TAU

Mira Conference at Argonne National Laboratory,

ALCF, Bldg. 240,# 1416, 12pm, March 6, 2013, Argonne, IL

Sameer Shende, U. Oregon

sameer@cs.uoregon.edu

http://tau.uoregon.edu

Slides: /soft/perftools/tau/ppt/tau_mc13.ppt (.pdf)

Acknowledgements: U. Oregon, ParaTools, Inc.

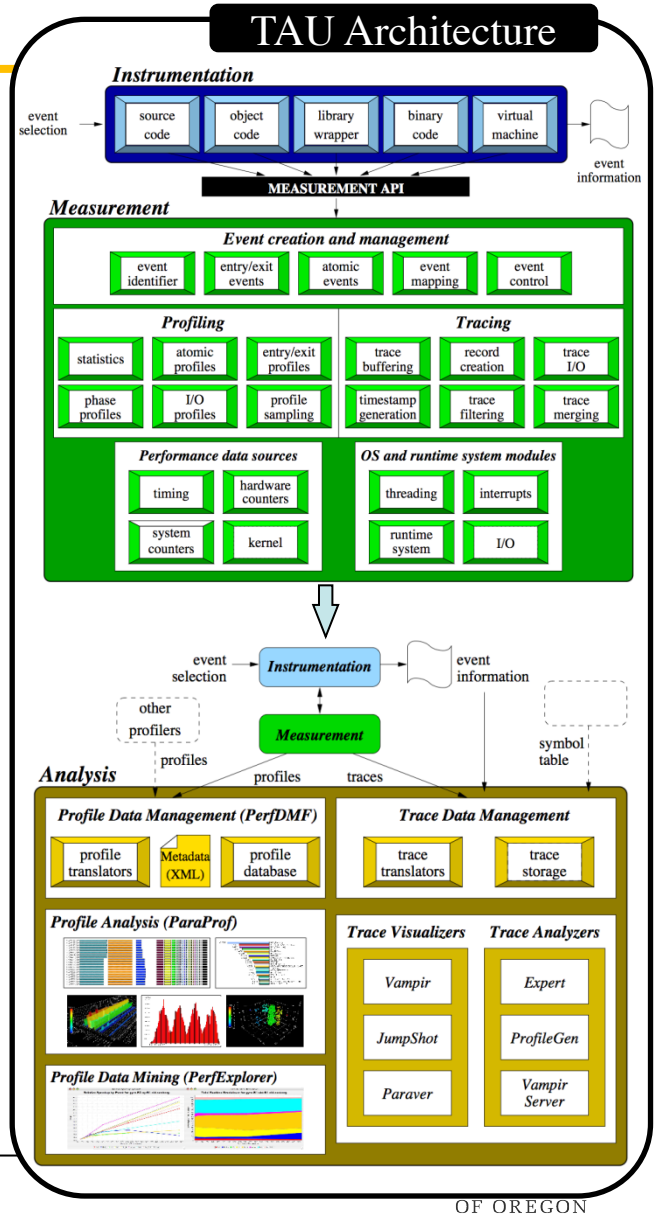
- Dr. Allen D. Malony, Professor, CIS Dept, and Director, NeuroInformatics Center, and CEO, ParaTools, Inc.
- Dr. Kevin Huck, Computer Scientist, ParaTools, Inc.
- Dr. John Linford, Computer Scientist, ParaTools, Inc.
- Wyatt Spear, Software engineer, UO, ParaTools, Inc.
- Suzanne Millstein, Software engineer, UO
- Scott Biersdorff, Software engineer, UO
- Nick Chaimov, Ph.D. student, UO
- David Ozog, Ph.D. student, UO
- David Poliakoff, Ph.D. student, UO
- Dr. Robert Yelle, Research faculty, UO

What is TAU?

- TAU is a performance evaluation tool
- It supports parallel profiling and tracing toolkit
- Profiling shows you how much (total) time was spent in each routine
- Tracing shows you *when* the events take place in each process along a timeline
- Profiling and tracing can measure time as well as hardware performance counters from your CPU
- TAU can automatically instrument your source code (routines, loops, I/O, memory, phases, etc.)
- It supports C++, C, UPC, Fortran, Python, and Java
- TAU runs on all HPC platforms and it is free (BSD style license)
- TAU has instrumentation, measurement and analysis tools
- To use TAU, you need to set a couple of environment variables and substitute the name of the compiler with a TAU shell script

TAU Performance System®

- Integrated toolkit for performance problem solving
  - Instrumentation, measurement, analysis, visualization
  - Portable performance profiling and tracing facility
  - Performance data management and data mining
- Supports both direct as well as indirect (sampling) performance measurement approach
- Open source
- Available on all HPC platforms
- http://tau.uoregon.edu



Performance Evaluation

- Profiling
  - Presents summary statistics of performance metrics
    - number of times a routine was invoked
    - exclusive, inclusive time/hpm counts spent executing it
    - number of instrumented child routines invoked, etc.
    - structure of invocations (calltrees/callgraphs)
    - memory, message communication sizes also tracked
- Tracing
  - Presents when and where events took place along a global timeline
    - timestamped log of events
    - message communication events (sends/receives) are tracked
      - shows when and where messages were sent
    - large volume of performance data generated leads to more perturbation in the program

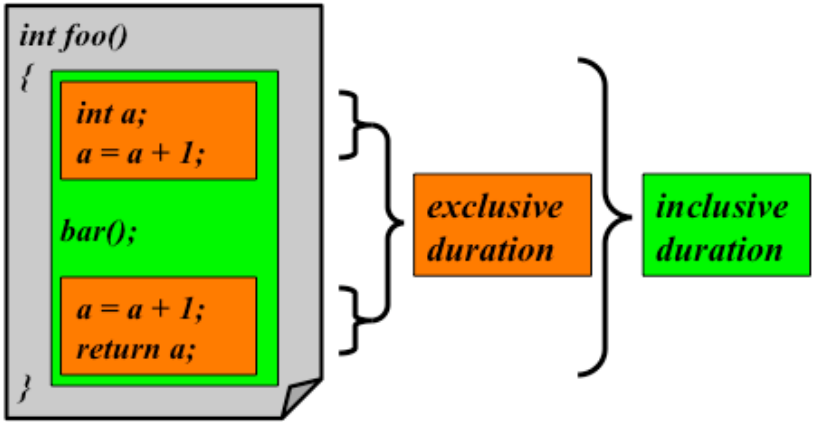
Labs!

Lab Instructions for Vesta

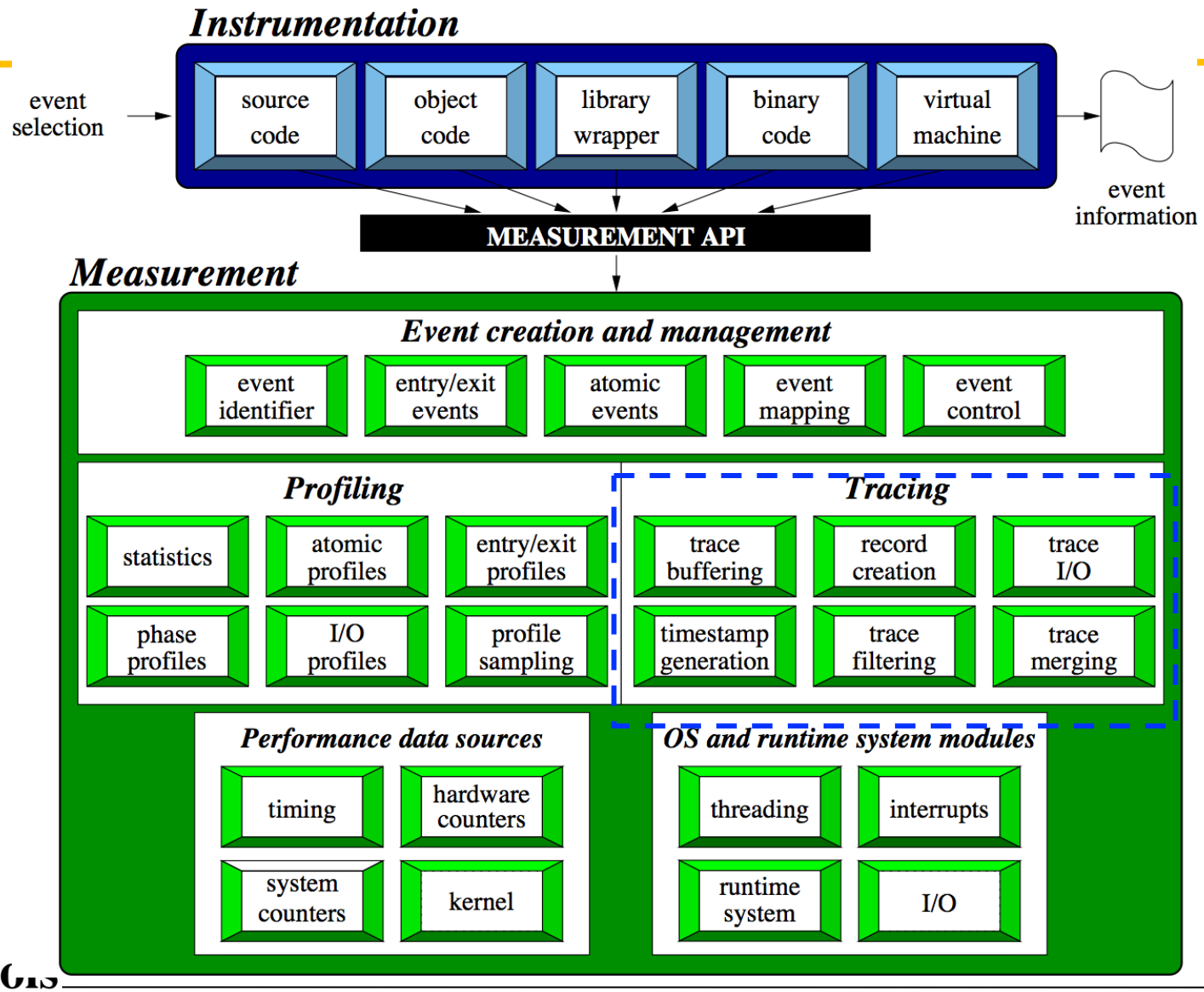
```
Get workshop.tar.gz and install on the SCRATCH filesystem:
% cp /soft/perftools/tau/workshop.tar.gz .
% tar zxf workshop.tar.gz
% cd workshop/first
% cat README
% make
% paraprof tauprofile.xml &
```

TAU Performance Profiling

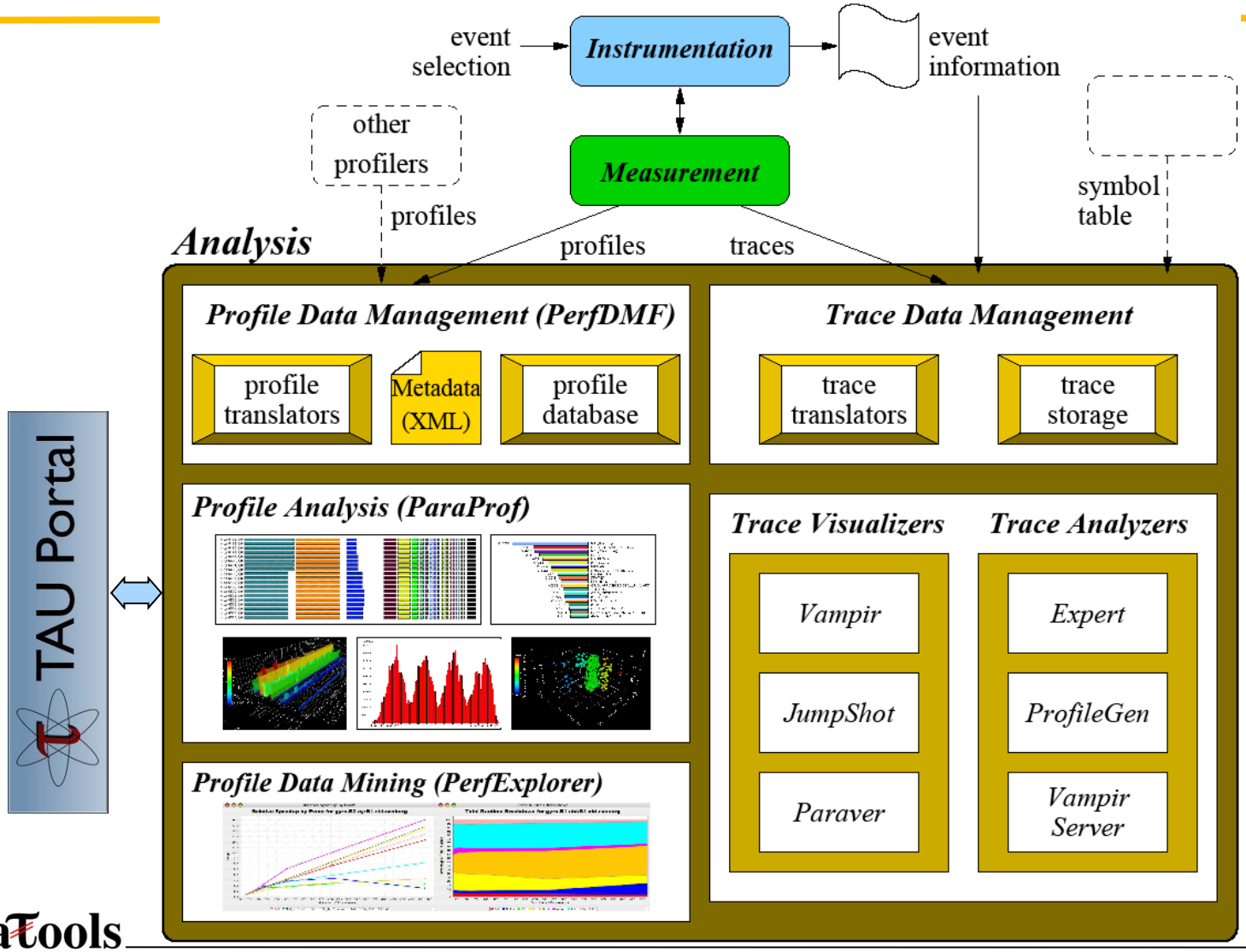
- Performance with respect to nested event regions
  - Program execution event stack (begin/end events)
- Profiling measures inclusive and exclusive data
- Exclusive measurements for region only performance
- Inclusive measurements includes nested "child" regions
- Support multiple profiling types
  - Flat, callpath, and phase profiling



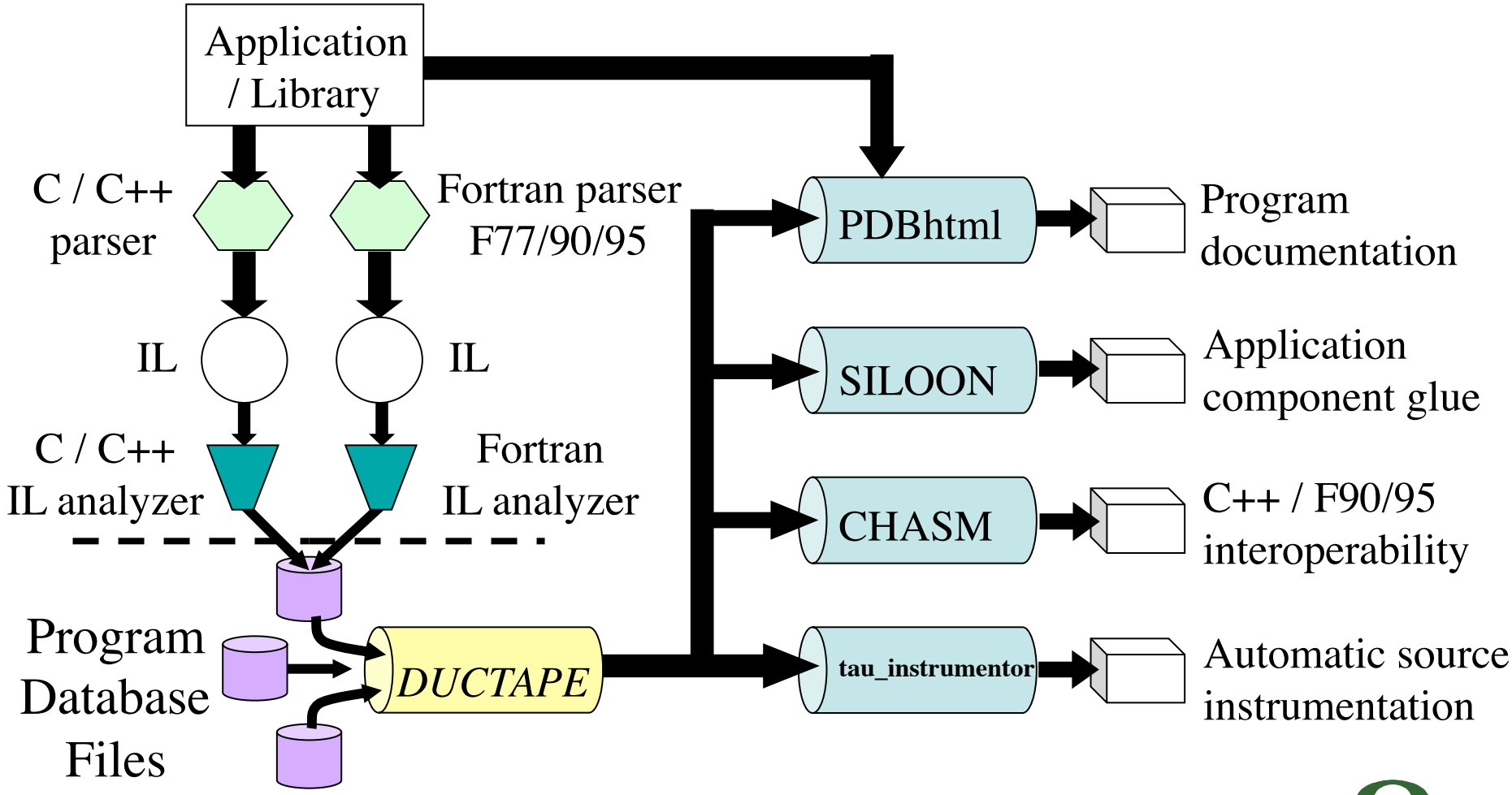
TAU Performance System Architecture



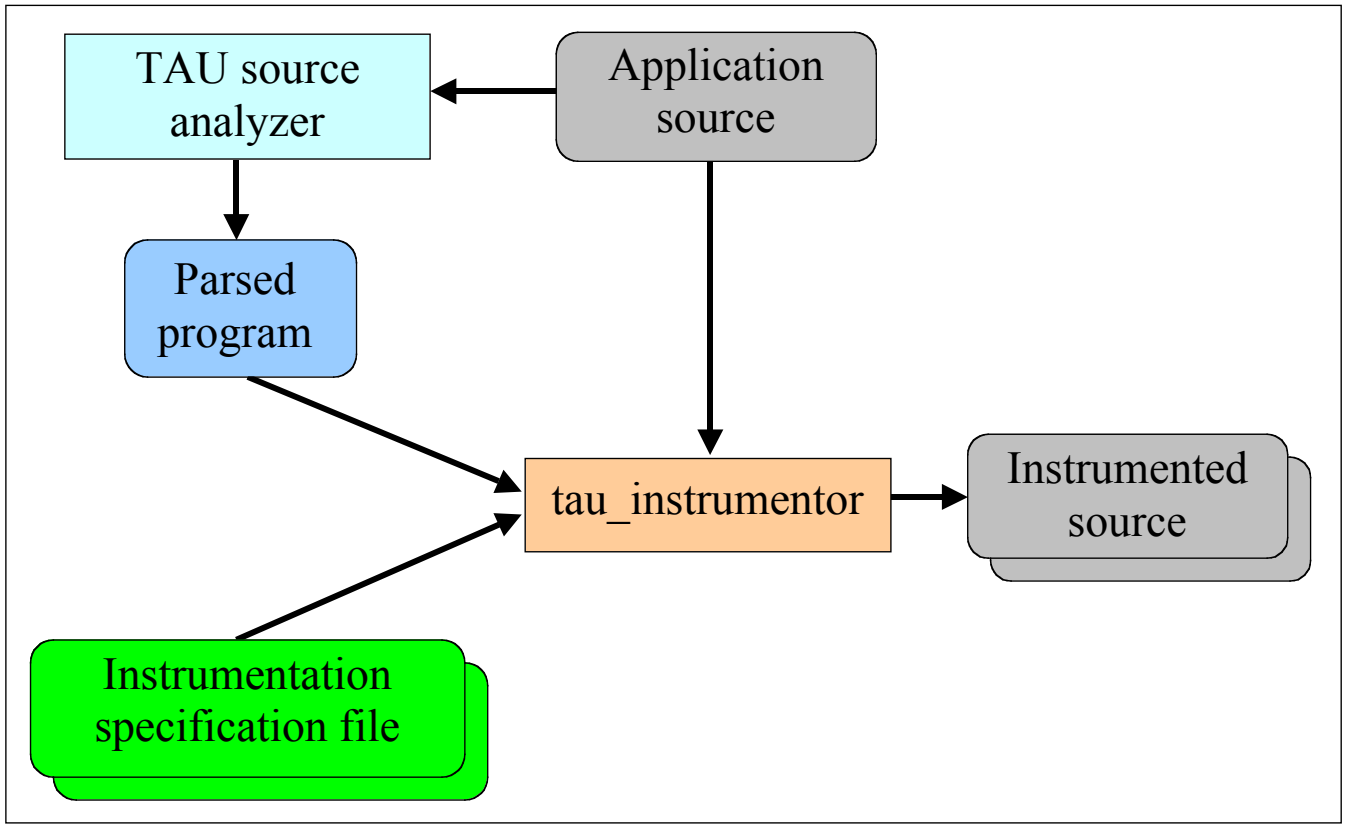
TAU Performance System Architecture



Program Database Toolkit (PDT)



Automatic Source-Level Instrumentation in TAU



Using TAU: A brief Introduction

- TAU supports several measurement options (profiling, tracing, profiling with hardware counters, etc.)
- Each measurement configuration of TAU corresponds to a unique stub makefile that is generated when you configure it
- To instrument source code using PDT
  - Choose an appropriate TAU stub makefile in <arch>/lib:

```
% soft add +tau-latest (on BG/Q)
% export TAU_MAKEFILE=/soft/perftools/tau/tau_latest/bgq/lib/Makefile.tau-bgqtimers-mpi-pdt
% export TAU_OPTIONS= '-optVerbose …' (see tau_compiler.sh -help)
```

  And use tau_f90.sh, tau_cxx.sh or tau_cc.sh as Fortran, C++ or C compilers:

```
% mpixlf90_r foo.f90
```

  changes to

```
% tau_f90.sh foo.f90
% qsub –A MiraCon2013 –q R.mc –n 256 –t 10 ./a.out
```

- Execute application and analyze performance data:

```
% pprof   (for text based profile display)
% paraprof  (for GUI)
```

TAU Measurement Configuration on BG/Q

```
% cd /soft/perftools/tau/tau_latest/bgq/lib; ls Makefile.*
Makefile.tau-bgqtimers-mpi-pdt-openmp-opari
Makefile.tau-bgqtimers-mpi-pthread-pdt
Makefile.tau-bgqtimers-papi-mpi-pdt
Makefile.tau-bgqtimers-papi-mpi-pdt-openmp-opari
Makefile.tau-bgqtimers-papi-mpi-pthread-pdt
Makefile.tau-gnu-bgqtimers-mpi-pthread-pdt
```

