CrayPat - Cray X1 Performance Analysis Tool

Steve Kaufmann and **Bill Homer**, *Cray Inc.*

ABSTRACT: *The new Cray Performance Analysis Tool, CrayPat was designed to analyze and evaluate the performance of applications running on the Cray X1 system. It is a single point of entry tool that supports multiple performance experiments. With CrayPat, instrumenting an application requires only a link step with no required recompilation. The instrumented application is then executed on the Cray X1 like any normal application to produce a binary experiment data file. CrayPat then evaluates the contents of the data file and generates reports, the content and format of which can be customized. Experiment data can also be exported to alternate file formats for further processing.*

1 Introduction

The Cray X1 represents the convergence of the Cray T3E (MPP) and the traditional Cray parallel vector processors (PVP). It s a highly scalable, cache coherent, shared-memory multiprocessor, using powerful vector processors as its building blocks.

The core of the Cray X1 system is its multi-streaming processor (MSP), an eight-chip multi-chip module containing four processor chips and four custom cache chips. Each processor chip consists of a superscalar processor with a two-pipe vector unit, and the four cache chips implement a 2 megabyte cache that is shared by the four processors.

The resources that an application consumes are often an important development consideration. The amount of CPU time, memory, cache, network, or disk resources needs to be at least understood so that applications can take advantage of the full potential of the Cray X1.

The CrayPat performance analysis tool collects data at all levels of parallelism: the SSP-level, thread-level, and process level. CrayPat then helps developers locate opportunities for improvements in both performance and system resource usage.

2 CrayPat Overview

The CrayPat tool is a new tool for the Cray X1 platform. It was developed mindful of the performance analysis tools that preceded it on the Cray PVP and Cray MPP computer systems, and inherits some of their best features.

CrayPat provides a single point of entry into performance analysis of applications on the Cray X: it performs *experiments* on running applications.

An experiment is an evaluation of an application as it executes. The way that experiments work is determined both by how an application is instrumented, and how it is executed.

CrayPat is applied to applications for single or multiple PEs with shared memory (SM) or distributed memory (DM) design. CrayPat also supports threaded applications, and both MSP and SSP mode applications.

Preparing an application for performance analysis always takes the following steps:

- instrument the application
- execute the instrumented application
- obtain reports

Users interact with CrayPat through command-line utilities. Future releases of CrayPat will feature an optional GUI and text-based interactive PerfShell.

CrayPat provides a number of experiments that collect data in different ways. This way, if several experiments are applied to the same application, the bias implicit in any given experiment is rendered acceptable.

Instrumentation of an application is the first preparatory step required for performance evaluation. Instrumentation sets up the capture of software state, hardware state and time:

- Software state can include thread and call stack information or the actual parameter values passed into a function entry point.
- Hardware state can include the Program Counter (PC) or some Hardware Performance Counter (HWPC) event values.
- Time stamps are recorded in high resolution using the Real-Time Clock (RTC) and HWPC cycle counter.

Instrumentation uses the application itself to collect state and timing information. The instrumented application is executed in the same manner and in the same environment as the original application. It can be executed multiple times with varying data sets, each iteration producing a new experiment data file. The CrayPat reporting features can accept multiple experiment data files for a single application - the more material, the more complete and thorough the performance evaluation.

CrayPat does not require that applications or parts of applications be recompiled. A single link, managed by CrayPat, is all that is required. Link details are contained in a special ELF section in an executable file. CrayPat uses these details to create the link operands and the instrumented application. The original application is not changed.

An instrumented applications overhead varies depending on the type of experiment. The default CrayPat experiments minimize application user and/or system time requirements to typically less than 10 percent of the original application. However, more sophisticated experiments can more than double their requirements.

3 pat_hwpc

The `pat_hwpc` utility is a stand-alone utility that executes a given application, records specified HWPC events, and writes a summary report to standard output. (Alternately, it can be used to attach to a process that is already executing.) HWPC events and other timing information can also be saved to a file for later evaluation by the CrayPat report facility.

The `pat_hwpc` utility by default collects those HWPC events that maximize the usefulness of the resulting report. Derived statistics, such as average vector length, megaflopts, rates, and percentages are displayed. You can change the type of HWPC groups the utility collects. The following groups are predefined for the your convenience:

- `papi` (performance API)
- `scalar_detail`
- `scalar_stall`
- `vector_detail`
- `vector_stall`

These are not required - you can ask for any HWPC event, and override a predefined event.

4 Hardware Performance Counters

The Cray X1 computer system has an abundance of HWPC events. The processing chip (P chip) contains over 120 individual events available for counting. Some of these events include:

- cycles
- number of instructions dispatched
- number of instruction TLB misses
- number of A register instructions graduated
- number of vector TLB misses
- number of mispredicted branches
- number of elemental vector instructions graduated Instrument

The cache chip (E chip) can count over 60 events. Some of these events include:

- cache line allocations
- processor requests processed
- updates received

The memory chip (M chip) can count over 50 events. Some of theses events include:

- total requests to local memory
- local ecache requests to local memory
- invalidations sent to a single MSP

5 Experiment Types

The CrayPat tool can instrument an application in one of two ways:

- asynchronously
- synchronously

Asynchronous Experiments

If an application is instrumented for an asynchronous experiment, the nature of the experiment is selected at run-time. Asynchronous experiments are statistical: they sample the state of the application at given intervals. The interval can be a time interval (for example, every 10 milliseconds), or it can be an HWPC event that overflows a defined value.

For each interval, a sample of the state of the application is collected. The nature of the data collected is determined by the asynchronous experiment performed on the application. These experiments include:

- `profil`, `mprofil` - OS domain profiling experiments that capture absolute PC values
- `samp_pc_time`, `samp_pc_ovfl` - user domain sampling experiments that capture relative PC values

- `samp_cs_time`, `samp_cs_ovfl` - user domain sampling experiments that capture relative PC values and call stack traces
- `samp_heap_time`, `samp_heap_ovfl` - user domain sampling experiments that capture relative PC values and dynamic heap state
- `samp_ru_time`, `samp_ru_ovfl` - user domain sampling experiments that capture relative PC values and system resource usage

Profiling experiments produce the most compact experiment data files, and incur the least amount of run-time overhead.

Synchronous Experiments

If an application is instrumented for a synchronous experiment, function entry points are counted and recorded. At the time of instrumentation, you choose which function entry points to record. For each instrumented function entry that is executed during runtime, a tracing record is recorded in the experiment data file.

Some function entry points have predefined trace wrappers, all of which can be traced. However, if your desired function entry point does not have a predefined trace wrapper, than only function entry points written in C, C++, or Fortran can be traced.

6 Profiling

If no other experiment type has been specified with the `PAT_RT_EXPERIMENT` run-time environment variable, the `profil` and `mprofil` profiling experiments are the default asynchronous experiments done at run-time. The `profil` experiment is the default for a SSP mode application, and the `mprofil` experiment is the default for a MSP mode application (the PC for only SSP 0 is recorded by default).

Profiling experiments add the lowest overhead to the instrumented applications and generate the most compact experiment data file. These experiments sample the PC every 10 milliseconds. Program counter sampling is done by the operating system. It is not in the user domain. The profiling rate is currently fixed, but future enhancements will allow the rate to have a higher resolution and to be changed at run-time.

Having collected the state of the PC during run-time, the CrayPat report can show the distribution of the PCs, and map the addresses to the source code line corresponding to the address. Reports can also aggregate the PCs per code block or per function.

Profiling is a common performance evaluating process available on every UNIX system, including Cray PVP and Cray MPP computer systems.

7 Sampling

The sampling experiments (those titled beginning with “`samp`”) are similar to the profiling experiments, except the PC is recorded in the user domain with a timestamp. Because the sampling is conducted in the user domain, sampling has a larger run-time overhead than profiling. But because each PC is timestamped, more is known of the control flow of the application.

The interrupt interval is by timer-based by default, set at 10 milliseconds. This is the fastest rate at which sampling can occur using timers. The value of the rate is increased at run-time using the `PAT_RT_RATE` environment variable. Increasing the value of the rate results in fewer samples recorded over a given time span.

The interrupt interval can alternately be controlled by the overflow of a HWPC event. This applies to the sampling experiments that end with “`ovfl`”. For example, to get an increased rate, at the expense of higher runtime overhead, you can set the `PAT_RT_HWPC_OVERFLOW` environment variable to `P:0:0:100000`. The application at run-time samples the state every 100,000 cycles.

Sampling in the user domain also allows recording of HWPC events. You can use the `PAT_RT_HWPC` environment variable to record the state of user-specified HWPC events at each interval.

