IBM AIX 5L Version 5.3 operating system performance tools update (part 2)

by Hsian-Fen Tsao
and Brad Cobb

IBM eServer Solutions Enablement

June 2005

Table of contents

Abstract

The introduction of new IBM® AIX 5L™ V5.3 operating system performance tools was already covered in part 1 of this series of white papers. In this paper (part 2), the reader will be introduced to the new features of existing AIX 5L V5.3 text- and traced-based performance tools (such as vmstat, iostat, sar, topas, trace, trcrpt, curt, and tprof). The importance of the tools from the tuning and analysis aspect will be demonstrated using new metrics. The tests use real-world data collected under Micro-Partitioning™ within a simultaneous multithreaded environment on an IBM eServer™ p5 server driven by IBM POWER5® technologies. Some examples with data collected under a dedicated partition are also included to show the reported differences between shared and dedicated partitions.

Introduction

POWER5 technology introduces simultaneous multithreading (SMT) and IBM Micro-Partitioning™ to significantly improve system performance and utilization. These technologies literally obsolesce the traditional, sample-based approach to measuring processor utilization. Therefore, a new hardware capability, a register called Process Utilization Resource Register (PURR), is implemented on POWER5 to aid performance management software in capturing relevant measurements. Consequently, performance tools in the AIX 5L Version 5.3 operating system have been enhanced to reflect new CPU utilization metrics. Additional performance tools are included and some tools have been updated.

AIX 5L V5.3 performance tools updates

The performance tools of AIX 5L V5.3 have been adapted for measuring new POWER5 features and new processor utilization metrics.

Sample configuration for this paper

These updated performance tools will be discussed by using sample data. The examples throughout this paper were taken from a 4-way POWER5 server with the following specifications:

- Virtualized into five logical partitions
- Each partition configured as 8/10ths of a POWER5 CPU
- Two virtual CPUs enabled with SMT
- Running with capacity capped and uncapped

The test workload consists of a Java™ 2 Enterprise Edition (J2EE™) application running under the following environments:

- AIX 5L V5.3 operating system
- IBM DB2® Universal Database™ (UDB)
- IBM WebSphere® Application Server

Updated text-based tools

Beginning with AIX V5.3, the vmstat, iostat, sar and topas commands automatically use new PURR-based data and calculation formulas for the %usr, %sys, %wait, and %idle values in SMT or shared processor environments. In addition, these commands add two new metrics: physical processor consumed and percentage of entitlement consumed by the partition. All of these tools provide a new feature called dynamic configuration support. They display a preheader with the configuration (*Example 1*). If a system configuration change is detected, a warning will be displayed followed by a new configuration preheader.

The vmstat command

The **vmstat** command collects and reports the overall picture of system-wide utilization (such as CPU, kernel thread, virtual memory, traps, and I/O activity). This report can be used to balance system load activity.

- Beginning with AIX V5.3, **vmstat** has two new columns: *pc* and *ec*. Both new metrics will only display on shared processor partition environments (*Example 1*).
- *pc* measures the number of physical processors consumed.
- *ec* calculates of the percentage of entitled capacity consumed.

```
# vmstat 5 3

System configuration: lcpu=4 mem=3072MB ent=0.80     <- Pre-header w/configuration

kthr    memory             page                  faults       cpu
----- -------------   ----------------------  ------------ -----------------------
 r  b   avm     fre   re  pi  po  fr  sr  cy  in  sy  cs    us sy id wa  pc  ec
 1  0 481664 146995   0   0   0   0   0   0   26  2628 925  92  1  7  0 1.03 128.5
 1  0 481735 145139   0   0   0   0   0   0   24  3495 920  91  1  7  0 1.04 130.2
```

Example 1: vmstat on SMT shared processor partition (capacity uncapped)

In *Example 1*, the activity begins on the partition and the percentage of used CPU increases to 92%. Because the partition is uncapped and the processor pool is not fully consumed, the percentage of entitlement consumed goes up to 130%, and the processor consumed is greater than 1. This means that the partition is running on more than a full processor, even though it has been given only 0.8 (80% of a processing unit).

