TAU

Tenth DOE ACTS Collection Workshop

Leveraging the Development of Computational Science and Engineering Software Through Sustainable High Performance Tools

Sutardja Dai Hall (CITRIS) Building Room 250, Thursday,11am-12:30pm, August 20, 2008

Sameer Shende

Director, Performance Research Laboratory

University of Oregon, Eugene, OR

sameer@cs.uoregon.edu

http://tau.uoregon.edu/tau.ppt

UNIVERSITY OF OREGON

Acknowledgements: University of Oregon

- Dr. Allen D. Malony, Professor, CIS Dept, and Director, NeuroInformatics Center
- Alan Morris, Senior software engineer
- Wyatt Spear, Software engineer
- Scott Biersdorff, Software engineer
- Dr. Robert Yelle, Research faculty
- Suzanne Millstein, Ph.D. student
- Aroon Nataraj, Ph.D. student
- Shangkar Mayanglambam, Ph.D. student
- Ivan Pulleyn, Systems administrator

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.)
- TAU runs on all HPC platforms and it is free (BSD style license)
- TAU has instrumentation, measurement and analysis tools
- Using TAU requires setting a couple of environment variables and substituting the name of the compiler with a TAU shell script

Using TAU: A Tutorial

Using TAU: A brief Introduction

- To instrument source code using PDT
  - Choose an appropriate TAU stub makefile in <arch>/lib:

% setenv TAU_MAKEFILE /usr/common/acts/TAU/tau_latest/ibm64/lib/Makefile.tau-mpi-pdt

% setenv TAU_OPTIONS '-optVerbose …' (see tau_compiler.sh)

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

% mpif90 foo.f90

changes to

% tau_f90.sh foo.f90

- Execute application and analyze performance data:

% pprof (for text based profile display)

% paraprof (for GUI)

TAU Measurement Configuration – Examples

% cd /usr/common/acts/TAU/tau-2.18.2p4/ibm64/lib; ls Makefile.*
Makefile.tau-pdt
Makefile.tau-mpi-pdt
Makefile.tau-papi-mpi-pdt
Makefile.tau-pthread-pdt
Makefile.tau-pthread-mpi-pdt
Makefile.tau-openmp-opari-pdt
Makefile.tau-openmp-opari-mpi-pdt
Makefile.tau-papi-openmp-opari-mpi-pdt
…

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

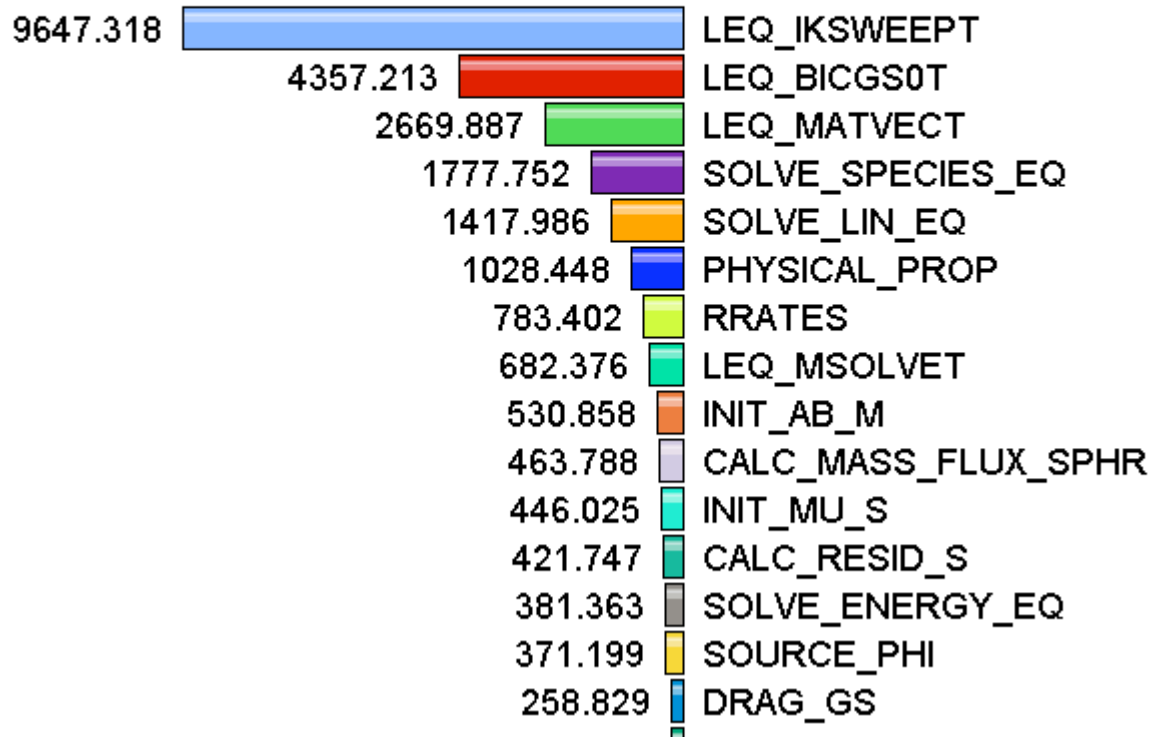
Makefile.tau-mpi-pdt

- Supports MPI instrumentation & PDT for automatic source instrumentation
- % setenv TAU_MAKEFILE /usr/common/acts/TAU/tau-2.18.2p4/ibm64/lib/Makefile.tau-mpi-pdt

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

```
% setenv TAU_MAKEFILE /usr/common/acts/TAU/tau_latest/ibm64
                                /lib/Makefile.tau-mpi-pdt
% set path=(/usr/common/acts/TAU/tau_latest/ibm64/bin $path)
Or
% module load tau
% make F90=tau_f90.sh
Or
% tau_f90.sh matmult.f90 -o matmult
(Or edit Makefile and change F90=tau_f90.sh)
% qsub run.job
% paraprof
To view. To view the data locally on the workstation,
% 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()

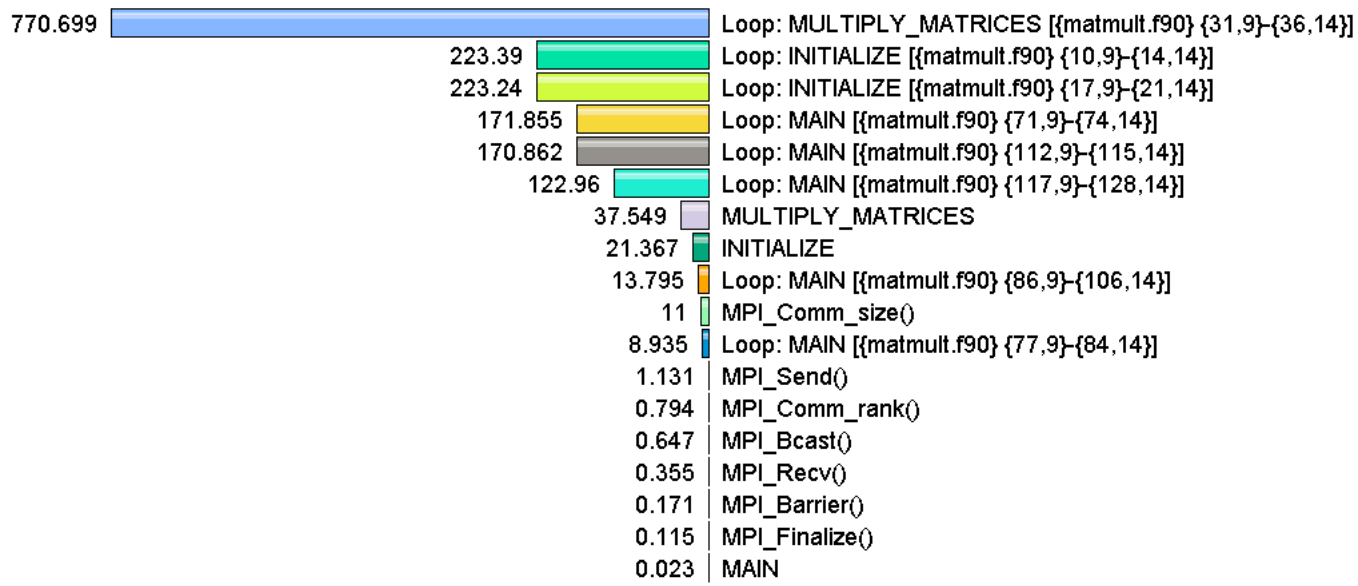
Solution: Generating a loop level profile

```
% setenv TAU_MAKEFILE /usr/common/acts/TAU/tau_latest/ibm64
                                /lib/Makefile.tau-mpi-pdt
% setenv TAU_OPTIONS '-optTauSelectFile=select.tau -optVerbose'
% cat select.tau
  BEGIN_INSTRUMENT_SECTION
  loops routine="#"
  END_INSTRUMENT_SECTION

% module load tau
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% qsub  run.job
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.

% paraprof app.ppk
```

Usage Scenarios: MFlops in Loops

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

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

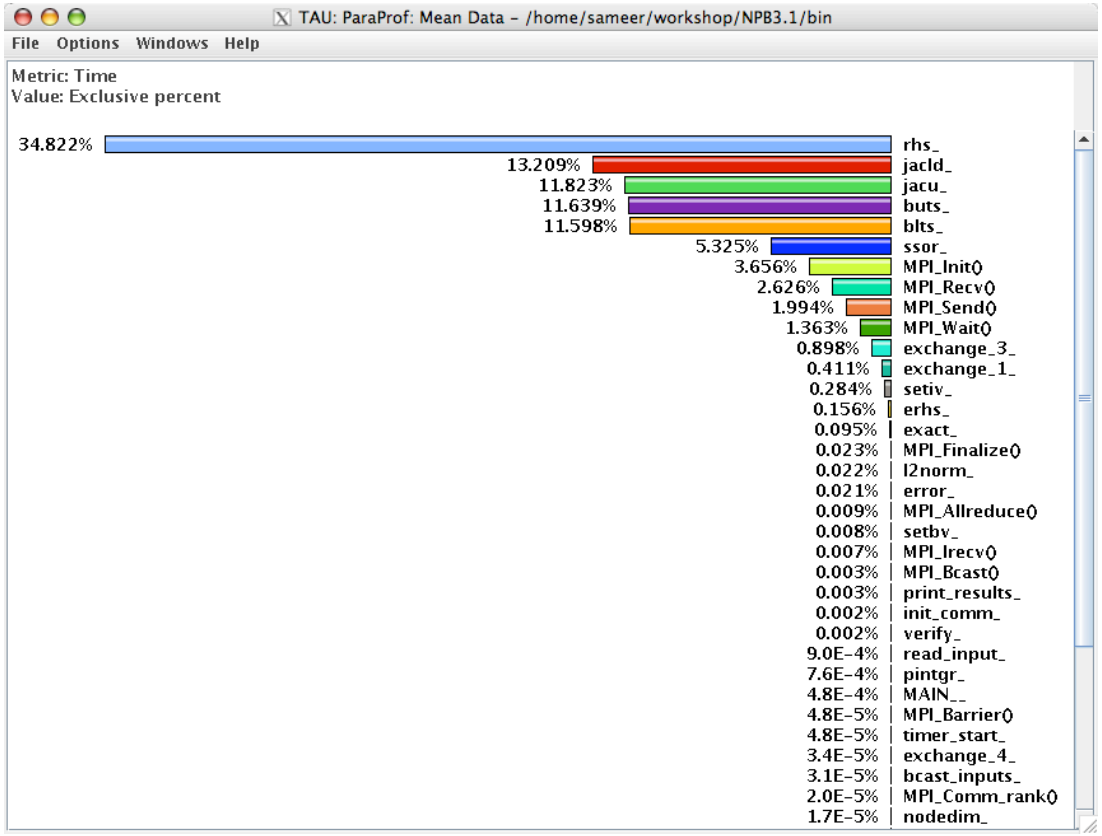


Generate a PAPI profile with 2 or more counters

```
% setenv TAU_MAKEFILE /usr/common/acts/TAU/tau_latest/ibm64
              /lib/Makefile.tau-papi-mpi-pdt
% setenv TAU_OPTIONS '-optTauSelectFile=select.tau -optVerbose'
% cat select.tau
  BEGIN_INSTRUMENT_SECTION
  loops routine="#"
  END_INSTRUMENT_SECTION

% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% setenv TAU_METRICS TIME:PAPI_FP_INS
% qsub  run.job
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk
  Choose Options -> Show Derived Panel -> Arg 1 = PAPI_FP_INS,
 Arg 2 = GET_TIME_OF_DAY, Operation = Divide -> Apply, choose.
```

Usage Scenarios: Compiler-based Instrumentation

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

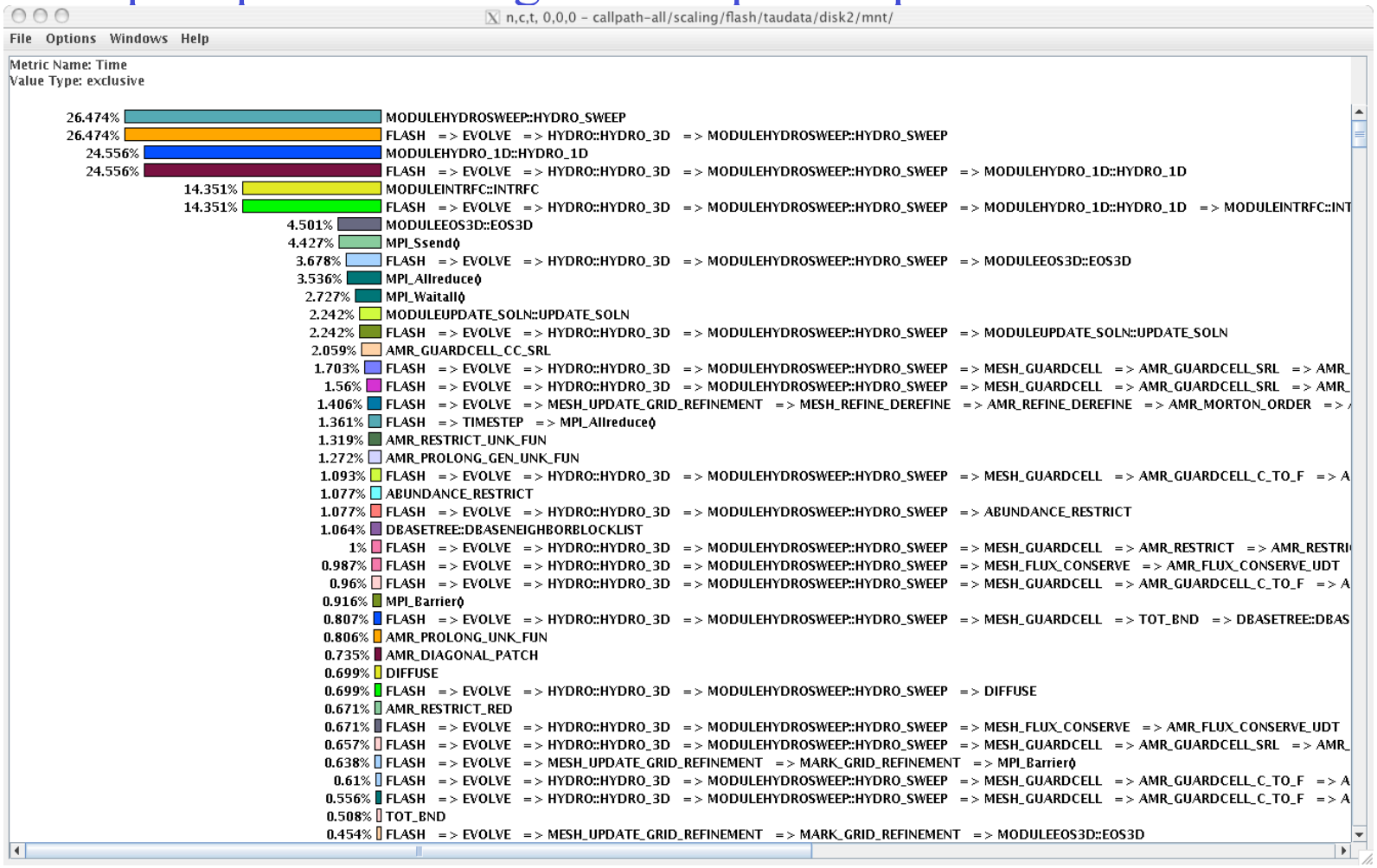


Use Compiler-Based Instrumentation

```
% setenv TAU_MAKEFILE /usr/common/acts/TAU/tau_latest/ibm64
                /lib/Makefile.tau-mpi-pdt
% setenv TAU_OPTIONS '-optCompInst -optVerbose'
% module load tau
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)

% qsub  run.job
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk
```

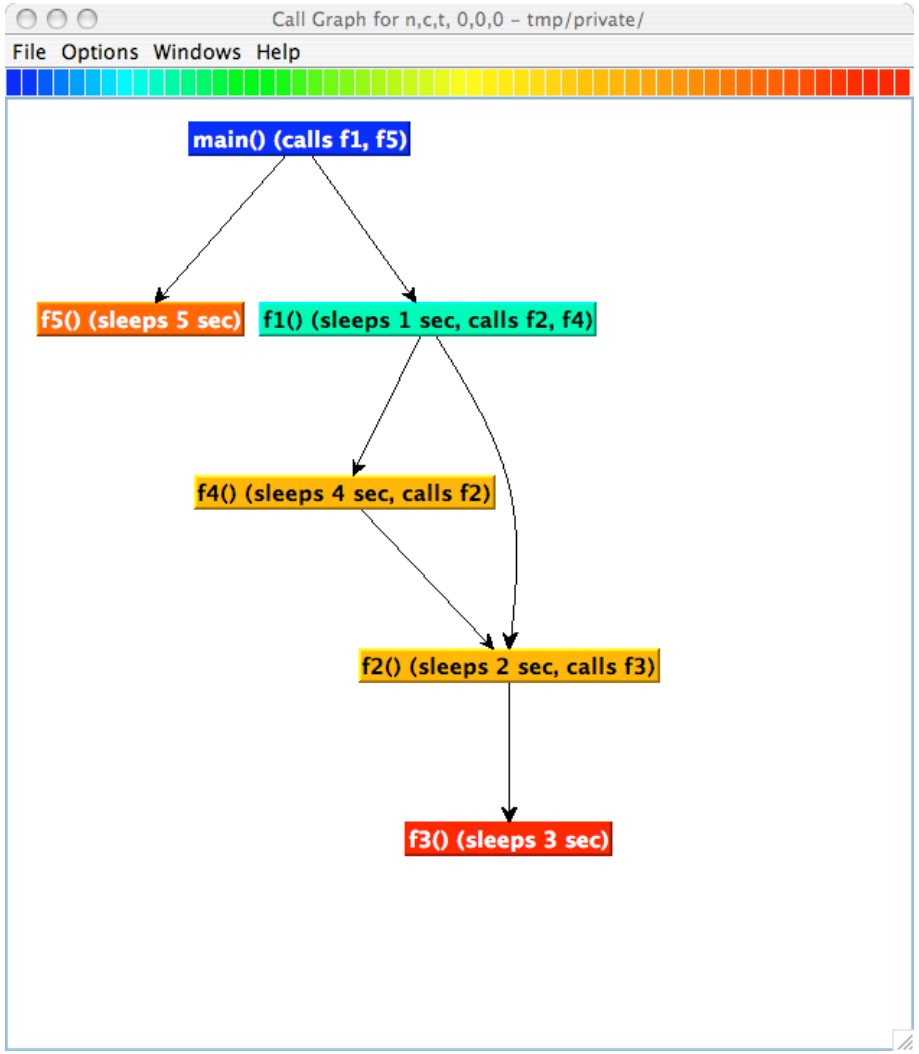
Usage Scenarios: Generating Callpath Profile

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



Callpath Profile

- Generates program callgraph



Generate a Callpath Profile

```
% setenv TAU_MAKEFILE /usr/common/acts/TAU/tau_latest/ibm64
              /lib/Makefile.tau-mpi-pdt
% set path=(/usr/common/acts/TAU/tau_latest/ibm64/bin $path)
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% setenv TAU_CALLPATH 1
% setenv TAU_CALLPATH_DEPTH 100


% qsub run.job
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk
(Windows -> Thread -> Call Graph)
```

Usage Scenario: Detect Memory Leaks

TAU: ParaProf: Mean Context Events - mem.ppk

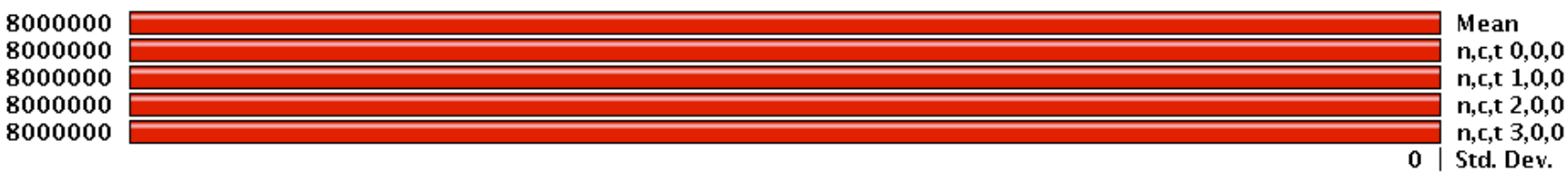
File Windows Help

Name	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
MAIN [{matrix.f90} {141,7}-{146,22}]					
MATRICES::ALLOCATE_MATRICES [{matrix.f90} {10,7}-{13,38}]					
MEMORY LEAK! malloc size <file=matrix.f90, variable=C, line=11>	1	8,000,000	8,000,000	8,000,000	0
malloc size <file=matrix.f90, variable=A, line=11>	1	8,000,000	8,000,000	8,000,000	0
malloc size <file=matrix.f90, variable=B, line=11>	1	8,000,000	8,000,000	8,000,000	0
malloc size <file=matrix.f90, variable=C, line=11>	1	8,000,000	8,000,000	8,000,000	0
MATRICES::DEALLOCATE_MATRICES [{matrix.f90} {14,7}-{17,40}]					
free size <file=matrix.f90, variable=A, line=15>	1	8,000,000	8,000,000	8,000,000	0
free size <file=matrix.f90, variable=B , line=15>	1	8,000,000	8,000,000	8,000,000	0

User Event Window: mem.ppk

File Options Windows Help

Name: MEMORY LEAK! malloc size <file=matrix.f90, variable=C, line=11> : MAIN [{matrix.f90} {141,7}-{146,22}] => MATRICES::ALLOCATE_MATRICES [{matrix.f90} {10,7}-{13,38}]
Value Type: Max Value



Detect Memory Leaks

```
% setenv TAU_MAKEFILE /usr/common/acts/TAU/tau_latest/ibm64
                /lib/Makefile.tau-mpi-pdt
% setenv TAU_OPTIONS '-optDetectMemoryLeaks -optVerbose'
% module load tau
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% setenv TAU_CALLPATH_DEPTH 100

% qsub  run.job
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk
(Windows -> Thread -> Context Event Window -> Select thread -> select...
    expand tree)
(Windows -> Thread -> User Event Bar Chart -> right click LEAK
-> Show User Event Bar Chart)
```

Usage Scenarios: Mixed Python+F90+C+pyMPI

- Goal: Generate multi-level instrumentation for Python+MPI+…

n,c,t 0,0,0 - scipy/taudata/rs/sameer/Users/

File Options Windows Help

Metric: Time
Value: Exclusive percent

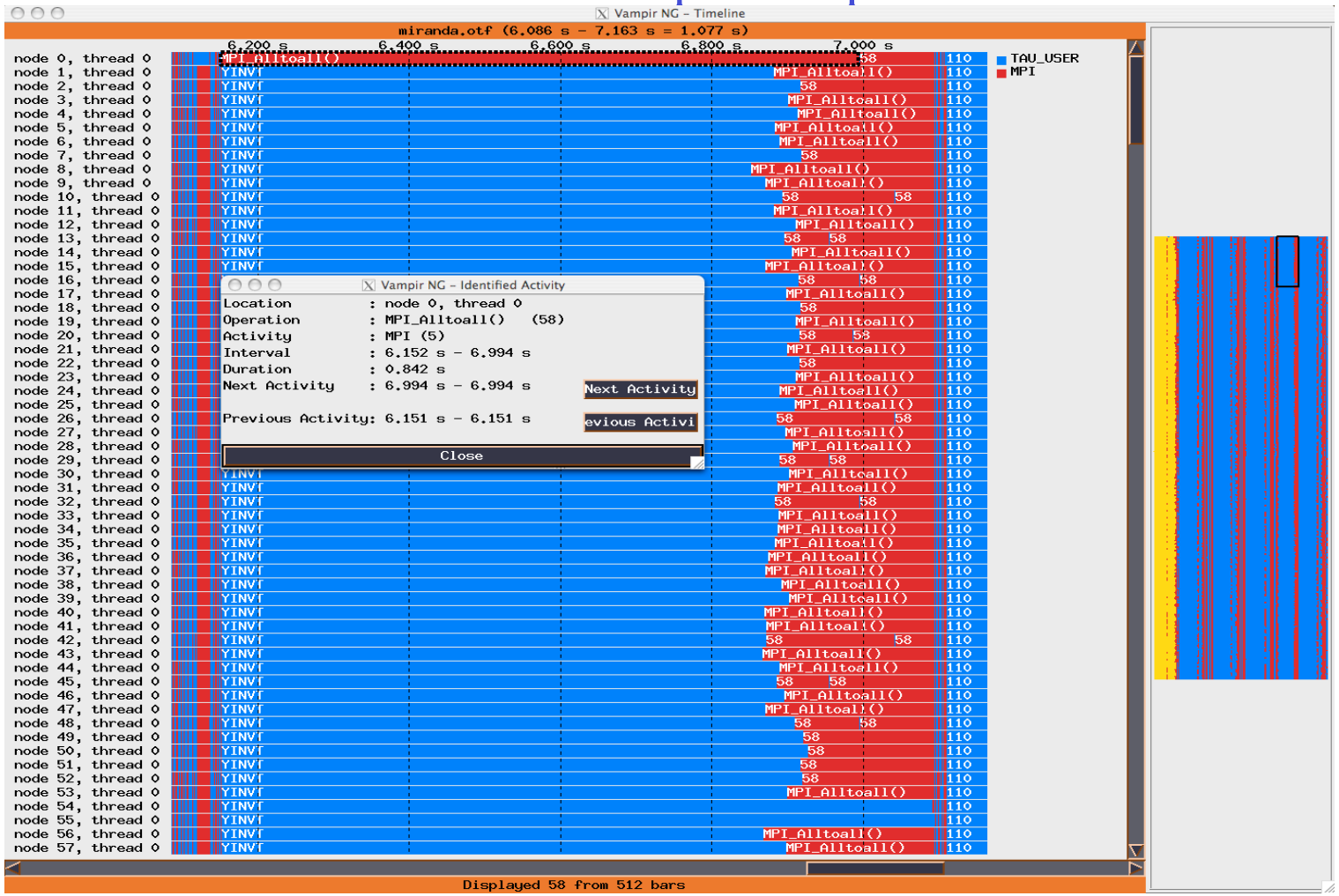
31.656%	write_array [/usr/lib/python2.4/site-packages/Gnuplot/utils.py, line=46]
26.056%	? [<string>, line=1]
13.3%	write
5.402%	__init__ [/usr/lib/python2.4/site-packages/Gnuplot/PlotItems.py, line=430]
4.561%	tolist
2.954%	start_new_thread
0.98%	join [/usr/lib/python2.4/string.py, line=308]
0.94%	choice [/usr/lib/python2.4/random.py, line=247]
0.803%	popen
0.777%	next [/usr/lib/python2.4/tempfile.py, line=127]
0.759%	__call__ [/usr/lib/python2.4/site-packages/Gnuplot/_Gnuplot.py, line=192]
0.561%	Data [/usr/lib/python2.4/site-packages/Gnuplot/PlotItems.py, line=476]
0.493%	get_command_option_string [/usr/lib/python2.4/site-packages/Gnuplot/PlotItems.py, line=177]
0.491%	len
0.42%	OurMain [hi.py, line=8]
0.414%	get
0.384%	apply
0.378%	normpath [/usr/lib/python2.4/posixpath.py, line=374]
0.341%	seed [/usr/lib/python2.4/random.py, line=98]
0.339%	mktemp [/usr/lib/python2.4/tempfile.py, line=337]
0.334%	start [/usr/lib/python2.4/threading.py, line=408]
0.319%	_get_default_tempdir [/usr/lib/python2.4/tempfile.py, line=176]
0.297%	urandom [/usr/lib/python2.4/os.py, line=711]
0.295%	refresh [/usr/lib/python2.4/site-packages/Gnuplot/_Gnuplot.py, line=206]

Generate a Multi-Language Profile w/ Python

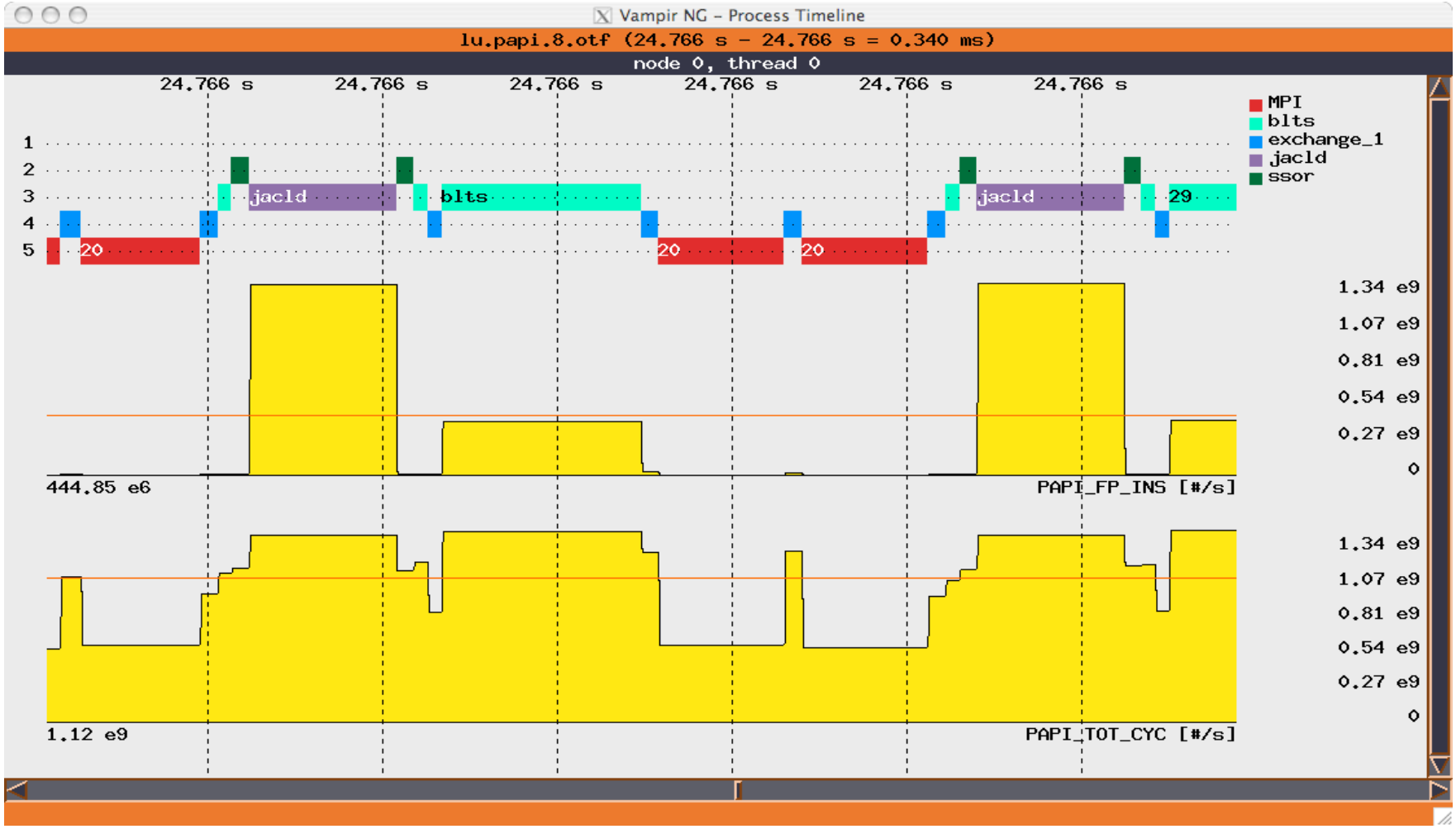
```
% setenv TAU_MAKEFILE /usr/common/acts/TAU/tau_latest/ibm64
                /lib/Makefile.tau-python-mpi-pdt
% set path=(/usr/common/acts/TAU/tau_latest/ibm64/bin $path)
% setenv TAU_OPTIONS '-optShared -optVerbose…'
(Python needs shared object based TAU library)
% make F90=tau_f90.sh CXX=tau_cxx.sh CC=tau_cc.sh  (build pyMPI w/TAU)
% cat wrapper.py
  import tau
  def OurMain():
      import App
  tau.run('OurMain()')
Uninstrumented:
% poe <dir>/pyMPI-2.4b4/bin/pyMPI ./App.py -procs 4
Instrumented:
% setenv PYTHONPATH <taudir>/ibm64/lib/bindings-python-mpi-pdt-pgi
(same options string as TAU_MAKEFILE)
setenv LD_LIBRARY_PATH <taudir>/ibm64/lib/bindings-icpc-python-mpi-pdt-
pgi\:$LD_LIBRARY_PATH
% poe <dir>/pyMPI-2.4b4-TAU/bin/pyMPI ./wrapper.py -procs 4
(Instrumented pyMPI with wrapper.py)
```

Usage Scenarios: Generating a Trace File

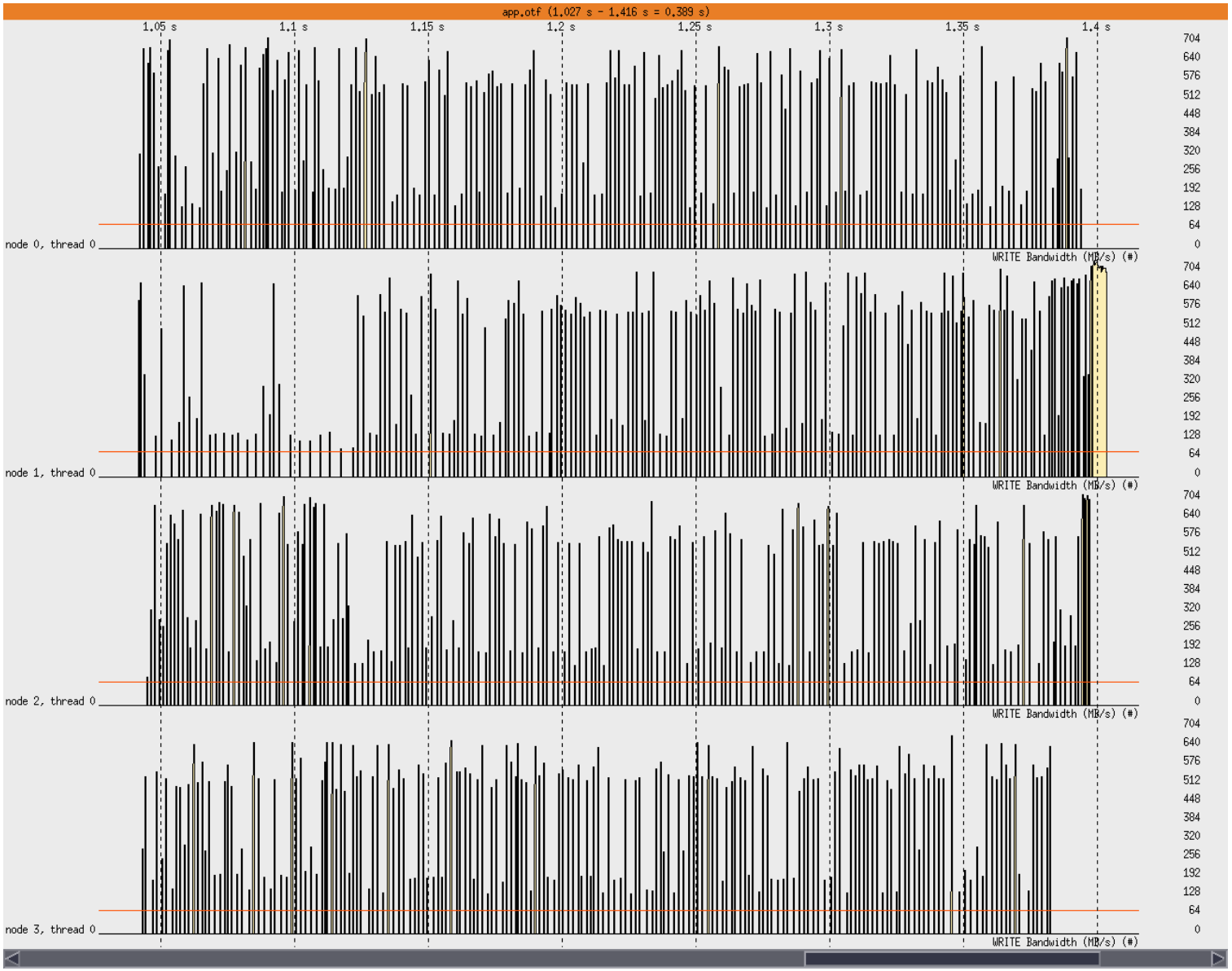
- Goal: What happens in my code at a given time? When?
- Event trace visualized in Vampir/Jumpshot



VNG Process Timeline with PAPI Counters



Vampir Counter Timeline Showing I/O BW

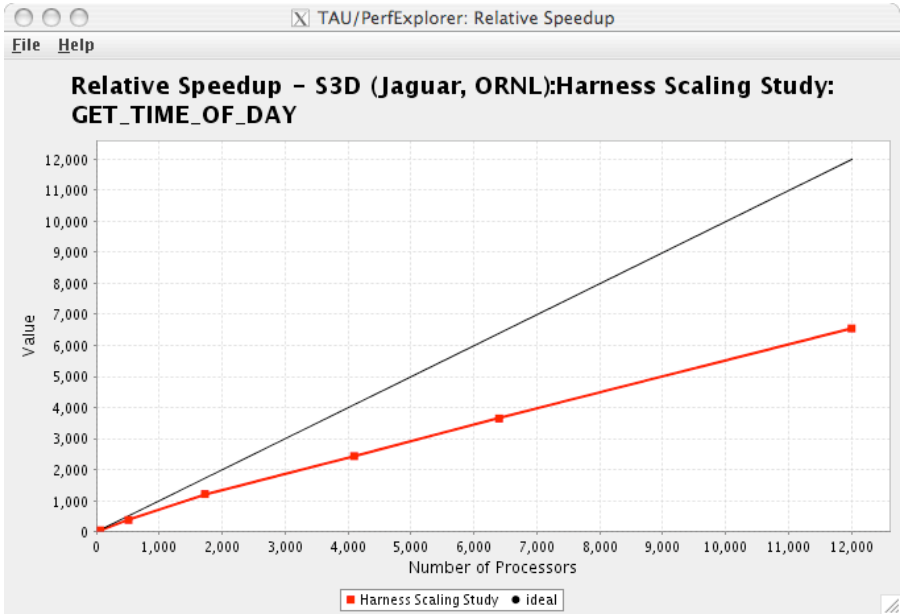


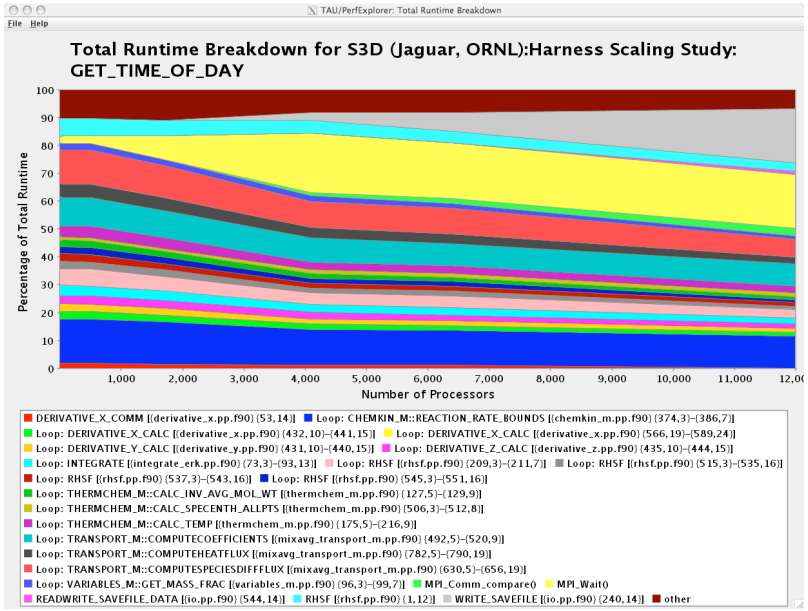
Generate a Trace File

