Sys Admin the journal for UNIX and Linux systems administrators

SysAdminMag.com

Implementing Micro-Partitioning on the IBM p5 595 Server

Chris Gibson

When management asked why our IBM pSeries hardware, in particular CPU, was so expensive and whether there was anything we could do to reduce the cost to the business, we reviewed the CPU usage on the majority of our LPARs and realized (like most Unix shops) that our systems were only about 20% utilized most of the time. Some of these systems had many dedicated processors doing very little work a great deal of the time. This idle time was very expensive when you consider each processor could cost in the range of $40K-$100K.

Rather than "wasting" CPU resources, we needed to find a way of increasing our CPU utilization and reducing our hardware costs. The answer was immediately available to us in the form of IBM's new micro-partitioning technology on the POWER5 (p5) platform. This technology, along with Virtual I/O, is part of the Advanced POWER Virtualization (APV) feature (standard with p595) on System p. Micro-partitioning allows a system to virtualize its processors for sharing by multiple LPARs. Virtual I/O (VIO) provides the capability for an I/O adapter to be shared by multiple LPARs on the same managed system. We chose not to deploy VIO to any business system as we were still concerned with its performance; thus, all LPARs would continue to use dedicated adapters.

In this article, I will describe how we moved to micro-partitioning and how we monitored and measured our shared processor pool. Topics covered will include: a brief introduction to micro-partitioning, sizing guidelines, p5 LPAR configuration pointers, monitoring/measuring the shared processor pool, and performance considerations. If you are new to Power5 and micro-partitioning, there are some excellent Redbooks available from IBM on these topics. I've provided links to the documents I found most useful. I recommend you review these documents to gain a better understanding of micro-partitioning. Like any new technology, implementing virtualization requires careful preparation and planning first.

To determine whether you and your organization are ready for virtualization, you can review the "Virtualization Company Assessment" article on the IBM Wiki site. The information there is quite useful in determining whether you have a grasp on virtualization concepts and are ready to implement virtualization on your p5 systems. See the references section for more information.

Our environment consisted of the following pSeries hardware: 12 POWER4 p650s, 2 POWER4 p690s and 3 POWER5 p595s. We had more than 120 LPARs, most of which were running AIX 5.2. See Figure 1 .

Micro-Partitioning

The old POWER4 (p4) technology allowed us to allocate whole processors only to an LPAR (i.e., CPUs were dedicated to a specific partition and could not be shared with other LPARs). The micro-partitioning feature on the new p5 systems allowed us to move away from the dedicated processor model and share our processing resources among several LPARs at once. With micro-partitioning, processors are assigned to the shared processor pool. The pool is then utilized by shared processor LPARs, which access the processors in the pool based on their entitled capacity and priority (weight).

Each processor can be shared by up to 10 shared processor LPARs. The minimum capacity that can be assigned is 1/10 of a processor (specified as 0.1 processing units). A micro-partition can be either capped or uncapped. A capped LPAR can access CPUs up to the number of processors specified in its profile. An uncapped partition can obtain more processing power if required from the pool, up to the desired number of virtual processors in the LPARs profile. A weight can be allocated to each uncapped LPAR so that, if several uncapped LPARs are competing for processing power at the same time, the more important systems will be given higher priority for processing resources. The p5 Hypervisor is the virtualization engine behind the APV technology on the p5 system. It is responsible for managing micro-partitioning (and all virtualization) on the p5 platform.

This technology would give us the ability to divide a physical processor's computing power into fractions of a processing unit and share them among multiple LPARs. For example, we could allocate as little as 0.10 processing units as opposed to dedicating an entire CPU. We saw two very big advantages to micro-partitioning. The first was better utilization of physical CPU resources (i.e., our dedicated processor systems were built with spare CPU capacity to cater for peaks in workload and future growth in application demands). This led to wasted CPU resources. With micro-partitions, we could build LPARs with minimal CPU resources but have the capability of dynamically accessing extra CPU power if required. And the second big advantage was more partitions. On a p4 system, once all the CPUs had been allocated to LPARs, you could not build any more partitions. For example on a p650 with 8 CPUs, if you built one LPAR per CPU, you were limited to 8 LPARs per p650. Using micro-partitioning, there can be many more partitions (theoretically up to 254 per 595) than physical CPUs.

