Paper SER15_05

Tuning SAS® 9 for Multiuser ETL Environments

Frank Bartucca, IBM® Corporation, Austin, TX

ABSTRACT

Whether it is economic conditions forcing a rediscovery of economy of scale or the pressure on IT shops under tight budgets to provide servers that support increasing levels of data heavy lifting, a clear trend has emerged towards server and workload consolidation onto larger servers. As a consequence large volumes of data are being processed by SAS programs these days in large multiuser ETL environments. We have found that performance in these environments greatly depends on the tuning and optimization of server I/O subsystems and application I/O options. We describe our methodology and process as we tune a specific SAS workload to be less disruptive of the overall performance of a multiuser system.

INTRODUCTION

The User Interface Design & Performance Analysis Department in the Analytical Solutions Division at SAS requested a server for the purpose of setting up a large data mart for performance testing of ETL flows. The SAS R&D Data Center provided an IBM @server pSeries® p650® server running AIX® with a host name of lemans. The IBM pSeries technical consultant, who was onsite at SAS, assisted with the set up and tuning of lemans and the results obtained are documented in this paper. We present practical information that SAS users and system administrators can use to tune their own pSeries servers.

STRUCTURE OF THIS PAPER

First we list the major features of the server and describe the initial configuration and setup. Then we determine the upper limit of the I/O subsystem's performance. We then select and run a SAS workload that is representative of SAS jobs run on lemans. For the first run we measure the I/O performance using default SAS and AIX options to establish baseline performance. The rest of the paper describes the iterative tuning of SAS and AIX options to yield a SAS job that we think would be less disruptive of the overall performance of a multiuser system.

CONCEPTS INCLUDED IN THIS PAPER

- Cached and Direct I/O (DIO)
- I/O buffering
- SAS I/O related options
- AIX tuning parameters
- File system buffer cache
- Logical volumes
- Performance monitoring
- SAN and RAID

SERVER SETUP

HARDWARE

- 4 1.2GHz Dual Core POWER4+® Chips
  - 2 POWER4+ CPUs per POWER4+ Chip
  - Total of 8 CPUs in the system
- 64KB L1 I-cache per CPU
- 32KB L1 D-cache per CPU
- 1.5MB L2 cache per POWER4+ Chip
- 32MB L3 cache
- 16GB main memory
- 2 imbedded Wide/Ultra-3 SCSI I/O controllers
  - Two 146.8 GB disks
  - One 4.7 GB SCSI DVD-ROM drive
- 2 PCI-X 2 Gb/s Fibre Channel adapters
  - The two Fibre Channel adapters are connected to a 16-way switch that is connected to a SAN.
  - Two TB of disk storage are allocated on the SAN for lemans.

SOFTWARE

- AIX 5.3
- SAS 9.1.3

MICROCODE

The first step in setting up the server was to run a microcode survey to make sure the system and device firmware were up to date. All of the information needed to run the microcode survey was found at: https://techsupport.services.ibm.com/server/aix.invscoutMDS

The inventory scout program was run as `root` to collect device microcode levels:

```
# invscout
```

The link Upload a data file on the above web page was used to uploaded to IBM the file `/var/adm/invscout/lemans.mup` produced by the `invscout` command. Within a few seconds a report was sent back with links to the microcode files and readme files for installing the microcode updates on lemans. We found that most of the I/O devices on the system required firmware updates.

AIX

To use the latest version of AIX we did a migration install using the AIX 5.3 install CDs to migrate lemans from AIX 5.2 to AIX 5.3. This migration went smoothly. We then applied the latest AIX 5.3 fixes.

SAS

SAS 9.1.3 was already installed on lemans and the installation remained intact through the AIX 5.2 to AIX 5.3 migration. We also installed a specific fix from Service Pack 3 for SAS 9.1.3 that improves AIX memory allocation efficiency for memory allocation requests from SAS 9.1.3.

