Performance Tools and PAPI

Philip J. Mucci

Center for Parallel Computers (PDC), Royal Institute of Technology (KTH), Stockholm, Sweden
Innovative Computing Laboratory, UT Knoxville

mucci at cs.utk.edu
http://www.cs.utk.edu/~mucci

October 3rd, 2005
PAPI

• **Performance Application Programming Interface**
• The purpose of PAPI is to implement a standardized portable and efficient API to access the hardware performance monitor counters found on most modern microprocessors.
• The goal of PAPI is to facilitate the optimization of parallel and serial code performance by encouraging the development of cross-platform optimization tools.

• PapiEx
• PerfSuite
• HPCToolkit
• TAU
• KOJAK
• non-PAPI: mpiP
PAPI Features

- Preset, Native and Derived Performance Metrics
- Full enumeration of platform-specific metrics
- Multiplexed Event Measurement
- Callbacks on overflow
- Full SVR4 Profiling, plus extensions (32, 64 bit, etc.)
- Bindings for C, Fortran, Matlab, and Java
- Overflow and profiling on multiple events simultaneously
- Complete memory hierarchy information
- Complete executable and shared information
- Thread safe API
- Efficient thread local storage and locking routines
- 2 API's, high (app. eng.) and low level (tool dev.)
PapiEx: PAPI Execute

• A simple tool that generates performance measurements for the entire run of a code. No recompilation.

• Monitors all subprocesses/threads.

• Monitor detailed memory usage.

• Automatically detects multi-threaded executables.

• Supports counter multiplexing with -m.

• Provides hooks for simple instrumentation of user source code if desired.
PerfSuite

- Three command-line tools:
  - psrun: obtain performance data
  - psprocess: present/transform data
  - psinv: machine inventory utility
- Command line tool similar to IRIX's perfex command.
- Does aggregate counting of the entire run. Also provides statistical profiling.
- Output is XML or Plain Text.
HPCToolkit from Rice U.

- Use event-based sampling and statistical profiling to profile unmodified applications: `hpcrun`
- Interpret program counter histograms: `hpcprof`
- Correlate source code, structure and performance metrics: `hpcview/hpcquick`
- Explore and analyze performance databases: `hpcviewer`
TAU Performance System

- Tuning and Analysis Utilities (11+ year project effort)
- Performance system framework for scalable parallel and distributed high-performance computing
- Targets a general complex system computation model
  - nodes / contexts / threads
  - Multi-level: system / software / parallelism
  - Measurement and analysis abstraction
- Integrated toolkit for performance instrumentation, measurement, analysis, and visualization
  - Portable performance profiling and tracing facility
  - Open software approach with technology integration
- University of Oregon, Forschungszentrum Jülich, LANL
TAU Performance System

- Multi-level performance instrumentation
  - Multi-language automatic source instrumentation
- Flexible and configurable performance measurement
- Widely-ported parallel performance profiling system
  - Computer system architectures and operating systems
  - Different programming languages and compilers
- Support for multiple parallel programming paradigms
  - Multi-threading, message passing, mixed-mode, hybrid
- Support for performance mapping
- Support for object-oriented and generic programming
- Integration in complex software systems and applications
KOJAK

• Joint open-source project between
  – Forschungszentrum Jülich, Germany
  – University of Tennessee, USA

• Performs automatic performance analysis of parallel applications (MPI and/or OpenMP)

• Uses pattern recognition to transform event traces into information about performance bottlenecks relevant to developers
MpiP Overview

- Scalable, light-weight MPI profiling library
  - Generates detailed text summary of MPI behavior
    - Time spent at each MPI function callsite
    - Bytes sent by each MPI function callsite (where applicable)
    - MPI I/O statistics
    - Configurable traceback depth for function callsites
  - Controllable from program using MPI_Pcontrol
    - Allows you to profile just one code module or cycle
    - Allows mpiP profile dumps mid-run
  - Requires only a relink with mpiP libraries

- mpiPview: Qt interface to browse the data
PAPIEX
papiex Usage

Usage: papiex [-lihmukord] [-L event] [-f[prefix]] [-F file] [-e event] ... -- <cmd> <cmd options>

-1          List the available events.
-L event    List information about specific event.
-i          Print information about the host machine.
-h          Print this message.
-V          Print version information.
-m          Enable multiplexing of hardware counters.
-u          Monitor user mode events. (default)
-k          Monitor kernel mode events.
-o          Monitor transient mode events.
-r          Report getrusage() information.
-x          Report memory information.
-d          Enable debugging output.
-f[prefix]  Output to <prefix><cmd>.papiex.<host>.<pid>.<tid>.
-e event    Monitor this hardware event.
PapiEx Output

