

TAU Performance System[®]

CScADS

Jul 25, 2012

Wyatt Spear, Sameer Shende, Allen D. Malony, Scott Biersdorff,
Suzanne Millstein

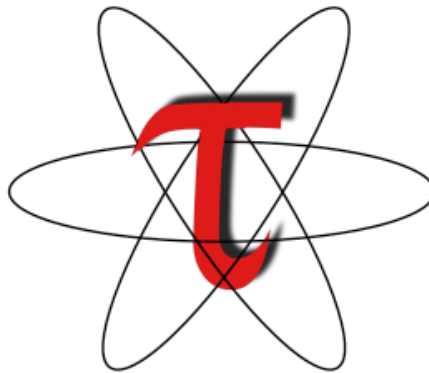
TAU team, University of Oregon

wspear@cs.uoregon.edu

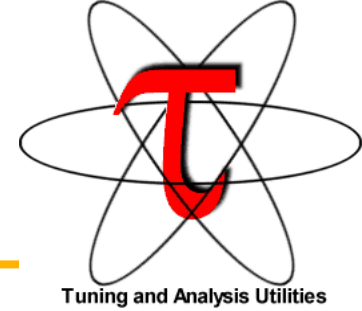
<http://tau.uoregon.edu>



Introduction



TAU Performance System

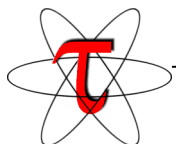


- <http://tau.uoregon.edu/>
- Multi-level performance instrumentation
 - Multi-language automatic source instrumentation
- Flexible and configurable performance measurement
- Widely-ported parallel performance profiling system
 - Computer system architectures and operating systems
 - Different programming languages and compilers
- Support for multiple parallel programming paradigms
 - Multi-threading, message passing, mixed-mode, hybrid
- Integration in complex software, systems, applications



For more information

- TAU Website: <http://tau.uoregon.edu>
 - Software
 - Release notes
 - Documentation
- TAU LiveDVD: <http://www.hpclinux.com>
 - Boot up on your laptop or desktop
 - Includes TAU and variety of other packages
 - Include documentation and tutorial slides



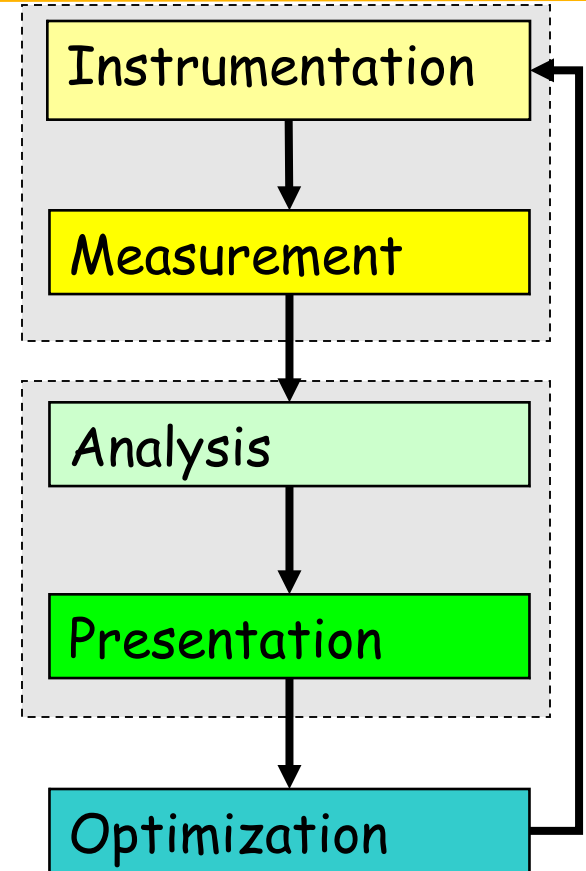
What is TAU?

- TAU is a performance evaluation tool
- It supports parallel profiling and tracing
- 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
- TAU uses a package called PDT for automatic instrumentation of the source code
- 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
 - paraprof is TAU's 3D profile browser
- To use TAU's automatic source instrumentation, you need to set a couple of environment variables and substitute the name of your compiler with a TAU shell script



Performance Optimization Cycle

- Design experiment
- Collect performance data
- Calculate metrics
- Analyze results
- Visualize results
- Identify bottlenecks and causes
- Tune performance



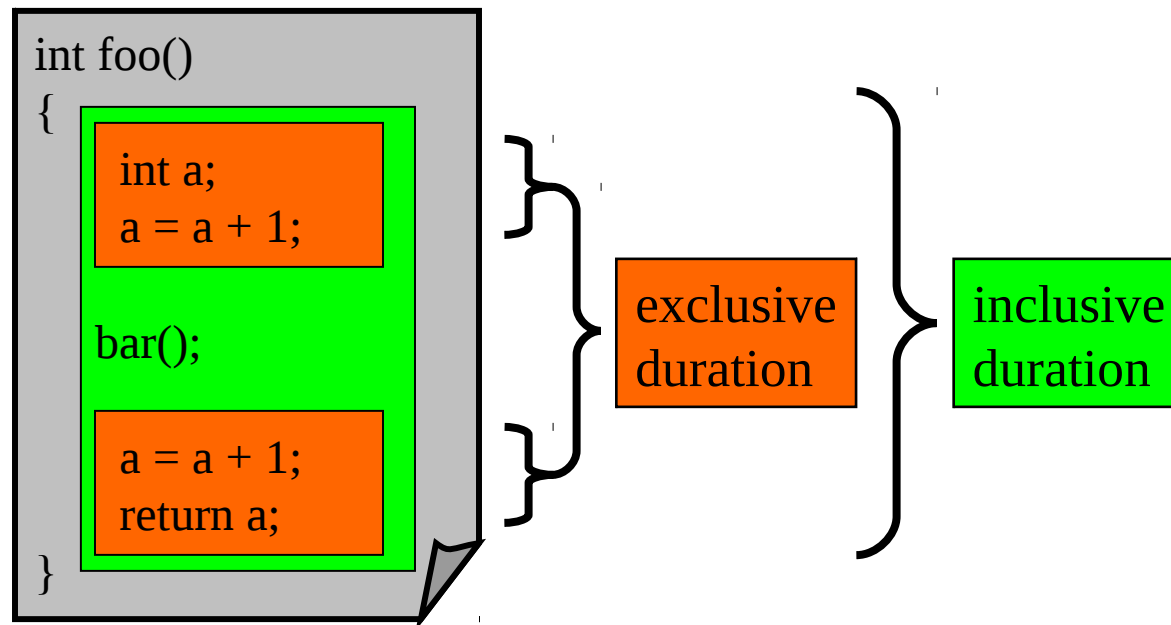
TAU Instrumentation Approach

- Supports both direct and indirect performance observation
 - Direct instrumentation of program (system) code (probes)
 - Instrumentation invokes performance measurement
 - Event measurement: performance data, meta-data, context
 - Indirect mode supports sampling based on periodic timer or hardware performance counter overflow based interrupts
- Support for standard program events
 - Routines, classes and templates
 - Statement-level blocks and loops
 - 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)
 - Flexible selection of event statistics
- Provides static events and dynamic events



Inclusive and Exclusive Profiles

- Performance with respect to code regions
- Exclusive measurements for region only
- Inclusive measurements includes child regions



Interval Events, Atomic Events in TAU

NODE 0;CONTEXT 0;THREAD 0:

%Time	Exclusive msec	Inclusive total msec	#Call	#Subrs	Inclusive usec/call	Name
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()

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

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

Atomic events
(trigger with
value)

27.1

1%

% setenv TAU_CALLPATH_DEPTH 0

% setenv TAU_TRACK_HEAP 1



Atomic Events, Context Events

%Time	Exclusive msec	Inclusive total msec	#Call	#Subrs	Inclusive Name usec/call
100.0	0.253	1,106	1	44	1106701 int main(int, char **) C
93.2	1.031	1,031	1	0	1031311 MPI_Init()
6.0	1	66	40	320	1650 void func(int, int) C
5.7	63	63	40	0	1588 MPI_Barrier()
0.8	9	9	1	0	9119 MPI_Finalize()
0.1	1	1	120	0	10 MPI_Recv()
0.0	0.141	0.141	120	0	1 MPI_Send()
0.0	0.085	0.085	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
40	40	40	40	0	Message size for broadcast
365	5.139E+04	44.39	3.091E+04	1.234E+04	Heap Memory Used (KB) : Entry
40	5.139E+04	3097	3.114E+04	1.227E+04	Heap Memory Used (KB) : Entry : MPI_Barrier()
40	5.139E+04	1.13E+04	3.134E+04	1.187E+04	Heap Memory Used (KB) : Entry : MPI_Bcast()
1	2067	2067	2067	0	Heap Memory Used (KB) : Entry : MPI_Comm_rank()
1	2066	2066	2066	0	Heap Memory Used (KB) : Entry : MPI_Comm_size()
1	5.139E+04	5.139E+04	5.139E+04	0.0006905	Heap Memory Used (KB) : Entry : MPI_Finalize()
1	57.56	57.56	57.56	0	Heap Memory Used (KB) : Entry : MPI_Init()
120	5.139E+04	1.13E+04	3.134E+04	1.187E+04	Heap Memory Used (KB) : Entry : MPI_Recv()
120	5.139E+04	1.129E+04	3.134E+04	1.187E+04	Heap Memory Used (KB) : Entry : MPI_Send()
1	44.39	44.39	44.39	0	Heap Memory Used (KB) : Entry : int main(int, char **) C
40	5.036E+04	2068	3.011E+04	1.227E+04	Heap Memory Used (KB) : Entry : void func(int, int) C

Atomic event

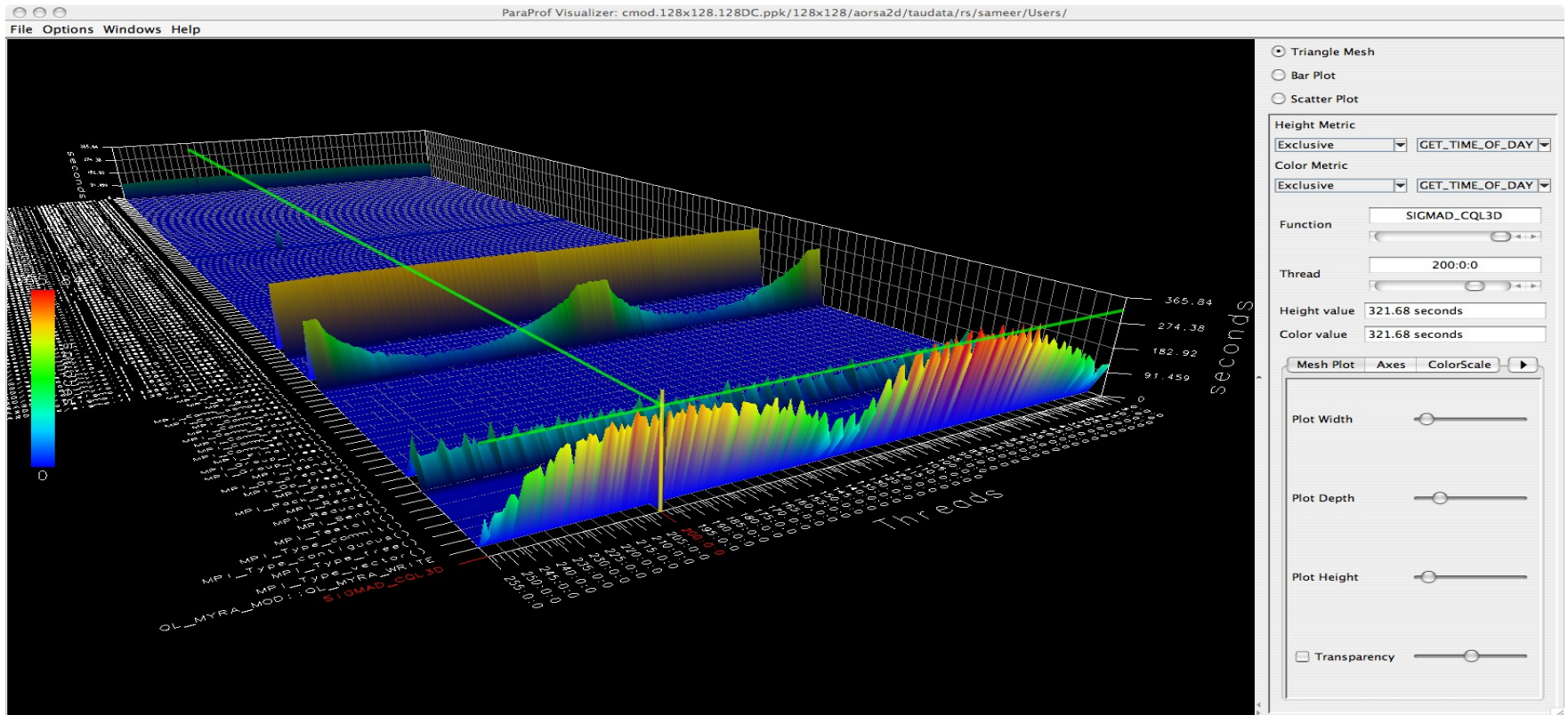
Context event
= atomic event
+ executing
context

% setenv TAU_CALLPATH_DEPTH 1

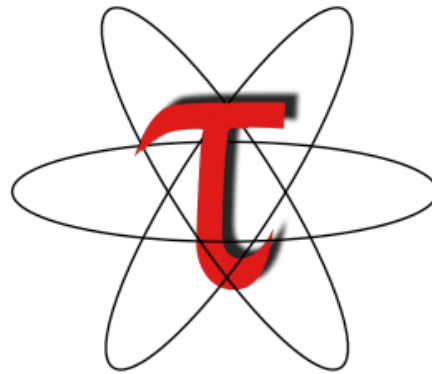
% setenv TAU_TRACK_HEAP 1



Parallel Profile Visualization: ParaProf



Overview of different methods of instrumenting applications



Instrumentation: Events in TAU

- Event types
 - Interval events (begin/end events)
 - measures performance between begin and end
 - metrics monotonically increase
 - Atomic events
 - used to capture performance data state
- Code events
 - Routines, classes, templates
 - Statement-level blocks, loops
- User-defined events
 - Specified by the user
- Abstract mapping events



Instrumentation Techniques

- Events defined by instrumentation access
- Instrumentation levels
 - Source code
 - Object code
 - Runtime system
 - Library code
 - Executable code
 - Operating system
- Different levels provide different information
- Different tools needed for each level
- Levels can have different granularity