The iostat command

The **iostat** command collects and reports system-wide CPU statistics, asynchronous input/output (AIO), and detailed I/O utilization for the entire system (disks, adapters, tty, and CD_ROMs). This command is especially useful in determining if a physical volume or adapter becomes a performance bottleneck and if there is potential to improve the performance.

Beginning with AIX V5.3, the **iostat** command reports the number of physical processors consumed (physc) and the percentage of entitlement consumed (% entc). Both new metrics will display only on shared processor partition environments.

The **iostat** command also has a new way to look at asynchronous I/O with the following flags specified:

Flags	What it does
-A	Shows CPU utilization and legacy AIO (*Example 2*)
-P	Shows CPU utilization and POSIX AIO
-q	Shows adapter individual queues and their request counts
-Q	Shows mounted filesystems and their associated adapter queue and request counts

- The **iostat** command, with –A or –P specified, produces reports that contain new columns (such as: avgc, avfc, maxgc, maxfc, and maxregs). These new columns replace the tty columns. (*Example 2*)

```
# iostat -A 5 5

System configuration: lcpu=4 drives=30 ent=0.80

aio: avgc avfc maxg maif maxr avg-cpu: %user %sys %idle %iow physc %entc
     0    0    0    0    4096           91.6  1.2  7.1   0.0  1.03  128.5

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
Hdisk14          1.2       206.2     3.2          0      1024
Hdisk15          0.8       206.2     3.2          0      1024
Hdisk17          0.4       193.4     3.0          0       960
[rest of data have been deleted]
```

Example 2: Show the statistics of "legacy" AIO in SMT uncapped shared partition

New Metrics	What it measure
avgc	Average global AIO request count
avfc	Average fastpath request count
maxg	Max. global AIO request count since the last time this value fetched
maxf	Max. fastpath request count since the last time this value fetched
maxr	Max. AIO request allowed on queue
physc	Percentage of physical processor consumed
%entc	Percentage of entitlement capacity consumed

The percentage of entitlement consumed goes up to 128% in example 2 due to the fact that the partition can get more capacity than it is actually allocated in uncapped mode.

When the system is identified as having an I/O problem, the next step is to find out where the problem originates. The dedicated adapter and disk for the partition are the items that need to be verified using the **iostat** command. For instance, using the percentage of time the physical disk was active in the iostat report can identify the file I/O activity that is unevenly distributed on physical disks.

Performance consideration: Some system resource is consumed in maintaining disk I/O history for the **iostat** report. For performance reasons, it is recommended that history accounting be stopped. This can be done via the System Management Interface Tool (SMIT).

The sar command

The **sar** command collects and reports system-wide statistics, such as CPU and I/O utilization. If **-P ALL** is given, **sar** reports statistics for each logical processor, followed by system-wide statistics.

Beginning with AIX V5.3, **sar** also adds two new metrics: *physc* and *%entc*. Both metrics will display only on shared processor and SMT environments *(Example 3).* When SMT is on or running in shared mode, only *physc* is displayed *(Example 4).*

- *physc* measures the fraction of time a logical processor was getting physical processor cycles.
- *%entc* calculates the relative entitlement consumed for each logical processor.

```
# sar -P ALL 5 5

System configuration: lcpu=4 ent=0.80

16:00:18 cpu    %usr    %sys    %wio   %idle   physc   %entc
16:00:23  0       27      65       0       7    0.00     0.5
          1        0       8       0      92    0.00     0.2
          2       10      45       0      45    0.00     0.2
          3        0      14       0      86    0.00     0.1
          U        -       -       0      98    0.79    99.0
          -        0       0       0      99    0.01     1.0
```

*Example 3: Report all logical processor's activity in **SMT shared** partition*

If the **–P** flag is specified and the partition is running with shared processors in uncapped mode, then *%entc* will report the percentage of granted entitled capacity against each CPU row and the percentage of allocated entitled capacity in the system-wide CPU row. A CPU row with **U** will also be reported to show the system-wide unused capacity (if there is unused capacity) (*Example 3*).

```
_____________________________________________________________________________
# sar -P ALL 5 5
System configuration: lcpu=4

10:44:41 cpu    %usr    %sys    %wio   %idle   physc
10:44:51  0       13       5       0      82    0.73
          1        0       0       0     100    0.27
          2       19       4       0      77    0.73
          3        0       0       0     100    0.27
          -       12       3       0      85    2.00<-system-wide stats for all logical processors
```

*Example 4: Report all logical processor's activity in **SMT dedicated** partition*

The topas command

The **topas** command reports the local system's statistics, including: CPU, network, I/O, processes, and workload management classes utilization. It uses the curses library to display its output in a format suitable for viewing on an 80x25 character-based display.