SAN AND FILE SYSTEM CONFIGURATION

The SAN resources, as supplied by the SAS R&D Data Center, provided three virtual path devices: `vpath0`, `vpath1` and `vpath2` as seen by AIX. All physical disks in the SAN were 10K rpm 146GB Fibre Channel disks. Multiple physical disks in the SAN were allocated in RAID5 configurations and accessed through multiple paths to implement the `vpath0` and `vpath1` devices. Multiple physical disks in the SAN were also allocated in RAID0 configurations and accessed through multiple paths to implement the `vpath2` device. We ran the `datapath query device` command to show the SAN configuration listed below:

```
# datapath query device

DEV#:   0  DEVICE NAME: vpath0  TYPE: 2145            POLICY:    Optimized
SERIAL: 60050768018080980800000000000002
==========================================================================
Path#      Adapter/Hard Disk          State     Mode      Select     Errors
    0          fscsi0/hdisk4           OPEN   NORMAL    1494403          0
    1          fscsi0/hdisk6           OPEN   NORMAL          0          0
    2          fscsi1/hdisk8           OPEN   NORMAL    1494404          0
    3         fscsi1/hdisk10           OPEN   NORMAL          0          0

DEV#:   1  DEVICE NAME: vpath1  TYPE: 2145            POLICY:    Optimized
SERIAL: 60050768018080980800000000000003
==========================================================================
Path#      Adapter/Hard Disk          State     Mode      Select     Errors
    0          fscsi0/hdisk5           OPEN   NORMAL        201          0
    1          fscsi0/hdisk7           OPEN   NORMAL          0          0
    2          fscsi1/hdisk9           OPEN   NORMAL        189          0
    3         fscsi1/hdisk11           OPEN   NORMAL          0          0

DEV#:   2  DEVICE NAME: vpath2  TYPE: 2145            POLICY:    Optimized
SERIAL: 60050768018080980800000000000008
==========================================================================
Path#      Adapter/Hard Disk          State     Mode      Select     Errors
    0         fscsi0/hdisk12           OPEN   NORMAL          0          0
    1         fscsi0/hdisk13           OPEN   NORMAL    5524671          0
    2         fscsi1/hdisk14           OPEN   NORMAL          0          0
    3         fscsi1/hdisk15           OPEN   NORMAL    5779119          0
```

Each hdisk in a vpath represents a pathway to the storage allocated in the SAN for the vpath device. By accessing the SAN through vpath devices the Data Path Optimizer Pseudo Device Driver can load balance across multiple paths.

AIX can use vpath devices in the same way it uses native hdisk devices to hold logical volumes for file systems. On lemans `hdisk13` and `hdisk15` implement the `/dev/raid0lv01` logical volume and `hdisk4` and `hdisk8` implement the `/dev/raid5lv01` logical volume. Both logical volumes have 998.5 GB `jfs2` file systems. The `/dev/raid0lv01` logical volume is mounted over `/raid0` and `/dev/raid5lv01` is mounted over `/raid5`.

File System Mount Point	Logical Volume	hdisk	Adapter
/raid0	/dev/raid0lv01	hdisk13	fscsi0
		hdisk15	fscsi1
/raid5	/dev/raid5lv01	hdisk4	fscsi0
		hdisk8	fscsi1

DATA MART SETUP

We generated large raw data files and file sets on lemans to simulate the ETL environment that a large retail store chain might implement. These raw data files and file sets were stored in the `/raid5` file system. The `/raid0` file system was used for WORK, UTILITY and other temporary files.

INITIAL I/O PERFORMANCE ASSESSMENT

We determined the upper limit of read and write file system performance by using `namefs` Direct I/O (DIO) mounts of the `/raid5` and `/raid0` file systems to read and write files using the `dd` command. The pseudo devices `/dev/null` and `/dev/zero` were used as data sync and data source respectively. The `filemon` program was used to measure performance. See http://publib.boulder.ibm.com/infocenter/pseries/topic/com.ibm.aix.doc/cmds/aixcmds2/filemon.htm for more information on `filemon`.

DIO MOUNTS

Direct I/O mounts were created using the following commands.

```
# mkdir /raid0dio /raid5dio
# mount -o dio /raid0 /raid0dio
# mount -o dio /raid5 /raid5dio
```

FILEMON

To run the following `filemon` command as a non `root` user the user ID must be a member of the `system` group.

```
$ filemon -dPo filemon.raid0_w -O lf,lv,pv -T 1048576
```

This command line allocated a 1MB trace buffer and pinned it in memory. The output was written to `filemon.sort` and statistics were recorded for the logical file system, logical volume and physical volume layers. The `-d` flag told `filemon` to wait until the `trcon` command was run to start tracing. The `filemon` command stopped tracing and wrote its report when we ran the `trcstop` command.

