NAME

pmcstat - performance measurement with performance monitoring hardware

SYNOPSIS

pmcstat [**-A**] [**-C**] [**-D** *pathname*] [**-E**] [**-F** *pathname*] [**-G** *pathname*] [**-I**] [**-L**] [**-M** *mapfilename*] [**-N**] [**-O** *logfilename*] [**-P** *event-spec*] [**-R** *logfilename*] [**-S** *event-spec*] [**-T**] [**-U**] [**-W**] [**-a** *pathname*] [**-c** *cpu-spec*] [**-d**] [**-e**] [**-f** *pluginopt*] [**-g**] [**-i** *lwp*] [**-l** *secs*] [**-m** *pathname*] [**-n** *rate*] [**-o** *outputfile*] [**-p** *event-spec*] [**-q**] [**-r** *fsroot*] [**-s** *event-spec*] [**-t** *process-spec*] [**-u** *event-spec*] [**-v**] [**-w** *secs*] [**-z** *graphdepth*] [*command* [*args*]]

DESCRIPTION

The **pmcstat** utility measures system performance using the facilities provided by hwpmc(4).

The **pmcstat** utility can measure both hardware events seen by the system as a whole, and those seen when a specified set of processes are executing on the system's CPUs. If a specific set of processes is being targeted (for example, if the **-t** *process-spec* option is specified, or if a command line is specified using *command*), then measurement occurs till *command* exits, or till all target processes specified by the **-t** *process-spec* options exit, or till the **pmcstat** utility is interrupted by the user. If a specific set of processes is not targeted for measurement, then **pmcstat** will perform system-wide measurements till interrupted by the user.

A given invocation of **pmcstat** can mix allocations of system-mode and process-mode PMCs, of both counting and sampling flavors. The values of all counting PMCs are printed in human readable form at regular intervals by **pmcstat**. The format of **pmcstat**'s human-readable textual output is not stable, and could change in the future. The output of sampling PMCs may be configured to go to a log file for subsequent offline analysis, or, at the expense of greater overhead, may be configured to be printed in text form on the fly.

Hardware events to measure are specified to **pmcstat** using event specifier strings *event-spec*. The syntax of these event specifiers is machine dependent and is documented in pmc(3).

A process-mode PMC may be configured to be inheritable by the target process' current and future children.

OPTIONS

The following options are available:

-A Skip symbol lookup and display address instead.

-C Toggle between showing cumulative or incremental counts for subsequent counting mode PMCs

specified on the command line. The default is to show incremental counts.

-D *pathname*
Create files with per-program samples in the directory named by *pathname*. The default is to create these files in the current directory.

-E Toggle showing per-process counts at the time a tracked process exits for subsequent process-mode PMCs specified on the command line. This option is useful for mapping the performance characteristics of a complex pipeline of processes when used in conjunction with the **-d** option. The default is to not to enable per-process tracking.

-F *pathname*
Print calltree (Kcachegrind) information to file *pathname*. If argument *pathname* is a "-" this information is sent to the output file specified by the **-o** option.

-G *pathname*
Print callchain information to file *pathname*. If argument *pathname* is a "-" this information is sent to the output file specified by the **-o** option.

-I Show the offset of the instruction pointer into the symbol.

-L List all event names.

-M *mapfilename*
Write the mapping between executable objects encountered in the event log and the abbreviated pathnames used for gprof(1) profiles to file *mapfilename*. If this option is not specified, mapping information is not written. Argument *mapfilename* may be a "-" in which case this mapping information is sent to the output file configured by the **-o** option.

-N Toggle capturing callchain information for subsequent sampling PMCs. The default is for sampling PMCs to capture callchain information.

-O *logfilename*
Send logging output to file *logfilename*. If *logfilename* is of the form *hostname*:*port*, where *hostname* does not start with a '.' or a '/', then **pmcstat** will open a network socket to host *hostname* on port *port*.

If the **-O** option is not specified and one of the logging options is requested, then **pmcstat** will print a textual form of the logged events to the configured output file.

-P *event-spec*
Allocate a process mode sampling PMC measuring hardware events specified in *event-spec*.

-R *logfilename*
Perform offline analysis using sampling data in file *logfilename*.

-S *event-spec*
Allocate a system mode sampling PMC measuring hardware events specified in *event-spec*.

-T Use a top(1)-like mode for sampling PMCs. The following hotkeys can be used:

A Toggle symbol resolution
Ctrl+a Switch to accumulative mode
Ctrl+d
Switch to delta mode
f Represent the "f" cost under threshold as a dot (calltree only)
I Toggle showing offsets into symbols
m Merge PMCs
n Change view
p Show next PMC
q Quit
Space Pause

-U Toggle capturing user-space call traces while in kernel mode. The default is for sampling PMCs to capture user-space callchain information while in user-space mode, and kernel callchain information while in kernel mode.

-W Toggle logging the incremental counts seen by the threads of a tracked process each time they are scheduled on a CPU. This is an experimental feature intended to help analyse the dynamic behaviour of processes in the system. It may incur substantial overhead if enabled. The default is for this feature to be disabled.

-a *pathname*
Perform a symbol and file:line lookup for each address in each callgraph and save the output to *pathname*. Unlike **-m** that only resolves the first symbol in the graph, this resolves every node in the callgraph, or prints out addresses if no lookup information is available. This option requires the **-R** option to read in samples that were previously collected and saved with the **-O** option.

-c *cpu-spec*
Set the cpus for subsequent system mode PMCs specified on the command line to *cpu-spec*.

Argument *cpu-spec* is a comma separated list of CPU numbers, or the literal '*' denoting all available CPUs. The default is to allocate system mode PMCs on all available CPUs.