Sampling also supports recording other states of the software and hardware. The `samp_heap_time` and `samp_heap_ovfl` experiments sample the PC in addition to the internal state of the dynamic heap. The `samp_ru_time` and `samp_ru_ovfl` experiments sample the PC and the state of the resources consumed by the application during runtime. Some of the resource parameters collected include:

- page faults
- TLB misses
- number of system calls executed
- context switches

8 Tracing

The tracing experiments count the number of times an event occurs, specifically, the number of times a function entry point is entered and returned. Each time a traced function entry point is executed various state information is recorded. This includes:

- time function entered and returned
- value returned by function
- value of formal parameters to function
- call stack trace and call stack size
- HWPC event values when function entered and returned

A number of trace function *groups* are predefined. They represent function entry points that are related in function and application. These groups include:

- MPI, SHMEM, UPC, CAF
- Pthreads
- OpenMP
- Heap
- System Calls
- IO

Having these predefined function groups allows recording state unique to the group. This gives the CrayPat report facility more information for further detailed analysis.

For example, if function entry points related to the dynamic heap are traced, state information on the internals of the data structures that manage the heap are also recorded. If MPI tracing is selected, additional state such as rank and communicator information is recorded.

The overhead of tracing is the highest of all experiments, especially if the HWPC are activated. Depending on the application and the programming model used, the experiment data file created by an application instrumented for tracing is quite large (on the order of hundreds of megabytes).

9 Application Programming Interface

The CrayPat tool provides an Application Programming Interface (API) to provide you with finer control over the recording of the state during run-time. The API encompasses a number of functions that you can insert into your application source code. These functions are only activated in the instrumented program. The API facilitates recording similar state to tracing.

The API functions available include:

- `PAT_profiling_state` - activate or deactivate profiling
- `PAT_sampling_state` - activate or deactivate sampling
- `PAT_tracing_state` - activate or deactivate tracing
- `PAT_record_ssp` - activate or deactivate recording state on SSPs
- `PAT_trace_user_l` - record an event trace with a list of values
- `PAT_trace_user_v` - record an event trace with an array of values
- `PAT_trace_user` - record an event trace
- `PAT_trace_function` - activate or deactivate recording state of instrumented function entry point

All of the API functions are available to C an C++. All but `PAT_trace_user_l` are available to Fortran.

You can use the API to control the size of the experiment data file by turning the state of data recording off and on at key points. The API can also be used to limit recording state, especially HWPC events, in certain programming models.

For example, since function entry points written in assembly language can not be traced, place a `PAT_trace_user` API call before and after the reference to the function entry point to trace, as shown here:

```
PAT_trace_user (“foo”);
foo ();
PAT_trace_user (“”);
```

Similarly, individual loops can be bracketed and are reported by the initial label. See the end of this paper for examples using the API.

10 Instrumenting an Application

The CrayPat tool `pat_build` utility is used to instrument an application without altering the original application. And, except for any CrayPat API function calls added to the original application source code, you are not required to recompile or otherwise rebuild their application. The `pat_build` utility creates the instrumented application from the original relocatable files and original libraries. When `pat_build` creates the instrumented application, it includes CrayPats run-time libraries that facilitate the recording of run-time state.

By default (with no special options) `pat_build` creates an instrumented application that performs an asynchronous experiment. If the `PAT_RT_EXPERIMENT` environment variable is not set at run-time, the default experiment is profiling.

If the `-t` or `-T` options are specified, `pat_build` creates an instrumented application that performs a synchronous experiment (tracing of function entry points).

See the end of this paper for examples of instrumenting applications using `pat_build`.

11 Executing the Instrumented Application

Once an application has been instrumented, it is executed in the same way as the original. The instrumented application is used in all the same ways the original application was used, except now each time the instrumented application is executed, a binary experiment data file is created.

If the application is instrumented asynchronously, you can execute the instrumented application multiple times, each time as a different asynchronous (profiling or sampling) experiment. There is no need to reinstrument the application to change the type of asynchronous experiment.

If the application is instrumented as a tracing experiment, you must reinstrument the application to trace additional function entry points. However, if you want a subset of the instrumented function entry points, setting the `PAT_RT_FUNCTION_LIMITS` environment variable at run-time suppresses instrumented function entry points from being recorded in the experiment data file

There are a number of environment variables in the CrayPat library that allow you to control various run-time features. These variables let you fine tune the run-time aspects of data collection, in addition to other details. Some of the more commonly used run-time environment variables are:

- `PAT_RT_EXPERIMENT` - experiment type
- `PAT_RT_RATE` - profiling or sampling rate in microseconds
- `PAT_RT_HWPC` - HWPC events to record for each sample or trace
- `PAT_RT_HWPC_OVERFLOW` - sampling rate in terms of HWPC event overflow
- `PAT_RT_FUNCTION_LIMITS` - instrumented function entry points to suppress
- `PAT_RT_FUNCTION_MAX` - maximum number of trace records to record in data file
- `PAT_RT_RECORD_SSP` - SSPs to record for a MSP mode program (default is to record SSP 0 only)

12 Generating a Report

The CrayPat `pat_report` utility analyzes state and event data in the experiment data file, created as a result of executing the instrument application. It then produces a report from that file which you can customize for content and format.

A report consists of information that is provided for any experiment, such as the program name, its arguments, its environment, execution time, placement, etc., as well as performance data that is specific to the type of experiment that was performed.

The performance data is presented in one or more tables, each having one or more columns of data values and a column of labels, or key values. Like a spreadsheet pivot table, the report table can have a hierarchical organization, with each higher level showing totals of the values at the next lower level, and with the flexibility to specify any order for the hierarchy.

For example, a table might show flops data labeled by function name and SSP. In this case, there are two ways to organize the table. The can be lines with total flops for each function, each followed by four lines showing the flops contributed by the individual SSPs. Alternatively, there can be lines with total flops for each SSP, each followed by lines showing the flops contributed by each function. For either alternative, a grand total of flops for the whole program is also shown. The order of the labels specified to the `pat_report` utility determines which alternative is used, and both alternatives can be shown in the same report.

The `pat_report` utility can export the contents of the experiment data file to various formats, including XML and tab- or space-delimited flat data files, appropriate for spreadsheet program such as Excel.

In particular, the `pat_report` utility can aggregate data or keep it segregated by SSP, thread, and process. Reports display such detail as HWPC event values, call trees (caller-callee relationships), and special processing for the function groups mentioned earlier.

13 Examples

In many cases, it is adequate to instrument an application for profiling, which collects enough data to show time spent in each function or subroutine (optionally broken out by line number, process, SSP number, or any combination). The following examples show some of the other possibilities.

`pat_hwpc` *Example*

The overall performance of an application is measured without instrumenting it. For this SSP mode Fortran program, the default report shows a variety of performance metrics based on the HWPC events. Note that for MSP mode applications, and multi-process applications, the numbers can be shown by process and by SSP, as well as for the whole application.

API Example

In the application used for the HWPC example, the regions of interest are subroutines, each of which is called as the entire body of a timing loop. The simplest approach to performance measurement in such cases is to instrument it to trace the subroutines. Here, as an alternative, CrayPat API calls were inserted into the source of the program to measure the performance of the entire loops. The advantage of the API method is that it works even if the subroutine calls are inlined, and it has lower overhead. The overhead is lower because a pair of API calls were used to measurement each loop, instead of a pair of API calls to measurement each iteration in each loop.

Cody Style Example

In this Fortran application, a sum-of-neighbors algorithm is coded in five different styles, using loops, array syntax, intrinsics, etc. Each style is in a separate subroutine, and the program is compiled to prevent those subroutines from being inlined. The program is instrumented to trace those subroutines, and at run-time, enough HWPC event data is collected to calculate the metrics of interest: megaflops rates and average vector lengths. One of the arguments specifies the problem size, and each subroutine is called with a sequence of problem sizes. Reports are generated to show the overall performance for each subroutine, as well as its performance as a function of problem size.

IO Example

To investigate the IO of this C program, it is instrumented to trace the functions in the CrayPat `TraceIO`, `TraceFIO`, and `TraceAIO` list files. Several reports are generated to show the overall IO activity, the IO by number of bytes transferred, and where in the application call tree the IO is performed

DM Example

This two-process Fortran MPI benchmark calls a series of subroutines, each engages in a particular pattern of communication. It is instrumented to sample the call stack by time, and reports are generated to show the relative times consumed by those subroutines, the time spent in the MPI and other subroutines they call, and the load balance between the two processes.

14 Compatibility With Previous Cray Performance Tools

Previous Cray PVP and Cray MPP computer systems supported a number of performance analysis tools, all of which have been replaced by CrayPat.