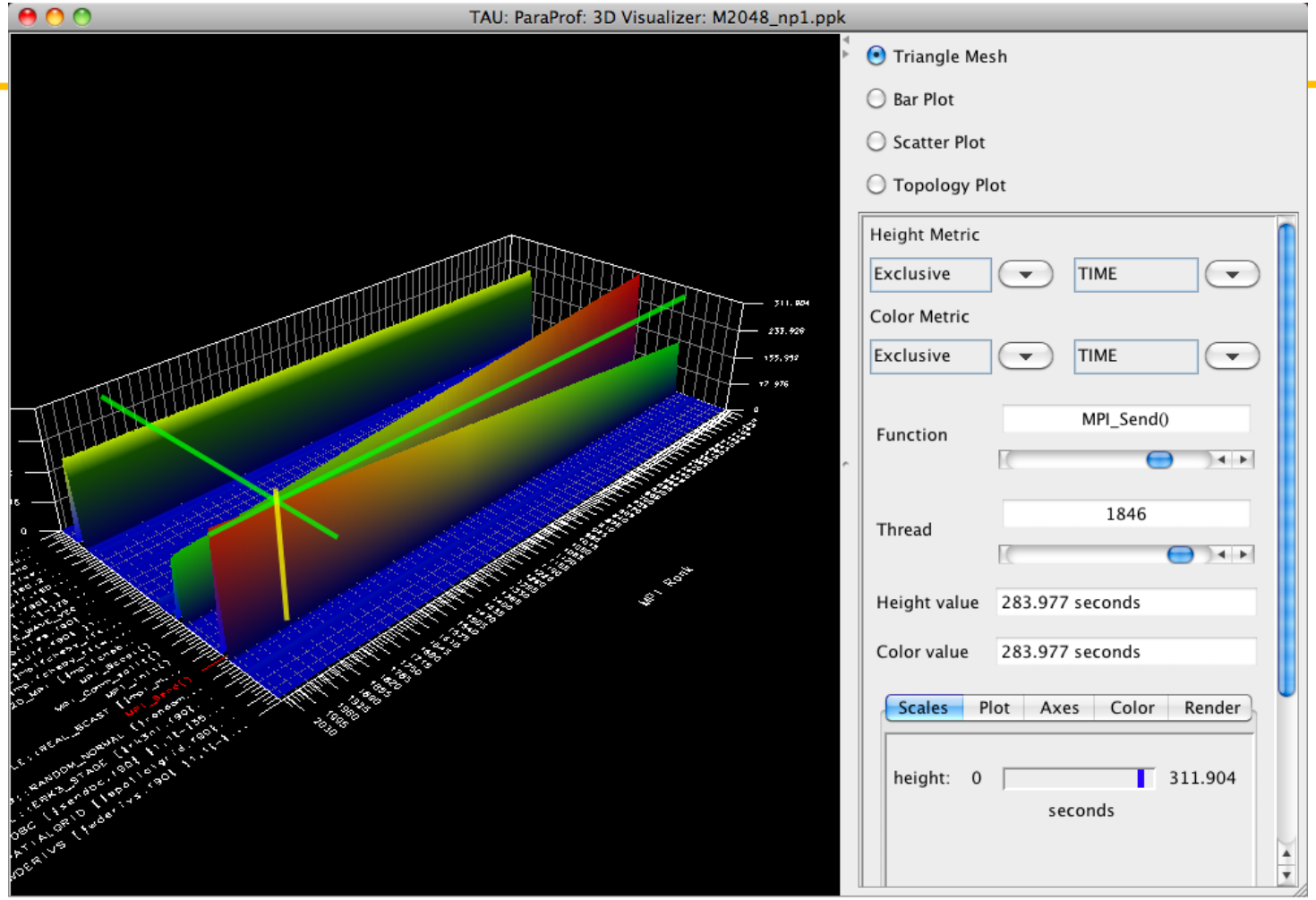
For an MPI+F90 application, you may want to start with:

Makefile.tau-bgqtimers-papi-mpi-pdt

- Supports MPI instrumentation & PDT for automatic source instrumentation

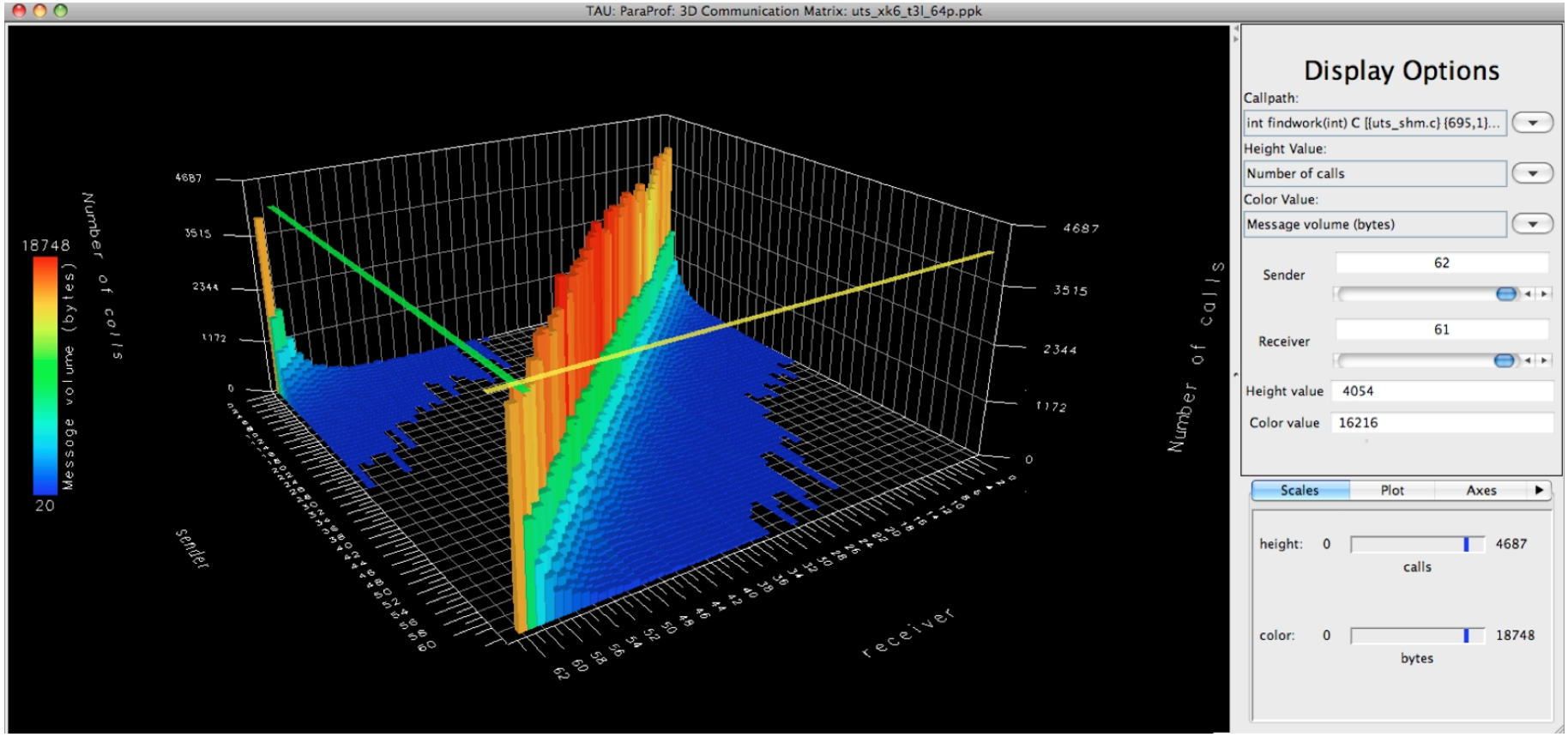
- `% soft add +tau-latest`
- `% export TAU_MAKEFILE=$TAU/Makefile.tau-bgqtimers-papi-mpi-pdt`
- `% make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh`
- `% qsub -q R.mc -n 2 --mode c16 -t 10 -A MiraCon2013 ./a.out`
- `% paraprof`

Parallel Profile Visualization: ParaProf



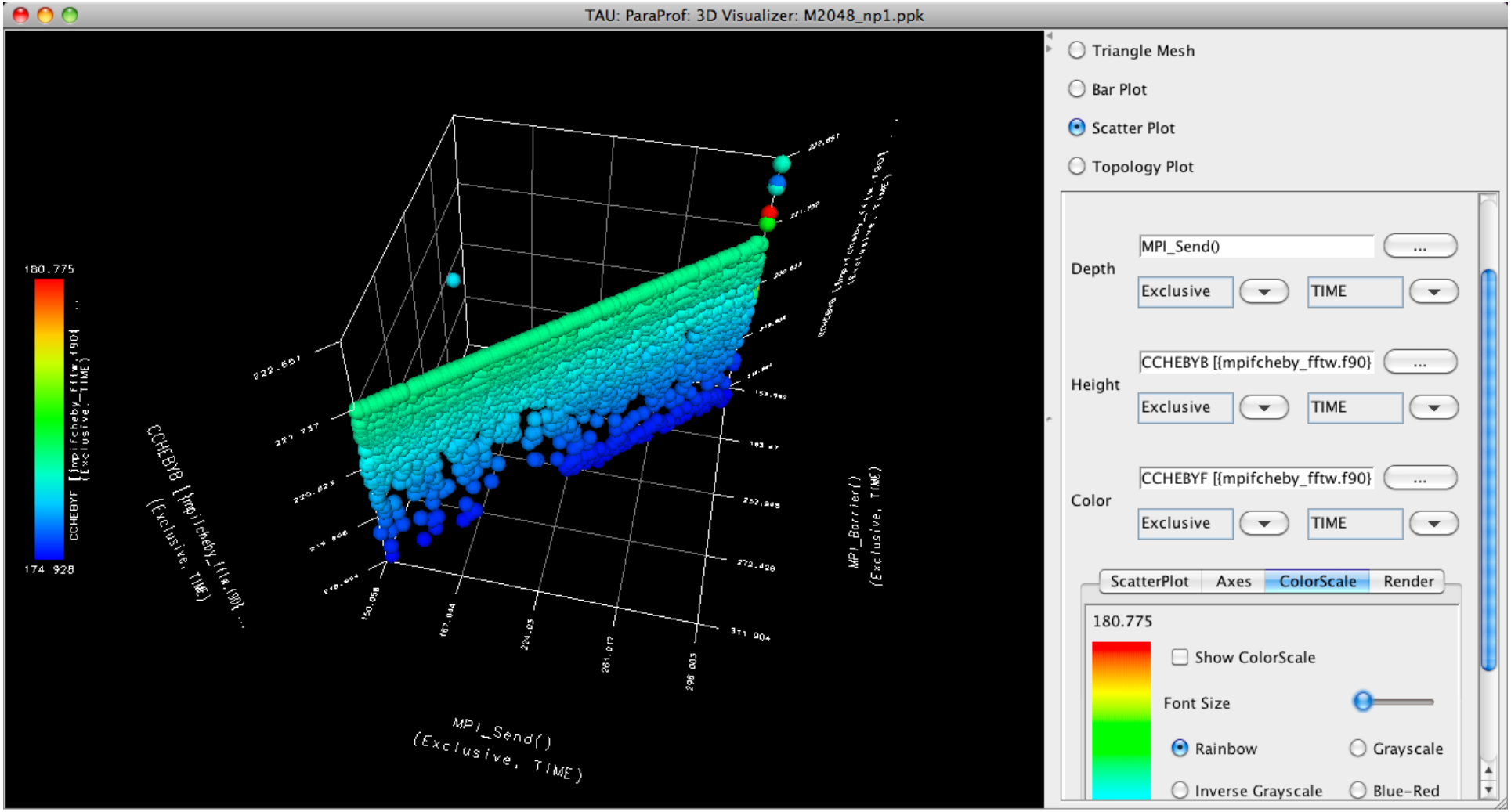

```
% soft add +tau-latest
% paraprof  (Windows -> 3D Visualization)
```

ParaProf: 3D Communication Matrix

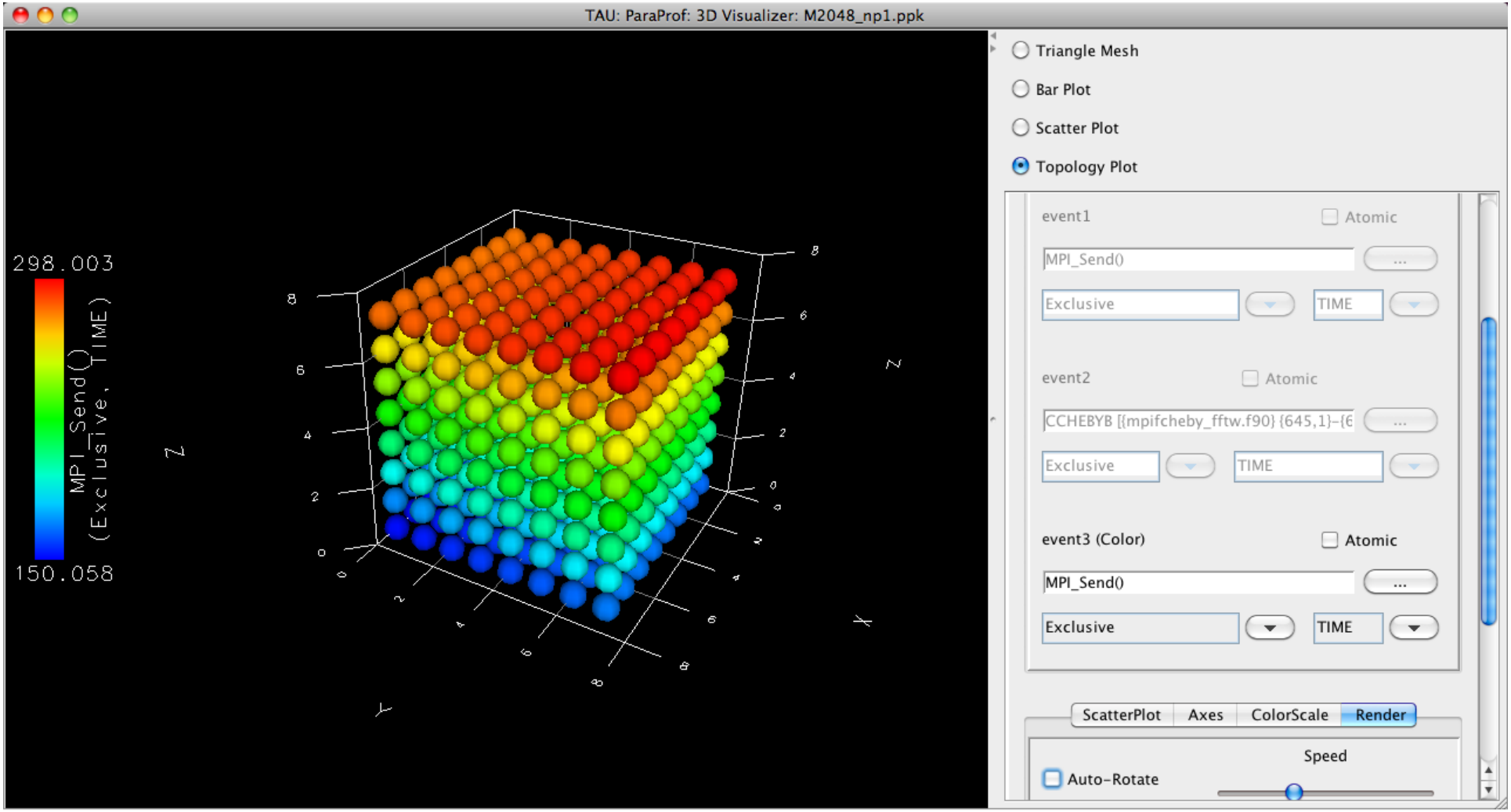



```
% qsub –env TAU_COMM_MATRIX=1 …
% paraprof (Windows -> 3D Communication Matrix)
```

ParaProf: Scatter Plot



ParaProf: Topology View: MPI_Send on BG/P



Interval, Atomic and Context Events in TAU

```
File  Edit  View  Terminal  Tabs  Help
Reading Profile files in profile.*

NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        0.007        0.256           1           5        256 MAIN
 97.3        0.132        0.249           5           5         50 FOO
 40.6        0.104        0.104           5           0         21 BAR
 36.3        0.013        0.093           3           3         31 G
---------------------------------------------------------------------------------------

USER EVENTS Profile :NODE 0, CONTEXT 0, THREAD 0
---------------------------------------------------------------------------------------
NumSamples   MaxValue   MinValue  MeanValue  Std. Dev.  Event Name
---------------------------------------------------------------------------------------
         1         16         16         16          0  MEMORY LEAK! malloc size <file=foo.f90, variable=X, line=7> : MAIN => FOO => BAR
         2         52         48         50          2  MEMORY LEAK! malloc size <file=foo.f90, variable=X, line=7> : MAIN => FOO => G => BAR
         1         80         80         80          0  free size <file=foo.f90, variable=X, line=10>
         1         80         80         80          0  free size <file=foo.f90, variable=X, line=10> : MAIN => FOO => G => BAR
         1        180        180        180          0  free size <file=foo.f90, variable=X, line=15>
         1        180        180        180          0  free size <file=foo.f90, variable=X, line=15> : MAIN => FOO => BAR
         1        180        180        180          0  malloc size <file=foo.f90, variable=X, line=13>
         1        180        180        180          0  malloc size <file=foo.f90, variable=X, line=13> : MAIN => FOO => BAR
         4         80         16         49      22.69  malloc size <file=foo.f90, variable=X, line=7>
         1         16         16         16          0  malloc size <file=foo.f90, variable=X, line=7> : MAIN => FOO => BAR
         3         80         48         60      14.24  malloc size <file=foo.f90, variable=X, line=7> : MAIN => FOO => G => BAR
---------------------------------------------------------------------------------------
                                                                                    1,1           All
```

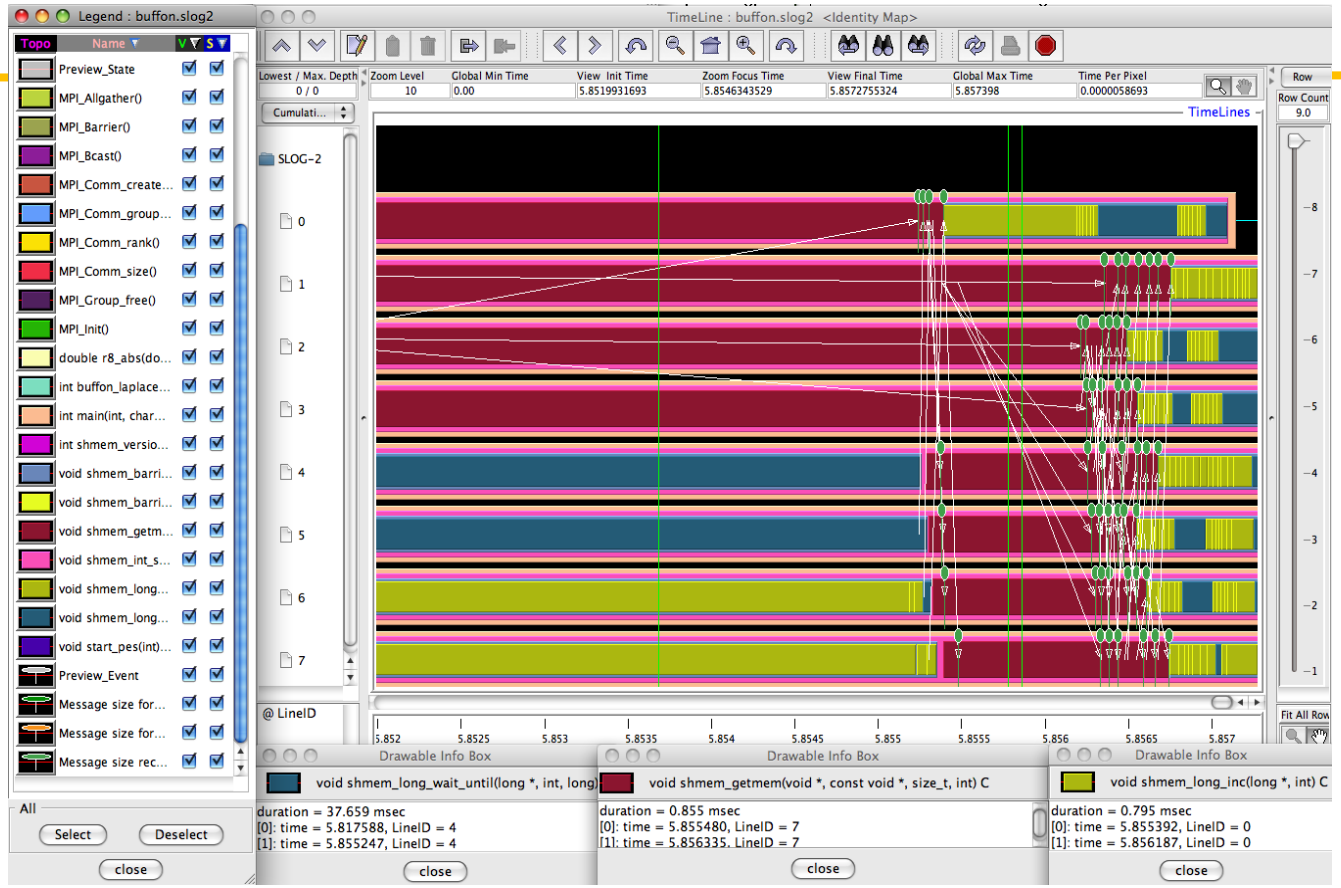
Interval Event

Context Event

Atomic Event

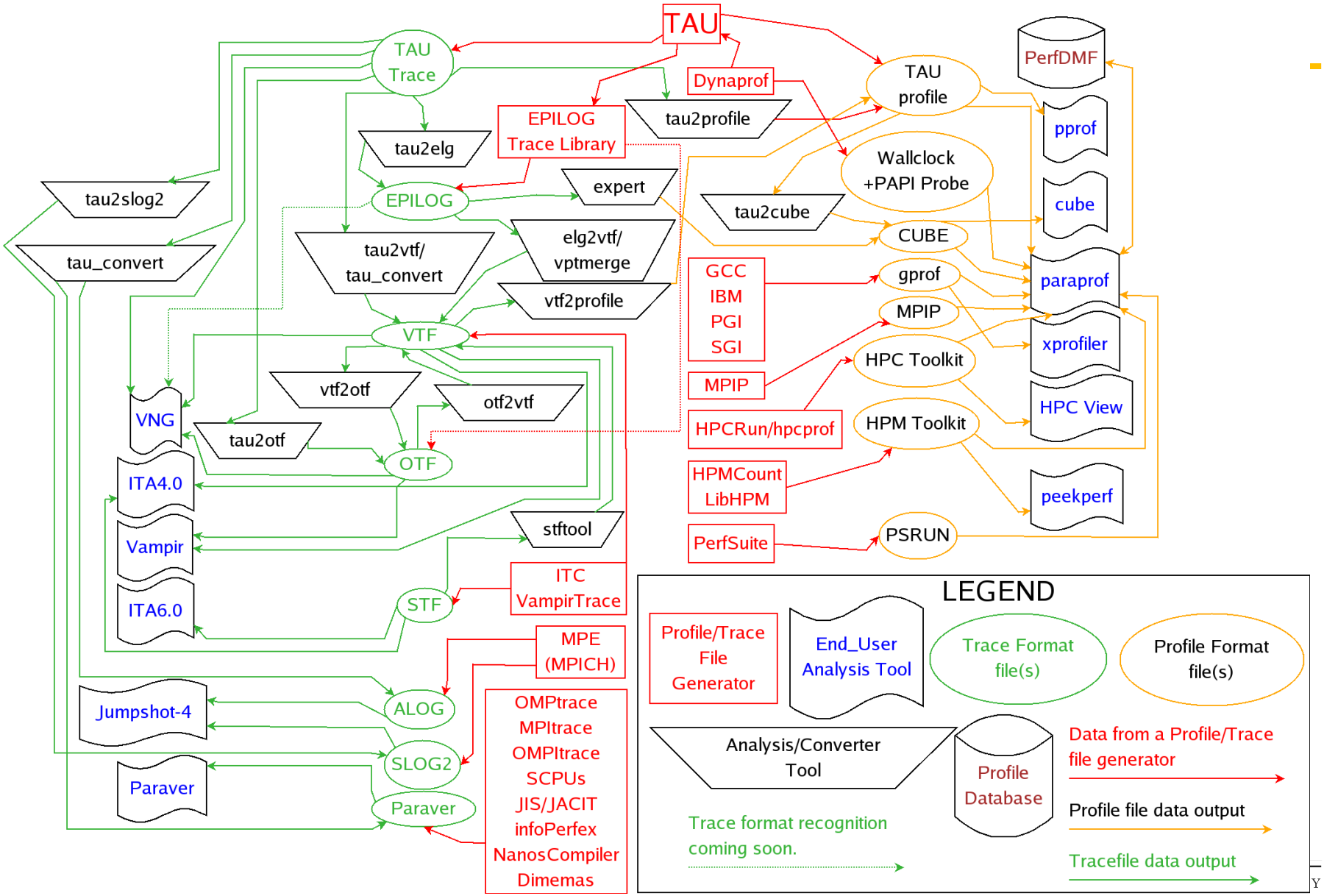
```
% pprof
```

Jumpshot [ANL]: Trace Visualization




```
% qsub –env TAU_TRACE=1 …
% tau_treemerge.pl
% tau2slog2 tau.trc tau.edf –o app.slog2
% jumpshot app.slog2
```

Building Bridges to Other Tools



TAU Instrumentation Approach

- **Support for *standard* program events**
  - Routines, classes and templates
  - Statement-level blocks
  - *Begin/End* events (*Interval* events)
- **Support for *user-defined* events**
  - *Begin/End* events specified by user
  - *Atomic* events (e.g., size of memory allocated/freed)
  - Selection of event statistics
- Support definition of “semantic” entities for mapping
- Support for event groups (aggregation, selection)
- Instrumentation optimization
  - Eliminate instrumentation in lightweight routines

Interval, Atomic and Context Events in TAU

```
File  Edit  View  Terminal  Tabs  Help
Reading Profile files in profile.*

NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        0.007        0.256           1           5        256 MAIN
 97.3        0.132        0.249           5           5         50 FOO
 40.6        0.104        0.104           5           0         21 BAR
 36.3        0.013        0.093           3           3         31 G
---------------------------------------------------------------------------------------

USER EVENTS Profile :NODE 0, CONTEXT 0, THREAD 0
---------------------------------------------------------------------------------------
NumSamples   MaxValue   MinValue  MeanValue  Std. Dev.  Event Name
---------------------------------------------------------------------------------------
         1         16         16         16          0  MEMORY LEAK! malloc size <file=foo.f90, variable=X, line=7> : MAIN => FOO => BAR
         2         52         48         50          2  MEMORY LEAK! malloc size <file=foo.f90, variable=X, line=7> : MAIN => FOO => G => BAR
         1         80         80         80          0  free size <file=foo.f90, variable=X, line=10>
         1         80         80         80          0  free size <file=foo.f90, variable=X, line=10> : MAIN => FOO => G => BAR
         1        180        180        180          0  free size <file=foo.f90, variable=X, line=15>
         1        180        180        180          0  free size <file=foo.f90, variable=X, line=15> : MAIN => FOO => BAR
         1        180        180        180          0  malloc size <file=foo.f90, variable=X, line=13>
         1        180        180        180          0  malloc size <file=foo.f90, variable=X, line=13> : MAIN => FOO => BAR
         4         80         16         49      22.69  malloc size <file=foo.f90, variable=X, line=7>
         1         16         16         16          0  malloc size <file=foo.f90, variable=X, line=7> : MAIN => FOO => BAR
         3         80         48         60      14.24  malloc size <file=foo.f90, variable=X, line=7> : MAIN => FOO => G => BAR
---------------------------------------------------------------------------------------
                                                                                    1,1           All
```