```
% setenv TAU_MAKEFILE /usr/common/acts/TAU/tau_latest/ibm64
                /lib/Makefile.tau-mpi-pdt
% set path=(/usr/common/acts/TAU/tau_latest/ibm64/bin $path)
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% setenv TAU_TRACE 1
% qsub  run.job
% 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).
```

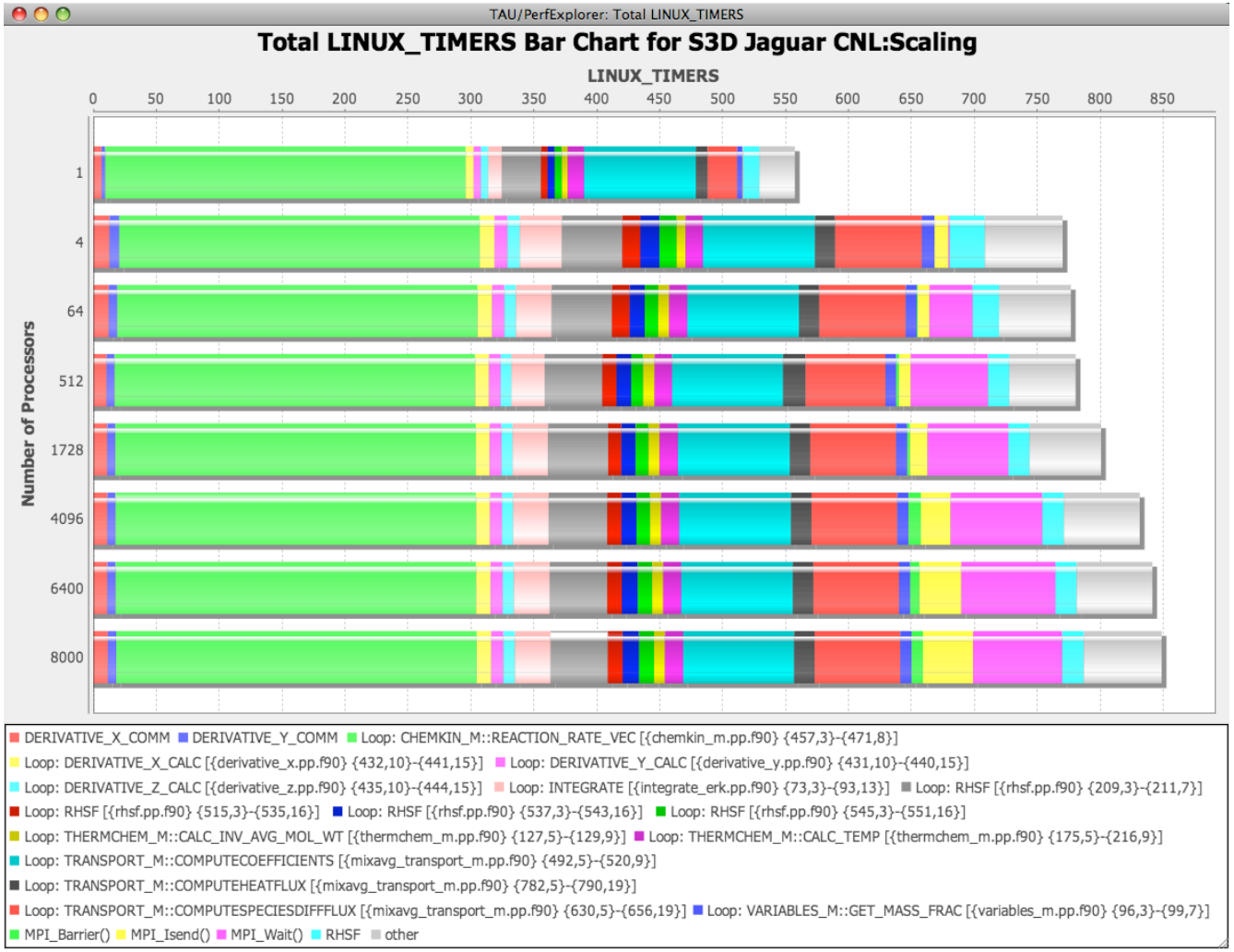
Usage Scenarios: Evaluate Scalability

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

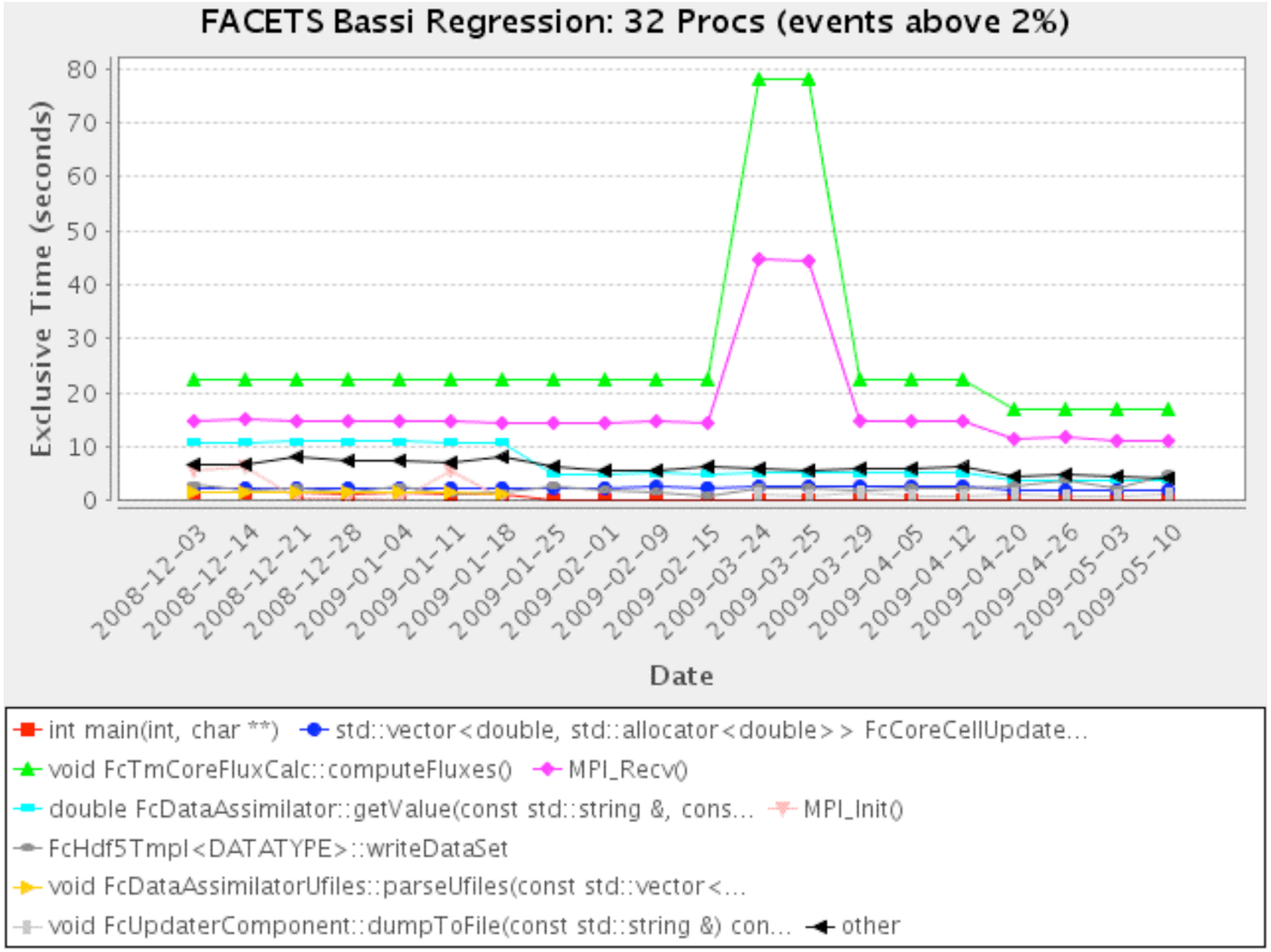




Usage Scenarios: Evaluate Scalability



Performance Regression Testing

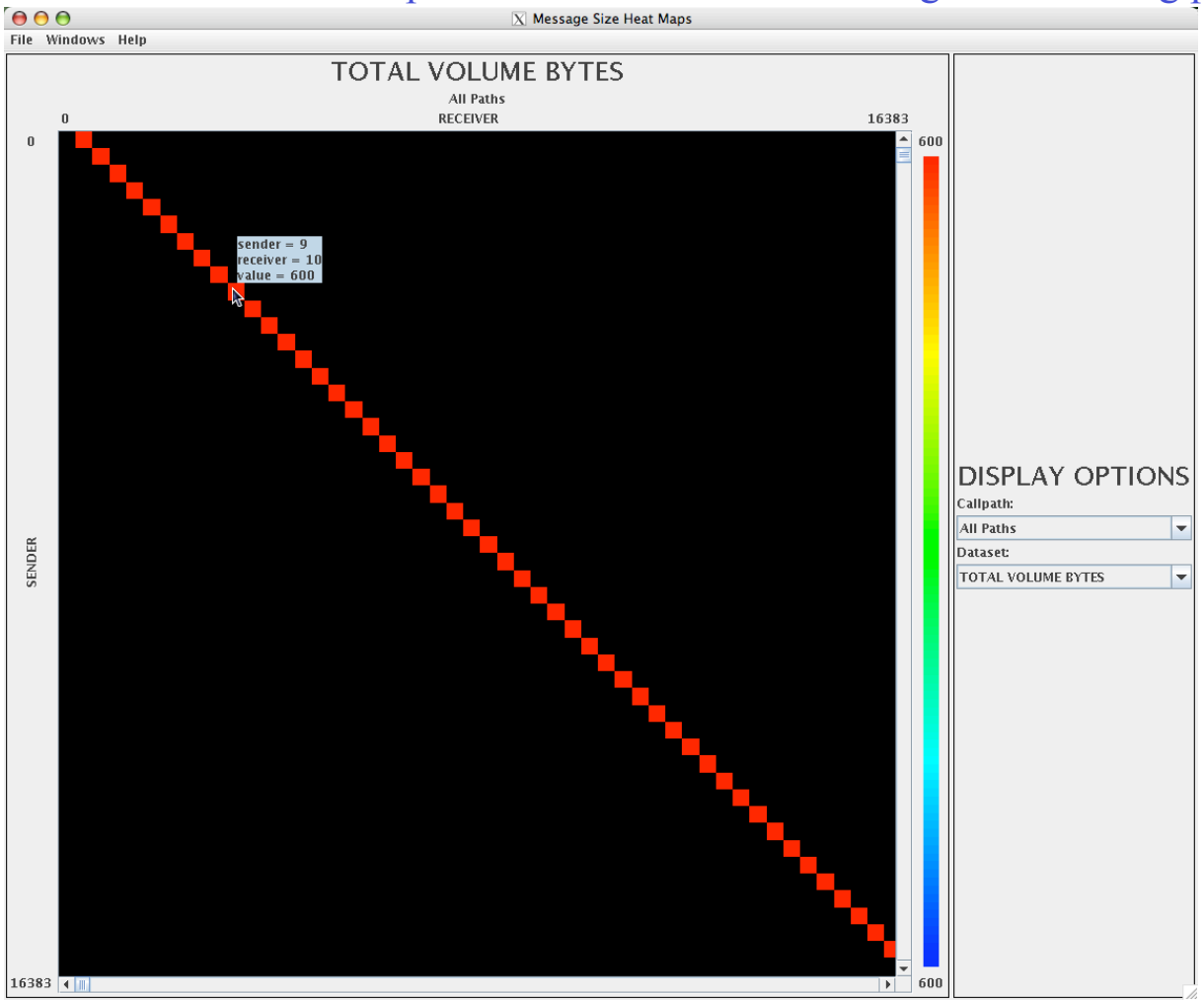


Evaluate Scalability using PerfExplorer Charts

```
% setenv TAU_MAKEFILE /usr/common/acts/TAU/tau_latest/ibm64
              /lib/Makefile.tau-mpi-pdt
% set path=(/usr/common/acts/TAU/tau_latest/ibm64/bin $path)
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% qsub  run1p.job
% paraprof --pack 1p.ppk
% qsub  run2p.job …
% paraprof --pack 2p.ppk … and so on.
On your client:
% perfdmf_configure
(Choose 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)
% perfexplorer
(Charts -> Speedup)
```

Communication Matrix Display

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

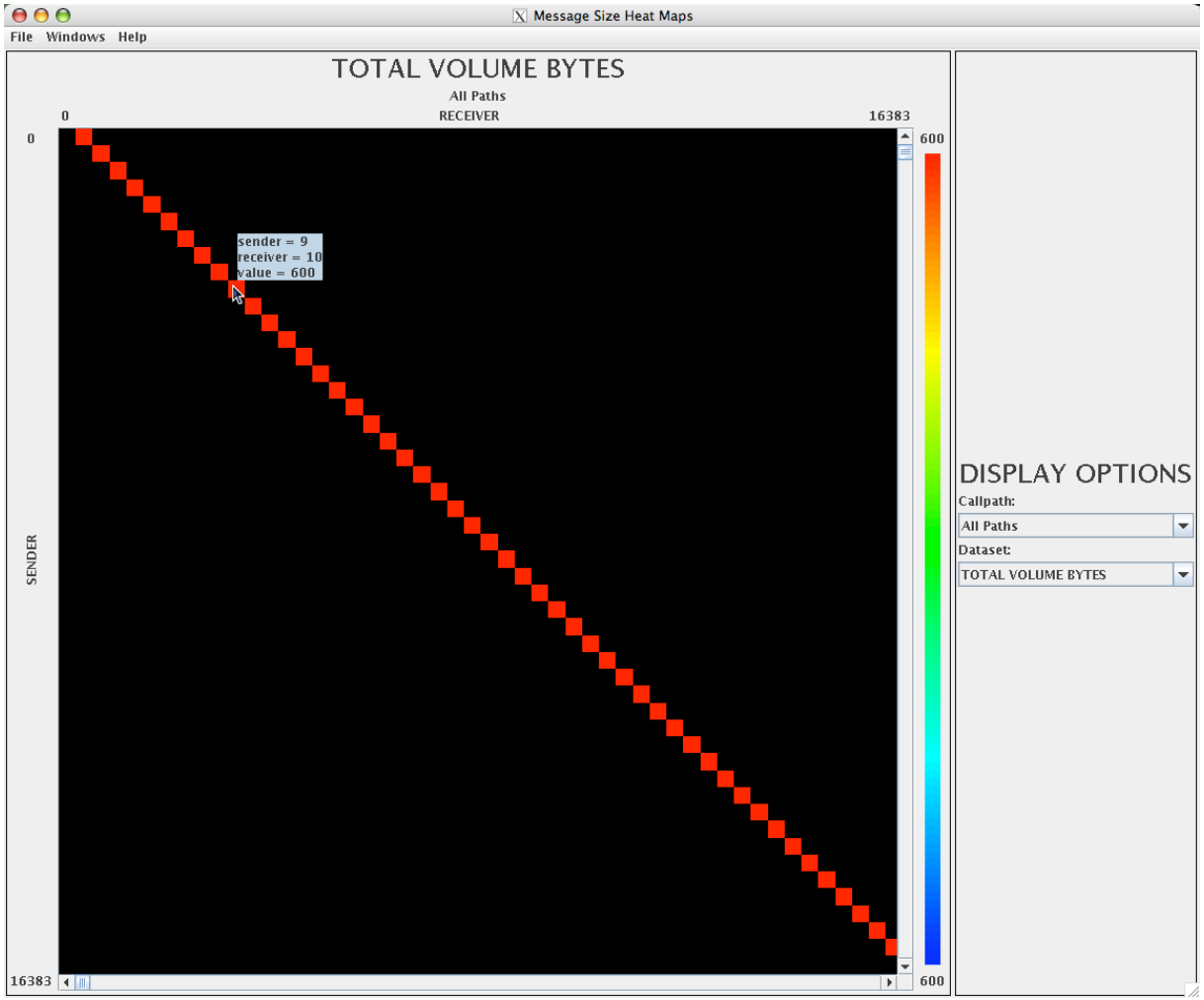


Evaluate Scalability using PerfExplorer Charts

```
% setenv TAU_MAKEFILE /usr/common/acts/TAU/tau_latest/ibm64
                              /lib/Makefile.tau-mpi-pdt
% module load tau
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% setenv TAU_COMM_MATRIX 1

% qsub run.job (setting the environment variables)

% paraprof
(Windows -> Communication Matrix)
```

Communication Matrix Display

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



Evaluate Scalability using PerfExplorer Charts

```
% setenv TAU_MAKEFILE
    /lib/Makefile.tau-mpi-pdt
% set path=(~sameer/pkgs/tau_latest/x86_64/bin $path)
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% setenv TAU_COMM_MATRIX 1

% qsub run.job (setting the environment variables)

% paraprof
(Windows -> Communication Matrix)
```

Part II: Rest of the talk

The Details...

Performance Tools FAQ/Concerns

- Does it automatically instrument my code? At the routine level? At the outer-loop level?
- Can it show me where time is spent in my code? PAPI Flops? L1 data cache misses? Can I measure more than one quantity in a trial?
- Does the tool support profiling (runtime summarization) as well as tracing (time-line based displays)? What about profile snapshots? Callpath (parent-child) profiles? Can I use it to easily benchmark codes?
- Can I observe the performance data at runtime as the application executes?
- Can it show me memory utilization? Memory leaks? Mallocs/frees? When and where?
- What about I/O? Can I observe bandwidth of reads/writes? Volume of I/O? What about Kernel events? User space+Kernel?
- What is the typical overhead? Can I reduce it to < 5%? < 1%? Can it compensate and remove timer overhead from performance data? Can it throttle away instrumentation in lightweight routines at runtime to reduce overhead?
- I already have profile data from <XYZ> tool. Can it import my legacy data?
- I prefer <XYZ> performance tool for visualization. Can it hook up with this tool? Are there converters?

Performance Tools FAQ/Concerns (contd.)

- Can I use it for multi-core CPUs? Compare the performance of application running on a single vs. multi-core processor? Can I observe multi-core data snoops, invalidates?
- Can I share the performance data with my colleagues in a secure manner (web/database)? Can it automatically track progress of my application over time (~ 6 mos)? Can I use it for scalability studies? Over multiple platforms?
- Are the GUI client tools available under Linux? MS Windows? Apple?
- Does it run on all Cray, IBM, SGI, HP … platforms? CNL? Catamount?
- Does it support MPI? MPI2? Threads? Hybrid MPI+Pthreads/MPI+OpenMP?
- Does it support Fortran? C++, C? Java? Python? Python+MPI+F90+C++…?
- Does it support Intel/PGI/PathScale/IBM/Cray/Sun compilers?
- Are tools available in command-line form & GUI? IDE GUI? Web-based? 3D?
- Is it already installed and supported on my HPC system? What about systems at NERSC? ANL? LLNL? LANL? NASA? DoD? NSF sites?...
- Is there support (phone/e-mail) available for the tool? Professional support? For instrumentation? Analysis?
- Will it work on the new <XYZ> HPC platform scheduled for release six months from now?
- Is it free? BSD license? …

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

Definitions – Profiling

- Profiling
  - Recording of summary information during execution
    - inclusive, exclusive time, # calls, hardware statistics, …
  - Reflects performance behavior of program entities
    - functions, loops, basic blocks
    - user-defined "semantic" entities
  - Very good for low-cost performance assessment
  - Helps to expose performance bottlenecks and hotspots
  - Implemented through
    - sampling: periodic OS interrupts or hardware counter traps
    - instrumentation: direct insertion of measurement code

Definitions – Tracing

- Tracing
  - Recording of information about significant points (events) during program execution
    - entering/exiting code region (function, loop, block, …)
    - thread/process interactions (e.g., send/receive message)
  - Save information in event record
    - timestamp
    - CPU identifier, thread identifier
    - Event type and event-specific information
  - Event trace is a time-sequenced stream of event records
  - Can be used to reconstruct dynamic program behavior
  - Typically requires code instrumentation

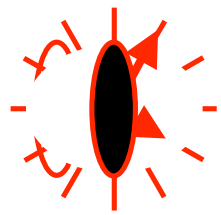
Event Tracing: Instrumentation, Monitor, Trace

CPU A:

```
void master {
  trace(ENTER, 1);
  ...
  trace(SEND, B);
  send(B, tag, buf);
  ...
  trace(EXIT, 1);
}
```

CPU B:

```
void worker {
  trace(ENTER, 2);
  ...
  recv(A, tag, buf);
  trace(RECV, A);
  ...
  trace(EXIT, 2);
}
```



timestamp

MONITOR

Event definition

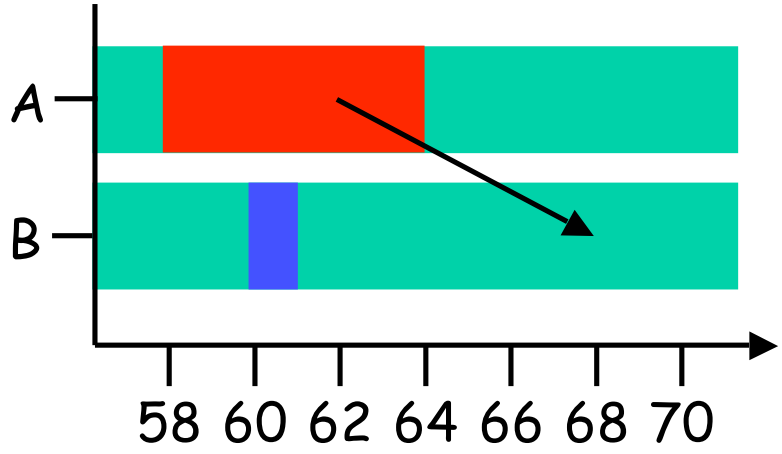
1	master
2	worker
3	...

...			
58	A	ENTER	1
60	B	ENTER	2
62	A	SEND	B
64	A	EXIT	1
68	B	RECV	A
69	B	EXIT	2
...			

Event Tracing: "Timeline" Visualization

1	master
2	worker
3	...

...			
58	A	ENTER	1
60	B	ENTER	2
62	A	SEND	B
64	A	EXIT	1
68	B	RECV	A
69	B	EXIT	2
...			



TAU Performance System Project

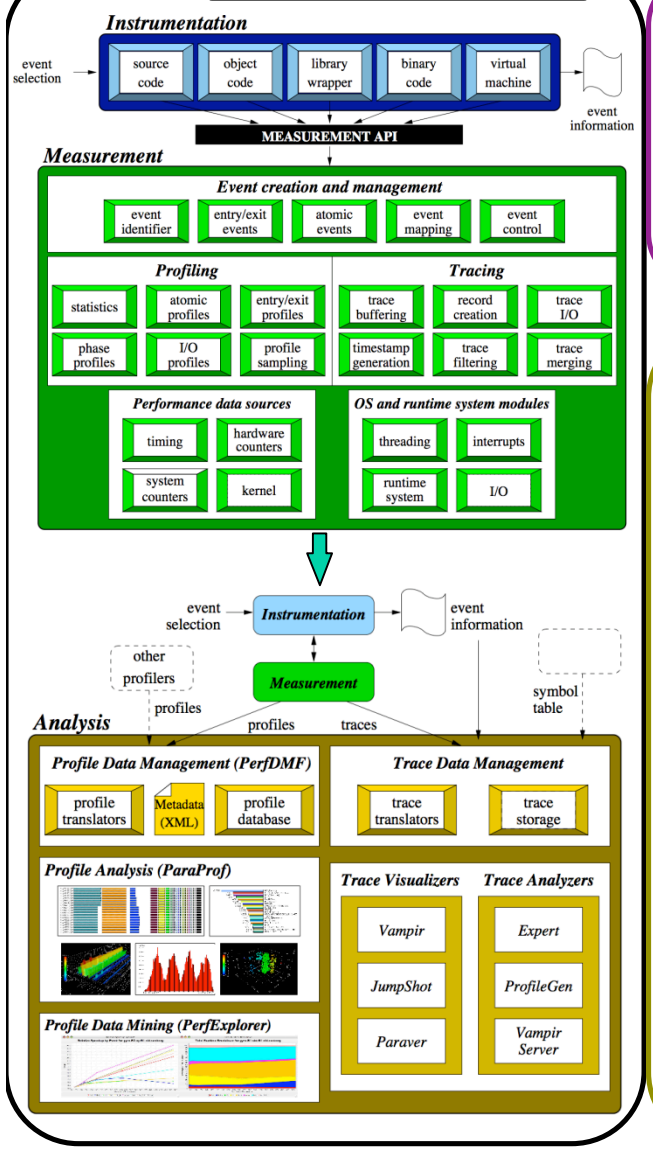
- ***T***uning and ***A***nalysis ***U***tilities (15+ year project effort)
- **Performance system framework for HPC systems**
  - Integrated, scalable, and flexible
  - Target parallel programming paradigms
- **Integrated toolkit for performance problem solving**
  - Instrumentation, measurement, analysis, and visualization
  - Portable performance profiling and tracing facility
  - Performance data management and data mining
- ***Partners***
  - LLNL, ANL, LANL
  - Research Centre Jülich, TU Dresden

TAU Parallel Performance System Goals

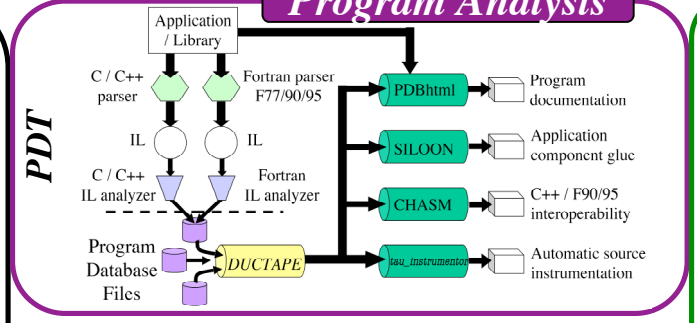
- **Portable (open source) parallel performance system**
  - Computer system architectures and operating systems
  - Different programming languages and compilers
- Multi-level, multi-language performance instrumentation
- **Flexible and configurable performance measurement**
- Support for multiple parallel programming paradigms
  - Multi-threading, message passing, mixed-mode, hybrid, object oriented (generic), component-based
- Support for performance mapping
- Integration of leading performance technology
- **Scalable (very large) parallel performance analysis**

TAU Performance System Components

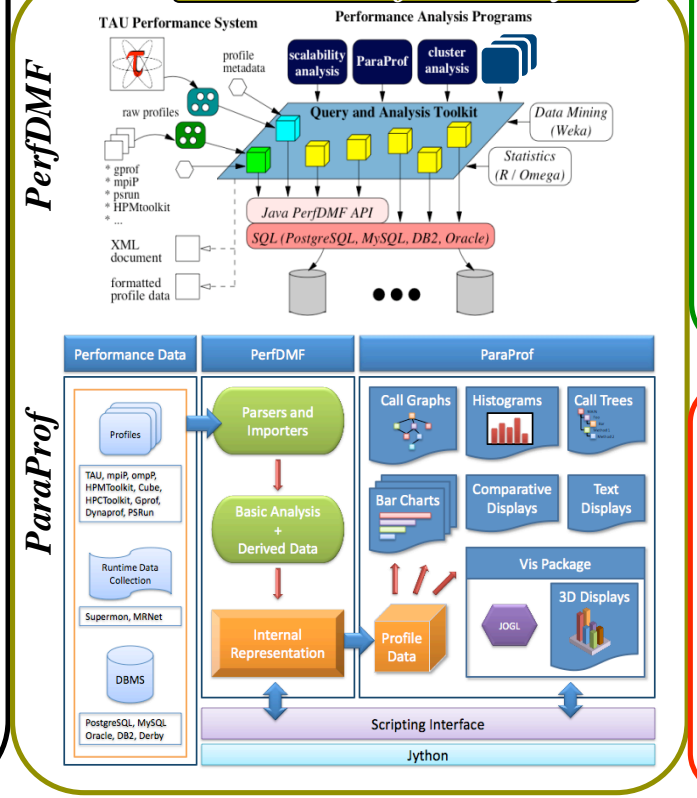
TAU Architecture



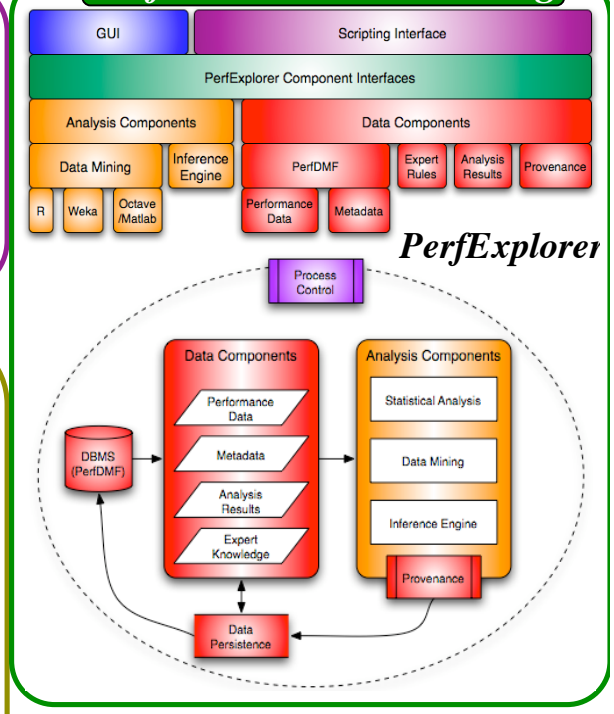
Program Analysis



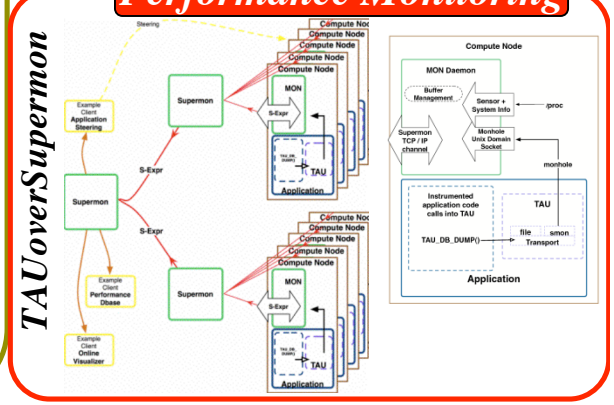
Parallel Profile Analysis



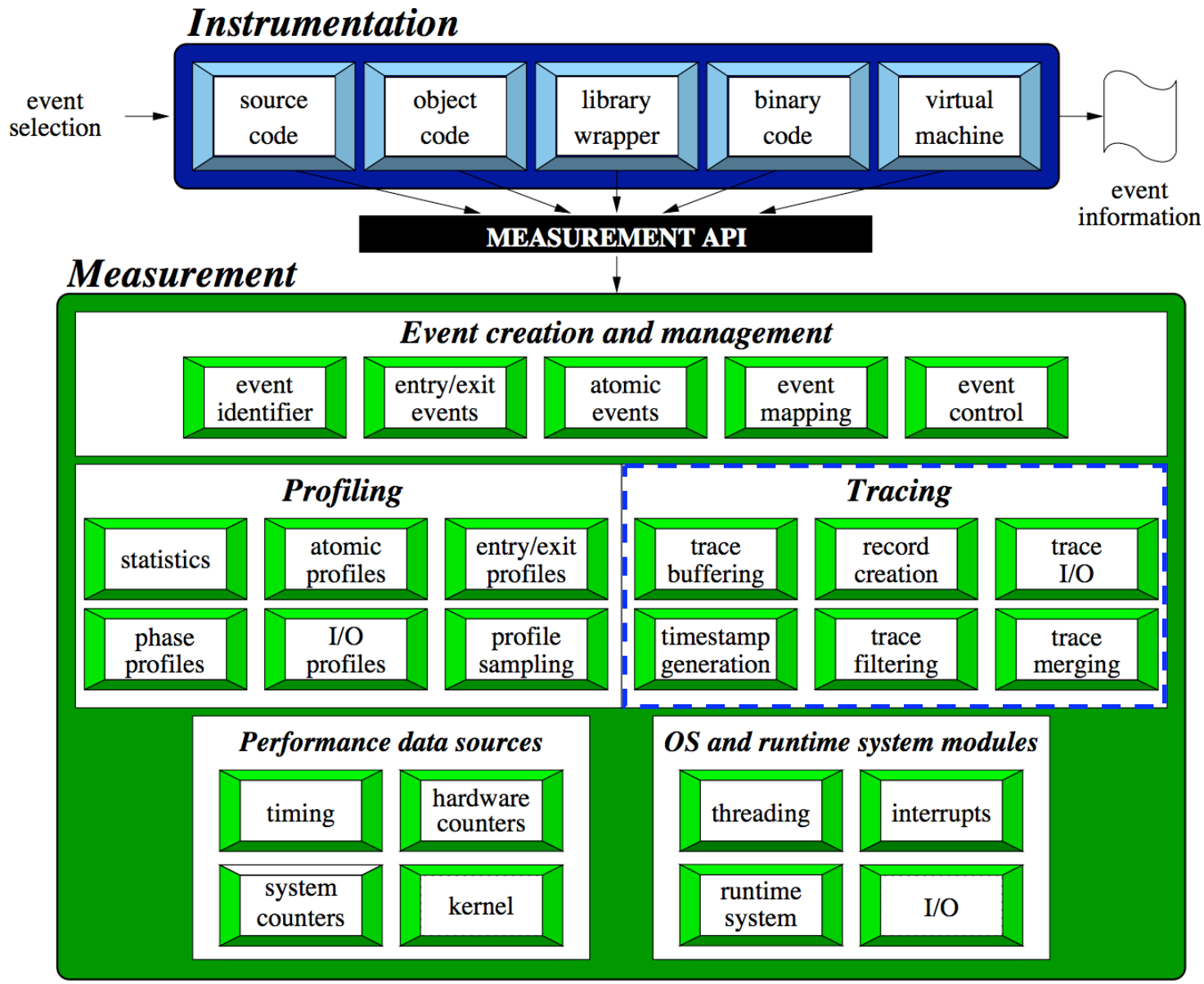
Performance Data Mining



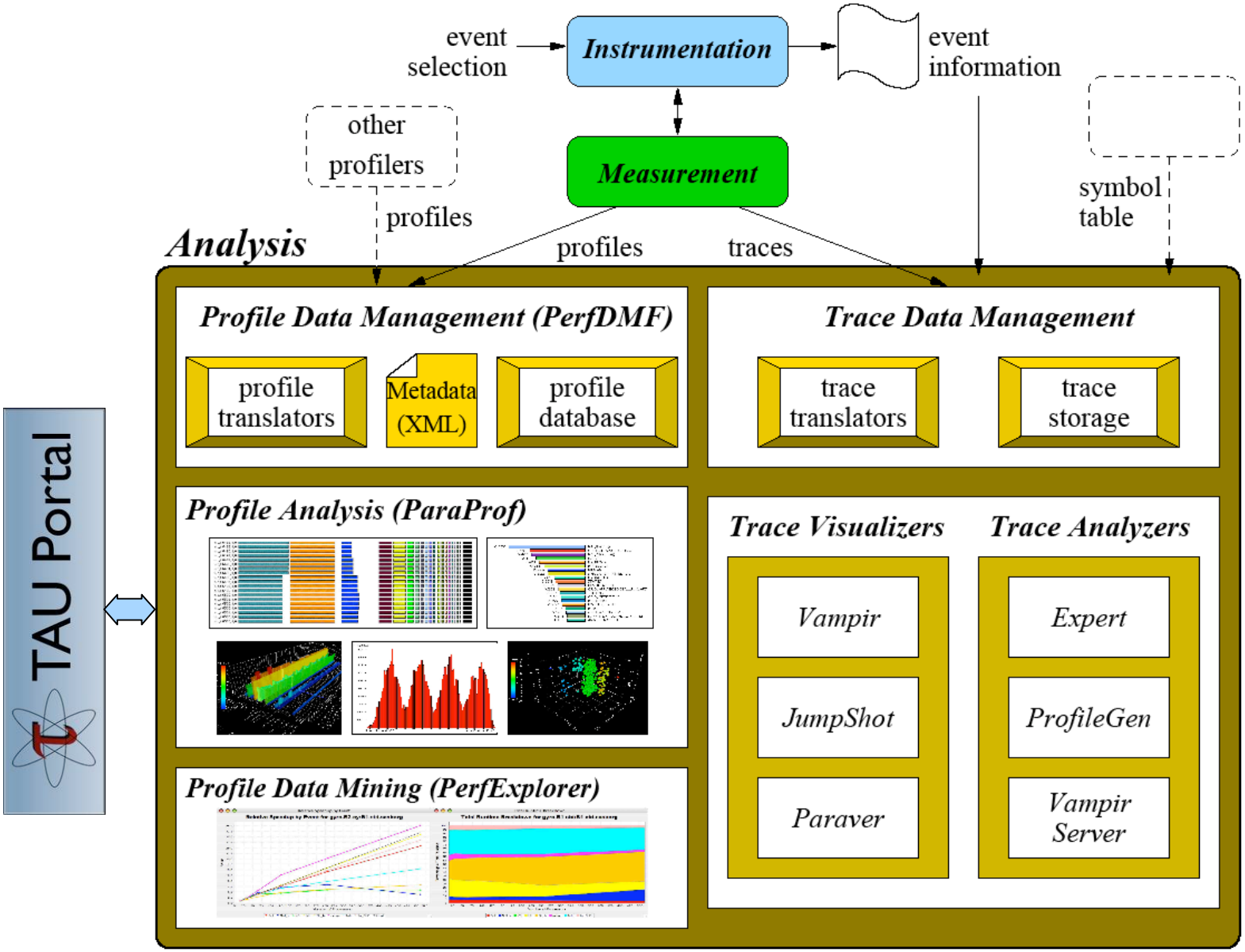
Performance Monitoring



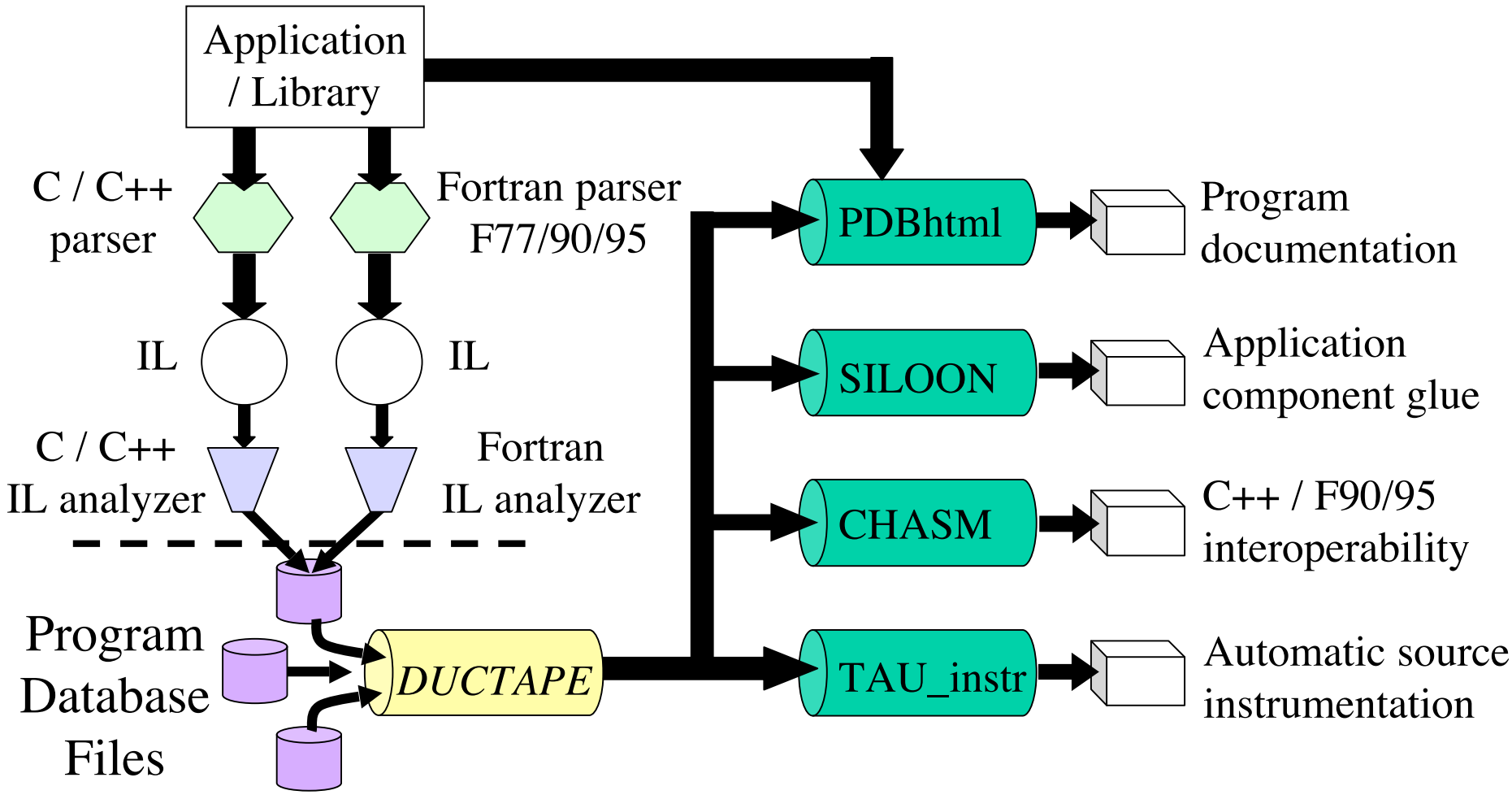
TAU Performance System Architecture



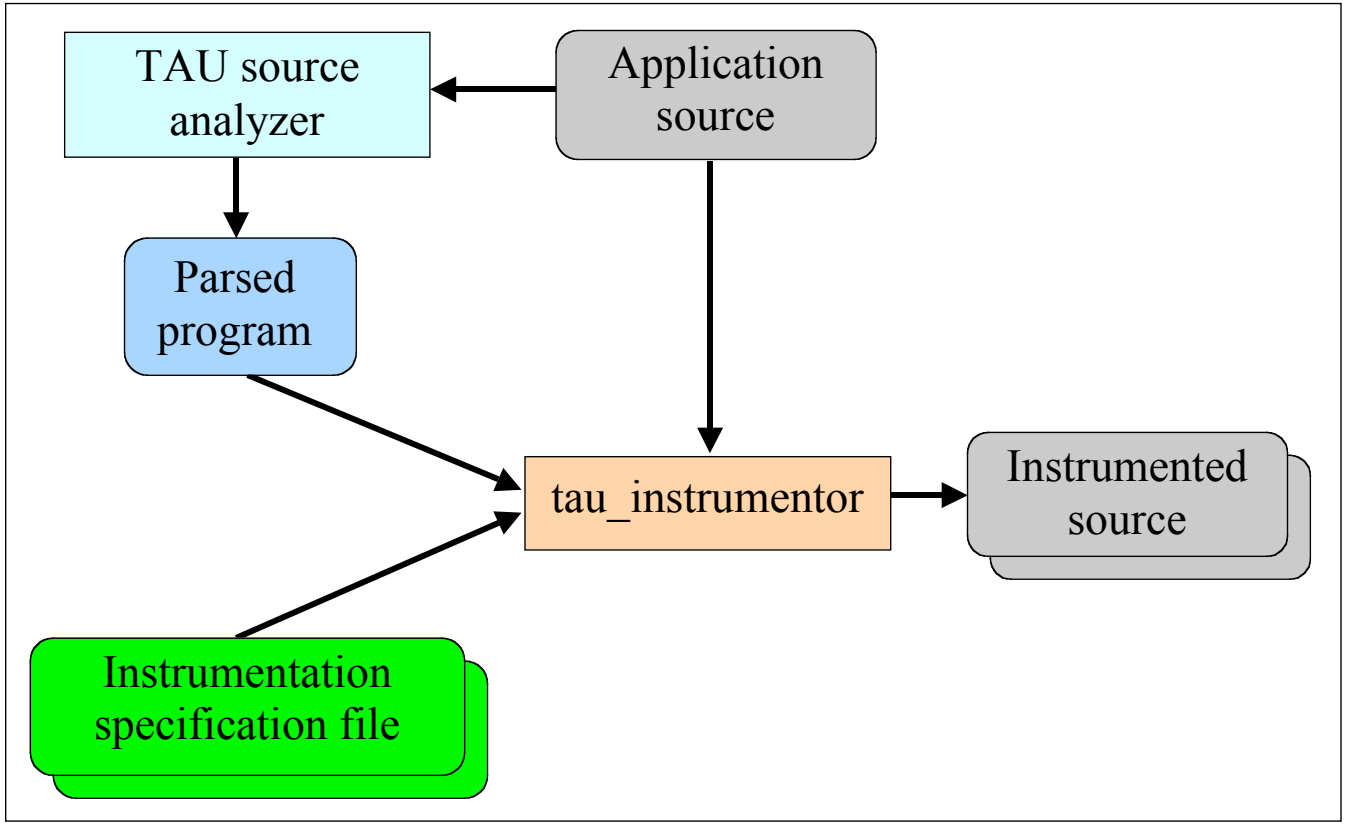
TAU Performance System Architecture



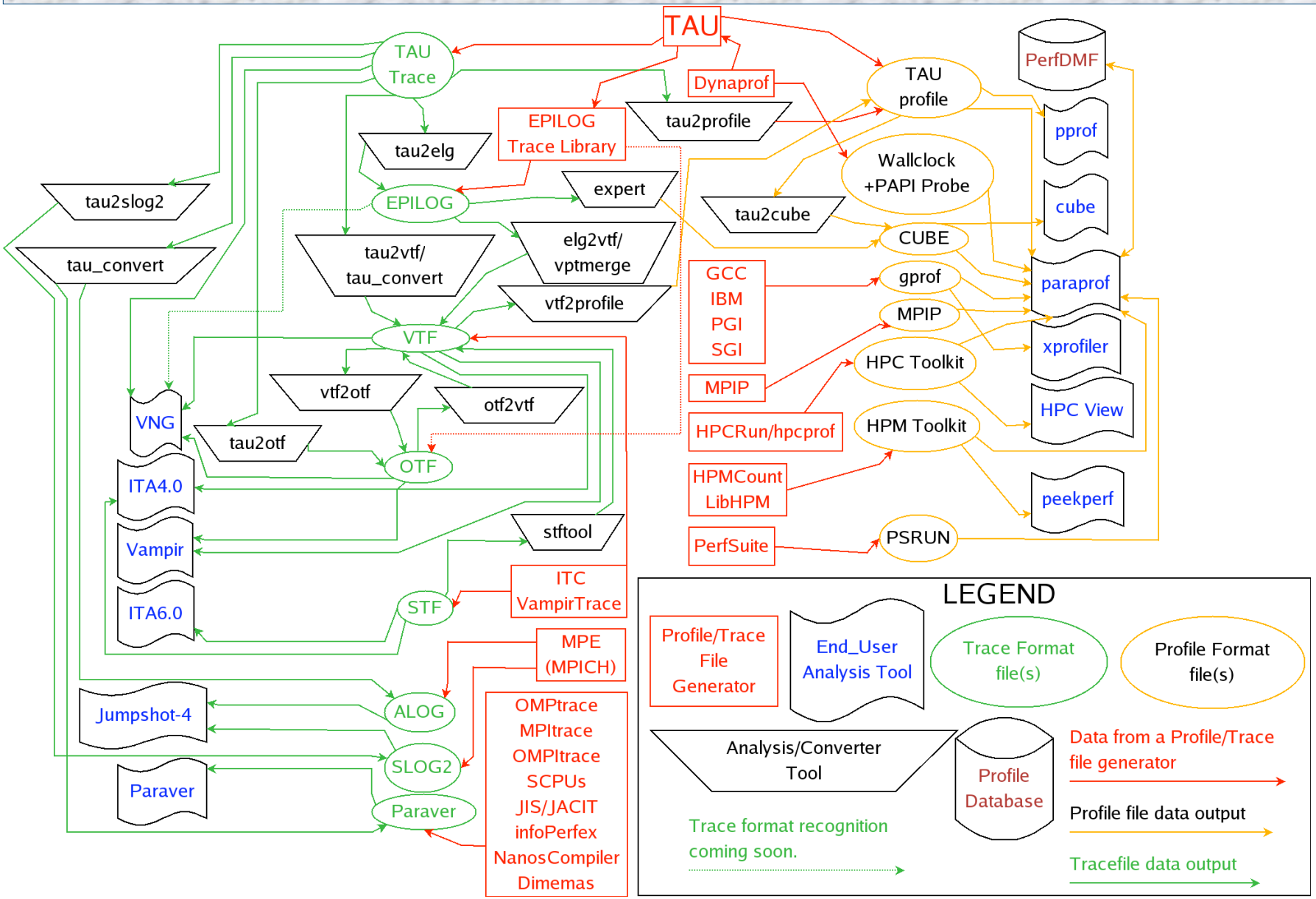
Program Database Toolkit (PDT)



Automatic Source-Level Instrumentation in TAU



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

TAU Instrumentation Mechanisms

- **Source code**
  - Manual (TAU API, TAU component API)
  - Automatic (robust)
    - C, C++, F77/90/95 (Program Database Toolkit (***PDT***))
    - OpenMP (directive rewriting (*Opari*), *POMP2* spec)
- **Object code**
  - Pre-instrumented libraries (e.g., MPI using *PMPI*)
  - Statically-linked and dynamically-linked
- **Executable code**
  - Binary and dynamic instrumentation (*Dyninst*)
  - Virtual machine instrumentation (e.g., Java using *JVMPI*)
- `TAU_COMPILER` to automate instrumentation process

Using TAU: A brief Introduction

- To instrument source code using PDT
  - Choose an appropriate TAU stub makefile (measurement option) from <taudir>/<arch>/lib directory:

% setenv TAU_MAKEFILE /usr/common/acts/TAU/tau_latest/ibm64/lib/Makefile.tau-mpi-pdt

% setenv TAU_OPTIONS '-optVerbose …' (see tau_compiler.sh)

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

% mpxlf90_r foo.f90

changes to

% tau_f90.sh foo.f90

- Execute application and analyze performance data:

% pprof (for text based profile display)

% paraprof (for GUI)

TAU Measurement Configuration – Examples

% cd /usr/common/acts/TAU/tau-2.18.2p4/ibm64/lib; ls Makefile.*
Makefile.tau-pdt
Makefile.tau-mpi-pdt
Makefile.tau-papi-mpi-pdt
Makefile.tau-pthread-pdt
Makefile.tau-pthread-mpi-pdt
Makefile.tau-openmp-opari-pdt
Makefile.tau-openmp-opari-mpi-pdt
Makefile.tau-papi-openmp-opari-mpi-pdt
…

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

Makefile.tau-mpi-pdt

- Supports MPI instrumentation & PDT for automatic source instrumentation
- % setenv TAU_MAKEFILE /usr/common/acts/TAU/tau-2.18.2p4/ibm64/lib/Makefile.tau-mpi-pdt

Installing TAU

- Install PAPI and PDT
  - PAPI:
    - ./configure –prefix=$HOME/pkgs/papi-3.6.2;
    - Make ; make install
  - PDT:
    - ./configure –prefix=$HOME/pkgs/pdt-3.14.1
    - make; make install
- Install TAU:
  - ./installtau –pdt=$HOME/pkgs/pdt-3.14.1 –papi=$HOME/pkgs/papi-3.6.2 –mpiinc=<dir> -mpilib=<dir>
    - Configures multiple typically requested versions for you in x86_64/lib/Makefile.tau-* configurations
  - tau_validate –html –build x86_64 >& results.html
  - mozilla results.html

Validating an Install

SeaMonkey

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop file:///home/sameer/tmp/resibm64.html Search Print

Home Bookmarks CentOS Support

```
Debug: Compiling with Instrumented Code

Executing> xlf90_r -I. -q64 -c simple.inst.f90 -qsuffix=f=f90 -I/u/sameer/pkgs/tau-2.17.1/include -I/usr/lpp/ppe.poe/include -I/usr/lpp/ssp/css/include -I/usr/lpp/ppe.poe/include/thread64 -o simple.o

** func   === End of Compilation 1 ===

** main   === End of Compilation 2 ===

1501-510  Compilation successful for file simple.inst.f90.

Looking for file: simple.o


        /u/sameer/pkgs/tau-2.17.1/ibm64/bin/tau_compiler.sh -optPdtDir="/u/sameer/pkgs/pdtoolkit-3.12/ibm64" -optPdtCOpts="-I/u/sameer/pkgs/tau-2.17.1/include -DPROFILING_ON           -DTAU_PAPI -I/u/sameer/pkgs/papi-3.5.0/src -I/u/sameer/pkgs/papi-3.5.0/inclu



Debug: Linking with TAU Options

Executing> xlf90_r -q64 simple.o -o simple -L/u/sameer/pkgs/tau-2.17.1/ibm64/lib -lTauMpi-phase-multiplecounters-mpi-papi-pdt -L/usr/lpp/ppe.poe/lib -lmpi_r -L/usr/lpp/ppe.poe/lib/threads -L/usr/lpp/ssp/css/lib -llapi_r -L/u/sameer/pkgs/tau-2.17.1/ibm64/lib -l


Target "all" is up to date.

/gpfs/home/sameer/pkgs/tau-2.17.1/examples/validate/pdtmpif> mkdir build-Makefile.tau-phase-multiplecounters-mpi-papi-pdt
/gpfs/home/sameer/pkgs/tau-2.17.1/examples/validate/pdtmpif> cp simple build-Makefile.tau-phase-multiplecounters-mpi-papi-pdt
pass/na   : PDT-MPI (Fortran)    : Makefile.tau-phase-multiplecounters-mpi-papi-pdt
```

PdtTest (/gpfs/home/sameer/pkgs/tau-2.17.1/ibm64/lib/Makefile.tau-phase-multiplecounters-mpi-papi-pdt)

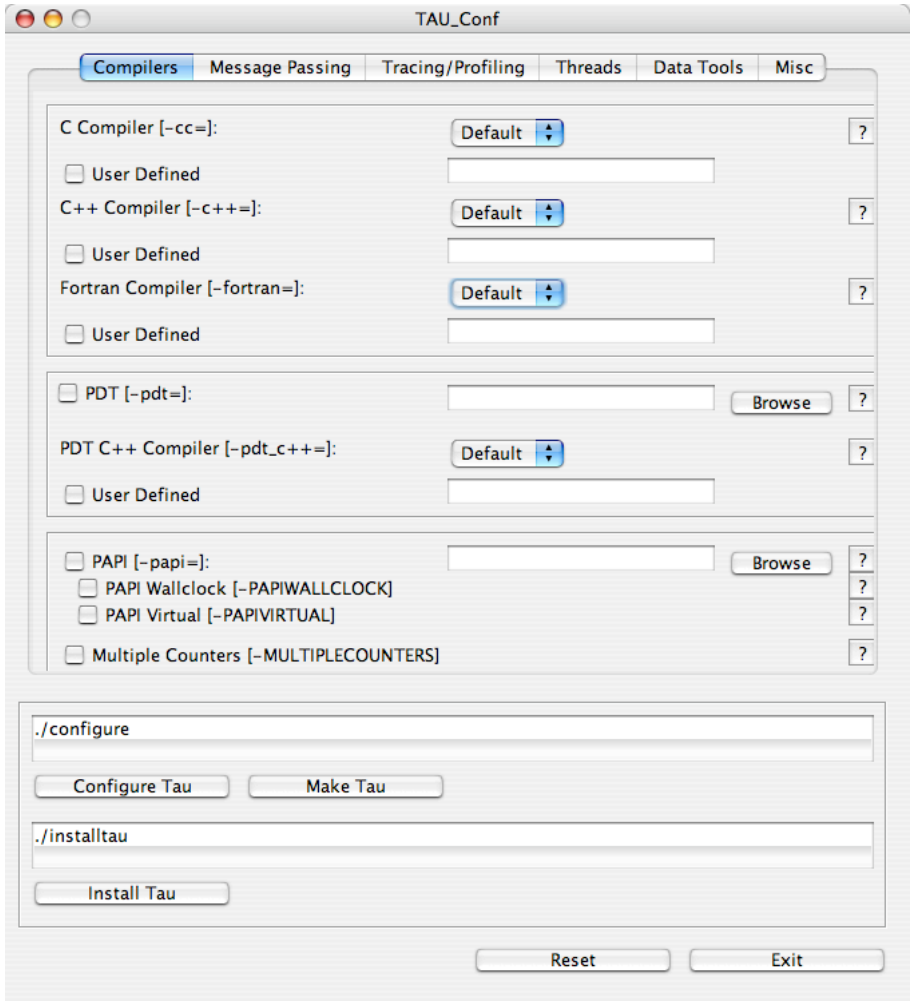
```
/gpfs/home/sameer/pkgs/tau-2.17.1/examples/validate/pdtmpif> grep "^TAU_F90 " /gpfs/home/sameer/pkgs/tau-2.17.1/ibm64/lib/Makefile.tau-phase-multiplecounters-mpi-papi-pdt
TAU_F90       = xlf90$(TAU_R)    $(F90_ABI) $(TAU_F90_OPT)   #ENDIF##IBM64_FORTRAN#

  na/na   : PDT-MPI (GFortran)    : Makefile.tau-phase-multiplecounters-mpi-papi-pdt
```

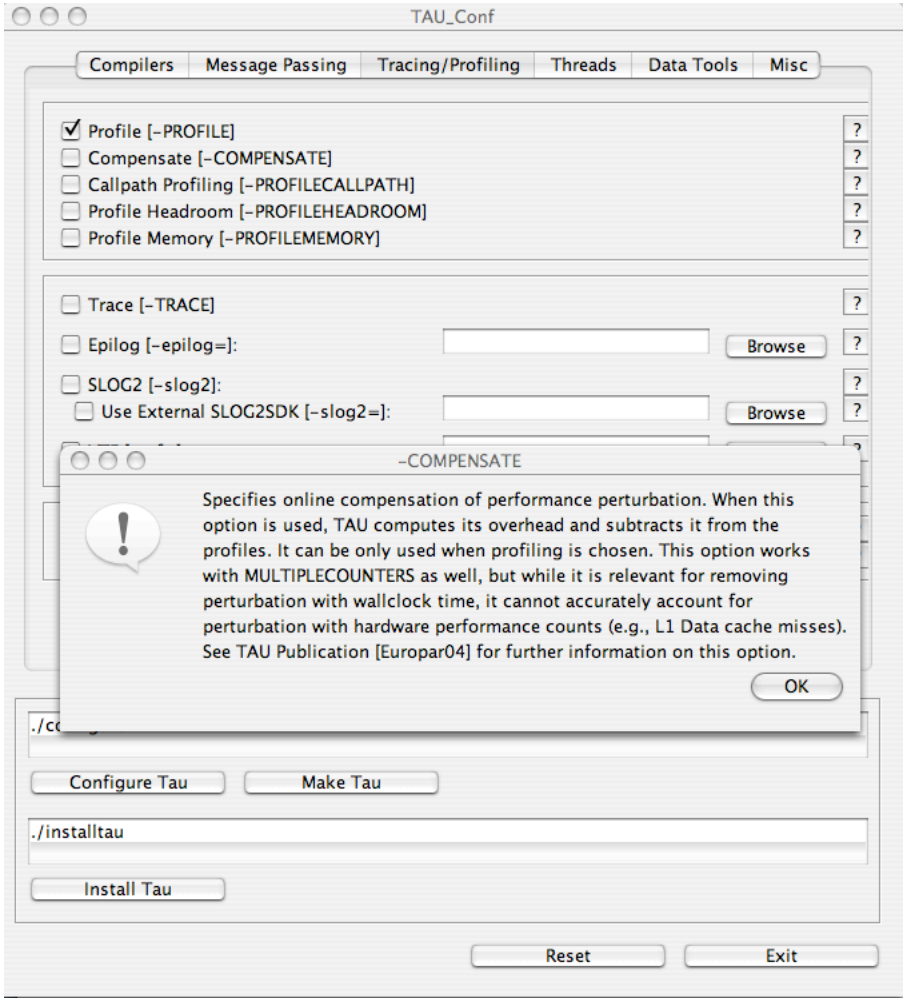
Stub Makefile	C		PDT (C)		PDT (C++)		PDT (Fortran)		PDT (GFortran)		Fortran (flink)		Fortran (cpplink)		Fortran (clink)		MPI (C)		MPI (Fortran)		PDT-MPI (C)		PDT-MPI (C++)		PDT-MPI (Fortran)		PDT-MPI (GFortran)	
	build	run	build	run	build	run	build	run	build	run	build	run	build	run	build	run	build	run	build	run	build	run	build	run	build	run	build	run
Makefile.tau-mpi-pdt	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A
Makefile.tau-mpi-pdt-trace	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A
Makefile.tau-pdt	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Makefile.tau-callpath-mpi-compensate-pdt	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A
Makefile.tau-mpi-compensate-pdt	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A
Makefile.tau-phase-multiplecounters-mpi-papi-compensate-pdt	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A
Makefile.tau-multiplecounters-papi-pthread-pdt	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Makefile.tau-multiplecounters-mpi-papi-pdt	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A
Makefile.tau-multiplecounters-papi-pdt	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Makefile.tau-multiplecounters-mpi-papi-pdt-trace	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A
Makefile.tau-depthlimit-mpi-pdt	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A
Makefile.tau-pthread-pdt	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Makefile.tau-callpath-mpi-pdt	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A
Makefile.tau-phase-multiplecounters-mpi-papi-pdt	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	pass	N/A	N/A	N/A

No Errors!

Done

TAU_SETUP: A GUI for Installing TAU





Upgrading TAU v2.17.1 configurations to 2.17.2

- Upgrade TAU
  - Previous installation in $HOME/pkgs/tau-2.18.2
    - cd tau-2.19
    - ./upgradetau $HOME/pkgs/tau-2.18.2
      - Builds all previous configurations in the current dir
      - You may also upgrade with a new package say PDT 3.15
    - ./upgradetau $HOME/pkgs/tau-2.18.2–pdt=$HOME/pkgs/pdt-3.15

- Validate your new installation
    - ./tau_validate –html –build x86_64 >& results.html
    - mozilla results.html

TAU Configuration – Examples

- ./configure -pdt=<dir> -mpi –arch=ibm64
  - on AIX 64 bit with PDT, MPI for IBM xlC compilers
- ./configure –papi=<dir> -pdt=<dir> -mpi -arch=ibm64
  make clean install
  - Use PAPI counters (one or more) with C/C++/F90 automatic instrumentation for AIX. Also instrument the MPI library.
- Typically configure multiple measurement libraries
  - .all_configs, .last_config files contain all and last configuration
  - tau_validate --html --build ibm64 >& results.html
  - ./upgradetau /path/to/old/tau-2.18
- Each configuration creates a unique <arch>/lib/Makefile.tau<options> stub makefile. It corresponds to the configuration options used. e.g.,
  - /usr/common/acts/TAU/tau_latest/ibm64/lib/Makefile.tau-mpi-pdt
  - /usr/common/acts/TAU/tau_latest/ibm64/lib/Makefile.tau-papi-mpi-pdt

Using TAU

- Install TAU

  % ./configure [options]; make clean install
- Replace the names of your compiler with tau_f90.sh, tau_cxx.sh and tau_cc.sh in your makefiles
- Set environment variables
  - Choose the measurement option and compile your code:
    - setenv TAU_MAKEFILE $TAU/Makefile.tau-papi-mpi-pdt
    - setenv TAU_OPTIONS '-optCompInst -optVerbose'
      - Uses compiler-based instrumentation
  - At runtime, if more than one metric desired:
    - Setenv TAU_METRICS TIME:PAPI_FP_INS:PAPI_NATIVE_<event>
      - Use papi_native_avail, papi_avail, and papi_event_chooser to select these preset and native event names
- Build the application, run it, analyze performance data

PAPI

- **P**erformance **A**pplication **P**rogramming **I**nterface
  - The purpose of the PAPI project is to design, standardize and implement a portable and efficient API to access the hardware performance monitor counters found on most modern microprocessors.
- Parallel Tools Consortium project started in 1998
- Developed by University of Tennessee, Knoxville
- http://icl.cs.utk.edu/papi/

Papi_avail

- List of PAPI preset counters:

```
cfe1.sameer 66> ./papi_avail | more
Available events and hardware information.
--------------------------------------------------------------------------------
Vendor string and code   : GenuineIntel (1)
Model string and code    : Itanium 2 (1)
CPU Revision             : 5.000000
CPU Megahertz            : 1500.000000
CPU's in this Node       : 28
Nodes in this System     : 1
Total CPU's              : 28
Number Hardware Counters : 4
Max Multiplex Counters   : 32
--------------------------------------------------------------------------------
The following correspond to fields in the PAPI_event_info_t structure.