Performance tools on the Cray PVP systems and the equivalent CrayPat experiments are as follows:

- ATExpert - tracing OpenMP, Pthreads, system calls
- Flowtrace - `samp_cs_time` and all tracing experiments
- `hpm` - `pat_hwpc` utility
- Jumptrace - tracing user functions
- Perftrace - `samp_pc_time` and all tracing experiments capturing HWPC event values
- `procstat` - `samp_ru_time`, tracing IO, heap, system calls
- `prof` - `profil`, `mprofil`, `samp_pc_time`

Performance tools on the Cray MPP systems and the equivalent CrayPat experiments are:

- MPP Apprentice - tracing MPI, SHMEM, UPC, CAF
- MPP pat - `profil`, `mprofil`, `samp_pc_time`, most tracing experiments

15 Future Development

The CrayPat tool is currently early in its development. Opportunities for a number of enhancements have already been identified, and work continues on a number of new features to make CrayPat as fully-featured and more user-friendly.

One of our top priorities is to optimize, where possible, the CrayPat run-time library to reduce the overhead of recording software and hardware state. This is especially true when accessing HWPC, as a context switch is done every time the HWPC events are acquired.

Also, new software and/or hardware state are being considered for recording. For example, the No Forward Progress interrupt is an event that signals a state of memory thrashing.

While collecting new state is important, the size of the experiment data file needs to be more compact. Especially for tracing experiments, the events recorded accumulate very quickly, resulting in very large data files. A long running application, if the sampling rate has not been reduced, also generates very large experiment data files. This is another area for improvement.

Report evaluation, analysis, and display are being reimplemented. The report component needs to increase the rate that it processes the experiment data file. It also requires enchancements in the area of processing the special function groups.

This initial release of the CrayPat tool does not officially support a GUI. An early GUI prototype, built on top of an interactive character-based interface called the PerfShell, was developed and has been evaluated by various internal users. Users will have access both to the PerfShell and the accompanying GUI when it becomes available. No specific Programming Environment release has been identified for GUI and PerfShell availability.

The Performance API (PAPI) provides a consistent interface across hardware platforms in accessing hardware performance counters. This project is managed by the Innovative Computer Laboratory at the University of Tennessee. The Cray X1 PAPI will be available for download from

`http://icl.cs.utk.edu/projects/papi/`

by June 2003.

16 Acknowledgements

The authors would like to thank colleagues of the Programming Environments and Testing group, and users in the Benchmarking and Applications group at Cray Inc. in Mendota Heights, MN for their contributions during the development, implementation, and testing of CrayPat.

17 Summary

CrayPat, a single point of entry into Cray X1 performance analysis, supports multiple levels of parallelization. It records various software and hardware state during run-time of instrumented applications. The instrumented application is executed multiple times and in the same manner as the original application. An API is supported to provide you finer control over what parts of an application the state is recorded, how much state is recorded, and when the state is recorded.

The resulting data can be viewed in a variety of ways, depending on the data collected and the programming model used by the application.

About the Authors

Steve Kaufmann and Bill Homer are Software Engineers in the Programming Environment group at Cray Inc. They can be reached at Cray Inc., 1340 Mendota Heights Road, Mendota Heights, MN 55122, or email at `sbk@cray.com` or `hom-er@cray.com`, respectively

`pat_hwpc` Example

```
$ pat_hwpc naskern
Command executed:  ./naskern

Exit status       0
Host name & type  sn801b crayx1 400 MHz
Operating system  UNICOS/mp 0.0.1_unreleased-irixdev-work_sv2-X1 05080403
Text page size    16 Mbytes
Other page size   16 Mbytes
Start time        Thu May  8 15:46:03 2003
End time          Thu May  8 15:46:06 2003
Elapsed time  3.147 seconds
User time     1.887 seconds   60%
System time   1.109 seconds   35%

Logical pe: 0  Node: 2  PID: 6831

  P counter data
CPU Seconds                            1.904457 sec
Cycles                 1381.532M/sec  2631068634 cycles
Instructions graduated  222.922M/sec   424545101 instr
Branches & Jumps         10.019M/sec    19080619 instr
Branches mispredicted     0.261M/sec      497848 misses    2.609%
Correctly predicted       9.758M/sec    18582771 misses   97.391%
Vector instructions      52.950M/sec   100841062 instr    23.753%
Scalar instructions     169.972M/sec   323704039 instr    76.247%
Vector ops             2412.304M/sec  4594129936 ops
Vector FP adds          585.222M/sec  1114530314 ops
Vector FP multiplies    557.634M/sec  1061989111 ops
Vector FP divides etc    12.950M/sec    24663146 ops
Vector FP misc           19.597M/sec    37321274 ops
Vector FP ops          1175.403M/sec  2238503845 ops      99.355%
Scalar FP ops             7.627M/sec    14524395 ops       0.645%
Total  FP ops          1183.029M/sec  2253028240 ops
FP ops per load                            1.148 flops/load
Scalar integer ops       15.748M/sec    29991061 ops
Scalar memory refs       20.200M/sec    38470129 refs      1.960%
Vector TLB misses         0.000M/sec         755 misses
Scalar TLB misses         0.001M/sec         992 misses
Instr  TLB misses         0.000M/sec         405 misses
Total  TLB misses         0.001M/sec        2152 misses
Dcache references        11.889M/sec    22642232 refs     58.857%
Dcache bypass refs        8.311M/sec    15827897 refs     41.143%
Dcache misses             8.880M/sec    16910661 misses   74.686%
Dcache hits               3.010M/sec     5731571 hits     25.314%
Vector integer adds       3.561M/sec     6781904 ops
Vector logical ops      152.406M/sec   290250040 ops
Vector shifts            57.743M/sec   109969332 ops
Vector int ops          213.710M/sec   407001276 ops
Vector loads            685.935M/sec  1306333248 refs
Vector stores           324.594M/sec   618174476 refs
Vector memory refs     1010.528M/sec  1924507724 refs     98.040%
Scalar memory refs       20.200M/sec    38470129 refs      1.960%
Total  memory refs     1030.728M/sec  1962977853 refs
Average vector length                     45.558
A-reg Instr              94.221M/sec   179439097 instr
Scalar FP Instr           7.627M/sec    14524395 instr
Syncs Instr              12.821M/sec    24417008 instr
Stall VLSU                2.334secs    933456649 clks
Stall VU                   4.842secs   1936861589 clks
Vector Load Alloc        685.275M/sec  1305076790 refs
Vector Load Index          0.008M/sec       15900 refs
Vector Load Stride       343.149M/sec   653513416 refs
Vector Store Alloc       323.717M/sec   616505004 refs
Vector Store Stride      185.204M/sec   352713815 refs
```