Interval Event

Context Event

Atomic Event

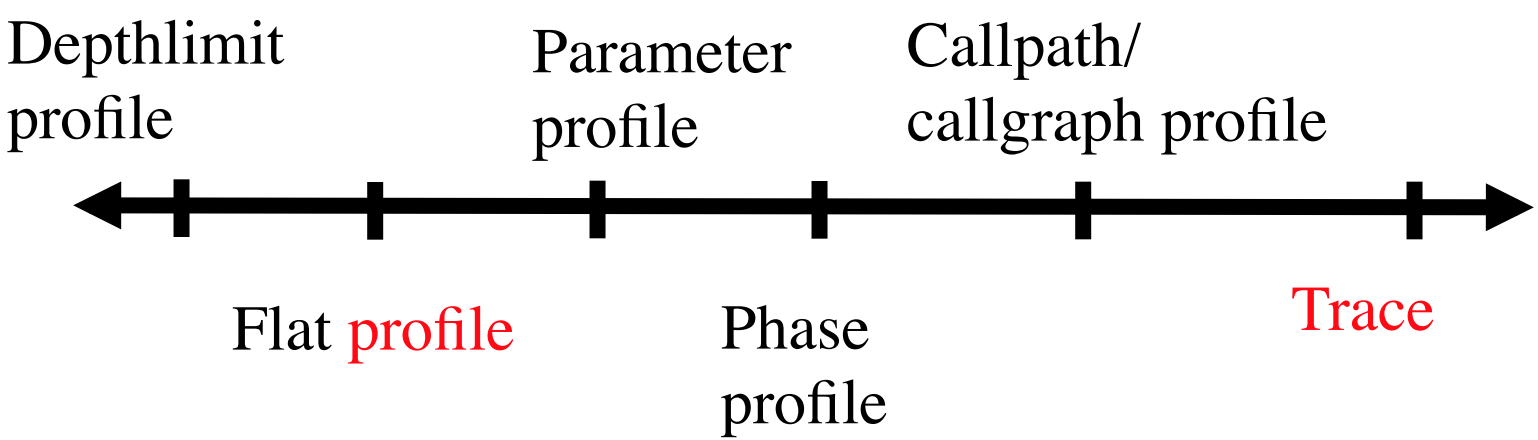
TAU Measurement Mechanisms

- **Parallel profiling**
  - Function-level, block-level, statement-level
  - Supports user-defined events and mapping events
  - Support for flat, callgraph/callpath, phase profiling
  - Support for memory profiling (headroom, malloc/leaks)
  - Support for tracking I/O (wrappers, read/write/print calls)
  - Parallel profiles written at end of execution
  - Parallel profile snapshots can be taken during execution
- **Tracing**
  - All profile-level events + inter-process communication
  - Inclusion of multiple counter data in traced events

Types of Parallel Performance Profiling

- ***Flat*** profiles
  - Metric (e.g., time) spent in an event (callgraph nodes)
  - Exclusive/inclusive, # of calls, child calls
- ***Callpath*** profiles (***Calldepth*** profiles)
  - Time spent along a calling path (edges in callgraph)
  - "*main=> f1 => f2 => MPI_Send*" (event name)
  - `TAU_CALLPATH_DEPTH` environment variable
- ***Phase*** profiles
  - Flat profiles under a phase (nested phases are allowed)
  - Default "main" phase
  - Supports static or dynamic (e.g., per-iteration) phases

Performance Evaluation Alternatives



Each alternative has:
- one metric/counter
- multiple counters

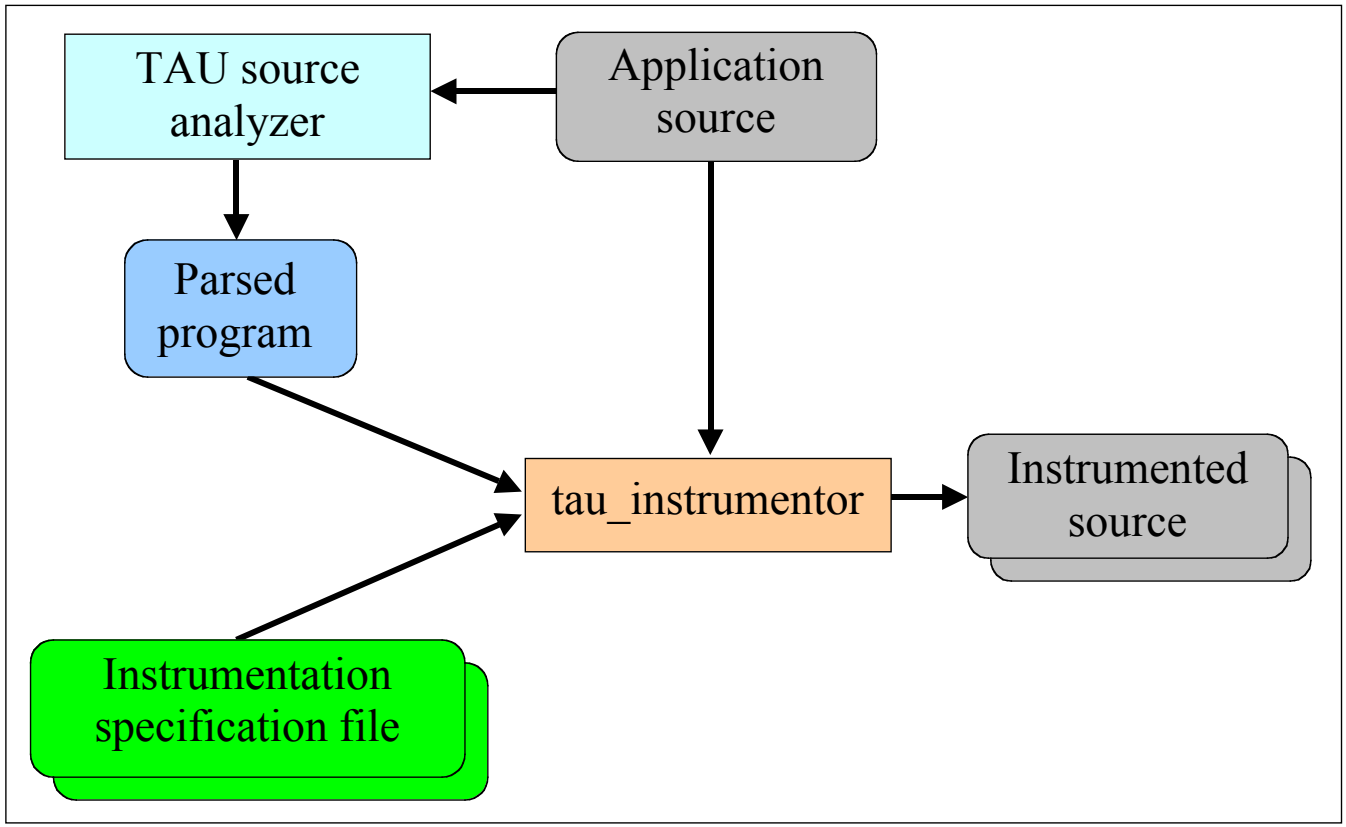
Volume of performance data

TAU: A Quick Reference

Runtime Environment Variables in TAU

Environment Variable	Default	Description
TAU_TRACE	0	Setting to 1 turns on tracing
TAU_CALLPATH	0	Setting to 1 turns on callpath profiling
TAU_TRACK_MEMORY_LEAKS	0	Setting to 1 turns on leak detection
TAU_TRACK_HEAP or TAU_TRACK_HEADROOM	0	Setting to 1 turns on tracking heap memory/headroom at routine entry & exit using context events (e.g., Heap at Entry: main=>foo=>bar)
TAU_CALLPATH_DEPTH	2	Specifies depth of callpath. Setting to 0 generates no callpath or routine information, setting to 1 generates flat profile and context events have just parent information (e.g., Heap Entry: foo)
TAU_TRACK_SIGNALS	0	Setting to 1 generate debugging callstack info when a program crashes
TAU_SAMPLING	0	Setting to 1 generates sample based profilles
TAU_COMM_MATRIX	0	Setting to 1 generates communication matrix display using context events
TAU_THROTTLE	1	Setting to 0 turns off throttling. Enabled by default to remove instrumentation in lightweight routines that are called frequently
TAU_THROTTLE_NUMCALLS	100000	Specifies the number of calls before testing for throttling
TAU_THROTTLE_PERCALL	10	Specifies value in microseconds. Throttle a routine if it is called over 100000 times and takes less than 10 usec of inclusive time per call
TAU_COMPENSATE	0	Setting to 1 enables runtime compensation of instrumentation overhead
TAU_PROFILE_FORMAT	Profile	Setting to “merged” generates a single file - tauprofile.xml. “snapshot” generates xml format
TAU_METRICS	TIME	Setting to a comma separted list generates other metrics. (e.g., --env TIME:BGQ_TIMERS:PAPI_FP_INS:PAPI_NATIVE_<event>)

Automatic Source-Level Instrumentation in TAU using Program Database Toolkit (PDT)



Automatic Instrumentation

- We now provide compiler wrapper scripts
  - Simply replace `CC` with `tau_cxx.sh`
  - Automatically instruments C++ and C source code, links with TAU MPI Wrapper libraries.
- Use `tau_cc.sh` and `tau_f90.sh` for C and Fortran

Before

```
CXX = mpixlcxx_r
F90 = mpixlf90_r
CFLAGS =
LIBS = -lm
OBJS = f1.o f2.o f3.o … fn.o

app: $(OBJS)
      $(CXX) $(LDFLAGS) $(OBJS) -o $@
      $(LIBS)
.cpp.o:
      $(CC) $(CFLAGS) -c $<
```

After

```
CXX = tau_cxx.sh
F90 = tau_f90.sh
CFLAGS =
LIBS = -lm
OBJS = f1.o f2.o f3.o … fn.o

app: $(OBJS)
      $(CXX) $(LDFLAGS) $(OBJS) -o $@
      $(LIBS)
.cpp.o:
      $(CC) $(CFLAGS) -c $<
```

TAU_COMPILER Commandline Options

- See `<taudir>/<arch>/bin/tau_compiler.sh -help`
- Compilation:

```
% mpixlf90_r -c foo.f90
```

Changes to

```
% gfparse foo.f90 $(OPT1)
% tau_instrumentor foo.pdb foo.f90 -o foo.inst.f90 $(OPT2)
% mpixlf90_r -c foo.inst.f90 $(OPT3)
```

- Linking:

```
% mpixlf90_r foo.o bar.o -o app
```

Changes to

```
% mpixlf90_r foo.o bar.o -o app $(OPT4)
```

- Where options OPT[1-4] default values may be overridden by the user:

```
F90 = tau_f90.sh
```

Compile-Time Environment Variables

- Optional parameters for TAU_OPTIONS: [tau_compiler.sh –help]

-optVerbose	Turn on verbose debugging messages
-optCompInst	Use compiler based instrumentation
-optNoCompInst	Do not revert to compiler instrumentation if source instrumentation fails.
-optLinkOnly	Do not instrument the source code. Simply link in the TAU libraries
-optTrackIO	Wrap POSIX I/O call and calculates vol/bw of I/O operations (Requires TAU to be configured with –iowrapper)
-optKeepFiles	Does not remove intermediate .pdb and .inst.* files
-optPreProcess	Preprocess Fortran sources before instrumentation
-optTauSelectFile=""	Specify selective instrumentation file for tau_instrumentor
-optTauWrapFile=""	Specify link_options.tau generated by tau_gen_wrapper
-optLinking=""	Options passed to the linker. Typically $(TAU_MPI_FLIBS) $(TAU_LIBS) $(TAU_CXXLIBS)
-optCompile=""	Options passed to the compiler. Typically $(TAU_MPI_INCLUDE) $(TAU_INCLUDE) $(TAU_DEFS)
-optPdtF95Opts=""	Add options for Fortran parser in PDT (f95parse/gfparse)
-optPdtF95Reset=""	Reset options for Fortran parser in PDT (f95parse/gfparse)
-optPdtCOpts=""	Options for C parser in PDT (cparse). Typically $(TAU_MPI_INCLUDE) $(TAU_INCLUDE) $(TAU_DEFS)
-optPdtCxxOpts=""	Options for C++ parser in PDT (cxxparse). Typically $(TAU_MPI_INCLUDE) $(TAU_INCLUDE) $(TAU_DEFS)

Compiling Fortran Codes with TAU

- If your Fortran code uses free format in .f files (fixed is default for .f), you may use:
  % export TAU_OPTIONS= '-optPdtF95Opts="-R free" -optVerbose '

- To use the compiler based instrumentation instead of PDT (source-based):
  % export TAU_OPTIONS= '-optCompInst -optVerbose'

- If your Fortran code uses C preprocessor directives (#include, #ifdef, #endif):
  % export TAU_OPTIONS= '-optPreProcess -optVerbose -optDetectMemoryLeaks'

- To use an instrumentation specification file:
  % export TAU_OPTIONS= '-optTauSelectFile=mycmd.tau -optVerbose -optPreProcess'
  % cat mycmd.tau
  BEGIN_INSTRUMENT_SECTION
  memory file="foo.f90" routine="#"
  # instruments all allocate/deallocate statements in all routines in foo.f90
  loops file="*" routine="#"
  io file="abc.f90" routine="FOO"
  END_INSTRUMENT_SECTION

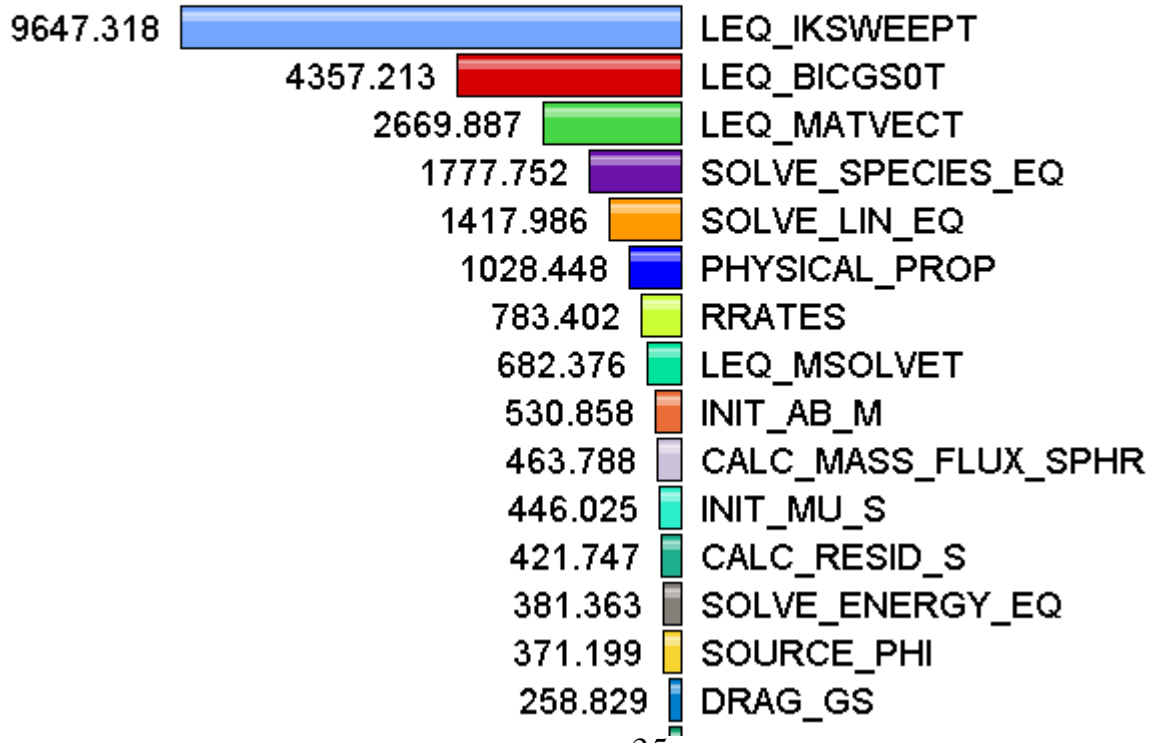
Steps of Performance Evaluation

- Collect basic routine-level timing profile to determine where most time is being spent
- Collect routine-level hardware counter data to determine types of performance problems
- Collect callpath profiles to determine sequence of events causing performance problems
- Conduct finer-grained profiling and/or tracing to pinpoint performance bottlenecks
  - Loop-level profiling with hardware counters
  - Tracing of communication operations

Usage Scenarios: Routine Level Profile

- Goal: What routines account for the most time? How much?
- Flat profile with wallclock time:

Metric: P_VIRTUAL_TIME
Value: Exclusive
Units: seconds



Solution: Generating a flat profile with MPI

```
% export TAU_MAKEFILE=/soft/perftools/tau/tau_latest/
                      /lib/Makefile.tau-bgqtimers-papi-mpi-pdt
% export PATH=/soft/perftools/tau/tau_latest/bgq/bin:$PATH
OR
% soft add +tau_latest
% tau_f90.sh matmult.f90 -o matmult
(Or edit Makefile and change F90=tau_f90.sh)

% qsub -q R.mc -n 4 -t 10 -A MiraCon2013 ./matmult
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.

% paraprof app.ppk
```

Usage Scenarios: Loop Level Instrumentation

- Goal: What loops account for the most time? How much?
- Flat profile with wallclock time with loop instrumentation:

Metric: GET_TIME_OF_DAY
Value: Exclusive
Units: microseconds

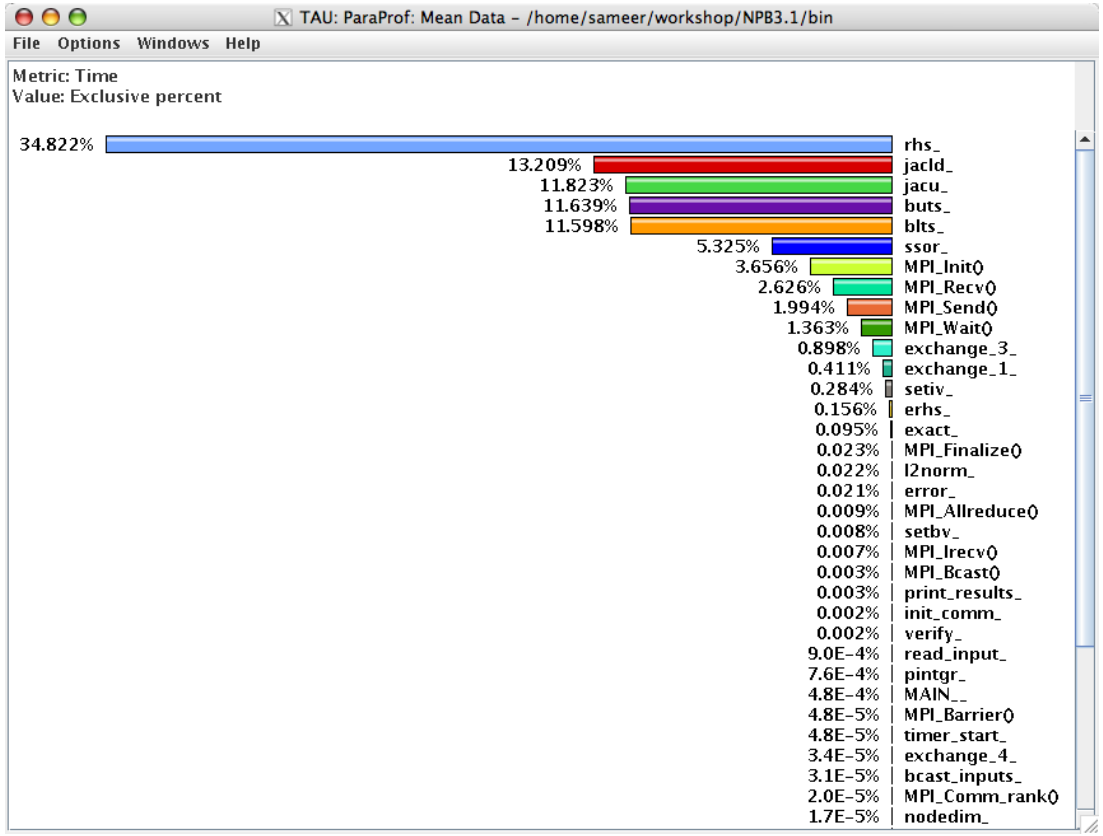
1729975.333	Loop: MULTIPLY_MATRICES [{matmult.f90} {31,9}-{36,14}]
443194	MPI_Recv()
81095	MAIN
49569	MPI_Bcast()
45669	Loop: MAIN [{matmult.f90} {86,9}-{106,14}]
12412	MPI_Send()
8959	Loop: INITIALIZE [{matmult.f90} {17,9}-{21,14}]
8953	Loop: INITIALIZE [{matmult.f90} {10,9}-{14,14}]
5609.2	MPI_Finalize()
2932.667	MULTIPLY_MATRICES
2577.667	Loop: MAIN [{matmult.f90} {117,9}-{128,14}]
2091.8	MPI_Barrier()
1875.667	Loop: MAIN [{matmult.f90} {112,9}-{115,14}]
1833	Loop: MAIN [{matmult.f90} {71,9}-{74,14}]
107	Loop: MAIN [{matmult.f90} {77,9}-{84,14}]
30	INITIALIZE
14.25	MPI_Comm_rank()
1	MPI_Comm_size()

Solution: Generating a loop level profile

```
% export TAU_MAKEFILE=/soft/perftools/tau/tau_latest/bgq
                     /lib/Makefile.tau-bgqtimers-papi-mpi-pdt
% export TAU_OPTIONS='-optTauSelectFile=select.tau -optVerbose'
% cat select.tau
  BEGIN_INSTRUMENT_SECTION
  loops routine="#"
  END_INSTRUMENT_SECTION

% export PATH=/soft/perftools/tau/tau_latest/ppc64/bin:$PATH
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% qsub -q R.mc -n 4 -t 10 -A MiraCon2013 ./a.out
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.

% paraprof app.ppk
```

Usage Scenarios: Compiler-based Instrumentation

- Goal: Easily generate routine level performance data using the compiler instead of PDT for parsing the source code

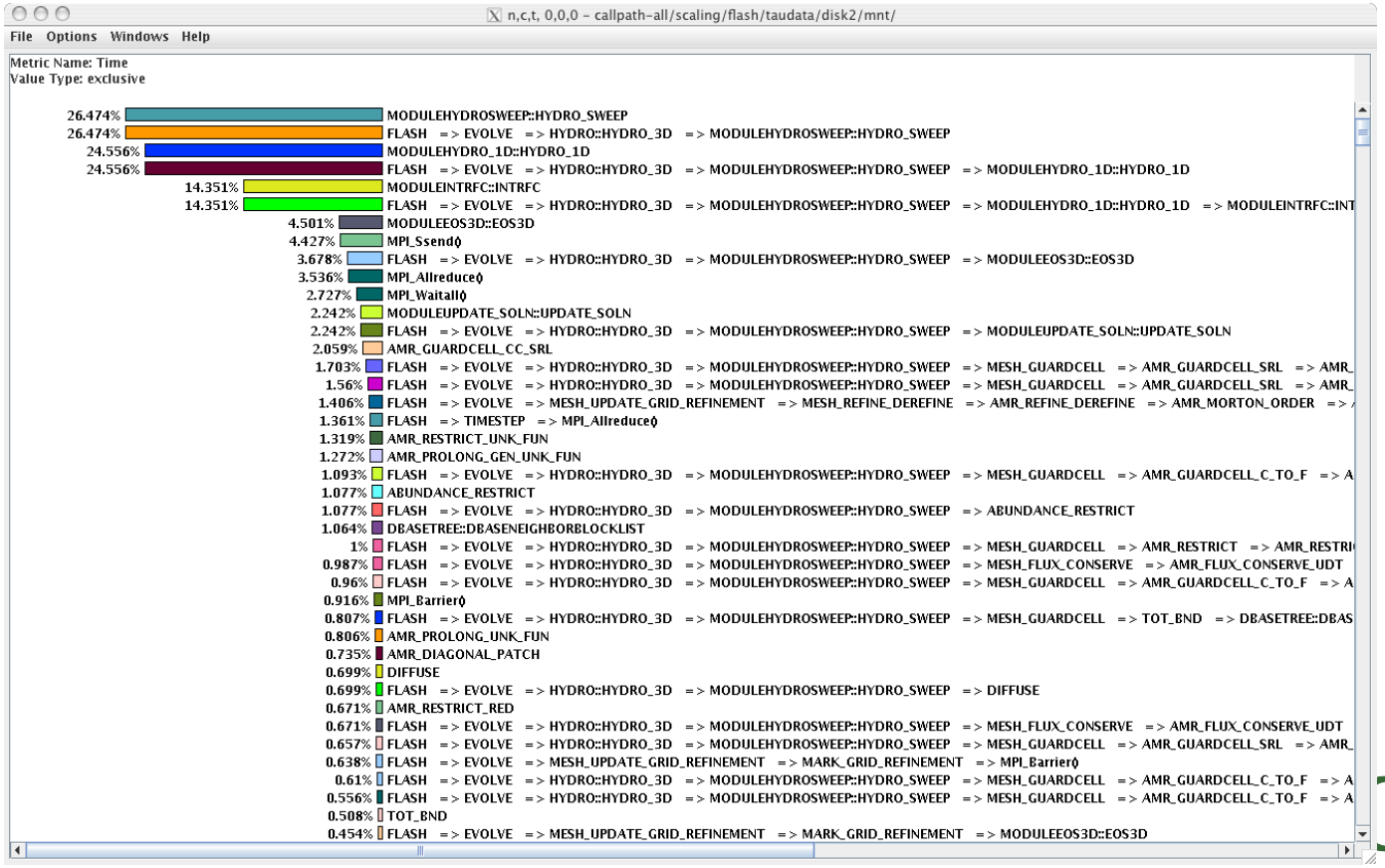


Use Compiler-Based Instrumentation

```
% export TAU_MAKEFILE=/soft/perftools/tau/tau_latest/bgq
                /lib/Makefile.tau-mpi
% export TAU_OPTIONS='-optCompInst -optVerbose'
% export PATH=/soft/perftools/tau/tau_latest/ppc64/bin:$PATH
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)

% qsub --mode c8 ./a.out
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk
```