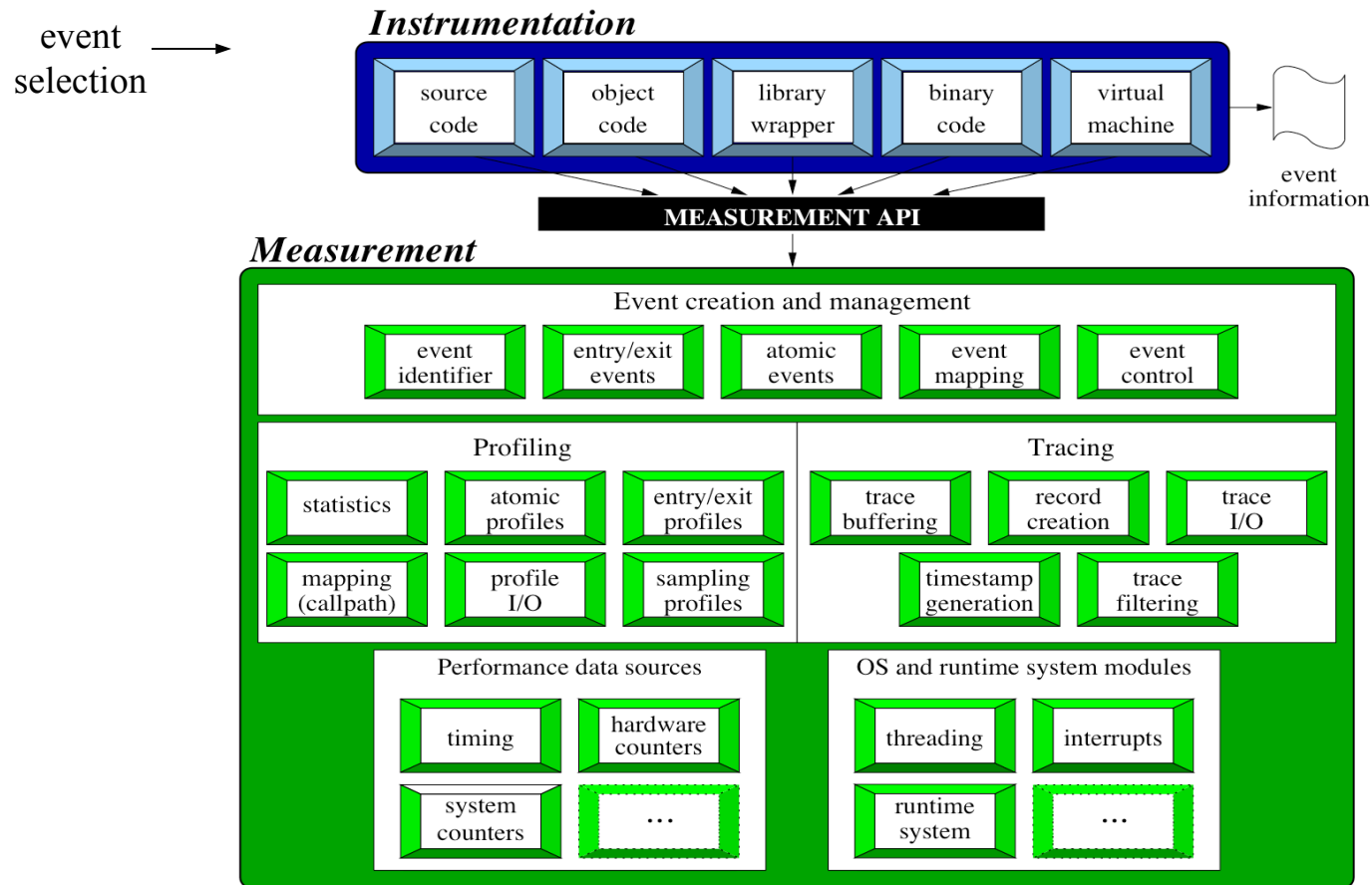


Instrumentation Techniques

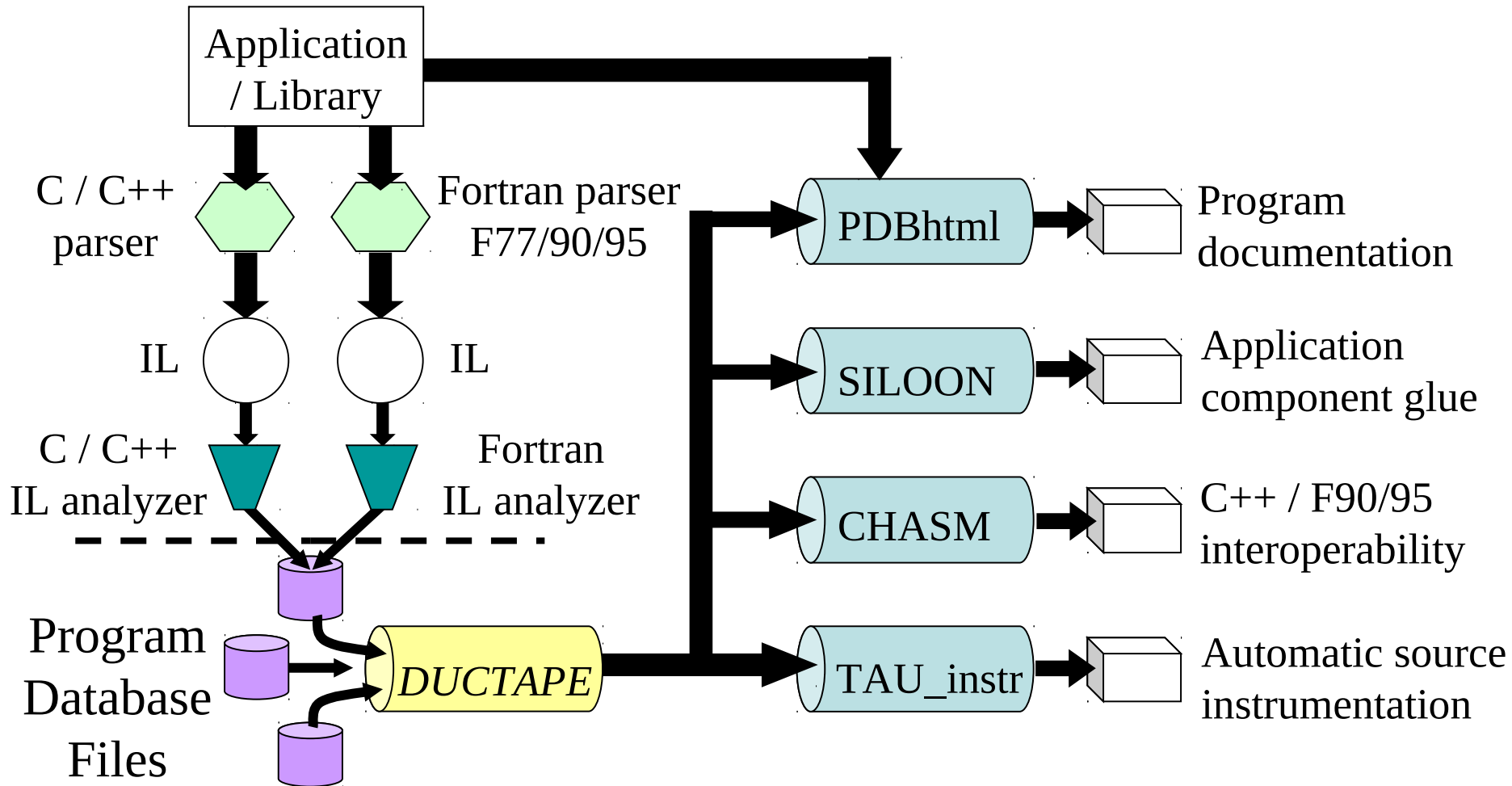
- Static instrumentation
 - Program instrumented prior to execution
- Dynamic instrumentation
 - Program instrumented at runtime
- Manual and automatic mechanisms
- Tool required for automatic support
 - Source time: preprocessor, translator, compiler
 - Link time: wrapper library, preload
 - Execution time: binary rewrite, dynamic
- Advantages / disadvantages



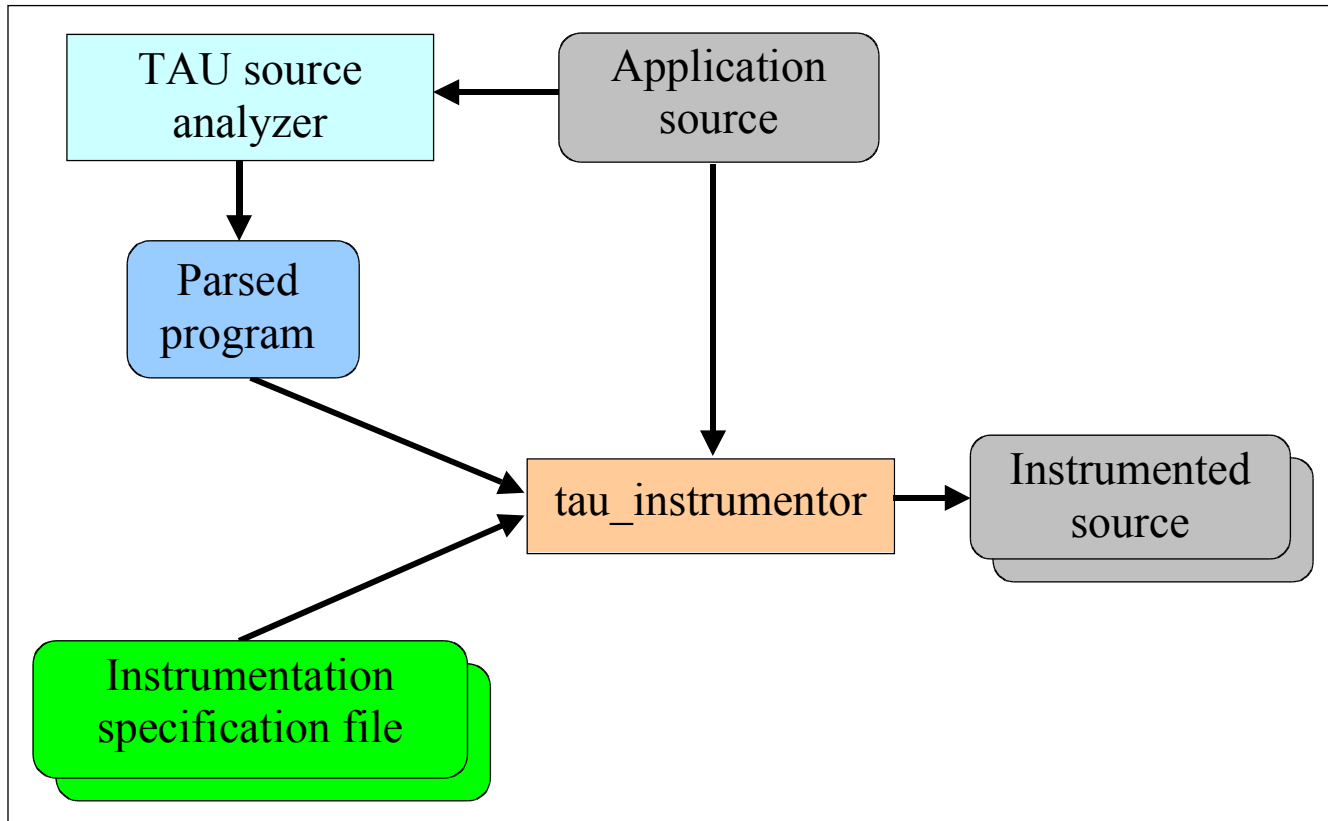
TAU Performance System Architecture



Program Database Toolkit (PDT)



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



Using TAU with source instrumentation

- TAU supports several measurement options (profiling, tracing, profiling with hardware counters, etc.)
- Each measurement configuration of TAU corresponds to a unique stub makefile and library that is generated when you configure it
- To instrument source code using PDT
 - Choose an appropriate TAU stub makefile in <arch>/lib:
% module load tau
% export TAU_MAKEFILE=\$TAULIBDIR/Makefile.tau-papi-mpi-pdt-pgi
% 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:
% 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

```
% cd $TAULIBDIR; ls Makefile.*
```

```
Makefile.tau-pdt-pgi
```

```
Makefile.tau-mpi-pdt-pgi
```

```
Makefile.tau-pthread-pdt-pgi
```

```
Makefile.tau-papi-mpi-pdt-pgi
```

```
Makefile.tau-papi-pthread-pdt-pgi
```

```
Makefile.tau-mpi-papi-pdt-pgi
```

```
Makefile.tau-mpi-pdt-openmp-opari
```

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

```
Makefile.tau-mpi-pdt-pgi
```

- Supports MPI instrumentation & PDT for automatic source instrumentation
- % export TAU_MAKEFILE=\$TAULIBDIR/Makefile.tau-mpi-pdt-pgi
- % tau_f90.sh matrix.f90 -o matrix
- % mpirun -np 256 ./matrix
- % paraprof



Usage Scenarios: Routine Level Profile

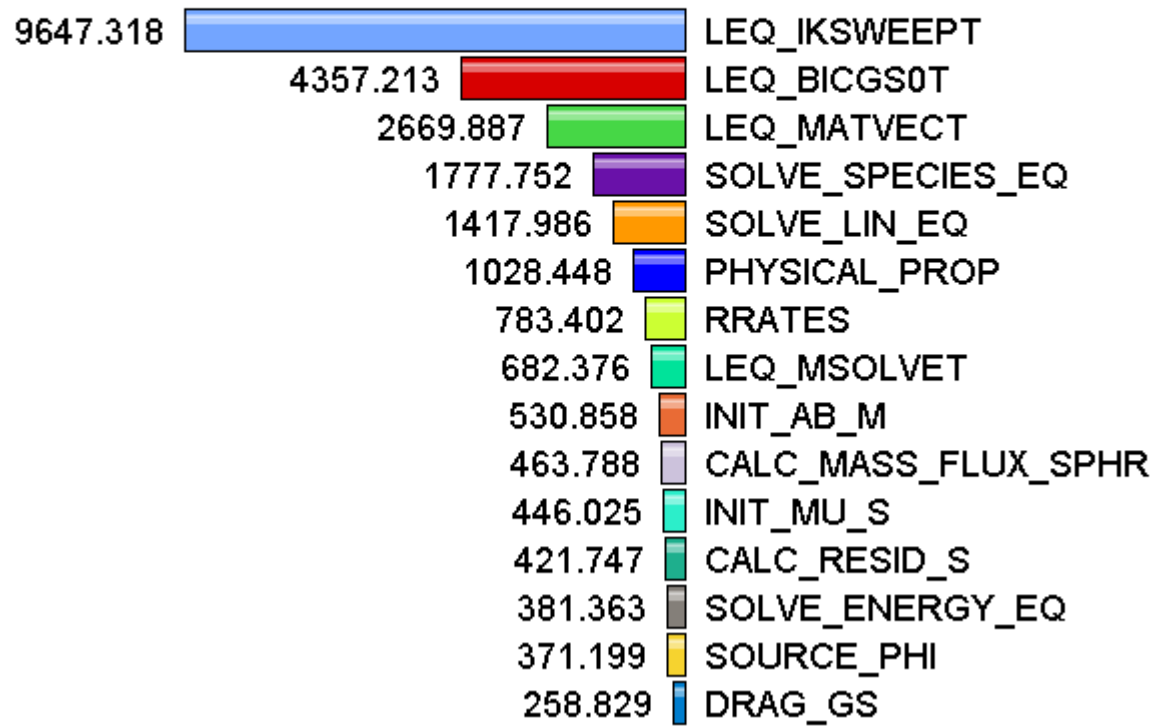
- Goal: What routines account for the most time? How much?

