close window

Web Exclusive

Print

Debugging AIX Memory Leaks

February 2016 | by Jaqui Lynch

One of the most challenging debugging projects in AIX or any other operating system is the memory leak.

Memory is classified in one of three ways – file system cache, free memory or computational memory. When a program starts, the code and working areas are loaded into computational memory. The application is supposed to release unused memory and the system is supposed to free up memory after a process ends. A memory leak is a program error where the program repeatedly allocates memory and uses it, but then neglects to free it. This causes programs to consume excessive memory and can lead to degraded system performance or even a system crash.

There are two kinds of memory leaks. A physical memory leak is when a program allocates memory and then loses the reference to it, most likely because the pointers were somehow overwritten. A logical memory leak is where memory gets allocated but the program never frees it.

The memory to be concerned about is computational memory, which consists of processes, shared memory segments, TCP/IP connections, etc. When there is insufficient computational memory, the system will start to page with the least recently used daemon (LRUD), copying older pages to the AIX page space and then reassigning those pages to the free memory list. Since disk is much slower than memory, this affects performance and paging should be avoided at all costs.

Tools for dealing with memory leaks

Memory leaks can be difficult to detect and even harder to correct. Fortunately, AIX has many tools that can be used to examine memory leaks.

lsps -a and lsps -s: Checks page space usage.
topas or nmon: Examines overall health and usage.
vmstat -I and vmstat -v: Provide more detailed information, including paging and avm (active virtual memory).
svmon -G and svmon -P: svmon can be used to look at global statistics (-G) or memory usage for specific processes (-P). Additional options allow you to look at segments, login names and command names.
ps gv: Provides process specific memory usage.
probevue: This dynamic tracing facility allows you to trace processes and look more deeply into what they're doing. Because it creates significant overhead and consumes a considerable amount of memory, probevue should be used with care.
dbx: Allows you to investigate malloc commands. It's another tool that should be used with great care. truss: This can be used by dynamically attaching it to a process in order to watch malloc and free commands.
profilers: A number of profiling commands can be used once the problem process has been determined – these include prof, tprof and gprof. Profilers can be used only if you have the program source as well as the program language.

How to determine if there's a memory leak

The first step to determining whether there's a memory leak is to run vmstat or svmon -G multiple times over a specific interval. Check the avm column in vmstat or the virtual column in svmon to determine if memory is continually increasing over time. This may indicate a memory leak, but you'll need to delve a little deeper. The next step is to run the ps gv command several times and look at the SIZE column.

vmstat
r b p w avm …………
10 0 0 0 5645686 …………
You can see that the system has 5645686 pages in AVM. If you ran this several times and graphed it, a memory leak would be indicated if the page number was continually increasing. If there is a memory leak, you should run ps gv several times to determine which process is the problem.

svmon -G
When looking at svmon output, the default units are 4K pages except when a pagesize is explicitly referenced. When running the command you can specify unit=MB or unit=GB to work in sizes that make more sense. I normally specify unit=GB.

svmon -G provides a global view of memory and can be used similarly to the vmstat output above. In **Example A**, the units being reported are GB rather than the default 4K pages. In the inuse column on the memory row, you can see that the LPAR has 250GB of memory: 85GB of memory is inuse and 164.96GB of memory is free. Those memory details are further broken down between working storage (computational), persistent (filesystems), client and other. Other represents memory that is not associated with a segment.

In this example, 21.5GB shows as inuse computational (work) memory. The virtual column for the memory row also shows 21.5GB is in use. This should match up to the avm column in the vmstat output above (and it does). To compare them you need to translate the 4K pages shown in vmstat to GBs. The conversion is as follows:

```
5645686 x 4 (gets it to 1KB pages)
Then divide by 1024 (1MB) and 1024 (1GB)
Or:
(((5645686*4)/1024)/1024)= 21.5GB
```

Depending on your preferences you can monitor growth in avm using either the vmstat command above or the svmon -G command. Either way, a continual increase indicates a memory leak.