PapiEx Version: 0.99rc2
Executable: /afs/pdc.kth.se/home/m/mucci/summer/a.out
Processor: Itanium 2
Clockrate: 900.000000
Parent Process ID: 8632
Process ID: 8633
Hostname: h05n05.pdc.kth.se
Options: MEMORY
Start: Wed Aug 24 14:34:18 2005
Domain: User
Real usecs: 1077497
Real cycles: 969742309
Proc usecs: 970144
Proc cycles: 873129600
PAPI_TOT_CYC: 850136123
PAPI_FP_OPS: 40001767
Mem Size: 4064
Mem Resident: 2000
Mem Shared: 1504
Mem Text: 16
Mem Library: 2992
Mem Heap: 576
Mem Locked: 0
Mem Stack: 32

Event descriptions:
Event: PAPI_TOT_CYC
   Derived: No
   Short Description: Total cycles
   Long Description: Total cycles
   Developer's Notes:

Event: PAPI_FP_OPS
   Derived: No
   Short Description: FP operations
   Long Description: Floating point operations
   Developer's Notes:
PapiEx Caliper Fortran Example

#include "papiex.h"

program zero

    real a, b, c;
    a = 0.1
    b = 1.1
    c = 2.1

    PAPIEX_START_ARG(1,"write")

    print *, "Doing 10000000 iters. of a += b * c on doubles."

    PAPIEX_STOP_ARG(1)

    PAPIEX_START_ARG(2,"do loop")

    do i=1,100000000
        a = a + b * c
    end do

    PAPIEX_STOP_ARG(2)

end
```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include "papiex.h"

volatile double a = 0.1, b = 1.1, c = 2.1;

int main(int argc, char **argv)
{
    int i;

    PAPIEX_START_ARG(1,"printf");
    printf("Doing 100000000 iters. of a += b * c on doubles.\n");
    PAPIEX_STOP_ARG(1);

    PAPIEX_START_ARG(2,"for loop");
    for (i=0;i<100000000;i++)
        a += b * c;
    PAPIEX_STOP_ARG(2);

    exit(0);
}
```
bash-3.00$ papiex -e PAPI_L1_DCM ./a.out
   Doing 10000000 iters. of a += b * c on doubles.

[normal papiex output...]
PAPI_L1_DCM: 6864

Caliper 1: write
   Executions: 1
   Real usecs: 42
   Real cycles: 144432
   Proc usecs: 42
   Proc cycles: 144304
PAPI_L1_DCM: 667

Caliper 2: do loop
   Executions: 1
   Real usecs: 769107
   Real cycles: 2608043669
   Proc usecs: 769019
   Proc cycles: 2607743748
PAPI_L1_DCM: 4167

Event descriptions:
Event: PAPI_L1_DCM
     Derived: No
     Short Description: L1D cache misses
     Long Description: Level 1 data cache misses
     Developer's Notes:
PerfSuite
PerfSuite Hardware Performance Summary Report
Version : 1.0
Created : Mon Dec 30 11:31:53 AM Central Standard Time 2002
Generator : psprocess 0.5
XML Source : /u/ncsa/anyuser/performance/psrun-ia64.xml

Execution Information
============================================================================================
Date         : Sun Dec 15 21:01:20 2002
Host         : user01

Processor and System Information
============================================================================================
Node CPUs    : 2
Vendor       : Intel
Family       : IPF
Model        : Itanium
CPU Revision : 6
Clock (MHz)  : 800.136

Memory (MB)  : 2007.16

Cache Information
============================================================================================
Cache levels : 3
--------------------------------
Level 1
Type         : data
Size (KB)    : 16
Linesize (B) : 32
Assoc        : 4
Type         : instruction
Size (KB)    : 16
Linesize (B) : 32
Assoc        : 4
--------------------------------
Level 2
Type         : unified
Size (KB)    : 96
Linesize (B) : 64
Assoc        : 6
--------------------------------
Level 3
Type         : unified
Size (KB)    : 4096
Linesize (B) : 64
Assoc        : 4
## PSRUN Sample Output