API Example

```
$ pat_build -w -T main -f naskern naskern+trace
$ PAT_RT_RECORD_SSP=0-3 PAT_RT_HWPC='P:*:0,P:25:1' aprun ./naskern+trace > naskern.stdout
$ pat_report -d mflops,P:0:0,vl -b function,ssp naskern+trace+118777tt.xf

Experiment:  trace

Experiment data file:
  /pesim/ptmp/homer/cpat_build/20/src/demos/msp/naskern+trace+118777tt.xf

Current path to data file:
  /ptmp/homer/cpat_build/20/src/demos/msp/naskern+trace+118777tt.xf

Original program:
  /ptmp/homer/cpat_build/20/src/demos/msp/naskern

Instrumented program:
  /pesim/ptmp/homer/cpat_build/20/src/demos/msp/./naskern+trace

Program arguments:  <none>

Traced functions:
 __pat_api_profiling_state .../src/lib/backend/api.c
 __pat_api_record_ssp      .../src/lib/backend/api.c
 __pat_api_sampling_state  .../src/lib/backend/api.c
 __pat_api_trace_function  .../src/lib/backend/api.c
 __pat_api_trace_user      .../src/lib/backend/api.c
 __pat_api_trace_user_v    .../src/lib/backend/api.c
 __pat_api_tracing_state   .../src/lib/backend/api.c
 _exit                     .../libc/src/proc/exit.c
 execve                    .../libc/src/proc/execve.c
 exit                      .../libc/src/gen/cuexit.c
 fork                      .../libc/src/proc/fork.c
 longjmp                   ==NA==
 main                      .../demos/msp/../naskern.f
 pthread_create            .../lib/libpthread/src/pt.c

Notes:
  Hardware performance counter values collected for:
  P:0:0
  P:1:0
  P:2:0
  P:3:0
  P:4:0
  P:5:0
  P:6:0
  P:7:0
  P:8:0
  P:9:0
  P:10:0
  P:11:0
  P:12:0
  P:13:0
  P:14:0
  P:15:0
  P:16:0
  P:17:0
  P:18:0
  P:19:0
  P:20:0
  P:21:0
  P:22:0
  P:23:0
  P:24:0
  P:25:1
  P:26:0
  P:27:0
  P:28:0
  P:29:0
  P:30:0
  P:31:0
```

```
Table 1:  -d mflops,P:0:0,vl
          -b pe,thread,function,ssp

   MFLOPS      P:0:0 Avg VL  PE
                               Thread
                                 Function
                                   SSP

  1202.39  749365025  45.55  Total

   1202.39  749365025  45.55  pe.0
                               thread.0
     7140.87   23496076  64.00  MXM
      1792.22   23404284  64.00  ssp.0
      1801.11   23288721  64.00  ssp.1
      1787.55   23465360  64.00  ssp.2
      1785.22   23496076  64.00  ssp.3

     3726.23   31869814  59.95  EMIT
       945.09   31592816  59.93  ssp.0
       931.04   31826946  59.96  ssp.1
       929.85   31869814  59.96  ssp.2
       937.85   31593568  59.96  ssp.3

     3022.06   32603389  37.76  VPENTA
       769.09   32027748  37.76  ssp.0
       767.22   32105889  37.76  ssp.1
       764.02   32240547  37.76  ssp.2
       755.51   32603389  37.76  ssp.3

     1620.50   78313597  29.04  BTRIX
       829.83   77497992  28.55  ssp.0
       266.44   78313597  29.49  ssp.1
       266.81   78203738  29.49  ssp.2
       268.21   77796118  29.49  ssp.3

     1110.96   78313210  61.33  CHOLSKY
       288.59   77609037  61.03  ssp.0
       279.17   77280659  61.53  ssp.1
       277.16   77839894  61.53  ssp.2
       274.00   78313210  61.27  ssp.3

      788.12  252811289  42.03  CFFT2D
       197.91  251749972  42.04  ssp.0
       197.42  252290199  42.03  ssp.1
       197.34  252392686  42.03  ssp.2
       197.01  252811289  42.03  ssp.3

      761.52  127927386  56.37  GMTRY
       580.09  127927386  57.84  ssp.0
        96.69   80004853  49.43  ssp.1
        94.69   81694289  49.43  ssp.2
        99.04   78131379  49.43  ssp.3

       43.46  123964455  31.88  main
        43.35  123964455  27.35  ssp.0
         0.07   63104348  35.01  ssp.1
         0.07   61799749  35.59  ssp.2
         0.07   60095489  35.01  ssp.3

        0.00      65809   0.25  (N/A)
         0.00      25755   0.25  ssp.0
         0.00      65809   0.00  ssp.1
         0.00      42006   0.00  ssp.2
         0.00      65767   0.00  ssp.3


Elapsed time in seconds for processes

  Process   118777: date: begin: Wed Apr 30 15:55:12 2003
                          end:   Wed Apr 30 15:55:14 2003
                          2.206793

                   utime: 1.836220
                   stime: 0.256458


Elapsed time in real-time clocks for threads

  pe/thread     0/0:     215104660 rtc (2.1510466 sec)
```

Coding Style Example

```
$ pat_build -w -T '/^count/ ising ising+trace
$ PAT_RT_RECORD_SSP=0-3 PAT_RT_HWPC='P:*:0,P:25:1' ising 8
$ pat_report -d mflops,flops,vl -b function ising+trace+5376tt.xf
Experiment:  trace

Experiment data file:
  /pesim/ptmp/homer/cpat_build/20/src/demos/msp/ising+trace+5376tt.xf

Current path to data file:
  /ptmp/homer/cpat_build/20/src/demos/msp/ising+trace+5376tt.xf

Original program:
  /ptmp/homer/cpat_build/20/src/demos/msp/ising

Instrumented program:
  /pesim/ptmp/homer/cpat_build/20/src/demos/msp/./ising+trace

Program arguments:  <none>

Traced functions:
 _exit          .../libc/src/proc/exit.c
 count1_        .../demos/msp/../ising.f90
 count2_        .../demos/msp/../ising.f90
 count3_        .../demos/msp/../ising.f90
 count4_        .../demos/msp/../ising.f90
 count5_        .../demos/msp/../ising.f90
 execve         .../libc/src/proc/execve.c
 fork           .../libc/src/proc/fork.c
 longjmp        ==NA==
 main           .../demos/msp/../ising.f90
 pthread_create .../lib/libpthread/src/pt.c
 try_           .../demos/msp/../ising.f90

Notes:
  Hardware performance counter values collected for:
  P:0:0
  P:1:0
  P:2:0
  P:3:0
  P:4:0
  P:5:0
  P:6:0
  P:7:0
  P:8:0
  P:9:0
  P:10:0
  P:11:0
  P:12:0
  P:13:0
  P:14:0
  P:15:0
  P:16:0
  P:17:0
  P:18:0
  P:19:0
  P:20:0
  P:21:0
  P:22:0
  P:23:0
  P:24:0
  P:25:1
  P:26:0
  P:27:0
  P:28:0
  P:29:0
  P:30:0
  P:31:0



Table 1:  -d mflops,flops,vl
          -b function

 MFLOPS |    FLOPs | Avg.VL |Function

  68.76 |319243515 |  54.09 |Total
|---------------------------------------------------------------------
| 503.54 | 47070782 |  55.90 |count3_
| 491.14 | 47017686 |  55.90 |count2_
| 480.29 | 46495370 |  39.08 |count5_
| 395.82 | 46495250 |  61.38 |count4_
| 342.03 | 56931216 |  56.55 |count1_
|  26.64 |  9415385 |  55.78 |main
|  17.70 | 65817826 |  55.84 |try_
|   0.00 |        0 |   0.50 |(N/A)
|   0.00 |        0 |   6.37 |_exit
|=====================================================================

Elapsed time in seconds for processes

  Process     5376: date: begin: Fri May  2 14:56:15 2003
                          end:   Fri May  2 14:56:18 2003