-d Toggle between process mode PMCs measuring events for the target process' current and future children or only measuring events for the target process. The default is to measure events for the target process alone. (it has to be passed in the command line prior to **-p**, **-s**, **-P**, or **-S**).

-e Specify that the gprof profile files will use a wide history counter. These files are produced in a format compatible with gprof(1). However, other tools that cannot fully parse a BSD-style gmon header might be unable to correctly parse these files.

-f *pluginopt*

Pass option string to the active plugin.
threshold=<float> do not display cost under specified value (Top).
skiplink=0|1 replace node with cost under threshold by a dot (Top).

-g Produce profiles in a format compatible with gprof(1). A separate profile file is generated for each executable object encountered. Profile files are placed in sub-directories named by their PMC event name.

-i *lwp* Filter on thread ID *lwp*, which you can get from ps(1) **-o** lwp.

-l *secs*

Set system-wide performance measurement duration for *secs* seconds. The argument *secs* may be a fractional value.

-m *pathname*

Print the sampled PCs with the name, the start and ending addresses of the function within they live. The *pathname* argument is mandatory and indicates where the information will be stored. If argument *pathname* is a "-" this information is sent to the output file specified by the **-o** option. This option requires the **-R** option to read in samples that were previously collected and saved with the **-O** option.

-n *rate*

Set the default sampling rate for subsequent sampling mode PMCs specified on the command line. The default is to configure PMCs to sample the CPU's instruction pointer every 65536 events.

-o *outputfile*

Send counter readings and textual representations of logged data to file *outputfile*. The default is

to send output to *stderr* when collecting live data and to *stdout* when processing a pre-existing logfile.

-p *event-spec*
Allocate a process mode counting PMC measuring hardware events specified in *event-spec*.

-q Decrease verbosity.

-r *fsroot*
Set the top of the filesystem hierarchy under which executables are located to argument *fsroot*. The default is /.

-s *event-spec*
Allocate a system mode counting PMC measuring hardware events specified in *event-spec*.

-t *process-spec*
Attach process mode PMCs to the processes named by argument *process-spec*. Argument *process-spec* may be a non-negative integer denoting a specific process id, or a regular expression for selecting processes based on their command names.

-u *event-spec*
Provide short description of event.

-v Increase verbosity.

-w *secs*
Print the values of all counting mode PMCs or sampling mode PMCs for top mode every *secs* seconds. The argument *secs* may be a fractional value. The default interval is 5 seconds.

-z *graphdepth*
When printing system-wide callgraphs, limit callgraphs to the depth specified by argument *graphdepth*.

If *command* is specified, it is executed using execvp(3).

EXAMPLES

To perform system-wide statistical sampling on an AMD Athlon CPU with samples taken every 32768 instruction retirals and data being sampled to file *sample.stat*, use:

```
pmcstat -O sample.stat -n 32768 -S k7-retired-instructions
```

To execute **firefox** and measure the number of data cache misses suffered by it and its children every 12 seconds on an AMD Athlon, use:

```
pmcstat -d -w 12 -p k7-dc-misses firefox
```

To measure instructions retired for all processes named "emacs" use:

```
pmcstat -t '^emacs$' -p instructions
```

To measure instructions retired for processes named "emacs" for a period of 10 seconds use:

```
pmcstat -t '^emacs$' -p instructions sleep 10
```

To count instruction tlb-misses on CPUs 0 and 2 on a Intel Pentium Pro/Pentium III SMP system use:

```
pmcstat -c 0,2 -s p6-itlb-miss
```

To collect profiling information for a specific process with pid 1234 based on instruction cache misses seen by it use:

```
pmcstat -P ic-misses -t 1234 -O /tmp/sample.out
```

To perform system-wide sampling on all configured processors based on processor instructions retired use:

```
pmcstat -S instructions -O /tmp/sample.out
```

If callgraph capture is not desired use:

```
pmcstat -N -S instructions -O /tmp/sample.out
```

To send the generated event log to a remote machine use:

```
pmcstat -S instructions -O remotehost:port
```

On the remote machine, the sample log can be collected using nc(1):

```
nc -l remotehost port > /tmp/sample.out
```

To generate gprof(1) compatible profiles from a sample file use:

```
pmcstat -R /tmp/sample.out -g
```

To print a system-wide profile with callgraphs to file *foo.graph* use:

```
pmcstat -R /tmp/sample.out -G foo.graph
```

DIAGNOSTICS

If option **-v** is specified, **pmcstat** may issue the following diagnostic messages:

#callchain/dubious-frames The number of callchain records that had an "impossible" value for a return address.

#exec handling errors The number of exec(2) events in the log file that named executables that could not be analyzed.

#exec/elf The number of exec(2) events that named ELF executables.

#exec/unknown The number of exec(2) events that named executables with unrecognized formats.

#samples/total The total number of samples in the log file.

#samples/unclaimed The number of samples that could not be correlated to a known executable object (i.e., to an executable, shared library, the kernel or the runtime loader).

#samples/unknown-object The number of samples that were associated with an executable with an unrecognized object format.

The **pmcstat** utility exits 0 on success, and >0 if an error occurs.

COMPATIBILITY

Due to the limitations of the *gmon.out* file format, gprof(1) compatible profiles generated by the **-g** option do not contain information about calls that cross executable boundaries. The generated *gmon.out* files are also only meaningful for native executables.

SEE ALSO

gprof(1), nc(1), execvp(3), pmc(3), pmclog(3), hwpmc(4), pmccontrol(8), sysctl(8)

HISTORY

The **pmcstat** utility first appeared in FreeBSD 6.0.

AUTHORS

Joseph Koshy <*jkoshy@FreeBSD.org*>

BUGS

The **pmcstat** utility cannot yet analyse hwpmc(4) logs generated by non-native architectures.