<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
<th>Counter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Conditional branch instructions mispredicted</td>
<td>4831072449</td>
</tr>
<tr>
<td>2</td>
<td>Conditional branch instructions correctly predicted</td>
<td>52023705122</td>
</tr>
<tr>
<td>3</td>
<td>Conditional branch instructions taken</td>
<td>47366258159</td>
</tr>
<tr>
<td>4</td>
<td>Floating point instructions</td>
<td>86124489172</td>
</tr>
<tr>
<td>5</td>
<td>Total cycles</td>
<td>59457754568</td>
</tr>
<tr>
<td>6</td>
<td>Instructions completed</td>
<td>1049339828741</td>
</tr>
<tr>
<td>7</td>
<td>Level 1 data cache accesses</td>
<td>30238866204</td>
</tr>
<tr>
<td>8</td>
<td>Level 1 data cache hits</td>
<td>972479062</td>
</tr>
<tr>
<td>9</td>
<td>Level 1 data cache misses</td>
<td>29224377672</td>
</tr>
<tr>
<td>10</td>
<td>Level 1 instruction cache reads</td>
<td>221828591306</td>
</tr>
<tr>
<td>11</td>
<td>Level 1 cache misses</td>
<td>29312740738</td>
</tr>
<tr>
<td>12</td>
<td>Level 2 data cache accesses</td>
<td>129470315862</td>
</tr>
<tr>
<td>13</td>
<td>Level 2 data cache misses</td>
<td>15569536443</td>
</tr>
<tr>
<td>14</td>
<td>Level 2 data cache reads</td>
<td>110524791561</td>
</tr>
<tr>
<td>15</td>
<td>Level 2 data cache writes</td>
<td>18622708948</td>
</tr>
<tr>
<td>16</td>
<td>Level 2 instruction cache reads</td>
<td>566330907</td>
</tr>
<tr>
<td>17</td>
<td>Level 2 store misses</td>
<td>1208372120</td>
</tr>
<tr>
<td>18</td>
<td>Level 2 cache misses</td>
<td>15401180750</td>
</tr>
<tr>
<td>19</td>
<td>Level 3 data cache accesses</td>
<td>4650999018</td>
</tr>
<tr>
<td>20</td>
<td>Level 3 data cache hits</td>
<td>186108211</td>
</tr>
<tr>
<td>21</td>
<td>Level 3 data cache misses</td>
<td>4451199079</td>
</tr>
<tr>
<td>22</td>
<td>Level 3 data cache reads</td>
<td>4613582451</td>
</tr>
<tr>
<td>23</td>
<td>Level 3 data cache writes</td>
<td>38456570</td>
</tr>
<tr>
<td>24</td>
<td>Level 3 instruction cache misses</td>
<td>3631385</td>
</tr>
<tr>
<td>25</td>
<td>Level 3 instruction cache reads</td>
<td>17631093</td>
</tr>
<tr>
<td>26</td>
<td>Level 3 cache misses</td>
<td>4470968725</td>
</tr>
<tr>
<td>27</td>
<td>Load instructions</td>
<td>111438431677</td>
</tr>
<tr>
<td>28</td>
<td>Load/store instructions completed</td>
<td>130391246662</td>
</tr>
<tr>
<td>29</td>
<td>Cycles Stalled Waiting for memory accesses</td>
<td>256484777623</td>
</tr>
<tr>
<td>30</td>
<td>Store instructions</td>
<td>18840914540</td>
</tr>
<tr>
<td>31</td>
<td>Cycles with no instruction issue</td>
<td>61889609525</td>
</tr>
<tr>
<td>32</td>
<td>Data translation lookaside buffer misses</td>
<td>2832692</td>
</tr>
</tbody>
</table>

## Event Index

<table>
<thead>
<tr>
<th>Event Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>5: PAPI_TOT_CYC</td>
</tr>
<tr>
<td>9: PAPI_L1_DCM</td>
</tr>
<tr>
<td>13: PAPI_L2_DCM</td>
</tr>
<tr>
<td>17: PAPI_L2_STM</td>
</tr>
<tr>
<td>21: PAPI_L3_DCM</td>
</tr>
</tbody>
</table>
PSRUN Sample Output

Statistics

Graduated instructions per cycle....................................... 1.765
Graduated floating point instructions per cycle......................... 0.145
% graduated floating point instructions of all graduated instructions.. 8.207
Graduated loads/stores per cycle...................................... 0.219
Graduated loads/stores per graduated floating point instruction...... 1.514
Mispredicted branches per correctly predicted branch.................. 0.093
Level 1 data cache accesses per graduated instruction................ 2.882
Graduated floating point instructions per level 1 data cache access... 2.848
Level 1 cache line reuse (data)....................................... 3.462
Level 2 cache line reuse (data)....................................... 0.877
Level 3 cache line reuse (data)....................................... 2.498
Level 1 cache hit rate (data)........................................ 0.776
Level 2 cache hit rate (data)........................................ 0.467
Level 3 cache hit rate (data)........................................ 0.714
Level 1 cache miss ratio (instruction)................................. 0.003
Level 1 cache miss ratio (data)....................................... 0.966
Level 2 cache miss ratio (data)....................................... 0.120
Level 3 cache miss ratio (data)....................................... 0.957
Bandwidth used to level 1 cache (MB/s)................................. 1262.361
Bandwidth used to level 2 cache (MB/s)................................. 1326.512
Bandwidth used to level 3 cache (MB/s)................................. 385.087
% cycles with no instruction issue................................... 10.410
% cycles stalled on memory access.................................... 43.139
MFLOPS (cycles)........................................................ 115.905
MFLOPS (wallclock)..................................................... 114.441
MIPS (cycles).......................................................... 1412.190
MIPS (wallclock)....................................................... 1394.349
CPU time (seconds)..................................................... 743.058
Wall clock time (seconds)............................................. 752.566
% CPU utilization........................................................ 98.737
HPCToolkit
HPCToolkit

• A statistical profiling package based on interrupts from the performance monitoring hardware.
• No instrumentation required, but compiling with -g helps.
• Does not work on statically linked programs.
• 2 phase, collections and visualization.
HPCToolkit Sample Output
hpcrun Usage

hpcrun [-lLVhr] [-t each,all] [-e event:[period]] [-o path] [-f flag]
     <cmd> -- <cmd options>

- l                List the available events.
- L                List detailed information about all events.
- V                Print version information.
- h                Print this message.
- r                Do not follow subprocesses.
- t [each,all]     Profile threaded applications.
- e event[:period] Sample event every period counts.
- o path           Directory for output data.
- f flag           PAPI profile mode.

