IBM Systems MAGAZINE

Issue Date: Open Systems edition
August | September 2006

Tuning a Perfect Note

Jaqui Lynch
jaqui.lynch@mainline.com

New commands in AIX 5.3 are music to administrators' ears

As more mission-critical work finds its way to distributed systems, performance management for those systems is becoming more important. The goal for systems management is not only to maximize system throughput, but also to reduce response time. To do this, it's necessary to not only monitor and tune the system resources, but also to work on profiling and tuning applications. In previous articles, we looked at specific areas of performance tuning, primarily those related to I/O tuning. In this article, I'll look at tuning in general and some of the new commands in AIX* 5.3, and I'll provide a starter set of tunables for the reader to test.

Time

In AIX, several major resource types must be monitored and tuned, including CPU, memory, I/O time, network time, and perhaps some serial or modem time and application programs. From the user's perspective, the only time they see is total execution time, so we'll start by looking at that.

Total execution time, from a user's perspective, consists of wall-clock time. The components of total running time include user-state CPU, system- or kernel-state CPU, I/O time and network time, and time when the processor is running a different application and your application is waiting (run Q time).

When examining performance problems, or even just taking a baseline measurement, it's important to take a phased approach to reviewing the problem. I normally run a script to gather all of the data I need (see "Helpful Links" sidebar) and I also run nmon in logging mode so I have some additional data. I then review the data. First, I look at processor-related issues, then I/O, then paging, and finally I check to see if anything looks untoward. I review all of the tunables settings and figure out what needs changing. I also check all firmware levels on the servers and cards, and I check the maintenance level on the software. Finally, if I still can't see the problem, I resort to running traces.

With the advent of AIX 5.3, not only were new commands added to the performance tools suite, but some of the default recommendations changed. Previously, the list of commands included the following: topas, sar, vmstat, iostat, ps, svmon, netstat, entstat, nfsstat, filemon, fileplace, netpmon, lspv, lsvg, lsattr, lsdev, lslv and lsps. In later maintenance levels of AIX 5.2 and/or AIX 5.3 these commands were supplemented with some new commands and were also made fractional LPAR aware.

There are a couple of applications that you can download from the IBM* Web sites that are extremely useful (again, see the "Helpful Links" sidebar). On Alphaworks, you can get the Graphical LPAR Monitor for System p5* platforms. This allows you to graphically monitor resources being used by one or more LPARs at the same time. Additionally, you can download nmon from the IBM Wiki and use that to monitor the system in real-time or log performance over a period of time. You can then download the nmon analyzer, which runs on Windows* and uses Microsoft* Excel, to crunch your nmon data.

Parameters

In AIX 5.3, you need to have transitioned from using vmtune and schedtune to the replacements - ioo, vmo and schedo. Before this year, the general recommendation for vmo (or vmtune) tuning was to set minperm to 5, maxperm to somewhere between 30 and 50 and maxclient to something lower than maxperm but higher than minperm. Since maxperm is a soft limit and maxclient is a hard limit, we also recommended in some cases that you should change strict_maxclient to set this to a soft limit. These recommendations have recently changed. Before going into the new recommendations, it's necessary to understand what these parameters do.

- minperm specifies the value for file pages (%), below which the page stealer steals both file and computational pages.
- maxperm specifies (for JFS) the value for file pages (%), above which the page stealer should steal only file pages. The default is 80%, but this isn't a hard limit. If you set maxperm to 30%, and the pages are available, then I/O can still use up to 100% of memory. What you're changing is which pages get stolen when a page is needed. Maxperm includes ALL file pages, not just JFS, whereas maxclient is only the JFS2/NFS/GPFS subset of pages.
- maxclient allows you to further control how much memory is used for caching client (NFS) and JFS2 pages. It can't be

set to a value higher than maxperm and it's a hard limit.

When the percentage of file pages falls between minperm and maxperm, then the page stealer steals file pages - unless the number of file repages is higher than the number of computational repages.

A key difference between maxperm and maxclient is how they're limited. When maxperm is set to 30%, it caps maxclient at 30% automatically, as maxclient can't exceed maxperm. However, maxperm is a soft limit (controlled by strict_maxperm, which defaults to 0 or off). This means that file caching can use more memory than 30% if it's available. Maxclient is a hard limit, so setting this low arbitrarily limits memory for JFS2 caching. We used to alter strict_maxclient. When this is set to 0, it alters the maxclient percentage to be a soft limit.