                          2.829560

                   utime: 1.479112
                   stime: 0.631095


Elapsed time in real-time clocks for threads

  pe/thread     0/0:     268189105 rtc (2.68189105 sec)

$ pat_report -d flops,mflops,vl -b ssp,function,argument2 ising+trace+5376tt.xf
...
Table 1:  -d flops,mflops,vl
          -b ssp,function,argument2

    FLOPs | MFLOPS | Avg.VL |SSP
                             |Function
                              |Arg#2

319243515 | 198.37 |  54.09 |Total
|---------------------------------------------------------------------
|137055978 |  85.16 |  54.46 |ssp.0
||--------------------------------------------------------------------
|| 65817826 |  48.93 |  55.84 |try_
||          |        |        | (0)
|| 14435740 | 350.23 |  56.50 |count1_
```

```
|||-----------------------------------------------------------------
|||  3178496 | 252.64 |  63.99 |(128)
|||  2798436 | 432.74 |  61.23 |(122)
|||  2368256 | 420.94 |  58.46 |(116)
|||  2020097 | 405.29 |  55.24 |(109)
|||  1644036 | 397.37 |  52.02 |(102)
|||  1184960 | 358.33 |  47.41 |(92)
|||   840051 | 311.96 |  42.35 |(81)
|||   401408 | 282.31 |  63.91 |(64)
|||=================================================================
|| 11922555 | 518.56 |  39.05 |count5_
|||-----------------------------------------------------------------
|||  2670086 | 485.20 |  45.06 |(128)
|||  2277014 | 631.05 |  42.95 |(122)
|||  1991030 | 570.99 |  40.83 |(116)
|||  1639257 | 620.14 |  38.36 |(109)
|||  1331514 | 457.63 |  35.88 |(102)
|||   998390 | 472.96 |  32.35 |(92)
|||   675546 | 344.56 |  28.47 |(81)
|||   339718 | 441.81 |  34.71 |(64)
|||=================================================================
|| 11900617 | 522.18 |  55.83 |count3_
|||-----------------------------------------------------------------
|||  2623744 | 563.46 |  63.97 |(128)
|||  2309260 | 575.23 |  60.97 |(122)
|||  1953232 | 563.86 |  57.97 |(116)
|||  1665388 | 510.29 |  54.47 |(109)
|||  1354440 | 498.71 |  50.97 |(102)
|||   975088 | 432.93 |  45.97 |(92)
|||   690505 | 413.19 |  40.47 |(81)
|||   328960 | 437.22 |  63.79 |(64)
|||=================================================================
|| 11887457 | 506.21 |  55.83 |count2_
|||-----------------------------------------------------------------
|||  2621699 | 531.82 |  63.97 |(128)
|||  2307279 | 573.44 |  60.97 |(122)
|||  1951379 | 554.93 |  57.97 |(116)
|||  1663599 | 510.06 |  54.47 |(109)
|||  1352779 | 460.07 |  50.97 |(102)
|||   973619 | 428.42 |  45.97 |(92)
|||   689164 | 352.33 |  40.46 |(81)
|||   327939 | 562.52 |  63.79 |(64)
|||=================================================================
|| 11676398 | 403.69 |  61.36 |count4_
|||-----------------------------------------------------------------
|||  2596864 | 283.23 |  63.73 |(128)
|||  2262246 | 491.40 |  62.90 |(122)
|||  1930936 | 436.86 |  62.01 |(116)
|||  1618705 | 452.79 |  60.89 |(109)
|||  1321206 | 458.48 |  59.69 |(102)
|||   960664 | 455.39 |  57.69 |(92)
|||   664241 | 431.39 |  55.36 |(81)
|||   321536 | 513.99 |  63.31 |(64)
|||=================================================================
||  9415385 |  75.47 |  55.78 |main
||          |        |        | (0x400040fffda0)
||        0 |   0.00 |   0.50 |(N/A)
||          |        |        | (0)
||        0 |   0.00 |   6.37 |_exit
||          |        |        | (0)
```

```
||===================================================================
| 60394975 |  58.62 |  53.92 |ssp.1
||-------------------------------------------------------------------
|| 14085574 | 336.77 |  56.57 |count1_
||          |        |        | (0)
|| 11681199 | 494.33 |  55.93 |count3_
||          |        |        | (0)
|| 11667759 | 479.40 |  55.92 |count2_
||          |        |        | (0)
|| 11589576 | 394.71 |  61.38 |count4_
||          |        |        | (0)
|| 11370867 | 455.75 |  39.08 |count5_
||          |        |        | (0)
||        0 |   0.00 |   0.00 |try_
||          |        |        | (0)
||        0 |   0.00 |   0.00 |(N/A)
||          |        |        | (0)
||        0 |   0.00 |   0.00 |main
||          |        |        | (0)
||        0 |   0.00 |   0.00 |_exit
||          |        |        | (0)
||===================================================================
| 61402911 |  60.79 |  53.81 |ssp.2
||-------------------------------------------------------------------
|| 14238198 | 341.43 |  56.57 |count1_
||          |        |        | (0)
|| 11922531 | 487.53 |  39.07 |count5_
||          |        |        | (0)
|| 11807767 | 498.18 |  55.93 |count3_
||          |        |        | (0)
|| 11794711 | 490.83 |  55.93 |count2_
||          |        |        | (0)
|| 11639704 | 394.12 |  61.39 |count4_
||          |        |        | (0)
||        0 |   0.00 |   0.00 |try_
||          |        |        | (0)
||        0 |   0.00 |   0.00 |(N/A)
||          |        |        | (0)
||        0 |   0.00 |   0.00 |main
||          |        |        | (0)
||        0 |   0.00 |   0.00 |_exit
||          |        |        | (0)
||===================================================================
| 60389651 |  60.80 |  53.95 |ssp.3
||-------------------------------------------------------------------
|| 14171704 | 339.79 |  56.57 |count1_
||          |        |        | (0)
|| 11681199 | 500.13 |  55.93 |count3_
||          |        |        | (0)
|| 11667759 | 488.61 |  55.92 |count2_
||          |        |        | (0)
|| 11589572 | 390.91 |  61.38 |count4_
||          |        |        | (0)
|| 11279417 | 462.09 |  39.12 |count5_
||          |        |        | (0)
||        0 |   0.00 |   0.00 |try_
||          |        |        | (0)
||        0 |   0.00 |   0.00 |(N/A)
||          |        |        | (0)
||        0 |   0.00 |   0.00 |main
||          |         |         | (0)
||        0 |    0.00 |    0.00 |_exit
||          |         |         | (0)
|======================================================================

Elapsed time in seconds for processes

  Process     5376: date: begin: Fri May  2 14:56:15 2003
                          end:   Fri May  2 14:56:18 2003
                          2.829560

                   utime: 1.479112
                   stime: 0.631095


Elapsed time in real-time clocks for threads

  pe/thread     0/0:     268189105 rtc (2.68189105 sec)
```