Default is to profile every 32768 cycles (PAPI_TOT_CYC).
hpcprof Usage

hpcprof [-hefrl] [-H dir] <cmd> <profile> ... <profile>

- h       Print this message.
- e       Dump all information.
- f       Dump results by file.
- r       Dump results by function.
- l       Dump results by line.
- H dir   Dump HTML into dir.

There are more options.
hpcprof Output

[mucci@h05n05:~]$ hpcprof -e ./a.out ./a.out.PAPI_TOT_CYC.h05n05.pdc.kth.se.13255.0
Columns correspond to the following events [event:period (events/sample)]
  PAPI_TOT_CYC:32767 - Total cycles (24755 samples)

Load Module Summary:
  100.0% /afs/pdc.kth.se/home/m/mucci/a.out

File Summary:
  100.0% <</afs/pdc.kth.se/home/m/mucci/a.out>>/afs/pdc.kth.se/home/m/mucci/main.c

Function Summary:
  100.0% <</afs/pdc.kth.se/home/m/mucci/a.out>>main

Line Summary:
  90.0% <</afs/pdc.kth.se/home/m/mucci/a.out>>/afs/pdc.kth.se/home/m/mucci/main.c:7
  5.8% <</afs/pdc.kth.se/home/m/mucci/a.out>>/afs/pdc.kth.se/home/m/mucci/main.c:6
  4.2% <</afs/pdc.kth.se/home/m/mucci/a.out>>/afs/pdc.kth.se/home/m/mucci/main.c:4

File <</afs/pdc.kth.se/home/m/mucci/a.out>>/afs/pdc.kth.se/home/m/mucci/main.c with profile annotations.
  1         main()
  2         {
  3         int i;
  4         4.2% for (i=0;i<10000000;i++)
  5         {
  6         5.8% double a = 1.0, b = 2.0, c = 3.0;
  7         90.0% a += b * c + (double)i;
  8         }
  9         }
TAU
Strategies for Empirical Performance Evaluation

Empirical performance evaluation as a series of performance experiments

- Experiment trials describing instrumentation and measurement requirements

Where/When/How axes of empirical performance space

- where are performance measurements made in program
  - routines, loops, statements…
- when is performance instrumentation done
  - compile-time, while pre-processing, runtime…
- how are performance measurement/instrumentation chosen
  - profiling with hw counters, tracing, callpath profiling…
TAU Instrumentation Approach

- Support for standard program events
  - Routines
  - Classes and templates
  - Statement-level blocks

- Support for user-defined events
  - Begin/End events ("user-defined timers")
  - Atomic events (e.g., size of memory allocated/freed)
  - Selection of event statistics

- Support definition of "semantic" entities for mapping

- Support for event groups

- Instrumentation optimization
TAU Instrumentation

- Flexible instrumentation mechanisms at multiple levels
  - Source code
    - manual
    - automatic
      - C, C++, F77/90/95 (Program Database Toolkit (PDT))
      - OpenMP (directive rewriting (Opari), POMP spec)
  - Object code
    - pre-instrumented libraries (e.g., MPI using PMPI)
    - statically-linked and dynamically-linked
  - Executable code
    - dynamic instrumentation (pre-execution) (DynInstAPI)
    - virtual machine instrumentation (e.g., Java using JVMPI)
Multi-Level Instrumentation

- Targets common measurement interface
  - *TAU API*

- Multiple instrumentation interfaces
  - Simultaneously active

- Information sharing between interfaces
  - Utilizes instrumentation knowledge between levels

- Selective instrumentation
  - Available at each level
  - Cross-level selection

- Targets a common performance model

- Presents a unified view of execution
  - Consistent performance events
Program Database Toolkit (PDT)

- Program code analysis framework
  - develop source-based tools

- High-level interface to source code information

- Integrated toolkit for source code parsing, database creation, and database query
  - Commercial grade front-end parsers
  - Portable IL analyzer, database format, and access API
  - Open software approach for tool development

- Multiple source languages

- Implement automatic performance instrumentation tools
  - `tau_instrumentor`
TAU Performance Measurement

- TAU supports profiling and tracing measurement
- Robust timing and hardware performance support using PAPI
- Support for online performance monitoring
  - Profile and trace performance data export to file system
  - Selective exporting
- Extension of TAU measurement for multiple counters
  - Creation of user-defined TAU counters
  - Access to system-level metrics
- Support for callpath measurement
- Integration with system-level performance data
  - Linux MAGNET/MUSE (Wu Feng, LANL)
TAU Measurement

- **Performance information**
  - Performance events
  - High-resolution timer library (real-time / virtual clocks)
  - General software counter library (user-defined events)
  - Hardware performance counters
    - PAPI (Performance API) (UTK, Ptools Consortium)
    - consistent, portable API