Name            Code            Avail   Deriv   Description (Note)
PAPI_L1_DCM     0x80000000      Yes     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_L2_ICM     0x80000003      Yes     No      Level 2 instruction cache misses
PAPI_L3_DCM     0x80000004      Yes     Yes     Level 3 data cache misses
PAPI_L3_ICM     0x80000005      Yes     No      Level 3 instruction cache misses
PAPI_L1_TCM     0x80000006      Yes     Yes     Level 1 cache misses
PAPI_L2_TCM     0x80000007      Yes     No      Level 2 cache misses


...
```

Papi_native_avail

- List of PAPI native counters:

```
cfe1.sameer 67> ./papi_native_avail | more
Available native events and hardware information.
--------------------------------------------------------------------------
Vendor string and code   : GenuineIntel (1)
Model string and code    : Itanium 2 (1)
CPU Revision             : 5.000000
CPU Megahertz            : 1500.000000
CPU's in this Node       : 28
Nodes in this System     : 1
Total CPU's              : 28
Number Hardware Counters : 4
Max Multiplex Counters   : 32
--------------------------------------------------------------------------
The following correspond to fields in the PAPI_event_info_t structure.

Symbol                           Event Code   Long Description
 Register Name[n]
 Register Value[n]
ALAT_CAPACITY_MISS_ALL           0x40000000   ALAT Entry Replaced -- both integer and floating point i
nstructions
ALAT_CAPACITY_MISS_FP            0x40000001   ALAT Entry Replaced -- only floating point instructions
ALAT_CAPACITY_MISS_INT           0x40000002   ALAT Entry Replaced -- only integer instructions
```

Papi_event_chooser on IA-64

- List of PAPI PRESET counters that may be counted together:

```
cfe1.sameer 68> ./papi_event_chooser
Usage: eventChooser NATIVE|PRESET evt1 evet2 ...

cfe1.sameer 72> ./papi_event_chooser PRESET PAPI_FP_OPS PAPI_L1_DCM PAPI_TOT_CYC
Test case eventChooser: Available events which can be added with given events.
--------------------------------------------------------------------------
Vendor string and code   : GenuineIntel (1)
Model string and code    : Itanium 2 (1)
CPU Revision             : 5.000000
CPU Megahertz            : 1500.000000
CPU's in this Node       : 28
Nodes in this System     : 1
Total CPU's              : 28
Number Hardware Counters : 4
Max Multiplex Counters   : 32
--------------------------------------------------------------------------
Name            Derived Description (Mgr. Note)
PAPI_L1_ICM     No      Level 1 instruction cache misses ()
PAPI_L2_ICM     No      Level 2 instruction cache misses ()
PAPI_L3_ICM     No      Level 3 instruction cache misses

...
```

PAPI_L1_ICM may be counted with these counters on IA64

Automatic Instrumentation

- TAU provides compiler wrapper scripts
  - Simply replace `mpxlf90_r` with `tau_f90.sh`
  - Automatically instruments Fortran source code, links with TAU MPI Wrapper libraries.
- Use `tau_cc.sh` and `tau_cxx.sh` for C/C++

Before

```
F90 = mpif90
CXX = mpicxx
CFLAGS =
LIBS = -lm
OBJS = f1.o f2.o f3.o … fn.o

app: $(OBJS)
      $(F90) $(LDFLAGS) $(OBJS) -o $@
$(LIBS)
.f90.o:
      $(F90) $(CFLAGS) -c $<
```

After

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

app: $(OBJS)
      $(F90) $(LDFLAGS) $(OBJS) -o $@
$(LIBS)
.f90.o:
      $(F90) $(CFLAGS) -c $<
```

TAU_COMPILER Commandline Options

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

```
% mpxlf90 -c foo.f90
```

Changes to

```
% f95parse foo.f90 $(OPT1)
% tau_instrumentor foo.pdb foo.f90 -o foo.inst.f90 $(OPT2)
% mpxlf90 -c foo.f90 $(OPT3)
```

- Linking:

```
% mpxlf90 foo.o bar.o -o app
```

Changes to

```
% mpxlf90 foo.o bar.o -o app $(OPT4)
```

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

```
% setenv TAU_OPTIONS '...'
% make F90=tau_f90.sh
```

TAU_COMPILER Options

- Optional parameters for $(TAU_COMPILER): [tau_compiler.sh –help]

-optVerbose	Turn on verbose debugging messages
-optCompInst	Use compiler based instrumentation
-optDetectMemoryLeaks	Turn on debugging memory allocations/ de-allocations to track leaks
-optKeepFiles	Does not remove intermediate .pdb and .inst.* files
-optPreProcess	Preprocess Fortran sources before instrumentation
-optTauSelectFile=""	Specify selective instrumentation file for tau_instrumentor
-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)

...

Overriding Default Options: TAU_OPTIONS

```
% cat Makefile
F90 = tau_f90.sh
OBJS = f1.o f2.o f3.o …
LIBS = -Lappdir -lapplib1 -lapplib2 …

app: $(OBJS)
    $(F90) $(OBJS) -o app $(LIBS)
.f90.o:
    $(F90) -c $<
% setenv TAU_OPTIONS '-optVerbose -
optTauSelectFile=select.tau'

% setenv TAU_MAKEFILE <taudir>/ibm64/lib/Makefile.tau-icpc-
mpi-pdt
```

Compiling Fortran Codes with TAU: Tips

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

- If it uses several module files, you may switch from the default Cleanscape Inc. parser in PDT to the GNU gfortran parser to generate PDB files:
  % setenv TAU_OPTIONS '-optPdtGnuFortranParser -optVerbose'

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

- To use an instrumentation specification file:
  % setenv 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

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

Optimization of Program Instrumentation

- Need to eliminate instrumentation in frequently executing lightweight routines
- Throttling of events at runtime (set by default in TAU 2.17.2+):

```
% setenv TAU_THROTTLE 1
```

  Turns off instrumentation in routines that execute over 100000 times (TAU_THROTTLE_NUMCALLS) and take less than 10 microseconds of inclusive time per call (TAU_THROTTLE_PERCALL)

- Selective instrumentation file to filter events

```
% tau_instrumentor [options] -f <file>   OR
% setenv TAU_OPTIONS '-optTauSelectFile=tau.txt'
```

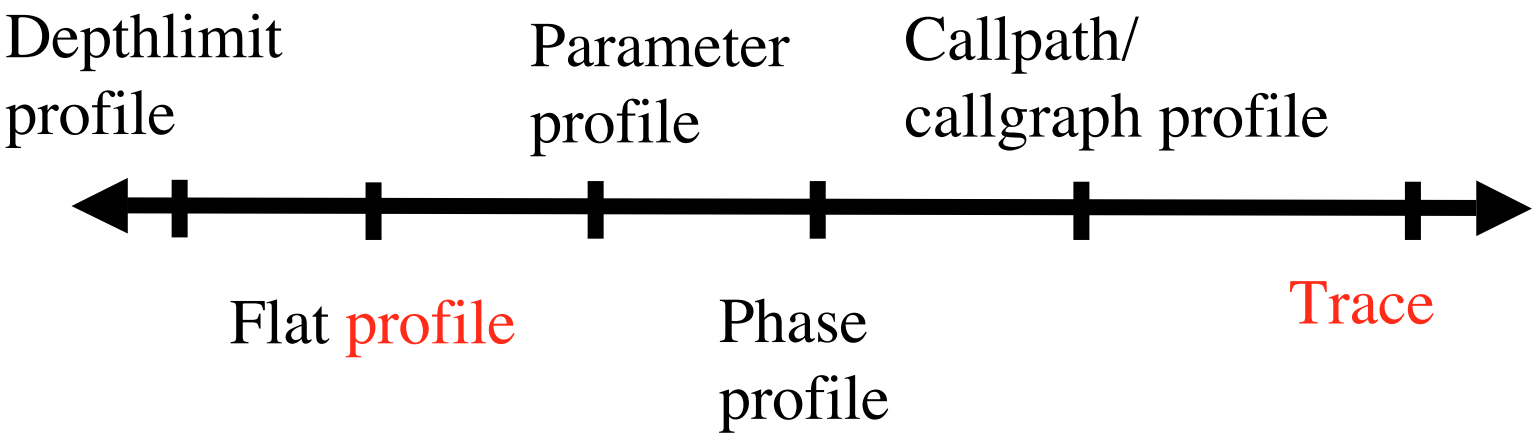
- Compensation of local instrumentation overhead

```
% configure -COMPENSATE
```

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



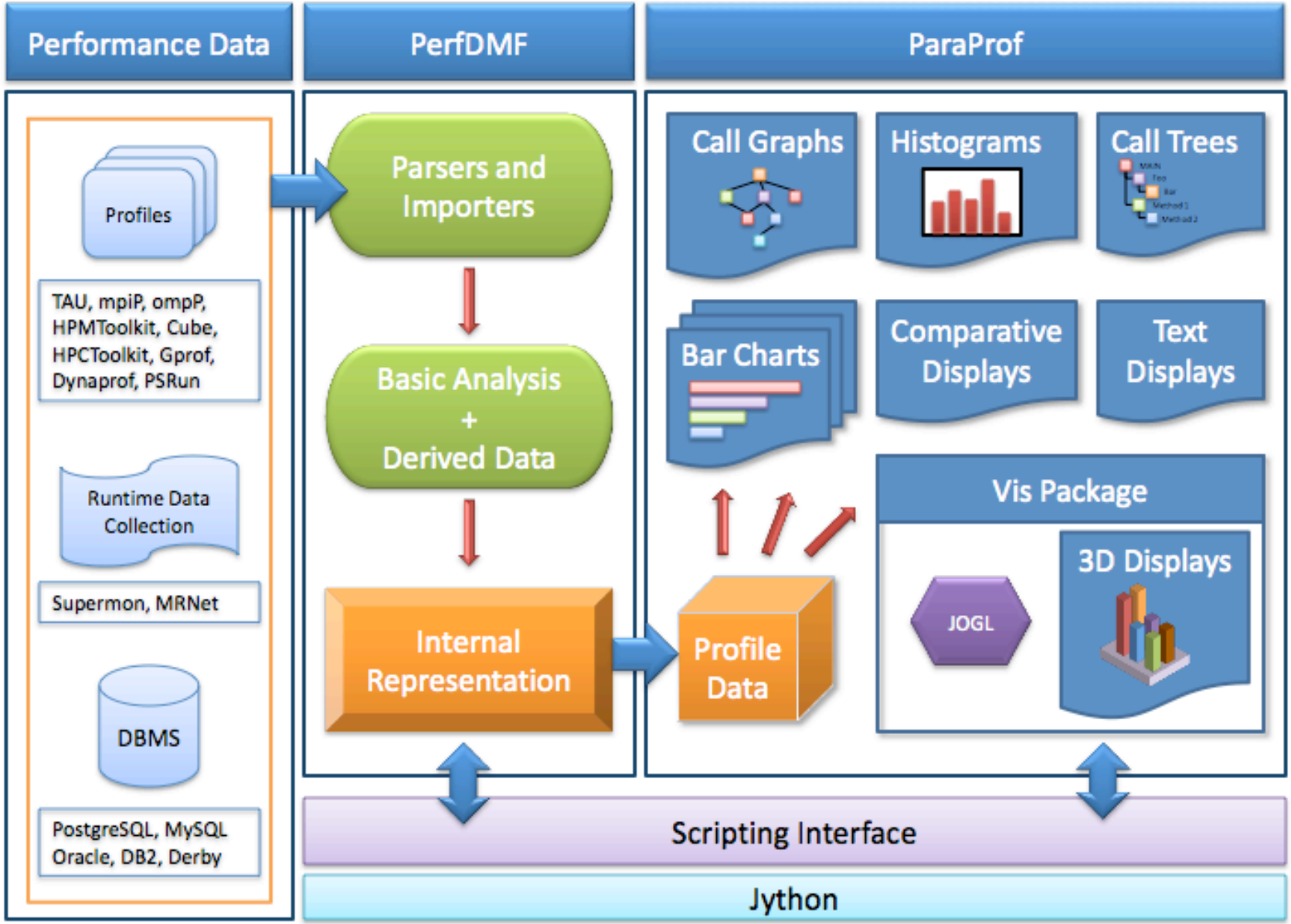
Each alternative has:
- one metric/counter
- multiple counters

Volume of performance data

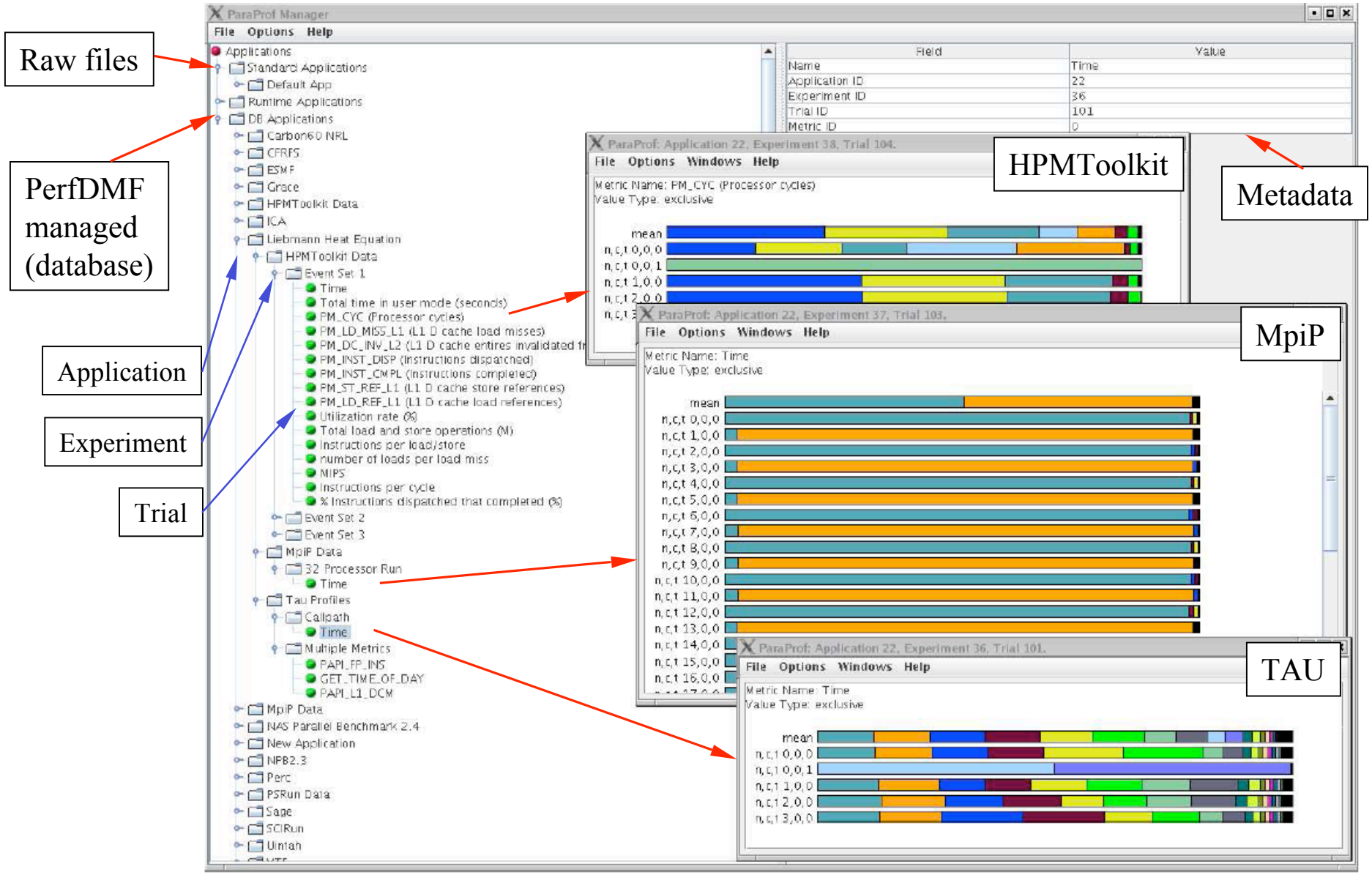
Performance Analysis and Visualization

- Analysis of parallel profile and trace measurement
- **Parallel profile analysis (*ParaProf*)**
  - Java-based analysis and visualization tool
  - Support for large-scale parallel profiles
- Performance data management framework (***PerfDMF***)
- **Parallel trace analysis**
  - Translation to *VTF* (V3.0), *EPILOG, OTF* formats
  - Integration with *Vampir / Vampir Server* (TU Dresden)
  - Profile generation from trace data
- Online parallel analysis and visualization
- Integration with *CUBE* browser (KOJAK, UTK, FZJ)

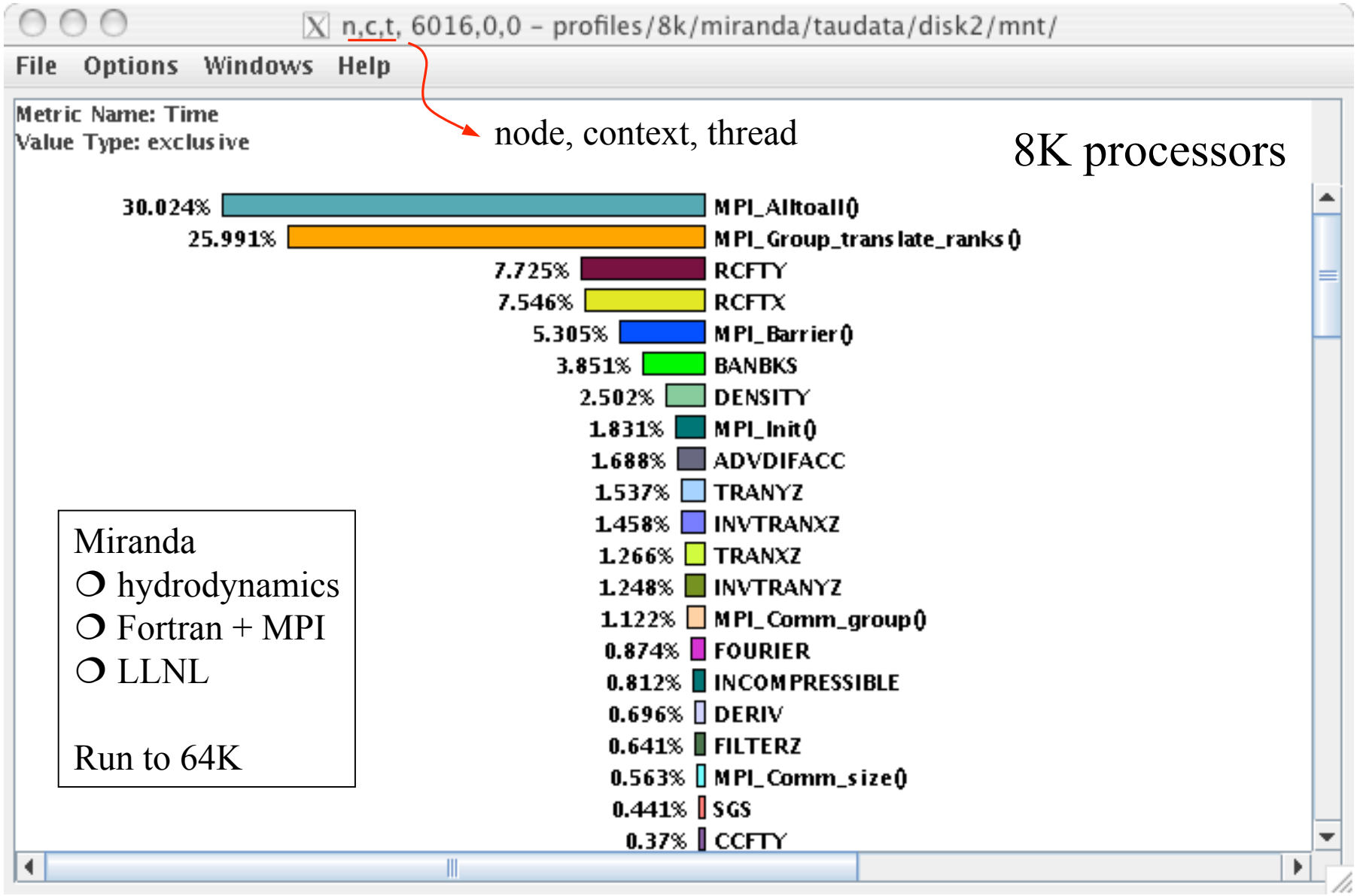
ParaProf – Parallel Performance Profile Analysis



ParaProf – Manager Window



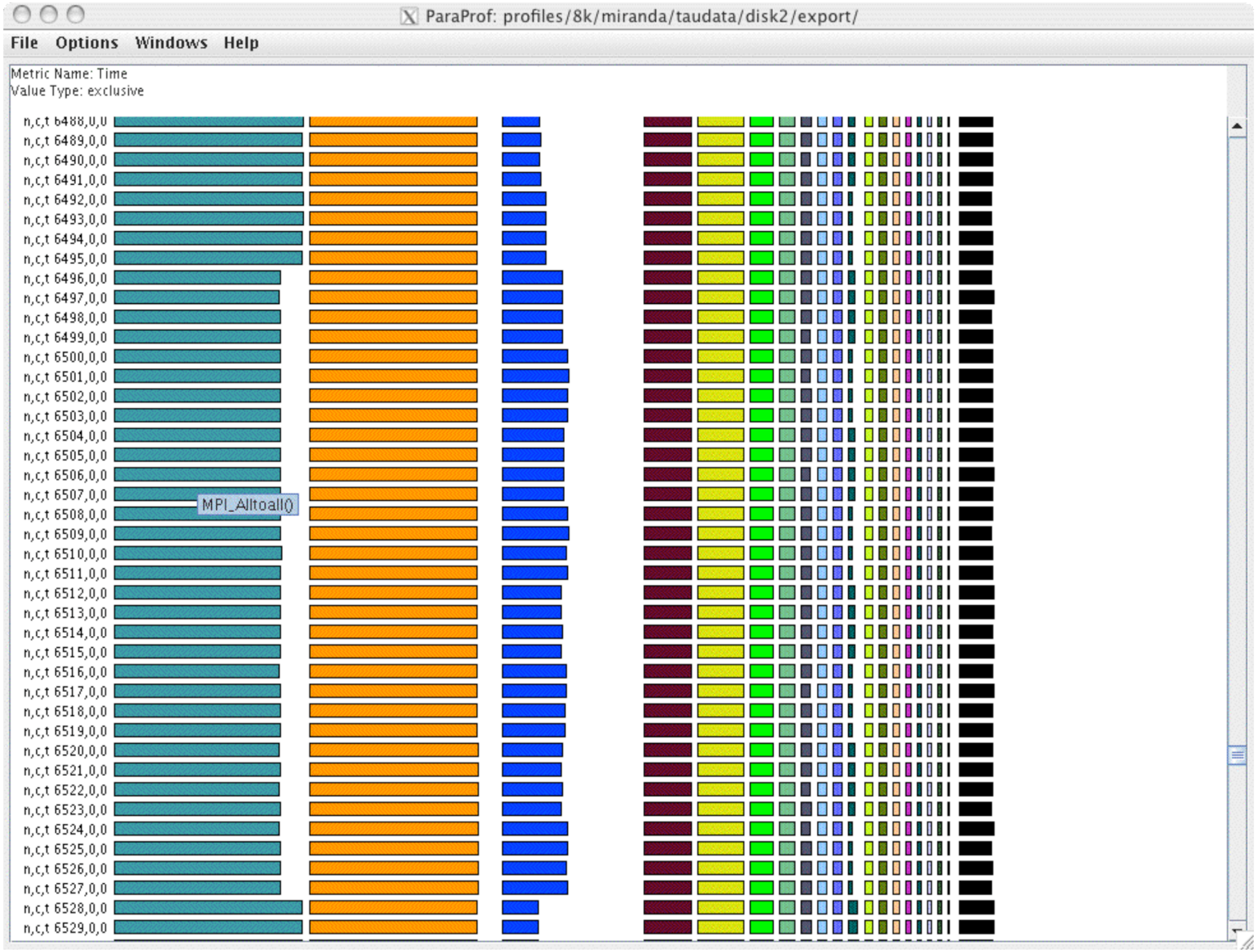
ParaProf – Flat Profile (Miranda, BG/L)



Miranda
- hydrodynamics
- Fortran + MPI
- LLNL

Run to 64K

ParaProf – Stacked View (Miranda)



ParaProf – Callpath Profile (Flash)

n,c,t, 0,0,0 - callpath-all/scaling/flash/taudata/disk2/mnt/

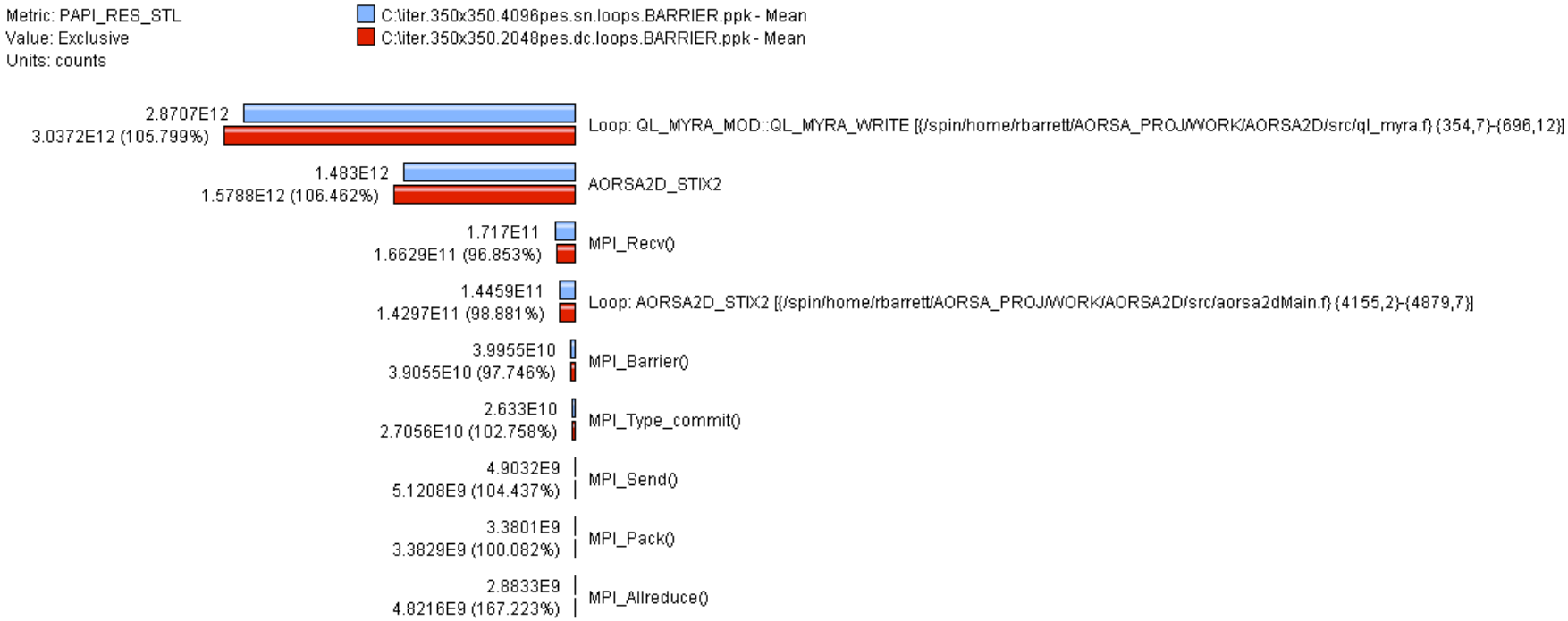
File Options Windows Help

Metric Name: Time
Value Type: exclusive

26.474%	MODULEHYDROSWEEP::HYDRO_SWEEP
26.474%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP
24.556%	MODULEHYDRO_1D::HYDRO_1D
24.556%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MODULEHYDRO_1D::HYDRO_1D
14.351%	MODULEINTRFC::INTRFC
14.351%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MODULEHYDRO_1D::HYDRO_1D => MODULEINTRFC::INT
4.501%	MODULEEOS3D::EOS3D
4.427%	MPI_Ssend()
3.678%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MODULEEOS3D::EOS3D
3.536%	MPI_Allreduce()
2.727%	MPI_Waitall()
2.242%	MODULEUPDATE_SOLN::UPDATE_SOLN
2.242%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MODULEUPDATE_SOLN::UPDATE_SOLN
2.059%	AMR_GUARDCELL_CC_SRL
1.703%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_GUARDCELL => AMR_GUARDCELL_SRL => AMR_
1.56%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_GUARDCELL => AMR_GUARDCELL_SRL => AMR_
1.406%	FLASH => EVOLVE => MESH_UPDATE_GRID_REFINEMENT => MESH_REFINE_DEREFINE => AMR_REFINE_DEREFINE => AMR_MORTON_ORDER =>
1.361%	FLASH => TIMESTEP => MPI_Allreduce()
1.319%	AMR_RESTRICT_UNK_FUN
1.272%	AMR_PROLONG_GEN_UNK_FUN
1.093%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_GUARDCELL => AMR_GUARDCELL_C_TO_F => A
1.077%	ABUNDANCE_RESTRICT
1.077%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => ABUNDANCE_RESTRICT
1.064%	DBASETREE::DBASENEIGHBORBLOCKLIST
1%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_GUARDCELL => AMR_RESTRICT => AMR_RESTRI
0.987%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_FLUX_CONSERVE => AMR_FLUX_CONSERVE_UDT
0.96%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_GUARDCELL => AMR_GUARDCELL_C_TO_F => A
0.916%	MPI_Barrier()
0.807%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_GUARDCELL => TOT_BND => DBASETREE::DBAS
0.806%	AMR_PROLONG_UNK_FUN
0.735%	AMR_DIAGONAL_PATCH
0.699%	DIFFUSE
0.699%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => DIFFUSE
0.671%	AMR_RESTRICT_RED
0.671%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_FLUX_CONSERVE => AMR_FLUX_CONSERVE_UDT
0.657%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_GUARDCELL => AMR_GUARDCELL_SRL => AMR_
0.638%	FLASH => EVOLVE => MESH_UPDATE_GRID_REFINEMENT => MARK_GRID_REFINEMENT => MPI_Barrier()
0.61%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_GUARDCELL => AMR_GUARDCELL_C_TO_F => A
0.556%	FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_GUARDCELL => AMR_GUARDCELL_C_TO_F => A
0.508%	TOT_BND
0.454%	FLASH => EVOLVE => MESH_UPDATE_GRID_REFINEMENT => MARK_GRID_REFINEMENT => MODULEEOS3D::EOS3D