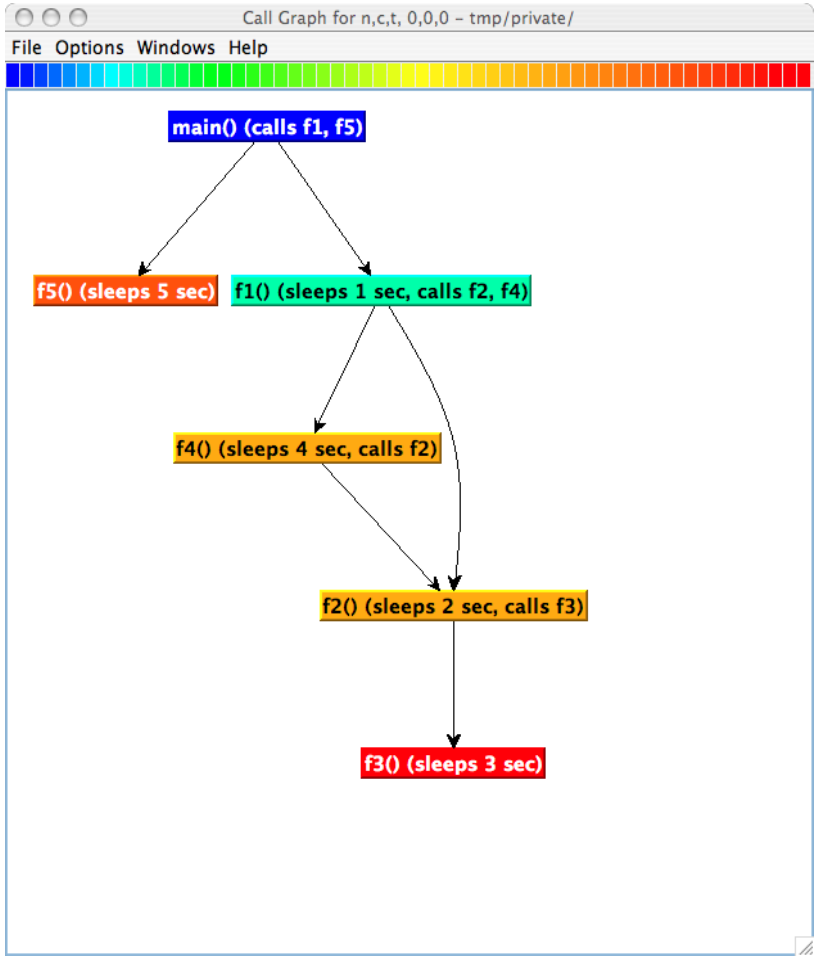
Usage Scenarios: Generating Callpath Profile

- Goal: Who calls my MPI_Barrier()? Where?
- Callpath profile for a given callpath depth:



Callpath Profile

- Generates program callgraph



Generate a Callpath Profile

```
% export TAU_MAKEFILE=/soft/perftools/tau/tau_latest/bgq
                              /lib/Makefile.tau-bgqtimers-papi-mpi-pdt
% export PATH=/soft/perftools/tau/tau_latest/ppc64/bin:$PATH
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% export TAU_CALLPATH=1
% export TAU_CALLPATH_DEPTH=100

% qsub -q R.mc -n 4 -t 10 -A <account> ./a.out
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk
(Windows -> Thread -> Call Graph)
```

Usage Scenarios: Instrument a Python program

- Goal: Generate a flat profile for a Python program

TAU: ParaProf: n,c,t 0,0,0 - flat.ppk

Metric: Time
Value: Exclusive
Units: seconds

5.473	dot
2.548	standard_normal
0.152	MachAr::__init__ [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/lib/machar.py} {50}]
0.112	_parse [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {385}]
0.105	_compile [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_compile.py} {38}]
0.056	any [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/fromnumeric.py} {540}]
0.048	append
0.046	len
0.046	Tokenizer::__next [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {188}]
0.045	any
0.035	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/linalg/linalg.py} {7}]
0.033	SubPattern::getwidth [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {146}]
0.033	_optimize_charset [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_compile.py} {213}]
0.028	Tokenizer::get [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {207}]
0.023	_parse_sub [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {307}]
0.022	add_newdoc [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/lib/function_base.py} {1154}]
0.019	abs
0.019	_compile_charset [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_compile.py} {184}]
0.016	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/__init__.py} {2}]
0.015	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/numeric.py} {1}]
0.014	SubPattern::__len__ [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {132}]
0.011	_mk_bitmap [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_compile.py} {264}]
0.011	Tokenizer::match [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {201}]
0.01	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/defmatrix.py} {1}]
0.009	SubPattern::__getitem__ [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {136}]
0.009	_simple [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_compile.py} {360}]
0.008	SubPattern::append [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {144}]
0.007	min
0.007	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/ctypes/__init__.py} {4}]
0.007	extend_all [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/numeric.py} {83}]
0.006	ord
0.005	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/__init__.py} {17}]
0.005	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/ma.py} {9}]
0.005	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/numerictypes.py} {76}]
0.005	SubPattern::__getslice__ [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {140}]
0.004	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/lib/index_tricks.py} {3}]
0.004	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/lib/__init__.py} {1}]
0.004	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/random/__init__.py} {2}]
0.004	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/opcode.py} {5}]
0.004	_extendLine [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/arrayprint.py} {270}]
0.004	_formatArray [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/arrayprint.py} {278}]

Usage Scenarios: Instrument a Python program

Original code:

```
% cat foo.py
#!/usr/bin/env python
import numpy
ra=numpy.random
la=numpy.linalg

size=2000
a=ra.standard_normal((size,size))
b=ra.standard_normal((size,size))
c=la.linalg.dot(a,b)
print c
```

Create a wrapper:

```
% cat wrapper.py
#!/usr/bin/env python

# setenv PYTHONPATH $PET_HOME/pkgs/tau-2.17.3/ppc64/lib/bindings-gnu-python-pdt

import tau

def OurMain():
    import foo

tau.run('OurMain()')
```

Generate a Python Profile

```
% export TAU_MAKEFILE=/soft/perftools/tau/tau_latest/bgq
                /lib/Makefile.tau-python-pdt
% export PATH=/soft/perftools/tau/tau_latest/ppc64/bin:$PATH
% cat wrapper.py
  import tau
  def OurMain():
      import foo
  tau.run('OurMain()')
Uninstrumented:
% ./foo.py
Instrumented:
% export PYTHONPATH= <taudir>/bgq/<lib>/bindings-python-pdt
(same options string as TAU_MAKEFILE)
% export LD_LIBRARY_PATH=<taudir>/bgq/lib/bindings-python-pdt:
$LD_LIBRARY_PATH
% ./wrapper.py

Wrapper invokes foo and generates performance data
% pprof/paraprof
```

Usage Scenarios: Mixed Python+F90+C+pyMPI

- Goal: Generate multi-level instrumentation for Python+MPI+C+F90+C++ ...

TAU: ParaProf: Mean Data - py.ppk

Metric: Time
Value: Exclusive percent

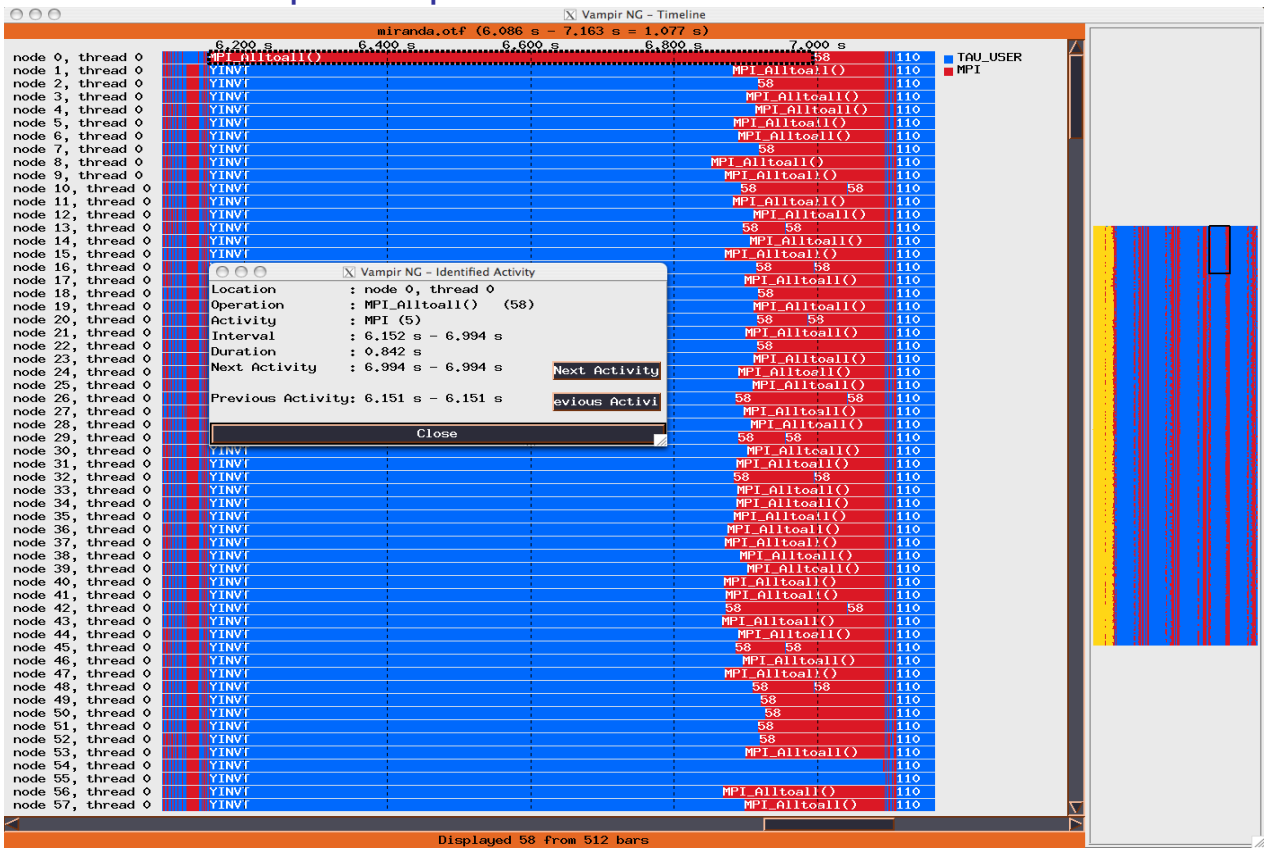
39.466%	void SAMINT::timestep(double, double) [{SAMINT.C} {72,1}-{78,1}]
6.563%	int pyMPI_Main_with_communicator(int, int *, char ***, MPI_Comm) C [{pyMPI_main.c} {23,1}-{80,1}]
5.18%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/linalg/linalg.py} {7}]
4.91%	void pyMPI_distutils_init(PyObject **) C [{pyMPI_distutils.c} {275,1}-{400,1}]
3.845%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/numeric.py} {1}]
3.633%	void pyMPI_site(void) C [{pyMPI_site.c} {23,1}-{121,1}]
3.607%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/__init__.py} {2}]
3.1%	MachAr::__init__ [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/lib/machar.py} {50}]
2.261%	_parse [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {385}]
2.123%	_compile [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_compile.py} {38}]
1.187%	any [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/fromnumeric.py} {540}]
1.034%	<module> [{/users3/sameer/py/py-c++-f90/example/samarcrun.py} {5}]
0.982%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/__init__.py} {17}]
0.941%	Tokenizer::__next [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {188}]
0.929%	append
0.906%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/ctypes/__init__.py} {4}]
0.895%	any
0.888%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/lib/__init__.py} {1}]
0.884%	len
0.863%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/lib/index_tricks.py} {3}]
0.806%	void pyMPI_pickle_init(PyObject **) C [{pyMPI_pickle.c} {43,1}-{61,1}]
0.739%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/random/__init__.py} {2}]
0.67%	SubPattern::getwidth [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {146}]
0.658%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/testing/numpytest.py} {1}]
0.643%	_optimize_charset [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_compile.py} {213}]
0.584%	Tokenizer::get [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {207}]
0.552%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/inspect.py} {24}]
0.541%	<module> [{/users3/sameer/py/py-c++-f90/samint.py} {7}]
0.537%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/memmap.py} {1}]
0.537%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/testing/__init__.py} {2}]
0.532%	<module> [{/users3/sameer/py/py-c++-f90/samarc.py} {9}]
0.475%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/ctypeslib.py} {1}]
0.462%	add_newdoc [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/lib/function_base.py} {1154}]
0.453%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/lib/function_base.py} {1}]
0.451%	_parse_sub [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {307}]
0.378%	_compile_charset [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_compile.py} {184}]
0.367%	abs
0.365%	samarc::runStep [{/users3/sameer/py/py-c++-f90/samarc.py} {105}]
0.365%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/fft/fftpack.py} {21}]
0.296%	SubPattern::__len__ [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {132}]
0.287%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/arrayprint.py} {4}]
0.27%	MPI_Finalize()
0.225%	Tokenizer::match [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {201}]
0.213%	_mk_bitmap [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_compile.py} {264}]
0.199%	<module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/defmatrix.py} {1}]
0.18%	MPI_Init()
0.178%	SubPattern::__getitem__ [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {136}]

Generate a Multi-Language Profile w/ Python

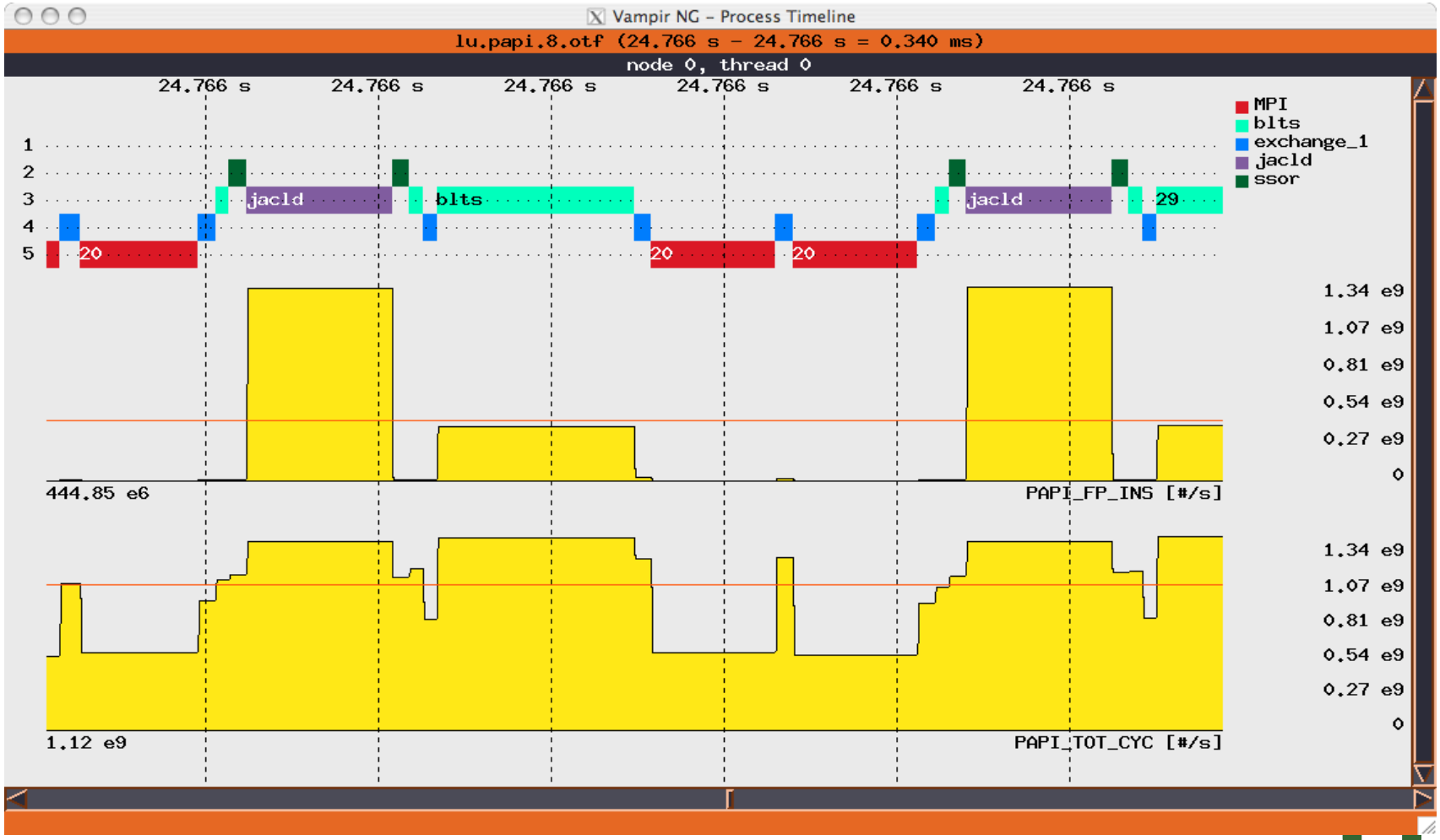
```
% export TAU_MAKEFILE=/soft/perftools/tau/tau_latest/bgq
                /lib/Makefile.tau-python-mpi-pdt
% export PATH=/soft/perftools/tau/tau_latest/ppc64/bin:$PATH
% export TAU_OPTIONS='-optShared -optVerbose…'
(Python needs shared object based TAU library)
% make F90=tau_f90.sh CXX=tau_cxx.sh CC=tau_cc.sh  (build libs, pyMPI w/TAU)
% cat wrapper.py
  import tau
  def OurMain():
      import App
  tau.run('OurMain()')
Uninstrumented:
% qsub -n 4 <dir>/pyMPI-2.5b0/bin/pyMPI ./App.py
Instrumented:
% export PYTHONPATH=<taudir>/bgq/<lib>/bindings-python-mpi-pdt
(same options string as TAU_MAKEFILE)
% export LD_LIBRARY_PATH=<taudir>/bgq/lib/bindings-python-mpi-pdt:
$LD_LIBRARY_PATH
% qsub -a -n 4 <dir>/pyMPI-2.5b0-TAU/bin/pyMPI ./wrapper.py
(Instrumented pyMPI with wrapper.py)
```

Usage Scenarios: Generating a Trace File

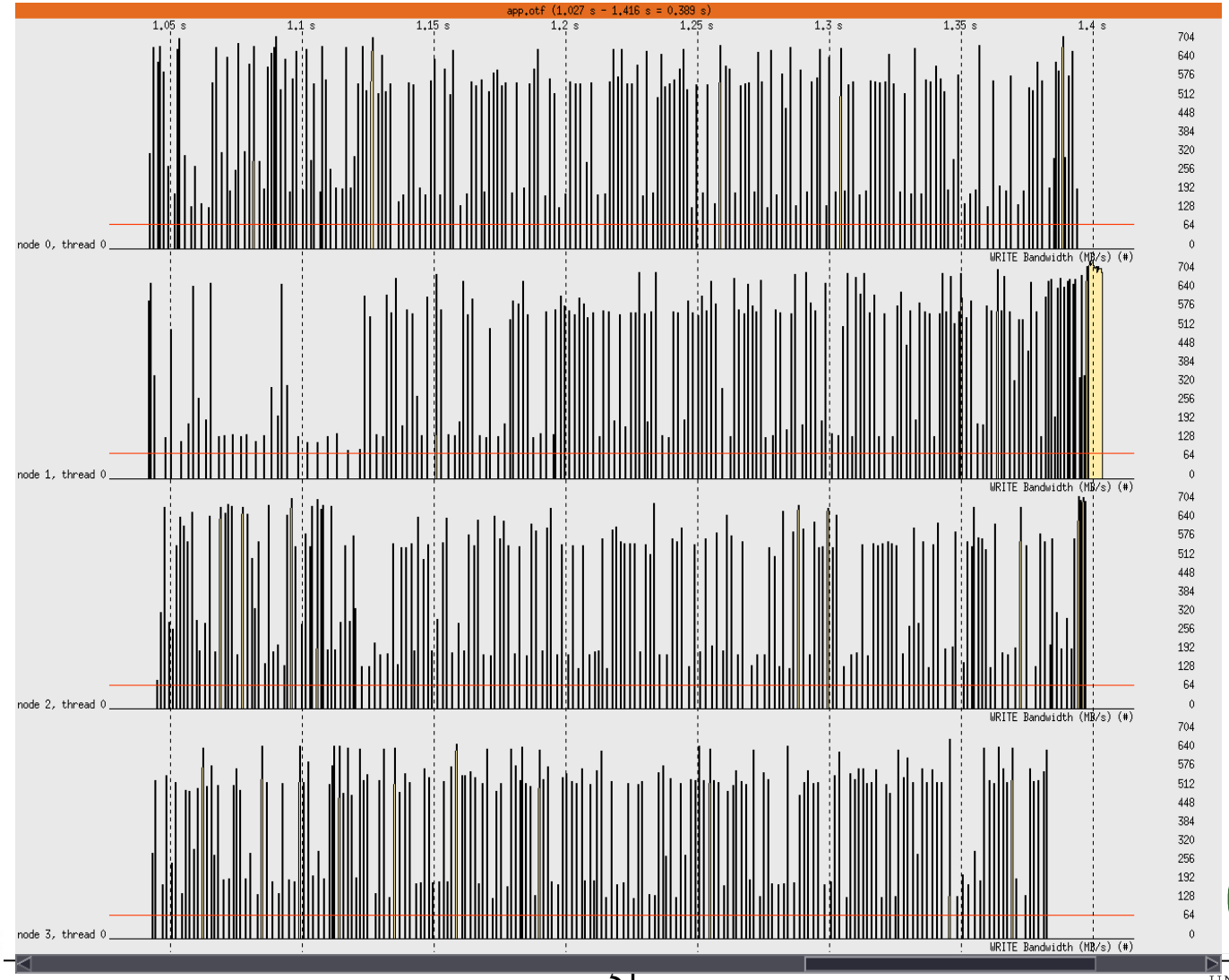
- Goal: Identify the temporal aspect of performance. What happens in my code at a given time? When?
- Event trace visualized in Vampir/Jumpshot