- **Organization**
  - Node, context, thread levels
  - Profile groups for collective events (runtime selective)
  - Performance data mapping between software levels
TAU Measurement Options

- **Parallel profiling**
  - Function-level, block-level, statement-level
  - Supports user-defined events
  - TAU parallel profile data stored during execution
  - Hardware counts values
  - Support for multiple counters
  - Support for callgraph and callpath profiling

- **Tracing**
  - All profile-level events
  - Inter-process communication events
  - Trace merging and format conversion
Grouping Performance Data in TAU

- **Profile Groups**
  - A group of related routines forms a profile group
  - Statically defined
    - TAU_DEFAULT, TAU_USER[1-5], TAU_MESSAGE, TAU_IO, …
  - Dynamically defined
    - group name based on string, such as “adlib” or “particles”
    - runtime lookup in a map to get unique group identifier
    - uses `tau_instrumentor` to instrument
  - Ability to change group names at runtime
  - Group-based instrumentation and measurement control
TAU Analysis

- Parallel profile analysis
  - *Pprof*
    - parallel profiler with text-based display
  - *ParaProf*
    - Graphical, scalable, parallel profile analysis and display

- Trace analysis and visualization
  - Trace merging and clock adjustment (if necessary)
  - Trace format conversion (ALOG, SDDF, VTF, Paraver)
  - Trace visualization using *Vampir* (Pallas/Intel)
Pprof Output (NAS LU Parallel)

- **Intel Quad PIII Xeon**
- **F90 + MPICH**
- **Profile**
  - Node
  - Context
  - Thread
- **Events**
  - code
  - MPI

### Buffers Files Tools Edit Search Mulo Help

Reading Profile files in profile.*

<table>
<thead>
<tr>
<th>%Time</th>
<th>Exclusive total msec</th>
<th>Inclusive total msec</th>
<th>#Call</th>
<th>#Subrs</th>
<th>Inclusive Name usec/call</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0</td>
<td>1</td>
<td>3:11.293</td>
<td>1</td>
<td>15</td>
<td>191293269 applu</td>
</tr>
<tr>
<td>99.6</td>
<td>3.667</td>
<td>3:10.453</td>
<td>3</td>
<td>37517</td>
<td>63487925 bcast_inputs</td>
</tr>
<tr>
<td>67.1</td>
<td>491</td>
<td>2:08.326</td>
<td>37200</td>
<td>37200</td>
<td>3450 exchange_1</td>
</tr>
<tr>
<td>44.5</td>
<td>6461</td>
<td>1:25.159</td>
<td>9300</td>
<td>18600</td>
<td>9157 buts</td>
</tr>
<tr>
<td>41.0</td>
<td>1:18.436</td>
<td>1:18.436</td>
<td>18600</td>
<td>0</td>
<td>4217 MPI_Recv()</td>
</tr>
<tr>
<td>29.5</td>
<td>6778</td>
<td>56.407</td>
<td>9300</td>
<td>18600</td>
<td>6065 bts</td>
</tr>
<tr>
<td>26.2</td>
<td>50142</td>
<td>50142</td>
<td>19204</td>
<td>0</td>
<td>2511 MPI_Send()</td>
</tr>
<tr>
<td>16.2</td>
<td>24451</td>
<td>31031</td>
<td>301</td>
<td>602</td>
<td>103096 rhs</td>
</tr>
<tr>
<td>3.9</td>
<td>7501</td>
<td>7501</td>
<td>9300</td>
<td>0</td>
<td>807 jacl</td>
</tr>
<tr>
<td>3.4</td>
<td>838</td>
<td>6594</td>
<td>604</td>
<td>1812</td>
<td>10318 exchange_3</td>
</tr>
<tr>
<td>3.4</td>
<td>6590</td>
<td>6590</td>
<td>9300</td>
<td>0</td>
<td>709 jacl</td>
</tr>
<tr>
<td>2.6</td>
<td>4989</td>
<td>4989</td>
<td>608</td>
<td>0</td>
<td>8206 MPI_Wait()</td>
</tr>
<tr>
<td>0.2</td>
<td>0.44</td>
<td>400</td>
<td>1</td>
<td>4</td>
<td>400081 init_comm</td>
</tr>
<tr>
<td>0.2</td>
<td>398</td>
<td>399</td>
<td>1</td>
<td>39</td>
<td>399534 MPI_Init()</td>
</tr>
<tr>
<td>0.1</td>
<td>140</td>
<td>247</td>
<td>1</td>
<td>47616</td>
<td>247086 setiv</td>
</tr>
<tr>
<td>0.1</td>
<td>131</td>
<td>131</td>
<td>57252</td>
<td>0</td>
<td>2 exact</td>
</tr>
<tr>
<td>0.1</td>
<td>89</td>
<td>103</td>
<td>1</td>
<td>2</td>
<td>103168 erhs</td>
</tr>
<tr>
<td>0.1</td>
<td>0.966</td>
<td>96</td>
<td>1</td>
<td>2</td>
<td>96458 read_input</td>
</tr>
<tr>
<td>0.0</td>
<td>95</td>
<td>95</td>
<td>9</td>
<td>0</td>
<td>10503 MPI_Bcast()</td>
</tr>
<tr>
<td>0.0</td>
<td>26</td>
<td>44</td>
<td>1</td>
<td>7937</td>
<td>44878 error</td>
</tr>
<tr>
<td>0.0</td>
<td>24</td>
<td>24</td>
<td>608</td>
<td>0</td>
<td>40 MPI_Irecv()</td>
</tr>
<tr>
<td>0.0</td>
<td>15</td>
<td>15</td>
<td>1</td>
<td>5</td>
<td>15630 MPI_Finalize()</td>
</tr>
<tr>
<td>0.0</td>
<td>4</td>
<td>12</td>
<td>1</td>
<td>1700</td>
<td>12335 setb</td>
</tr>
<tr>
<td>0.0</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>2893 12norm</td>
</tr>
<tr>
<td>0.0</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>0</td>
<td>491 MPI_Allreduce()</td>
</tr>
<tr>
<td>0.0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>3874 plintgr</td>
</tr>
<tr>
<td>0.0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1007 MPI_Barrier()</td>
</tr>
<tr>
<td>0.0</td>
<td>0.116</td>
<td>0.837</td>
<td>1</td>
<td>4</td>
<td>837 exchange_4</td>
</tr>
<tr>
<td>0.0</td>
<td>0.512</td>
<td>0.512</td>
<td>1</td>
<td>0</td>
<td>512 MPI_Keyval_create()</td>
</tr>
<tr>
<td>0.0</td>
<td>0.121</td>
<td>0.353</td>
<td>1</td>
<td>2</td>
<td>353 exchange_5</td>
</tr>
<tr>
<td>0.0</td>
<td>0.024</td>
<td>0.191</td>
<td>1</td>
<td>2</td>
<td>191 exchange_6</td>
</tr>
<tr>
<td>0.0</td>
<td>0.103</td>
<td>0.103</td>
<td>6</td>
<td>0</td>
<td>17 MPI_Type_contiguous()</td>
</tr>
</tbody>
</table>
ParaProf (NAS LU Parallel)