DD

We used the `dd` command to drive the data transfer, the `/dev/zero` pseudo device as a zero-latency infinite source of bytes with the value 0 and the `/dev/null` pseudo device as a zero-latency infinite bit bucket. We used a block size of `1024k` to make sure that the physical volume (`pv`) layer would use its maximum transfer size so that we could examine the `filemon` output and determine the maximum transfer size for `hdisk4`, `hdisk8`, `hdisk13` and `hdisk15`. The following command lines set up the `filemon` trace, start the trace, run the `dd` command and then stop the trace.

```
$ filemon -dPo filemon.raid0_w -O lf,lv,pv -T 1048576
$ trcon; dd if=/dev/zero of=/raid0dio/zeros bs=1024k count=1024; trcstop

$ filemon -dPo filemon.raid0_r -O lf,lv,pv -T 1048576
$ trcon; dd of=/dev/null if=/raid0dio/zeros bs=1024k count=1024; trcstop

$ filemon -dPo filemon.raid5_w -O lf,lv,pv -T 1048576
$ trcon; dd if=/dev/zero of=/raid5dio/zeros bs=1024k count=1024; trcstop

$ filemon -dPo filemon.raid5_r -O lf,lv,pv -T 1048576
$ trcon; dd of=/dev/null if=/raid5dio/zeros bs=1024k count=1024; trcstop
```

RESULTS FROM RUNNING DD

We examined the filemon reports and determined the following characteristics:

```
Logical Volume      Write Speed         Read Speed
==============      =============       =============
/dev/raid0lv01      118892.7 KB/s       117877.7 KB/s
/dev/raid5lv01      164033.2 KB/s       144576.4 KB/s

hdisk               MAX Transfer Size
==============      =================
hdisk4              256 KB
hdisk8              256 KB
hdisk13             256 KB
hdisk15             256 KB
```

The results showed that the maximum transfer size was 256 KB. Therefore, for maximum transfer efficiency when moving data between the file system cache and the SAN, we wanted AIX to use 256 KB transfers as often as possible. We tuned the following parameters to increase the occurrence of 256 KB transfers:

```
Name                              Default value   New Value
j2_minPageReadAhead               2               16
j2_nPagesPerWriteBehindCluster    32              128
sync_release_ilock                0               1
```

A description of these parameters can be found at:
http://publib.boulder.ibm.com/infocenter/pseries/topic/com.ibm.aix.doc/cmds/aixcmds3/ioo.htm

INITIAL PROC SORT PERFORMANCE

Although sometimes hidden behind more complex data manipulation processes, the sorting and merging of data sets is a major aspect of the day-to-day operation of large data marts. Large sort jobs, however, can disrupt the performance of multi-user systems when they occupy much of the file system cache. Tuning a system for efficient sorting and merging and tuning large sort jobs to minimize their impact on system resources covers a large part of the performance tuning problem.

SETUP

In this tuning exercise a 17 GB compressed data set from the data mart on lemans was sorted by a major and minor key. The data set contained 175,889,848 store sales records. The size of the data set when uncompressed was 60 GB. With 16 GB of memory installed on lemans PROC SORT required a disk resident SAS utility file that we specified with the option UTILLOC. The utility and work directories were located on /raid0 which was set up for temporary storage. The input data set as well as the destination for the output data set were on the /raid5 file system. We set the SORTSIZE option to `4G` and the MEMSIZE option to `8G`. The `filemon` program was run to collect I/O data during the sort. We also ran the `topas` performance monitor in a separate window. See http://publib.boulder.ibm.com/infocenter/pseries/topic/com.ibm.aix.doc/aixbman/prftungd/sysperfmon3.htm for more information on `topas`.

INITIAL ADJUSTMENTS

On the first run the report from `filemon` showed that the trace had missed a large number of events. We ran the `filemon` command at a higher priority and used a larger trace buffer to eliminate this problem:

```
# nice --10 filemon -dPo filemon.raid5_r -O lf,lv,pv -T 16777216
```

The command now had to be run as `root` to use `nice` to raise its priority.

The `topas` monitor also showed that the demand for a large number of pages by the file system cache was causing part of the SAS process to be paged out. This contention between computational pages (SAS and other programs) and non-computational pages (file system) caused the system to thrash. To protect computational pages from being paged out we changed the default settings of the following AIX performance parameters:

```
Name            Default value   New Value
maxperm         80              60
strict_maxperm  0(off)          1(on)
```

With the `strict_maxperm` parameter turned on the new value of `maxperm` limited the percentage of real memory that was used to store non-computational pages in the file system cache. After making these changes SAS and other

programs running on the system during subsequent sort runs occupied 39.2% of real memory without paging. The `filemon` output subsequently showed no activity on the page space logical volumes as expected.

