Issue Date: AIX EXTRA eNewsletter
May 2004, Posted On: 5/1/2004

AIX EXTRA: Tuning AIX With ioo and vmo

Jaqui Lynch

The vmtune command that systems administrators have historically used to tune I/O and memory usage on AIX systems is being replaced by the vmo and ioo commands, which provide many of the same facilities. ioo handles all the I/O-related parameters, while vmo handles the virtual memory management ones.

When using these commands to tune systems, it’s imperative to test all of your chosen options in advance on a test server, as not everyone will realize the same results from these commands and the various options chosen. Recommendations I make in this article have been used on multiple systems, but I can’t guarantee them for every environment.

The ioo parameters to tune include:

Sequential

- minpgahead and maxpgahead
- j2_minpgahead and j2_maxpgahead
- numclust

Random

- maxrandwrt
- j2_maxRandomWrite
- j2_nRandomCluster
- sync_release_ilock

The vmo parameters are:

- minfree and maxfree
- minperm, maxperm and maxclient

Sequential Read-Ahead
minpgahead and maxpgahead determine the number of pages to be read ahead when the system determines it’s reading sequential data. The primary reason for tuning these is to enhance performance for programs that access large files sequentially. Basically, the first access to the file causes the first page to be read in. If the program goes directly to the next page without accessing any other pages in the file, the Virtual Memory Manager (VMM) determines that the application is using sequential access and reads in the requested page plus the next set of (minpgahead) pages. So, if minpgahead is set to two, it reads in the next two pages as well as the requested one. As the application continues reading sequentially, the VMM doubles minpgahead each time to determine how many pages to read in, with maxpgahead determining the cap (default is eight). Sequential read-ahead terminates when the application ends or accesses a page out of order.

You can tune these values separately for normal file systems (jfs) and enhanced file systems (jfs2) by setting the values for minpgahead/maxpgahead for jfs and j2_minpgahead and j2_maxpgahead for jfs2 filesystems.

While maxpgahead goes up to 4096 (in powers of two), IBM recommends that you don’t exceed 512, due to kernel limitations. A good starting point is minpgahead=2 and maxpgahead=16 for a 64 KB stripe size. If you do increase maxpgahead, you’ll also need to increase the vmo maxfree parameter, which should always be >= to minfree plus maxpgahead.

VMM Write-Behind and the Syncd Daemon
When pages are updated in memory they aren’t 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 are written to disk, blocking anyone trying to access that file while the lock is held. On a busy system with a high I/O workload, this can cause I/O wait and affect performance. You can deal with this by:

- Changing syncd to run more often.
- Setting sync_release_ilock to one, which causes sync to flush all I/O to a file without holding the i-node lock, then use the i-node lock when it does the commit.
- Turning on random and/or sequential write-behind.

Generally, I turn on sync_release_ilock and customize random write-behind. I rarely modify the syncd interval.

numclust controls sequential write-behind. Files are partitioned into 16 KB-partitions or four pages (clusters) by default. If all four pages in a cluster are dirty, the VMM schedules them to go to disk when the first page in the next cluster is modified. You can increase the defaults of one cluster for jfs and eight clusters for enhanced jfs to delay the writes.

Random write-behind can improve performance on a system with much random I/O. If many pages have been modified, you'll see large bursts of I/O when the syncd daemon runs. This affects the consistency of performance. maxrandwrt can provide a threshold where dirty pages will be written out to disk. I usually start with 32 (the default is 0 or never), which 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. J2_RandomCluster is also available to specify how many clusters apart two consecutive writes need to be in order to be considered random writes.

One last item to tune is the combination of minperm, maxperm and maxclient. Like most UNIX systems, AIX leaves in memory the pages of files that have been read or written. If they're referenced again, it saves an I/O operation. These pages may be from local or remote (Network File System (NFS)) file systems. Since reads usually involve just a pagein, whereas computational or working set pages tend to involve both a pagein and a pageout, it's useful to tweak the minperm and maxperm values to favor computational pages.

maxperm specifies the value for file pages (%) above which the page stealer should steal only file pages. The default is 80 percent, but if you set maxperm to 30 percent and the pages are available, I/O can still use up to 100 percent of memory. What you're changing is which pages get stolen when a page is needed. minperm specifies the value for file pages (%) below which the page stealer steals both file and computational pages. When the percentage of file pages falls between minperm and maxperm, the page stealer steals file pages unless the number of file repages is higher than the number of computational repages. maxclient allows you to further control how much memory is used for caching client (NFS) pages. It can't be set to a higher value than maxperm.

An additional parameter, called strict_maxperm, when set to one, forces the maxperm percentage as a hard limit. This should only be used to meet a very specific need.

I normally start with these settings:

- vmo –p –o minperm%=10
- vmo –p –o maxperm%=50 I normally end up down at 30
- vmo –p –o maxclient%=50 Must be <= maxperm%
- ioo –p –o maxrandwrt=32
- ioo -p -o sync_release_ilock=1

I've seen these make a substantial difference in response time on transactional systems. For a sequential workload system, you could also change minpgahead and maxpgahead.