Sizing Guidelines

To size our p4 LPARs as p5 micro-partitions, we decided to make the machine (i.e., the p5 Hypervisor) do the hard work for us. We reviewed the average CPU usage for each system to get an idea of the typical utilization. This was done by loading several months of nmon performance data from each system into the nmon2web tool and looking at the resulting long-term reports (see the references section for a link to nmon2web).

We then set the minimum/desired processing units to a value suitable for the anticipated "typical" workload. To cater for peaks in workload, we uncapped the LPAR and allocated an appropriate weight. It was then up to the hypervisor to assign free CPU to whichever LPAR (or LPARs) required it. Our micro-partitions would be uncapped, the Desired Virtual Processors would be higher than required to allow consumption of unused processor, and we would use weights to prioritize unused processor to critical production systems. Also, SMT (Simultaneous Multi-Threading) would be enabled to increase CPU utilization by allowing multiple program threads to use a single virtual processor at once (also known as hyperthreading).

Virtual processors are allocated in whole numbers. Each virtual processor can represent between 0.1 and 1.0 CPUs, known as processing units. The number of virtual processors configured for an LPAR establishes the usable range of processing units. For example, an LPAR with two virtual processors can operate with between 0.1 and 2.0 processing units. The "Desired Processing Units" parameter sets the desired amount of physical CPU that will always be available to an LPAR, known as the entitled capacity. The "Desired Virtual Processor" option sets the preferred number of virtual processors that will be online when the LPAR is activated. It also establishes the upper limit of "uncapped" processor resources that can be utilized by the LPAR. For example, an uncapped LPAR with two (desired) virtual processors can access, at most, two physical processors.

The "Maximum Processing Units" setting defines the number of processing units that can be dynamically assigned to an LPAR. For example, an LPAR with two processing units and a maximum of six will be able to dynamically increase its processing units by allocating an additional four processing units (via the DLPAR operation on the HMC). The "Maximum Virtual Processors" parameter determines the number of virtual processors that could be dynamically assigned to an LPAR. For example, an LPAR with one virtual processor and a maximum of three would be able to dynamically increase the number of virtual processors to three by allocating an additional two virtual processors (this can be done manually with a DLPAR operation via the HMC).

The important point here is that the "Desired Processing Units" and "Desired Virtual Processors" settings are the key to controlling how much uncapped resource a micro-partition can access in the shared pool.

For example, one of our Web servers was previously configured with four dedicated CPUs. The CPU utilization for this system was less than 50%. A great deal of CPU was being under-utilized. So, the sizing for this micro-partition was:

```
Desired processing units = 2.0
Maximum processing units = 6.0
Minimum processing units = 2.0
Desired virtual processors = 4
Minimum virtual processors = 2
Maximum virtual processors = 6
Sharing mode = uncapped
Weight = 100
```

The LPAR is activated with 2.0 desired processing units and 4 desired virtual processors. This means that LPAR will always have access to at least 2.0 processing units but could operate in the range of 2.0 to 4 processing units. If the LPAR requires processing resources beyond its desired processing units of 2.0, the hypervisor will automatically attempt to allocate additional processing capacity to this partition until it reaches the desired number of virtual processors (4). This behavior is depicted in Figure 2 .

Although all 4 virtual processors will appear to be online, AIX will "fold away" any VPs that are not in use and will bring them online when required; this is known as processor folding. Processor folding (introduced in AIX 5.3 TL3) will put idle VPs to sleep and awaken them when the workload demands additional processing power. The processors will still appear when commands such as **mpstat** are run. If the workload can be satisfied by using less than 4 processors, some of the CPUs will appear idle.

The "Minimum Processing Units" setting defines the number of processing units that must be available in the shared pool for this LPAR to start (in this case 2.0). If the minimum is not available, then the LPAR will not start.

Again, it is important to note that the maximum settings don't represent the amount of processing units that an uncapped LPAR can access. The maximum amount of processing power that an uncapped LPAR can access is limited by the "Desired Virtual Processor" setting.

