Avoiding paging with shared memory in AIX 5L Version 5.2 with Sybase database servers

Peter Barnett
Storage and Technology Group
Ramesh Chitor
ISV Business Strategy and Enablement
July 2006

Table of contents

Abstract

This white paper presents methods for reducing or eliminating paging space consumption on AIX 5L. The authors replicated several Sybase server and user scenarios, including instances where memory is constrained or unconstrained, and instances where multiple servers exist on a single host. Through memory allocation and various virtual memory tuning methods, the tests proved that it is possible to avoid paging with Sybase database servers.

Introduction

IBM® AIX 5L™ Version 5.2 retains the mapped paging spaces of running processes, even when they are paged back into physical memory. In a Sybase ASE relational database management system or other RDBMS server with limited memory and large batch processes, this creates the risk of running out of paging space. However, this exposure is secondary to the degradation of performance that can occur with paging. Although scripts are available to clean paging space, and AIX 5L V5.3 has implemented garbage collection for paging space, it is still wise to eliminate paging through judicious memory allocation and correct virtual memory tuning.

For this white paper, the authors have recreated the conditions of paging space exhaustion and demonstrated how to avoid it, even with limited physical memory. In every client case encountered, the authors have eliminated paging through memory allocation and virtual memory tuning. The authors also show that pinning shared memory is a viable option to avoid paging in Sybase servers. The traditional recommendation is that paging space (also known as *swap space*) be allocated at four times the size of physical memory. But, the authors go on to say that this recommendation applies to the early allocation paging algorithm. The default delayed-allocation algorithm for AIX 5L V5.3 requires less paging space. The *Best Practices Guide* recommends allocating two times the amount of physical memory for systems of 256 megabytes or less and an amount equal to physical memory (or 4 gigabytes, whichever is less) for large memory systems. The AIX support line typically recommends creating a second swap space of the same size, so that it can be activated when the primary swap space needs to be flushed.

In database servers, chronic paging can result from the overallocation of memory as databases grow (either because of incorrect or incomplete virtual memory tuning, or from an overconcentration of batch jobs that generate file cache demands). No matter how much paging space is allocated, chronic paging fills it up over time and causes the system to become unresponsive. This paper shows how to avoid paging in database servers, using Sybase as the illustration, by monitoring database memory requirements, appropriate virtual memory tuning, and the careful scheduling of batch jobs (such as loads and dumps).

AIX 5L V5.2 does not release the pages of running processes even after there is available physical memory and the working set is paged back into it. This algorithm is presented as an intelligent design, which avoids the need to remap the page spaces again in the event that the same running process pages out memory.
AIX 5L V5.3 implements a tunable-paging-space garbage-collection routine. For AIX 5L V5.2 environments, AIX support staff and IBM clients have created scripts to clean out paging space periodically.

In AIX 5L V5.2, transient processes release all paging space as they terminate and, therefore, the use of paging space fluctuates over time. The authors observed this in the field and in laboratory testing.

Additionally, in the case of persistent processes such as RDBMS, although computational memory is repaged into physical memory, the swap space is not given back.

Sybase ASE is designed to employ monolithic shared memory (for managing all internal threads and to communicate between engines). Because Sybase ASE typically stays up for a long time, its paged-out pages are not recovered by the AIX 5L V5.2 system until Sybase ASE is restarted or the operating system is rebooted.

In a typical Sybase server, dedicated shared memory can consume 75 percent of physical memory. In a relatively small system (for this example, 6 gigabytes), the fixed-memory requirement of the operating system and extensions (900 megabytes) together with Sybase memory (4.5 gigabytes) can leave so little free physical memory (only 600 megabytes) that paging will develop. During batch maintenance jobs, this memory is used up, and because most of the pages are in use, the LRU daemon cannot free enough pages to satisfy allocation demands. As a result, idle pages of the Sybase server itself are pushed into paging space.

Most of the nonreturned paging space can be known to belong to Sybase ASE. The reason for this is that the page is released only when Sybase ASE is stopped and restarted.

If the Sybase data server is never restarted and the operating system is never rebooted, the system will eventually run out of paging space and become unresponsive. Although the system can still be pinged and Sybase users might actually be able to connect to the data server, no one can log on to the system as a UNIX® user. Furthermore, those already logged on (even as root) are not able to run most commands. The best way to recover is for the Sybase superuser (sa) to log into the data server and shut it down, which releases enough paging space to allow the UNIX system administrator to regain control of the operating system. It also avoids possible database corruption that can result from a shutdown (–Fr) or a Hardware Management Console (HMC) reboot while the data server is running. Rebooting from the console or HMC is the last resort.

Eliminating paging when memory is constrained

It is possible to eliminate paging through the combination of adequate memory for transient processes and virtual-memory tuning according to IBM AIX® best practices. The paging of any application, including Sybase, results in degraded performance. Thus, avoiding potential paging problems is the payoff for the primary discipline of optimizing performance. Whether through an ad hoc script or an operating system process, merely cleaning up paging space does not address the root of the problem.

