way to snapshot the current memory situation.
- top – this utility provides a quick, regularly updated snapshot of several aspects of system performance. The name varies according to the distribution, as follows:
 - Linux – top
 - Solaris – top (not provided, download from <http://www.sunfreeware.com/>)
 - AIX – topas
 - HP/UX – glance or glanceplus
- ps – Utilize the –eo pid,ppid,user,rss,size,cmd options to highlight top virtual memory and real memory consumers. (This is useful instead of top when you want to snapshot the processes into a log file.)

Six Case Studies

#1 Description of Problem	A data warehouse ETL cycle was experiencing ORA-00600 [2103] errors (Timeout on control file enqueue) in various Oracle processes. No pattern to occurrences and not reproducible at will.
Description of Platform	HP Superdome 24x32GB w/ 32TB HDS SAN; HP/UX 11; Oracle 9.2.0.6, 8TB data warehouse
Diagnosis	ETL analyst had discovered that setting sort_area_size session parameter to a large value would reduce elapsed time and resources for index builds, in isolation. Change was promoted to production where parallel ETL processes were running.
Proximal Cause	Excessive disk I/O due to paging activity (i.e., a swap storm).
Root Cause	Overcommitment of real memory due to parallel scheduling of multiple, memory-intensive index builds in the ETL process.
Remediation	The index build processes were modified to use more reasonable values for the session sort_area_size parameter. In addition, the ETL resource scheduler was modified to include the specified sort area size as a critical resource to be managed.
#2 Description of Problem	A data warehouse query/reporting environment was sporadically throwing error TNS-12170 (TNS: Connect timeout occurred) when users attempted to connect. The symptom only occurred when the system appeared to be busy, but it was not reproducible at will.
Description of Platform	IBM AIX P590 4x96GB w/ 60TB ESS; AIX 5.4; Oracle 10.1.2, 20TB data warehouse
Diagnosis	Multiple reports which utilized vast amounts of PGA (as well as other Oracle memory resources) were being scheduled to run at the top of each hour.
Proximal Cause	Excessive disk I/O due to paging activity (i.e., a swap storm).
Root Cause	Multiple processes starting at once were resulting in heavy paging activity, particularly page-outs on the Oracle SGA.
Remediation	SGA was pinned to real memory, and scheduling was modified so that reports commenced according to a more staggered schedule.
#3 Description of Problem	Several Apache servers running in distinct virtual machines on a single server were occasionally non-responsive or would return HTTP errors 503 (service temporarily unavailable) and 408 (request timeout), as well as other HTTP errors.

Description of Platform	Dell 2x2GB with 1x160GB SATA, RHEL4 VMs on RHEL4, using VMware Server 2.
Diagnosis	To resolve earlier performance problems, additional guest machines were deployed.
Proximal Cause	Excessive paging was occurring on the virtual machines and the host machine.
Root Cause	The mathematics of memory allocation was not allowing for the overhead of the VMware Server process, nor for additional overhead memory consumed by each of the virtual machines.
Remediation	Fewer virtual machines were deployed, with a more rational memory formula. The host was redeployed as VMware ESX, to provide additional memory management flexibility.

#4 Description of Problem	A business analytics data warehouse / datamart was not completing ETL in adequate time to be available for the day's business. Measured I/O elapsed times were found to be much slower than when the system was benchmarked at the time of installation, and were much slower than rated throughput.
Description of Platform	IBM AIX P570 8x16GB, AIX 5.3L, DS4800 SAN, total 8TB over 3 databases, Oracle 9.2.0.8, Informatica, Business Objects
Diagnosis	All three databases were up during ETL and each had recently been changed to specify a pga_aggregate_target =2G, and a new version of Informatica had been installed, along with some new ETL processes.
Proximal Cause	Huge swap storms were occurring, but were unnoticed, between 4am and 7am every day.
Root Cause	Configuration changes had gradually nudged the memory footprint during ETL to be just 1GB greater than the real memory available.
Remediation	One database was rehosted, and the remaining databases were changed to have a pga_aggregate_target of only 1600M. Monitoring is in place to alert, in the event that swapping exceeds a specified threshold.

#5 Description of Problem	A production database with many on-line users running a vendor-supplied software application experiences slower and slower performance the longer that it remains up, impacting service levels. Bouncing the server temporarily resolves the issue, but is an unacceptable long-term workaround. AWR reflects the poor performance, but it does not indicate a cause.
Description of Platform	IBM AIX P580 4x8GB, AIX 5.3L, ESS SAN. Oracle 9.2.0.6
Diagnosis	The Oracle listener is observed to be utilizing increasing and unbounded amounts of memory. By the time that the listener is utilizing just over 2.5GB, virtual memory paging begins to occur.
Proximal Cause	Memory utilization of Oracle listener is resulting in virtual memory thrashing.
Root Cause	A memory leak in the Oracle listener is using more and more memory.
Remediation	Short term: The Oracle listener can be bounced one or two times daily Long term: A patch to the Oracle listener – addressing the space leak – is applied.