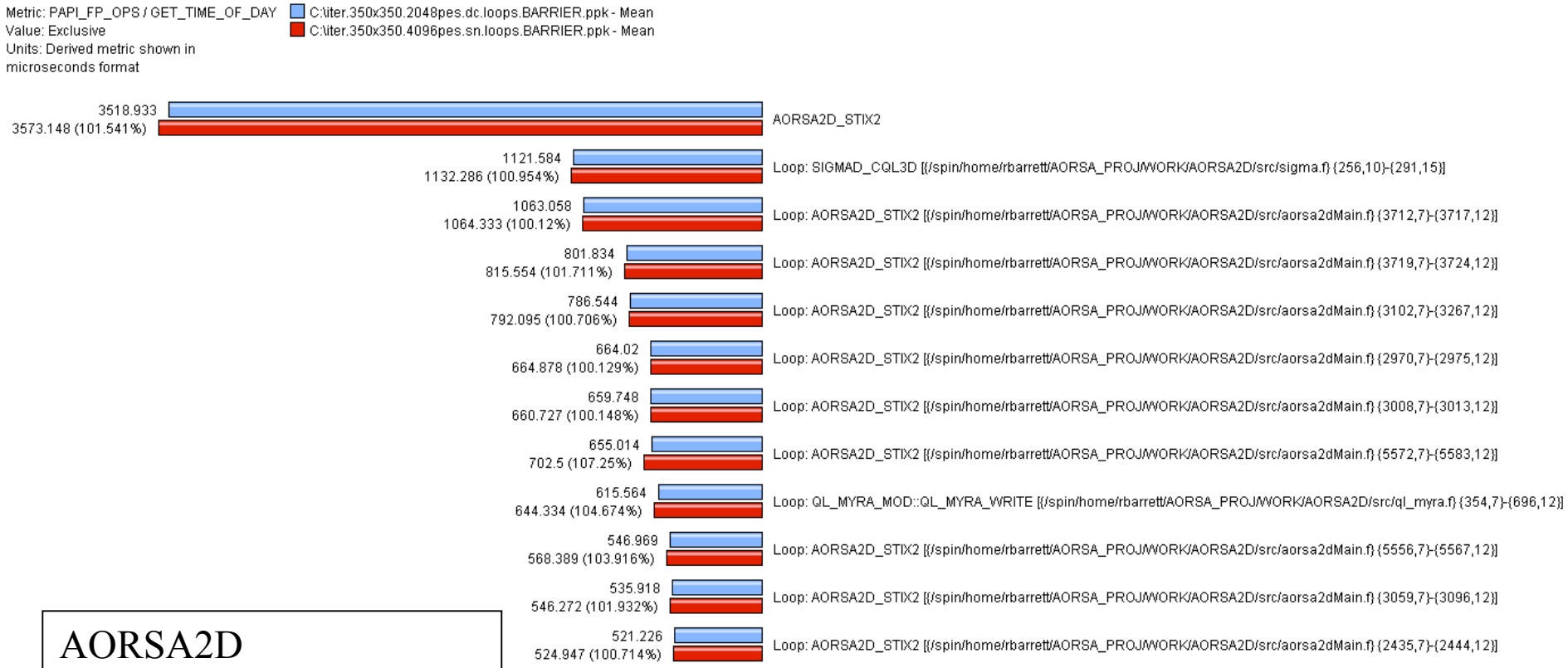
Flash
- thermonuclear flashes
- Fortran + MPI
- Argonne

Comparing Effects of Multi-Core Processors



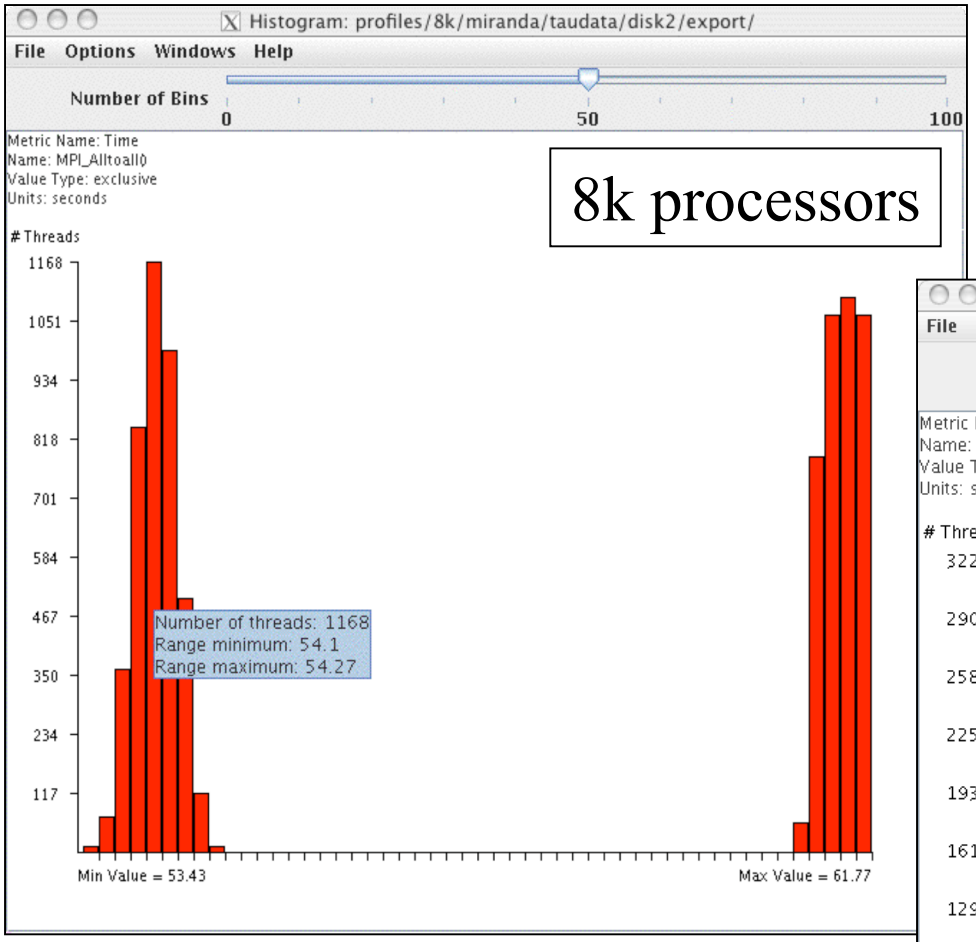
AORSA2D
- magnetized plasma simulation
- Blue is single node
- Red is dual core
- Cray XT3 (4K cores)

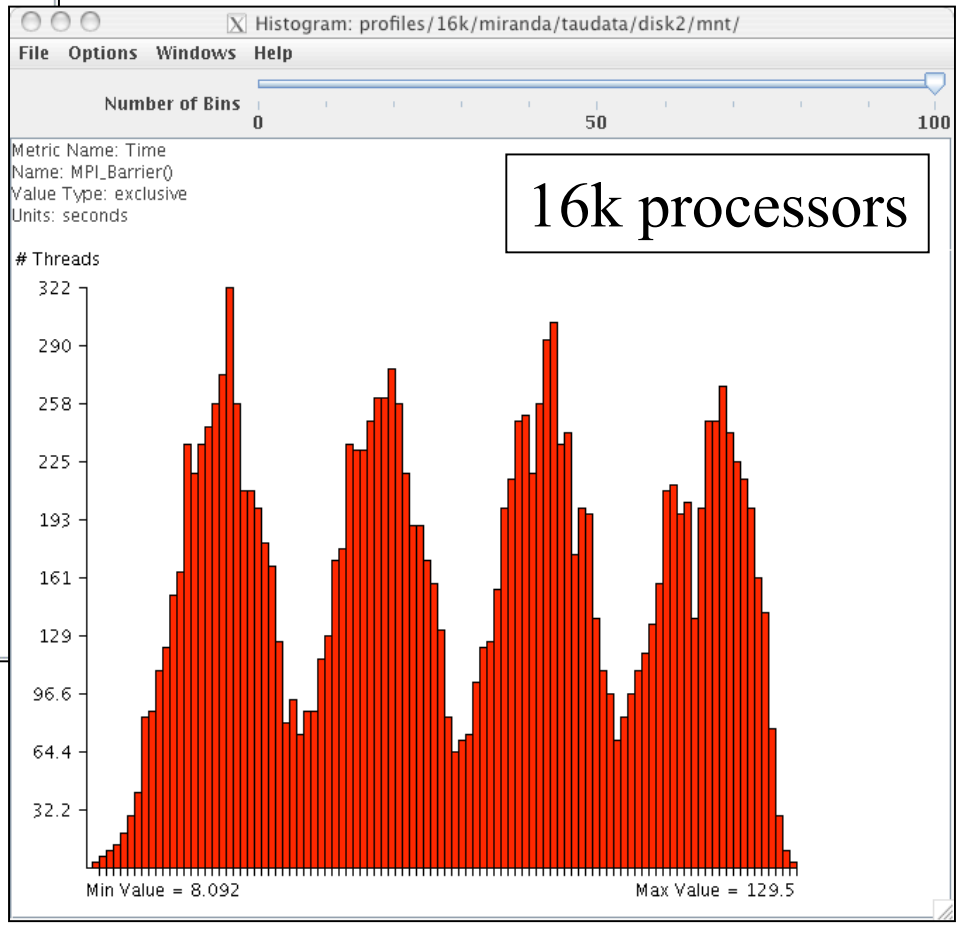
Comparing FLOPS (AORSA2D, Cray XT3)



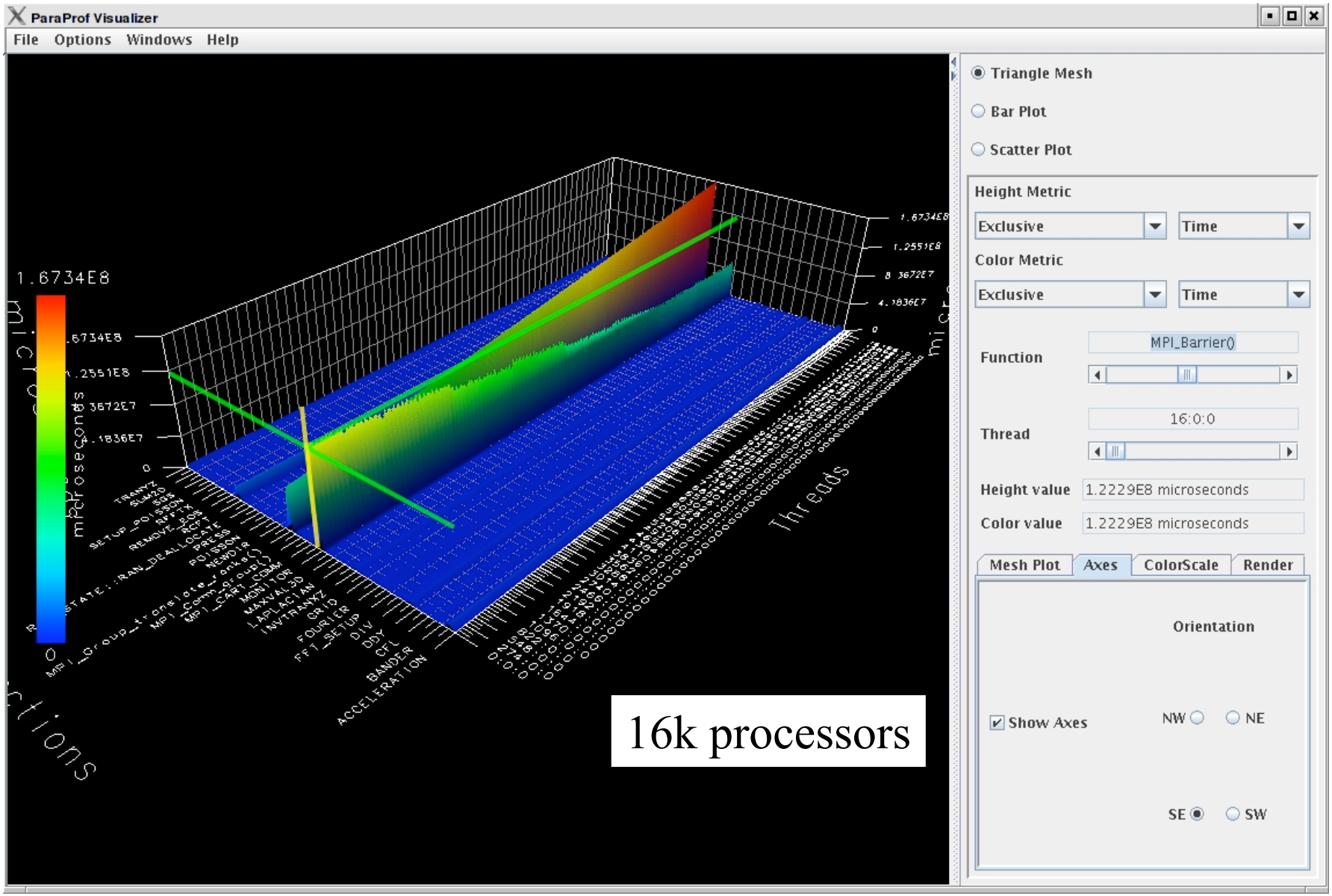
AORSA2D
- Blue is dual core
- Red is single node
- Cray XT3 (4K cores)
- Data generated by Richard Barrett, ORNL

ParaProf – Scalable Histogram View (Miranda)

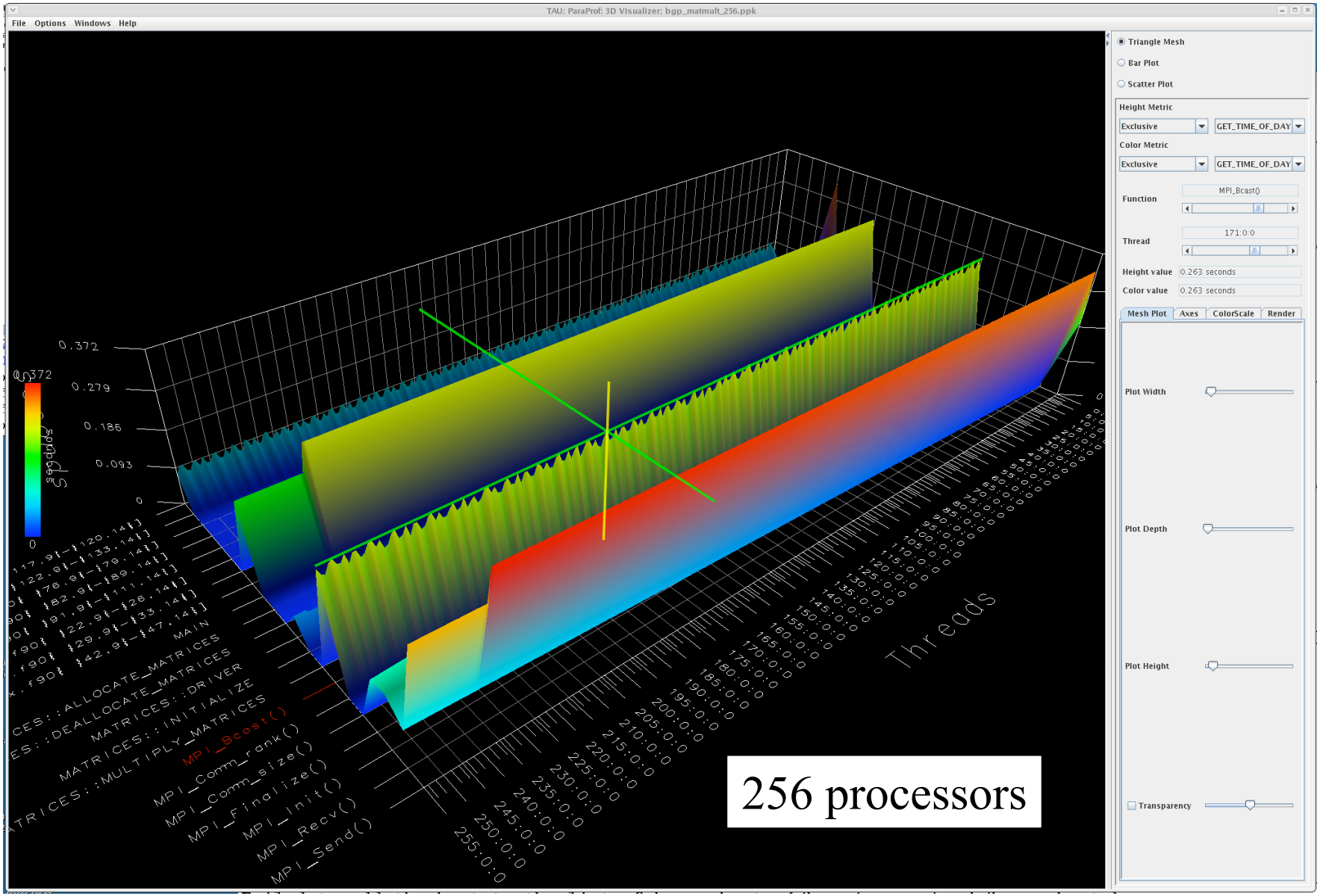




ParaProf – Full Profile (Miranda)

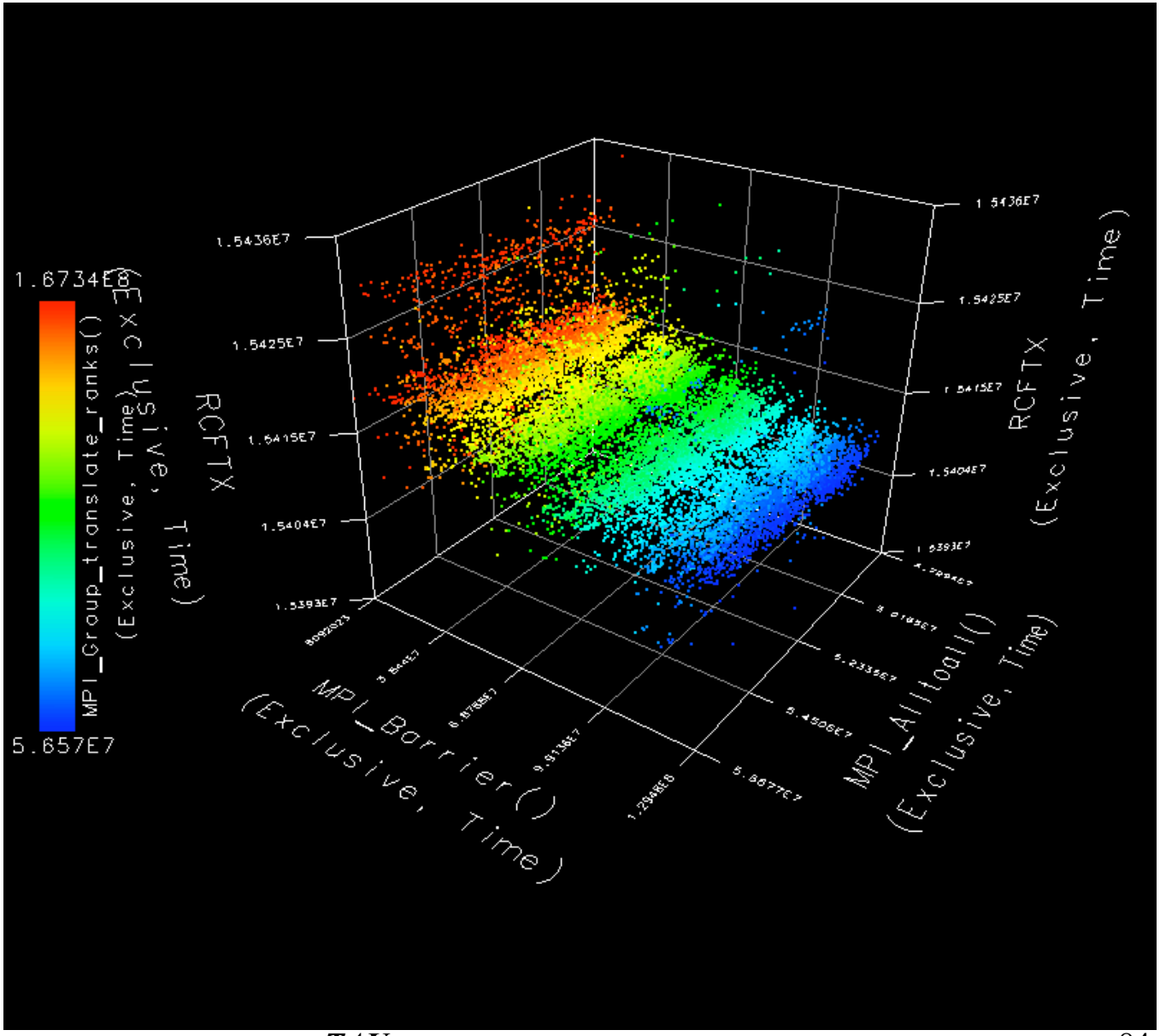


ParaProf –Full Profile (Matmult, ANL BGP)



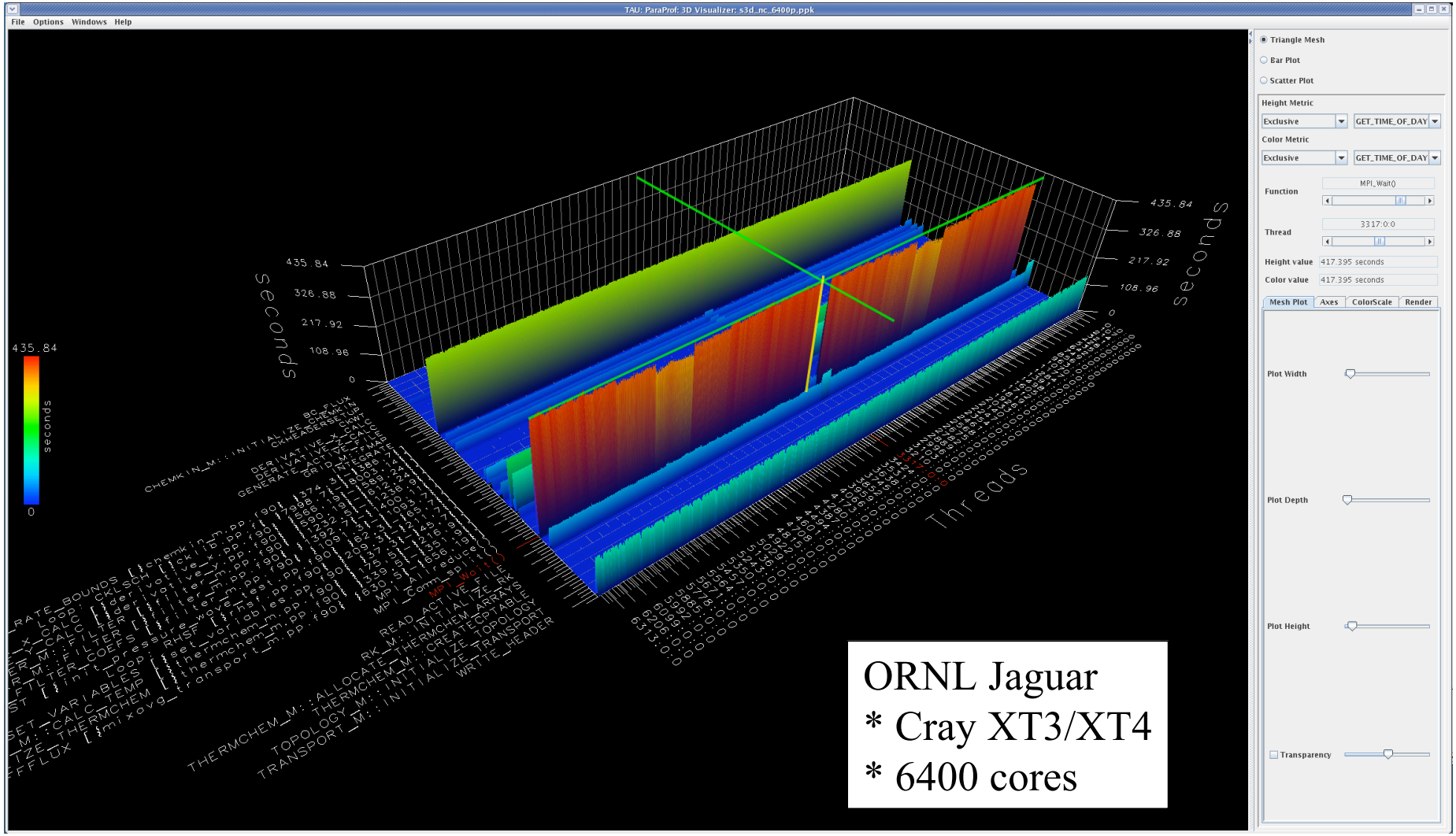
ParaProf – 3D Scatterplot (Miranda)

- Each point is a "thread" of execution
- A total of four metrics shown in relation
- ParaProf's visualization library
  - JOGL

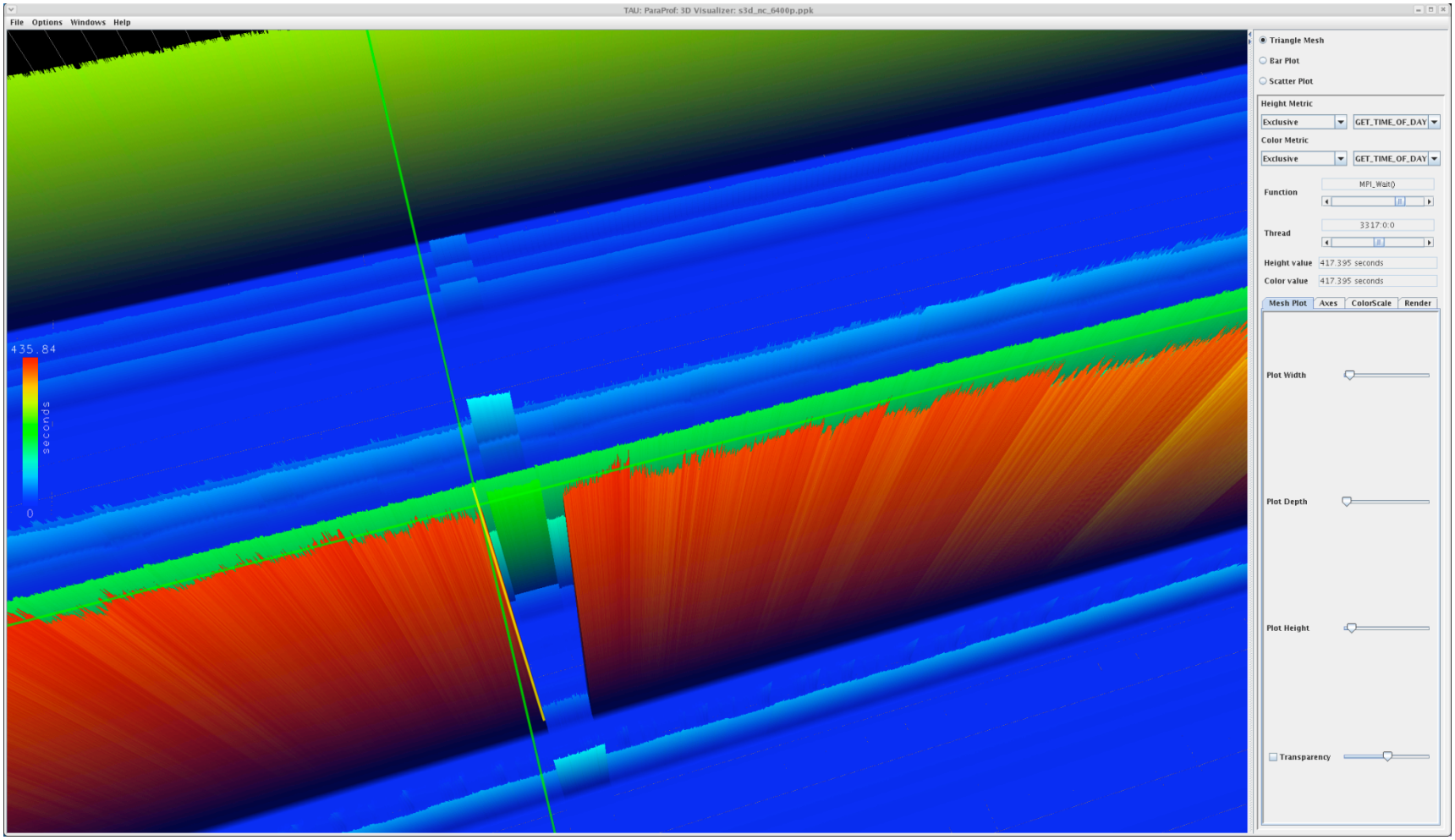


Visualizing Hybrid Problems (S3D, XT3+XT4)

- S3D combustion simulation (DOE SciDAC PERI)



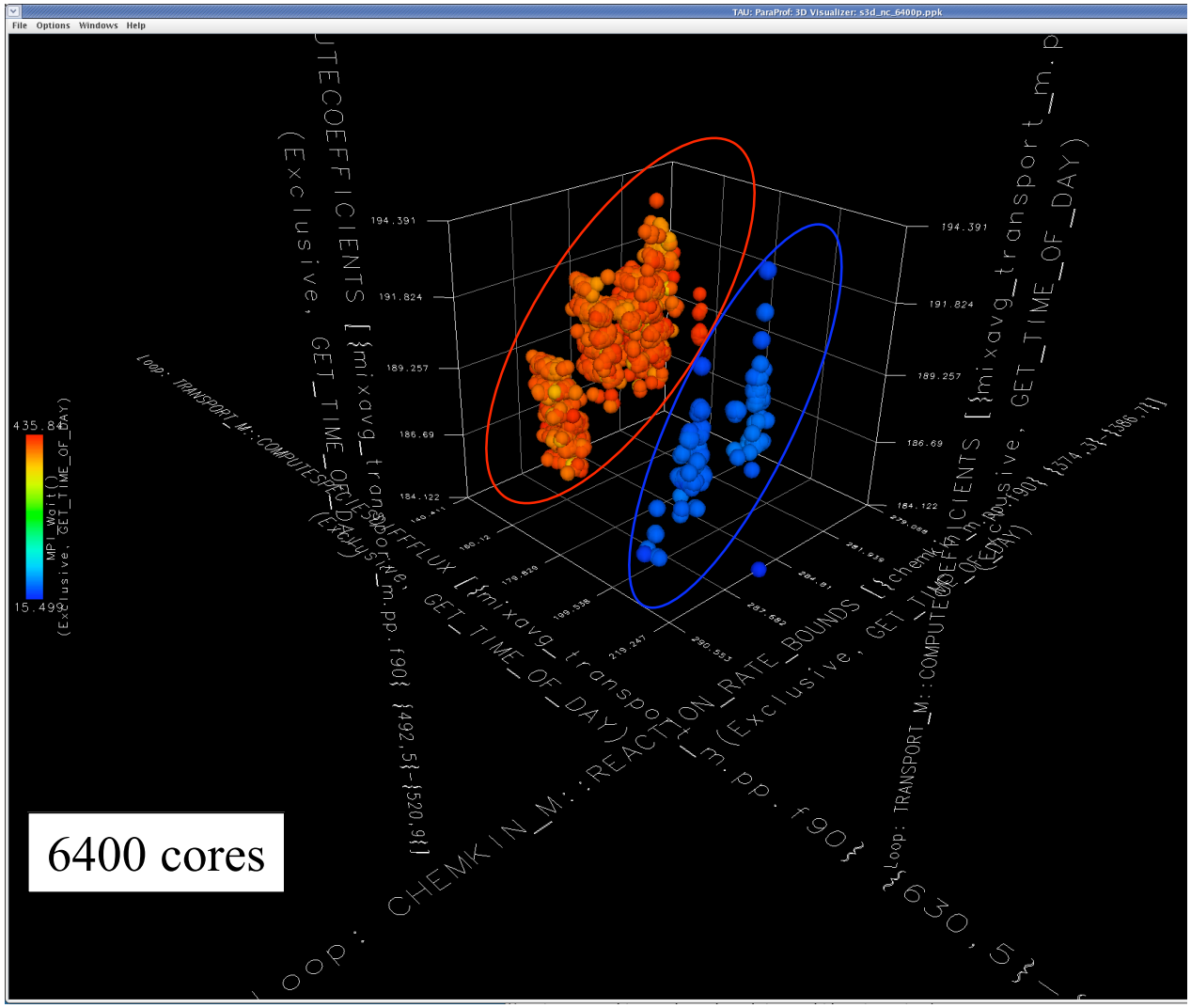
Zoom View of Hybrid Execution (S3D, XT3+XT4)



