close window

IBM Systems MAGAZINE

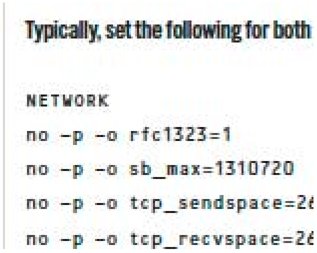
Cover Story

Print

In Tune With Oracle

An AIX administrator’s perspective on getting started

October | November 2008 | by Jaqui Lynch



Typically, set the following for both

```
NETWORK
no -p -o rfc1323=1
no -p -o sb_max=1310720
no -p -o tcp_sendspace=2
no -p -o tcp_recvspace=2
```

Figure 1

Figure 2

Figure 3

Previously, I’ve covered AIX* performance tuning in general, primarily for AIX 5.3. In this article, I’ll share AIX tuning information as it relates to Oracle, specifically looking at tunables in AIX 5.3 and 6.1 that can assist in an Oracle environment. I’ll also look at some of the things you may want to review in the Oracle-provided AWR report to assist with tuning. Let’s assume that our example system is using enhanced journaled file systems (JFS2) or raw logical volumes (LVs), and that it’s at AIX 5.3 or 6.1.

To start, let’s quickly review our starter set of tunables for AIX 5.3 and 6.1 (see Figure 1). Some of these are flagged as needing determination based on the output from lvmo –a and vmstat –v. Figure 2 is an example of the output of the vmstat –v command, which shows values since boot. Thus, it’s important to take two snapshots to see if these numbers are changing. In this case, they were growing rapidly and require changes. The “pending disk I/Os blocked with no pbuf” line clearly indicates that one or more I/Os are being blocked trying to get pinned memory buffers. This indicates queuing at the Logical Volume Manager (LVM) layer and must be corrected because AIX can’t get a buffer to store information about a pending I/O request, causing the request to be delayed. Figure 3 shows the output from the lvmo –a command, which indicates that datavg has insufficient pbufs (look at the pervg_blocked_io_count). This can be corrected globally by using ioo to set pv_min_pbuf to a larger number, such as 2048, or it can be corrected just for this one volume group using: lvmo –v datavg –o pv_pbuf_count=2048. The preferred method is to use lvmo on the individual volume group: lvmo –v datavg –o pv_pbuf_count=2048.

This system also appears to be having problems with page space buffers (psbufs line), JFS buffers (fsbufs line) and JFS2 buffers (external pager line). JFS buffers are corrected by increasing numfsbufs with ioo. JFS2 buffers are corrected by increasing j2_dynamicBufferPreallocation using ioo. The values in the client line are NFS or Vxfs. NFS buffers can be increased by using nfso to increase nfs_v3_pdts or nfs_v3_vm_bufs. If you're using v4 of NFS, use v4 instead of v3. Lastly, if page space buffers are the problem, stop the paging or add more page spaces. Best practice is two to four page spaces, all the same size and all on different disks. I usually have a 4 GB page space as hd6 and then two to three other page spaces in a paging volume group that no other file systems are allocated on. Under no circumstances should you put two page spaces on the same disk. A good first step is to ensure you're using the starting point tunables in Figure 1. I've seen cases (like the one from our example) where just setting those tunables made the problems disappear, so always start there.

Minfree and maxfree are other tunables under vmo that can affect paging. They're now allocated on a memory-pool basis, and you can tell how many memory pools you have by issuing the vmo –a (for 6.1, vmo –a –F) command. If you overallocate these values, you could see high values in the "fre" column of a vmstat and yet you'll be paging. The defaults are 960 and 1088; I typically use 1000 and 1200 depending on other settings. These values are actually calculations and can be determined as follows:

```
minfree = (max (960,(120 * lcpus) / memory pools))
maxfree = minfree + (Max(maxpgahead,j2_maxPageReadahead)
* lcpus) / memory pools
```

So, if you have the following:

```
Memory pools = 3 (from vmo -a)
J2_maxPageReadahead = 128
CPUS = 6 SMT on, so lcpu = 12
So minfree = (max(960,(120 * 12)/3)) = 1440 / 3 = 480 (or 960
since that's the max)
And maxfree = (128 * 12) / 3 = 512 (+960) = 1472
```

I'd leave minfree at 960 but would increase maxfree to 1472.

Oracle Disk

Many of the most common Oracle performance problems are related to either I/O or locking. In particular, data layout can affect performance more than any I/O tunable the administrator can set. Since changing these later is extremely painful, planning is important to avoid these problems. One solution is to use Oracle Automatic Storage Management (ASM), which will automatically manage the disk space for you. That's beyond the scope of this article, so I'll focus on JFS2 implementations with a couple of comments on using raw LVs.

The trend in the industry right now is to provide fewer larger hdisks to the server. For example, the server may be given one 500 GB hdisk that's spread across several disks in the disk subsystem, rather than 10 x 50 GB or 5 x 100 GB hdisks. However, I/O performance depends on bandwidth, not size. While that data may be spread across multiple disks in the back end, this doesn’t help with queuing in the front end. At the server, the hdisk driver has an in-process and a wait queue. The in-process queue for the hdisk can contain up to queue_depth I/Os and the hdisk driver submits the I/Os to the adapter driver. Why is this important? If your data’s striped by LVM across five hdisks, you can have more I/Os in process simultaneously. With one big hdisk, you’ll be queuing. By default, I now tune queue_depth on the hdisks. Multipath I/O drivers such as subsystem device driver (SDD) won’t submit more than queue_depth I/Os to an hdisk, which can affect performance. So you either need to increase queue_depth or disable that limit. In SDD, use the “datapath qdepth disable” command.