If we determine that the desired number of processing units for this LPAR is too low, we can use DLPAR via the HMC and manually increase the desired processing units up to its maximum (in this case 6). Likewise, if it is determined that the desired number of virtual processors is insufficient, we could use DLPAR via the HMC and manually increase the number of virtual processors up to its maximum of 6. If we use the HMC to change the desired virtual processors to 6, then the upper limit of uncapped processors that the LPAR could access becomes 6.

The minimum and maximum setting, for processing units and virtual processors, represents the ranges between which the desired values can be dynamically changed (usually via the HMC). It is for these reasons that we deliberately set the maximum processing units and maximum virtual processors to a higher number. Thus, we can make adjustments to a partitions configuration without the need for an outage.

Our weighting scheme was designed to ensure that our critical production systems (typically customer facing or external applications performing one of the primary functions of our business, i.e., paying claims and/or selling insurance) were given priority to CPU resources. Weights were assigned based on the criticality of a service, as follows:

Tier	Weight	Description
1	255	Customer supporting critical services
2	200	All other business critical services
3	100	All other production services
4	50	Business development and test services
5	25	Supporting infrastructure services
6	10	Infrastructure development and test services

To view the weight of an LPAR, we could run the **lparstat** command with the **-i** flag and look for the "Variable Capacity Weight":

```
$ lparstat -i | grep Var
Variable Capacity Weight                   : 100
```

LPAR Configuration Guidelines

Configuring a micro-partition on a p5 system is similar to creating an LPAR on a p4 system via the HMC. The biggest difference is in configuring the shared processor options in the LPARs profile. When creating the LPAR, you will be prompted with a choice between Shared and Dedicated Processors. After selecting Shared, you will then need to configure the LPARs processing units, sharing mode, weight, and virtual processors. Starting with the processing units, you will need to enter a minimum, desired, and maximum value.

Clicking on the "Advanced" button will allow you to enter the sharing mode of the LPAR, which is either capped or uncapped. If you choose uncapped, you will need to enter a weight for this LPAR. Entering the minimum, desired, and maximum number of virtual processors is next. We ensured that the "Allow shared processor pool utilization authority" option was ticked, because this allows us to view the available physical processors in the shared pool by running the **lparstat** command on the LPAR. To enable this feature, double-click on the LPAR (not its profile), select Hardware, then "Processors and Memory" and tick the "Allow shared processor pool utilization authority" box.

```
$ lparstat 1 3

System configuration: type=Shared mode=Uncapped smt=On lcpu=2 \
  mem=2048 psize=21 ent=0.75

%user  %sys  %wait  %idle physc %entc  lbusy   app  vcsw phint
-----  ----  -----  ----- ----- ----- ------   ---  ---- -----
  2.6   4.0    0.0   93.4  0.06   8.4    4.0 17.47  1861    28
  0.8   3.0    0.0   96.2  0.04   5.3    2.0 17.46   916    26
  0.8   2.9    0.0   96.4  0.04   5.1    2.0 17.33   925    16
```

The "app" column shows the available physical processors in the shared pool. Based on this output, you can determine (among other things) how much processing power is free in your shared processor pool. Looking at the output, we can see that there are 21 processors in the pool (psize) and that roughly 17 processors are available/free in the pool (app). Also we can see that the LPAR is consuming 5.1% (%entc) of its entitled capacity (ent) or 5.1% of 0.75, which equates to 0.04 of a physical CPU (physc).

Migrating

The first phase of this project targeted 50 LPARs for migration to the p595s. The remaining 70 LPARs were scheduled for 2007. Each of these LPARs required an LPAR definition on the p595. Rather than manually create an LPAR definition for each LPAR via the HMC, I used a handy Perl script called createTgtLPAR to automate the LPAR creation process.