The new recommendations are to leave maxclient and maxperm at their default settings of 80, but to still set minperm to something like 5. We also don't change the strict settings. Instead, we alter other parameters as follows:

```
vmo -p -o minperm%=5
vmo -p -o lru_file_repage=0
vmo -p -o lru_poll_interval=10
```

Effectively, what setting the two lru values above does is the following: If filesystem buffer pages exceed minperm (in this case 5%), the LRUD (AIX page-stealing daemon) will show preference to executables, always choosing filesystem pages if possible. Setting lru_file_repage to 0 causes the LRUD to do this. The lru_poll_interval is set to improve the responsiveness of the LRUD process while it's running. The combination of these two parameters should give better and more flexible results than limiting maxperm and maxclient. You're basically setting a simple rule. Lru_file_repage is available on AIX 5.2 as well, so long as the system is at ML5 or higher.

Previously, when you ran vmtune it would tell you the settings for what is currently in use (numperm and numclient). These can now be obtained using the vmstat -v command as can be seen below:

```
vmstat -v
  20.0 minperm percentage
  80.0 maxperm percentage
  38.9 numperm percentage
  37.2 numclient percentage
  80.0 maxclient percentage
  2 pending disk I/Os blocked with no pbuf
  1717 paging space I/Os blocked with no psbuf
  47887 filesystem I/Os blocked with no fsbuf
  0 client filesystem I/Os blocked with no fsbuf
  310 ext pager filesystem I/Os blocked with no fsbuf
```

As you can see, both numperm and numclient are reported in percentages - these two fields tell you how much memory is being used by JFS, JFS2, GPFS and NFS filesystems. To figure out the JFS number, subtract numclient from numperm.

The last five lines of the vmstat -v report are useful when you're looking for I/O problems. The first line is for disk I/Os that were blocked because there were no pbufs. Pbufs are pinned memory buffers used to hold I/O requests at the logical volume manager layer. Prior to AIX v5.3, this was a systemwide parameter. It's now tuneable on a volume-group basis using the lvmo command. The ioo parameter that controls the default number of pbufs to add when a disk is added to a volume group is pv_min_pbuf, and it defaults to 512. This specifies the minimum number of pbufs per PV that the LVM uses, and it's a global value that applies to all VGs on the system. If you see the pbuf blocked I/Os field above increasing over time, you may want to use the lvmo -a command to find out which volume groups are having problems with pbufs and then slowly increase pbufs for that volume group using the lvmo command. I normally increase the global value to 1,024.

Paging space I/Os blocked with no psbuf refers to the number of paging space I/O requests blocked because no psbuf was available. These are pinned memory buffers used to hold I/O requests at the virtual memory manager layer. If you see these increasing, then you need to either find out why the system is paging or increase the size of the page datasets.

Filesystem I/Os blocked with no fsbufs refers to the number of filesystem I/O requests blocked because no fsbuf was available. Fsbufs are pinned memory buffers used to hold I/O requests in the filesystem layer. If this is constantly increasing, then it may be necessary to use ioo to increase numfsbufs so that more bufstructs are available. The default numfsbufs value is determined by the system and seems to normally default to 196. I regularly increase this to either 1,024 or 2,048.

Client filesystem I/Os blocked with no fsbuf refers to the number of client filesystem I/O requests blocked because no fsbuf was available. Fsbufs are pinned memory buffers used to hold I/O requests in the filesystem layer. This includes NFS, VxFS (Veritas) and GPFS filesystems.

Finally, ext pager filesystem I/Os blocked with no fsbuf refers to the number of external pager client filesystem I/O requests blocked because no fsbuf was available. JFS2 is an external pager client filesystem. If I see this growing, I typically set j2_nBufferPerPagerDevice=1024.

Other Tuning Suggestions

It's recommended that you tune the amount of memory to keep free. Maxfree specifies the number of frames on the free list at which page-stealing is to stop. Maxfree must be at least eight greater than minfree, which is the minimum number of frames on the free list at which the VMM starts to steal pages to replenish the free list. The difference between minfree and maxfree should always be equal to or greater than maxpgahead. Minfree and maxfree used to default to 120 and 128 respectively, though in recent OS levels they now use a calculation - so it's quite common to see minfree=960 and maxfree=1088 or even higher numbers. You should also look at maxpgahead and j2_maxPageReadAhead and set them accordingly for sequential read ahead. This is done using ioo.

Additional performance benefits can be gained by tuning the VMM write-behind parameters. When pages are updated in memory, they're not immediately written out to disk. Instead, dirty pages accumulate in memory until one of the following occurs:

- The syncd daemon runs (usually every 60 seconds)
- The number of free pages gets down to minfree
- Someone issues a sync command
- A VMM write-behind threshold is reached

When the syncd daemon runs, it obtains a lock on the i-node and holds that lock until all the dirty pages have been written to disk. Anyone trying to access that file will be blocked during the time the lock is held. On a busy system with a high I/O workload, this can cause a lot of I/O wait and dramatically effect performance. This can be dealt with in three general ways:

1. Change syncd to run more often
2. Set sync_release_ilock to 1 - this causes sync to flush all I/O to a file without holding the i-node lock, and it will then use the i-node lock when it does the commit (this can be dangerous!)
3. Turn on random and/or sequential write-behind

My general approach is to customize random write-behind. I rarely, if ever, modify the syncd interval.