#6 Description of Problem	A relatively new Oracle e-Business production server is practically grinding to a halt during heavy loads. The DBA initiates vmstat, but it requires several minutes to return its first output result.
Description of Platform	HP Itanium, 4x128GB, RHEL4 Linux, EMC SAN, Oracle 9.2.0.8, Apps 11.5.10.2. Oracle SGA is configured at 64GB.
Diagnosis	A decision had been (wisely) made to utilize hugepages for the 64GB SGA. Linux had been configured correctly for 64GB of hugepages, but /etc/security/limits.conf had not been changed to allow Oracle permission to use the hugepages.
Proximal Cause	A debilitating swap storm resulted because only 64GB of real memory was available for an Oracle instance with a 64GB SGA!!!

Root Cause	The specification of 64GB of hugepages to which Oracle did not have access, resulted in cutting the amount of available memory on the system to 64GB.
Remediation	Appropriate changes were made to /etc/security/limits.conf to allow Oracle to utilize the configured hugepages (which are never paged, so are pinned to real memory).

Are HugePages a Good Idea?

Most contemporary architectures today allow multiple page sizes. Typically, the traditional 4K page is the default for most of the workload, but much larger pages – usually in the range of 1MB to 4MB – are available to handle large, contiguous memory areas such as the Oracle SGA much more efficiently than is possible with the 4K page size. This is due to a reduction in the number of page table entries (PTEs) in the operating system. For example, a 64GB SGA (see Case Study #6, above), requires 16,000,000 page table entries when 4K pages are used, but only 64,000 page table entries if 1MB pages are used.²

In addition, hugepages are never demand paged to/from the disk, so they will always be resident in real memory. This is clearly a benefit for an Oracle SGA, for example, but if there is already an existing paging problem on a system, making a portion of the memory nonpageable is simply going to intensify the problem on whatever real memory remains for the workload.

Oracle will automatically “see” and utilize hugepages for its SGA, provided that it has permission to access the hugepages. Consider the ramifications for startup order if the system hosts multiple database instances, including an ASM instance.

These factors tend to suggest advisability of the use of hugepages for Oracle:

- The SGA is very large by contemporary standards (>16GB in 2009).
- The system is dedicated to a single Oracle instance or a small, stable number of Oracle instances.
- There is a professional DBA staff which understands the ramifications and implementation of hugepages and pinning of memory.
- High CPU utilization by the kswapd process (on Linux) or its equivalent, even when there is little or no disk paging activity occurring.
- Disk I/O due to virtual memory demand paging is either minimal, or is very clearly understood.

These factors tend to recommend against the implementation of hugepages:

- Virtual memory concepts are not clearly understood by the technical staff.
- You prefer to have a low-maintenance server environment.
- You currently benefit from the use of Oracle’s automatic memory management (AMM) feature.³
- The number, availability, configuration and size of Oracle instances varies from day-to-day (such as in a development or test environment).
- You lack administrative (root) access privileges, or at least a good working relationship with the system administrator.

Detailed implementation instructions for hugepage implementation are on My Oracle Support, as referenced in the References section, below.

² The size of the huge page is typically a property of the hardware architecture and cannot be defined or changed.

³ Oracle AMM and hugepages are mutually exclusive implementations.

Virtual Memory and Virtual Machines

Virtual memory and virtual machines can play nicely together, but there are even more opportunities for peril when both the host and the guest could individually or together incur a swap storm. As a general rule, the responsibility for memory management should be “pushed” to the specific guests. However, keep in mind that a swap storm – even one isolated to a single guest machine – is likely to have performance ramifications for the host system and for other guests running on that host system.

Carefully measure the actual, observed memory requirements – rather than simply attempting to sum the specified memory configuration; each VM will consume between 120% and 150% of the specified memory. Also take into account the memory requirements for the VM hosting code and services running on the host machine.

For serious, permanent hosting of virtual machines, the use of a true hypervisor kernel, such as VMware’s ESXi, is strongly recommended. Hypervisors typically provide memory management capabilities, such as transparent page sharing, that are not present on VM host products running on a general purpose operating system.

References

Huge Pages

[Doc 744769.1 - How to Configure HugePages for Oracle Database on 64-bit Linux Platforms](#)

[Doc 745895.1 - HugePages and Oracle Database 11g Automatic Memory Management \(AMM\) on Linux](#)

[Doc 361323.1 - HugePages on Linux: What It Is... and What It Is Not...](#)

[Doc 361468.1 - HugePages on 64-bit Linux](#)

[Doc 422844.1 - Using Large Memory Pages on 64-Bit Windows Systems](#)

About the Author

Randy Cunningham is Principal Solutions Architect for SageLogix, Inc. He was previously with IBM Global Services in Denver. He has worked with Oracle since version 4, and has been consulting to Oracle clients exclusively for over 15 years. His primary consulting focus is on architecture, security, availability and performance optimization of large, business-critical databases.

For questions, please contact info@sagelogix.com