The createTgtLPAR script is part of a package from IBM called the mig2p5 tools. These assist in the migration of older RS/6000 and pSeries systems to the new p5 platform. The tools (in conjunction with NIM) can automate several tasks normally required to upgrade and move an LPAR to p5. For example, the tools can migrate a mksysb of the old system to AIX 5.3, create a new LPAR on the p5 system based on the hardware configuration of the old system, size the LPAR based on its workload, allocate required CPU, memory, IO (even virtual I/O), and then install this mksysb image to the new p5 LPAR. The mig2p5 scripts are listed as follows:

```
/usr/lpp/bos.sysmgt/nim/methods/getSrcUtil
/usr/lpp/bos.sysmgt/nim/methods/getSrcCfg
/usr/lpp/bos.sysmgt/nim/methods/getTgtRsrc
/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg
/usr/lpp/bos.sysmgt/nim/methods/createTgtLPAR
```

The tools have been incorporated into the nim_move_up utility. For more information, see:

```
http://publib.boulder.ibm.com/infocenter/pseries/v5r3/ \
  index.jsp?topic=/com.ibm.aix.cmds/doc/aixcmds4/nim_move_up.htm
```

Once the LPARs were created and their configuration was finished (i.e., all necessary network and Fibre Channel adapters were assigned, cabled, and configured), we were ready to start migrating each LPAR. Each LPAR was migrated via a standard NIM mksysb restore, followed by a SAN re-zone of the data volume groups to the new FC WWPNs and an import of each data volume group on the new LPAR. Migrating 50 LPARs took many months of coordination and negotiation with our business units, but by the end of the migration phase we had 50 micro-partitions up and running across our three p595s. The number of processors and micro-partitions in each shared pool varied on each 595 (e.g., 595-1 had 17 processors in the pool with 11 micro-partitions, 595-2 had 21 processors in the pool with 17 micro-partitions, and 595-3 had 36 processors in the pool with 22 micro-partitions). Now, all we had to do was manage them.

Monitoring and Measuring the Shared Processor Pool

One of our main concerns during the planning stage was how we were going to monitor and measure our shared processor pool. We set ourselves the objective that the amount of unused processing capacity available in the shared processor pool would be monitored to ensure that sufficient processing power was available if required. To cater for peak load, we wanted to ensure that between 10-30% of the shared processor pool would be available during core business hours. The question was how?

Fortunately, I discovered several tools of use in this area. The **topas** command in AIX 5.3 has been enhanced to display cross-partition statistics and recording of this information. The recording functionality was introduced with Technology Level 5 for AIX 5.3. This new topas panel displays metrics for all LPARs running on a p5 system. Dedicated and shared partitions are displayed in separate sections with appropriate metrics. The **-C** option to the **topas** command allowed us to view cross-partition information:

```
$ topas -C
Topas CEC Monitor      Interval:  10       Sat Nov 18 22:44:39 2007
Partitions  Memory (GB)            Processors
Shr: 22     Mon: 122  InUse: 109   Shr: 17  PSz: 36   Shr_PhysB:  8.40
Ded:  0     Avl:   -               Ded:  0  APP: 27.6 Ded_PhysB:  0.00

Host        OS  M Mem InU Lp  Us Sy Wa Id  PhysB  Ent  %EntC Vcsw PhI
----------------------------shared-------------------------------
lpar01   A53 U 8.0 8.0  2  90  5  0  0   1.00  0.50 199.3  430 248
lpar02   A53 U  10 9.9  2  92  1  0  5   1.00  0.75 133.1  402 266
lpar03   A53 U 4.0 4.0  2  93  1  0  5   1.00  0.75 133.0  476 258
lpar04   A53 U 4.0 3.9  2  96  3  0  0   1.00  0.75 132.8  199 488
lpar05   A53 U 8.0 8.0  4  88  7  1  2   1.48  1.25 118.7 1459 415
lpar06   A53 U  12  11  4  63  2  6 28   1.07  1.50  71.2 1842 276
lpar07   A53 U 8.0 8.0  2   5  2  0  0   0.52  0.75  69.7 7028 149
lpar08   A53 U 4.0 3.5  4   4 33  0 62   0.56  1.25  44.9 13301 129
lpar09   A53 U 4.0 3.9  2  23 14 10 51   0.33  0.75  44.2 4877 109
lpar10   A53 C 0.2 0.2  2   5 16  0 78   0.03  0.10  28.7  692   9
lpar11   A53 U 4.0 4.0  2   1  8  0 90   0.07  0.50  13.2 1452  29
lpar12   A53 C 4.0 2.8  2   5  4  0 89   0.09  0.75  12.3  586  42
lpar13   A53 C 4.0 0.4  2   0  5  0 93   0.01  0.10  10.9  507   3
lpar14   A53 U 2.0 2.0  2   1  3  0 94   0.03  0.50   6.5  574  10
lpar15   A53 U 0.5 0.4  4   0  3  0 96   0.03  0.50   5.3 1252  13
lpar16   A53 U  10 8.0  2   0  2  0 96   0.03  0.75   4.1  541  17
lpar17   A53 U  12  10  4   0  1  0 97   0.03  1.00   3.1  594  13
lpar18   A53 U 8.0 8.0  6   0  2  0 97   0.08  2.50   3.0  544  19
lpar19   A53 U 4.0 3.7  2   0  1  0 98   0.02  0.75   2.5  520   9
lpar20   A53 C 4.0 3.0  2   0  1  0 98   0.02  1.00   1.7  501   5
lpar21   A53 U 8.0 6.1  4   0  0  0 99   0.02  1.25   1.2  536  11
lpar22   A53 U 8.0 6.1  4   0  0  0 97   0.08  2.50   3.0  536  11
```