Numclust is used to control sequential write-behind. Files are partitioned into 16k partitions or four pages by default - these are called clusters. If all four pages in a cluster are dirty, then - when the first page in the next cluster is modified - the VMM will schedule the four dirty pages to go to disk. The default is one cluster for JFS and eight clusters for enhanced JFS, but they can be increased to delay the writes. The JFS2 equivalent is j2_nRandomCluster which defaults to 0. The j2_nRandomCluster option specifies the number of clusters apart two consecutive writes must be in order to be considered random.

Random write-behind can make a dramatic difference to system performance with a great deal of random I/O. If many pages have been modified, then you will see large bursts of I/O when the syncd daemon runs and this will affect the consistency of performance. Maxrandwrt can be used to provide a threshold where dirty pages will be written out to disk. I tend to start with 32 (the default is 0 or never). This means that, when 32 pages are dirty, any subsequent dirty pages will be written to disk. The initial 32 pages will be written out when the syncd daemon runs. For enhanced jfs, the equivalent is j2_maxRandomWrite, and it defaults to 0.

Fine-Tuned

By default, AIX is a smooth-running system. However, it benefits from some help. In this article, I have touched on some of the more important tuning variables, some of the new commands and how our recommendations are changing. In addition to the "Reference" sidebar, be sure to review the starter set of tunables for your system provided in the sidebar above. Please keep in mind that these should be tested and your mileage may vary.

Helpful Links

1. Lparmon - www.alphaworks.ibm.com/tech/lparmon

2. Nmon - www.ibm.com/collaboration/wiki/display/WikiPtype/nmon
3. Nmon Analyser - www.haw.ibm.com/collaboration/wiki/display/WikiPtype/nmonanalyser
4. Instructions for running the performance script and the script itself can be found at www.circle4.com/jaqui/perf-script-instructions.txt and www.circle4.com/jaqui/perf-script-sh.txt
5. Jaqui's AIX* Blog, which has a base set of performance tunables for AIX 5.3 - www.circle4.com/blosxomjl.cgi/
6. vmo command - publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.doc/cmds/aixcmds6/vmo.htm
7. ioo command - publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.doc/cmds/aixcmds3/ioo.htm
8. vmstat command - publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.doc/cmds/aixcmds3/ioo.htm
9. lvmo command - publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.doc/cmds/aixcmds3/ioo.htm

Reference

Starter set of tuneables
Current recommended AIX* 5.3 Performance Parameters: Please ensure you test these first before implementing in production as your mileage may vary

Network
no -p -o rfc1323=1
no -p -o sb_max=1310720
no -p -o tcp_sendspace=262144
no -p -o tcp_recvspace=262144
no -p -o udp_sendspace=65536
no -p -o udp_recvspace=655360
nfso -p -o rfc_1323=1
nfso -p -o nfs_socketsize=60000
nfso -p -o nfs_tcp_socketsize=600000

Memory Settings
vmo -p -o minperm%=5
Leave maxperm and maxclient at default of 80
Leave strict_maxperm and strict_maxclient at their defaults

vmo -p -o minfree=960
vmo -p -o maxfree=1088
vmo -p -o lru_file_repage=0
vmo -p -o lru_poll_interval=10

IO Settings
Leave mipgahead and J2_minPageReadAhead at their defaults of 2
ioo -p -o j2_maxPageReadAhead=128
ioo -p -o maxpgahead=16
ioo -p -o j2_maxRandomWrite=32
ioo -p -o maxrandwrt=32
ioo -p -o j2_nBufferPerPagerDevice=1024
ioo -p -o pv_min_pbuf=1024
ioo -p -o numfsbufs=2048

If doing lots of raw I/O you may want to change lvm_bufcnt
Default is 9
ioo -p -o lvm_bufcnt=12

Others left to default that you may want to tweak include:
ioo -p -o numclust=1
ioo -p -o j2_nRandomCluster=0
ioo -p -o j2_nPagesPerWriteBehindCluster=32

These are starting points only.

Jaqui Lynch, *a senior systems engineer focusing on System p* and Linux* platforms at Mainline Information Systems, has worked in the IS industry for more than 28 years. She's been responsible for a wide variety of projects and OSs across multiple vendor platforms, including mainframes, UNIX* systems, midrange systems and personal workstations. Jaqui can be reached at jaqui.lynch@mainline.com.*