VNG Process Timeline with PAPI Counters



Vampir Counter Timeline Showing I/O BW



ParaTools

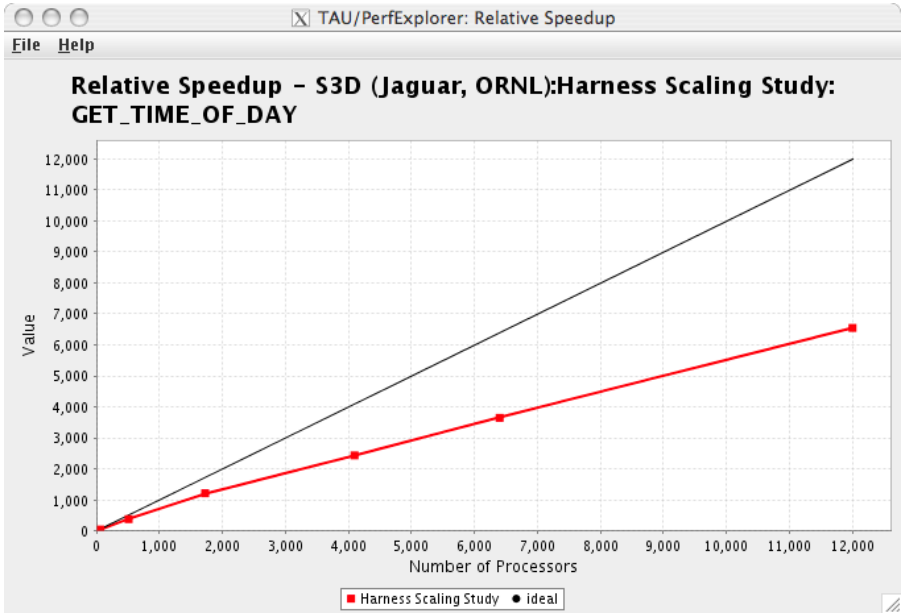
UNIVERSITY OF OREGON

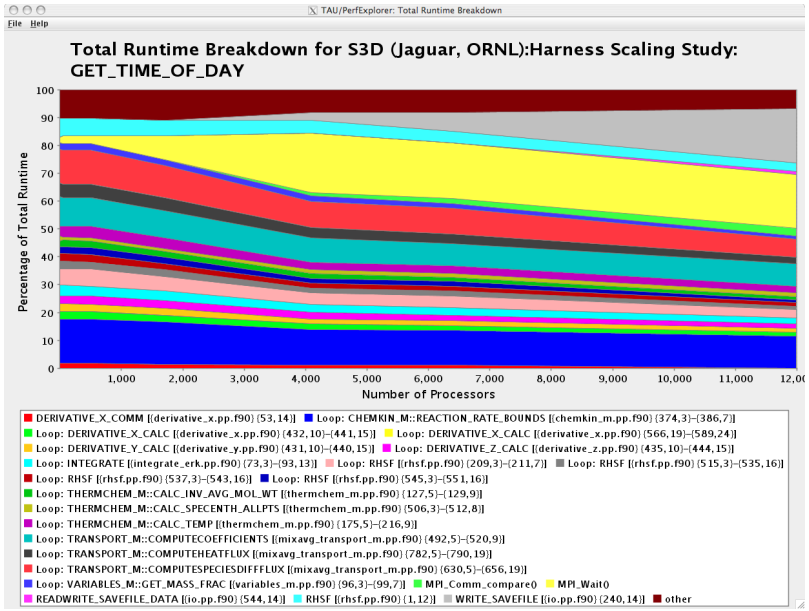
Generate a Trace File

```
% export TAU_MAKEFILE=/soft/perftools/tau/tau_latest/bgq
              /lib/Makefile.tau-bgqtimers-papi-mpi-pdt
% export PATH=/soft/perftools/tau/tau_latest/ppc64/bin:
$PATH
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% export TAU_TRACE=1
% qsub -q R.mc -n 4 -t 10 -A MiraCon2013  ./a.out
% tau_treemerge.pl
(merges binary traces to create tau.trc and tau.edf files)
JUMPSHOT:
% tau2slog2 tau.trc tau.edf -o app.slog2
% jumpshot app.slog2
   OR
VAMPIR:
% tau2otf tau.trc tau.edf app.otf -n 4 -z
(4 streams, compressed output trace)
% vampir app.otf
(or vng client with vngd server).
```

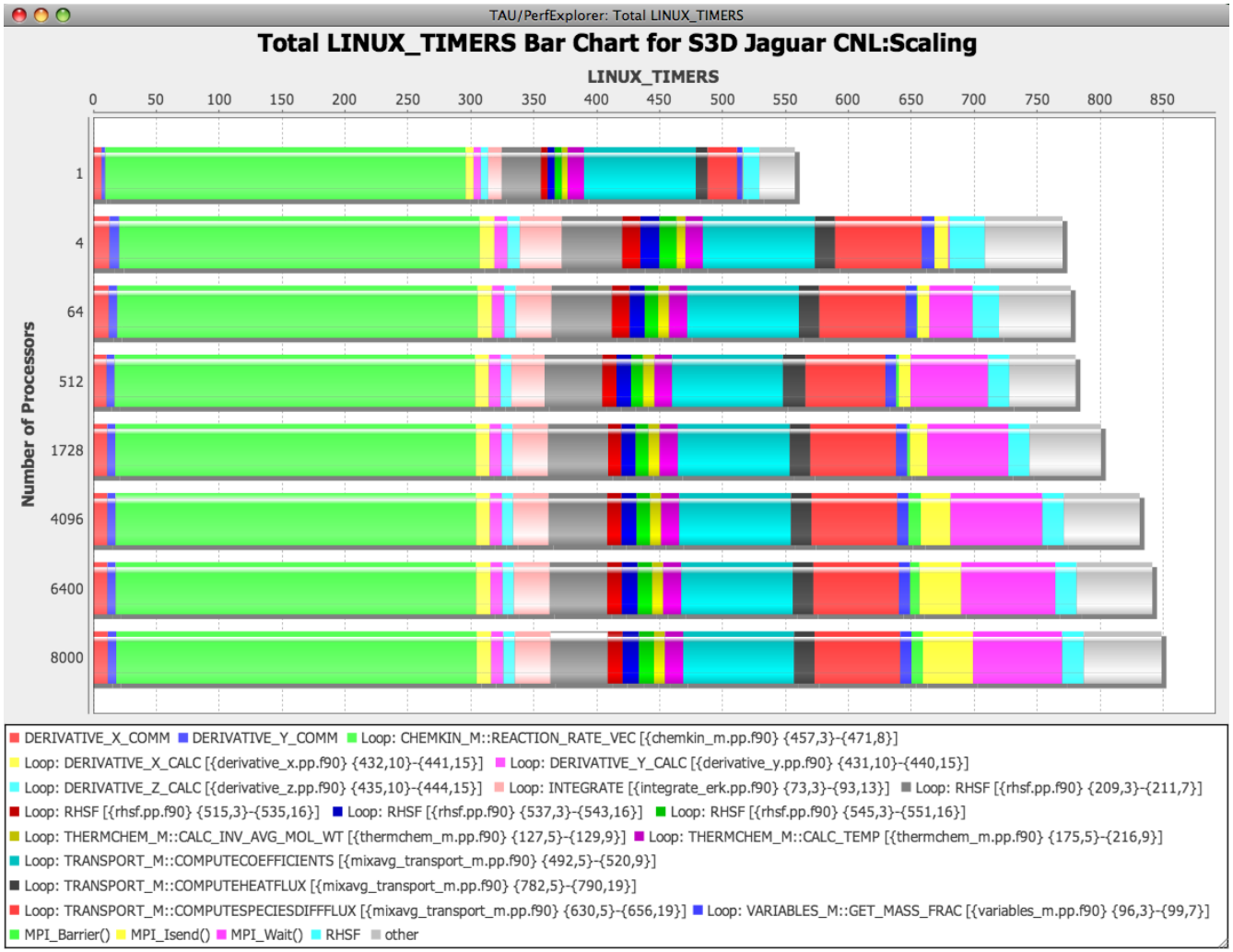
Usage Scenarios: Evaluate Scalability

- Goal: How does my application scale? What bottlenecks occur at what core counts?
- Load profiles in PerfDMF database and examine with PerfExplorer

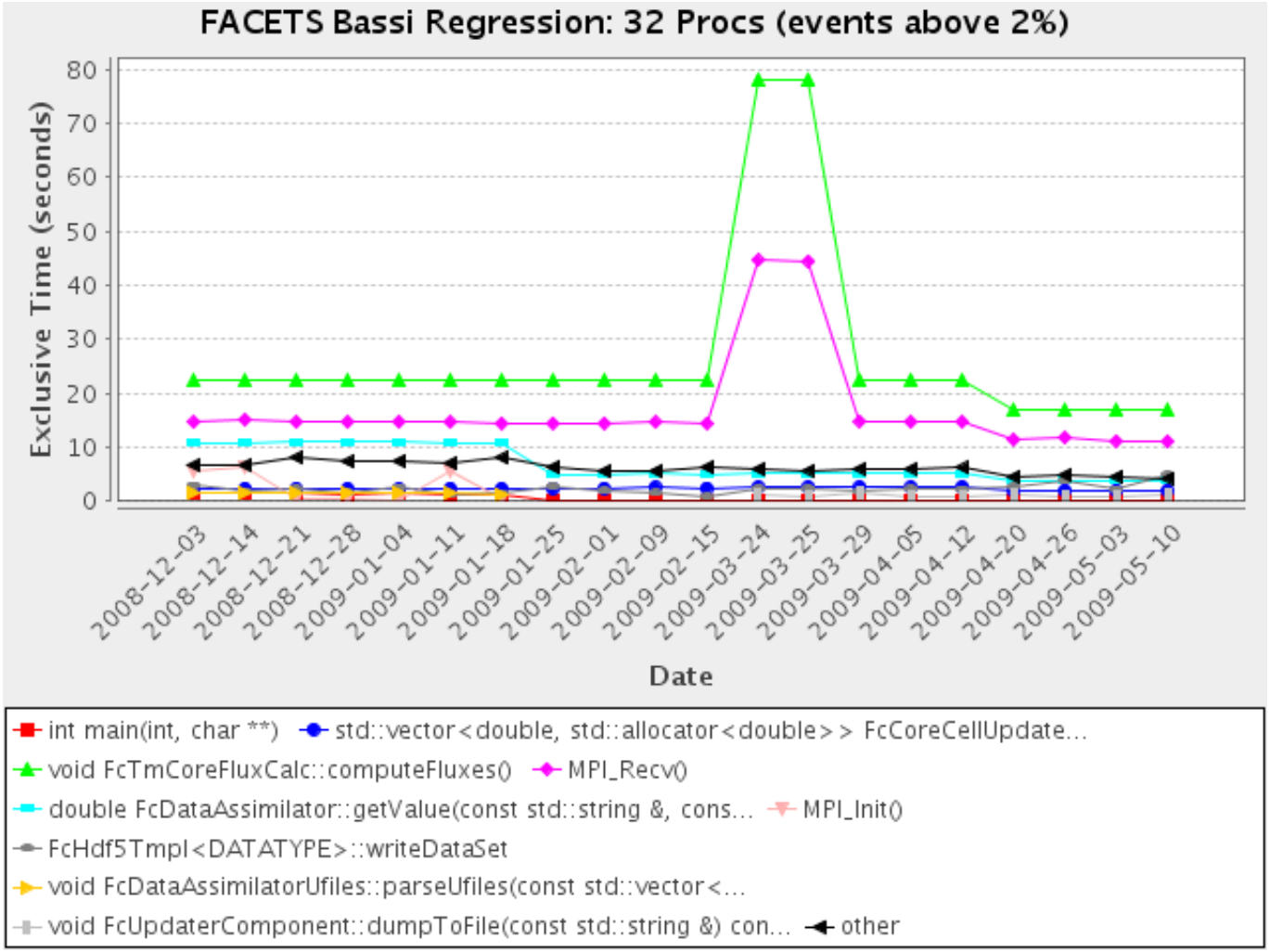




Usage Scenarios: Evaluate Scalability



Performance Regression Testing

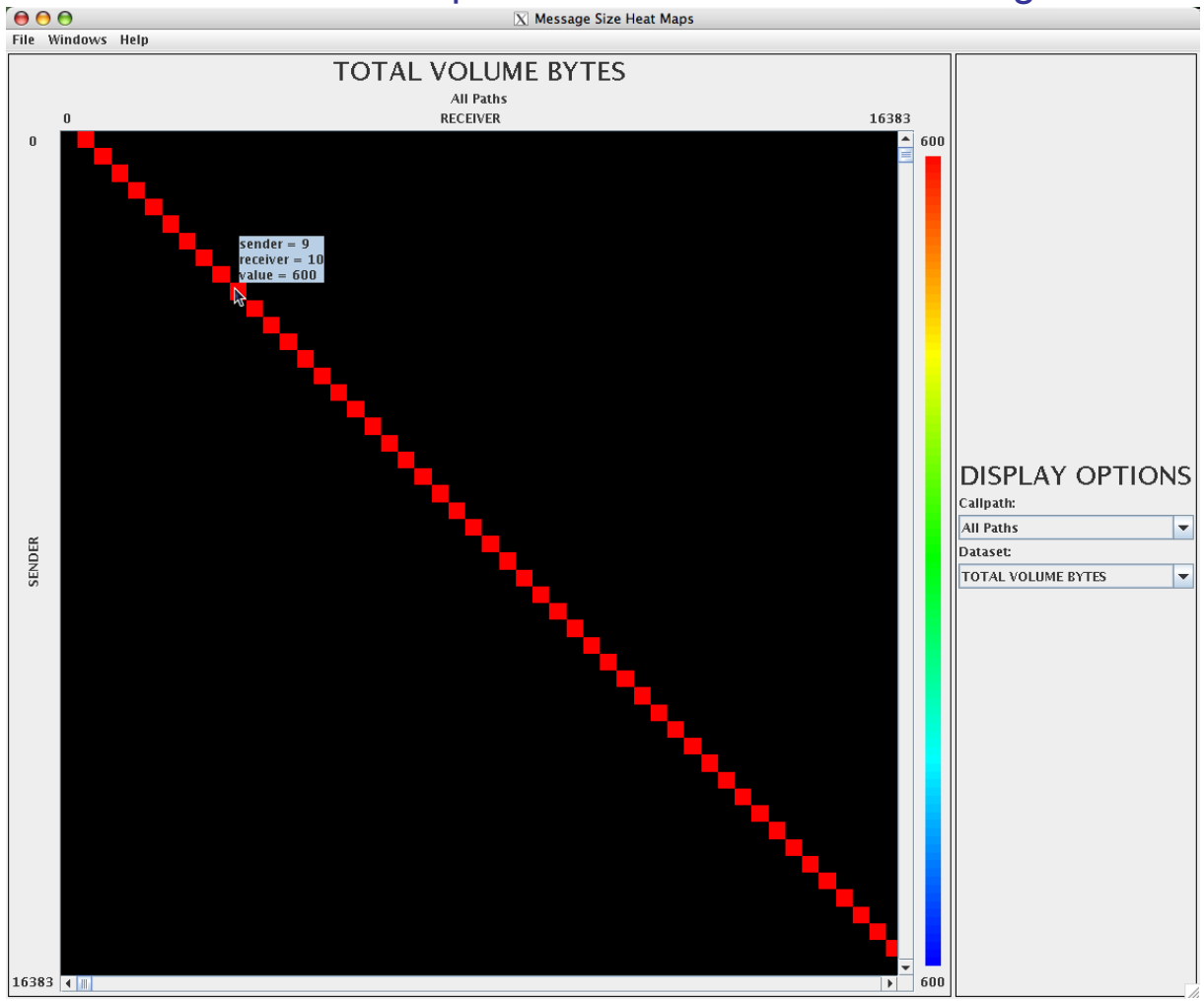


Evaluate Scalability using PerfExplorer Charts

```
% export TAU_MAKEFILE=/soft/perftools/tau/tau_latest/bgq
                              /lib/Makefile.tau-bgqtimers-papi-mpi-pdt

% export PATH=/soft/perftools/tau/tau_latest/ppc64/bin:$PATH

% make F90=tau_f90.sh

(Or edit Makefile and change F90=tau_f90.sh)

% mpirun -np 1 ./a.out

% paraprof --pack 1p.ppk

% mpirun -np 2 ./a.out …

% paraprof --pack 2p.ppk … and so on.

On your client:

% taudb_configure --create-default

(Chooses derby, blank user/passwd, yes to save passwd, defaults)

% perfexplorer_configure

(Yes to load schema, defaults)

% paraprof

(load each trial: DB -> Add Trial -> Type (Paraprof Packed Profile) -> OK) OR use
    taudb_loadtrial

Then,

% perfexplorer

(Select experiment, Menu: Charts -> Speedup)
```

Communication Matrix Display

- Goal: What is the volume of inter-process communication? Along which calling path?

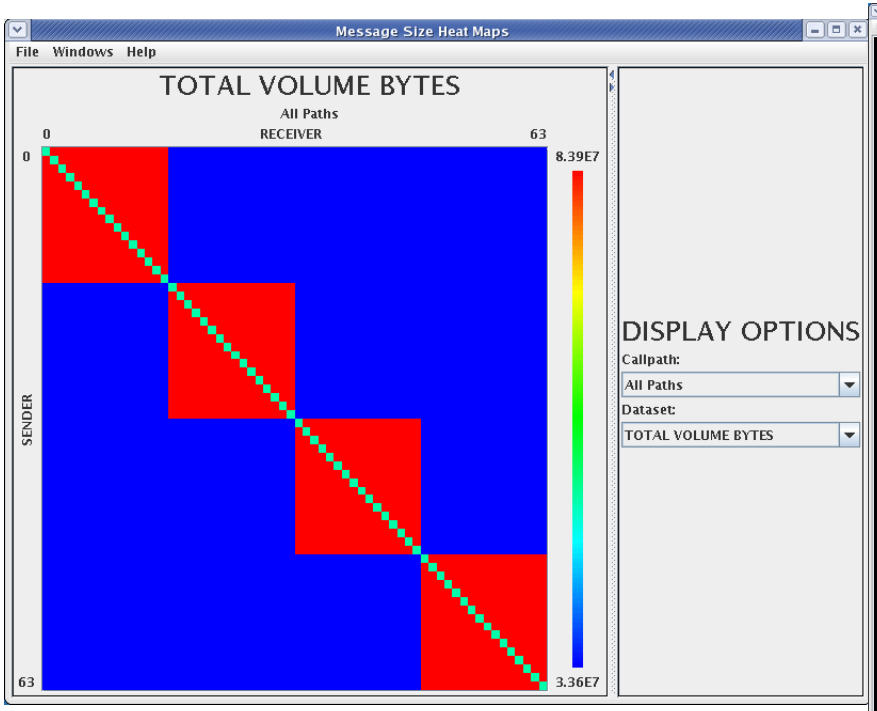


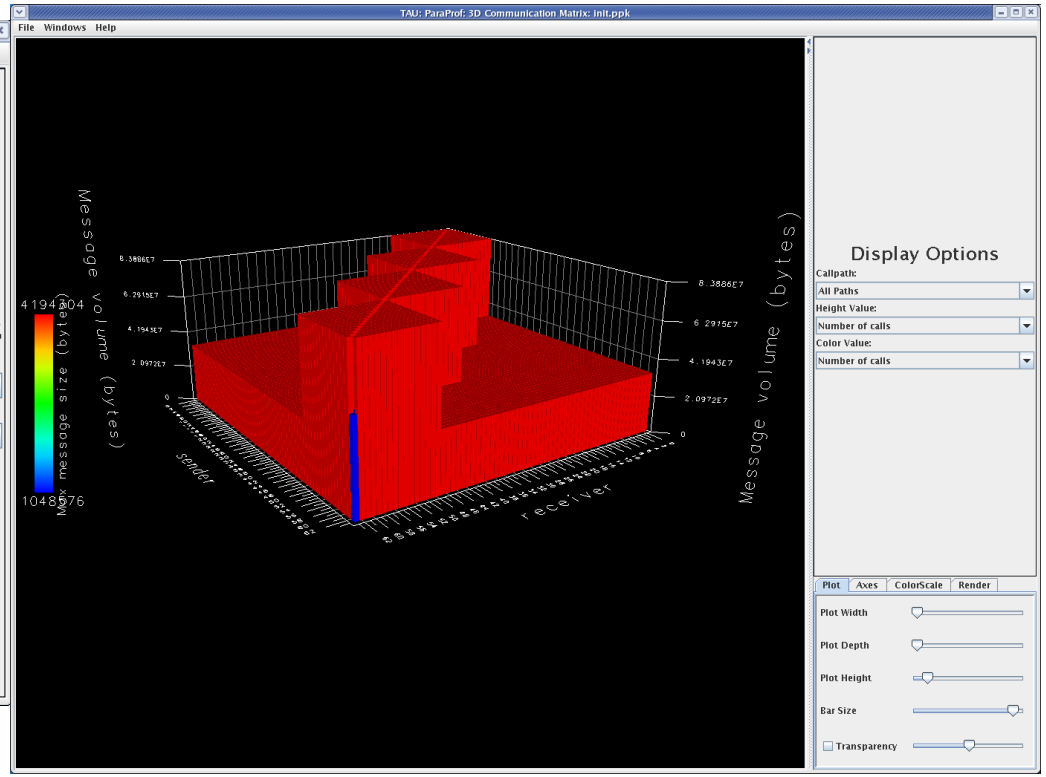
Evaluate Scalability using PerfExplorer Charts

```
% export TAU_MAKEFILE=/soft/perftools/tau/tau_latest/bgq
                              /lib/Makefile.tau-bgqtimers-papi-mpi-pdt

% export PATH=/soft/perftools/tau/tau_latest/ppc64/bin:$PATH

% make F90=tau_f90.sh

(Or edit Makefile and change F90=tau_f90.sh)

% export TAU_COMM_MATRIX=1


% qsub --mode c8 ./a.out (setting the environment variables)


% paraprof

(Windows -> Communication Matrix, Windows -> 3D Communication Matrix)
```

Communication Matrix Display

- Goal: What is the volume of inter-process communication? Along which calling path?





Interval Events, Atomic Events in TAU

```
xterm
NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        0.187        1,105           1          44    1105659 int main(int, char **) C
 93.2        1,030        1,030           1           0    1030654 MPI_Init()
  5.9        0.879           65          40         320       1637 void func(int, int) C
  4.6           51           51          40           0       1277 MPI_Barrier()
  1.2           13           13         120           0        111 MPI_Recv()
  0.8            9            9           1           0       9328 MPI_Finalize()
  0.0        0.137        0.137         120           0          1 MPI_Send()
  0.0        0.086        0.086          40           0          2 MPI_Bcast()
  0.0        0.002        0.002           1           0          2 MPI_Comm_size()
  0.0        0.001        0.001           1           0          1 MPI_Comm_rank()
---------------------------------------------------------------------------------------

USER EVENTS Profile :NODE 0, CONTEXT 0, THREAD 0
---------------------------------------------------------------------------------------
NumSamples   MaxValue   MinValue  MeanValue  Std. Dev.  Event Name
---------------------------------------------------------------------------------------
       365  5.138E+04      44.39   3.09E+04  1.234E+04  Heap Memory Used (KB) : Entry
       365  5.138E+04       2064  3.115E+04   1.21E+04  Heap Memory Used (KB) : Exit
        40         40         40         40          0  Message size for broadcast
---------------------------------------------------------------------------------------
                                                                  27,1          1%
```

Interval event e.g., routines (start/stop)

Atomic events (trigger with value)

% setenv TAU_CALLPATH_DEPTH 0
% setenv TAU_TRACK_HEAP 1

Context Events (default)

```
xterm
NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        0.357        1,114           1          44    1114040 int main(int, char **) C
 92.6        1,031        1,031           1           0    1031066 MPI_Init()
  6.7           72           74          40         320       1865 void func(int, int) C
  0.7            8            8           1           0       8002 MPI_Finalize()
  0.1            1            1         120           0         12 MPI_Recv()
  0.1        0.608        0.608          40           0         15 MPI_Barrier()
  0.0        0.136        0.136         120           0          1 MPI_Send()
  0.0        0.095        0.095          40           0          2 MPI_Bcast()
  0.0        0.001        0.001           1           0          1 MPI_Comm_size()
  0.0            0            0           1           0          0 MPI_Comm_rank()
---------------------------------------------------------------------------------------

USER EVENTS Profile :NODE 0, CONTEXT 0, THREAD 0
---------------------------------------------------------------------------------------
NumSamples   MaxValue   MinValue  MeanValue  Std. Dev.  Event Name
---------------------------------------------------------------------------------------
       365  5.139E+04      44.39  3.091E+04  1.234E+04  Heap Memory Used (KB) : Entry
         1      44.39      44.39      44.39          0  Heap Memory Used (KB) : Entry : int main(int, char **) C
         1       2068       2068       2068          0  Heap Memory Used (KB) : Entry : int main(int, char **) C => MPI_Comm_rank()
         1       2066       2066       2066          0  Heap Memory Used (KB) : Entry : int main(int, char **) C => MPI_Comm_size()
         1  5.139E+04  5.139E+04  5.139E+04          0  Heap Memory Used (KB) : Entry : int main(int, char **) C => MPI_Finalize()
         1      57.58      57.58      57.58          0  Heap Memory Used (KB) : Entry : int main(int, char **) C => MPI_Init()
        40  5.036E+04       2069  3.011E+04  1.228E+04  Heap Memory Used (KB) : Entry : int main(int, char **) C => void func(int, int) C
        40  5.139E+04       3098  3.114E+04  1.227E+04  Heap Memory Used (KB) : Entry : void func(int, int) C => MPI_Barrier()
        40  5.139E+04   1.13E+04  3.134E+04  1.187E+04  Heap Memory Used (KB) : Entry : void func(int, int) C => MPI_Bcast()
       120  5.139E+04   1.13E+04  3.134E+04  1.187E+04  Heap Memory Used (KB) : Entry : void func(int, int) C => MPI_Recv()
       120  5.139E+04   1.13E+04  3.134E+04  1.187E+04  Heap Memory Used (KB) : Entry : void func(int, int) C => MPI_Send()
       365  5.139E+04       2065  3.116E+04   1.21E+04  Heap Memory Used (KB) : Exit
                                                                                    3.7
```

Context event = atomic event + executing context

% setenv TAU_CALLPATH_DEPTH 2

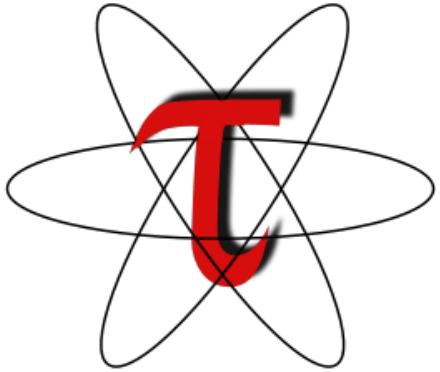
% setenv TAU_TRACK_HEAP 1

Binary Rewriting: DyninstAPI [U.Wisc] and TAU

livetau@paratools01:~

```
/home/livetau% cd ~/tutorial
/home/livetau/tutorial% # Build an uninstrumented bt NAS Parallel Benchmark
/home/livetau/tutorial% make bt CLASS=W NPROCS=4
/home/livetau/tutorial% cd bin
/home/livetau/tutorial/bin% # Run the instrumented code
/home/livetau/tutorial/bin% mpirun -np 4 ./bt_W.4
/home/livetau/tutorial/bin%
/home/livetau/tutorial/bin% # Instrument the executable using TAU with DyninstAPI
/home/livetau/tutorial/bin%
/home/livetau/tutorial/bin% tau_run ./bt_W.4 -o ./bt.i
/home/livetau/tutorial/bin% rm -rf profile.* MULT*
/home/livetau/tutorial/bin% mpirun -np 4 ./bt.i
/home/livetau/tutorial/bin% paraprof
/home/livetau/tutorial/bin%
/home/livetau/tutorial/bin% # Choose a different TAU configuration
/home/livetau/tutorial/bin% ls $TAU/libTAUsh
libTAUsh-depthlimit-mpi-pdt.so*              libTAUsh-papi-pdt.so*
libTAUsh-mpi-pdt.so*                         libTAUsh-papi-pthread-pdt.so*
libTAUsh-mpi-pdt-upc.so*                     libTAUsh-param-mpi-pdt.so*
libTAUsh-mpi-python-pdt.so*                  libTAUsh-pdt.so*
libTAUsh-papi-mpi-pdt.so*                    libTAUsh-pdt-trace.so*
libTAUsh-papi-mpi-pdt-upc.so*                libTAUsh-phase-papi-mpi-pdt.so*
libTAUsh-papi-mpi-pdt-upc-udp.so*            libTAUsh-pthread-pdt.so*
libTAUsh-papi-mpi-pdt-vampirtrace-trace.so* libTAUsh-python-pdt.so*
libTAUsh-papi-mpi-python-pdt.so*
/home/livetau/tutorial/bin% ls $TAU/libTAUsh-
/home/livetau/tutorial/bin%
/home/livetau/tutorial/bin% tau_run -XrunTAUsh-papi-mpi-pdt-vampirtrace-trace bt_W.4 -o bt.vpt
/home/livetau/tutorial/bin% setenv VT_METRICS PAPI_FP_INS:PAPI_L1_DCM
/home/livetau/tutorial/bin% mpirun -np 4 ./bt.vpt
/home/livetau/tutorial/bin% vampir bt.vpt.otf &
/home/livetau/tutorial/bin%
```

Using PAPI and TAU



Hardware Counters

Hardware performance counters available on most modern microprocessors can provide insight into:

1. Whole program timing
2. Cache behaviors
3. Branch behaviors
4. Memory and resource access patterns
5. Pipeline stalls
6. Floating point efficiency
7. Instructions per cycle

Hardware counter information can be obtained with:

1. Subroutine or basic block resolution
2. Process or thread attribution

What's PAPI?