RESULTS

Highlights from the SAS log file and the `filemon` report are shown.

From the SAS log file:

```
   NOTE: The SAS System used:
         real time           33:35.62
         user cpu time       24:43.67
         system cpu time     13:59.67
         Memory                           4212800k
         Page Faults                      738014
         Page Reclaims                    4419392
         Page Swaps                       0
         Voluntary Context Switches       3949910
         Involuntary Context Switches     41236
```

From the `filemon` report:

```
Most Active Files
   ------------------------------------------------------------------------
   #MBs       #opns   #rds    #wrs     file
   ------------------------------------------------------------------------
   120804.2   2       966434  966434   utB1B2000002.utl
   34702.9    1       103426  2117568  sorted_stg_sales_fb.sas7bdat.lck
   16557.4    1       1059674 0        sorted_stg_sales.sas7bdat
   ------------------------------------------------------------------------
Most Active Logical Volumes
   ------------------------------------------------------------------------
   util  #rblk      #wblk      KB/s     volume            description
   ------------------------------------------------------------------------
   0.58  124062200  123704176  61456.9  /dev/raid0lv01    /raid0
   0.47  33915688   66071968   24801.3  /dev/raid5lv01    /raid5
   ------------------------------------------------------------------------
Most Active Physical Volumes
   ------------------------------------------------------------------------
   util  #rblk    #wblk    KB/s     volume         description
   ------------------------------------------------------------------------
   1.00  17056168 32818472 12371.1  /dev/hdisk4    SAN Volume Controller Device
   1.00  16859520 33253496 12430.2  /dev/hdisk8    SAN Volume Controller Device
   0.99  62029776 62478216 30883.4  /dev/hdisk15   SAN Volume Controller Device
   0.99  62032424 61226592 30573.6  /dev/hdisk13   SAN Volume Controller Device
   ------------------------------------------------------------------------
Detailed Physical Volume Stats   (512 byte blocks)
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk4  description: SAN Volume Controller Device
   reads:  40543
   read sizes  (blks): avg 420.7   min 8   max 512   sdev 132.8
   writes: 744815
   write sizes (blks): avg  44.1   min 8   max 512   sdev  83.7
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk8  description: SAN Volume Controller Device
   reads:  40201
   read sizes  (blks): avg 419.4   min 8   max 512   sdev 133.9
   writes: 767552
   write sizes (blks): avg  43.3   min 8   max 512   sdev  83.4
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk15  description: SAN Volume Controller Device
   reads:  126561
   read sizes  (blks): avg 490.1   min 8   max 512   sdev  97.3
   writes: 122844
   write sizes (blks): avg 508.6   min 8   max 512   sdev  41.1
```

```
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk13  description: SAN Volume Controller Device
   reads:  126470
   read sizes  (blks): avg 490.5   min 8   max 512   sdev  96.4
   writes: 120319
   write sizes (blks): avg 508.9   min 8   max 512   sdev  39.3
```

ANALYSIS

In this job a 17 GB compressed data set was sorted by a major and minor key and the new data set was saved to disk. The input data set `sorted_stg_sales.sas7bdat` was read by SAS using 16 KB reads. The output data set `sorted_stg_sales_fb.sas7bdat` was read and written by SAS using 16 KB reads and writes. We ran PROC CONTENTS against both data sets and the SAS log showed a data set page size of 16 KB. The temporary utility file `utB1B2000002.utl` was read and written by SAS with 64 KB reads and writes.

An advantage that this job gained by going through the file system cache was that larger and fewer I/O request were presented to the disk subsystem. For every five I/O requests presented to the file system layer by SAS there were approximately 2 requests sent to the disk subsystem.

We used information in the `filemon` report to compare the total amount a data moved through the file system level and the total amount of data moved through the disk I/O level. In this case only 1.3% more data were moved through the file system level which meant that this job offered little opportunity for reuse of data in the file system cache. This type of job, when run on busy systems, tends to pollute the file system cache with low reuse data and potentially replaces high reuse data.

The sort job used most of the system memory available on lemans and consumed a little more than one CPU worth of processing power on this eight-way system. The RAID5 logical disk was used at 43% of its I/O capacity and the RAID0 logical disk was used at 21% of its I/O capacity.