Beginning with AIX V5.3, the new POWER5 processors are able to track the consumption of physical processing resources through two new metrics (*physc* and *%entc)*, which display only in shared mode *(Example 5).*

- *physc* measures the number of physical processors granted to the partition.
- *%entc* measures the percentage of entitled capacity granted to a partition.

```
# topas 5 5

Topas Monitor for host:    lcb21                        EVENTS/QUEUES    FILE/TTY
Wed Feb 23 18:51:46 2005   Interval:  2                 Cswitch   1106  Readch  2304.5K
                                                        Syscal    3663  Writech 2636.7K
Kernel    1.2   |#                                  |  Reads     1065  Rawin         0
User     91.5   |########################           |  Writes     661  Ttyout      140
Wait      0.0   |                                   |  Forks        0  Igets         0
Idle      7.3   |###                                |  Execs        1  Namei       185
Physc =   1.05                          %Entc= 131.8  Runqueue   1.0  Dirblk        0
                                                      Waitqueue   0.0

Network   KBPS   I-Pack  O-Pack   KB-In  KB-Out
en0        1.1     12.0     4.0     1.0     1.2      PAGING           MEMORY
lo0        0.0      0.0     0.0     0.0     0.0      Faults      231  Real,MB    3071
                                                     Steals        0  % Comp     62.2
Disk     Busy%     KBPS     TPS KB-Read KB-Writ      PgspIn        0  % Noncomp  22.5
hdisk1     1.0      4.0     1.0     0.0     8.0      PgspOut       0  % Client   23.0
hdisk7     0.0      0.0     0.0     0.0     0.0      PageIn        0
hdisk9     0.0      0.0     0.0     0.0     0.0      PageOut     192  PAGING SPACE
hdisk12    0.0      0.0     0.0     0.0     0.0      Sios        192  Size,MB    2048
                                                                      % Used      0.8
Name            PID  CPU%  PgSp Owner               NFS (calls/sec)  % Free     99.1
sort         372920  12.5   4.2 root                ServerV2       0
mksort       376838  12.5   0.1 root                ClientV2       0   Press:
db2fmcd      610350   0.0   0.7 root                ServerV3       0   "h" for help
topas        385076   0.0   1.3 root                ClientV3       0   "q" to quit
```

Example 5: Display logical partition statistics with 'topas' in SMT shared partition

With **–L** flag, the **topas** command displays similar logical partition statistics as **lparstat** and **mpstat** mentioned in the white paper *AIX 5L V5.3 performance tools update - part 1* (*Example 6).*

```
# topas -L 5                                                    Note: user comment after <-

 Interval:   2      Logical Partition: lcb21          Wed Feb 23 18:54:06 2005    <-lparstat -i
 Psize:   4         Shared SMT  ON                     Online Memory:   3072.0
 Ent: 0.80          Mode: UnCapped                    Online Logical CPUs:  4
 Partition CPU Utilization                            Online Virtual CPUs:  2

 %usr  %sys  %wait  %idle physc  %entc   %lbusy   app   vcsw phint  %hypv   hcalls <-lparstat -h
   92    1     0      7     1.0    125.50  25.00    2.95  1024 4      0.0     0

 LCPU  minpf majpf  intr  csw  icsw runq lpa scalls usr sys _wt idl   pc    lcsw <-mpstat
 Cpu0      0     0   351   272  113    1 100    142  18  57   0  25   0.00  294
 Cpu1      0     0    62    88   44    0 100     40  12  16   0  71   0.00  298
 Cpu2      0     0    50    92   33    0 100   1886   1   1   0  98   0.06  242
 Cpu3      0     0   204     2    2    1 100   7971  99   1   0   0   0.94  190
```

Example 6: Topas –L displays a LPAR sreen with 'lparstat' & 'mpstat' metrics in SMT shared partition

The metrics (*Example 5 and 6*) of psize, app, physc, %entc, %lbusy, lcse, vcsw, and phint are shared mode only.