- Flat profile with wallclock time:

Metric: P_VIRTUAL_TIME

Value: Exclusive

Units: seconds



Solution: Generating a flat profile with MPI

```
% module load tau
% export TAU_MAKEFILE=$TAULIBDIR/Makefile.tau-mpi-pdt-pgi
% export PATH=$TAUROOTDIR/x86_64/bin:$PATH
% tau_f90.sh matmult.f90 -o matmult
(Or edit Makefile and change F90=tau_f90.sh)

% qsub -I -l nodes=1:ppn=8 -X
% mpirun -np 8 ./matmult
% pprof
% paraprof &
OR
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.

% paraprof app.ppk
% paraprof &
```

Automatic Instrumentation

- We now provide compiler wrapper scripts
 - Simply replace `ftn` 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

```
CXX = CC
F90 = ftn
CFLAGS =
LIBS = -lm
OBS = f1.o f2.o f3.o ... fn.o

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

After

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

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

Compile-Time Environment Variables

- Optional parameters for TAU_OPTIONS: [tau_compiler.sh -help]
 - optVerbose Turn on verbose debugging messages
 - optComplnst Use compiler based instrumentation
 - optNoComplnst Do not revert to compiler instrumentation if source instrumentation fails.
 - optDetectMemoryLeaks Turn on debugging memory allocations/de-allocations to track leaks
 - optTrackIO Turn on tracking POSIX IO by linking TAU's wrapper library
 - 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)



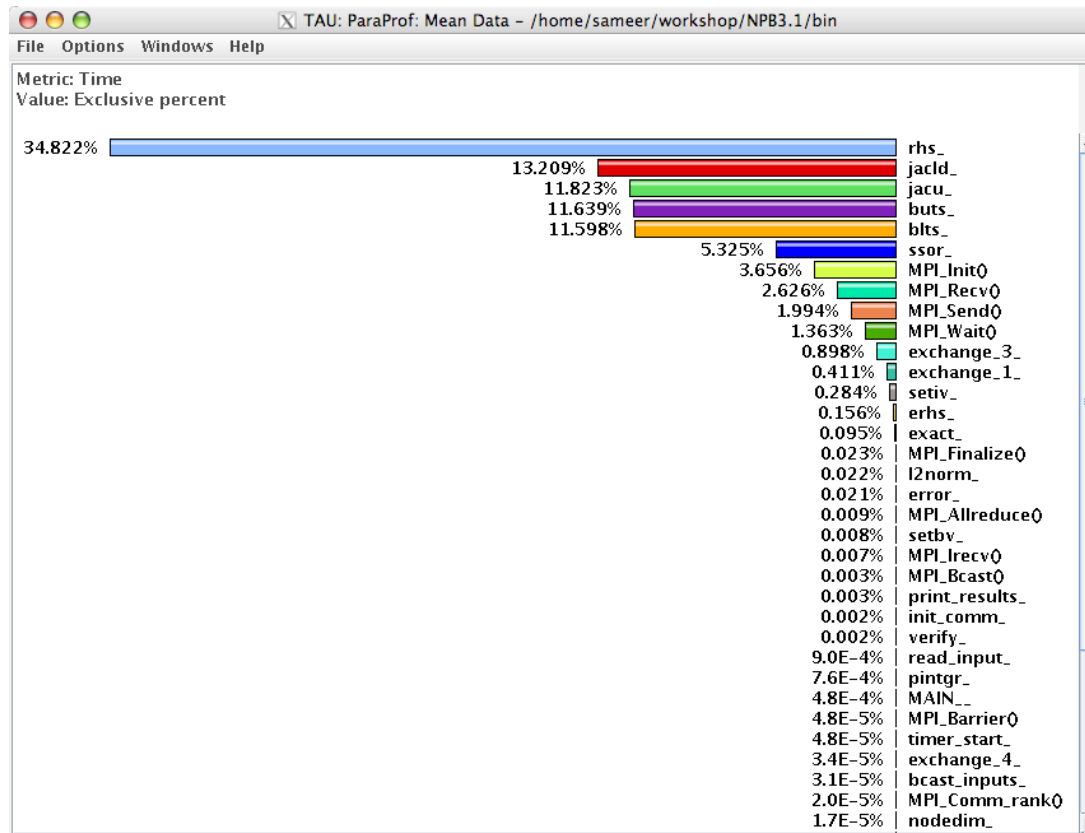
Compiling Fortran Codes with TAU

- 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='-optComplnst -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`



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=$TAULIBDIR/Makefile.tau-mpi-pdt-pgi
% export TAU_OPTIONS='-optCompInst -optVerbose'
% module load tau
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)

% mpirun -np 8 ./a.out
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk
```



Re-writing Binaries

- Support for both static and dynamic executables
- Specify the list of routines to instrument/exclude from instrumentation
- Specify the TAU measurement library to be injected
- Simplify the usage of TAU:
 - To instrument:
 - `% tau_run a.out -o a.inst`
 - To perform measurements, execute the application:
 - `% mpirun -np 4 ./a.inst`
 - To analyze the data:
 - `% paraprof`

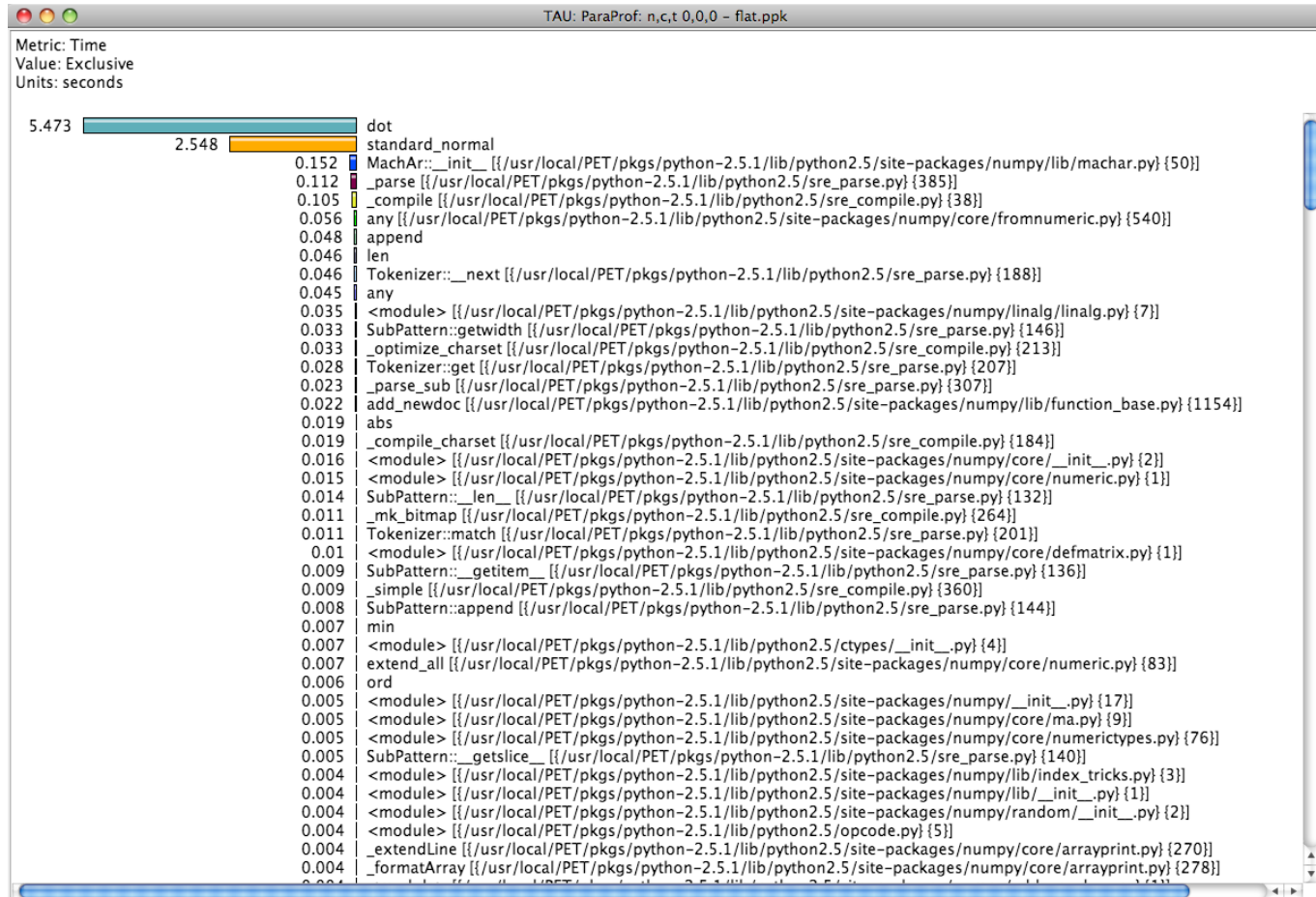


tau_run with NAS PBS

```
livetau@paratools01:~/tutorial$ cd ~/tutorial
livetau@paratools01:~/tutorial$ # Build an uninstrumented bt NAS Parallel Benchmark
livetau@paratools01:~/tutorial$ make bt CLASS=W NPROCS=4
livetau@paratools01:~/tutorial$ cd bin
livetau@paratools01:~/tutorial/bin$ # Run the instrumented code
livetau@paratools01:~/tutorial/bin$ mpirun -np 4 ./bt_W.4
livetau@paratools01:~/tutorial/bin$ # Instrument the executable using TAU with DyninstAPI
livetau@paratools01:~/tutorial/bin$ tau_run ./bt_W.4 -o ./bt.i
livetau@paratools01:~/tutorial/bin$ rm -rf profile.* MULT*
livetau@paratools01:~/tutorial/bin$ mpirun -np 4 ./bt.i
livetau@paratools01:~/tutorial/bin$ paraprof
livetau@paratools01:~/tutorial/bin$ # Choose a different TAU configuration
livetau@paratools01:~/tutorial/bin$ ls $TAU/libTAUsh
libTAUsh-depthlimit-mpi-pdt.so*      libTAUsh-papi-pdt.so*
libTAUsh-mpi-pdt.so*                 libTAUsh-papi-pthread-pdt.so*
libTAUsh-mpi-pdt-upc.so*              libTAUsh-param-mpi-pdt.so*
libTAUsh-mpi-python-pdt.so*           libTAUsh-pdt.so*
libTAUsh-papi-mpi-pdt.so*             libTAUsh-pdt-trace.so*
libTAUsh-papi-mpi-pdt-upc.so*          libTAUsh-phase-papi-mpi-pdt.so*
libTAUsh-papi-mpi-pdt-upc-udp.so*      libTAUsh-pthread-pdt.so*
libTAUsh-papi-mpi-pdt-vampirtrace-trace.so* libTAUsh-python-pdt.so*
libTAUsh-papi-mpi-python-pdt.so*
livetau@paratools01:~/tutorial/bin$ tau_run -XrunTAUsh-papi-mpi-pdt-vampirtrace-trace bt_W.4 -o bt.vpt
livetau@paratools01:~/tutorial/bin$ setenv VT_METRICS PAPI_FP_INS:PAPI_L1_DCM
livetau@paratools01:~/tutorial/bin$ mpirun -np 4 ./bt.vpt
livetau@paratools01:~/tutorial/bin$ vampir bt.vpt.otf &
```