PROC SORT PERFORMANCE USING DIO

The next step was to investigate the performance of the sort job without the use of the file system cache. This was done by accessing `/raid0` and `/raid5` through DIO mounts. We reasoned that in a multiuser system using DIO for the sort job would make the file system cache more available to jobs that could make better use of it. We also wanted the performance of the sort job itself to remain acceptable.

RESULTS

Highlights from the SAS log file and the `filemon` report are shown.

From the SAS log file:

```
   NOTE: The SAS System used:
         real time           52:32.08
         user cpu time       24:41.30
         system cpu time     8:29.05
         Memory                            4212800k
         Page Faults                       154
         Page Reclaims                     4203677
         Page Swaps                        0
         Voluntary Context Switches        9194784
         Involuntary Context Switches      9886
```

From the `filemon` report:

```
Most Active Files
   ------------------------------------------------------------------------
   #MBs       #opns   #rds     #wrs     file
   ------------------------------------------------------------------------
   120804.2   2       966434   966434   utA116000002.utl
   34702.9    1       103426   2117568  sorted_stg_sales_fb.sas7bdat.lck
   16557.4    1       1059674  0        sorted_stg_sales.sas7bdat
   ------------------------------------------------------------------------
Most Active Logical Volumes
   ------------------------------------------------------------------------
   util  #rblk      #wblk      KB/s     volume           description
   ------------------------------------------------------------------------
```

```
   0.47  123703552  123703552  39242.5  /dev/raid0lv01   /raid0
   0.37  37238368   67846032   16668.0  /dev/raid5lv01   /raid5
   ------------------------------------------------------------------------
Most Active Physical Volumes
   ------------------------------------------------------------------------
   util  #rblk    #wblk     KB/s    volume          description
   ------------------------------------------------------------------------
   0.99  18629632 34148408 8371.4   /dev/hdisk4     SAN Volume Controller Device
   0.99  18608736 33697624 8296.6   /dev/hdisk8     SAN Volume Controller Device
   0.23  61948800 61872896 19640.0  /dev/hdisk13    SAN Volume Controller Device
   0.23  61754752 61830920 19602.6  /dev/hdisk15    SAN Volume Controller Device
   ------------------------------------------------------------------------
Detailed Physical Volume Stats   (512 byte blocks)
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk4  description: SAN Volume Controller Device
   reads:  596445
   read sizes  (blks): avg  31.2 min   8   max  32   sdev 3.5
   writes: 1070721
   write sizes (blks): avg  31.9 min   8   max 120   sdev 1.6
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk8  description: SAN Volume Controller Device
   reads:  595849
   read sizes  (blks): avg  31.2 min   8   max  32   sdev 3.5
   writes: 1056589
   write sizes (blks): avg  31.9 min   8   max 128   sdev 1.6
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk13  description: SAN Volume Controller Device
   reads:  483975
   read sizes  (blks): avg 128.0 min 128   max 128   sdev 0.0
   writes: 483397
   write sizes (blks): avg 128.0 min   8   max 128   sdev 0.7
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk15  description: SAN Volume Controller Device
   reads:  482459
   read sizes  (blks): avg 128.0 min 128   max 128   sdev 0.0
   writes: 483070
   write sizes (blks): avg 128.0 min   8   max 128   sdev 0.7
```

ANALYSIS

Using DIO decreased the use of system CPU time by 53.3% but increased run time by 56.4%. The extra run time came from an increase in I/O wait time. There was no read ahead to overlap with I/O startup latency as there was when the file system cache was used. The average read block size and the average write block size shown in the physical volume report were smaller and resulted in less efficient I/O. There were also more I/O transactions therefore there was more total startup latency.

Comparison of this job with the previous run which used the file system cache showed that the additional 5.5 minutes of system CPU time was spent moving data between SAS I/O buffers and the file system cache. In this respect DIO is more efficient because when DIO is used the I/O adapters use DMA to move data directly to and from SAS I/O buffers without the need for the CPU driven memory-to-memory copy needed when the file system cache is used.

The user CPU time was essentially the same with or without DIO. This is what we expected because the processing measured by user CPU time is essentially unchanged by the external I/O method.

Even though 19 minutes was added to the sort job's run time more total work could now be done on the system. Sixty percent of memory was freed up and there were seven other CPUs ready to do work.

PROC SORT PERFORMANCE USING DIO; DATA SET PAGESIZE=256K

When Direct I/O is used, SAS is in direct control of the transfer size of the low level I/O. The previous analysis pointed out some of the benefits of Direct I/O; therefore getting SAS to perform more efficient Direct I/O transfers was worth pursuing.