- Gap represents XT3 nodes
  - MPI_Wait takes less time, other routines take more time

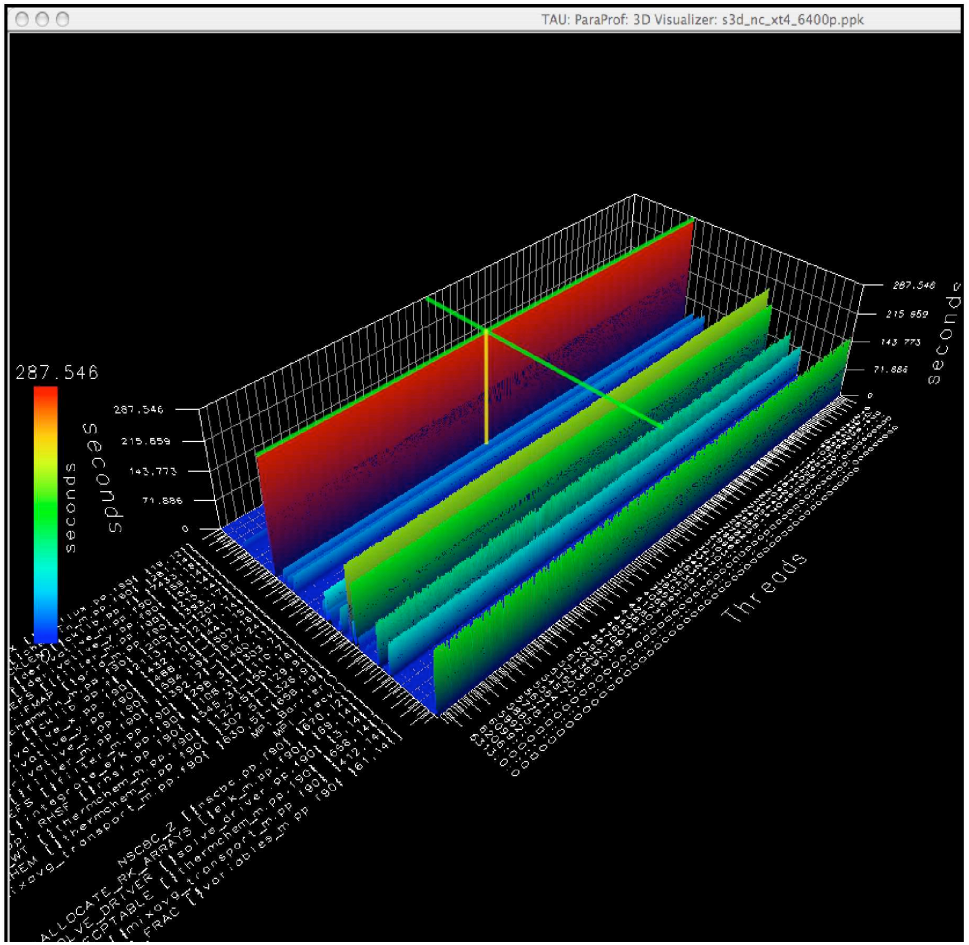
Visualizing Hybrid Execution (S3D, XT3+XT4)

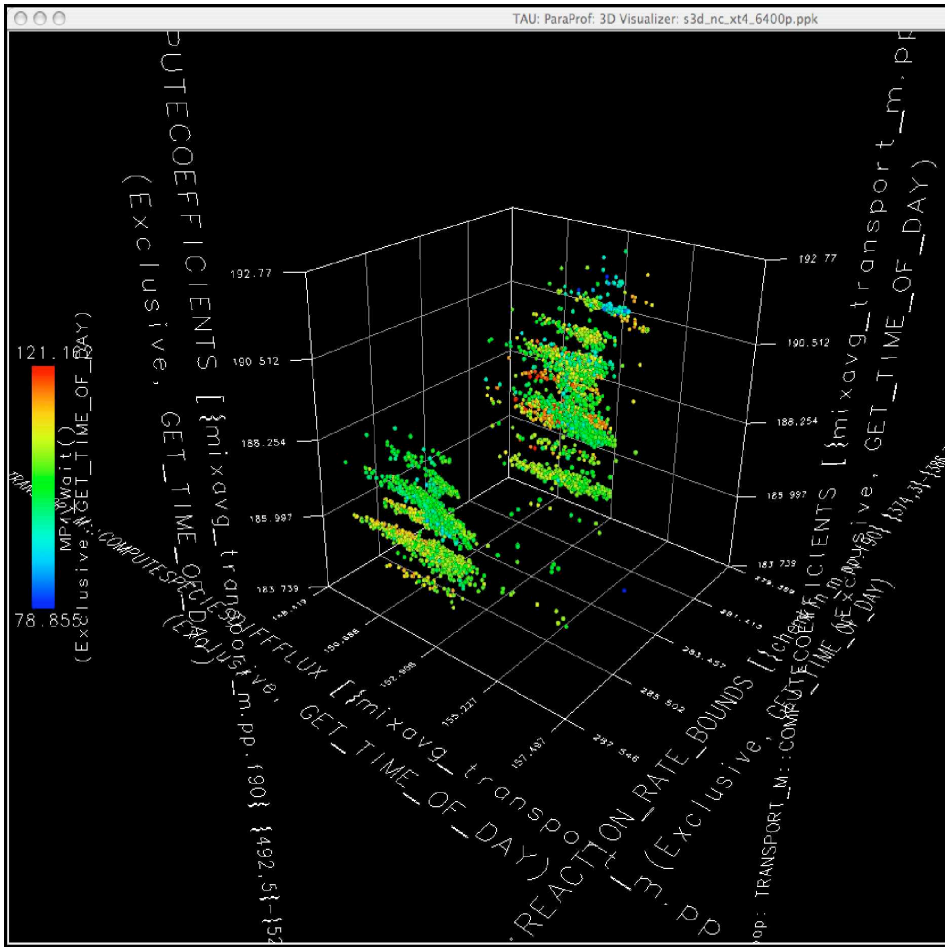
- Hybrid execution
- Process metadata is used to map performance to machine type
- Memory speed accounts for performance difference



S3D Run on XT4 Only

- Better balance across nodes
- More performance uniformity





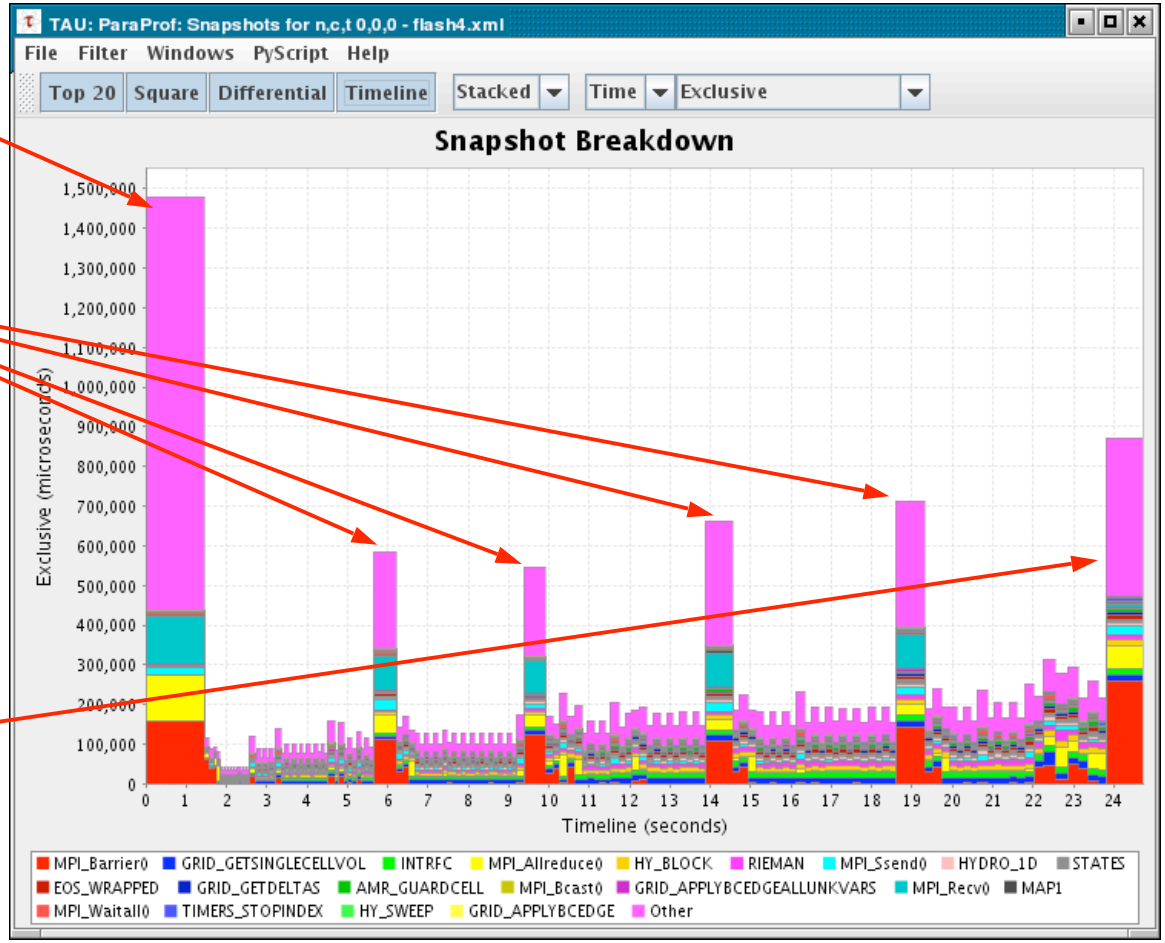
ParaProf – Profile Snapshots (Flash)

- Profile snapshots are parallel profiles recorded at runtime
- Used to highlight profile changes during execution

Initialization

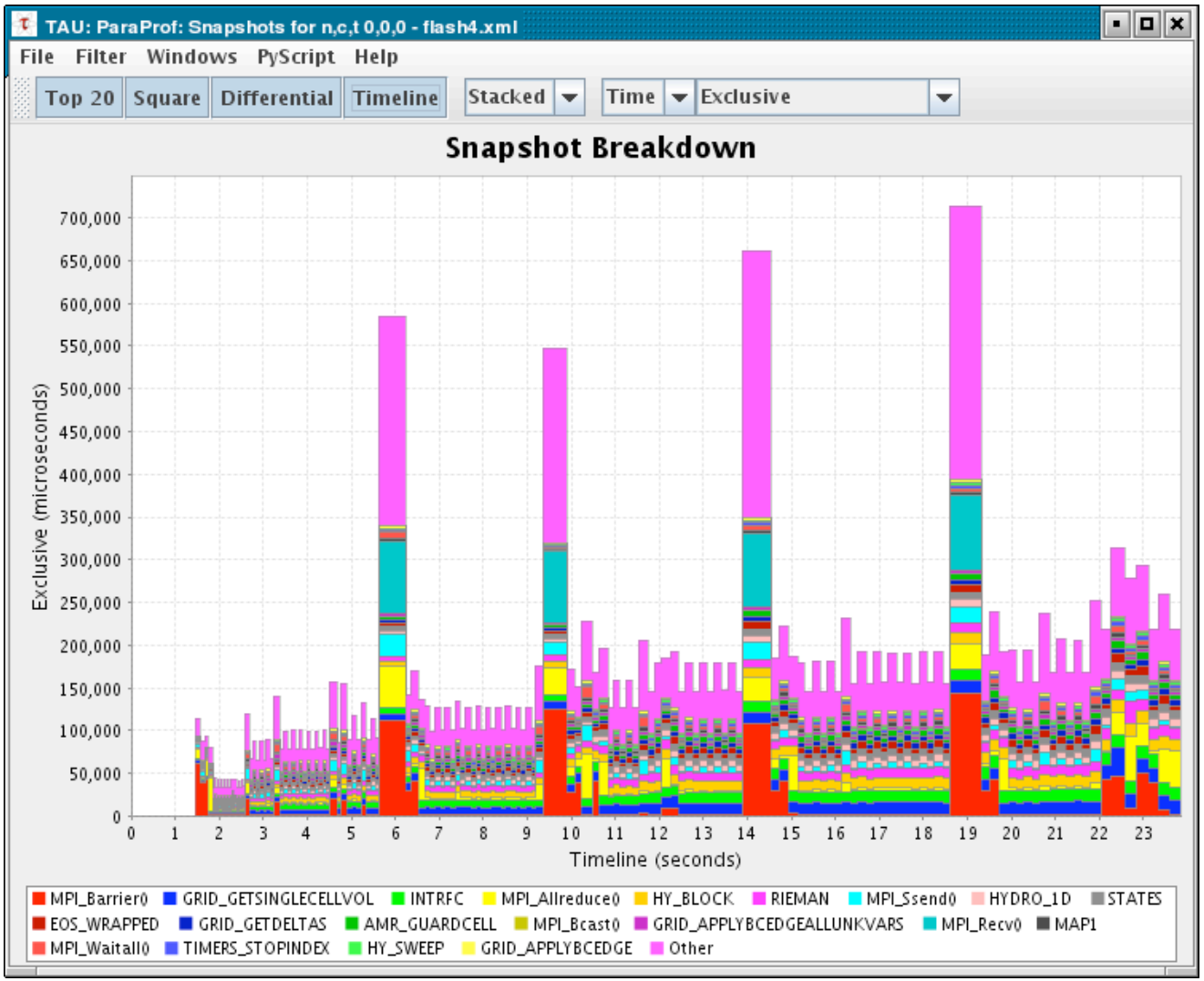
Checkpointing

Finalization



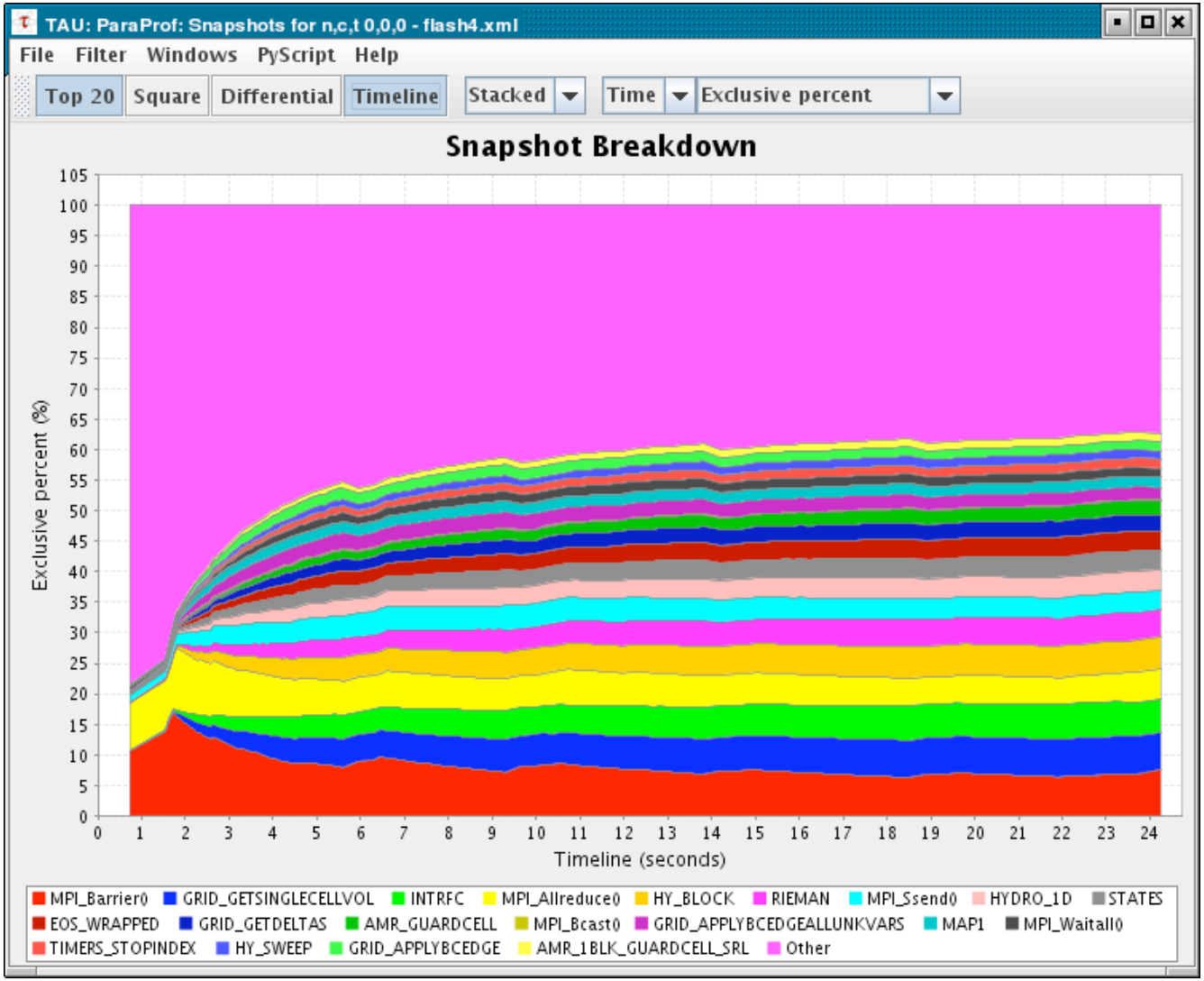
Filtered Profile Snapshots (Flash)

- Only show main loop iterations

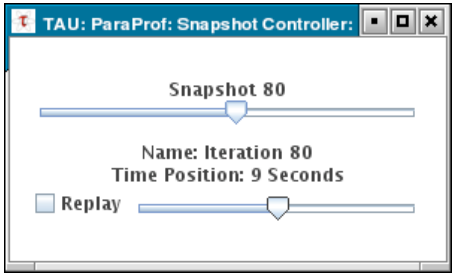


Profile Snapshots with Breakdown (Flash)

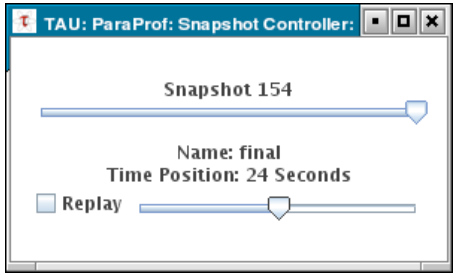
- Breakdown as a percentage

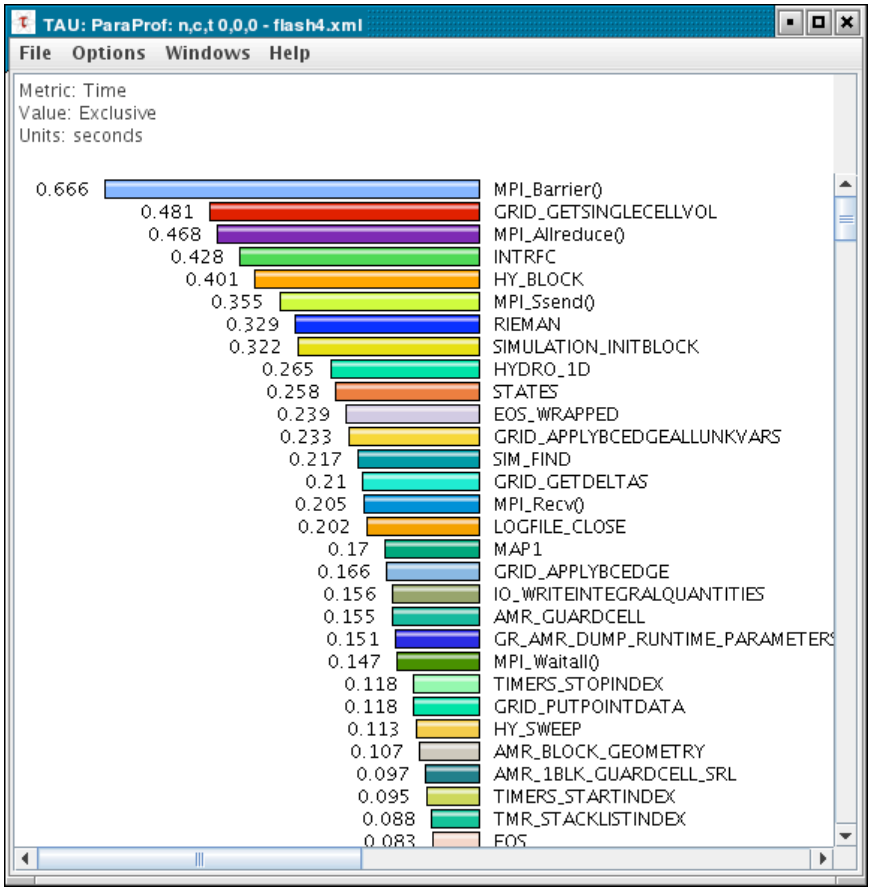


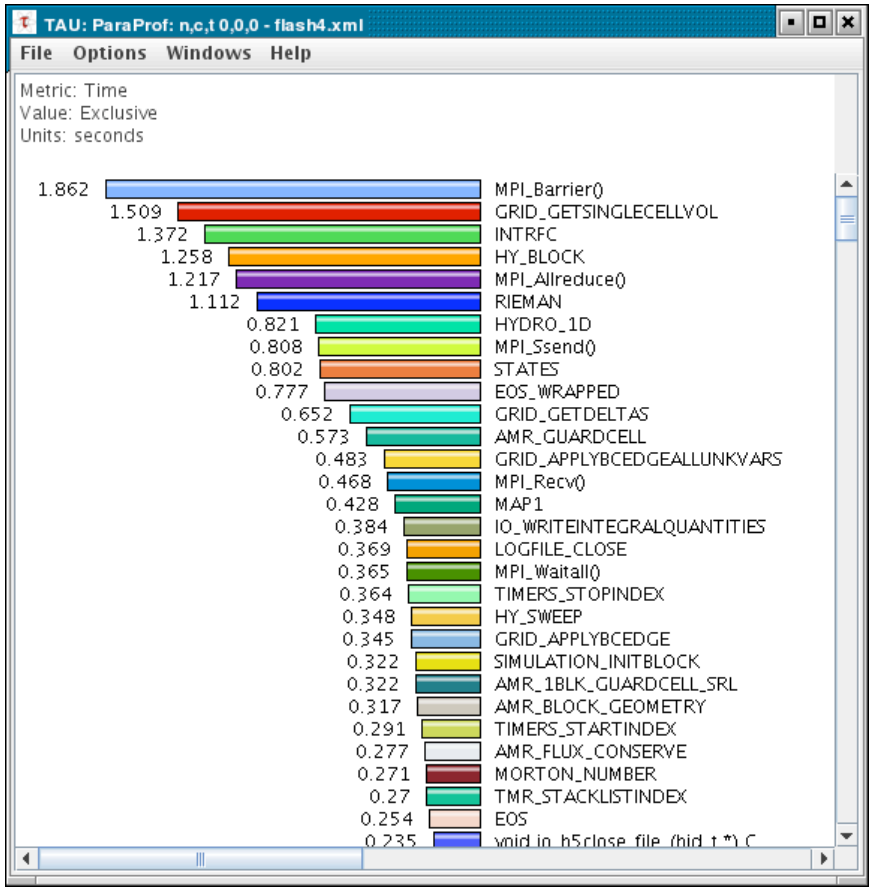
Profile Snapshot Replay (Flash)



All windows dynamically update

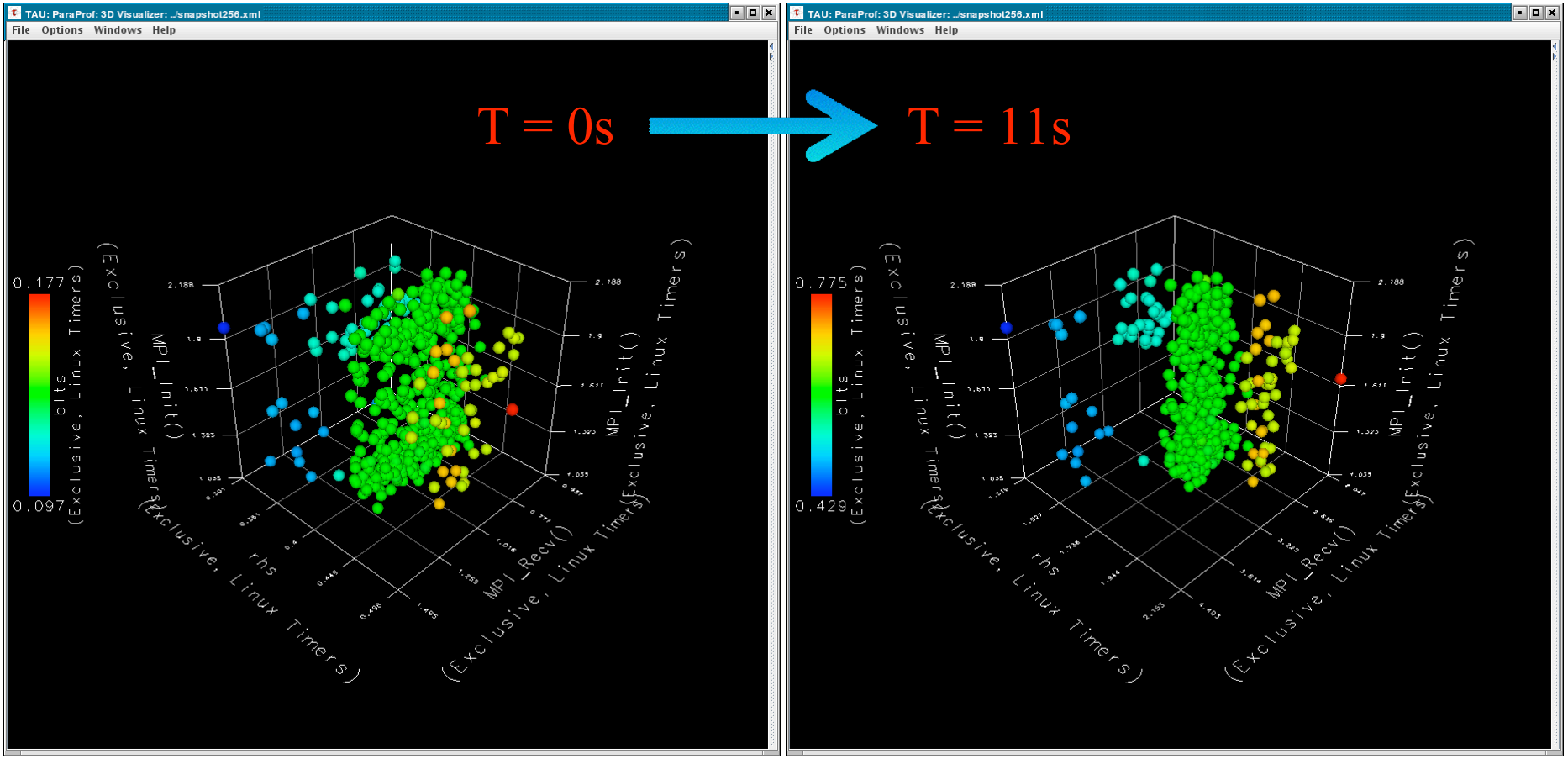






Snapshot Dynamics of Event Relations (Flash)

- Follow progression of various displays through time
- 3D scatter plot shown below



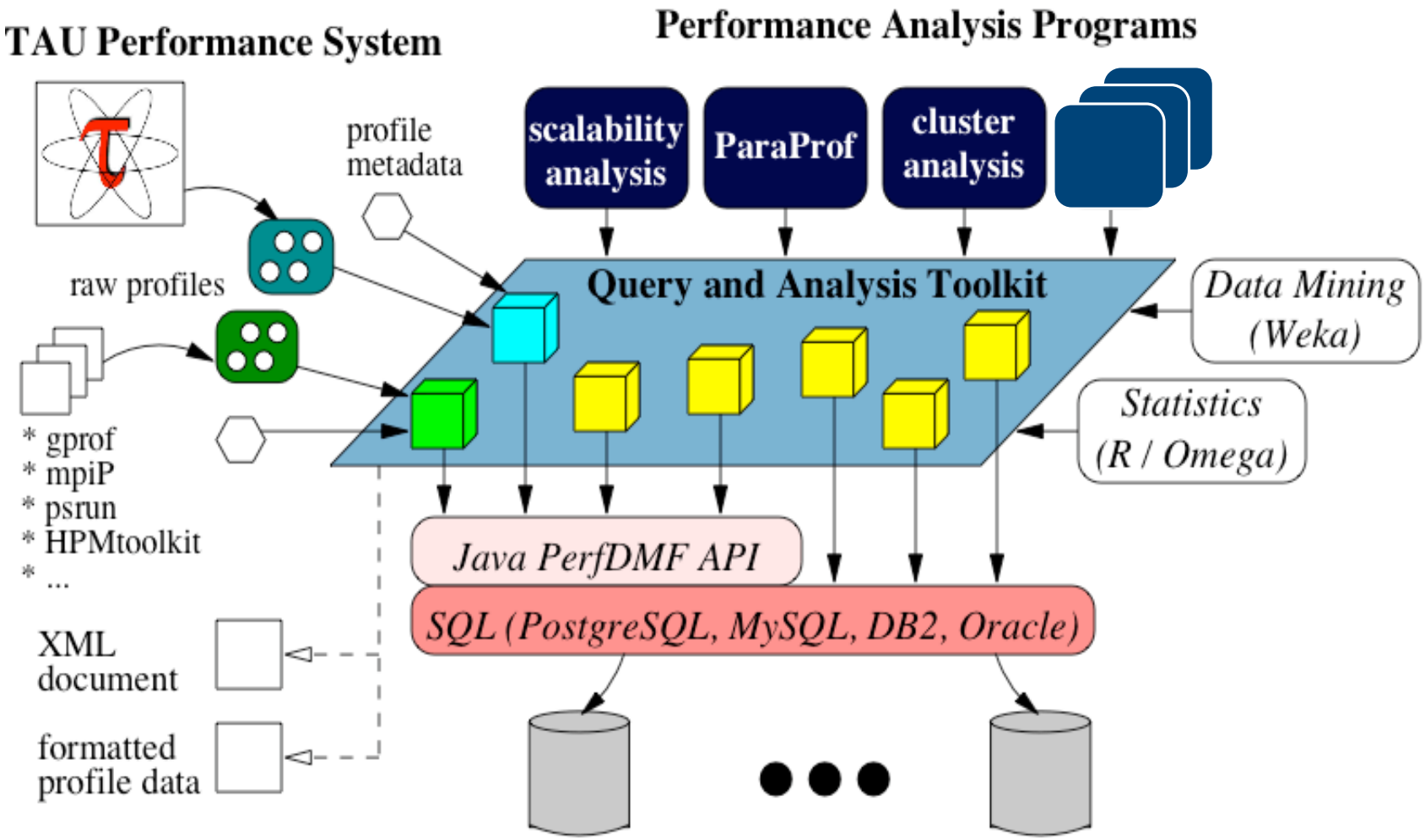
Performance Data Management

- Need for robust processing and storage of multiple profile performance data sets
- Avoid developing independent data management solutions
  - Waste of resources
  - Incompatibility among analysis tools
- Goals
  - Foster multi-experiment performance evaluation
  - Develop a common, reusable foundation of performance data storage, access and sharing
  - A core module in an analysis system, and/or as a central repository of performance data

PerfDMF Approach

- Performance Data Management Framework
- Originally designed to address critical TAU requirements
- Broader goal is to provide an open, flexible framework to support common data management tasks
- Extensible toolkit to promote integration and reuse across available performance tools
  - Supported profile formats: TAU, CUBE 2 & 3 (Kojak), Dynaprof, HPC Toolkit (Rice), HPM Toolkit (IBM), gprof, mpiP, psrun (PerfSuite), Open|SpeedShop, …
  - Supported DBMS: PostgreSQL, MySQL, Oracle, DB2, Derby/Cloudscape
  - Profile query and analysis API

PerfDMF Architecture

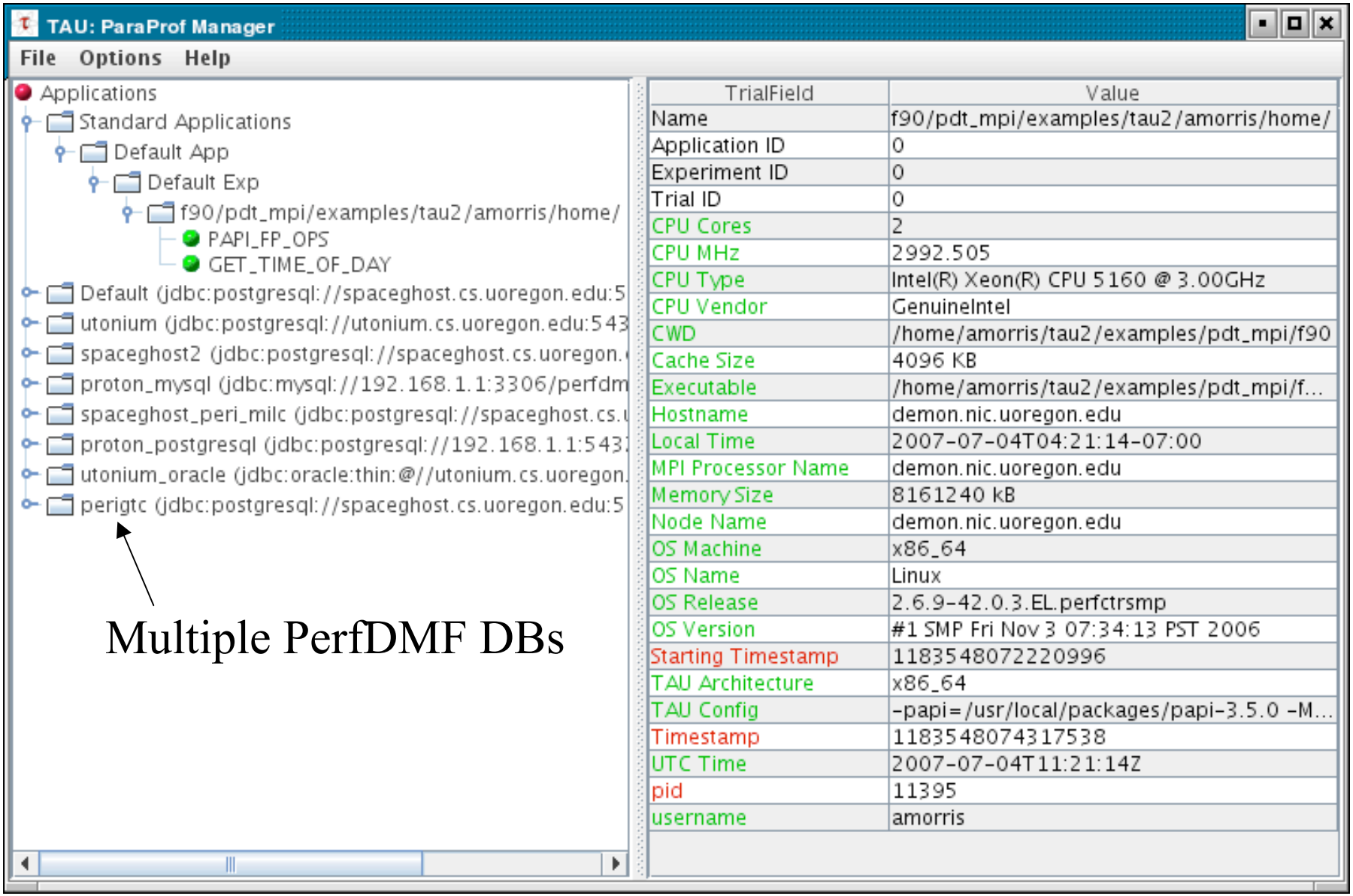


K. Huck, A. Malony, R. Bell, A. Morris, "**Design and Implementation of a Parallel Performance Data Management Framework**," ICPP 2005.