- **Global profiles** across all nodes
  - **Routine profile**
  - **Event legend**
  - **Individual profile**
TAU + Vampir (NAS Parallel Benchmark – LU)

Timeline display
Callgraph display
Parallelism display
Communications display
PETSc ex19 (Tracing)

Commonly seen communication behavior
TAU’s EVH1 Execution Trace in Vampir

MPI_Alltoall is an execution bottleneck
Performance Analysis and Visualization

- Analysis of parallel profile and trace measurement
- Parallel profile analysis
  - ParaProf
  - Profile generation from trace data
- Performance database framework (PerfDBF)
- Parallel trace analysis
  - Translation to VTF 3.0 and EPILOG
  - Integration with VNG (Technical University of Dresden)
- Online parallel analysis and visualization
ParaProf Framework Architecture

- Portable, extensible, and scalable tool for profile analysis
- Try to offer “best of breed” capabilities to analysts
- Build as profile analysis framework for extensibility
Profile Manager Window

ParaProf Manager

Clicking on different values causes ParaProf to display the clicked on metric.

The sub-window below allow you to generate new metrics based on those that were gathered during the run. The operand number options for Operand A and B correspond the numbers prefixing the values.

Structured AMR toolkit (SAMRAI++), LLNL
Node / Context / Thread Profile Window

COUNTER NAME: P_WALL_CLOCK_TIME (seconds)

345.5474  MPI_Allreduce()
116.4951  algsl::HyperbolicLevelIntegrator3::advance_bdry_fill_create
103.2566  algsl::HyperbolicLevelIntegrator3::advanceLevel()
 59.0096  algsl::HyperbolicLevelIntegrator3::fill_new_level_create
 37.4482  mesh::GriddingAlgorithm3::load_balance_boxes
 32.8548  algsl::HyperbolicLevelIntegrator3::advance_bdry_fill_comm
 21.4095  mesh::GriddingAlgorithm3::findRefinementBoxes()
 13.4925  algsl::HyperbolicLevelIntegrator3::coarsen_fluxsum_create
 12.6572  algsl::HyperbolicLevelIntegrator3::coarsen_sync_create
 10.4408  mesh::GriddingAlgorithm3::find_boxes_containing_tags
  8.9215  MPI_Init()
  8.6893  mesh::GriddingAlgorithm3::bdry_fill_tags_create
  7.2717  MPI_Bcast()
  7.1321  MPI_Wait()
  4.0833  algsl::HyperbolicLevelIntegrator3::error_bdry_fill_comm
  3.6778  MPI_Finalize()
  3.1405  MPI_Isend()
  3.0156  MPI_Waitall()
  2.3457  mesh::GriddingAlgorithm3::remove_intersections_regrid_all
  1.7275  MPI_Test()
  1.6515  algsl::HyperbolicLevelIntegrator3::fill_new_level_comm
  1.3919  MPI_Comm_rank()
Derived Metrics
Full Profile Window (Metric-specific)
ParaProf Enhancements