We changed the data set page size of the input data set from `16K` to `256K` using the following program:

```
options fullstimer compress=yes;
libname in '/raid5dio/disk3/compress_data';
libname out '/raid5dio/frbart';
data out.sorted_stg_sales_256k (BUFSIZE=256K);
     set in.sorted_stg_sales;
run;
```

The sort job was then rerun with the new input data set and `-bufsize 256K` set on the SAS command line. The option `-bufsize 256K` caused new data sets to be created with a 256 KB page size.

RESULTS

Highlights from the SAS log file and the `filemon` report are shown.

From the SAS log file:

```
NOTE: The SAS System used:
      real time           38:59.46
      user cpu time       24:12.47
      system cpu time     6:31.93
      Memory                              4214650k
      Page Faults                         65
      Page Reclaims                       2829005
      Page Swaps                          0
      Voluntary Context Switches          6089601
      Involuntary Context Switches        16173
```

From the `filemon` report:

```
Most Active Files
   ------------------------------------------------------------------------
   #MBs       #opns   #rds    #wrs    file
   ------------------------------------------------------------------------
   120804.2   2       966434  966434  ut405A000002.utl
   32571.8    1       1       130294  sorted_stg_sales_256k_fb.sas7bdat.lck
   16442.0    1       65770   0       sorted_stg_sales_256k.sas7bdat
   ------------------------------------------------------------------------
Most Active Logical Volumes
   ------------------------------------------------------------------------
   util   #rblk      #wblk      KB/s     volume          description
   ------------------------------------------------------------------------
   0.65   123703552  123703552  52872.9  /dev/raid0lv01  /raid0
   0.16   33674088   66706496   21452.1  /dev/raid5lv01  /raid5
   ------------------------------------------------------------------------
Most Active Physical Volumes
   ------------------------------------------------------------------------
     util  #rblk    #wblk    KB/s     volume        description
   ------------------------------------------------------------------------
     1.00  16733376 33425960 10719.5  /dev/hdisk8   SAN Volume Controller Device
     1.00  16940712 33280536 10732.7  /dev/hdisk4   SAN Volume Controller Device
     0.32  61764224 61867464 26421.1  /dev/hdisk15  SAN Volume Controller Device
     0.32  61939328 61836232 26451.8  /dev/hdisk13  SAN Volume Controller Device
   ------------------------------------------------------------------------
Detailed Physical Volume Stats   (512 byte blocks)
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk8  description: SAN Volume Controller Device
   reads:  32715
   read sizes  (blks): avg 511.5   min   8   max 512   sdev 15.4
   writes: 65290
   write sizes (blks): avg 512.0   min   8   max 512   sdev  4.4
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk4  description: SAN Volume Controller Device
   reads:  33120
   read sizes  (blks): avg 511.5   min   8   max 512   sdev 15.0
```

```
   writes: 65004
   write sizes (blks): avg 512.0   min   8   max 512   sdev  3.4
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk15  description: SAN Volume Controller Device
   reads:  482533
   read sizes  (blks): avg 128.0   min 128   max 128   sdev  0.0
   writes: 483348
   write sizes (blks): avg 128.0   min   8   max 128   sdev  0.5
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk13  description: SAN Volume Controller Device
   reads:  483901
   read sizes  (blks): avg 128.0   min 128   max 128   sdev  0.0
   writes: 483104
   write sizes (blks): avg 128.0   min   8   max 128   sdev  0.5
```

ANALYSIS

When compared to the previous run, the use of a `256K` page size for the input and output data sets instead of the original `16K` page size reduced run time by 25.8% and system CPU time by 23.0%. This improvement in run time brought it close to the run time achieved when using the file system cache. An additional 5.5 minutes of run time for this sort job was a good tradeoff when 9.6 GB of main memory was freed up on a system with 16 GB of main memory. These results suggested further work to investigate changing the page size of the utility file.

PROC SORT PERFORMANCE USING DIO; DATA SET AND UTILITY FILE PAGESIZE=256K

We couldn't recall if there was a way to specify the page size of the utility file that sort uses. We tried `BUFSIZE=256K` as an option to PROC SORT. This generated a "`Syntax Error`" message in the log file. However, `PAGESIZE` was one of the options listed in the error message after the words "`expecting one of the following`". We tried `PAGESIZE=256K` and it worked.

RESULTS

Highlights from the SAS log file and the `filemon` report are shown.