It is possible to calculate adequate memory (meaning that it is adequate for current operations, not necessarily for growth) by combining the requirements of the Sybase ASE, AIX and its extensions, and by observing the requirements of transient processes when paging occurs. In a particular client example, the authors calculated adequate memory to within 50 megabytes and eliminated paging simply by reducing Sybase excess memory by that amount. Obviously, this calculation is too close to accommodate growth; it was done merely to prove the point that it is easy to eliminate paging.

For this paper, the authors created a close approximation of an actual enterprise database on AIX 5L V5.2 and Sybase 12.5 in the IBM Linux® lab at 590 Madison Avenue, New York City. The IBM eServer™ pSeries® 650 logical partition (LPAR) was allocated 3 gigabytes of memory and, of this memory, 80 percent was dedicated to the Sybase data server. The operating system was tuned according to best practices. The authors created three types of stress jobs, one that did backups of all the databases, one that copied and compressed files totaling more than 1 gigabyte, and one that ran ISQL scripts involving 200 users. When all these jobs were run repeatedly, with very little "think time" between them, the LPAR ran out of paging space and the system stopped responding in 30 minutes.

By adjusting the rate and timing of the jobs, but otherwise leaving them unchanged, the authors provided the system with enough time to recover memory, eliminating paging without changing either Sybase memory or system memory. The jobs ran indefinitely without filling up swap space.

Eliminating paging when physical memory is sufficient

A second condition might result in a depletion of paging space that has been observed in several client environments involving database servers with large amounts of shared or pinned memory. This is a case in which there is adequate free memory to serve transient jobs, but virtual memory (VM) is either untuned or incorrectly tuned for a database server.

The authors reproduced this condition in the IBM Linux Lab test environment by giving the operating system 25 percent more memory, but also setting certain operating system VM tuning parameters (minperm% and lru_file_repage) to their defaults. Paging occurred even though there was free physical memory, a phenomenon that AIX performance support staff has documented. Eventually, although the LPAR had 1.5 gigabytes of free memory, when the three sets of stress jobs were run simultaneously, the LPAR ran out of paging space in an hour.

The authors recently achieved a 41 percent performance improvement on a client's database server that was in the condition described before. This was done by making seven simple changes to the AIX `vmo` command's tuning variables, in accordance with the recommendations from AIX performance support staff. Assuming database shared memory is 75 percent of the total memory, the following tuning parameters were set:

- `maxperm%=25`
- `minperm%=5`
- `maxclient%=20`
- `lru_file_repage=0`
- `lru_poll_interval=10`
- `strict_maxperm=0 (default)`
- `strict_maxclient=1 (default)`

Certain IBM pSeries performance specialists recommend leaving the maxperm% and maxclient% parameters at their defaults (80%). The authors have tried both this approach and the approach of setting the maxperm% and maxclient% parameters according to the size of the database shared memory. They found that the two approaches work equally well. In the same way, paging has been eliminated in all other cases of this syndrome that has been encountered. (See Barry Saad's *AIX 5.3 Memory Tuning-Updated Guideline* for a comparison of the classical VMO tuning methods and the recommended VMO tuning methods that apply to AIX 5L V5.2 ml 5 and later [as well as to AIX V5.3]).

Even when memory is overallocated, if the operating system is not tuned or incorrectly tuned, paging can be a problem. You have to tune VM and I/O parameters for database servers, whether using the Oracle Database Server, Sybase ASE, or IBM DB2® product.

Pinning Sybase shared memory to prevent paging

In this section, the authors describe tests of pinning Sybase shared memory with AIX 5L V5.2. There is disagreement on the wisdom of pinning memory for database servers among database administrators (DBAs), system administrators (SAs) and performance experts. For example, IBM pSeries Advanced Technical Support (ATS) specialists on other databases generally recommend against pinning. In contrast, traditional database shops do a lot of pinning.

There is a misconception in some Sybase shops that pinning Sybase memory does not work on AIX. The authors proved that it does work by implementing pinned shared memory with the following configuration parameter:

```
lock shared memory =1
vmo tunable: v_pinshm=1
```

The operating system correctly reported the amount of memory pinned by Sybase. The authors also deliberately overallocated Sybase memory and then tried to pin it. This froze the system, because it attempted to pin more memory than was available, either physically or in swap space.

The authors pinned enough memory in Sybase to recreate the situation of not having enough free memory to run the transient stress tests. Given unpinned memory, serious paging can result, but in the case of pinned memory, there was no paging at all. However, the stress jobs, especially the Sybase backups, reported having to wait until there was enough memory for them to run.