Several important metrics are displayed in relation to shared processing; specifically, these are: the active physical CPUs in the pool (PSz), the available processors in the pool (APP), the busy shared processors (Shr_PhysB), an LPARs entitlement (Ent), the entitlement consumed for an LPAR (% Entc), and the sharing mode (M) of either capped (C) or uncapped (U). This view gave us a way to quickly see (per 595) the shared processor activity in the pool and in each micro-partition.

The **topas -R** command could record the Cross-Partition data. This was of great use to us, because we needed to collect statistics on our shared pool usage. Recordings covered a 24-hour period and were retained for 8 days before being automatically deleted. This allowed a week's worth of data to be retained. Recordings were made to the /etc/perf/ directory with file names in the form **topas_cec.YYMMDD**. To convert the collected data into a format that could be used for reporting, we used the new **topasout** command.

We chose to run the **topas** recorder on a single (non-business function) LPAR per p595. For this purpose, we deployed a VIO server and client to each 595. The VIO client would become our shared processor pool reporting LPAR. Given that the VIO server and client would not be under much load, we configured the LPARs with minimal resources (i.e., 0.1 processing units and 256M of memory). For more information on configuring a VIO server and client, please refer to the Redbook, "Advanced POWER Virtualisation on IBM System p5". We created two new file systems in rootvg to collect the topas data.

```
/etc/perf
/etc/perf/archive
```

Both file systems were roughly 600M in size. The /etc/perf file system held the data collected daily by the topas recorder and /etc/perf/archive contained an archive of old daily data. The **topas_cec** files were copied to the archive directory at 2 minutes to midnight to ensure we didn't lose any data. This archive was created by the following script in root's crontab:

```
58 23 * * * /usr/local/bin/cpperf.ksh > /var/log/cppperf.log 2>&1
```

The script contained the following commands:

```
cp -p /etc/perf/topas_cec* /etc/perf/archive/
gzip -f /etc/perf/archive/topas_cec*
```

We then started the **topas -R** command like so:

```
# /usr/lpp/perfagent/config_topas.sh add
AIX Topas Recording (topas -R) enabled
Sending output to nohup.out
```

Each day a new **topas_cec** file is written to and saved for future reporting purposes.

```
# ls -ltr /etc/perf/topas_cec* | tail -1

-rw-r--r--   1 root  sys   9905952 Jan 13 22:53 /etc/perf/topas_cec.070113
```