Updated trace-based tools

The following trace-based tools (trace, trcrpt, curt, and tprof) have been updated with AIX V5.3. All trace-based tools can use PURR metrics to calculate CPU utilization. They adjust processor times using a preemption hook. Most POWER Hypervisor™ calls are traceable and will appear in the **trcrpt** output.

The trace command

trace is used to collect the system events. The data collected by the trace function are recorded in the trace log. A report from the trace log can be generated with the **trcrpt** command. Beginning with AIX V5.3, traces add the following enhancements on SMT and shared-partition environments:

- There are new flags to trace selected processes and threads. Traces can be invoked on a single program as well as on interrupt events. It is possible to specify whether the trace is to be propagated to subsequently created processes or threads. This is only valid for trace channel 0. The new flags that deliver these functions are as follows:
  - **–A <process-list>**: With this flag, the trace is invoked only for the existing process.
  - **–t <thread-list>**: With this flag, the trace is used only for the existing thread.
  - **–x/-X <specified program>**: With this flag, the trace is executed for the specified program, including all processes and threads the program creates. But the trace will not automatically stop when the program exits if –X specified.
  - **–P <propagation>**: With this flag, the trace should be propagated:
    - Across process creation (-Pp)
    - Across thread creation (-Pt)
    - Or, there should be no propagation (-Pn).
  - **–l**: With this flag, interrupt tracing should be invoked.
- With the flag **-r PURR** specified (64-bit kernel only), PURR register values can be collected at each trace hook for better results. For example:
  `trace –adf –C all –r PURR –o trace.raw`

AIX V5.3 allows tracing of virtual processor dispatch, phantom interrupts, POWER Hypervisor calls, and lock contention with new hooks.

- Preemption hook ID 419 marks undispatched time. It shows CPU preemption, prod, and confer, indicating logical processor preemption. This useful information provides trace output such as the preemption reason, duration, and physical processor ID. (*Example 7*)
- Trace hook ID 47F marks phantom interrupts to indicate that the current interrupt is not for this partition. (*Example 8*)
- Trace hook ID 492 marks POWER Hypervisor calls (hcalls). Most hcalls can be traced. Virtual Memory Manager (VMM)-related calls are not traced, but show in **lparstat –H**. (*Example 9*)
- Subhook ID 112 locks code for krlock transitions and shared-mode handoff operations. (*Example 10*)

The trcrpt command

trcrpt reads the trace log generated by the **trace** command, formats the trace entries, and writes a report, either to standard output or to a file.

The **trcrpt** command can display **elapsed purr** with the flag **-O PURR=on**. To format the trace log generated by using the **trace** example (shown in the shaded box on the previous page) and also using the PURR calculation, use the following:

```
trcrpt -r –C all –O PURR=on trace.log > trace.r
```

Either SMIT or the command line can be used to run the **trcrpt** command.

```
ID   PROCESS NAME   CPU  TID     I   ELAPSED_SEC     DELTA_MSEC   APPL    SYSCALL KERNEL  INTERRUPT
419  wait             2  16393       0.005782249      0.000179                                cpu preemption data  EI
                                                                                                  vProcIndex=0006
                                                                                        rtrdelta=04D  enqdelta=002A  exdelta=028C
                                                                                    start wait=C5B8FBD40900    end wait=C5B8FBD41087
                                                                                  SRR0=00000000001EB14C  SRR1=8000000000009032
```

Example 7: Hook 419 indicates logical processor preemption

The trace report in Example 7 is explained at next page:

New Information	What it means
Interrupt Type	Interrupt types: • EI: external interrupt • LpEvent • LpProd: this processor has been prodded • Timeout • IPL • Terminate • Unblocked: switching back to SMT • Preempted: SPLPAR timeslice
vProcIndex	Index to the physical processor • If a virtual processor number is dispatched to the same vProcIndex as the previous dispatch, affinity is maintained
rtrdelta	• TB delta until ready to run: number of tics VP had nothing to do (after h_cede or h_confer)
enqdelta	• TB delta until enqueued: wait on frozen Q (entitled capacity had expired)
exdelta	• TB delta until running: wait on dispatcher for physical CPU

```
ID  PROCESS NAME   CPU TID     I   ELAPSED_SEC     DELTA_MSEC   APPL    SYSCALL KERNEL  INTERRUPT

100 IBM.CSMAgentRM 0    659527  1   4.898986848      0.001816         I/O INTERRUPT iar=328C cpuid=00
492 IBM.CSMAgentRM 0    659527  2   4.898987466      0.000618         h_call: start H_XIRR iar=3A86BF0 p1=184DD50
                                                                      p2=74D6F08 p3=C5B9383B35DA
492 IBM.CSMAgentRM 0    659527  2   4.898991393      0.003927         h_call: end H_XIRR  iar=3A86BF0 rc=0000
47F IBM.CSMAgentRM 0    659527  2   4.898992161      0.000768         phantom interrupt cpuid=00
```

Example 8: New trace hook 47F indicates that the current interrupt is not for this partition.

```
ID  PROCESS NAME  CPU  TID    I   ELAPSED_SEC    DELTA_MSEC  APPL   SYSCALL KERNEL  INTERRUPT
492 wait          2    20491      0.005785630    0.000725          h_call: start H_CEDE iar=500B p1=C5B8FBD868E9
                                                                    p2=0000 p3=0000
492 wait          2    20491      0.005782369    0.000120          h_call: end H_CEDE iar=C5B8FBD85E1F rc=0000
```

Example 9: These are hook 492 marks for POWER Hypervisor calls.

```
ID  PROCESS NAME CPU PID    TID    I  SYSTEM CALL ELAPSED_SEC  DELTA_MSEC  APPL  SYSCALL KERNEL  INTERRUPT

112 -667718-          4  667718  1327241  thread_waitl  1.335507872    0.000570          lock:       miss lock
 addr=780000022B3C9EC lock status=10000000 requested_mode=LOCK_SWRITE return addr=37BC name=00000000.00000000
112 -667718-          4  667718  667718                 1.335508674    0.000320          krlock:  cupid=04
 addr=780000022B385A4 action=spin
```

Example 10: Krlock tracing is done using KERN_LOCK (112) subhooks.

- List of subhooks:
  - hkwd_KRLOCK
  - hkwd_KRLOCK_ALLOC: allocates the krlock
  - hkwd_KRLOCK_FREE : frees the krlock
  - hkwd_KRLOCK_ACQUIRE : acquires the krlock
  - hkwd_KRLOCK_RELEASE : releases the krlock
  - hkwd_KRLOCK_HANDOFF : hands off krlock to another thread after lock is no longer needed
  - hkwd_KRLOCK_CONFER : confers its cycles to another processor (the one that. owns the lock).
  - hkwd_KRLOCK_PROD : prods another processor
  - hkwd_KRLOCK_SPIN: spins on krlock; a thread is unsuccessfully trying to get the krlock

Performance consideration: Lock contention is statistical. Reducing the number of virtual processors in a partition will usually decrease lock contention. The AIX® Simple Performance Lock Analysis Tool (**SPLAT)** can be used to help with the analysis of AIX kernel lock contention.

The curt command

The **curt** command is a CPU utilization reporting tool that maps CPU consumption to system activity. It takes an AIX trace file as input and provides detailed CPU statistics on: the system, processor, processes, kernel threads, pthread, application, system calls, POWER Hypervisor calls, and NFS.

Beginning with AIX V5.3, **curt** adds information (bold in *Example 11*), such as physical affinity and phantom interrupt statistics. It also shows POWER Hypervisor (HCALL) summary reports similar to system call reports, and the number of preemptions, as well as the number of H_CEDE and H_CONFER for each physical processor. The **curt** command has a new network file system (NFS) statistic in system and processor summary reports. It displays a new type of kproc (N) and marks all NFS kproc. The **curt** command also adds an NFS Summary section with AIX V5.2 and AIX V5.3 sub-sections, as well as AIX V5.3 in a shared processor environment.