These tests show that pinning memory in Sybase not only works, but also prevents paging. However, take care not to overallocate pinned memory and, thus, degrade the performance of transient jobs that rely on free memory. These tests also indirectly confirm that Sybase itself is being paged out; otherwise, the authors would have observed paging because of the database backups and SQL jobs.

In a final test in which just enough memory was allocated for observed performance, Sybase memory was unpinned, the intervals between the jobs were reduced to a minimum, and the authors were still able to run three sets of stress jobs at one time without any paging. Free memory remained at a minimum, but owing to operating-system tuning, the least recently used (LRU) page always managed to find enough physical memory for the transient jobs.

Multiple Sybase ASE servers on a single host

In most industry applications, applications such as Sybase ASE tend to have processes with large virtual address spaces. This is typically the result of attaching to large shared memory segments used by Sybase ASE and large copy-on-write (COW) segments that are mapped but sometimes never actually get touched. The net effect of this is that, on this host supporting multiple Sybase data servers, the virtual address space requirements can grow quite large, typically exceeding the physical memory size of 8 gigabytes (as in this example). Consequently, a fair amount of swap disk must be configured to support these multiple Sybase servers with large virtual address space that runs concurrently. As a result, the

system administrator needs to configure one-and-a-half or two times the amount of RAM for swap for the ASE processes to use all of the physical memory fully without running out of virtual swap space.

Knowing how much memory Sybase actually uses

There are differences in interpretation concerning memory allocation in Sybase that can result in overcommitting memory. There is a Sybase configuration parameter (`allocate max shared memory`) whose default is "allocation on demand." As more users come online, the administrator sees the operating system allocate additional shared memory segments. Some Sybase DBAs prefer to allocate maximum memory in advance (`allocate max shared memory=1`), so that it is all available on Sybase startup.

However, Sybase differentiates between *logical memory* (the memory required by the details of configuration) and *max memory* (an amount that cannot be exceeded in configuration). The difference between these two values is referred to as *headroom* (that is, memory that can be configured without halting and restarting the database server). You can observe the amount of headroom through the transact SQL `sp_configure "memory"` command.

Suppose that *max memory* is configured at 4 gigabytes, but the detailed configuration only requires 3 gigabytes. It is generally believed that when you use the `allocate max shared memory=1` parameter, 4 gigabytes of shared memory is actually allocated on startup. This is only partly true. In the output of the `ipcs -am` command, the Sybase shared memory segments actually add up to 4 gigabytes. However, the amount of computational memory reported in the AIX `vmstat` output (the avm column) on a dedicated Sybase server (minus the operating system kernel shared memory) is almost exactly equal to the Sybase logical (configured) memory value.

The difference between these two interpretations can be misleading. The DBA might think that memory is not overcommitted because Sybase comes up and runs. In fact, Sybase is using 3 gigabytes, not 4. If, in the mean time, the DBA adds users, databases, caches and other objects (up to the max memory amount), system memory might be overcommitted, and paging can develop. Even though the `max memory` parameter is unchanged, the headroom is used up.

Summary

This paper discussed various methods for reducing paging space consumption on AIX 5L. The authors replicated several Sybase server and user scenarios, including instances where memory is constrained, instances where memory is not constrained, and instances where multiple servers exist on a single host. Through memory allocation and various virtual memory tuning methods, the tests proved that it is possible to avoid paging with Sybase database servers.

Resources

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

- IBM System p™ Information Center
  http://publib.boulder.ibm.com/infocenter/pseries/index.jsp

- IBM Publications Center
  www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi?CTY=US

- IBM Redbooks™
  www.redbooks.ibm.com/

- *Optimizing Sybase ASE for IBM AIX 5L*
  ibm.com/servers/enable/site/peducation/wp/b78a/index.html

- *Exploiting Micro-Partitioning and SMT with Sybase ASE on IBM AIX 5L Version 5.3 / POWER5*
  ibm.com/servers/enable/site/peducation/wp/9c82/index.html

- *Sybase ASE 12.5.2 for IBM Linux on POWER™: An Installation and Migration Guide*
  ibm.com/servers/enable/linux/power/pdfs/sybase_lop.pdf

- *Lowering TCO for applications running Sybase ASE on IBM LoP, POWER5™*
  ibm.com/servers/enable/site/peducation/wp/a2a6/index.html

About the authors

Peter H. Barnett, Ph.D.
IBM Systems and Technology Group

Peter Barnett is a certified AIX and System p specialist with a particular competence in financial sector solution providers. He is a member of the IBM Northeast Field Technical Support Specialist (FTSS) team.

Ramesh Chitor
ISV Business Strategy and Enablement

Ramesh Chitor works as a solutions engagement manager, advisory software engineer for ISV Business Strategy & Enablement with primary responsibilities in middleware, databases and middleware support.

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.

AIX, AIX 5L, DB2, eServer, IBM, pSeries, Redbooks and System p are trademarks of International Business Machines Corporation in the United States, other countries, or both.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

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.