To view detailed and summary reports directly from the **topas_cec** files, we used the topasout command:

```
# topasout -R summary /etc/perf/topas_cec.060828 | more
#Report: CEC Summary  --- hostname: lpar01                   version:1.1
Start:08/28/06 09:33:21   Stop:08/28/06 23:59:22  Int: 5 Min  Range: 866 Min
Partition Mon: 15  UnM:  0  Shr: 15  Ded:  0  Cap:  0  UnC: 15
       -CEC------  -Processors------------------------ -Memory (GB)------------
Time   ShrB  DedB  Mon  UnM  Avl UnA  Shr Ded  PSz  APP  Mon  UnM  Avl  UnA  InU
09:38  0.61  0.00 12.8  0.0  0.0   0 12.8   0 32.0 31.4 94.0  0.0  0.0  0.0 51.3
09:43  0.80  0.00 12.8  0.0  0.0   0 12.8   0 32.0 31.2 94.0  0.0  0.0  0.0 51.5
09:48  1.08  0.00 12.8  0.0  0.0   0 12.8   0 32.0 30.9 94.0  0.0  0.0  0.0 51.6
09:53  1.00  0.00 12.8  0.0  0.0   0 12.8   0 32.0 31.0 94.0  0.0  0.0  0.0 51.6
09:58  0.70  0.00 12.8  0.0  0.0   0 12.8   0 32.0 31.3 94.0  0.0  0.0  0.0 51.6
10:03  1.77  0.00 12.1  0.0  0.0   0 12.1   0 32.0 30.2 87.6  0.0  0.0  0.0 50.8
10:08  0.81  0.00 12.0  0.0  0.0   0 12.0   0 32.0 31.2 86.0  0.0  0.0  0.0 50.8
10:13  1.13  0.00 12.0  0.0  0.0   0 12.0   0 32.0 30.9 86.0  0.0  0.0  0.0 51.0
10:18  1.39  0.00 12.0  0.0  0.0   0 12.0   0 32.0 30.6 86.0  0.0  0.0  0.0 51.1
...etc...

# topasout -R detailed /etc/perf/topas_cec.060828 | more
#Report: CEC Detailed --- hostname: lpar01                   version:1.1
Start:08/28/06 09:33:21   Stop:08/28/06 23:59:22  Int: 5 Min  Range: 866 Min

Time: 09:38:20 -----------------------------------------------------------------
Partition Info   Memory (GB)        Processors
Monitored  : 15  Monitored  : 94.0  Monitored  : 12.8  Shr Physcl Busy:  0.61
UnMonitored:  0  UnMonitored:  0.0  UnMonitored:  0.0  Ded Physcl Busy:  0.00
Shared     : 15  Available  :  0.0  Available  :  0.0
Dedicated  :  0  UnAllocated:  0.0  Unallocated:  0.0  Hypervisor
Capped     :  0  Consumed   : 51.3  Shared     : 12.8  Virt Cntxt Swtch:  9911
UnCapped   : 15                     Dedicated  :  0.0  Phantom Intrpt  :    15
                                    Pool Size  : 32.0
                                    Avail Pool : 31.4
Host         OS  M  Mem  InU Lp Us Sy Wa Id PhysB  Ent  %EntC  Vcsw PhI
-------------------------------------shared-------------------------------------
```

```
lpar01     A53 U  8.0  6.8  2 10  3  0 85  0.17  1.0  16.79   929   1
lpar02     A53 U  4.0  4.0  2  0  3  0 96  0.04  0.8   4.89   589   1
lpar03     A53 U  2.0  2.0  2  7 11  0 80  0.12  0.5  24.14  1534   2
lpar04     A53 U  4.0  3.5  2  0  4  0 94  0.04  0.5   8.52  1460   2
lpar05     A53 U 10.0  6.2  2  3  1  0 95  0.05  0.8   6.16   610   1
lpar06     A53 U  8.0  5.5  4  0  0  0 98  0.02  1.2   1.80   436   0
lpar07     A53 U  4.0  1.9  2  1  2  0 96  0.04  0.8   4.84   768   1
lpar08     A53 U  4.0  3.8  2  3  3  0 93  0.06  0.8   7.93   542   2
...etc...
```

We wanted to be able to easily graph this information for use in monthly reporting of our shared pool usage. To achieve this we used the "Performance Graph Viewer" (pGraph). This is a Java program designed to read files containing performance data and produce graphs. The tool is capable of producing graphs related to CPU, memory, disk, I/O, and network.

We copied the pGraph tool to /etc/perf/pGraph on each of the reporting LPARs. To graph the processor pool usage, we did the following.