- Open Source software from U. Tennessee, Knoxville
- http://icl.cs.utk.edu/papi
- Middleware to provide a consistent programming interface for the performance counter hardware found in most major micro-processors.
- Countable events are defined in two ways:
  - Platform-neutral *preset* events
  - Platform-dependent native events
- Presets can be derived from multiple *native events*
- All events are referenced by name and collected in EventSets

PAPI Utilities: *papi_avail*

```
$ utils/papi_avail -h
Usage: utils/papi_avail [options]
Options:

General command options:
    -a, --avail   Display only available preset events
    -d, --detail  Display detailed information about all preset events
    -e EVENTNAME  Display detail information about specified preset or native event
    -h, --help    Print this help message

Event filtering options:
        --br          Display branch related PAPI preset events
        --cache       Display cache related PAPI preset events
        --cnd         Display conditional PAPI preset events
        --fp          Display Floating Point related PAPI preset events
        --ins         Display instruction related PAPI preset events
        --idl         Display Stalled or Idle PAPI preset events
        --l1          Display level 1 cache related PAPI preset events
        --l2          Display level 2 cache related PAPI preset events
        --l3          Display level 3 cache related PAPI preset events
        --mem         Display memory related PAPI preset events
        --msc         Display miscellaneous PAPI preset events
        --tlb         Display Translation Lookaside Buffer PAPI preset events

This program provides information about PAPI preset and native events.
PAPI preset event filters can be combined in a logical OR.
```

PAPI Utilities: *papi_avail*

```
$ utils/papi_avail
Available events and hardware information.
--------------------------------------------------------------------------------
PAPI Version             : 4.0.0.0
Vendor string and code   : GenuineIntel (1)
Model string and code    : Intel Core i7 (21)
CPU Revision             : 5.000000
CPUID Info               : Family: 6  Model: 26  Stepping: 5
CPU Megahertz            : 2926.000000
CPU Clock Megahertz      : 2926
Hdw Threads per core     : 1
Cores per Socket         : 4
NUMA Nodes               : 2
CPU's per Node           : 4
Total CPU's              : 8
Number Hardware Counters : 7
Max Multiplex Counters   : 32
--------------------------------------------------------------------------------
The following correspond to fields in the PAPI_event_info_t structure.

    [MORE...]
```

PAPI Utilities: *papi_avail*

```
    [CONTINUED…]

--------------------------------------------------------------------------------
The following correspond to fields in the PAPI_event_info_t structure.

    Name        Code     Avail Deriv Description (Note)
PAPI_L1_DCM  0x80000000  No    No   Level 1 data cache misses
PAPI_L1_ICM  0x80000001  Yes   No   Level 1 instruction cache misses
PAPI_L2_DCM  0x80000002  Yes   Yes  Level 2 data cache misses

[…]

PAPI_VEC_SP  0x80000069  Yes   No   Single precision vector/SIMD instructions
PAPI_VEC_DP  0x8000006a  Yes   No   Double precision vector/SIMD instructions
-------------------------------------------------------------------------
Of 107 possible events, 34 are available, of which 9 are derived.

avail.c                                  PASSED
```

PAPI Utilities: *papi_avail*

```
$ utils/papi_avail -e PAPI_FP_OPS
[…]
--------------------------------------------------------------------------------
The following correspond to fields in the PAPI_event_info_t structure.

Event name:                    PAPI_FP_OPS
Event Code:                    0x80000066
Number of Native Events:       2
Short Description:            |FP operations|
Long Description:             |Floating point operations|
Developer's Notes:            ||
Derived Type:                 |DERIVED_ADD|
Postfix Processing String:    ||
 Native Code[0]: 0x4000801b |FP_COMP_OPS_EXE:SSE_SINGLE_PRECISION|
 Number of Register Values: 2
 Register[ 0]: 0x0000000f |Event Selector|
 Register[ 1]: 0x00004010 |Event Code|
 Native Event Description: |Floating point computational micro-ops, masks:SSE* FP single precision Uops|

 Native Code[1]: 0x4000081b |FP_COMP_OPS_EXE:SSE_DOUBLE_PRECISION|
 Number of Register Values: 2
 Register[ 0]: 0x0000000f |Event Selector|
 Register[ 1]: 0x00008010 |Event Code|
 Native Event Description: |Floating point computational micro-ops, masks:SSE* FP double precision Uops|
--------------------------------------------------------------------------
```

PAPI Utilities: *papi_native_avail*

```
UNIX> utils/papi_native_avail
Available native events and hardware information.
--------------------------------------------------------------------------------
[…]
Event Code   Symbol  | Long Description |
--------------------------------------------------------------------------------
0x40000010   BR_INST_EXEC  | Branch instructions executed                      |
  40000410   :ANY  | Branch instructions executed                              |
  40000810   :COND  | Conditional branch instructions executed                 |
  40001010   :DIRECT  | Unconditional branches executed                        |
  40002010   :DIRECT_NEAR_CALL  | Unconditional call branches executed         |
  40004010   :INDIRECT_NEAR_CALL  | Indirect call branches executed            |
  40008010   :INDIRECT_NON_CALL  | Indirect non call branches executed         |
  40010010   :NEAR_CALLS  | Call branches executed                             |
  40020010   :NON_CALLS  | All non call branches executed                      |
  40040010   :RETURN_NEAR  | Indirect return branches executed                  |
  40080010   :TAKEN  | Taken branches executed                                 |
--------------------------------------------------------------------------------
0x40000011   BR_INST_RETIRED  | Retired branch instructions                    |
  40000411   :ALL_BRANCHES  | Retired branch instructions (Precise Event)      |
  40000811   :CONDITIONAL  | Retired conditional branch instructions (Precise  |
           | Event)                                                            |
  40001011   :NEAR_CALL  | Retired near call instructions (Precise Event)      |
--------------------------------------------------------------------------------
[…]
```

PAPI Utilities: *papi_native_avail*

```
UNIX> utils/papi_native_avail -e DATA_CACHE_REFILLS
Available native events and hardware information.
--------------------------------------------------------------------------------
[…]
--------------------------------------------------------------------------------
The following correspond to fields in the PAPI_event_info_t structure.

Event name:                    DATA_CACHE_REFILLS
Event Code:                    0x4000000b
Number of Register Values:     2
Description:                  |Data Cache Refills from L2 or System|
 Register[ 0]:   0x0000000f  |Event Selector|
 Register[ 1]:   0x00000042  |Event Code|

Unit Masks:
 Mask Info:                   |:SYSTEM|Refill from System|
  Register[ 0]:  0x0000000f  |Event Selector|
  Register[ 1]:  0x00000142  |Event Code|
 Mask Info:                   |:L2_SHARED|Shared-state line from L2|
  Register[ 0]:  0x0000000f  |Event Selector|
  Register[ 1]:  0x00000242  |Event Code|
 Mask Info:                   |:L2_EXCLUSIVE|Exclusive-state line from L2|
  Register[ 0]:  0x0000000f  |Event Selector|
  Register[ 1]:  0x00000442  |Event Code|
```

PAPI Utilities: *papi_event_chooser*

```
$ utils/papi_event_chooser PRESET PAPI_FP_OPS
Event Chooser: Available events which can be added with given events.
--------------------------------------------------------------------------------
[…]
--------------------------------------------------------------------------------
    Name        Code     Deriv Description (Note)
PAPI_L1_DCM  0x80000000  No    Level 1 data cache misses
PAPI_L1_ICM  0x80000001  No    Level 1 instruction cache misses
PAPI_L2_ICM  0x80000003  No    Level 2 instruction cache misses
[…]
PAPI_L1_DCA  0x80000040  No    Level 1 data cache accesses
PAPI_L2_DCR  0x80000044  No    Level 2 data cache reads
PAPI_L2_DCW  0x80000047  No    Level 2 data cache writes
PAPI_L1_ICA  0x8000004c  No    Level 1 instruction cache accesses
PAPI_L2_ICA  0x8000004d  No    Level 2 instruction cache accesses
PAPI_L2_TCA  0x80000059  No    Level 2 total cache accesses
PAPI_L2_TCW  0x8000005f  No    Level 2 total cache writes
PAPI_FML_INS 0x80000061  No    Floating point multiply instructions
PAPI_FDV_INS 0x80000063  No    Floating point divide instructions
-------------------------------------------------------------------------
Total events reported: 34
event_chooser.c                          PASSED
```

PAPI Utilities: *papi_event_chooser*

```
$ utils/papi_event_chooser PRESET PAPI_FP_OPS PAPI_L1_DCM
Event Chooser: Available events which can be added with given events.
--------------------------------------------------------------------------------
[…]
--------------------------------------------------------------------------------

    Name        Code     Deriv Description (Note)
PAPI_TOT_INS 0x80000032  No   Instructions completed
PAPI_TOT_CYC 0x8000003b  No   Total cycles
-------------------------------------------------------------------------
Total events reported: 2
event_chooser.c                          PASSED
```

PAPI Utilities: *papi_event_chooser*

```
$ utils/papi_event_chooser NATIVE RESOURCE_STALLS:LD_ST X87_OPS_RETIRED
    INSTRUCTIONS_RETIRED
[…]
--------------------------------------------------------------------------------
UNHALTED_CORE_CYCLES    0x40000000
 |count core clock cycles whenever the clock signal on the specific core is running (not
    halted). Alias to event CPU_CLK_UNHALTED:CORE_P|
 |Register Value[0]: 0x20003       Event Selector|
 |Register Value[1]: 0x3c          Event Code|
-------------------------------------------------------------------------
UNHALTED_REFERENCE_CYCLES       0x40000002
 |Unhalted reference cycles. Alias to event CPU_CLK_UNHALTED:REF|
 |Register Value[0]: 0x40000       Event Selector|
 |Register Value[1]: 0x13c         Event Code|
-------------------------------------------------------------------------
CPU_CLK_UNHALTED        0x40000028
 |Core cycles when core is not halted|
 |Register Value[0]: 0x60000       Event Selector|
 |Register Value[1]: 0x3c          Event Code|
    0x40001028  :CORE_P  |Core cycles when core is not halted|
    0x40008028  :NO_OTHER  |Bus cycles when core is active and the other is halted|
-------------------------------------------------------------------------
Total events reported: 3
event_chooser.c                          PASSED
```

Usage Scenarios: Calculate mflops in Loops

- Goal: What MFlops am I getting in all loops?
- Flat profile with PAPI_FP_INS/OPS and time with loop instrumentation:

Metric: PAPI_FP_INS / GET_TIME_OF_DAY
Value: Exclusive
Units: Derived metric shown in microseconds format

770.699	Loop: MULTIPLY_MATRICES [{matmult.f90} {31,9}-{36,14}]
223.39	Loop: INITIALIZE [{matmult.f90} {10,9}-{14,14}]
223.24	Loop: INITIALIZE [{matmult.f90} {17,9}-{21,14}]
171.855	Loop: MAIN [{matmult.f90} {71,9}-{74,14}]
170.862	Loop: MAIN [{matmult.f90} {112,9}-{115,14}]
122.96	Loop: MAIN [{matmult.f90} {117,9}-{128,14}]
37.549	MULTIPLY_MATRICES
21.367	INITIALIZE
13.795	Loop: MAIN [{matmult.f90} {86,9}-{106,14}]
11	MPI_Comm_size()
8.935	Loop: MAIN [{matmult.f90} {77,9}-{84,14}]
1.131	MPI_Send()
0.794	MPI_Comm_rank()
0.647	MPI_Bcast()
0.355	MPI_Recv()
0.171	MPI_Barrier()
0.115	MPI_Finalize()
0.023	MAIN

ParaProf: Mflops Sorted by Exclusive Time

TAU: ParaProf: Mean Data – s3d.loops.12000.ppk

Metric: PAPI_FP_INS / LINUX_TIMERS
Value: Exclusive
Units: Derived metric shown in microseconds format

1142.273	Loop: CHEMKIN_M::REACTION_RATE_VEC [{chemkin_m.pp.f90} {457,3}-{471,8}]
884.23	Loop: TRANSPORT_M::COMPUTECOEFFICIENTS [{mixavg_transport_m.pp.f90} {492,5}-{520,9}]
0.014	MPI_Wait()
121.843	Loop: TRANSPORT_M::COMPUTESPECIESDIFFFLUX [{mixavg_transport_m.pp.f90} {630,5}-{656,19}]
0.021	MPI_Isend()
0.06	Loop: RHSF [{rhsf.pp.f90} {209,3}-{211,7}]
183.176	Loop: INTEGRATE [{integrate_erk.pp.f90} {73,3}-{93,13}]
0	MPI_Comm_split()
0	MPI_Barrier()
148.608	RHSF
269.897	Loop: TRANSPORT_M::COMPUTEHEATFLUX [{mixavg_transport_m.pp.f90} {782,5}-{790,19}]
742.83	Loop: THERMCHEM_M::CALC_TEMP [{thermchem_m.pp.f90} {175,5}-{216,9}]
163.317	Loop: RHSF [{rhsf.pp.f90} {537,3}-{543,16}]
174.081	Loop: RHSF [{rhsf.pp.f90} {545,3}-{551,16}]
124.129	Loop: RHSF [{rhsf.pp.f90} {515,3}-{535,16}]
1153.827	Loop: DERIVATIVE_X_CALC [{derivative_x.pp.f90} {432,10}-{441,15}]
2.177	DERIVATIVE_X_COMM
1422.272	Loop: DERIVATIVE_Y_CALC [{derivative_y.pp.f90} {431,10}-{440,15}]
1459.472	Loop: DERIVATIVE_Z_CALC [{derivative_z.pp.f90} {435,10}-{444,15}]
178.375	Loop: VARIABLES_M::GET_MASS_FRAC [{variables_m.pp.f90} {96,3}-{99,7}]
374.708	Loop: THERMCHEM_M::CALC_INV_AVG_MOL_WT [{thermchem_m.pp.f90} {127,5}-{129,9}]
0	MPI_Irecv()
3.778	DERIVATIVE_Y_COMM
244.308	Loop: THERMCHEM_M::CALC_SPECENTH_ALLPTS [{thermchem_m.pp.f90} {506,3}-{512,8}]
1.294	INTEGRATE
0.002	S3D
7.111	DERIVATIVE_Z_COMM
733.265	Loop: DERIVATIVE_X_CALC [{derivative_x.pp.f90} {490,13}-{509,18}]
765.947	Loop: DERIVATIVE_X_CALC [{derivative_x.pp.f90} {461,13}-{480,18}]
950.372	Loop: DERIVATIVE_Y_CALC [{derivative_y.pp.f90} {488,13}-{507,18}]
973.95	Loop: DERIVATIVE_Y_CALC [{derivative_y.pp.f90} {459,13}-{478,18}]
249.638	Loop: TRANSPORT_M::COMPUTESTRESSTENSOR [{mixavg_transport_m.pp.f90} {711,5}-{737,10}]
1119.025	Loop: DERIVATIVE_Z_CALC [{derivative_z.pp.f90} {464,13}-{483,18}]
0.328	TRANSPORT_M::COMPUTESPECIESDIFFFLUX
1257.338	Loop: DERIVATIVE_Z_CALC [{derivative_z.pp.f90} {493,13}-{512,18}]
84.649	RK_M::CONTROLLER
30.583	DERIVATIVE_X_CALC
33.374	DERIVATIVE_Y_CALC
33.405	DERIVATIVE_Z_CALC
61.032	THERMCHEM_M::CALC_INV_AVG_MOL_WT
95.032	GET_VELOCITY_VEC
103.526	Loop: TRANSPORT_M::COMPUTECOEFFICIENTS [{mixavg_transport_m.pp.f90} {527,5}-{529,10}]
196.839	Loop: FILTER_M::FILTER [{filter_m.pp.f90} {1232,5}-{1236,9}]
293.594	THERMCHEM_M::CALC_TEMP
141.127	TRANSPORT_M::COMPUTECOEFFICIENTS
24.465	COMPUTESCALARGRADIENT
0	Loop: INTEGRATE [{integrate_erk.pp.f90} {108,3}-{110,7}]
0.245	VARIABLES_M::GET_MASS_FRAC

low mflops?

Generate a PAPI profile with 2 or more counters

```
% export TAU_MAKEFILE=$TAU_ROOT/lib/Makefile.tau-bgqtimers-papi-mpi-pdt
% export TAU_OPTIONS='-optTauSelectFile=select.tau -optVerbose'
% cat select.tau
  BEGIN_INSTRUMENT_SECTION
  loops routine="#"
  END_INSTRUMENT_SECTION

% export PATH=$TAU_ROOT/bin:$PATH
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
%
% qsub --env TAU_METRICS=TIME:PAPI_FP_INS:PAPI_L1_DCM -n 4 -t 15 ./a.out
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk
  Choose Options -> Show Derived Metrics Panel -> "PAPI_FP_INS", click "/", "TIME", click
    "Apply"  and choose the derived metric.
```

Derived Metrics in ParaProf

TAU: ParaProf Manager

- Applications
  - Standard Applications
    - Default App
      - Default Exp
        - is_papi.ppk
          - P_WALL_CLOCK_TIME
          - PAPI_FP_INS
          - PAPI_RES_STL
          - (PAPI_FP_INS / P_WALL_CLOCK_TIME)
  - Default (jdbc:derby:/Users/sameer/.ParaProf/perfdmf)
    - hpcc
    - ecco
      - pleiades_comm
        - 64p
        - 120p
        - 240p
        - 480p
  - perfexplorer_working (jdbc:derby:/Users/sameer/.ParaProf/perfexplorer_working)

TrialField	Value
Name	is_papi.ppk
Application ID	0
Experiment ID	0
Trial ID	0
CPU Cores	6
CPU MHz	2600.000
CPU Type	6-Core AMD Opteron(tm) Processor 23 (D0)
CPU Vendor	AuthenticAMD
CWD	/lustre/widow2/scratch/sameer/workshop/NPB3.1/bin
Cache Size	512 KB
Command Line	./is.C.8
Executable	/var/spool/alps/4660266/is.C.8
File Type Index	0
File Type Name	ParaProf Packed Profile
Hostname	nid00772
Local Time	2011-06-26T23:52:45-04:00
MPI Processor Name	nid00772
Memory Size	16385772 kB
Node Name	nid00772
OS Machine	x86_64
OS Name	Linux
OS Release	2.6.16.60-0.69.1_1.0102.5589.2.2.73-cnl
OS Version	#1 SMP Tue Nov 16 18:03:32 CST 2010
Starting Timestamp	1309146753453169
TAU Architecture	craycnl
TAU Config	-papi=/opt/xt-tools/papi/3.7.2/v23/ -arch=craycnl -pdt=/ccs/proj/perc/TOOLS/pdt/pdtoolkit-3.16-pf/ -pdt_c++=g++ -mpi...
TAU Makefile	/ccs/proj/perc/TOOLS/tau/tau-2.20.2/craycnl/lib/Makefile.tau-papi-mpi-pdt-pgi
TAU MetaData Merge Time	3.5E-05 seconds
TAU Version	2.20.2
TAU_CALLPATH	off
TAU_CALLPATH_DEPTH	100
TAU_COMM_MATRIX	off
TAU_COMPENSATE	off
TAU_CUPTI_API	runtime
TAU_PROFILE	on
TAU_PROFILE_FORMAT	profile
TAU_SAMPLING	off
TAU_THROTTLE	on
TAU_THROTTLE_NUMCALLS	100000
TAU_THROTTLE_PERCALL	10
TAU_TRACE	off
TAU_TRACK_HEADROOM	off
TAU_TRACK_HEAP	off
TAU_TRACK_IO_PARAMS	off
TAU_TRACK_MEMORY_LEAKS	off
TAU_TRACK_MESSAGE	off
Timestamp	1309146765912145
UTC Time	2011-06-27T03:52:45Z
pid	30138

Expression: "PAPI_FP_INS"/"P_WALL_CLOCK_TIME" Clear

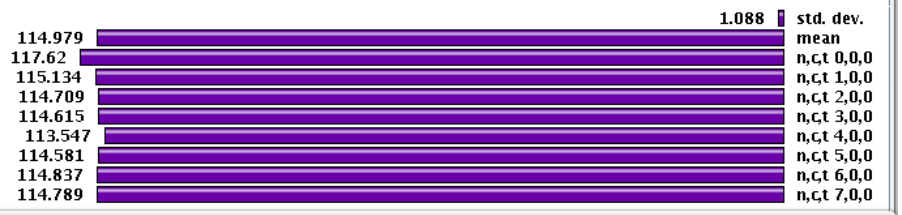
+ - * / = ( ) Apply

ParaProf's Source Browser: Loop Level Instrumentation

TAU: ParaProf: Function Data Window: s3d_callpath_papi.ppk

File Options Windows Help

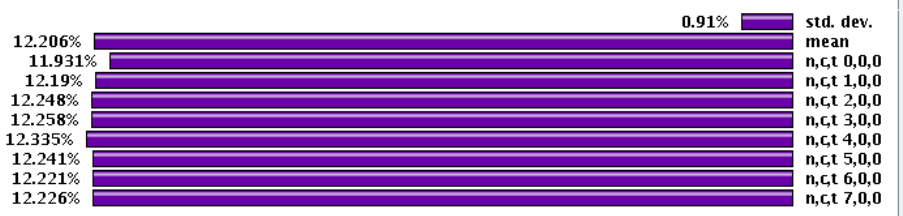
Name: Loop: TRANSPORT_M::COMPUTESPECIESDIFFFLUX [{mixavg_transport_m.pp.f90} {630,5}-{656,19}]
Metric Name: PAPI_FP_INS / GET_TIME_OF_DAY
Value: Exclusive
Units: Derived metric shown in microseconds format