IO Example

```
$ pat_build -t $PAT_SV2/lib/TraceIO -t $PAT_SV2/lib/TraceAIO -t $PAT_SV2/lib/TraceFio equake
equake+tio
$ aprun equake+tio < equake.5
$ pat_report -d time%,time,traces,io -b function equake+tio+58269t.xf
Experiment:  trace

Experiment data file:
  /hosts/da/ptmp/sbk/127124847-ssp/trace-sn702-31010/equake+tio+58269t.xf

Current path to data file:
  /ptmp/sbk/127124847-ssp/trace-sn702-31010/equake+tio+58269t.xf

Original program:
  /users/sbk/SV2/PAT-extra/tests/Source/ssp/equake

Instrumented program:
  /hosts/da/ptmp/sbk/127124847-ssp/equake+tio

Program arguments:  <none>

Traced functions:
 _exit        .../libc/src/proc/exit.c
 acquire_lock .../src/mp/sv2/libmutexs.c
 close        .../libc/src/sys/close.c
 execve       .../libc/src/proc/execve.c
 init_lock    .../libc/src/mp/libmutexc.c
 lseek        .../libc/src/sys/lseek.c
 main         .../Source/ssp/../equake.c
 open         .../libc/src/sys/open.c
 read         .../libc/src/sys/read.c
 release_lock .../libc/src/mp/libmutexc.c
 spin_lock    .../src/mp/sv2/libmutexs.c
 stat_lock    .../libc/src/mp/libmutexc.c
 write        .../libc/src/sys/write.c

Notes:
  Hardware performance counter values collected for:
   P:0:0
   P:1:0
   P:2:0
   P:16:0
   P:17:0



Table 1:  -d time%,time,traces,io
          -b function

 Time% |        Time | Traces |  Input | Output |Function

100.0% |  310.319100 |    555 |1648240 |   3106 |Total
|----------------------------------------------------------------------
| 99.1% |  307.470239 |      1 |     -- |     -- |main
|  0.6% |    2.013595 |    403 |1648240 |     -- |read
|  0.3% |    0.813111 |    150 |     -- |   3106 |write
|  0.0% |    0.019983 |      0 |     -- |     -- |(N/A)
|  0.0% |    0.002172 |      1 |     -- |     -- |_exit
|======================================================================
Elapsed time in seconds for processes

  Process    58269: date: begin: Wed May  7 17:14:50 2003
                          end:   Wed May  7 17:20:01 2003
                          311.266656

                   utime: 299.462732
                   stime: 3.596126


Elapsed time in real-time clocks for threads

  pe/thread     0/0:   31032347568 rtc (310.32347568 sec)


$ pat_report -d time,traces -b function,ar1:return equake+tio+58269t.xf
...
Table 1:  -d time,traces
          -b function,ar1:return

        Time | Traces |Function
                      |Arg#1:Return

  310.319100 |    555 |Total
|--------------------------------------------------------------------
|  307.470239 |      1 |main
|             |        | (1):0
|    2.013595 |    403 |read
||-------------------------------------------------------------------
||    2.009012 |    402 |(0):4096
||    0.004583 |      1 |(0):1648
||===================================================================
|    0.813111 |    150 |write
||-------------------------------------------------------------------
||    0.214141 |     27 |(2):10
||    0.116898 |     28 |(2):4
||    0.089263 |     17 |(2):8
||    0.049679 |     10 |(1):69
||    0.047720 |      8 |(1):68
||    0.042913 |      7 |(1):67
||    0.040001 |      8 |(2):1
||    0.032051 |      6 |(2):43
||    0.026901 |      5 |(2):5
||    0.025619 |      6 |(2):3
||    0.022529 |      5 |(2):24
||    0.017038 |      4 |(2):9
||    0.015015 |      2 |(1):70
||    0.012926 |      3 |(2):16
||    0.012023 |      3 |(2):2
||    0.008869 |      2 |(2):13
||    0.005761 |      1 |(2):19
||    0.004934 |      1 |(2):18
||    0.004400 |      1 |(2):35
||    0.004241 |      1 |(2):17
||    0.004160 |      1 |(2):22
||    0.004092 |      1 |(2):6
||    0.004019 |      1 |(2):27
||    0.004000 |      1 |(2):20
||    0.003919 |      1 |(2):36
```

```
||=====================================================================
|    0.019983 |      0 |(N/A)
|             |        | (0):0
|    0.002172 |      1 |_exit
|             |        | (0):0
|======================================================================

Elapsed time in seconds for processes

  Process    58269: date: begin: Wed May  7 17:14:50 2003
                          end:   Wed May  7 17:20:01 2003
                          311.266656

                   utime: 299.462732
                   stime: 3.596126


Elapsed time in real-time clocks for threads

  pe/thread     0/0:   31032347568 rtc (310.32347568 sec)


$ pat_report -d traces -b calltree equake+tio+58269t.xf
...
Table 1:  -d traces
          -b calltree

 Traces |Calltree

    555 |Total
|----------------------------------------------------------------------
|    553 |main
||---------------------------------------------------------------------
||    405 |arch_init
||        | readpackfile
||||-------------------------------------------------------------------
||||    399 |fscanf
||||        | _doscan
||||||-----------------------------------------------------------------
||||||    296 |number
||||||        | _filbuf
||||||        |  read
||||||    103 |_filbuf
||||||        | read
||||||=================================================================
||||      6 |fprintf
||||        | _doprnt
||||        |  _dowrite
||||        |   fwrite
||||        |    write
||||===================================================================
||    117 |fprintf
||        | _doprnt
||||-------------------------------------------------------------------
||||     82 |_dowrite
||||        | fwrite
||||||-----------------------------------------------------------------
||||||     65 |write
||||||     17 |_xflsbuf
||||||        | write
```

```
|||||=====================================================================
||||     35 |_xflsbuf
||||        | write
||||======================================================================
||     27 |fflush
||        | _xflsbuf
||        |  write
||      4 |arch_readnodevector
||        | fscanf
||        |  _doscan
||        |   _filbuf
||        |    read
||========================================================================
|      1 |<no caller>
|        | main
|      1 |exit
|        | _exit
|      0 |(N/A)(exclusive)
|=========================================================================

Elapsed time in seconds for processes

  Process    58269: date: begin: Wed May  7 17:14:50 2003
                          end:   Wed May  7 17:20:01 2003
                          311.266656

                   utime: 299.462732
                   stime: 3.596126


Elapsed time in real-time clocks for threads

  pe/thread     0/0:   31032347568 rtc (310.32347568 sec)

$ pat_report -d traces -b function,callers equake+tio+58269t.xf
...
Table 1:  -d traces
          -b function,callers

 Traces |Function
        |Caller

    555 |Total
|-------------------------------------------------------------------------
|    403 |read
|        | _filbuf
|||-----------------------------------------------------------------------
|||    296 |number
|||        | _doscan
|||        |  fscanf
|||        |   readpackfile
|||        |    arch_init
|||        |     main
|||    107 |_doscan
|||        | fscanf
|||||---------------------------------------------------------------------
|||||    103 |readpackfile
|||||        | arch_init
|||||        |  main
|||||      4 |arch_readnodevector
```

```
|||||        | main
|||====================================================================
|    150 |write
||-------------------------------------------------------------------
||     79 |_xflsbuf
|||------------------------------------------------------------------
|||     35 |_doprnt
|||        | fprintf
|||        |  main
|||     27 |fflush
|||        | main
|||     17 |fwrite
|||        | _dowrite
|||        |  _doprnt
|||        |   fprintf
|||        |    main
|||==================================================================
||     71 |fwrite
||        | _dowrite
||        |  _doprnt
||        |   fprintf
||||||---------------------------------------------------------------
||||||     65 |main
||||||      6 |readpackfile
||||||        | arch_init
||||||        |  main
||===================================================================
|      1 |main
|        | <no caller>
|      1 |_exit
|        | exit
|      0 |(N/A)
|====================================================================

Elapsed time in seconds for processes

  Process    58269: date: begin: Wed May  7 17:14:50 2003
                          end:   Wed May  7 17:20:01 2003
                          311.266656

                   utime: 299.462732
                   stime: 3.596126


Elapsed time in real-time clocks for threads

  pe/thread     0/0:   31032347568 rtc (310.32347568 sec)
```