Change directory to /etc/perf/archive (our collection point for **topas -R** data archive):

```
# cd /etc/perf/archive
```

Select the file(s) that match the period we wish to graph:

```
# ls -ltr | topas_cec* | tail -1
-rw-r--r--   1 root     system      3297232 Nov 23 09:43 topas_cec.061117
```

Format the data with **topasout**:

```
# topasout topas_cec.061117
```

A new file is created, which will be imported into pGraph:

```
-rw-r--r--   1 root     system     19652328 Nov 23 09:44 topas_cec.061117_01
```

Change directory to /etc/perf/pGraph and run the pGraph program (we export our display to our X workstation first):

```
# export DISPLAY=xws23:12
# /usr/java14/bin/java -cp pGraph.jar.zip pGraph.Viewer
```

Note that we could also concatenate the files and produce a report for several days' worth of data:

```
# cd /etc/perf/archive
# for i in `ls *0612*`
do
 cat $i >> topas_cec_Dec_06
done

# topasout topas_cec_Dec_06
```

Once the pGraph window appears, do the following steps. Select File, Open Single File. Enter Path/Folder Name of /etc/perf/. Select the file to open.

Once the file is loaded, the filename appears in the bottom left corner of the window. Click the CPU button. This might take a minute or so to load. The CPU stats window appears. To view just the Pool usage stats, click on the "None" button then tick just the "Pool Usage" option. The graph can then be saved as a PNG image. Click the "Save" button and enter a path and filename. See Figure 3 .

The resulting graph is then used in our monthly report on our shared processor pool usage. Looking at the graph for 595-3, on average 5.4 shared pool processors were in use, leaving another 30.6 processors free. This indicates that spare capacity exists in the shared processor pool on 595-3 and that we could deploy more LPARs to this system. The pGraph tool is available here:

```
http://www-941.ibm.com/collaboration/wiki/display/WikiPtype/Performance
+Graph+Viewer'
```

We also wrote a small script to monitor the shared pool on each 595 to ensure that the available pool did not fall below a certain threshold. This script (based on output from the **lparstat** command) was installed on each of our monitoring LPARs and then integrated into our Nagios monitoring tool so that, if the pool was low on resources, we would be sent an email and an SMS warning us of this situation. Nagios is an open source program for monitoring hosts, services, and networks. It is very customizable, so it can monitor virtually anything you can think of as you can integrate your own scripts easily into the framework. For more information, visit:

```
http://nagios.org/
```

Another handy tool worth mentioning is lparmon. It presents a basic graphical monitor of the shared pool utilization. It can also display processor usage for an LPAR if desired. We installed and ran the lparmon agent on each of our three reporting LPARs. I created a copy of the lparmon xml configuration for each 595 so that we could run multiple sessions of the tool and monitor the pool on each system:

```
lparmonv2 $ grep ipaddress lparmon.xml.*
lparmon.xml.1:            <ipaddress>192.168.1.10</ipaddress>
lparmon.xml.2:            <ipaddress>192.168.1.20</ipaddress>
lparmon.xml.3:            <ipaddress>192.168.1.30</ipaddress>
```

After this, it was simply a matter of running the **lparmon** command from one LPAR with this small script (again, we export our display to our X workstation):

```
#!/usr/bin/ksh
#
# Script name: lm2 - lparmon v2 wrapper.
#

case "$1" in
1)
  echo "LPARMON for 595-1"
  cp lparmon.xml.1 lparmon.xml
  ./lparmon &
  ;;

2)
  echo "LPARMON for 595-2"
  cp lparmon.xml.2 lparmon.xml
  ./lparmon &
  ;;

3)
  echo "LPARMON for 595-3"
  cp lparmon.xml.3 lparmon.xml
  ./lparmon &
  ;;

*)
```

```
  echo "Which 595 do you want to run lparmon on? e.g. ./lm2 2 \
    will run lparmon for 595-2"
  exit 1
  ;;
esac
exit 0
```

```
$ export DISPLAY=xws23:12
$ ./lm2 1
$ ./lm2 2
$ ./lm2 3
```

See Figure 4 .