TAU: ParaProf: Function Data Window: s3d_callpath_papi.ppk

File Options Windows Help

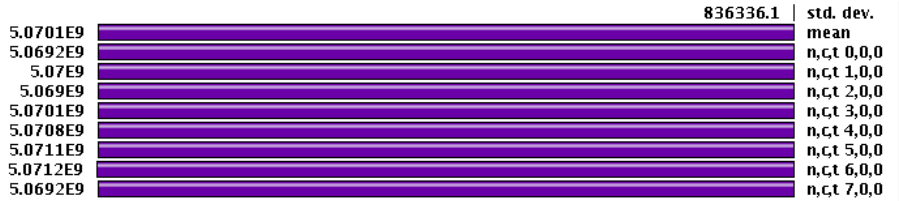
Name: Loop: TRANSPORT_M::COMPUTESPECIESDIFFFLUX [{mixavg_transport_m.pp.f90} {630,5}-{656,19}]
Metric Name: GET_TIME_OF_DAY
Value: Exclusive percent



TAU: ParaProf: Function Data Window: s3d_callpath_papi.ppk

File Options Windows Help

Name: Loop: TRANSPORT_M::COMPUTESPECIESDIFFFLUX [{mixavg_transport_m.pp.f90} {630,5}-{656,19}]
Metric Name: PAPI_L1_DCM
Value: Exclusive
Units: counts



TAU: ParaProf: Source Browser: /mnt/epsilon/Users/sameer/rs/taudata/s3d/harness/flat/papi8

File Help

```
606      grad_mixMW(:,:,:,m) = grad_mixMW(:,:,:,m)*avmolwt(:,:,:)
607     end do
608
609     ! compute grad_P
610     if (baro_switch) then
611        allocate(grad_P(nx,ny,nz,3))
612        grad_P = 0.0
613        if (vary_in_x == 1) then
614           call derivative_x( nx,ny,nz, Press, grad_P(:,:,:,1), scale_1x, 1 )
615        endif
616        if (vary_in_y == 1) then
617           call derivative_y( nx,ny,nz, Press, grad_P(:,:,:,2), scale_1y, 1 )
618        endif
619        if (vary_in_z == 1) then
620           call derivative_z( nx,ny,nz, Press, grad_P(:,:,:,3), scale_1z, 1 )
621        endif
622     endif
623
624     ! Changed by Ramanan - 01/24/05
625     ! Ds_mixavg is now \rho*D
626     !
627     !grad_P/press and avmolwt*grad_T/Temp can be optimized by division before the loop.
628     ! compute diffusive flux for species n in direction m.
629     diffFlux(:,:,:,n_spec,:) = 0.0
630     DIRECTION: do m=1,3
631        SPECIES: do n=1,n_spec-1
632
633           if (baro_switch) then
634              ! driving force includes gradient in mole fraction and baro-diffusion:
635              diffFlux(:,:,:,n,m) = - Ds_mixavg(:,:,:,n) * ( grad_Ys(:,:,:,n,m)  &
636                                    + Ys(:,:,:,n) * ( grad_mixMW(:,:,:,m)    &
637                                    + (1 - molwt(n)*avmolwt) * grad_P(:,:,:,m)/Press))
638           else
639              ! driving force is just the gradient in mole fraction:
640              diffFlux(:,:,:,n,m) = - Ds_mixavg(:,:,:,n) * ( grad_Ys(:,:,:,n,m)  &
641                                    + Ys(:,:,:,n) * grad_mixMW(:,:,:,m) )
642           endif
643
644           ! Add thermal diffusion:
645           if (thermDiff_switch) then
646              diffFlux(:,:,:,n,m) = diffFlux(:,:,:,n,m)    &
647                   - Ds_mixavg(:,:,:,n) * Rs_therm_diff(:,:,:,n) * molwt(n) &
648                   * avmolwt * grad_T(:,:,:,m) / Temp
649           endif
650
651           ! compute contribution to nth species diffusive flux
652           ! this will ensure that the sum of the diffusive fluxes is zero.
653           diffFlux(:,:,:,n_spec,m) = diffFlux(:,:,:,n_spec,m) - diffFlux(:,:,:,n,m)
654
655        enddo SPECIES
656     enddo DIRECTION
657
658     if (baro_switch) then
659        deallocate(grad_P)
660     endif
661
662     return
663   end subroutine computeSpeciesDiffFlux
664
665 !!$=============================================================================
666
667
668   subroutine computeStressTensor( grad_u)
```

Estimation of tool intrusiveness

PAPI Utilities: *papi_cost*

```
$ utils/papi_cost -h
This is the PAPI cost program.
It computes min / max / mean / std. deviation for PAPI start/stop pairs;
   for PAPI reads, and for PAPI_accums.
   Usage:

    cost [options] [parameters]
    cost TESTS_QUIET

Options:

  -b BINS        set the number of bins for the graphical
                  distribution of costs. Default: 100
  -d             show a graphical distribution of costs
  -h             print this help message
  -s             show number of iterations above the first
                  10 std deviations
  -t THRESHOLD   set the threshold for the number of
                  iterations. Default: 100,000
```

PAPI Utilities: *papi_cost*

```
$ utils/papi_cost
Cost of execution for PAPI start/stop and PAPI read.
This test takes a while. Please be patient...
Performing start/stop test...

Total cost for PAPI_start/stop(2 counters) over 1000000 iterations
min cycles    : 63
max cycles    : 17991
mean cycles   : 69.000000
std deviation: 34.035263
 Performing start/stop test...

Performing read test...

Total cost for PAPI_read(2 counters) over 1000000 iterations
min cycles    : 288
max cycles    : 102429
mean cycles   : 301.000000
std deviation: 144.694053
 cost.c                                PASSED
```

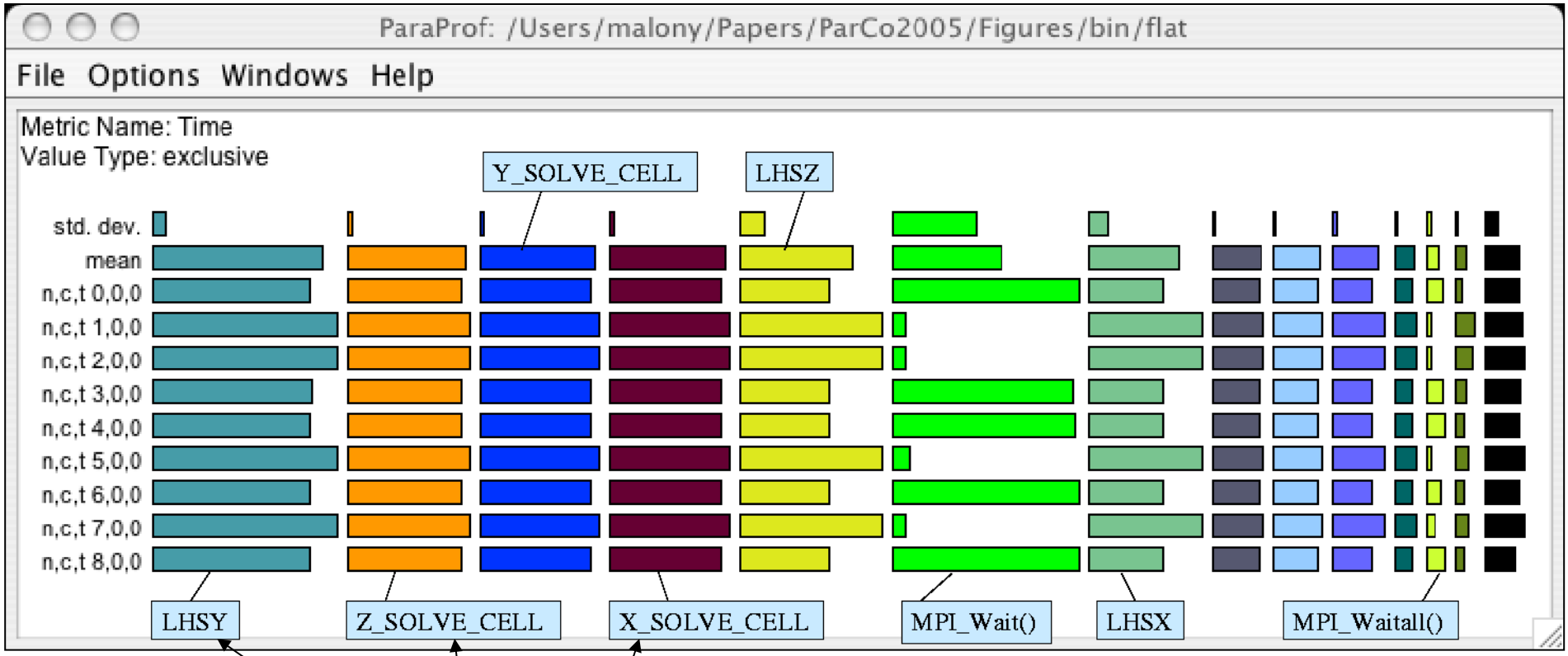
PAPI Utilities: *papi_cost*

```
Cost distribution profile

      63:**************************** 999969 counts ****************************
     153:
     243:
     [...]
    1863:
    1953:********************
    2043:
    2133:********************
    2223:
    2313:
    2403:********************
    2493:********************
    2583:****************************************
    2673:****************************************
    2763:********************************************************************************
    2853:****************************************
    2943:
    3033:********************
    3123:****************************************
    3213:****************************************
    3303:
    3393:
    3483:
    3573:
    3663:********************
```

Profile Measurement – Three Flavors

- Flat profiles
  - Time (or counts) spent in each routine (nodes in callgraph).
  - Exclusive/inclusive time, no. of calls, child calls
  - E.g,: MPI_Send, foo, …
- Callpath Profiles
  - Flat profiles, **plus**
  - Sequence of actions that led to poor performance
  - Time spent along a calling path (edges in callgraph)
  - E.g., “main=> f1 => f2 => MPI_Send” shows the time spent in MPI_Send when called by f2, when f2 is called by f1, when it is called by main. Depth of this callpath = 4 (TAU_CALLPATH_DEPTH environment variable)
- Phase based profiles
  - Flat profiles, **plus**
  - Flat profiles under a phase (nested phases are allowed)
  - Default “main” phase has all phases and routines invoked outside phases
  - Supports static or dynamic (per-iteration) phases
  - E.g., “IO => MPI_Send” is time spent in MPI_Send in IO phase

Phase Profiling (NAS BT, Flat Profile)

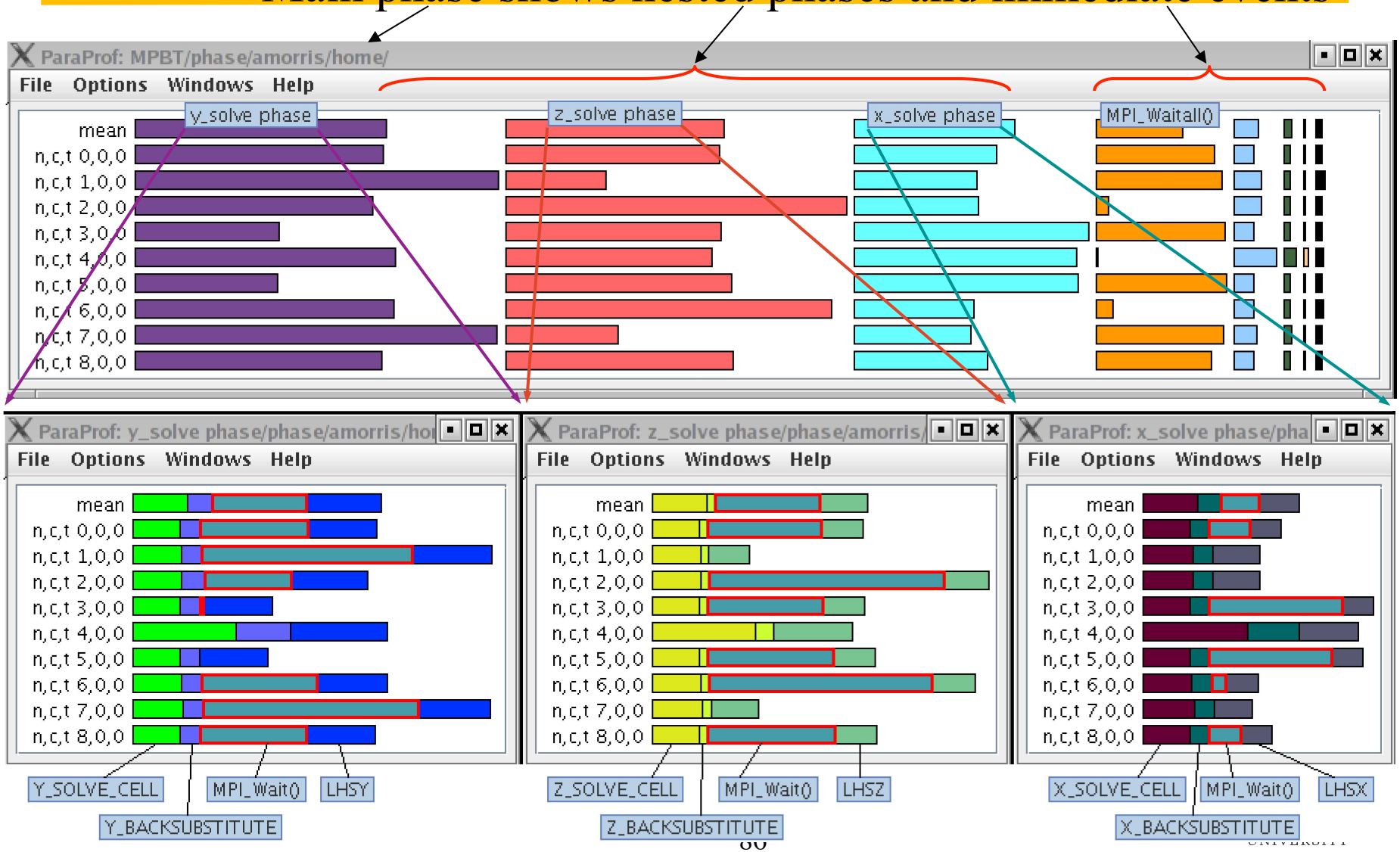


Application routine names reflect phase semantics

How is MPI_Wait() distributed relative to solver direction?

NAS BT – Phase Profile (Main and X, Y, Z)

Main phase shows nested phases and immediate events

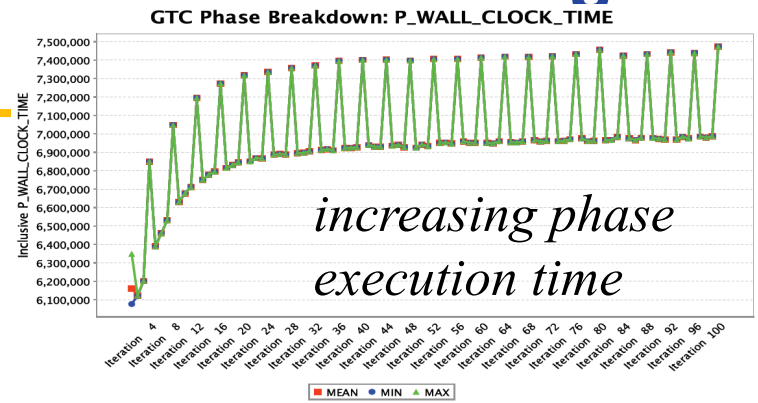


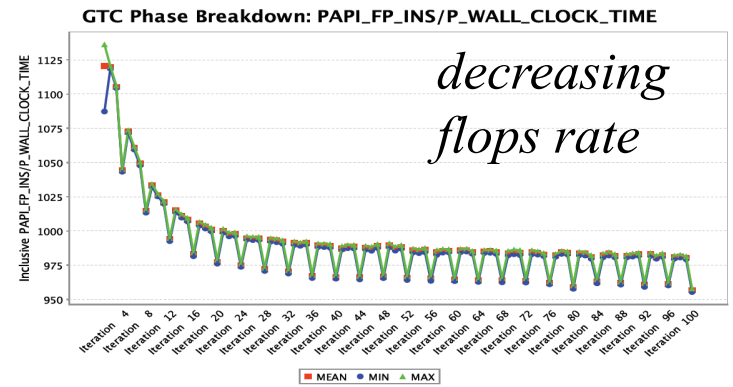
TAU Timers and Phases

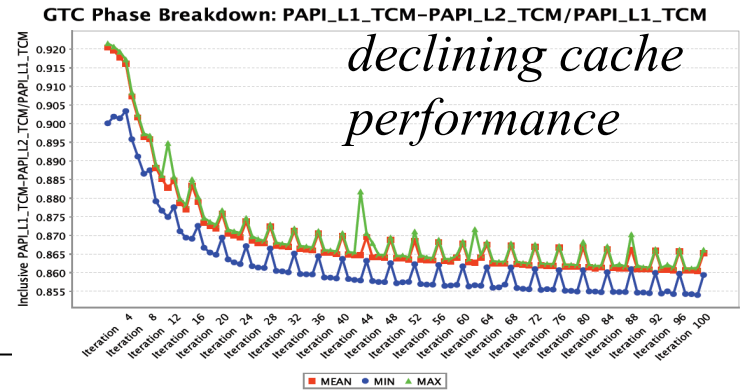
- Static timer
  - Shows time spent in all invocations of a routine (foo)
  - E.g., “foo()” 100 secs, 100 calls
- Dynamic timer
  - Shows time spent in each invocation of a routine
  - E.g., “foo() 3” 4.5 secs, “foo 10” 2 secs (invocations 3 and 10 respectively)
- Static phase
  - Shows time spent in all routines called (directly/indirectly) by a given routine (foo)
  - E.g., “foo() => MPI_Send()” 100 secs, 10 calls shows that a total of 100 secs were spent in MPI_Send() when it was called by foo.
- Dynamic phase
  - Shows time spent in all routines called by a given invocation of a routine.
  - E.g., “foo() 4 => MPI_Send()” 12 secs, shows that 12 secs were spent in MPI_Send when it was called by the 4th invocation of foo.

Performance Dynamics: Phase-Based Profiling

- Profile phases capture performance with respect to application-defined 'phases' of execution
  - Separate full profile produce for each phase
- GTC particle-in-cell simulation of fusion turbulence
- Phases assigned to iterations
- Data change affects cache







Memory and I/O evaluation

Library interposition/wrapping: tau_exec, tau_wrap

- TAU provides a wealth of options to measure the performance of an application
- Need to simplify TAU usage to easily evaluate performance properties, including I/O, memory, and communication
- Designed a new tool (*tau_exec*) that leverages runtime instrumentation by pre-loading measurement libraries
- Works on dynamic executables (default under Linux, not on IBM Blue Gene where we must compile with -dynamic)
- Substitutes I/O, MPI, and memory allocation/deallocation routines with instrumented calls
  - Interval events (e.g., time spent in write())
  - Atomic events (e.g., how much memory was allocated)
- Measure I/O and memory usage

TAU Execution Command (tau_exec)

- Uninstrumented execution (compiled with –Wl,–Bdynamic on BG/P)
  - % qsub –n 256 –t 10 ./a.out
- Track MPI performance (-T <options>)
  - % tau_exec –qsub -T bgqtimers,mpi,pdt -- qsub –n 256 –t 10 ./a.out
- Track I/O and MPI performance (MPI by default, use –T serial for serial)
  - % tau_exec –io qsub -T bgqtimers,mpi,pdt -- qsub –n 256 –t 10 ./a.out
- Track memory operations
  - % tau_exec –memory –env TAU_TRACK_MEMORY_LEAKS=1 qsub -T bgqtimers,mpi,pdt -- qsub –n 256 –t 10 ./a.out

Library wrapping: tau_gen_wrapper

- How to instrument an external library without source?
  - Source may not be available
  - Library may be too cumbersome to build (with instrumentation)
- Build a library wrapper tools
  - Used PDT to parse header files
  - Generate new header files with instrumentation files
  - Three methods to instrument: runtime preloading, linking, redirecting headers to re-define functions
- Application is instrumented
- Add the –optTauWrapFile=<wrapperdir>/link_options.tau file to TAU_OPTIONS env var while compiling with tau_cc.sh, etc.
- Wrapped library
  - Redirects references at routine callsite to a wrapper call
  - Wrapper internally calls the original
  - Wrapper has TAU measurement code

HDF5 Library Wrapping

```
$ tau_gen_wrapper hdf5.h /usr/lib/libhdf5.a -f select.tau

Usage : tau_gen_wrapper <header> <library> [-r|-d|-w (default)] [-g groupname] [-i
headerfile] [-c|-c++|-fortran] [-f <instr_req_file> ]
• instruments using runtime preloading (-r), or -Wl,-wrap linker (-w), redirection
of header file to redefine the wrapped routine (-d)
• instrumentation specification file (select.tau)
• -g group may be specified (hdf5)
• tau_exec loads libhdf5_wrap.so shared library using –loadlib=<libwrap_pkg.so>
• creates the wrapper/ directory with linkoptions.tau passed to the TAU_OPTIONS
environment variable using –optTauWrapFile=<file>

NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        0.057            1           1          13       1236 .TAU Application
 70.8        0.875        0.875           1           0        875 hid_t H5Fcreate()
  9.7         0.12         0.12           1           0        120 herr_t H5Fclose()
  6.0        0.074        0.074           1           0         74 hid_t H5Dcreate()
  3.1        0.038        0.038           1           0         38 herr_t H5Dwrite()
  2.6        0.032        0.032           1           0         32 herr_t H5Dclose()
  2.1        0.026        0.026           1           0         26 herr_t H5check_version()
  0.6        0.008        0.008           1           0          8 hid_t H5Screate_simple()
  0.2        0.002        0.002           1           0          2 herr_t H5Tset_order()
  0.2        0.002        0.002           1           0          2 hid_t H5Tcopy()
  0.1        0.001        0.001           1           0          1 herr_t H5Sclose()
```

A New Approach: tau_exec

- Runtime instrumentation by pre-loading the measurement library
- Works on dynamic executables (default under Linux)
- Substitutes I/O, MPI and memory allocation/deallocation routines with instrumented calls
- Track interval events (e.g., time spent in write()) as well as atomic events (e.g., how much memory was allocated) in wrappers
- Accurately measure I/O and memory usage

Tracking I/O in static binaries (IBM Blue Gene)

- The linker can substitute TAU's I/O wrapper and intercept POSIX I/O Calls
- We can track parameters that flow through the I/O calls
- Configure TAU with –iowrappers
- Use –optTrackIO in TAU_OPTIONS