It is possible to specify the use of the PURR register to calculate CPU time with flag **-r PURR**. For instance:

```
curt -est -r PURR -i curttrace.r -n gen.names -o curt.out
```

```
_____________________________________________________________________________________
              Processor Summary Processor Number 0
              -------------------------------------
 processing       percent       percent
 total time        total time     busy time
     (msec)       (incl. idle)  (excl. idle)  processing category
===========   ===========   ===========  ==================
     233.26         82.95         82.95  APPLICATION
      20.84          7.41          7.41  SYSCALL
       8.68          3.09          3.09  HCALL
       1.40          0.50          0.50  KPROC (excluding IDLE and NFS)
       0.00          0.00          0.00  NFS
      14.95          5.32          5.32  FLIH
       2.18          0.78          0.78  SLIH
       8.58          3.05          3.05  DISPATCH (all procs. incl. IDLE)
       0.63          0.22          0.22  IDLE DISPATCH (only IDLE proc.)
-----------    ----------       -------
     281.21        100.00        100.00  CPU(s) busy time
       0.01          0.00                IDLE
-----------    ----------
     281.22                              TOTAL

Avg. Thread Affinity = 0.99

Total number of process dispatches = 914
Total number of idle dispatches       = 497

Total Physical CPU time (msec) = 290.55      <- also added to system summary section
Physical CPU percentage          = 5.81
Physical processor affinity          =  0.781977
Dispatch Histogram for processor (PHYSICAL CPUid : times_dispatched).

  PHYSICAL   CPU 0 : 1032
Total number of preemptions    = 1032
Total number of H_CEDE       = 986     with preeemption = 986
Total number of H_CONFER    = 0        with preeemption = 0

                Hypervisor Calls Summary
                ------------------------
   Count    Total Time   % sys  Avg Time  Min Time  Max Time  HCALL (Address)
               (msec)     time    (msec)    (msec)    (msec)
========  ===========  ======  ========  ========  ========  =================
    1699       4.9612   0.89%    0.0029    0.0013    0.0087  H_CEDE(9032)
     589       1.4809   0.27%    0.0025    0.0021    0.0042  H_XIRR(3a86bf0)
    1102       0.9092   0.16%    0.0008    0.0006    0.0016  H_PUT_TCE(3aa4290)
     588       0.5441   0.10%    0.0009    0.0008    0.0018  H_EOI(3a83d44)
     698       0.5029   0.09%    0.0007    0.0005    0.0015  H_PUT_TCE(3a9a12c)
     397       0.4186   0.08%    0.0011    0.0006    0.0023  H_PROD(2941c)
     153       0.3495   0.06%    0.0023    0.0013    0.0047  H_CEDE(3a86bf0)

                      Global Flih Summary
                      ===================
Count   Total Time     Avg Time       Min Time       Max Time       Flih type
=====   ==========     ========       ========       ========       =========
1       0.0018         0.0018         0.0018         0.0018         9(PHANTOM)
588     8.0206         0.0136         0.0019         0.0475         5(IO_INTR)
```

Example 11: Beginning with AIX 5.3, curt adds new information (marked in bold).

The tprof command

A program execution is usually a mix of application, library, kernel, and kernel extensions. Frequently, a program that has not been well tuned spends most of its CPU cycles in a few statements or subroutines. These hot spots can be considered a performance bottleneck and **tprof** can pinpoint them. The **tprof** command is a sample-based profiler that supports the profiling of the following metrics:

- User programs (the **–x** ***Program*** flag)
- AIX kernel and kernel extensions (the **–k** and **–e** flags, respectively)
- Shared libraries (the **–s** flag)
- Processes (the **–p** ***ProcessList*** flag)
- Threads (the **–t** flag)
- Java™ programs (the **–j** flag).
- Micro-profiling (the **–m** flag)

For example: `tprof –skutj`

The raw data for **tprof** is obtained through the trace utility. When **tprof** is called, the trace is invoked and instructed to collect data from trace hook ID 234. Then **tprof** generates the profiling report at the level of object files, processes, threads, and subroutines.

Beginning with AIX V5.3, **tprof** can optionally use delta PURR samples, which is equivalent to using weighted samples and correcting inflated idle and spin times. The use of PURR-based metrics to calculate CPU utilization is preferred in SMT or Micro-Partitioning environments.