Usage Scenarios: Instrument a Python program

- Goal: Generate a flat profile for a Python program



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
- 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

```
[sameer@zorak]$ 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)

- group (hdf5)

tau_exec loads libhdf5_wrap.so shared library using -loadlib=<libwrap_pkg.so>

- creates the wrapper/ directory with -opt

NODE 0;CONTEXT 0;THREAD 0:

%Time	Exclusive msec	Inclusive total msec	#Call	#Subrs	Inclusive Name 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()
0.1	0.001	0.001	2	0	0 herr_t H5open()
0.0	0	0	1	0	0 herr_t H5Tclose()

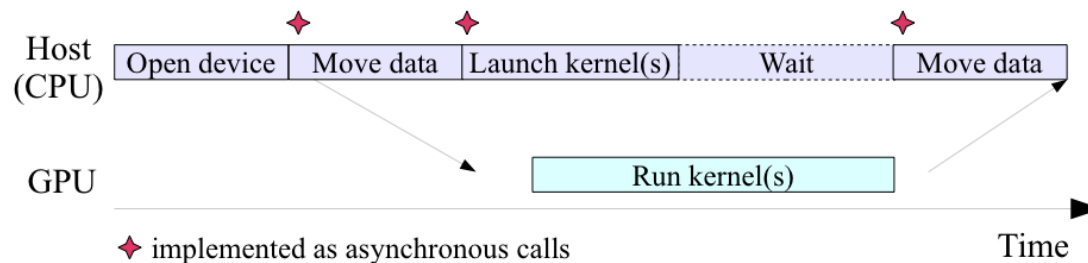
Profiling GPGPU Executions

- GPGPU compilers (e.g., CAPS hmpp and PGI) can now automatically generate GPGPU code using manual annotation of loop-level constructs and routines (hmpp)
- The loops (and routines for HMPP) are transferred automatically to the GPGPU
- TAU intercepts the runtime library routines and examines the arguments
- Shows events as seen from the host
- Profiles and traces GPGPU execution

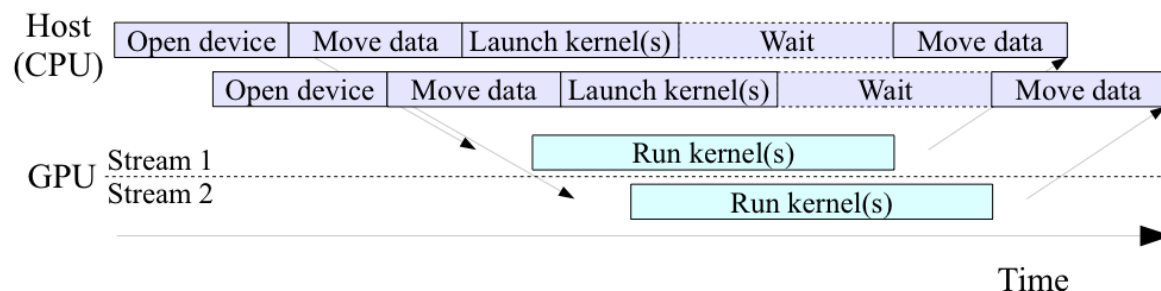


Host (CPU) - GPU Scenarios

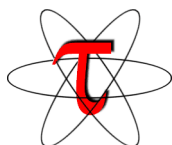
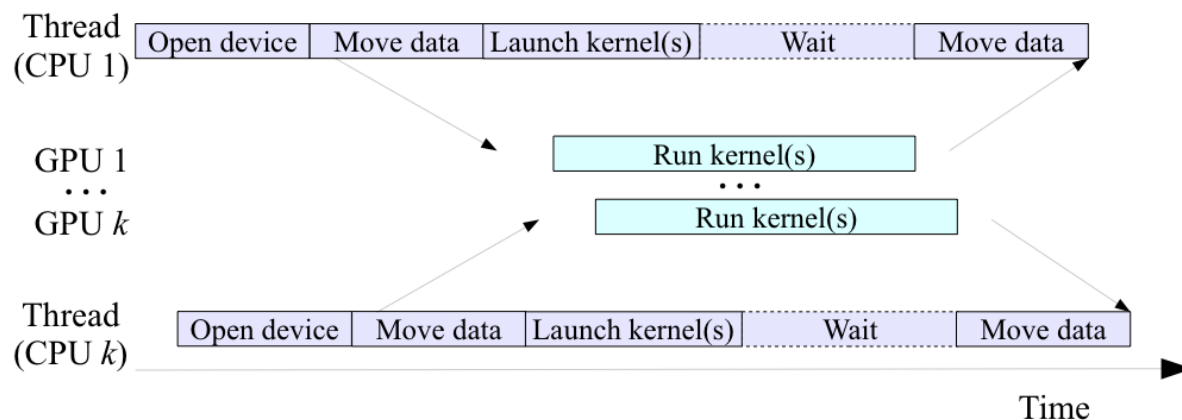
- Single GPU



- Multi-stream

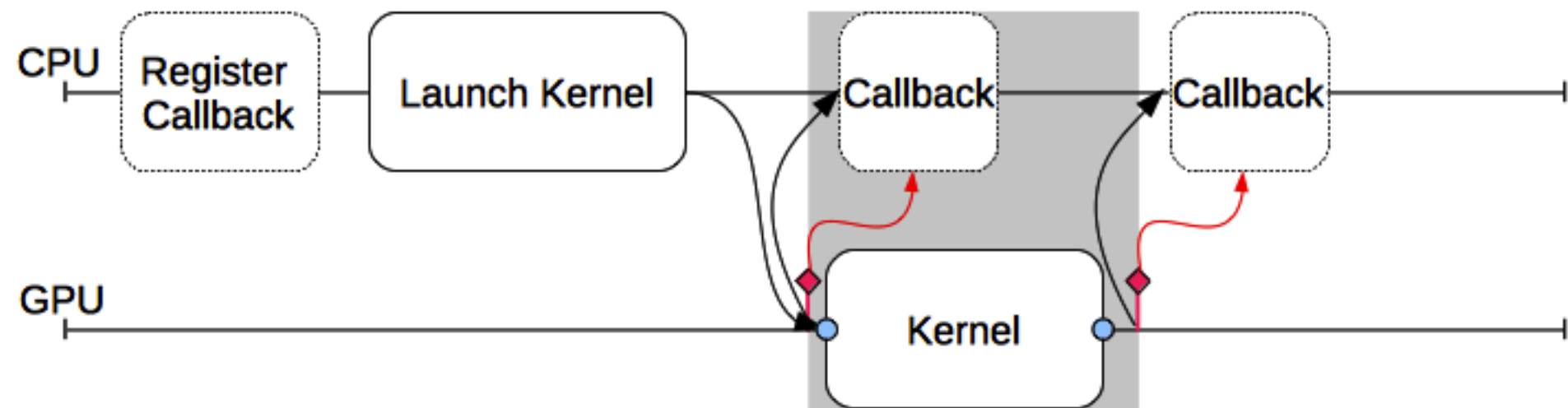


- Multi-CPU, Multi-GPU



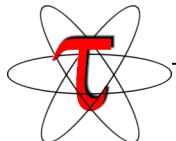
Host-GPU Measurement – Callback Method

- GPU driver libraries provide callbacks for certain routines and captures measurements
- Measurement tool registers the callbacks and processes performance data
- Application code is not modified



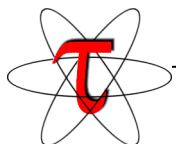
Method Support and Implementation

- Synchronous method
 - Place instrumentation appropriately around GPU calls (kernel launch, library routine, ...)
 - Wrap (synchronous) library with performance tool
- Event queue method
 - Utilize CUDA and OpenCL event support
 - Again, need instrumentation to create and insert events in the streams with kernel launch and process events
 - Can be implemented with driver library wrapping
- Callback method
 - Utilize language-level callback support in OpenCL
 - Utilize NVIDIA CUDA Performance Tool Interface (CUPTI)
 - Need to appropriately register callbacks

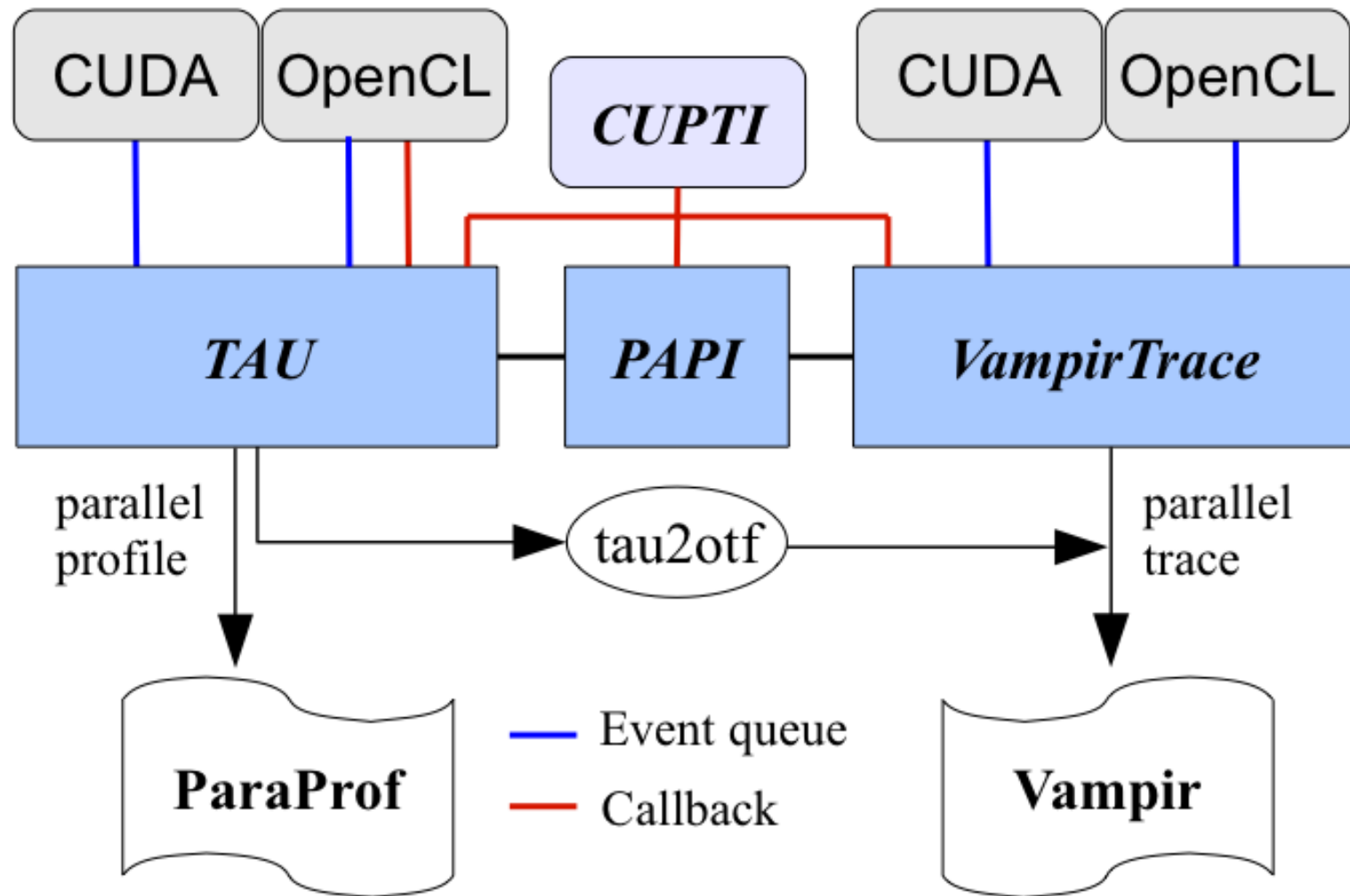


GPU Performance Measurement Tools

- Support the Host-GPU performance perspective
- Provide integration with existing measurement system to facilitate tool use
- Utilize support in GPU driver library and device
- Tools
 - TAU performance system
 - Vampir
 - PAPI
 - NVIDIA CUPTI

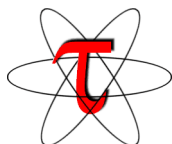


GPU Performance Tool Interoperability



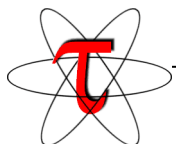
NVIDIA CUPTI

- NVIDIA is developing CUPTI to enable the creation of profiling and tracing tools
- Callback API
 - Interject tool code at the entry and exist to each CUDA runtime and driver API call
- Counter API
 - Query, configure, start, stop, and read the counters on CUDA-enabled devices
- CUPTI is delivered as a dynamic library
- CUPTI is released with CUDA 4.0



TAU for Heterogeneous Measurement

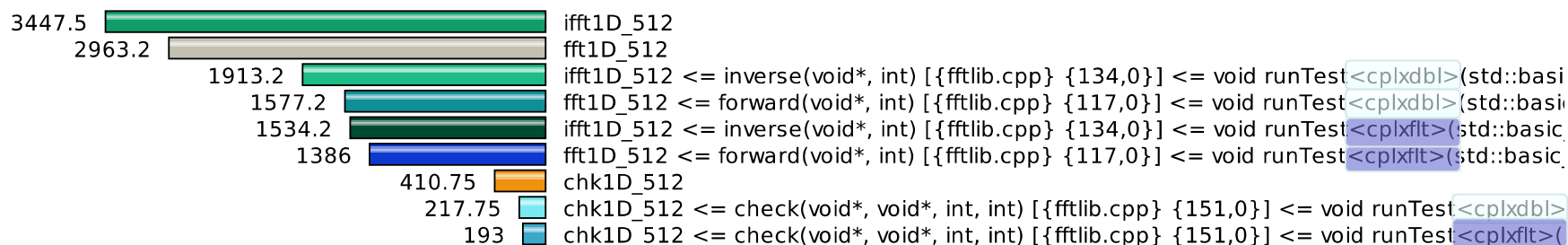
- Multiple performance perspectives
- Integrate Host-GPU support in TAU measurement framework
 - Enable use of each measurement approach
 - Include use of PAPI and CUPTI
 - Provide profiling and tracing support
- Tutorial
 - Use TAU library wrapping of libraries
 - Use `tau_exec` to work with binaries
 - % `./a.out` (uninstrumented)
 - % `tau_exec -T serial -cuda ./a.out`
 - % `paraprof`



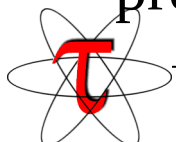
SHOC FFT Profile with Callsite Info

- TAU is able to associate callsite context information with kernel launch so that different kernel calls can be distinguished

Metric: TAUGPU_TIME
Value: Exclusive
Units: microseconds

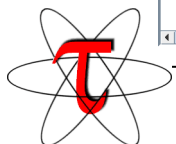
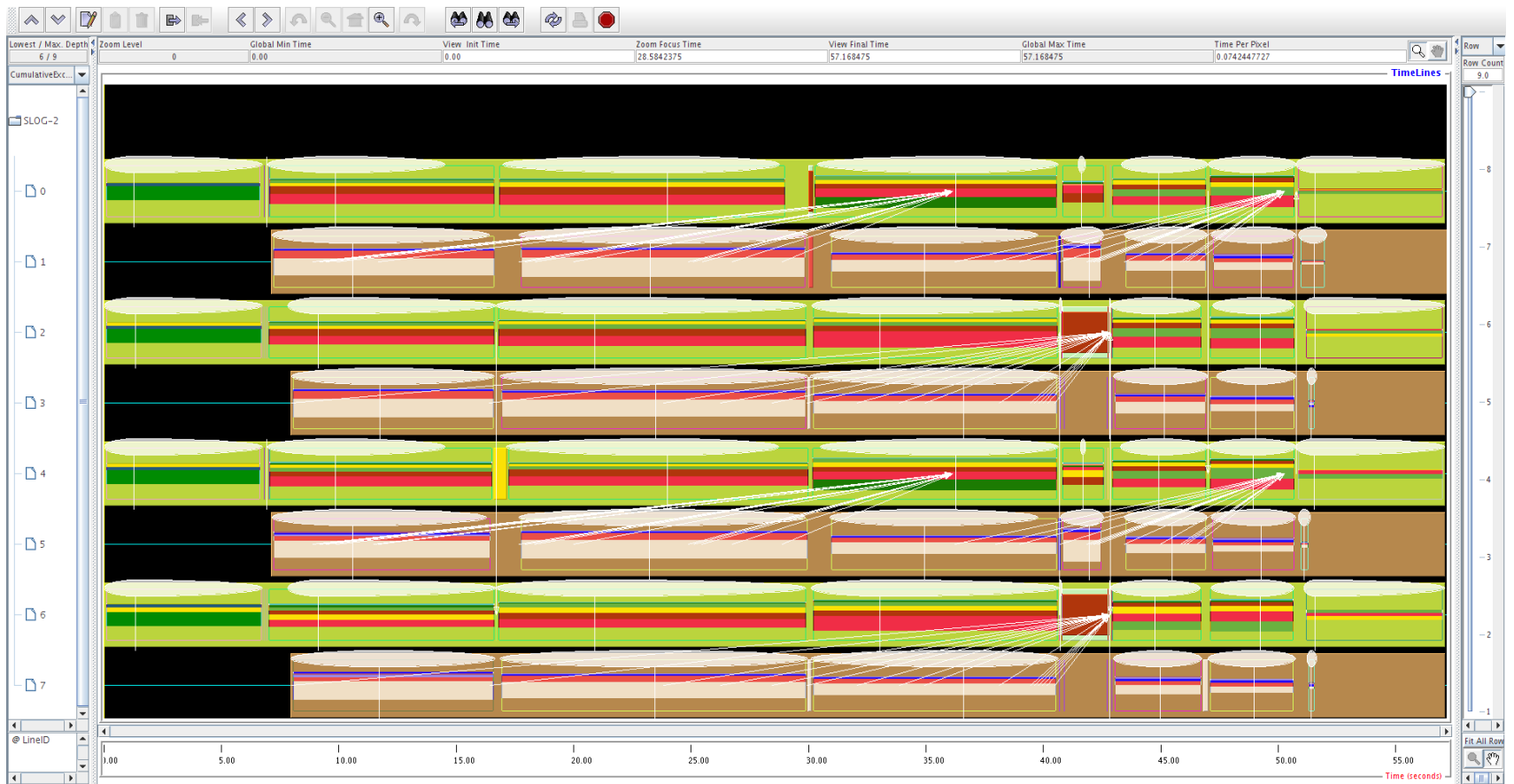


Each kernel (ifft1D_512, fft1D_512 and chk1D_512) is broken down by call-site, either during the **single** precession or **double** precession step.

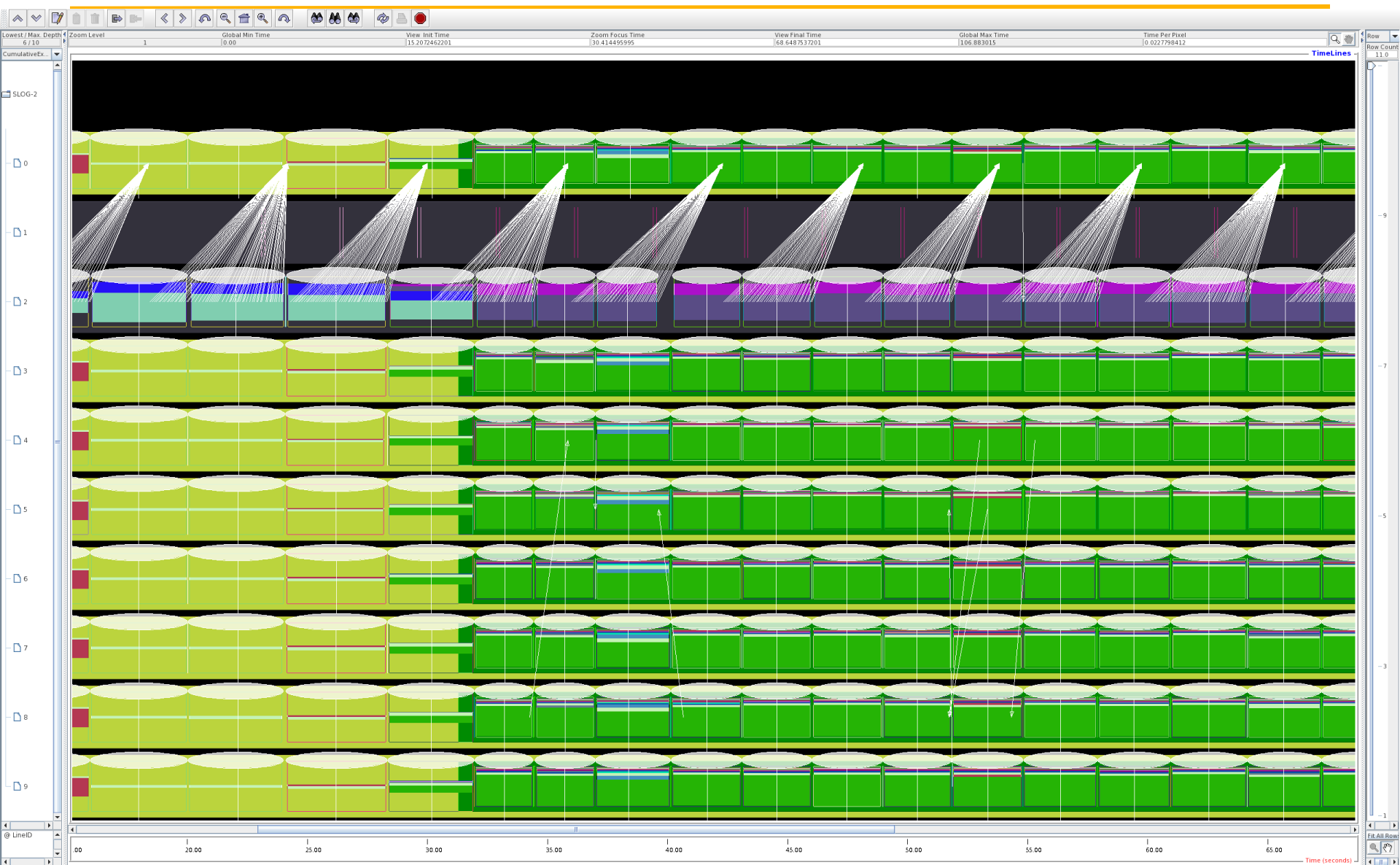


Example: CUDA Linpack

- TAU traces with Jumpshot visualization

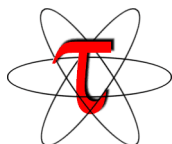


Example: NAMD with CUPTI

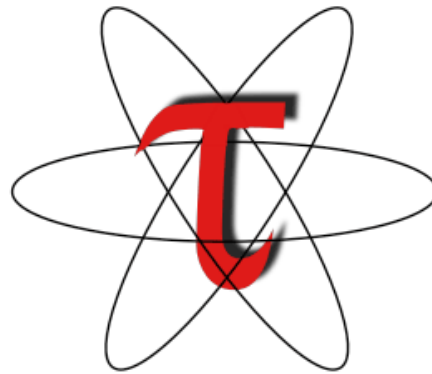


Profiling PGI Accelerator Primitives

- PGI compiler allows users to annotate source code to identify loops that should be accelerated
- When a program is compiled with TAU, its measurement library intercepts the PGI runtime library layer to measure time spent in the runtime library routines and data transfers
- TAU also captures the arguments:
 - array data dimensions and sizes, strides, upload and download times, variable names, source file names, row and column information, and routines



Custom profiling



Selective Instrumentation File

- Specify a list of routines to exclude or include (case sensitive)
- # is a wildcard in a routine name. It cannot appear in the first column.