From the SAS log file:

```
   NOTE: The SAS System used:
         real time           33:59.30
         user cpu time       24:01.52
         system cpu time     5:22.80
         Memory                              4214650k
         Page Faults                         101
         Page Reclaims                       4208930
         Page Swaps                          0
         Voluntary Context Switches          1735701
         Involuntary Context Switches        6816
```

From the `filemon` report:

```
Most Active Files
   --------------------------------------------------------------------
   #MBs       #opns   #rds    #wrs     file
   --------------------------------------------------------------------
   120804.5   2       241609  241609   utB14A000002.utl
   32571.8    1       1       130294   sorted_stg_sales_256k_fb.sas7bdat.lck
   16442.0    1       65770   0        sorted_stg_sales_256k.sas7bdat
   --------------------------------------------------------------------
Most Active Logical Volumes
   --------------------------------------------------------------------
   util  #rblk      #wblk      KB/s     volume           description
   --------------------------------------------------------------------
   0.50  123703808  123703808  60654.4  /dev/raid0lv01   /raid0
   0.20  33673816   66706496   24609.2  /dev/raid5lv01   /raid5
   --------------------------------------------------------------------
Most Active Physical Volumes
   --------------------------------------------------------------------
   util  #rblk     #wblk     KB/s     volume        description
```

```
   ------------------------------------------------------------------------
   1.00  16836632  33490464  12338.2  /dev/hdisk4   SAN Volume Controller Device
   1.00  16837184  33216032  12271.0  /dev/hdisk8   SAN Volume Controller Device
   0.25  61881856  61801624  30322.2  /dev/hdisk15  SAN Volume Controller Device
   0.25  61821952  61902472  30332.3  /dev/hdisk13  SAN Volume Controller Device
   ------------------------------------------------------------------------
Detailed Physical Volume Stats   (512 byte blocks)
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk4  description: SAN Volume Controller Device
   reads:  32899
   read sizes  (blks): avg 511.8   min   8   max 512   sdev  9.2
   writes: 65415
   write sizes (blks): avg 512.0   min   8   max 512   sdev  3.9
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk8  description: SAN Volume Controller Device
   reads:  32902
   read sizes  (blks): avg 511.7   min   8   max 512   sdev 10.7
   writes: 64879
   write sizes (blks): avg 512.0   min   8   max 512   sdev  4.0
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk15  description: SAN Volume Controller Device
   reads:  120863
   read sizes  (blks): avg 512.0   min 512   max 512   sdev  0.0
   writes: 120725
   write sizes (blks): avg 511.9   min   8   max 512   sdev  6.3
   ------------------------------------------------------------------------
   VOLUME: /dev/hdisk13  description: SAN Volume Controller Device
   reads:  120746
   read sizes  (blks): avg 512.0   min 512   max 512   sdev  0.0
   writes: 120920
   write sizes (blks): avg 511.9   min   8   max 512   sdev  6.0
```

ANALYSIS

The use of DIO and 256 KB page sizes for the utility file and the input and output data sets improved run time to the point of being only 1.2% slower than the first run which used the file system cache and 16 KB page size data sets and a 64 KB page size utility file. Also, system CPU time was reduced by 62% compared to the first run. The `Detailed Physical Volume Stats` showed that the size of almost all of the physical volume transfers was 256 KB. When compared to the previous run, the addition of `PAGESIZE=256K` reduced run time by 12.8% and system CPU time by 17.6%.

PROC SORT PERFORMANCE USING DIO, DATA SET AND UTILITY FILE PAGESIZE=256K AND BUFNO=16

One of the advantages of using the file system cache was that the AIX Virtual Memory Manager (VMM) used read ahead and write behind to pipeline I/O to reduce latency. We enabled SAS to pipeline I/O requests by setting the BUFNO option to some number greater than 1. For our last experiment we reran the last job but added `-bufno 16` to the command line.

RESULTS

Highlights from the SAS log file are shown.

From the SAS log file:

```
   NOTE: The SAS System used:
         real time           31:32.07
         user cpu time       23:59.68
         system cpu time     4:45.56
         Memory                              4218234k
         Page Faults                         79
         Page Reclaims                       4212107
         Page Swaps                          0
         Voluntary Context Switches          1675767
         Involuntary Context Switches        3942
```

ANALYSIS

Comparing the memory used in the last run with the current run showed that SAS used 14 X 256 KB more memory. We speculate that at a minimum SAS uses two 256 KB I/O buffers and added 14 more for a total of 16 to meet the request for 16 buffers. When compared to the previous run, the addition of `-bufno 16` reduced run time by 7.2% and system CPU time by 11.5%.