With the **-R** flag specified, **tprof** uses PURR-based metrics to calculate the CPU utilization percentage and displays a **pre-header** to indicate that PURR-based metrics were used. It is also possible to see POWER Hypervisor calls, such as **h_cede**. (*Example 12*)

```
_____________________________________________________________________________
PURR was used to calculate percentages.

Process                                Freq  Total Kernel   User Shared  Other
=======                                ====  ===== ======   ==== ======  =====
wait                                      2   3.24   3.24   0.00   0.00   0.00
/home/db2inst1/sqllib/bin/db2fm          40   2.60   0.38   0.00   2.21   0.00
db2sysc                                  14  13.38   0.44   0.00  12.93   0.00
[ rest of data has been removed]

Profile: /unix
Total % For All Processes (/unix) = 23.74

Subroutine                                        %   Source
==========                                      ====== ======
.waitproc                                         1.08 rnel/proc/dispatch.c
.xmalloc                                          0.94 rnel/alloc/xmalloc.c
[data has been removed]
.net_malloc                                       0.07 el/net/kern_malloc.c
.h_cede                                           0.07 hcalls.s
h_cede_end_point                                  0.07 hcalls.s
```

Example 12: Partial report of PURR-based profiling in SMT dedicated partition

Summary

This white paper provided an overview of modified text- and trace-based tools. The reader now has a rather thorough grasp of the enhancement of performance commands (vmstat, iostat, sar, topas, trace, trcrpt, curt, and tprof). These well established AIX tools are useful to monitor system performance from both an overall and detailed perspective. These tools are also helpful when analyzing performance bottlenecks for AIX V5.3 on the POWER5 platform.

AIX performance tools are easy-of-use, are well documented, and have powerful features. The importance, and advantage, of using the new and enhanced AIX V5.3 performance analysis and monitoring tools have been demonstrated in many real-life cases. Developers, systems analysts, and database administrators can greatly benefit from using them to understand application characteristics and system resource utilization. These tools also provide assistance in identifying performance bottlenecks and tuning overall performance.

Additional information

- White paper: AIX 5L V5.3 performance tools update (part 1)
  ibm.com/servers/enable/site/peducation/abstracts/abs_429e.html

- IBM eServer pSeries, and AIX Information Center
  http://publib.boulder.ibm.com/infocenter/pseries/index.jsp

- IBM Publications Center
  www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi?CTY=US
- Redbooks
  ibm.com/redbooks
    - Advanced POWER Virtualization on IBM eServer p5 Servers: Introduction and Basic Configuration (SG24-7940-00)
    - Partitioning Implementations for IBM eServer p5 Servers (SG24-7039-02)
    - Redbooks Technote: What is a Hardware Management Console?
      http://publib-b.boulder.ibm.com/Redbooks.nsf/5193609f3941e9cf85256bc300724cfc/c9cbd629131dc2d485256d790059ac16?OpenDocument

- White paper: IBM eServer p5 AIX 5L Support for Micro-Partitioning and Simultaneous Multi-threading
  ibm.com/servers/aix/whitepapers/aix_support.pdf

- AIX 5L V5.3 Performance Management Guide and
  AIX 5L V5.3 Performance Tools Guide and Reference
  http://publib.boulder.ibm.com/infocenter/pseries/index.jsp
  (click on "**AIX Documentation Page**")

About the Author

Hsian-Fen Tsao
pSeries technical consultant

Hsian-Fen Tsao is an IBM technical consultant for pSeries and AIX software vendors. Before joining the Solution Enablement Group in Austin, Texas she worked in the pSeries and AIX performance group for 10 years. Her experiences include database (online transaction processing and decision support systems), Web servers, TCP/IP, Java, WebSphere, and the POWER5 virtualization engine.

Brad Cobb
Senior pSeries technical consultant

Brad Cobb is a senior technical consultant for the IBM Solutions Enablement organization. For more than 10 years, Brad has helped solution providers port, tune, debug, and enhance their applications to run on the AIX operating system. Aside from helping developers, Brad enjoys filing for patents, publishing articles, speaking at developer conferences, and attempting to master the ever humbling trumpet.

Trademarks and special notices

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both: IBM, the IBM logo, eServer, pSeries, AIX, AIX 5L, POWER, POWER5, Micro-Partitioning, POWER Hypervisor, DB2, DB2 Universal Database, and WebSphere.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.