Some vendors do a nice job of setting the queue_depth, but if you’re using large logical unit numbers from multiple disks in the back end, you’ll want to grow this. You can use the iostat –D or the sar –d commands to figure this out. In particular, look at the avgsqsz, avgwqsz and sqfull fields to determine if you should increase queue_depth. Don’t increase queue_depth beyond the disk manufacturer’s recommendations. lsattr –El hdisk? shows the current queue_depth setting.

For Fibre Channel, the adapter also has an in-process queue, which can hold up to num_cmd_elems of I/Os. The adapter submits the I/Os to the disk subsystem and it uses direct memory access (DMA) to perform the I/O. You may need to consider changing two settings on the adapter. By default, num_cmd_elems is set to 200 and max_xfer_size is set to 0x100000. The latter equates to a DMA size of 16 MB. For a heavy I/O load, I increase the DMA size to 0x200000, which is 128 MB, and I’ve set num_cmd_elems as high as 2048. Do this before the hdisks, etc., are assigned or you’ll have to rmdev them all to set these values. lsattr –El fcs? shows the current settings. Before changing these, check with your disk vendor.

Volume Groups and File Systems

Once the Fibre cards and hdisks are correctly set, you should look at the setup for volume groups, LVs and file systems. I try to have fewer volume groups with several disks in them as it gives me the flexibility to easily move file systems using mirroring technology. This lets me move a file system from one hot disk to another without an outage. I’ve seen setups where there’s one volume group per file system and it’s not conducive to flexibility in solving performance problems.

Oracle recommends striping LVs or physical partitions (PPs) across the disks. Striping PPs provides better flexibility if you plan to add disks to the stripe set later. Stripe LVs across as many disks in the volume group as makes sense. Choose a reasonable stripe size. Make sure each instance is broken out so its redo and control files are in their own JFS2 file system. Redo logs and control files should be in their own file system with an agblksize set to 512. The I/O size is always a multiple of 512 and you’ll do unnecessary I/O if you leave it at the default of 4096. The other file systems can be left at the default with the exception of the database (DBF) files. The agblksize for these should be calculated using: db_block_size * db_file_multiblock_read_count. If the block size is more than 4096, Oracle recommends using an agblksize of 4096 for database files; otherwise it recommends using 1024 or 2048. To find the current block size for a file system, use the “lsfs –q” command.

Asynchronous I/O and Concurrent I/O

Asynchronous I/O (AIO) is set at the OS level and Oracle needs to know it's configured. AIO is used to improve performance for I/O to raw LVs as well as file systems. It's set up differently on AIX 5.3 and AIX 6.1. In 5.3, use chdev to set three parameters—minservers, maxservers and maxreqs. In 6.1, use ioo. If you do an "ioo –a | grep aio" on a 6.1 system, you'll see the new parameters. Gather statistics using the "iostat –A" command. From these, you can tell how many AIOs are in use. Additionally, if you see maxg getting close to maxreqs, increase maxreqs.

In 6.1, AIO changes and the subsystems are now loaded by default. The aio0 device goes away as does the new aioo command. Additionally, AIO is enabled by default and the only value that needs tuning normally is the aio_maxservers or posix_aio_maxservers option in ioo.

Concurrent I/O (CIO) is a feature of AIX with JFS2 that bypasses the buffer caching and reduces double buffering, where an I/O comes into memory, is stored there and then copied into the application buffer. CIO also removes inode locking for the file system during write operations, so it should only be used where the application takes care of data serialization. For Oracle, this means CIO should be used for database DBF files, redo logs and control files, and flashback log files. It shouldn't be used for Oracle binaries or archive log files. Use of CIO, for a mixed or random-access workload, can make a significant difference in memory use (reducing paging), CPU utilization (no more copying memory pages between the two memory locations) and performance in general. However, since it bypasses readahead, sequential operations may not perform as well.

In more recent versions of Oracle (10g and 11g), you no longer set CIO on the file system itself. (This was done using the "cio" option.) You now set two parameters in the init.ora—"filesystemio_options = setall" and "disk_async_io = true." This turns on both CIO and AIO. Oracle will correctly determine when to use CIO, but you should still follow the aforementioned recommendations on how to split up the file systems. It's also critical that you don't put any files in the file systems that will use CIO unless Oracle knows how to manage the write serialization for them.

Getting Oracle Information

To determine what's happening in Oracle, it may be necessary to review a Statspack or an AWR (10g or 11g) report. Statspack is for older versions of Oracle, so I'll briefly explain the AWR here. If the performance problem is reproducible, ask the DBA to take a snap, then reproduce the problem and take another snap. Your DBA should then be able to get you an AWR comparing the data from the two snaps. The first place to look is at the "Top 5 Timed Events." If you see "log file" or "latch : redo" in those events, you have issues with your redo logs or checkpoints. Top waits of reads and writes with "buffer busy waits," "write complete waits," "DB file parallel writes" and "enqueue waits" indicate I/O issues. A well-performing system should show the top events as CPU time and reads. A section in the AWR called "Advisory Statistics" can be very useful in determining optimum buffer-pool sizes. Be familiar with these reports; they're good indicators to the administrator on where to look to resolve problems.

Get Started

This is a very quick review of some things you should examine when setting up Oracle systems. It's not an all-encompassing list but it's a good starting point for most administrators.

Resources

- Tuning a Perfect Note
  www.ibmsystemsmag.com/opensystems/augustseptember06/coverstory/6269p1.aspx
- Oracle Performance Tuning Guide B14211-03 March 2008
  http://download.oracle.com/docs/cd/B19306_01/server.102/b14211.pdf