```
BEGIN_EXCLUDE_LIST
Foo
Bar
D#EMM
END_EXCLUDE_LIST
```

- Specify a list of routines to include for instrumentation

```
BEGIN_INCLUDE_LIST
int main(int, char **)
F1
F3
END_INCLUDE_LIST
```

- Specify either an include list or an exclude list!



Selective Instrumentation File

- Optionally specify a list of files to exclude or include (case sensitive)

- * and ? may be used as wildcard characters in a file name

```
BEGIN_FILE_EXCLUDE_LIST
```

```
f*.f90
```

```
Foo?.cpp
```

```
END_FILE_EXCLUDE_LIST
```

- Specify a list of routines to include for instrumentation

```
BEGIN_FILE_INCLUDE_LIST
```

```
main.cpp
```

```
foo.f90
```

```
END_FILE_INCLUDE_LIST
```

Selective Instrumentation File

- User instrumentation commands are placed in INSTRUMENT section
- ? and * used as wildcard characters for file name, # for routine name
- \ as escape character for quotes
- Routine entry/exit, arbitrary code insertion
- Outer-loop level instrumentation

```
BEGIN_INSTRUMENT_SECTION
loops file="foo.f90" routine="matrix#"
memory file="foo.f90" routine="#"
io routine="matrix#"
[static/dynamic] phase routine="MULTIPLY"
dynamic [phase/timer] name="foo" file="foo.cpp" line=22 to line=35
file="foo.f90" line = 123 code = "  print *, \" Inside foo\""
exit routine = "int foo()" code = "cout <<\"exiting foo\"<<endl;"
END_INSTRUMENT_SECTION
```

Instrumentation Specification

```
% tau_instrumentor
```

```
Usage : tau_instrumentor <pdbfile> <sourcefile> [-o <outputfile>] [-noinline]  
[-g groupname] [-i headerfile] [-c|-c++|-fortran] [-f <instr_req_file> ]
```

```
For selective instrumentation, use -f option
```

```
% tau_instrumentor foo.pdb foo.cpp -o foo.inst.cpp -f selective.dat
```

```
% cat selective.dat
```

```
# Selective instrumentation: Specify an exclude/include list of routines/files.
```

```
BEGIN_EXCLUDE_LIST
```

```
void quicksort(int *, int, int)
```

```
void sort_5elements(int *)
```

```
void interchange(int *, int *)
```

```
END_EXCLUDE_LIST
```

```
BEGIN_FILE_INCLUDE_LIST
```

```
Main.cpp
```

```
Foo?.c
```

```
*.C
```

```
END_FILE_INCLUDE_LIST
```

```
# Instruments routines in Main.cpp, Foo?.c and *.C files only
```

```
# Use BEGIN_[FILE]_INCLUDE_LIST with END_[FILE]_INCLUDE_LIST
```

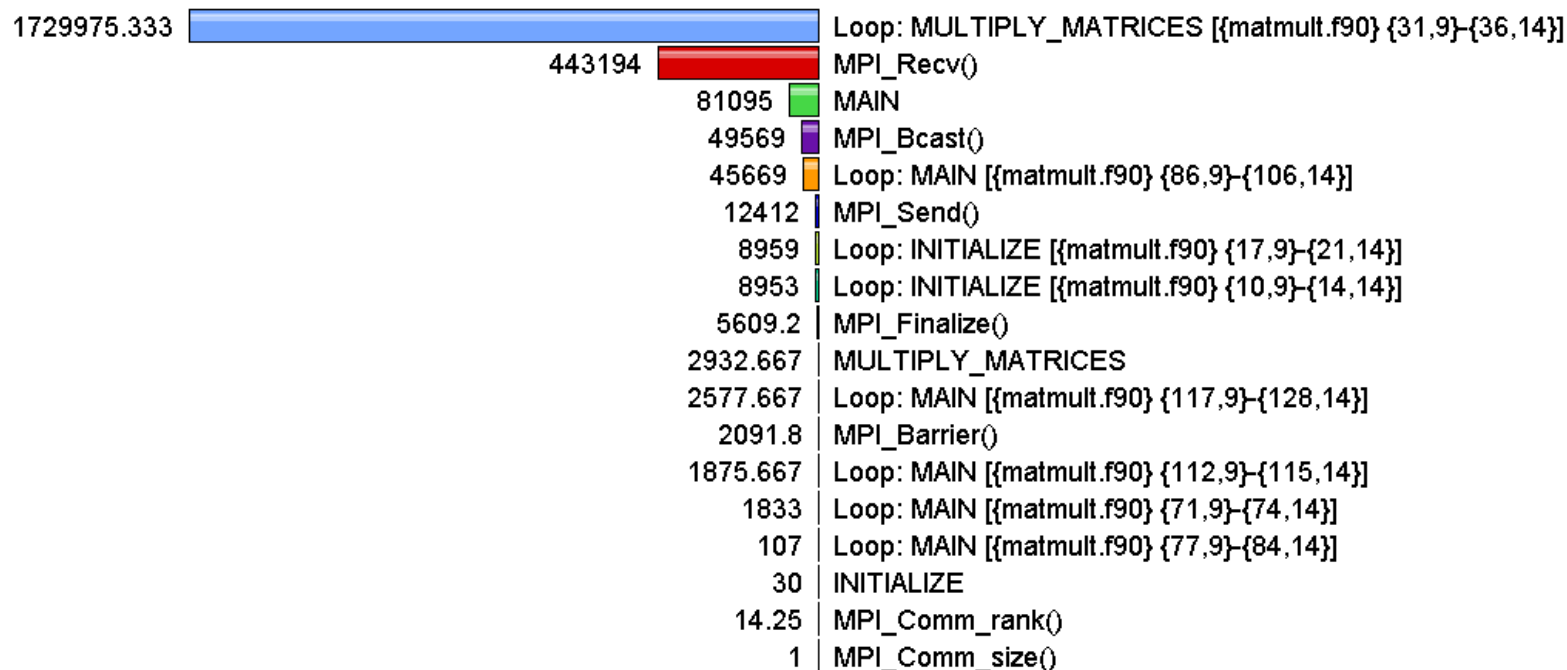
Usage Scenarios: Loop Level Instrumentation

- 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



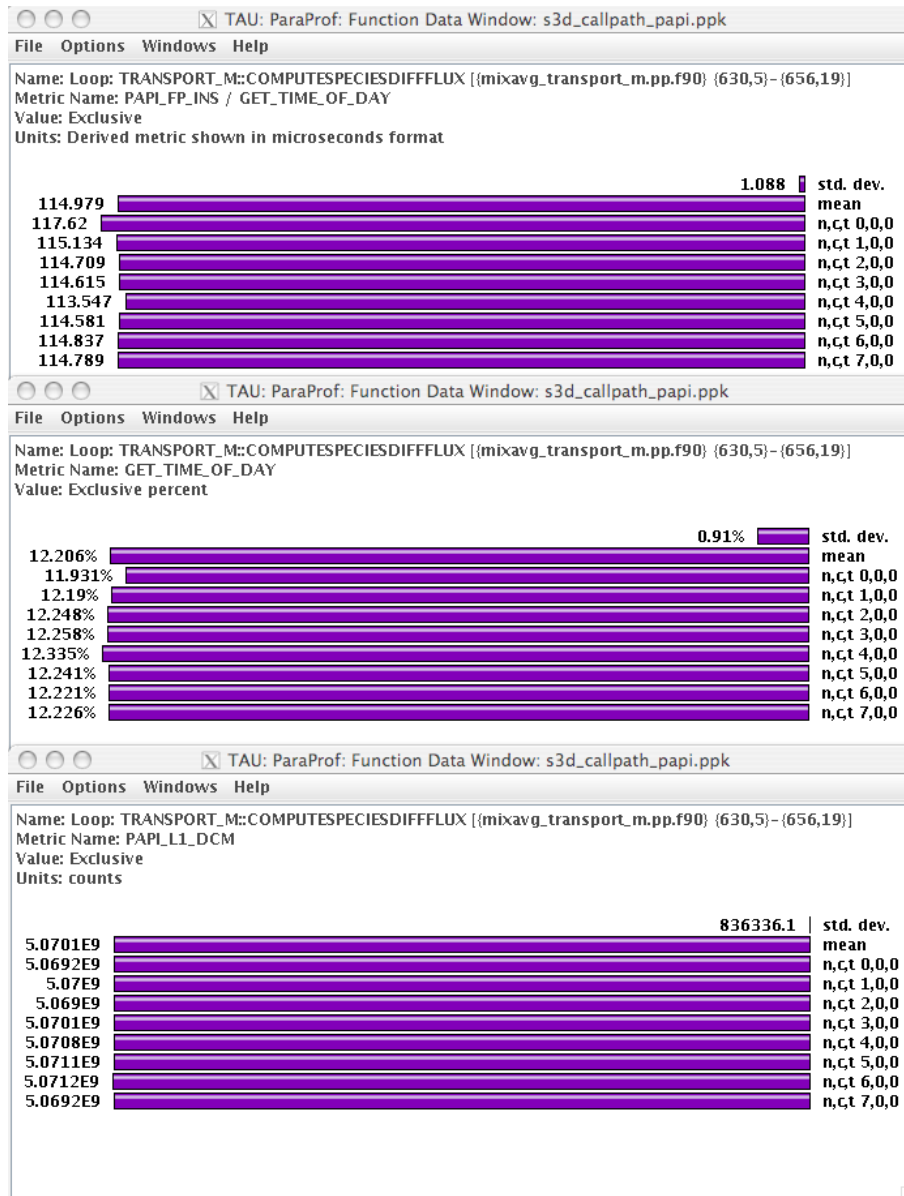
Solution: Generating a loop level profile

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

% export PATH=$TAUROOTDIR/x86_64/bin:$PATH
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% mpirun -np 8 ./a.out
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.

% paraprof app.ppk
```