One more run was done with `-bufno 64` and compared to the `-bufno 16` run. The `-bufno 64` run took an additional 16 seconds off the run time, 20 seconds off system CPU time and increased SAS memory use by 48 X 256 KB. These results suggest future work to plot real time and system CPU time against the number of SAS I/O buffers.

Compared to the very first run, which used the file system cache, this last run completed in 6% less time, system CPU time was reduced by 68%, user CPU time was reduced by 3% and file system cache resources were no longer consumed by the sort job.

SUMMARY

Before attempting performance tuning we made sure that the p650 firmware and AIX were at the recommended maintenance levels and we made sure that the SAS installation was functioning properly. We presented basic information about the hardware to provide a framework of understanding of the I/O subsystem. Performance characteristics of the SAN were determined and used for initial performance tuning of AIX. We discovered that with AIX 5.3 most of the default settings of I/O and VMM tuning parameters were already set to reasonable values for enterprise level work. This had not been the case with pre-AIX 5.3 systems.

The `filemon` utility, which is based on the AIX Trace Facility, the `topas` monitor and the output of the `fullstimer` option in the SAS log provided data for our tuning decisions. A large sort job was selected as a representative workload. The AIX tuning parameters `maxperm` and `strict_maxperm` were adjusted to keep the system from paging out the SAS process. Direct I/O was selected as the first tuning step because the size of the input data set was larger than available memory and an analysis of the first job runs showed poor file system cache reuse.

Direct I/O freed up the file system cache but increased run time. The rest of the tuning effort focused on compensating for the loss of some of the benefits of using the file system cache. The data set page size of the input data set was changed from 16 KB to 256 KB and the output data set page size was specified as 256 KB. This brought the run time back down to an acceptable level, further reduced system CPU time and continued to provide the benefit of freeing up the file system cache. The PAGESIZE option of PROC SORT specified the page size that SAS used for the utility file. We used a utility file page size of 256 KB which further reduced run time and system CPU time. Finally, increasing the number of I/O buffers that SAS used reduced run time below that of the first run and provided a further reduction in the use of system CPU time.

DISCUSSION

AIX parameters and SAS options were tuned to minimize the impact of a large sort job on system resources so as to minimize the job's performance impact when run in a multiuser environment. The tuning environment used was a controlled single-user environment not a multiuser environment. This controlled environment was used to facilitate controlled performance experiments. Ultimately the changes suggested in this paper need to be validated in a multiuser environment and as such may not yield the results obtained in the single-user environment.

CONCLUSION

AIX and SAS are instrumented products that provide tools and facilities for producing performance reports. Both products provide tuning parameters that can be changed to improve both the run time and the efficiency of use of system resources. A successful tuning effort requires up-to-date hardware and software, incremental tuning changes, a data driven approach, careful analysis of performance data and an accurate conceptual model of system I/O.

RELATED PUBLICATIONS

AIX 5L Version 5.3 Performance Management
http://publib.boulder.ibm.com/infocenter/pseries/topic/com.ibm.aix.doc/aixbman/prftungd/prftungd.htm

AIX 5L Version 5.3 Performance Tools Guide and Reference
http://publib.boulder.ibm.com/infocenter/pseries/topic/com.ibm.aix.doc/aixbman/prftools/prftools.htm

Performance Toolbox Version 2 and 3 Guide and Reference
http://publib.boulder.ibm.com/infocenter/pseries/topic/com.ibm.aix.doc/perftool/prfusrgd/prfusrgd.htm

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Author: Frank Bartucca
Email: ibmsas@us.ibm.com

SPECIAL NOTICES

The information contained in this document has not been submitted to any formal IBM test and is distributed "AS IS". While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. The use of this information or the implementation of any techniques described herein is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. Customers attempting to adapt these techniques to their own environments do so at their own risk. IBM is not responsible for printing errors in this publication that result in pricing or information inaccuracies. The pronoun "we" as used in this document refers to the authors.

Any performance data contained in this paper were obtained in a controlled environment. Some measurements quoted in this paper may have been made on development-level systems or software. There is no guarantee these measurements will be the same on generally available systems. Some measurements quoted in this paper may have been estimated through extrapolation. Actual results may vary. Users of this paper must verify the applicable data for their specific environment.