More information on lparmon can be found here:

```
http://www.alphaworks.ibm.com/tech/lparmon
```

Considerations

You should review the information in the IBM Redbooks before implementing micro-partitioning. Make yourself aware of the performance considerations. The "Advanced POWER Virtualization on IBM eServer p5 servers: Architecture and Performance Considerations" Redbook covers memory and processor affinity with micro-partitioning, virtual processor dispatch latency, general rules on virtual processor sizing, general rules on the number of micro-partitions per managed system, and the overhead when running micro-partitioning. Despite some misconceptions from one or two of our developers, however, we have not found any significant degradation in performance as a result of micro-partitioning. Workloads that constantly consume their entitled capacity may perform better with dedicated processors.

The traditional methods for monitoring CPU utilization can be misleading in shared processor environments (particularly with uncapped LPARs). To correctly report CPU performance, the AIX performance tools have been enhanced to use the p5 Processor Utilization Resource Register (PURR). Processor utilization is now relative to an LPARs CPU entitlement, so us/sys/wa/id relate to the percentage of physical processor utilization consumed. To make matters more confusing, an uncapped LPAR can access more than its entitled capacity. So, at 90-100% CPU utilization, an LPAR could be accessing processor above its entitlement, which makes the utilization numbers meaningless. It is advisable to focus on actual physical CPU consumed (pc) and the percentage of entitlement consumed (%entc). Commands like **lparstat** can provide you with statistics for shared processor usage.

Be aware of the licensing implications when using uncapped LPARs. Check with your application software vendors as to how they license their software. Some base their licensing on the maximum number of CPUs a system could possibly access. So for an uncapped LPAR, this could be all of the CPUs in the shared pool. Depending on the licensing used, you may need to configure an LPAR as capped to restrict it to a certain number of CPUs.

Conclusion

To date, we have reduced our hardware costs by moving many p4 servers to only a few p5 systems. As our consolidation project continues in 2007, we will continue moving all of our workload to the p595s. This will allow us to continue extracting greater utilization from our existing p5 processors and avoid the additional cost of buying new hardware. Our current utilization of each processor pool is around 30%. We are aiming for our shared processing resources to be 70% utilized, as opposed to the 20% we used to tolerate with dedicated CPUs. Given the current utilization and spare capacity of the pool, we are confident that we can size systems to meet the demands of the company while maintaining our current unit costs.

References

Advanced POWER Virtualization on IBM System p5 --
**http://www.redbooks.ibm.com/Redbooks.nsf/ \
RedbookAbstracts/sg247940.html?OpenDocument**

Advanced POWER Virtualization on IBM eServer p5 Servers: Architecture and Performance Considerations -- **http://www.redbooks.ibm.com/abstracts/sg245768.html?Open**

POWER5 Hypervisor --
http://www-941.ibm.com/collaboration/wiki/display/virtualisation/POWER5+Hypervisor

Virtualization Company Assessments --
http://www-941.ibm.com/collaboration/wiki/display/virtualisation/Virtualisation+Company+Assessments

Micro-Partitioning performance implications --
**http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/ \
com.ibm.aix.prftungd/doc/prftungd/micro_part_perf_impl.htm**

Configuring Processor Resources for System p5 Shared-Processor Pool Micro-Partitions --
**http://www.ibmsystemsmag.com/ME2/Audiences/Segments/Publications/Print.asp?Module= \
Publications::Article&id=551082D743DC41A1B6E2C80B16B8CCC6**

Create Web Pages from nmon Files Using nmon2web Tool: Part 3 --
http://users.ca.astound.net/~baspence/AIXtip/nmon2rrdv3.htm

Chris Gibson works for one of Australia's largest insurance companies as a Senior AIX Systems Administrator. Chris has been working with Unix systems for more than 12 years. Since 1999, he has been involved with RS/6000, SP, and pSeries systems running AIX. His areas of expertise include AIX 5L and NIM. He has also worked with HACMP, CSM, and PSSP. He is an IBM Certified Advanced Technical Expert -- pSeries and AIX 5L. He can be reached at: ***cg@gibsonnet.net****.*