ParaProf's Source Browser: Loop Level Instrumentation



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 subroutine computeStressTensor( grad_u)
668
669

```

Manually Instrumenting a C code

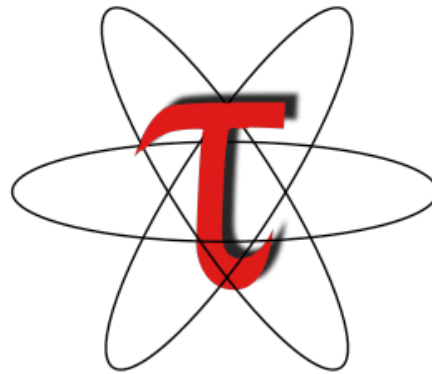
```
#include <TAU.h>
int foo(int x) {
    TAU_START("foo");
    for (i = 0; i < x; i++) { // do work
    }
    TAU_STOP("foo");
}

int main(int argc, char **argv) {
    TAU_INIT(&argc, &argv);
    TAU_START("main");
    TAU_PROFILE_SET_NODE(rank);
    ...
    TAU_STOP("main");
}

% gcc -I<taudir>/include foo.c -o foo -L<taudir>/<arch>/lib -lTAU
% ./a.out
% pprof; paraprof
```

NOTE: Replace TAU_START("foo") with call TAU_START('foo')
in Fortran. See <taudir>/include/TAU.h for full API.

Throttling effect of frequently called small routines



Optimization of Program Instrumentation

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

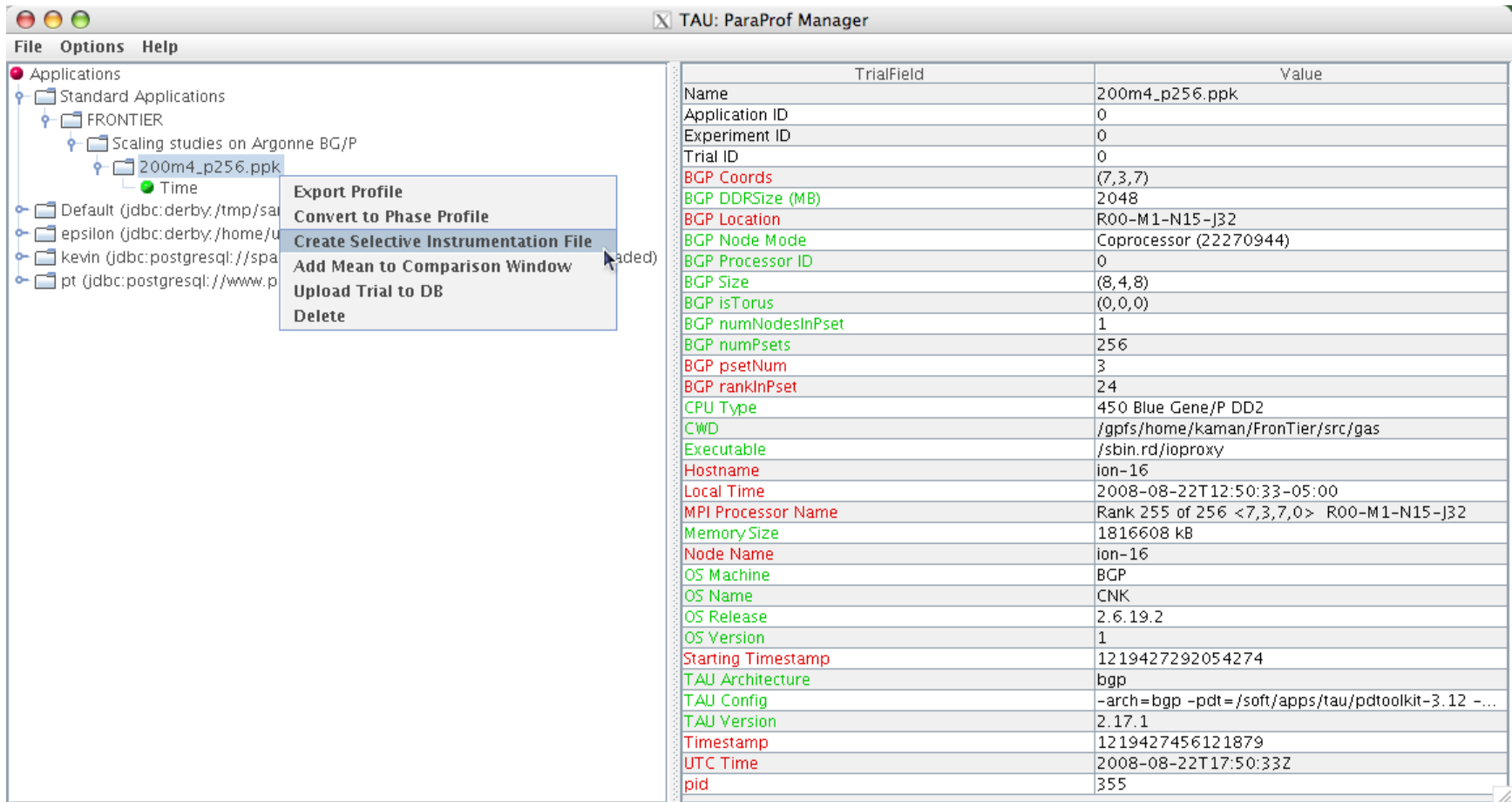
```
% export TAU_THROTTLE=1
```

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). Use TAU_THROTTLE=0 to disable.
- Selective instrumentation file to filter events

```
% tau_instrumentor [options] -f <file> OR  
% export TAU_OPTIONS='-optTauSelectFile=tau.txt'
```
- Compensation of local instrumentation overhead

```
% export TAU_COMPENSATE=1 (in tau-2.19.2+)
```

ParaProf: Creating Selective Instrumentation File



TAU: ParaProf Manager

File Options Help

Applications

- Standard Applications
 - FRONTIER
 - Scaling studies on Argonne BG/P
 - 200m4_p256.ppk
 - Time
- Default (jdbc:derby:/tmp/sa
- epsilon (jdbc:derby:/home/u
- kevin (jdbc:postgresql://spa
- pt (jdbc:postgresql://www.p

Export Profile

Convert to Phase Profile

Create Selective Instrumentation File

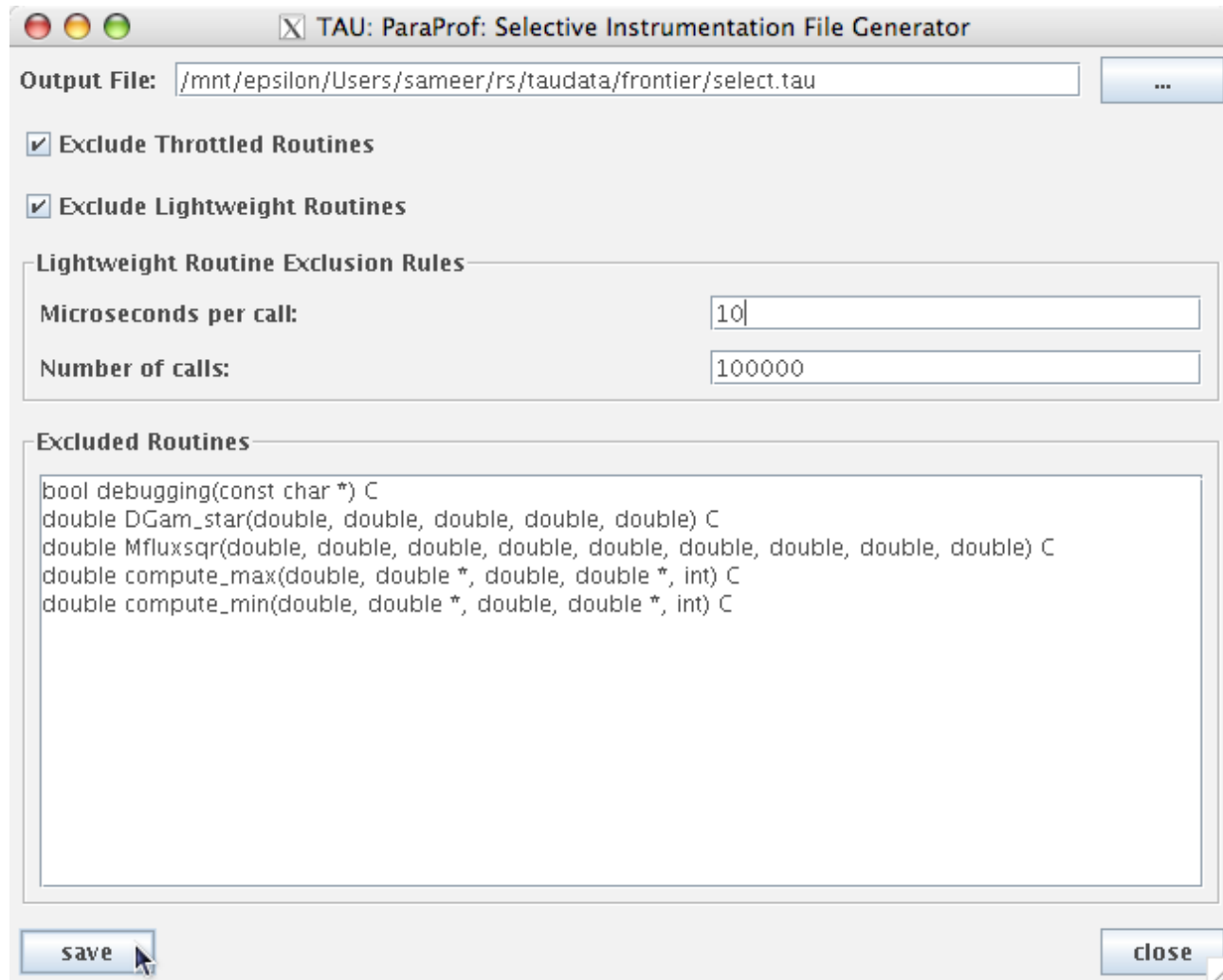
Add Mean to Comparison Window

Upload Trial to DB

Delete

TrialField	Value
Name	200m4_p256.ppk
Application ID	0
Experiment ID	0
Trial ID	0
BGP Coords	(7,3,7)
BGP DDRSize (MB)	2048
BGP Location	R00-M1-N15-J32
BGP Node Mode	Coprocessor (22270944)
BGP Processor ID	0
BGP Size	(8,4,8)
BGP isTorus	(0,0,0)
BGP numNodesInPset	1
BGP numPsets	256
BGP psetNum	3
BGP rankInPset	24
CPU Type	450 Blue Gene/P DD2
CWD	/gpfs/home/kaman/FronTier/src/gas
Executable	/sbin.rd/ioproxy
Hostname	ion-16
Local Time	2008-08-22T12:50:33-05:00
MPI Processor Name	Rank 255 of 256 <7,3,7,0> R00-M1-N15-J32
Memory Size	1816608 kB
Node Name	ion-16
OS Machine	BGP
OS Name	CNK
OS Release	2.6.19.2
OS Version	1
Starting Timestamp	1219427292054274
TAU Architecture	bgp
TAU Config	-arch=bgp -pdt=/soft/apps/tau/pdtoolkit-3.12 -...
TAU Version	2.17.1
Timestamp	1219427456121879
UTC Time	2008-08-22T17:50:33Z
pid	355

Choosing Rules for Excluding Routines



A screenshot of a macOS-style dialog box titled "TAU: ParaProf: Selective Instrumentation File Generator". The dialog has a title bar with standard window controls (red, yellow, green buttons) and a close button (X). The main content area includes an "Output File:" label followed by a text field containing the path "/mnt/epsilon/Users/sameer/rs/taudata/frontier/select.tau" and a browse button "...". Below this are two checked checkboxes: "Exclude Throttled Routines" and "Exclude Lightweight Routines". A section titled "Lightweight Routine Exclusion Rules" contains two input fields: "Microseconds per call:" with the value "10" and "Number of calls:" with the value "100000". At the bottom of this section is a large text area titled "Excluded Routines" containing the following C code snippets:

```
bool debugging(const char *) C
double DGam_star(double, double, double, double, double) C
double Mfluxsqr(double, double, double, double, double, double, double, double, double) C
double compute_max(double, double *, double, double *, int) C
double compute_min(double, double *, double, double *, int) C
```

 At the bottom of the dialog are two buttons: "save" and "close".

TAU: ParaProf: Selective Instrumentation File Generator

Output File: /mnt/epsilon/Users/sameer/rs/taudata/frontier/select.tau ...

☒ Exclude Throttled Routines

☒ Exclude Lightweight Routines

Lightweight Routine Exclusion Rules

Microseconds per call: 10

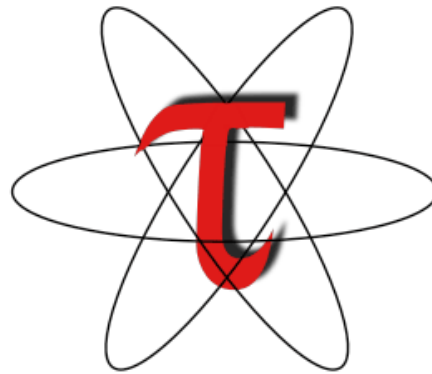
Number of calls: 100000

Excluded Routines