- Readers completely separated from the GUI
- Access to performance profile database

- Profile translators
  - mpiP, papiprof, dynaprof

- Callgraph display
  - prof/gprof style with hyperlinks

- Integration of 3D performance plotting library

- Scalable profile analysis
  - Statistical histograms, cluster analysis, …

- Generalized programmable analysis engine

- Cross-experiment analysis
Empirical-Based Performance Optimization

Experiment management

Process

Performance Tuning

hypotheses

Performance Diagnosis

properties

Performance Experimentation

characterization

Performance Observation

observability requirements

Experiment Schemas

Experiment Trials
TAU Performance Database Framework

Performance analysis programs

Performance data description

PerfDML translators

Raw performance data

Other tools

• profile data only
• XML representation
• project / experiment / trial

PerfDB

ORDB

PostgreSQL
PerfDBF Cross-Trial Analysis
OpenMP + MPI Ocean Modeling

FP instructions

Integrated OpenMP + MPI events
KOJAK
KOJAK Analysis process

- Source code
- Automatic multilevel instrumentation
- Executable
- Execution on parallel machine
- Event Trace
- Automatic pattern analysis
- High-Level Profile

Which type of problem?

Where in the source code? Which call path?

Which process / thread?

Where in my physical / virtual topology
MPIP Output

@ Command: /afs/pdc.kth.se/home/m/mucci/mpiP-2.7/testing/.sweep-ops-stack.exe
/tmp/SPnodes-mucci-0
@ Version: 2.7
@ MPIP Build date: Aug 17 2004, 17:04:36
@ Start time: 2004 08 17 17 08 48
@ Stop time: 2004 08 17 17 08 48
@ MPIP env var: [null]
@ Collector Rank: 0
@ Collector PID: 17412
@ Final Output Dir: .
@ MPI Task Assignment: 0 h05n05-e.pdc.kth.se
@ MPI Task Assignment: 1 h05n35-e.pdc.kth.se
@ MPI Task Assignment: 2 h05n05-e.pdc.kth.se
@ MPI Task Assignment: 3 h05n35-e.pdc.kth.se

@--- MPI Time (seconds) -----------------------------------------------

<table>
<thead>
<tr>
<th>Task</th>
<th>AppTime</th>
<th>MPITime</th>
<th>MPI%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.084</td>
<td>0.0523</td>
<td>62.21</td>
</tr>
<tr>
<td>1</td>
<td>0.0481</td>
<td>0.015</td>
<td>31.19</td>
</tr>
<tr>
<td>2</td>
<td>0.087</td>
<td>0.0567</td>
<td>65.20</td>
</tr>
<tr>
<td>3</td>
<td>0.0495</td>
<td>0.0149</td>
<td>29.98</td>
</tr>
<tr>
<td>*</td>
<td>0.269</td>
<td>0.139</td>
<td>51.69</td>
</tr>
</tbody>
</table>

@--- Aggregate Time (top twenty, descending, milliseconds) ----------

<table>
<thead>
<tr>
<th>Call</th>
<th>Site</th>
<th>Time</th>
<th>App%</th>
<th>MPI%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barrier</td>
<td>1</td>
<td>112</td>
<td>41.57</td>
<td>80.42</td>
</tr>
<tr>
<td>Recv</td>
<td>1</td>
<td>26.2</td>
<td>9.76</td>
<td>18.89</td>
</tr>
<tr>
<td>Allreduce</td>
<td>1</td>
<td>0.634</td>
<td>0.24</td>
<td>0.46</td>
</tr>
<tr>
<td>Bcast</td>
<td>1</td>
<td>0.3</td>
<td>0.11</td>
<td>0.22</td>
</tr>
<tr>
<td>Send</td>
<td>1</td>
<td>0.033</td>
<td>0.01</td>
<td>0.02</td>
</tr>
</tbody>
</table>

@--- Aggregate Sent Message Size (top twenty, descending, bytes) ---

<table>
<thead>
<tr>
<th>Call</th>
<th>Site</th>
<th>Count</th>
<th>Total</th>
<th>Avrg</th>
<th>Sent%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allreduce</td>
<td>1</td>
<td>8</td>
<td>4.8e+03</td>
<td>600</td>
<td>46.15</td>
</tr>
<tr>
<td>Bcast</td>
<td>1</td>
<td>8</td>
<td>4.8e+03</td>
<td>600</td>
<td>46.15</td>
</tr>
<tr>
<td>Send</td>
<td>1</td>
<td>2</td>
<td>800</td>
<td>400</td>
<td>7.69</td>
</tr>
</tbody>
</table>
### MPIP Output (2)