Metadata Collection

- Integration of XML metadata for each profile
- Three ways to incorporate metadata
  - Measured hardware/system information (TAU, PERI-DB)
    - CPU speed, memory in GB, MPI node IDs, …
  - Application instrumentation (application-specific)
    - TAU_METADATA() used to insert any name/value pair
    - Application parameters, input data, domain decomposition
  - PerfDMF data management tools can incorporate an XML file of additional metadata
    - Compiler flags, submission scripts, input files, …
- Metadata can be imported from / exported to PERI-DB
  - PERI SciDAC project (UTK, NERSC, UO, PSU, TAMU)

Metadata for Each Experiment



TrialField	Value
Name	f90/pdt_mpi/examples/tau2/amorris/home/
Application ID	0
Experiment ID	0
Trial ID	0
CPU Cores	2
CPU MHz	2992.505
CPU Type	Intel(R) Xeon(R) CPU 5160 @ 3.00GHz
CPU Vendor	GenuineIntel
CWD	/home/amorris/tau2/examples/pdt_mpi/f90
Cache Size	4096 KB
Executable	/home/amorris/tau2/examples/pdt_mpi/f...
Hostname	demon.nic.uoregon.edu
Local Time	2007-07-04T04:21:14-07:00
MPI Processor Name	demon.nic.uoregon.edu
Memory Size	8161240 kB
Node Name	demon.nic.uoregon.edu
OS Machine	x86_64
OS Name	Linux
OS Release	2.6.9-42.0.3.EL.perfctrsmp
OS Version	#1 SMP Fri Nov 3 07:34:13 PST 2006
Starting Timestamp	1183548072220996
TAU Architecture	x86_64
TAU Config	-papi=/usr/local/packages/papi-3.5.0 -M...
Timestamp	1183548074317538
UTC Time	2007-07-04T11:21:14Z
pid	11395
username	amorris

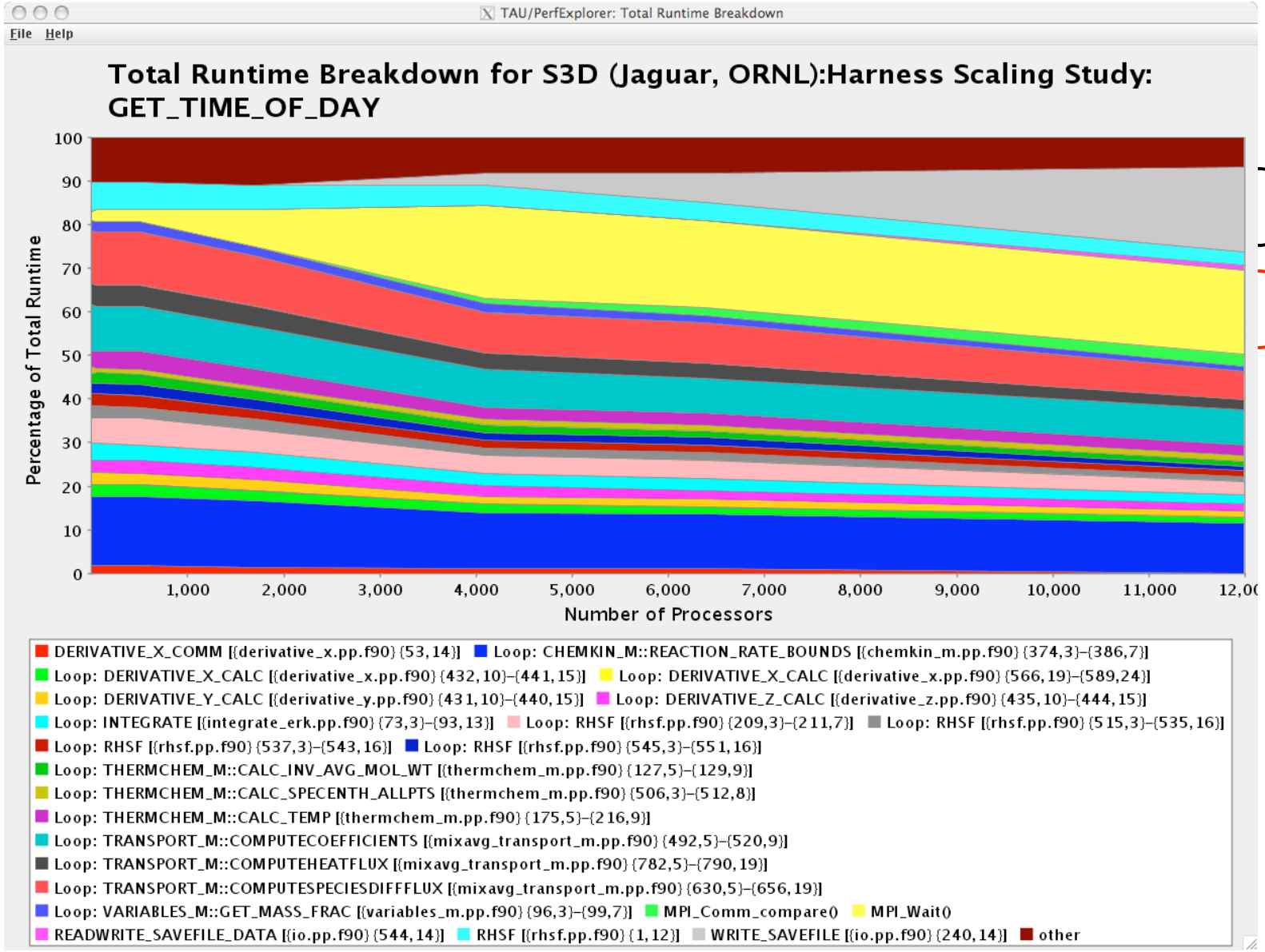
Performance Data Mining

- Conduct parallel performance analysis process
  - In a systematic, collaborative and reusable manner
  - Manage performance complexity
  - Discover performance relationship and properties
  - Automate process
- Multi-experiment performance analysis
- Large-scale performance data reduction
  - Summarize characteristics of large processor runs
- Implement extensible analysis framework
  - Abstraction / automation of data mining operations
  - Interface to existing analysis and data mining tools

Performance Data Mining (PerfExplorer)

- Performance knowledge discovery framework
  - Data mining analysis applied to parallel performance data
    - comparative, clustering, correlation, dimension reduction, …
  - Use the existing TAU infrastructure
    - TAU performance profiles, PerfDMF
- Technology integration
  - Java API and toolkit for portability
  - Built on top of PerfDMF
  - R-project/Omegahat, Octave/Matlab statistical analysis
  - WEKA data mining package
  - JFreeChart for visualization, vector output (EPS, SVG)

PerfExplorer: S3D Total Runtime Breakdown



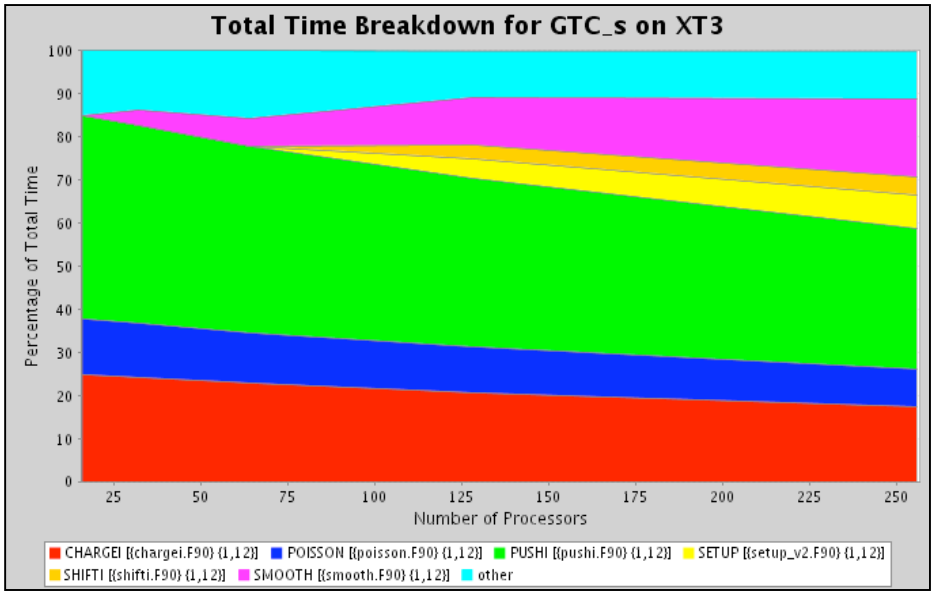
WRITE_ SAVEFILE

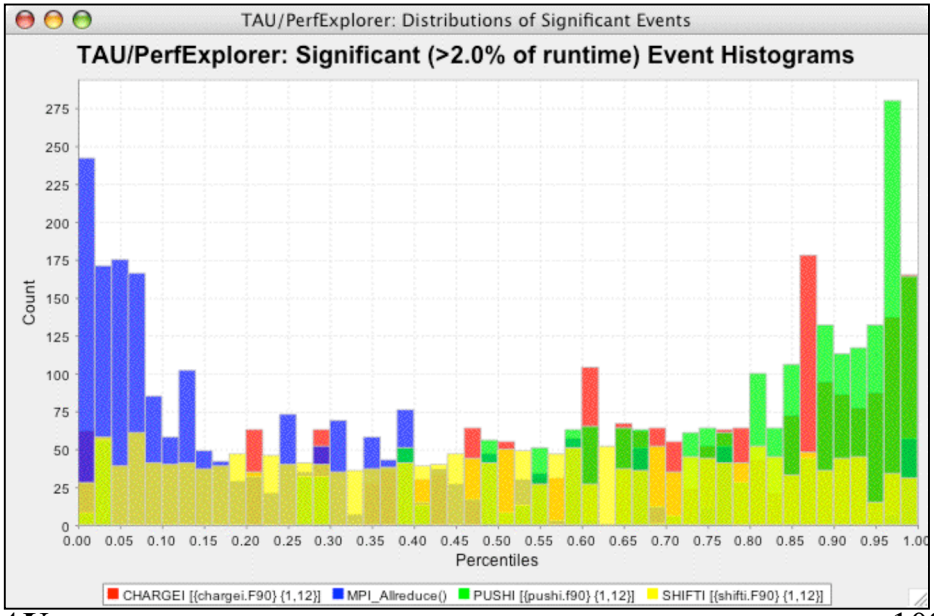
MPI_Wait

12,000 cores!

Relative Comparisons (GTC, XT3, DOE PERI)

- Total execution time
- Timesteps per second
- Relative efficiency
- Relative efficiency per event
- Relative speedup
- Relative speedup per event
- Group fraction of total
- Runtime breakdown
- Correlate events with total runtime
- Relative efficiency per phase
- Relative speedup per phase
- Distribution visualizations

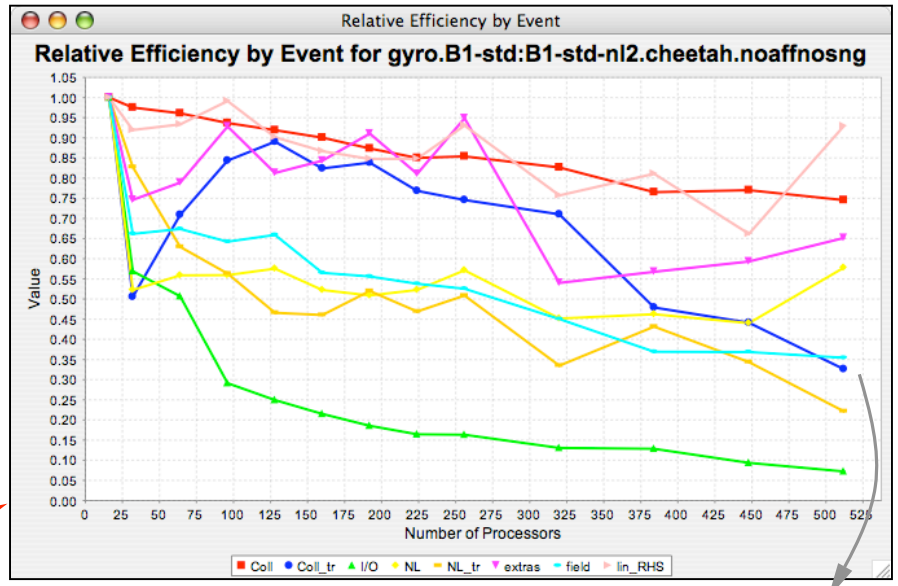


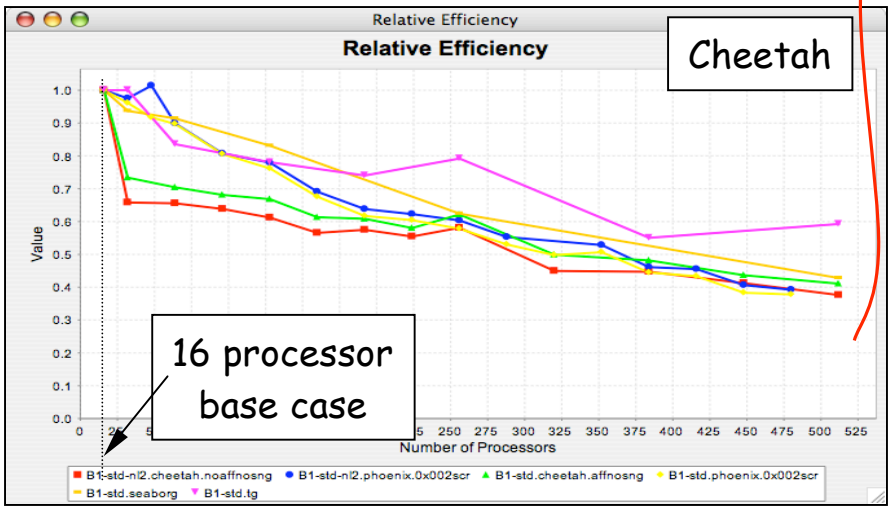


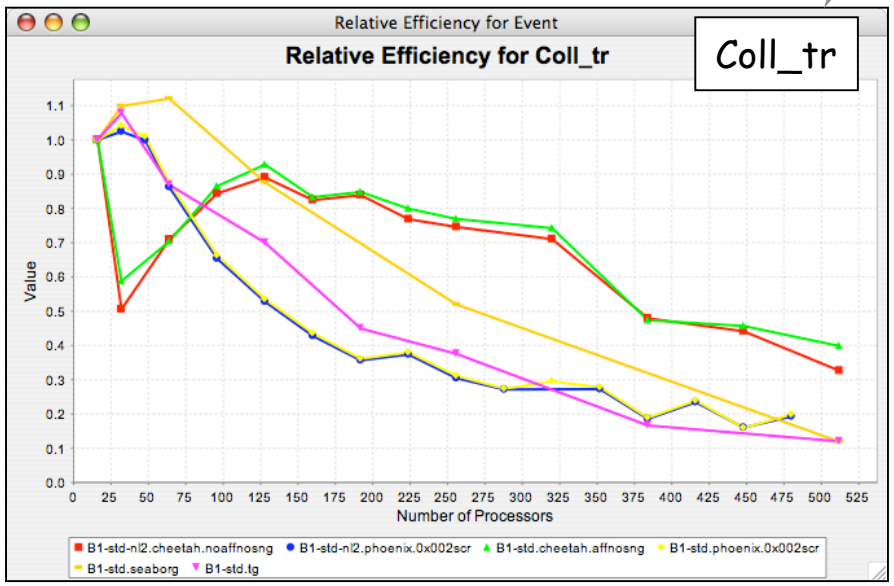
Data: GYRO on various architectures

PerfExplorer – GYRO Relative Efficiency

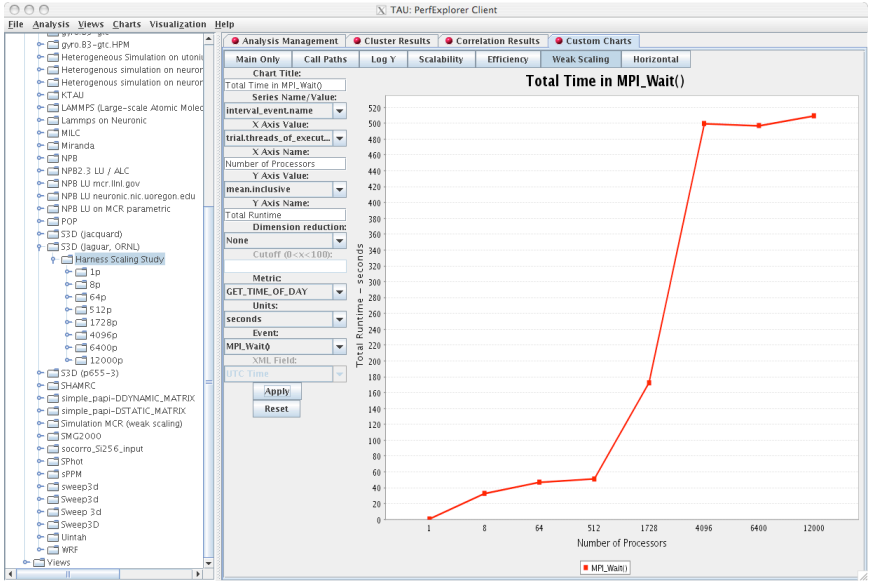
- By experiment (B1-std)
  - Total runtime (Cheetah (red))
- By event for one experiment
  - Coll_tr (blue) is significant
- By experiment for one event
  - Shows how Coll_tr behaves for all experiments
  - Data generated by Pat Worley, ORNL

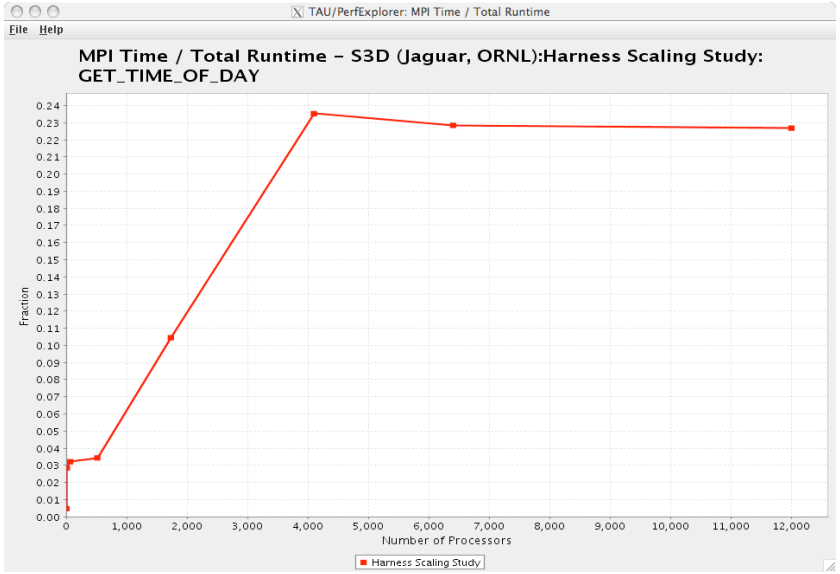


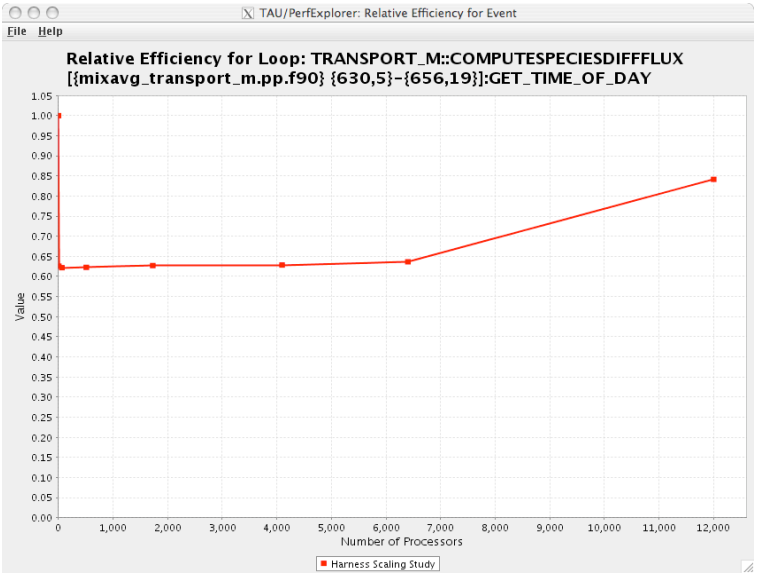


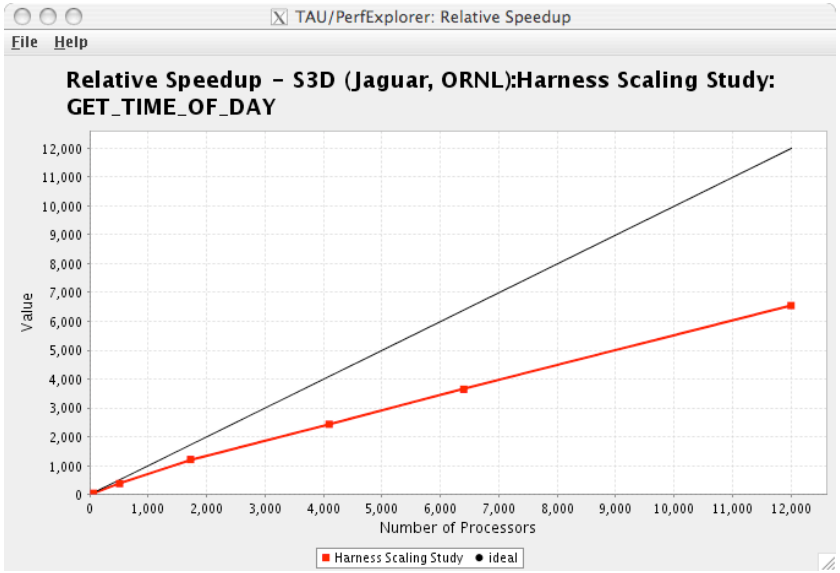


PerfExplorer: Cross Experiment Analysis for S3D

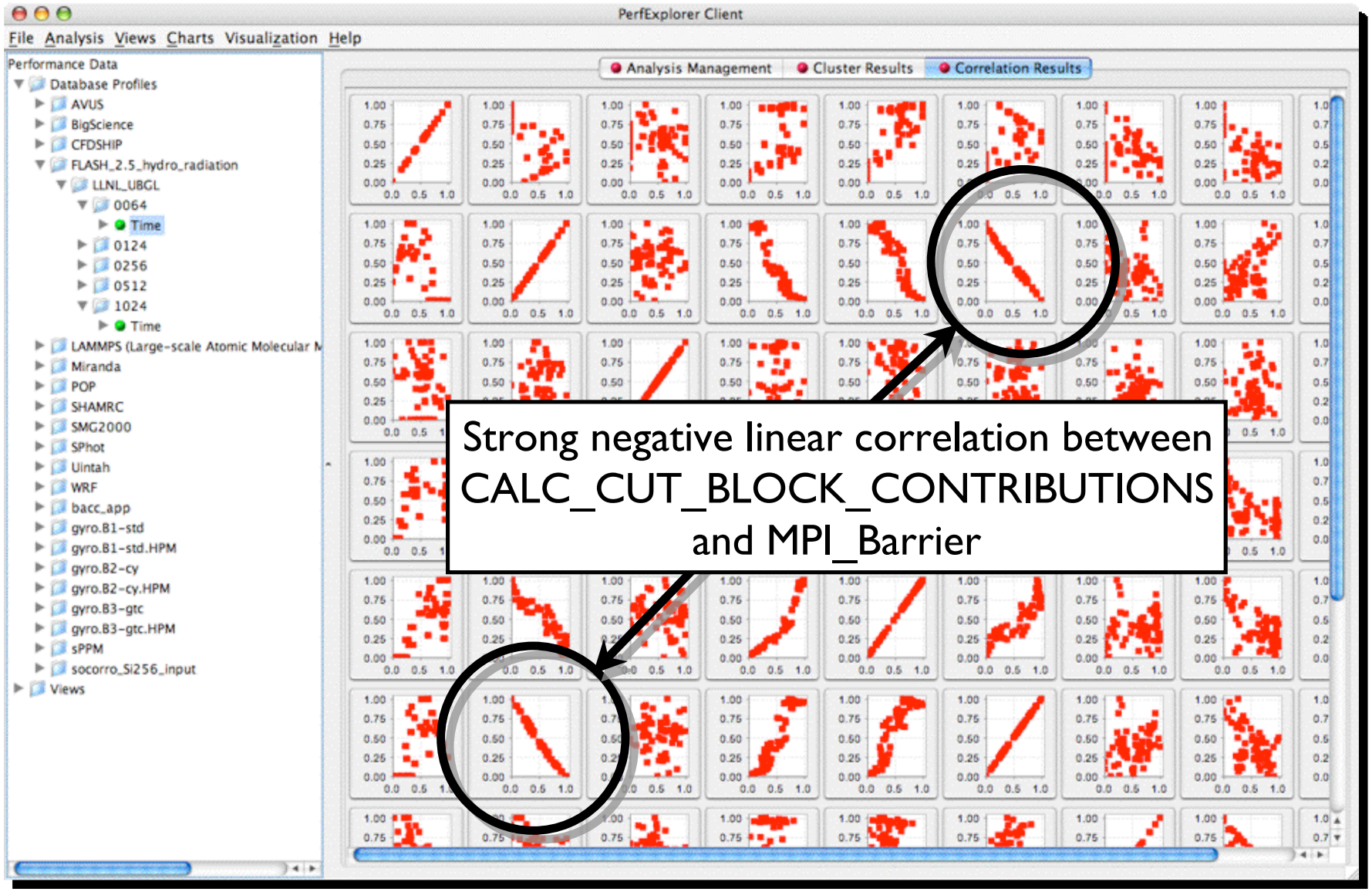








Correlation Analysis

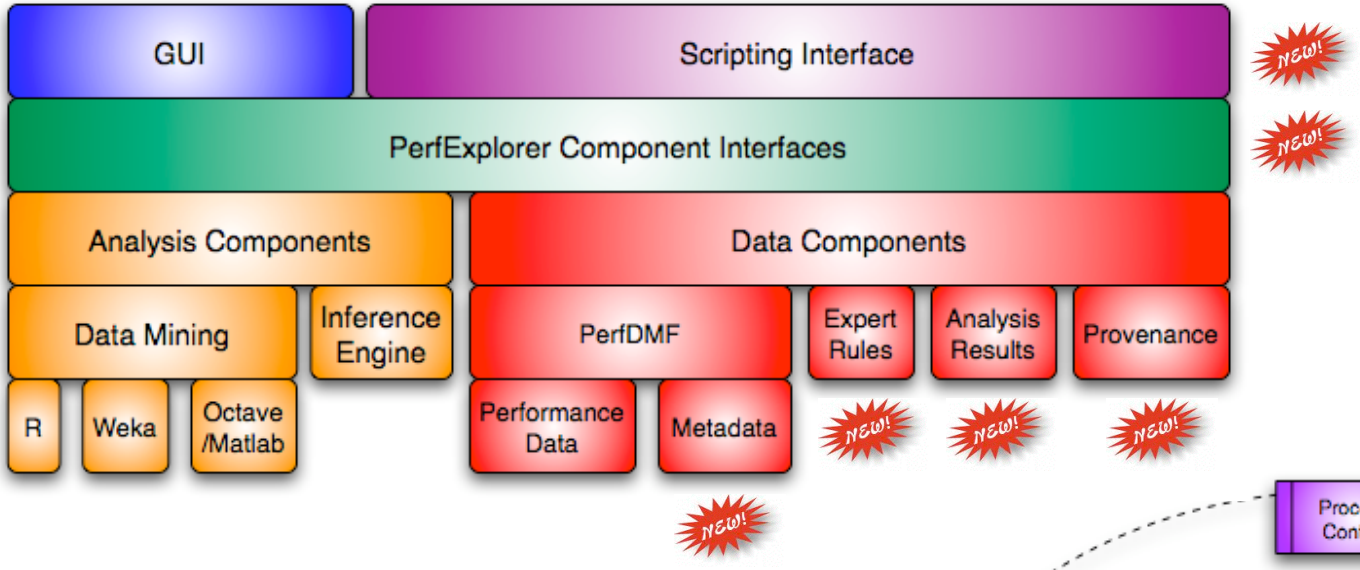


Data: FLASH on BGL(LLNL), 64 nodes

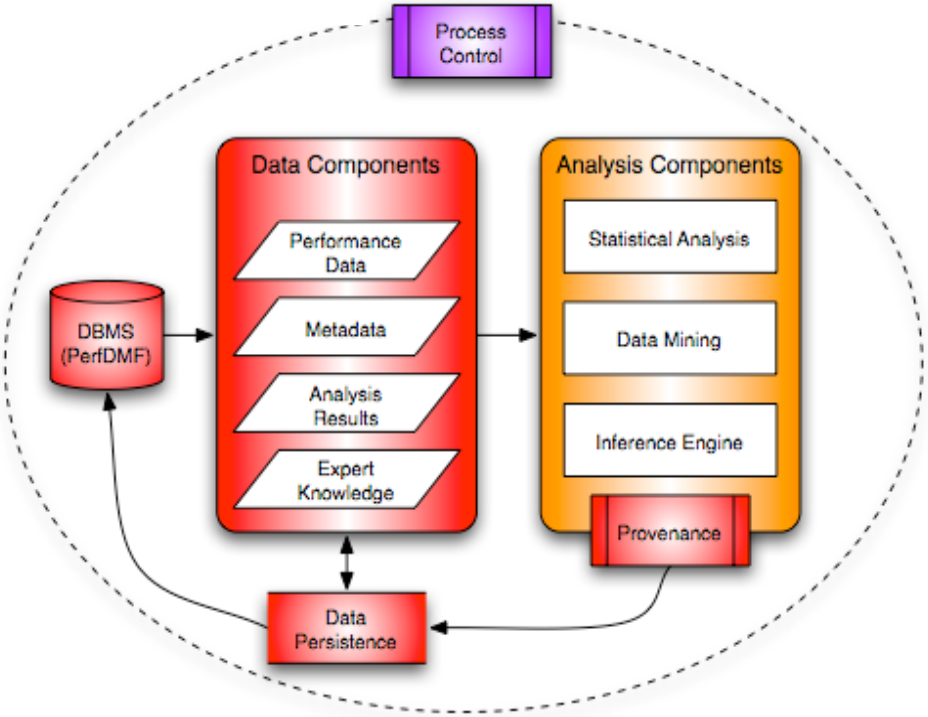
PerfExplorer v2 – Requirements and Features

- Component-based analysis process
  - Analysis operations implemented as modules
  - Linked together in analysis process and workflow
- Scripting
  - Provides process/workflow development and automation
- Metadata input, management, and access
- Inference engine
  - Reasoning about causes of performance phenomena
  - Analysis knowledge captured in expert rules
- Persistence of intermediate results
- Provenance
  - Provides historical record of analysis results

PerfExplorer v2: Architecture and Interaction



Interaction workflow

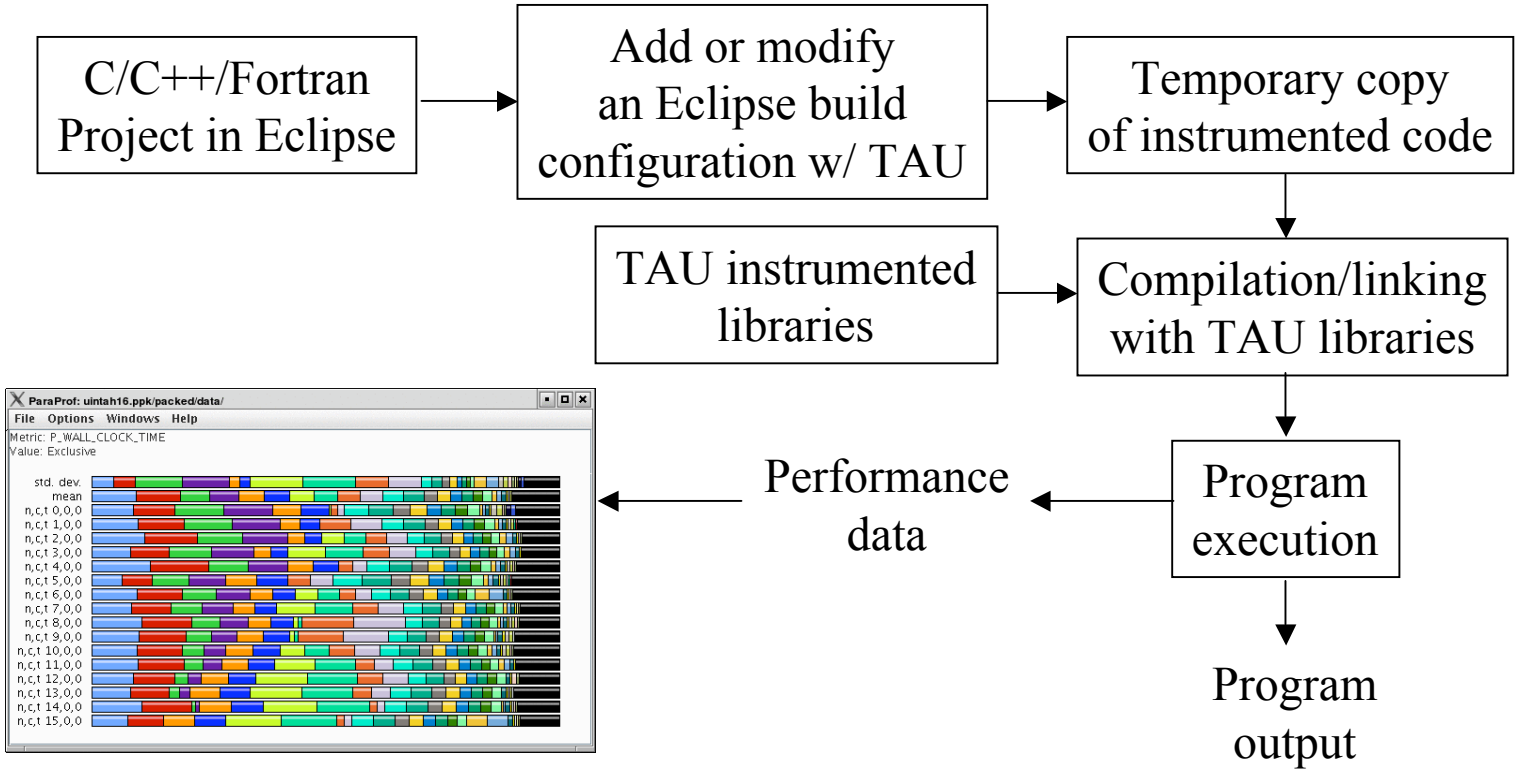


TAU Integration with IDEs

- 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 ...