```
bool debugging(const char *) C
double DGam_star(double, double, double, double, double) C
double Mfluxsqr(double, double, double, double, double, double, double, double, double) C
double compute_max(double, double *, double, double *, int) C
double compute_min(double, double *, double, double *, int) C
```

save close

Observing I/O and Memory Activity



Library interposition/wrapping: `tau_exec`

- 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)
- 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



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



TAU Execution Command (tau_exec)

- Uninstrumented execution
 - % mpirun -np 256 ./a.out
- Track MPI performance
 - % mpirun -np 256 **tau_exec** ./a.out
- Track I/O and MPI performance (MPI enabled by default)
 - % mpirun -np 256 **tau_exec** **-io** ./a.out
- Track memory operations
 - % setenv TAU_TRACK_MEMORY_LEAKS 1
 - % mpirun -np 256 **tau_exec** **-memory** ./a.out
- Track I/O performance and memory operations
 - % mpirun -np 256 **tau_exec** **-io** **-memory** ./a.out
- **Track GPGPU operations**
 - % mpirun -np 256 **tau_exec** **-cuda** ./a.out
- **Collect events from callstack sampling**
 - %mpirun -np 256 **tau_exec** **-ebs** ./a.out



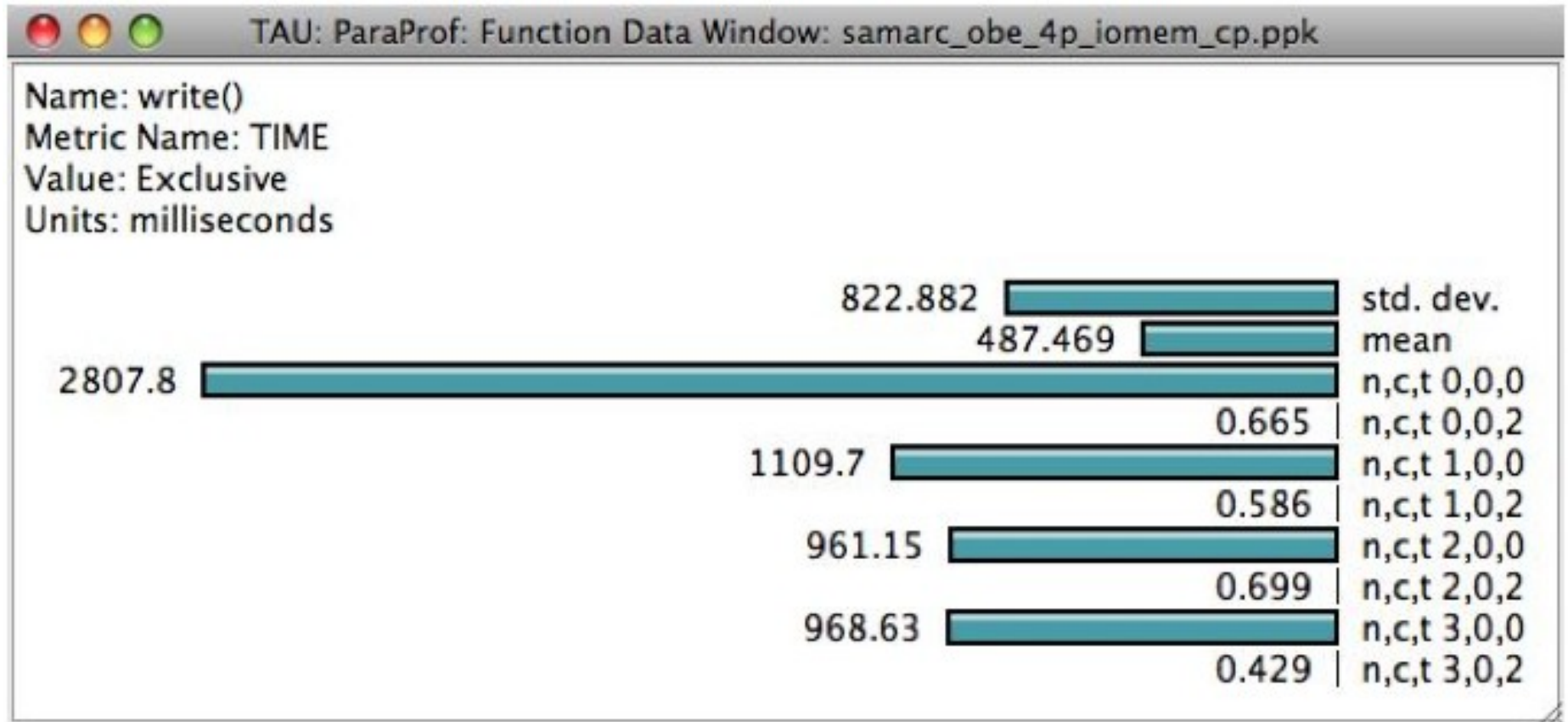
I/O Calls Supported

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
writew	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							
Name		Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
.TAU application							
MPI_Finalize0							
MPI_Init0							
fscanf0							
read0							
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()

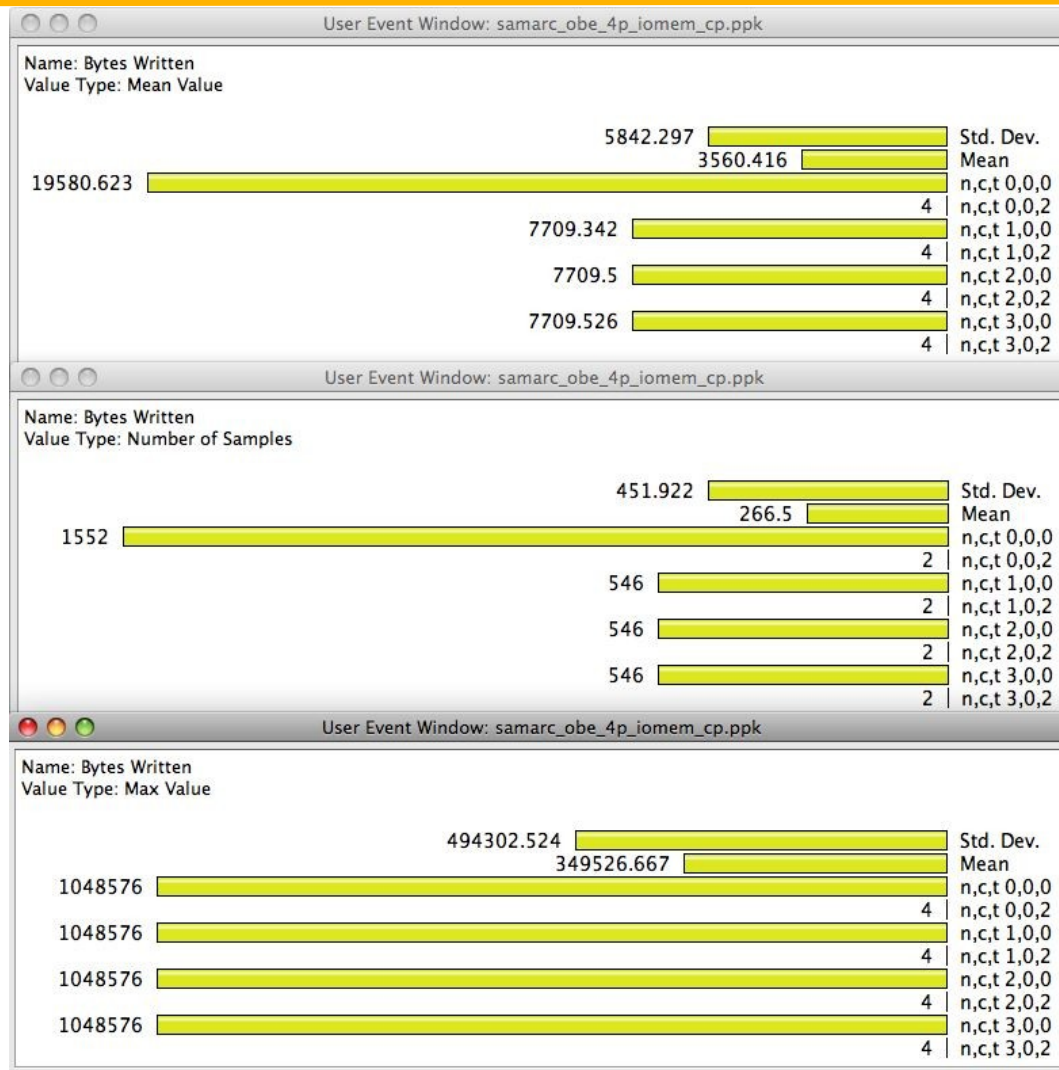


Volume of I/O by File, Memory

TAU: ParaProf: Context Events for thread: n,c,t, 1,0,0 - samarc_obc_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()						



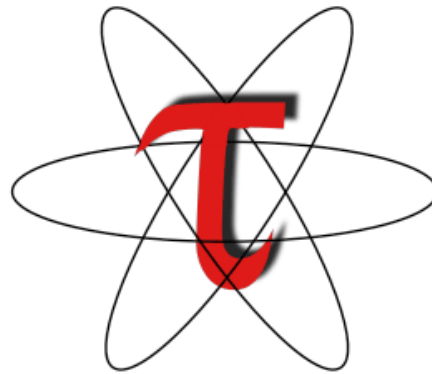
Bytes Written



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()						

Generating event traces

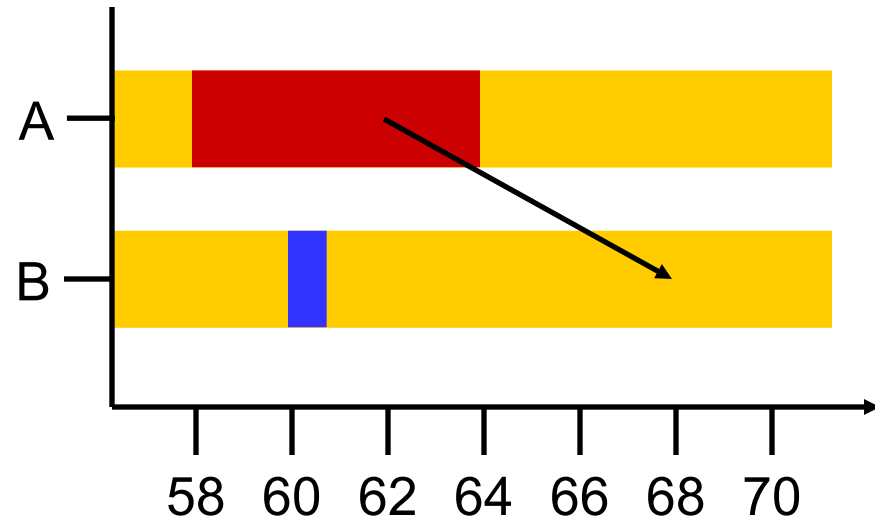


Tracing Analysis and Visualization

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
...			



Profiling / Tracing Comparison

- Profiling
 - ☺ Finite, bounded performance data size
 - ☺ Applicable to both direct and indirect methods
 - ☹ Loses time dimension (not entirely)
 - ☹ Lacks ability to fully describe process interaction
- Tracing
 - ☺ Temporal and spatial dimension to performance data
 - ☺ Capture parallel dynamics and process interaction
 - ☹ Some inconsistencies with indirect methods
 - ☹ Unbounded performance data size (large)
 - ☹ Complex event buffering and clock synchronization

Trace Formats

- Different tools produce different formats
 - Differ by event types supported
 - Differ by ASCII and binary representations
 - Vampir Trace Format (VTF)
 - KOJAK/Scalasca (EPILOG)
 - Jumpshot (SLOG-2)
 - Paraver
- Open Trace Format (OTF)
 - Supports interoperoperation between tracing tools

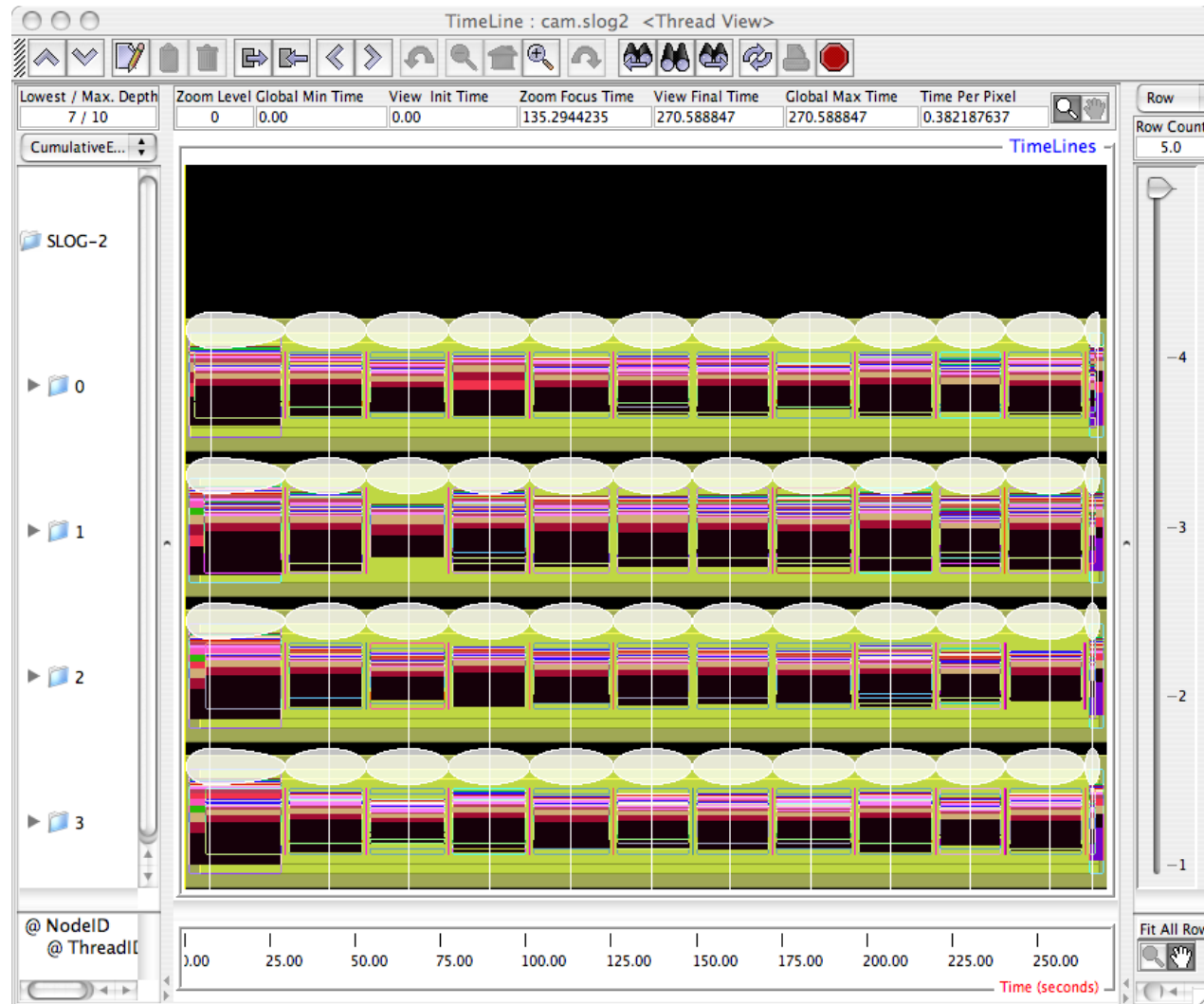
Generate a Trace File