---

#### Callsite Time statistics (all, milliseconds): 16

<table>
<thead>
<tr>
<th>Name</th>
<th>Site</th>
<th>Rank</th>
<th>Count</th>
<th>Max</th>
<th>Mean</th>
<th>Min</th>
<th>App%</th>
<th>MPI%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allreduce</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0.105</td>
<td>0.087</td>
<td>0.069</td>
<td>0.21</td>
<td>0.33</td>
</tr>
<tr>
<td>Allreduce</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.118</td>
<td>0.08</td>
<td>0.042</td>
<td>0.33</td>
<td>1.07</td>
</tr>
<tr>
<td>Allreduce</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0.11</td>
<td>0.072</td>
<td>0.046</td>
<td>0.18</td>
<td>0.79</td>
</tr>
<tr>
<td>Allreduce</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>0.102</td>
<td>0.072</td>
<td>0.042</td>
<td>0.29</td>
<td>0.97</td>
</tr>
<tr>
<td>Barrier</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>51.9</td>
<td>17.3</td>
<td>0.015</td>
<td>61.86</td>
<td>99.44</td>
</tr>
<tr>
<td>Barrier</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0.073</td>
<td>0.0457</td>
<td>0.016</td>
<td>0.29</td>
<td>0.91</td>
</tr>
<tr>
<td>Barrier</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>54.9</td>
<td>18.8</td>
<td>0.031</td>
<td>64.90</td>
<td>99.53</td>
</tr>
<tr>
<td>Barrier</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1.56</td>
<td>1.02</td>
<td>0.035</td>
<td>6.20</td>
<td>20.68</td>
</tr>
<tr>
<td>Bcast</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0.073</td>
<td>0.0535</td>
<td>0.034</td>
<td>0.13</td>
<td>0.20</td>
</tr>
<tr>
<td>Bcast</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.037</td>
<td>0.023</td>
<td>0.009</td>
<td>0.10</td>
<td>0.31</td>
</tr>
<tr>
<td>Bcast</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0.084</td>
<td>0.046</td>
<td>0.008</td>
<td>0.11</td>
<td>0.46</td>
</tr>
<tr>
<td>Bcast</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>0.03</td>
<td>0.0275</td>
<td>0.025</td>
<td>0.11</td>
<td>0.37</td>
</tr>
<tr>
<td>Recv</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>14.6</td>
<td>14.6</td>
<td>14.6</td>
<td>30.48</td>
<td>97.71</td>
</tr>
<tr>
<td>Recv</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>11.6</td>
<td>11.6</td>
<td>11.6</td>
<td>23.37</td>
<td>77.98</td>
</tr>
<tr>
<td>Send</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.013</td>
<td>0.013</td>
<td>0.013</td>
<td>0.02</td>
<td>0.02</td>
</tr>
<tr>
<td>Send</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.04</td>
</tr>
<tr>
<td>Send</td>
<td>1</td>
<td>*</td>
<td>32</td>
<td>54.9</td>
<td>4.34</td>
<td>0.008</td>
<td>51.69</td>
<td>100.00</td>
</tr>
</tbody>
</table>

---

#### Callsite Message Sent statistics (all, sent bytes)

<table>
<thead>
<tr>
<th>Name</th>
<th>Site</th>
<th>Rank</th>
<th>Count</th>
<th>Max</th>
<th>Mean</th>
<th>Min</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allreduce</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>800</td>
<td>600</td>
<td>400</td>
<td>1200</td>
</tr>
<tr>
<td>Allreduce</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>800</td>
<td>600</td>
<td>400</td>
<td>1200</td>
</tr>
<tr>
<td>Allreduce</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>800</td>
<td>600</td>
<td>400</td>
<td>1200</td>
</tr>
<tr>
<td>Allreduce</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>800</td>
<td>600</td>
<td>400</td>
<td>1200</td>
</tr>
<tr>
<td>Bcast</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>800</td>
<td>600</td>
<td>400</td>
<td>1200</td>
</tr>
<tr>
<td>Bcast</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>800</td>
<td>600</td>
<td>400</td>
<td>1200</td>
</tr>
<tr>
<td>Bcast</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>800</td>
<td>600</td>
<td>400</td>
<td>1200</td>
</tr>
<tr>
<td>Bcast</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>800</td>
<td>600</td>
<td>400</td>
<td>1200</td>
</tr>
<tr>
<td>Send</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>Send</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>Send</td>
<td>1</td>
<td>*</td>
<td>18</td>
<td>800</td>
<td>577.8</td>
<td>400</td>
<td>1.04e+04</td>
</tr>
</tbody>
</table>

---

@--- End of Report ---@
Mpipview: An MpiP Output Viewer

- Organizes and condenses mpiP output
  - Allow users to find key mpiP data quickly
  - Hides complexity of large scale runs until needed
  - Shows source code for the MPI callsites reported on
  - Design based on our experience using mpiP on ASC apps

- Easy to use - parses mpiP text output file
  - mpipview irs.8.default.mpiP
MPI Callsite Timing Summaries

- Shows timing stats summaries, sorted by % of MPI
- Clicking on summary displays callsite’s source code
  - Callsites indicate where an MPI call was called from
  - Isend[6] indicates the 6th MPI callsite reached was an MPI_Isend