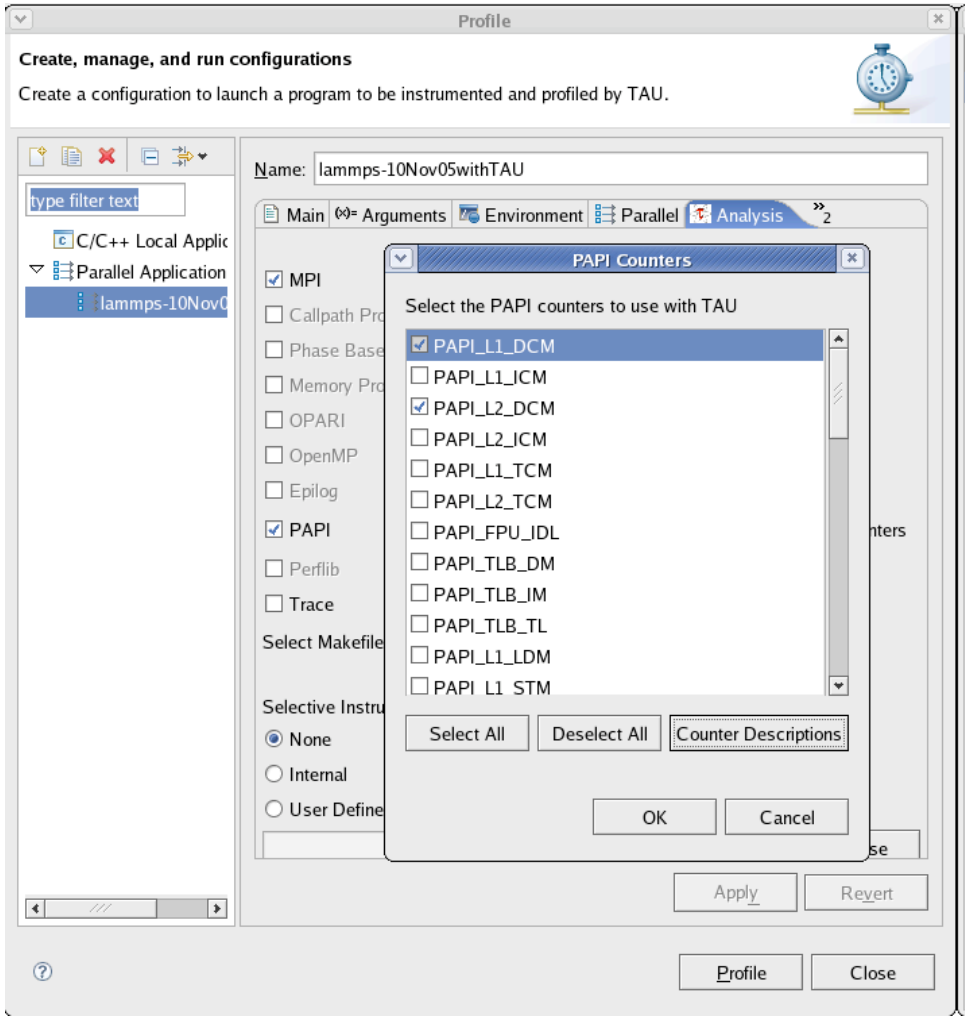
- initialize
- multiply_matrices
- main

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's 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

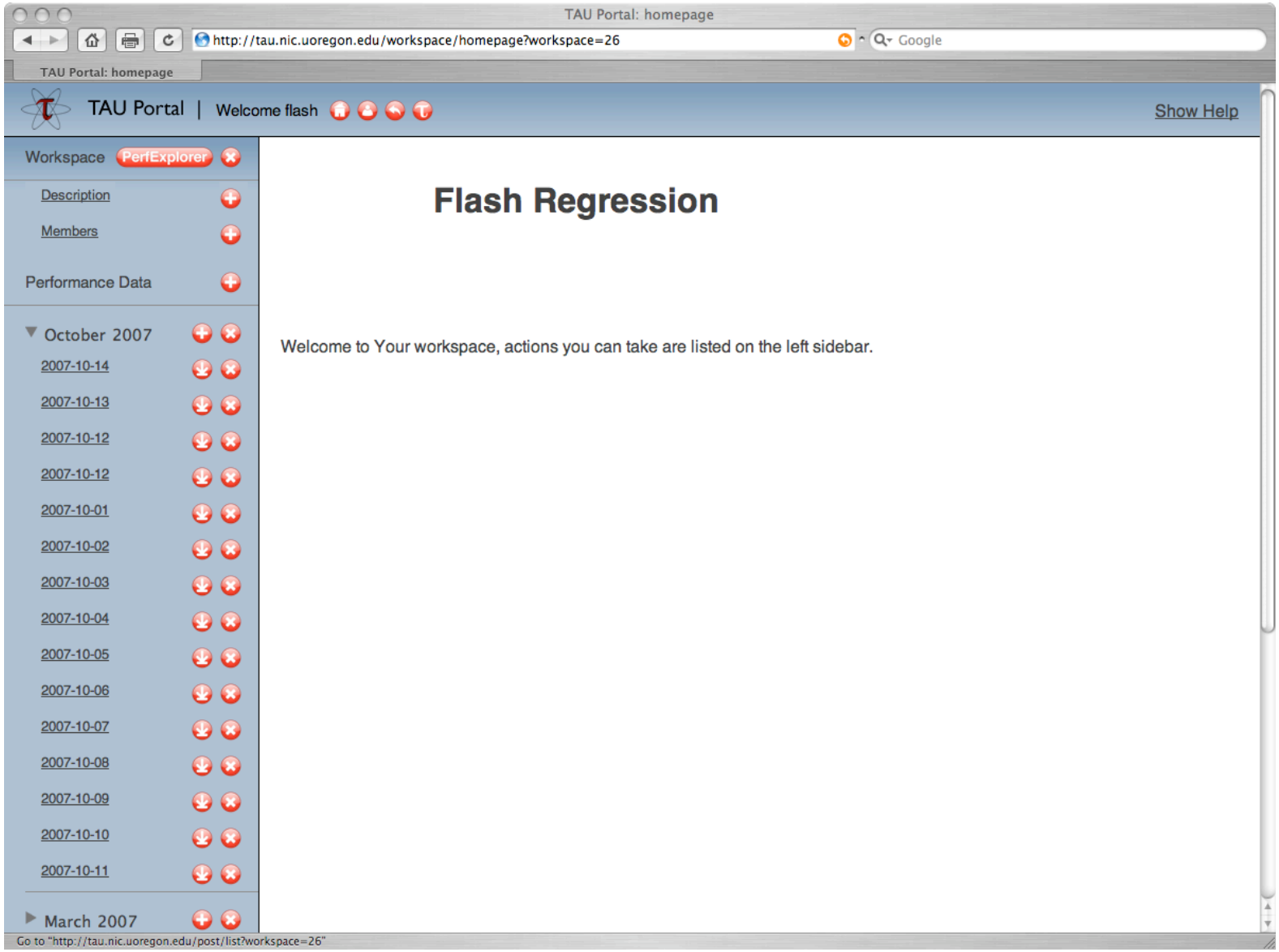
TAU Eclipse Plug-In Features

- Performance data collection
  - Graphical selection of TAU stub makefiles and compiler options
  - Automatic instrumentation, compilation and execution of target C, C++ or Fortran projects
  - Selective instrumentation via source editor and source outline views
  - Full integration with the Parallel Tools Platform (PTP) parallel launch system for performance data collection from parallel jobs launched within Eclipse
- Performance data management
  - Automatically place profile output in a PerfDMF database or upload to TAU-Portal
  - Launch ParaProf on profile data collected in Eclipse, with performance counters linked back to the Eclipse source editor

TAU Portal

- Web-based access to TAU
- Support collaborative performance study
  - Secure performance data sharing
  - Does not require TAU installation
  - Launch TAU performance tools with Java WebStart
    - ParaProf, PerfExplorer
- FLASH regression testing
  - Nightly regression testcases
  - Uploaded to the database automatically
  - Interactive review of performance through TAU portal
  - Multi-experiment analysis

Portal: Nightly Performance Regression Testing



TAU Portal: Launch ParaProf/PerfExplorer

TAU Portal: edit

http://tau.nic.uoregon.edu/trial/edit/356?workspace=26

TAU Portal | Welcome flash

Hide Help

Workspace PerfExplorer

Description

Members

Performance Data

October 2007

2007-10-14
2007-10-13
2007-10-12
2007-10-12
2007-10-01
2007-10-02
2007-10-03
2007-10-04
2007-10-05
2007-10-06
2007-10-07
2007-10-08
2007-10-09
2007-10-10
2007-10-11

March 2007

April 2007

May 2007

June 2007

2007-10-14 trial metadata ParaProf

Name	2007-10-14
Time	2007 October 14
Number of Nodes	4
Contexts per Node	1
Threads per Context	1
Compiler	
CPU Cores	2
CPU MHz	2992.504
CPU Type	Intel(R) Xeon(R) CPU 5160
CPU Vendor	GenuineIntel
CWD	/mnt/netapp/home/users
Cache Size	4096 KB
Executable	/mnt/netapp/home/users
Hostname	sigma.cs.uoregon.edu
Local Time	2007-10-14T00:17:28-0
MPI Processor Name	sigma.cs.uoregon.edu
Memory Size	4040776 kB
Node Name	sigma.cs.uoregon.edu
OS Machine	x86_64
OS Name	Linux
OS Release	2.6.9-55.0.9.ELsmp
OS Version	#1 SMP Thu Sep 27 18:28:
TAU Architecture	x86_64

Trial Edit

This is the page where you can view and edit TAU profiles. The table show all the metadata associated with this trial; you may edit any of the attributes by click on its value. You may also create new attributes by clicking on the *--New attribute--/--New value--* field.

Here is some information about the actions you can take on this page:

Table

Click to download the profile file in PPK format.

ParaProf This will lanuch TAU's paraprof profile viewer.

Click to remove this profile from the workspace.

Sidebar

The Left sidebar help you manage the content of each workspace. At the top are the links to view/edit the workspace description, or its members. Next the sidebar list all the performance data in the workspace. The profiles are grouped by experiment (create a new experiment using the besides the 'Performance Data' heading). With each experiment you can add new profiles, or remove it from the workspace. For each profile you can click its name to view the details, download the porfile, or remove it from the workspace. *You can move a profile between experiments by dragging the row up or down.*

PerfExplorer Click here to lanuch TAU's *PerfExplorer* application and view/analyze all the proformance data in this workspace.

This button will add a another object (of the kind written to the left) to the workspace.

Clicking here will download the profile, file, or image.

This will remove the profile, experiment, file, or image from the workspace.

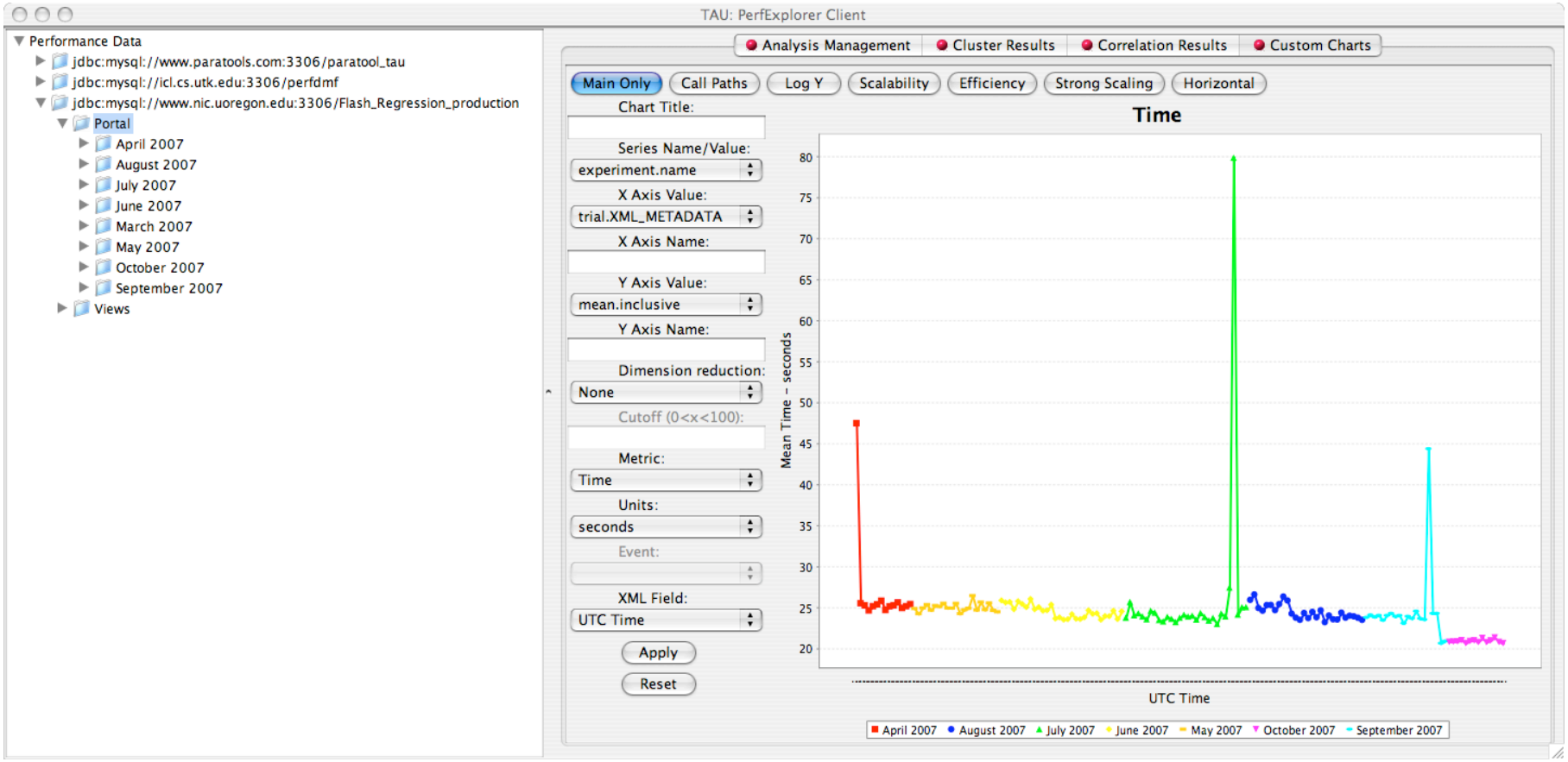
Header

You may return to your homepage at any time by using this button.

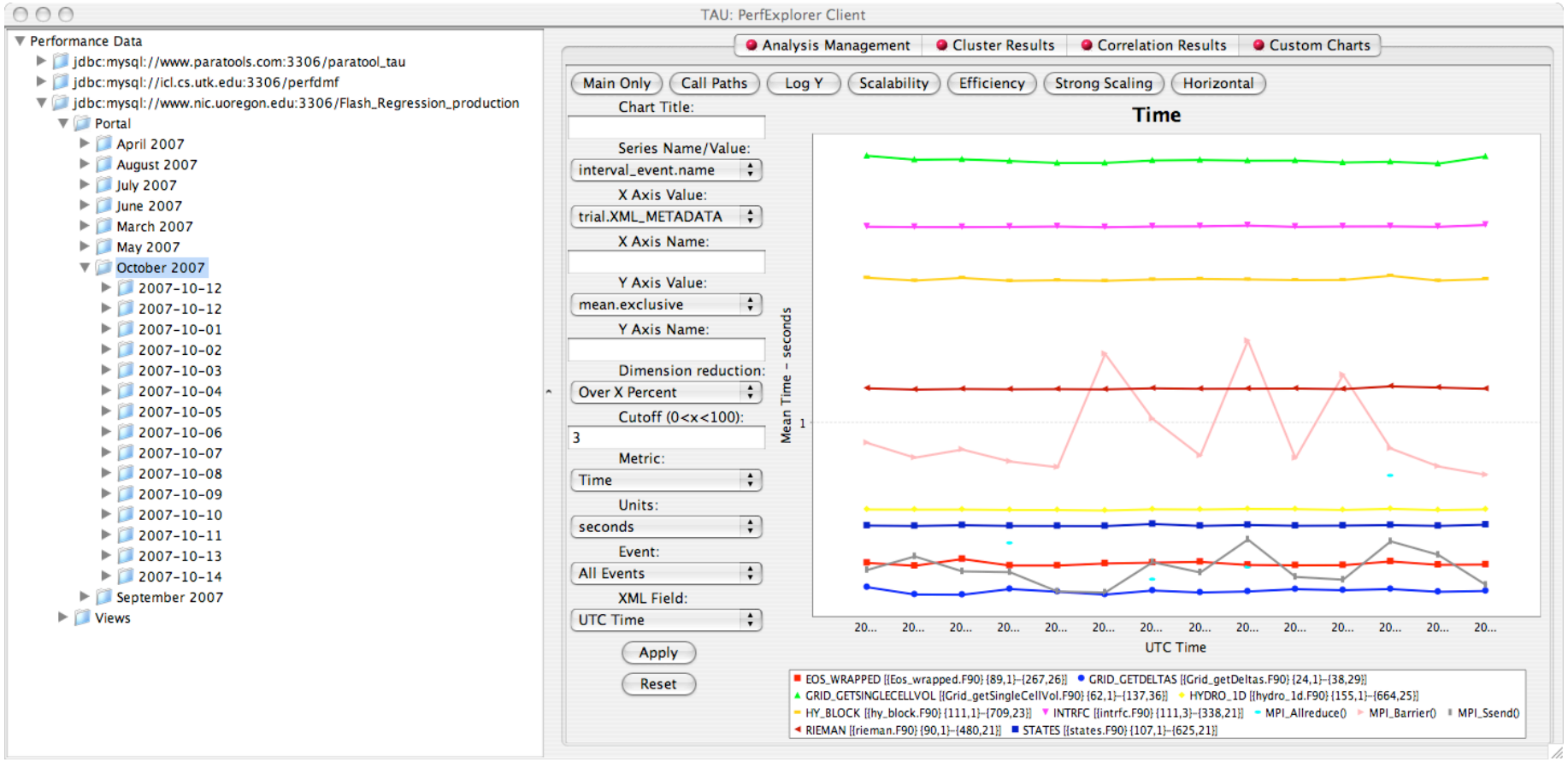
Click here to edit any of your user attributes that you specified when signing up of the TAU portal.

Logout by using this button (do this if you want switch user).

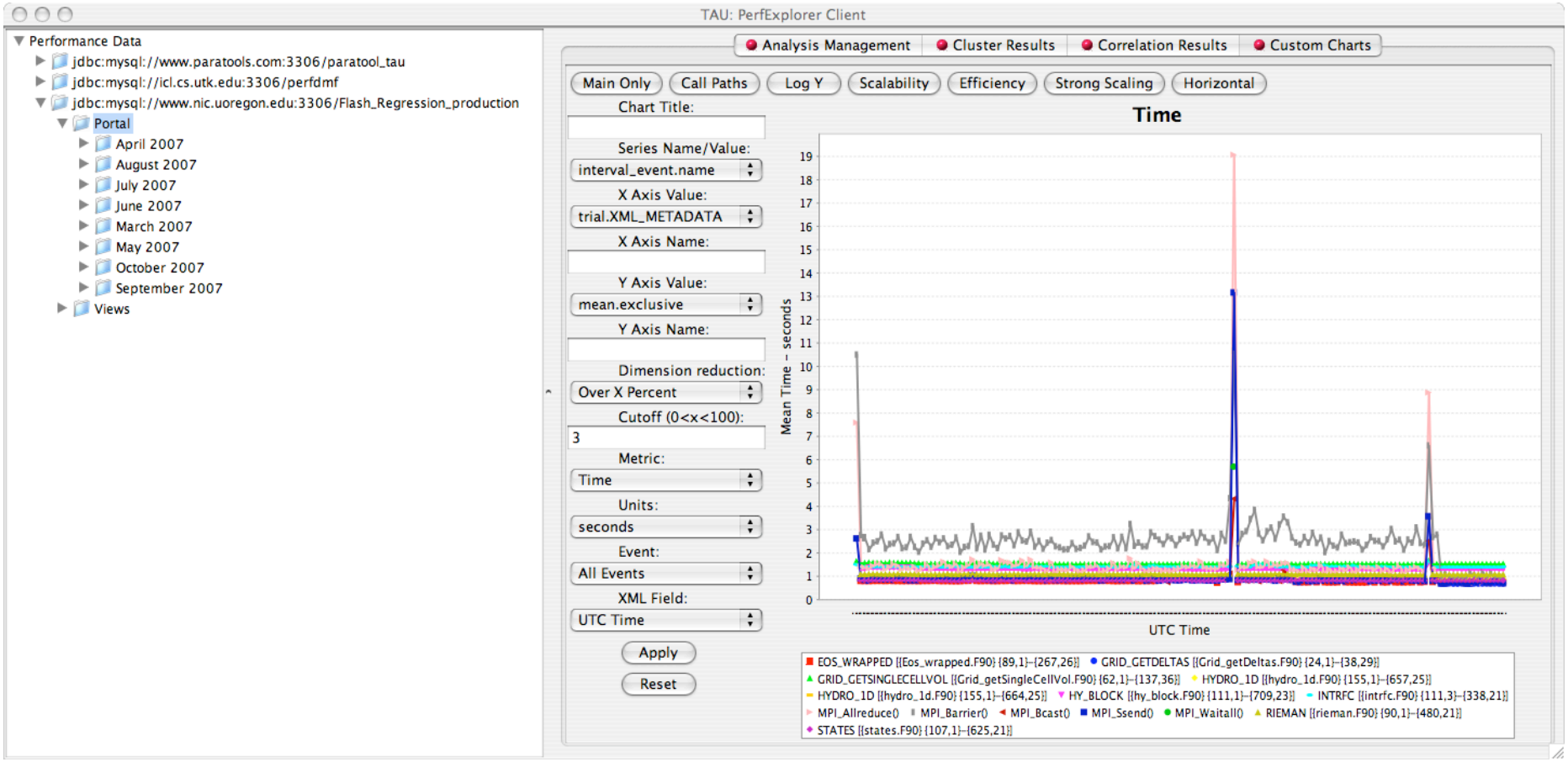
PerfExplorer: Regression Testing



PerfExplorer: Limiting Events (> 3%), Oct 2007



PerfExplorer: Exclusive Time for Events (2007)

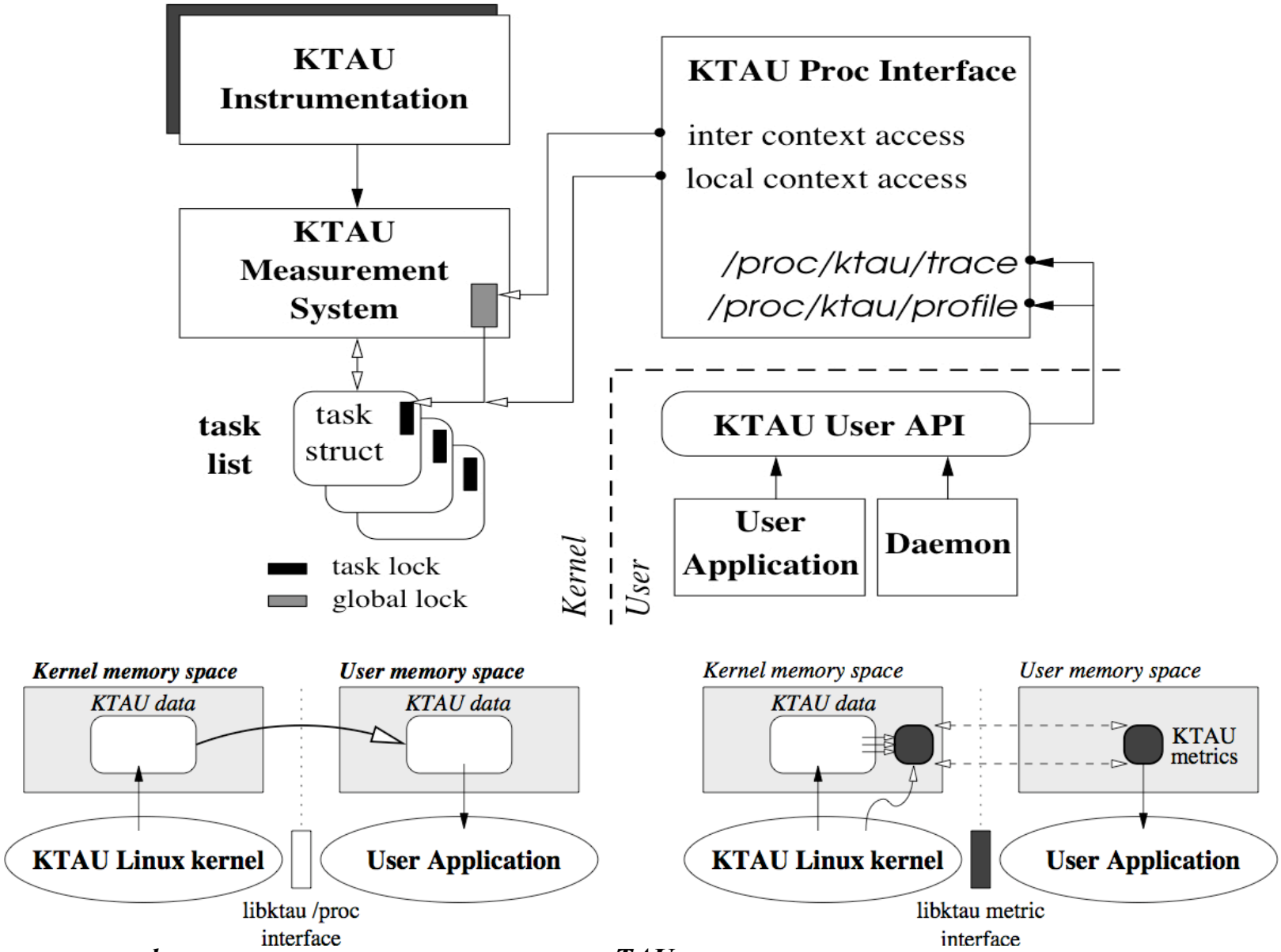


Full System Performance: the KTAU Project

- Trend toward extremely large scales
  - System-level influences are increasingly dominant performance bottleneck contributors
  - Application sensitivity at scale to the system
  - Complex I/O path and subsystems another example
  - Isolating system-level factors non-trivial
- OS Kernel instrumentation and measurement is important to understanding system-level influences
- How to correlate application and OS performance?
- KTAU / TAU (Part of the ANL/UO ZeptoOS Project)

A. Nataraj, A. Malony, S. Shende, and A. Morris, "**Kernel-level Measurement for Integrated Performance Views: the KTAU Project**," *Cluster 2006*.

KTAU System Architecture



Applying KTAU+TAU

- How does real OS-noise affect real applications?
  - Requires OS + application performance measurement
  - Estimate application slowdown due to noise components
    - interrupts and scheduling are significant

A. Nataraj, A. Morris, A. Malony, M. Sottile, and P. Beckman, "**The Ghost in the Machine: Observing the Effects of Kernel Operation on Parallel Application Performance**," *SC'07*. Wednesday, 10:30-12:00.

- Performance of multi-layered I/O systems
  - Requires measurement and analysis of multi-component I/O subsystems in system
  - Tracking of I/O "long path" and assignment to application
  - Working with Argonne on PVFS2

TAU Monitoring

- Runtime access to parallel performance data
- Monitoring modes
  - Offline / Post-mortem observation and analysis
    - least requirements for a specialized transport
  - Online observation
    - long running applications, especially at scale
    - Dumping snapshots to file-system can be suboptimal
  - Online observation with feedback into application
- TAUoverSupermon (Sottile and Minnich, LANL)

A. Nataraj, M. Sottile, A. Morris, A. Malony, and S. Shende, "**TAUoverSupermon: Low-overhead Online Parallel Performance Monitoring**," *Euro-Par 2007*.

- TAUoverMRNET (Arnold and Miller, UWisconsin)

Project Affiliations (selected)

- Lawrence Livermore National Lab
  - Hydrodynamics (*Miranda*), radiation diffusion (*KULL*)
  - Open Trace Format (*OTF*) implementation on BG/L
- Lawrence Berkeley National Lab
  - ACTS Toolkit
- Argonne National Lab
  - ZeptoOS project and KTAU
  - Astrophysical thermonuclear flashes (*Flash*)
- Center for Simulation of Accidental Fires and Explosion
  - University of Utah, ASCI ASAP Center, C-SAFE
  - Uintah Computational Framework (*UCF*)
- Oak Ridge National Lab
  - Contribution to the Joule Report/PERI for *S3D*, *GYRO*, *AORSA3D*
- NASA Goddard Space Flight Center, NASA Ames
  - GEOS/GCM, modelE, ...

Project Affiliations (continued)

- Sandia National Lab
  - Simulation of turbulent reactive flows (*S3D*)
  - Combustion code (*CFRFS*)
- Los Alamos National Lab
  - Monte Carlo transport (*MCNP*)
  - SAIC's Adaptive Grid Eulerian (*SAGE, RAGE*)
  - perflib integration (Jeff Brown)
- *CCSM* / *ESMF* / *WRF* climate/earth/weather simulation
  - NSF, NOAA, DOE, NASA, …
- Common component architecture (*CCA*) integration
- Performance Engineering Research Institute (*PERI*)

Concluding Discussion

- Performance tools must be used effectively
- More intelligent performance systems for productive use
  - Evolve to application-specific performance technology
  - Deal with scale by “full range” performance exploration
  - Autonomic and integrated tools
  - Knowledge-based and knowledge-driven process
- Performance observation methods do not necessarily need to change in a fundamental sense
  - More automatically controlled and efficiently use
- Develop next-generation tools and deliver to community
- Open source with support by ParaTools, Inc.
- http://tau.uoregon.edu