DM Example

```
$ pat_build PMB-MPI1 PMB-MPI1+async
$ PAT_RT_EXPERIMENT=samp_cs_time mpirun -np 2 ./PMB-MPI1+async
$ pat_report -b ct1,ct2 -s percent=relative PMB-MPI1+async+26200sd.xf
Experiment:  samp_cs_time

Experiment data file:
  /pesim/ptmp/homer/Pallas/pmb/src/PMB-MPI1+async+26200sd.xf

Current path to data file:
  /ptmp/homer/Pallas/pmb/src/PMB-MPI1+async+26200sd.xf

Original program:
  /ptmp/homer/Pallas/pmb/src/PMB-MPI1

Instrumented program:
  /pesim/ptmp/homer/Pallas/pmb/src/./PMB-MPI1+async

Program arguments:  <none>

Traced functions:
 _exit  .../libc/src/proc/exit.c
 execve .../libc/src/proc/execve.c
 main   .../Pallas/pmb/src/pmb.c


Table 1:  -d samples%,cum_samples%,samples,counters
          -b ct1,ct2

 Samp% | Cum.Samp% | Samp |Calltree#1
                           |Calltree#2

100.0% |    100.0% |  593 |Total
|----------------------------------------------------------------------
| 99.7% |     99.7% |  591 |main
||---------------------------------------------------------------------
|| 17.6% |     17.6% |  104 |Init_Buffers
|| 12.2% |     29.8% |   72 |Exchange
||  8.8% |     38.6% |   52 |Allgatherv
||  8.3% |     46.9% |   49 |PingPong
||  7.1% |     54.0% |   42 |PingPing
||  6.9% |     60.9% |   41 |Sendrecv
||  6.9% |     67.9% |   41 |Allgather
||  6.4% |     74.3% |   38 |Reduce_scatter
||  6.4% |     80.7% |   38 |Allreduce
||  5.6% |     86.3% |   33 |Alltoall
||  3.7% |     90.0% |   22 |Output
||  3.2% |     93.2% |   19 |Bcast
||  2.9% |     96.1% |   17 |Reduce
||  2.7% |     98.8% |   16 |Warm_Up
||  0.7% |     99.5% |    4 |(N/A)
||  0.3% |     99.8% |    2 |Basic_Input
||  0.2% |    100.0% |    1 |Init_Communicator
||=====================================================================
|  0.3% |    100.0% |    2 |exit
|       |           |      | __dm_exit_barrier
|======================================================================

Elapsed time in seconds for processes
```

```
  Process    26197: date: begin: Mon May  5 18:21:21 2003
                          end:   Mon May  5 18:21:27 2003
                          6.034060

                   utime: 5.063177
                   stime: 0.797301

  Process    26200: date: begin: Mon May  5 18:21:21 2003
                          end:   Mon May  5 18:21:27 2003
                          6.052026

                   utime: 4.673398
                   stime: 1.157005


Elapsed time in real-time clocks for threads

  pe/thread     0/0:     571676677 rtc (5.71676677 sec)
  pe/thread     1/0:     575910908 rtc (5.75910908 sec)

$ pat_report -d samp%,samp -b ct -s percent=relative PMB-MPI1+async+26200sd.xf
...
Table 1:  -d samp%,samp
          -b ct

 Samp% | Samp |Calltree

100.0% |  593 |Total
|---------------------------------------------------------------------
| 99.7% |  591 |main
||--------------------------------------------------------------------
|| 17.6% |  104 |Init_Buffers
||       |      | set_buf
||       |      |  ass_buf
|| 12.5% |   74 |Exchange
|||-------------------------------------------------------------------
||| 56.8% |   42 |MPI_Recv
||||------------------------------------------------------------------
|||| 97.6% |   41 |MPI_CRAY_recv_wait
|||||-----------------------------------------------------------------
||||| 51.2% |   21 |MPI_CRAY_recv_wait(exclusive)
||||| 36.6% |   15 |bcopy
|||||       |      | __bcopy_wordstrm
||||| 12.2% |    5 |__bcopy_prv
|||||=================================================================
||||  2.4% |    1 |MPI_Recv(exclusive)
||||==================================================================
||| 32.4% |   24 |MPI_Isend
||||------------------------------------------------------------------
|||| 79.2% |   19 |MPI_CRAY_request_send
||||  8.3% |    2 |MPI_CRAY_progress_incoming
||||  8.3% |    2 |MPI_Isend(exclusive)
||||  4.2% |    1 |MPI_CRAY_progress_ack
||||==================================================================
|||  8.1% |    6 |MPI_Waitall
||||------------------------------------------------------------------
|||| 33.3% |    2 |MPI_Waitall(exclusive)
|||| 33.3% |    2 |MPI_CRAY_progress_ack
|||| 16.7% |    1 |MPI_CRAY_type_free
```

```
|||| 16.7% |     1 |MPI_CRAY_progress_incoming
||||==================================================================
|||  2.7% |     2 |Exchange(exclusive)
|||==================================================================
||  8.8% |    52 |Allgatherv
|||------------------------------------------------------------------
||| 98.1% |    51 |MPI_Allgatherv
||||-----------------------------------------------------------------
|||| 62.7% |    32 |MPI_CRAY_bcast
|||||----------------------------------------------------------------
||||| 93.8% |    30 |MPI_CRAY_bcast(exclusive)
|||||  3.1% |     1 |bcopy
|||||       |       | __bcopy_wordstrm
|||||  3.1% |     1 |__bcopy_prv
|||||================================================================
|||| 37.3% |    19 |MPI_CRAY_gatherv
|||||----------------------------------------------------------------
||||| 57.9% |    11 |MPI_CRAY_gatherv(exclusive)
||||| 42.1% |     8 |bcopy
||||||---------------------------------------------------------------
|||||| 87.5% |     7 |__bcopy_wordstrm
|||||| 12.5% |     1 |bcopy(exclusive)
||||=================================================================
|||  1.9% |     1 |MPI_Barrier
|||==================================================================
||  8.3% |    49 |PingPong
|||------------------------------------------------------------------
||| 57.1% |    28 |MPI_Recv
|||       |       | MPI_CRAY_recv_wait
|||||----------------------------------------------------------------
||||| 71.4% |    20 |MPI_CRAY_recv_wait(exclusive)
||||| 28.6% |     8 |bcopy
|||||       |       | __bcopy_wordstrm
|||||================================================================
||| 40.8% |    20 |MPI_Send
|||  2.0% |     1 |MPI_Barrier
|||==================================================================
||  7.1% |    42 |PingPing
|||------------------------------------------------------------------
||| 81.0% |    34 |MPI_Recv
|||       |       | MPI_CRAY_recv_wait
|||||----------------------------------------------------------------
||||| 73.5% |    25 |MPI_CRAY_recv_wait(exclusive)
||||| 26.5% |     9 |bcopy
|||||       |       | __bcopy_wordstrm
|||||================================================================
||| 11.9% |     5 |MPI_Wait
||||-----------------------------------------------------------------
|||| 80.0% |     4 |MPI_CRAY_progress_ack
|||| 20.0% |     1 |MPI_Wait(exclusive)
||||=================================================================
|||  7.1% |     3 |MPI_Isend
||||-----------------------------------------------------------------
|||| 66.7% |     2 |MPI_CRAY_request_send
|||| 33.3% |     1 |MPI_Isend(exclusive)
|||==================================================================
||  6.9% |    41 |Sendrecv
|||------------------------------------------------------------------
||| 95.1% |    39 |MPI_Sendrecv
||||-----------------------------------------------------------------
```

```
|||| 48.7% |   19 |MPI_CRAY_send_wait
|||||---------------------------------------------------------------
||||| 84.2% |   16 |MPI_CRAY_progress_incoming
||||||--------------------------------------------------------------
|||||| 50.0% |    8 |MPI_CRAY_progress_incoming(exclusive)
|||||| 50.0% |    8 |bcopy
||||||       |      | __bcopy_wordstrm
||||||==============================================================
||||| 15.8% |    3 |MPI_CRAY_send_wait(exclusive)
|||||===============================================================
|||| 17.9% |    7 |MPI_Sendrecv(exclusive)
|||| 17.9% |    7 |MPI_CRAY_request_recv
|||| 12.8% |    5 |MPI_CRAY_progress_incoming
|||||---------------------------------------------------------------
||||| 80.0% |    4 |MPI_CRAY_progress_incoming(exclusive)
||||| 20.0% |    1 |bcopy
|||||===============================================================
||||  2.6% |    1 |MPI_CRAY_type_free
||||================================================================
|||  4.9% |    2 |MPI_Barrier
|||=================================================================
||  6.9% |   41 |Allgather
|||-----------------------------------------------------------------
||| 97.6% |   40 |MPI_Allgather
||||----------------------------------------------------------------
|||| 60.0% |   24 |MPI_CRAY_gather
|||||---------------------------------------------------------------
||||| 66.7% |   16 |MPI_CRAY_gather(exclusive)
||||| 25.0% |    6 |bcopy
|||||       |      | __bcopy_wordstrm
|||||  8.3% |    2 |__bcopy_prv
|||||===============================================================
|||| 40.0% |   16 |MPI_CRAY_bcast
|||||---------------------------------------------------------------
||||| 75.0% |   12 |MPI_CRAY_bcast(exclusive)
||||| 25.0% |    4 |bcopy
|||||       |      | __bcopy_wordstrm
||||================================================================
|||  2.4% |    1 |MPI_Barrier
|||=================================================================
||  6.4% |   38 |Reduce_scatter
|||-----------------------------------------------------------------
||| 97.4% |   37 |MPI_Reduce_scatter
||||----------------------------------------------------------------
|||| 51.4% |   19 |MPI_CRAY_reduce
|||||---------------------------------------------------------------
||||| 52.6% |   10 |MPI_CRAY_reduce(exclusive)
||||| 42.1% |    8 |MPI_CRAY_op_strm_func_sum3
|||||  5.3% |    1 |MPI_CRAY_op_func_sum3
|||||===============================================================
|||| 43.2% |   16 |MPI_CRAY_scatterv
|||||---------------------------------------------------------------
||||| 56.2% |    9 |MPI_CRAY_scatterv(exclusive)
||||| 31.2% |    5 |bcopy
|||||       |      | __bcopy_wordstrm
||||| 12.5% |    2 |__bcopy_prv
|||||===============================================================
||||  5.4% |    2 |MPI_Reduce_scatter(exclusive)
||||================================================================
|||  2.6% |    1 |MPI_Barrier
```

```
|||==================================================================
||  6.4% |   38 |Allreduce
||       |      | MPI_Allreduce
||||-----------------------------------------------------------------
|||| 76.3% |   29 |MPI_CRAY_reduce
|||||----------------------------------------------------------------
||||| 72.4% |   21 |MPI_CRAY_reduce(exclusive)
||||| 27.6% |    8 |MPI_CRAY_op_strm_func_sum3
|||||================================================================
|||| 23.7% |    9 |MPI_CRAY_bcast
|||||----------------------------------------------------------------
||||| 88.9% |    8 |MPI_CRAY_bcast(exclusive)
||||| 11.1% |    1 |__bcopy_prv
||||=================================================================
||  5.8% |   34 |Alltoall
|||------------------------------------------------------------------
||| 94.1% |   32 |MPI_Alltoall
||||-----------------------------------------------------------------
|||| 50.0% |   16 |MPI_Alltoall(exclusive)
|||| 43.8% |   14 |bcopy
|||||----------------------------------------------------------------
||||| 71.4% |   10 |__bcopy_wordstrm
||||| 28.6% |    4 |bcopy(exclusive)
|||||================================================================
||||  6.2% |    2 |__bcopy_prv
||||=================================================================