Identifying the process with the memory leak

Once a memory leak is detected, the next step is to determine the problem process and then debug that process. To identify the actual process, ps gv or svmon -P can be used as starting points.

```
ps gv
PID       TTY STAT  TIME   PGIN  SIZE     RSS      LIM  TSIZ   TRS %CPU %MEM
9895988   - A       4:44   23445 631572   636492   xx   1453   4920 0.3  1.0      cmd
```

You can see that process 9895988 has 631572 pages in the SIZE column. Again, by monitoring this over time you can determine which process is the problem. By using the ps gv with the flags, you can ensure the output is sorted in SIZE order:

```
ps gv | head -n 1; ps vg | egrep -v "SIZE" | sort +5 -r | head -n 3
```

The command runs ps gv and sorts by SIZE showing only the top five. By comparing three or four sets of results, you can see if one of the processes is continually increasing in SIZE. If so, you now know the process that needs to be debugged.

svmon -P
Another way to find this information is with svmon -P:

```
svmon -P -t 1 -i 5 3
```

This lists the top item (-t 1) using a 5-second (-i 5) interval and repeats it three (3) times. Alternatively, this command shows the memory consumption of process id nnnnnn in full detail:

```
svmon -P nnnnnn  -O segment=on,pidlist=on,range=on,mapping=on,shmid=on,filename=on,affinity=detail
```

In the svmon -P output start by looking at the Virtual column; this lists the pages in virtual memory for that process. This should be stable over time (as opposed to continually increasing). If it is increasing, this is the offending process.

At this point you've determined that there is a memory leak and have identified the problem process. Follow-on actions will depend on what the offending process is. If this is a third-party application, it will be necessary to work with the vendor to debug it. If it's Java, you can find guides online – just search on “aix java memory leak.” If this is a home-grown C or C++ application, you can use one of the various AIX profilers (prof, tprof or gprof) to trace and analyze the application. This will allow you to see the memory allocation and free commands in the application. You may also be able to take advantage of the AIX Memory leak detector that is available on sourceforge. (Note: Because I haven't used this, I recommend testing it first on a non-production system.) IBM also provides some malloc debug tools (see references). However, these may not work with your application, as much of the documentation on these is dated.

The Importance of Staying Current

AIX provides several useful commands to assist in memory leak identification and debugging. Proper use of these commands should make it much easier to diagnose and resolve memory leaks.

Additionally, it's important to stay current on applications as well as operating system levels. As an example, there are known memory issues with AIX 6.1 tl09 sp1 and AIX 7.1 tl03 sp1. Back-level firmware can also impact memory. Finally, there are specific recommendations for Java versions to be run on the various POWER server architectures.

Memory leaks can greatly slow performance and even lead to a system crash if they cause significant paging. It's important to have a well planned approach to identifying memory leaks, along with established, easy-to-follow guidelines for pinpointing the offending process. It's possible to accomplish everything you need using native AIX commands, although large, complex environments may want to consider third-party products that are less manually intensive while providing the same capabilities.

close window

Print Email

Example A - Sample svmon -G Output

```
svmon -G -i 2 2

Unit: GB
-----------------------------------------------------------------------
----------------
                     size           inuse         free            pin
virtual    available   mmode
memory       250.00        85.0      165.02         16.3         21.5
220.54     Ded
pg space     200.00        0.07

                 work         pers       clnt        other
pin            8.90            0        0.01         7.39
in use     21.5              0        63.5

                   size                    inuse        free
pin             virtual       mmode
memory     65536000    22274954    43261049     4274745     5640958
Ded
pg space   52428800       17775

               work         pers         clnt        other
pin          2333817      0         2816      1938112
in use     5640958      0     16633996
```

References

< Return to main article

Understanding Java and Memory Usage
https://www-01.ibm.com/support/knowledgecenter/SSYKE2_7.0.0/com.ibm.java.aix.70.doc/diag/problem_determination/aix_memory.html

AIX Memory Problem determination for WAS
http://www-01.ibm.com/support/docview.wss?uid=swg27036053&aid=1

Sourceforge aixmem application
http://sourceforge.net/projects/aixmem/

Malloc Debug Information
https://www-01.ibm.com/support/knowledgecenter/ssw_aix_53/com.ibm.aix.genprogc/doc/genprogc/debug_malloc.htm http://www.ibm.com/developerworks/aix/library/au-mallocdebug.html

< Return to main article