Tracking I/O in static binaries

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-bgqtimers-papi-mpi-pdt
% export PATH=$TAU_ROOT/bin:$PATH
% export TAU_OPTIONS='-optTrackIO -optVerbose'
% make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh
% mpirun -n 4 ./a.out
% paraprof -pack ioprofile.ppk
% export TAU_TRACK_IO_PARAMS 1
% mpirun -n 4 ./a.out (to track parameters used in POSIX I/O
   calls as context events)
```

Issues

- Heap memory usage reported by the mallinfo() call is not 64-bit clean.
  - 32 bit counters in Linux roll over when > 4GB memory is used
  - We keep track of heap memory usage in 64 bit counters inside TAU
- Compensation of perturbation introduced by tool
  - Only show what application uses
  - Create guards for TAU calls to not track I/O and memory allocations/de-allocations performed inside TAU
- Provide broad POSIX I/O and memory coverage

I/O Calls Supported

Unbuffered I/O	Buffered I/O	Communication	Control	Asynchronous I/O
open	fopen	socket	fcntl	aio_read
open64	fopen64	pipe	rewind	aio_write
close	fdopen	socketpair	lseek	aio_suspend
read	freopen	bind	lseek64	aio_cancel
write	fclose	accept	fseek	aio_return
readv	fprintf	connect	dup	lio_listio
writev	fscanf	recv	dup2	
creat	fwrite	send	mkstep	
creat64	fread	sendto	tmpfile	
		recvfrom		
		pclose		

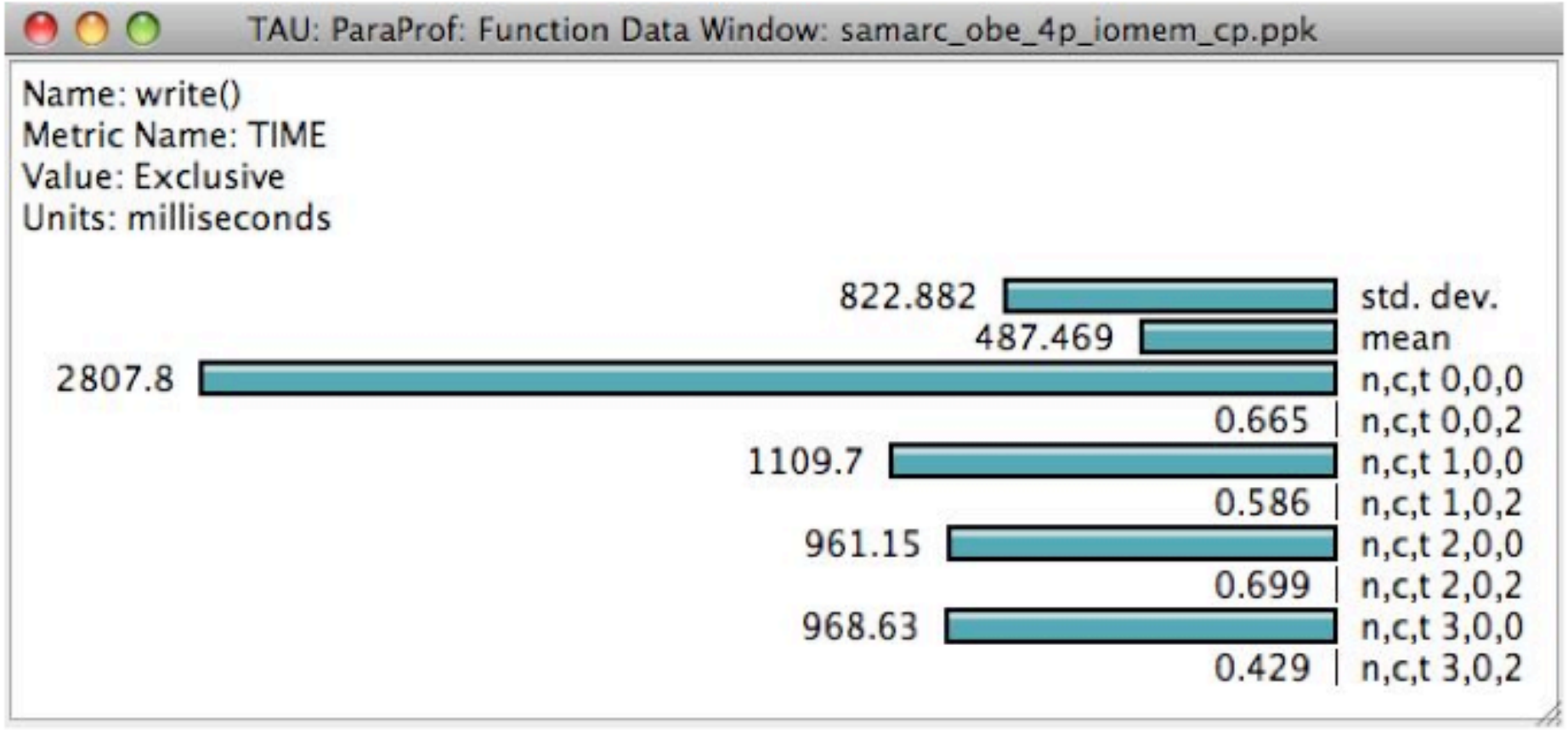
Tracking I/O in Each File

TAU: ParaProf: Context Events for thread: n,c,t, 1,0,0 - IOR_mana_iothreads_posix.ppk

File Windows Help

Name	Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
.TAU application						
MPI_Finalize()						
MPI_Init()						
fscanf()						
read()						
Bytes Read	20,024	32	8,192	4	625.75	2,014.699
Bytes Read <file="/opt/openmpi/tm/intel/1.4/etc/openmpi-mca-params.conf">	2,812	1	2,812	2,812	2,812	0
Bytes Read <file="/opt/openmpi/tm/intel/1.4/share/openmpi/help-mpi-btl-openib.txt">	8,192	1	8,192	8,192	8,192	0
Bytes Read <file="/opt/openmpi/tm/intel/1.4/share/openmpi/mca-btl-openib-device-params.ini">	8,727	2	8,192	535	4,363.5	3,828.5
Bytes Read <file="/sys/class/infiniband/mthca0/node_type">	8	1	8	8	8	0
Bytes Read <file="/sys/class/infiniband/mthca0/ports/1/gids/0">	41	1	41	41	41	0
Bytes Read <file="/sys/class/infiniband_verbs/abi_version">	8	1	8	8	8	0
Bytes Read <file="/sys/class/infiniband_verbs/uverbs0/abi_version">	8	1	8	8	8	0
Bytes Read <file="/sys/class/infiniband_verbs/uverbs0/device/device">	24	3	8	8	8	0
Bytes Read <file="/sys/class/infiniband_verbs/uverbs0/device/vendor">	24	3	8	8	8	0
Bytes Read <file="/sys/class/infiniband_verbs/uverbs0/ibdev">	64	1	64	64	64	0
Bytes Read <file="/sys/devices/system/cpu/cpu0/topology/core_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu0/topology/physical_package_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu1/topology/core_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu1/topology/physical_package_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu2/topology/core_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu2/topology/physical_package_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu3/topology/core_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu3/topology/physical_package_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu4/topology/core_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu4/topology/physical_package_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu5/topology/core_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu5/topology/physical_package_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu6/topology/core_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu6/topology/physical_package_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu7/topology/core_id">	7	1	7	7	7	0
Bytes Read <file="/sys/devices/system/cpu/cpu7/topology/physical_package_id">	7	1	7	7	7	0
Bytes Read <file="pipe">	4	1	4	4	4	0
READ Bandwidth (MB/s)	2,932.118	32	1,170.286	0.001	91.629	269.282
READ Bandwidth (MB/s) <file="/opt/openmpi/tm/intel/1.4/etc/openmpi-mca-params.conf">	312.444	1	312.444	312.444	312.444	0
READ Bandwidth (MB/s) <file="/opt/openmpi/tm/intel/1.4/share/openmpi/help-mpi-btl-openib.txt">	1,170.286	1	1,170.286	1,170.286	1,170.286	0
READ Bandwidth (MB/s) <file="/opt/openmpi/tm/intel/1.4/share/openmpi/mca-btl-openib-device-params.i	1,291.5	2	1,024	267.5	645.75	378.25
READ Bandwidth (MB/s) <file="/sys/class/infiniband/mthca0/node_type">	4	1	4	4	4	0
READ Bandwidth (MB/s) <file="/sys/class/infiniband/mthca0/ports/1/gids/0">	0.304	1	0.304	0.304	0.304	0
READ Bandwidth (MB/s) <file="/sys/class/infiniband_verbs/abi_version">	4	1	4	4	4	0
READ Bandwidth (MB/s) <file="/sys/class/infiniband_verbs/uverbs0/abi_version">	4	1	4	4	4	0
READ Bandwidth (MB/s) <file="/sys/class/infiniband_verbs/uverbs0/device/device">	16	3	8	4	5.333	1.886
READ Bandwidth (MB/s) <file="/sys/class/infiniband_verbs/uverbs0/device/vendor">	20	3	8	4	6.667	1.886
READ Bandwidth (MB/s) <file="/sys/class/infiniband_verbs/uverbs0/ibdev">	32	1	32	32	32	0

Time Spent in POSIX I/O write()

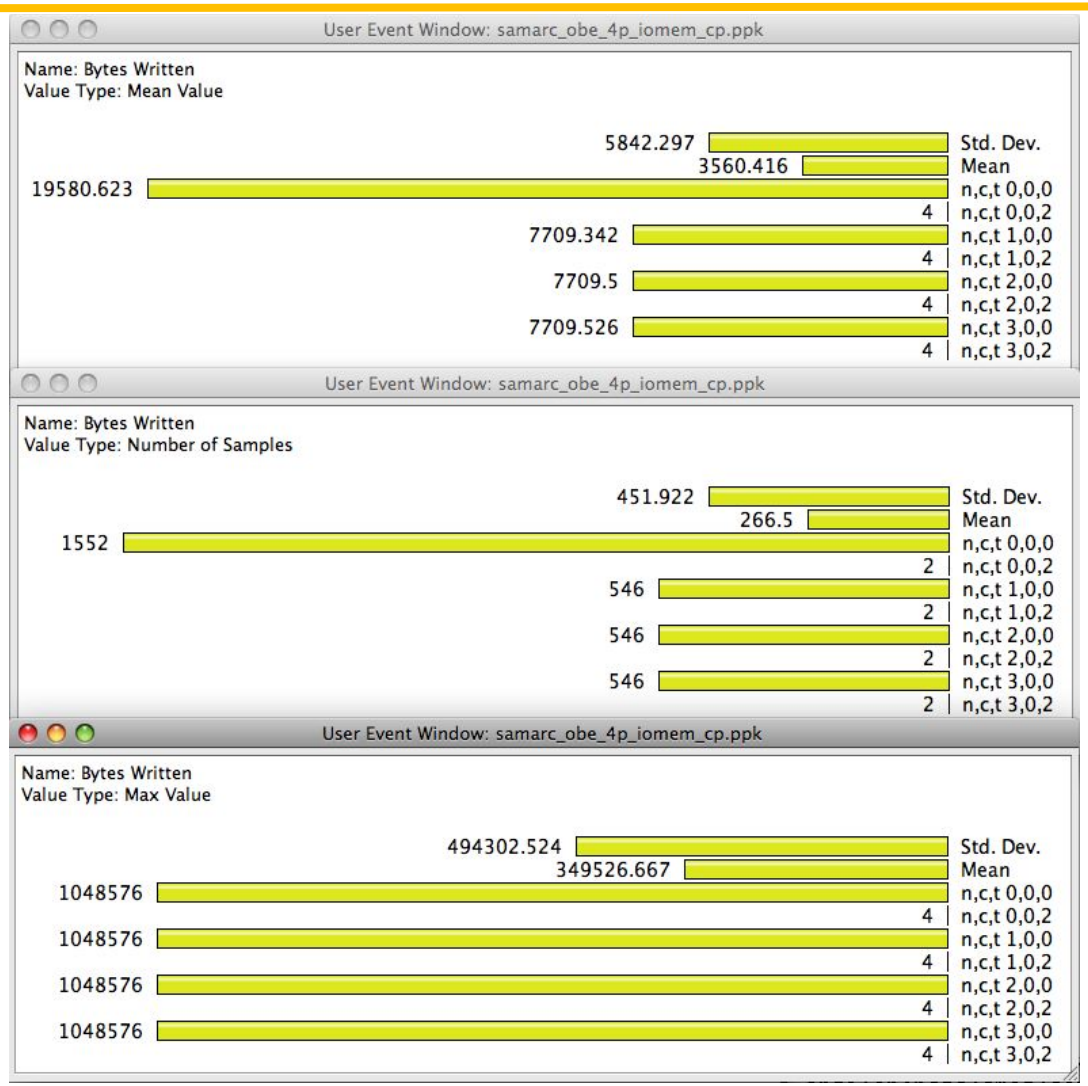


Volume of I/O by File, Memory

TAU: ParaProf: Context Events for thread: n,c,t, 1,0,0 – samarc_obe_4p_iomem_cp.ppk

Name	Total	MeanValue	NumSamples	MinValue	MaxValue	Std. Dev.
.TAU application						
read()						
fopen64()						
fclose()						
OurMain()						
malloc size	25,235	1,097.174	23	11	12,032	2,851.143
free size	22,707	1,746.692	13	11	12,032	3,660.642
OurMain [{wrapper.py}{3}]						
read()						
malloc size	3,877	323.083	12	32	981	252.72
free size	1,536	219.429	7	32	464	148.122
fopen64()						
fclose()						
<module> [{obe.py}{8}]						
writeRestartData [{samarcInterface.py}{145}]						
samarcWriteRestartData						
write()						
WRITE Bandwidth (MB/s) <file="samarc/restore.00002/nodes.00004/proc.00001">		74.565	117	0	2,156.889	246.386
WRITE Bandwidth (MB/s) <file="samarc/restore.00001/nodes.00004/proc.00001">		77.594	117	0	1,941.2	228.366
WRITE Bandwidth (MB/s)		76.08	234	0	2,156.889	237.551
Bytes Written <file="samarc/restore.00002/nodes.00004/proc.00001">	2,097,552	17,927.795	117	1	1,048,576	133,362.946
Bytes Written <file="samarc/restore.00001/nodes.00004/proc.00001">	2,097,552	17,927.795	117	1	1,048,576	133,362.946
Bytes Written	4,195,104	17,927.795	234	1	1,048,576	133,362.946
open64()						

Bytes Written



Memory Leaks in MPI

TAU: ParaProf: Context Events for thread: n,c,t, 0,0,0 – samarc_obe_4p_iomem_cp.ppk

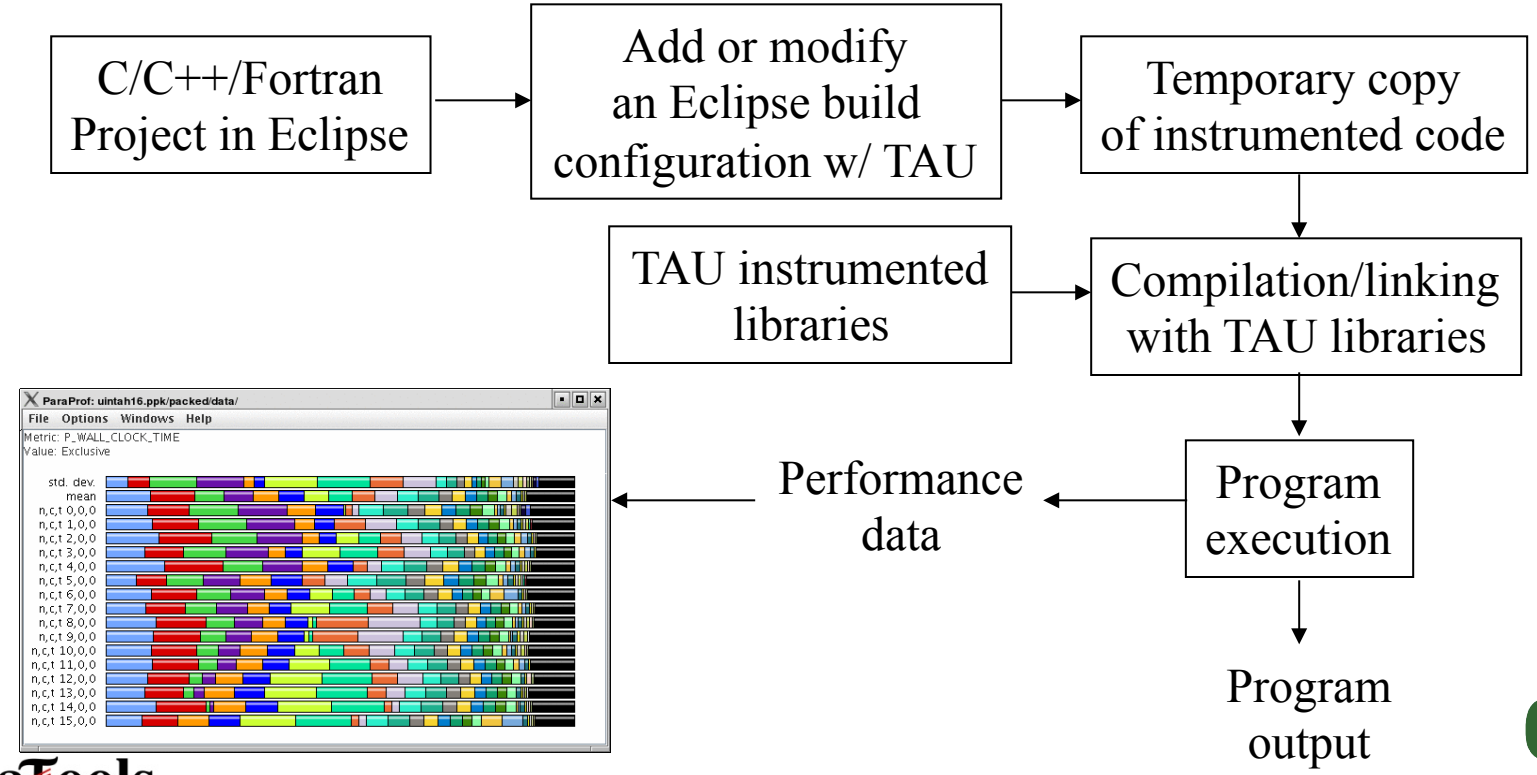
Name	Total	MeanValue	NumSamples	MaxValue	MinValue	Std. Dev.
.TAU application						
MPI_Finalize()						
free size	23,901,253	22,719.822	1,052	2,099,200	2	186,920.948
malloc size	5,013,902	65,972.395	76	5,000,000	2	569,732.815
MEMORY LEAK!	5,000,264	500,026.4	10	5,000,000	3	1,499,991.2
read()						
Bytes Read	4	4	1	4	4	0
READ Bandwidth (MB/s) <file="pipe">		0.308	1	0.308	0.308	0
Bytes Read <file="pipe">	4	4	1	4	4	0
READ Bandwidth (MB/s)		0.308	1	0.308	0.308	0
write()						
WRITE Bandwidth (MB/s)		0.635	102	12	0	1.472
Bytes Written <file="/dev/infiniband/rdma_cm">	24	24	1	24	24	0
Bytes Written	1,456	14.275	102	28	4	5.149
WRITE Bandwidth (MB/s) <file="/dev/infiniband/uverbs0">		0.528	97	12	0.089	1.32
Bytes Written <file="pipe">	64	16	4	28	4	12
WRITE Bandwidth (MB/s) <file="/dev/infiniband/rdma_cm">		1.714	1	1.714	1.714	0
Bytes Written <file="/dev/infiniband/uverbs0">	1,368	14.103	97	24	12	4.562
WRITE Bandwidth (MB/s) <file="pipe">		2.967	4	5.6	0	2.644
writev()						
WRITE Bandwidth (MB/s)		4.108	2	7.667	0.549	3.559
Bytes Written	297	148.5	2	230	67	81.5
WRITE Bandwidth (MB/s) <file="socket">		4.108	2	7.667	0.549	3.559
Bytes Written <file="socket">	297	148.5	2	230	67	81.5
readv()						
Bytes Read	112	28	4	36	20	8
READ Bandwidth (MB/s) <file="socket">		25.5	4	36	10	11.079
Bytes Read <file="socket">	112	28	4	36	20	8
READ Bandwidth (MB/s)		25.5	4	36	10	11.079
MPI_Comm_free()						
free size	10,952	195.571	56	1,024	48	255.353
read()						
MPI_Type_free()						
MPI_Init()						
fopen64()						
free size	231,314	263.456	878	568	35	221.272
MEMORY LEAK!	1,105,956	1,868.169	592	7,200	32	3,078.574
malloc size	1,358,286	901.318	1,507	7,200	32	2,087.737
OurMain()						
fclose()						

TAU Integration with IDEs

- High performance software development environments
  - Tools may be complicated to use
  - Interfaces and mechanisms differ between platforms / OS
- Integrated development environments
  - Consistent development environment
  - Numerous enhancements to development process
  - Standard in industrial software development
- Integrated performance analysis
  - Tools limited to single platform or programming language
  - Rarely compatible with 3rd party analysis tools
  - Little or no support for parallel projects

TAU and Eclipse

- Provide an interface for configuring TAU's automatic instrumentation within Eclipse's build system
- Manage runtime configuration settings and environment variables for execution of TAU instrumented programs



TAU and Eclipse

Fortran – matmult.f90 – Eclipse SDK

File Edit Refactor Navigate Search Project Run Window Help

Fortran Projects | Navigator

- matmultiply
  - Binaries
  - Includes
  - .settings
  - Debug_ia32
  - Debug_ia32(tau-icpc-mpi-pdt)
  - matmult.f90
  - .cdtbuild
  - .cdtproject
  - .project
  - profile.0.0.0
  - profile.1.0.0
  - profile.2.0.0
  - profile.3.0.0

matmult.f90

```
!*********************************************************************
!     matmult.f90 - simple matrix multiply implementation
!*********************************************************************
      subroutine initialize(a, b, n)
        double precision a(n,n)
        double precision b(n,n)
        integer n

! first initialize the A matrix
        do i = 1,n
          do j = 1,n
            a(j,i) = i
          end do
        end do

! then initialize the B matrix
        do i = 1,n
          do j = 1,n
            b(j,i) = i
          end do
        end do

      end subroutine initialize

      subroutine multiply_matrices(answer, buffer, b, matsize)
        double precision buffer(matsize), answer(matsize)
        double precision b(matsize, matsize)
        integer i, j
! multiply the row with the column
```

Outline | Make ...

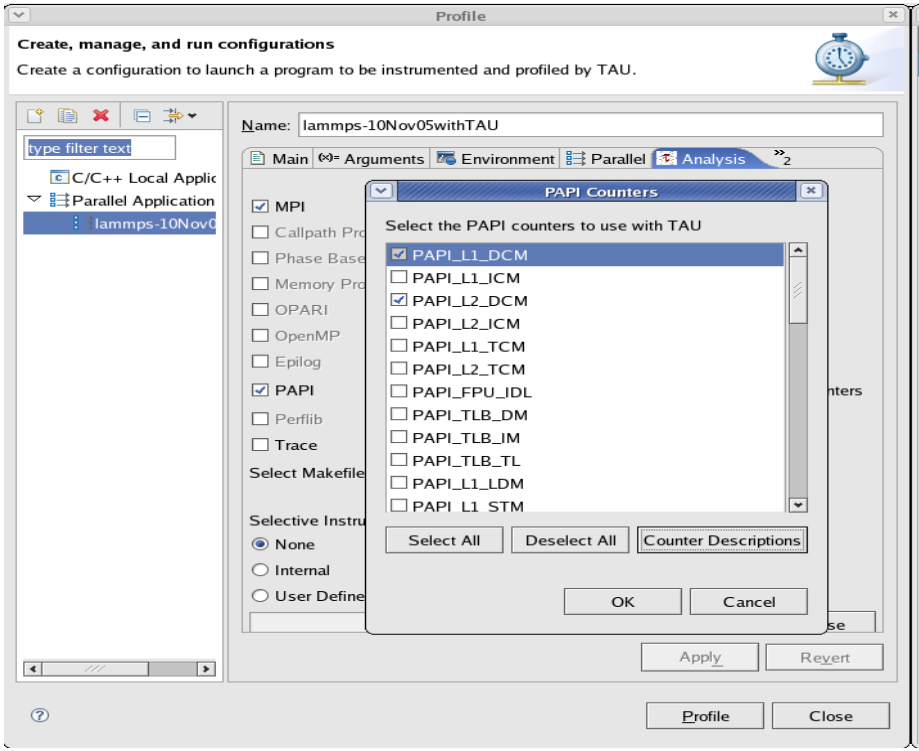
- initialize
- multiply_matrices
- main

Problems | Console | Properties | Performance Data Manager | Launch ParaProf

- AORSA2D
- matmultiply
  - Experiment
    - The New Trial: 2006-12-02 20:36:59
- mm
- ring

PerfDMF

Choosing PAPI Counters with TAU in Eclipse



Counter	Definition
PAPI_L1_DCM	Level 1 data cache misses
PAPI_L1_ICM	Level 1 instruction cache misses
PAPI_L2_DCM	Level 2 data cache misses
PAPI_L2_ICM	Level 2 instruction cache misses
PAPI_L1_TCM	Level 1 cache misses
PAPI_L2_TCM	Level 2 cache misses
PAPI_FPU_IDL	Cycles floating point units are idle
PAPI_TLB_DM	Data translation lookaside buffer misses
PAPI_TLB_IM	Instruction translation lookaside buffer misses
PAPI_TLB_TL	Total translation lookaside buffer misses
PAPI_L1_LDM	Level 1 load misses
PAPI_L1_STM	Level 1 store misses
PAPI_L2_LDM	Level 2 load misses
PAPI_L2_STM	Level 2 store misses
PAPI_STL_ICY	Cycles with no instruction issue
PAPI_HW_INT	Hardware interrupts
PAPI_BR_TKN	Conditional branch instructions taken
PAPI_BR_MSP	Conditional branch instructions mispredicted
PAPI_TOT_INS	Instructions completed
PAPI_FP_INS	Floating point instructions
PAPI_BR_INS	Branch instructions
PAPI_VEC_INS	Vector/SIMD instructions
PAPI_RES_STL	Cycles stalled on any resource
PAPI_TOT_CYC	Total cycles
PAPI_L1_DCH	Level 1 data cache hits
PAPI_L2_DCH	Level 2 data cache hits
PAPI_L1_DCA	Level 1 data cache accesses
PAPI_L2_DCA	Level 2 data cache accesses
PAPI_L2_DCR	Level 2 data cache reads
PAPI_L2_DCW	Level 2 data cache writes

Labs!

Lab Instructions

```
Get workshop.tar.gz and install on the SCRATCH filesystem:
% cp /soft/perftools/tau/workshop.tar.gz .
% tar zxf workshop.tar.gz
% cd workshop/first
% cat README
% make
% paraprof tauprofile.xml &
```

Lab Instructions

To profile a code using TAU:

1. Change the compiler name to tau_cxx.sh, tau_f90.sh, tau_cc.sh:
   F90 = tau_f90.sh
2. Choose TAU stub makefile
   % soft add +tau_latest
   % export TAU_MAKEFILE=$TAU/Makefile.tau-[options]
   % make F90=tau_f90.sh
3. If stub makefile has -papi in its name, set the TAU_METRICS environment variable:
   % qsub --env TAU_METRICS=TIME:PAPI_L2_DCM:PAPI_TOT_CYC...
4. Build and run workshop examples, then run pprof/paraprof

More Information

- PAPI References:
  - PAPI documentation page available from the PAPI website: http://icl.cs.utk.edu/papi/
- TAU References:
  - TAU Users Guide and papers available from the TAU website: http://tau.uoregon.edu/
- VAMPIR References
  - VAMPIR website http://www.vampir.eu/
- Scalasca/KOJAK References
  - Scalasca documentation page http://www.scalasca.org/
- Eclipse PTP References
  - Documentation available from the Eclipse PTP website: http://www.eclipse.org/ptp/

Acknowledgements

- Department of Energy
  - Office of Science
  - Argonne National Laboratory
  - ORNL
  - NNSA/ASC Trilabs (SNL, LLNL, LANL)
- HPCMP DoD PETTT Program
- National Science Foundation
- University of Tennessee
  - Daniel Terpstra
- University of Oregon
  - Allen D. Malony, K. Huck, W. Spear, S. Biersdorff
- TU Dresden
  - Holger Brunst, Andreas Knupfer
  - Wolfgang Nagel
- Research Centre Juelich, Germany
  - Bernd Mohr
  - Felix Wolf