|||  2.9% |    1 |Alltoall(exclusive)
|||  2.9% |    1 |MPI_Barrier
|||==================================================================
||  3.7% |   22 |Output
|||------------------------------------------------------------------
||| 95.5% |   21 |Display_Times
|||       |      | fflush
|||       |      |  _xflsbuf
|||       |      |   write
|||       |      |    __write
|||  4.5% |    1 |MPI_Gather
|||       |      | MPI_CRAY_gather
|||==================================================================
||  3.2% |   19 |Bcast
|||------------------------------------------------------------------
||| 94.7% |   18 |MPI_Bcast
|||       |      | MPI_CRAY_bcast
|||||----------------------------------------------------------------
||||| 66.7% |   12 |MPI_CRAY_bcast(exclusive)
||||| 33.3% |    6 |bcopy
||||||---------------------------------------------------------------
|||||| 83.3% |    5 |__bcopy_wordstrm
|||||| 16.7% |    1 |bcopy(exclusive)
|||||================================================================
|||  5.3% |    1 |MPI_Barrier
|||==================================================================
||  2.9% |   17 |Reduce
||       |      | MPI_Reduce
||       |      |  MPI_CRAY_reduce
|||||----------------------------------------------------------------
||||| 64.7% |   11 |MPI_CRAY_reduce(exclusive)
||||| 35.3% |    6 |MPI_CRAY_op_strm_func_sum3
|||||================================================================
||  2.7% |   16 |Warm_Up
```

```
|||------------------------------------------------------------------
||| 31.2% |    5 |PingPong
||||-----------------------------------------------------------------
|||| 60.0% |    3 |MPI_Send
|||| 20.0% |    1 |MPI_Recv
||||       |      | MPI_CRAY_recv_wait
|||| 20.0% |    1 |MPI_Barrier
||||=================================================================
||| 25.0% |    4 |Allgather
|||       |      | MPI_Allgather
|||||----------------------------------------------------------------
||||| 50.0% |    2 |MPI_CRAY_gather
||||||---------------------------------------------------------------
|||||| 50.0% |    1 |bcopy
||||||       |      | __bcopy_wordstrm
|||||| 50.0% |    1 |MPI_CRAY_gather(exclusive)
||||||===============================================================
||||| 50.0% |    2 |MPI_CRAY_bcast
||||||---------------------------------------------------------------
|||||| 50.0% |    1 |bcopy
||||||       |      | __bcopy_wordstrm
|||||| 50.0% |    1 |MPI_CRAY_bcast(exclusive)
|||||================================================================
||| 12.5% |    2 |Alltoall
||||-----------------------------------------------------------------
|||| 50.0% |    1 |MPI_Alltoall
|||| 50.0% |    1 |MPI_Barrier
||||=================================================================
||| 12.5% |    2 |Allreduce
|||       |      | MPI_Allreduce
|||       |      |  MPI_CRAY_bcast
||||||---------------------------------------------------------------
|||||| 50.0% |    1 |bcopy
||||||       |      | __bcopy_wordstrm
|||||| 50.0% |    1 |MPI_CRAY_bcast(exclusive)
||||||===============================================================
|||  6.2% |    1 |Bcast
|||       |      | MPI_Bcast
|||       |      |  MPI_CRAY_bcast
|||       |      |   bcopy
|||       |      |    __bcopy_wordstrm
|||  6.2% |    1 |Reduce_scatter
|||       |      | MPI_Reduce_scatter
|||       |      |  MPI_CRAY_reduce
|||  6.2% |    1 |Allgatherv
|||       |      | MPI_Allgatherv
|||       |      |  MPI_CRAY_gatherv
|||       |      |   bcopy
|||       |      |    __bcopy_wordstrm
|||==================================================================
||  0.3% |    2 |Basic_Input
|||------------------------------------------------------------------
||| 50.0% |    1 |MPI_Bcast
|||       |      | MPI_CRAY_bcast
||| 50.0% |    1 |Construct_BList
|||       |      | Set_Bmark
|||       |      |  Get_Def_Index
|||       |      |   LWR
|||==================================================================
||  0.2% |    1 |Init_Communicator
||       |      | Set_Communicator
||       |      |  MPI_Comm_split
||       |      |   MPI_CRAY_comm_split
||       |      |    MPI_CRAY_allgather
||       |      |     MPI_CRAY_bcast
||  0.2% |    1 |MPI_Barrier
||=====================================================================
|  0.3% |    2 |exit
|       |      | __dm_exit_barrier
|||--------------------------------------------------------------------
||| 50.0% |    1 |__dm_barrier
|||       |      | nanosleep
|||       |      |  __nanosleep
||| 50.0% |    1 |sched_yield
|||       |      | __sched_yield
|======================================================================

Elapsed time in seconds for processes

  Process    26197: date: begin: Mon May  5 18:21:21 2003
                          end:   Mon May  5 18:21:27 2003
                          6.034060

                   utime: 5.063177
                   stime: 0.797301

  Process    26200: date: begin: Mon May  5 18:21:21 2003
                          end:   Mon May  5 18:21:27 2003
                          6.052026

                   utime: 4.673398
                   stime: 1.157005


Elapsed time in real-time clocks for threads

  pe/thread     0/0:     571676677 rtc (5.71676677 sec)
  pe/thread     1/0:     575910908 rtc (5.75910908 sec)

$ pat_report -d samp%,samp -b fu,pe -s percent=relative PMB-MPI1+async+26200sd.xf
...
Table 1:  -d samp%,samp
          -b fu,pe

 Samp% | Samp |Function
               |PE

100.0% |  593 |Total
|---------------------------------------------------------------------
| 17.5% |  104 |ass_buf
||--------------------------------------------------------------------
|| 47.1% |   49 |pe.0
|| 52.9% |   55 |pe.1
||====================================================================
| 14.0% |   83 |__bcopy_wordstrm
||--------------------------------------------------------------------
|| 44.6% |   37 |pe.0
|| 55.4% |   46 |pe.1
||====================================================================
| 11.3% |   67 |MPI_CRAY_recv_wait
```

```
||------------------------------------------------------------------
|| 61.2% |   41 |pe.0
|| 38.8% |   26 |pe.1
||==================================================================
| 11.1% |   66 |MPI_CRAY_bcast
||------------------------------------------------------------------
|| 48.5% |   32 |pe.0
|| 51.5% |   34 |pe.1
||==================================================================
|  7.3% |   43 |MPI_CRAY_reduce
||------------------------------------------------------------------
|| 18.6% |    8 |pe.0
|| 81.4% |   35 |pe.1
||==================================================================
|  3.9% |   23 |MPI_Send
||------------------------------------------------------------------
|| 30.4% |    7 |pe.0
|| 69.6% |   16 |pe.1
||==================================================================
|  3.7% |   22 |MPI_CRAY_op_strm_func_sum3
||------------------------------------------------------------------
|| 90.9% |   20 |pe.0
||  9.1% |    2 |pe.1
||==================================================================
|  3.5% |   21 |MPI_CRAY_request_send
||------------------------------------------------------------------
|| 47.6% |   10 |pe.0
|| 52.4% |   11 |pe.1
||==================================================================
|  3.5% |   21 |__write
|       |      | pe.0
|  3.0% |   18 |MPI_CRAY_gather
||------------------------------------------------------------------
|| 66.7% |   12 |pe.0
|| 33.3% |    6 |pe.1
||==================================================================
|  2.9% |   17 |MPI_Alltoall
||------------------------------------------------------------------
|| 52.9% |    9 |pe.0
|| 47.1% |    8 |pe.1
||==================================================================
|  2.5% |   15 |MPI_CRAY_progress_incoming
||------------------------------------------------------------------
|| 60.0% |    9 |pe.0
|| 40.0% |    6 |pe.1
||==================================================================
|  2.2% |   13 |__bcopy_prv
||------------------------------------------------------------------
|| 30.8% |    4 |pe.0
|| 69.2% |    9 |pe.1
||==================================================================
|  1.9% |   11 |MPI_CRAY_gatherv
||------------------------------------------------------------------
|| 63.6% |    7 |pe.0
|| 36.4% |    4 |pe.1
||==================================================================
|  1.9% |   11 |MPI_Barrier
|       |      | pe.1
|  1.5% |    9 |MPI_CRAY_scatterv
||------------------------------------------------------------------
```

```
|| 55.6% |     5 |pe.0
|| 44.4% |     4 |pe.1
||======================================================================
|  1.2% |     7 |MPI_CRAY_progress_ack
||----------------------------------------------------------------------
|| 42.9% |     3 |pe.0
|| 57.1% |     4 |pe.1
||======================================================================
|  1.2% |     7 |bcopy
||----------------------------------------------------------------------
|| 57.1% |     4 |pe.0
|| 42.9% |     3 |pe.1
||======================================================================
|  1.2% |     7 |MPI_Sendrecv
||----------------------------------------------------------------------
|| 85.7% |     6 |pe.0
|| 14.3% |     1 |pe.1
||======================================================================
|  1.2% |     7 |MPI_CRAY_request_recv
||----------------------------------------------------------------------
|| 42.9% |     3 |pe.0
|| 57.1% |     4 |pe.1
||======================================================================
|  0.5% |     3 |MPI_Isend
|       |       | pe.1
|  0.5% |     3 |MPI_CRAY_send_wait
|       |       | pe.1
|  0.3% |     2 |Exchange
|       |       | pe.1
|  0.3% |     2 |MPI_Reduce_scatter
||----------------------------------------------------------------------
|| 50.0% |     1 |pe.0
|| 50.0% |     1 |pe.1
||======================================================================
|  0.3% |     2 |MPI_CRAY_type_free
|       |       | pe.1
|  0.3% |     2 |MPI_Waitall
||----------------------------------------------------------------------
|| 50.0% |     1 |pe.0
|| 50.0% |     1 |pe.1
||======================================================================
|  0.2% |     1 |MPI_CRAY_op_func_sum3
|       |       | pe.0
|  0.2% |     1 |__sched_yield
|       |       | pe.0
|  0.2% |     1 |MPI_Wait
|       |       | pe.1
|  0.2% |     1 |__nanosleep
|       |       | pe.1
|  0.2% |     1 |Alltoall
|       |       | pe.1
|  0.2% |     1 |LWR
|       |       | pe.0
|  0.2% |     1 |MPI_Recv
|       |       | pe.0
|=======================================================================

Elapsed time in seconds for processes

  Process    26197: date: begin: Mon May  5 18:21:21 2003
```

```
                          end:   Mon May  5 18:21:27 2003
                          6.034060

                   utime: 5.063177
                   stime: 0.797301

  Process    26200: date: begin: Mon May  5 18:21:21 2003
                          end:   Mon May  5 18:21:27 2003
                          6.052026

                   utime: 4.673398
                   stime: 1.157005


Elapsed time in real-time clocks for threads

  pe/thread     0/0:     571676677 rtc (5.71676677 sec)
  pe/thread     1/0:     575910908 rtc (5.75910908 sec)
```