```
% export TAU_MAKEFILE=$TAULIBDIR/Makefile.tau-mpi-pdt-pgi
% export PATH=$TAUROOTDIR/x86_64/bin:$PATH
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% qsub -I -l nodes=1:ppn=8 -X
% export TAU_TRACE=1
% mpirun -np 8 ./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
PARAVER:
% tau_convert -paraver tau.trc tau.edf app.prv
% paraver app.prv
```

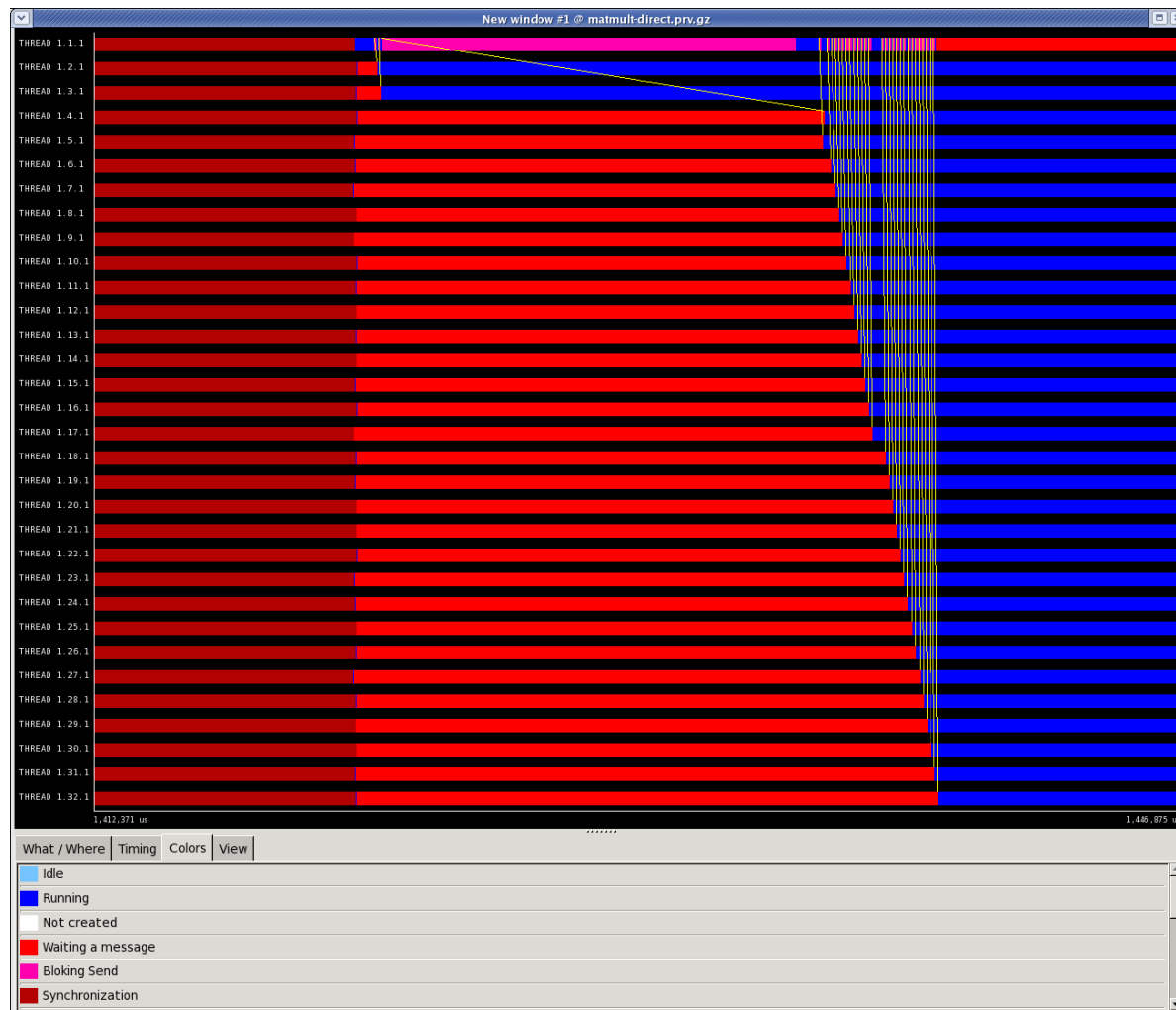
Jumpshot

- <http://www-unix.mcs.anl.gov/perfvis/software/viewers/index.htm>
- Developed at Argonne National Laboratory as part of the MPICH project
 - Also works with other MPI implementations
 - Jumpshot is bundled with the TAU package
- Java-based tracefile visualization tool for postmortem performance analysis of MPI programs
- Latest version is Jumpshot-4 for SLOG-2 format
 - Scalable level of detail support
 - Timeline and histogram views
 - Scrolling and zooming
 - Search/scan facility

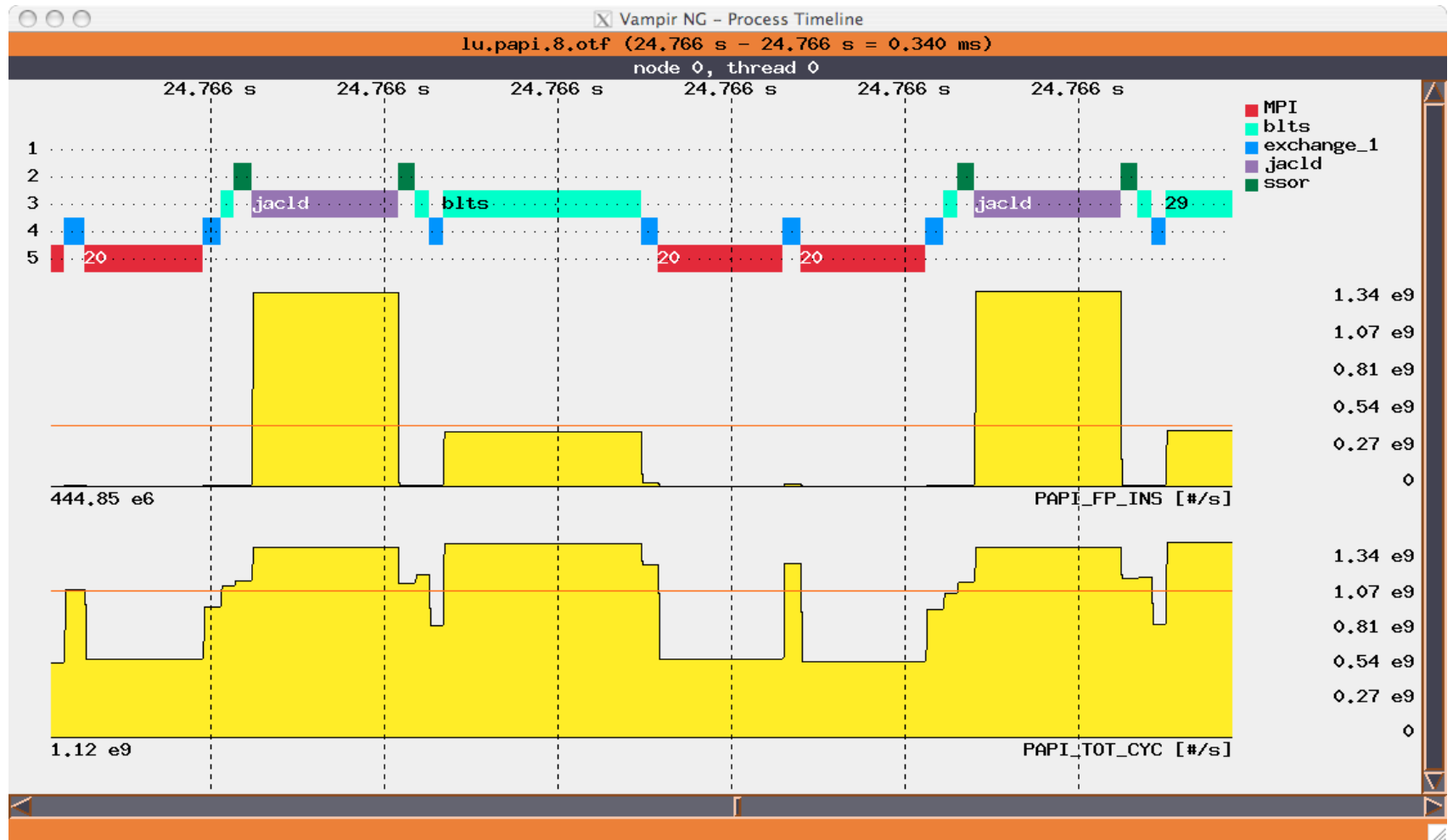
Jumpshot



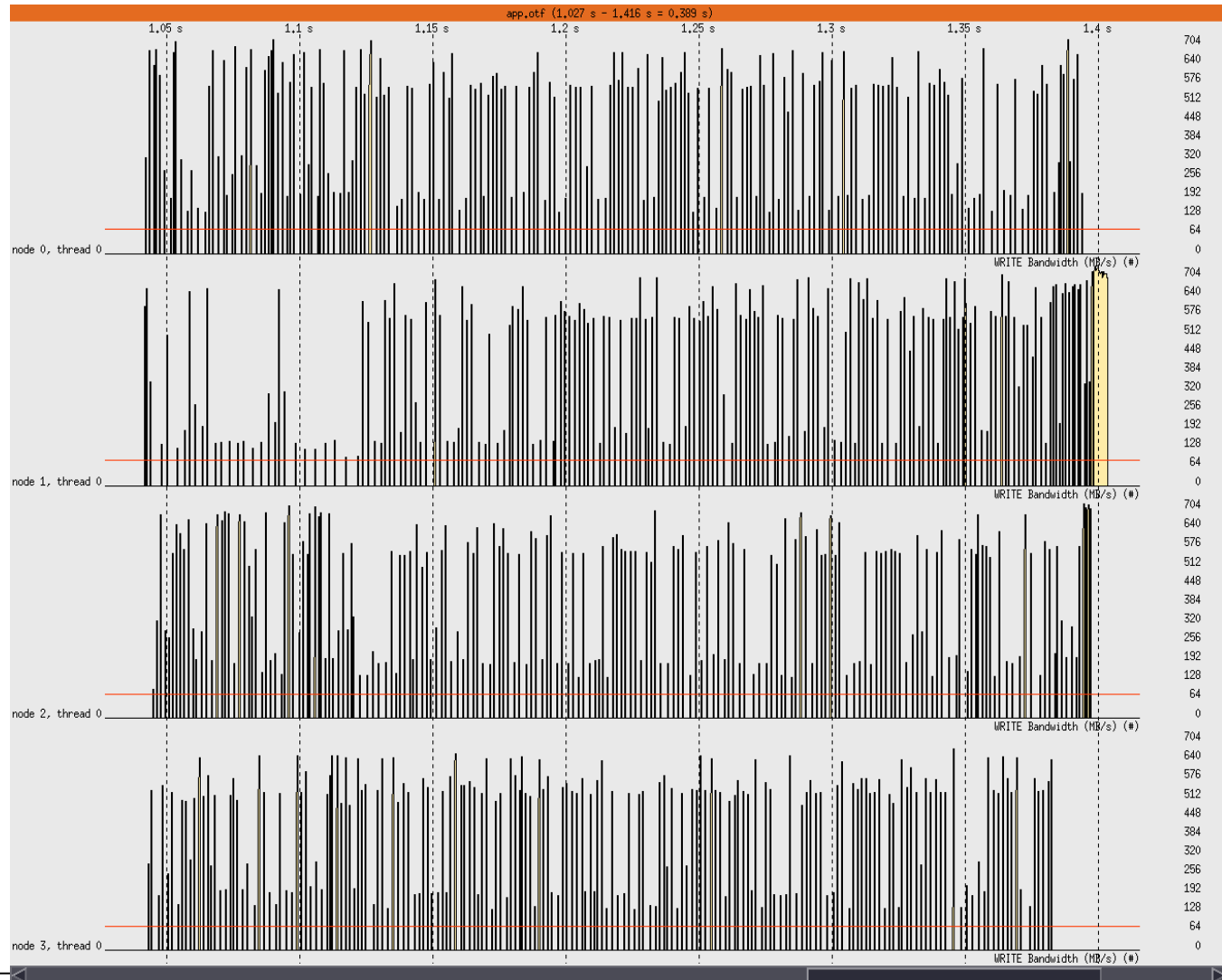
ParaVer [<http://www.bsc.es/paraver>]



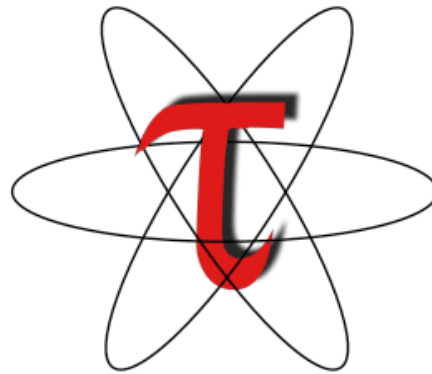
VNG Process Timeline with PAPI Counters



Vampir Counter Timeline Showing I/O BW



Running the application, generation of performance data



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 (for use with tau_exec -memory)
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_SAMPLING	1	Generates sample based profile
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. "snapshot" generates xml format
TAU_METRICS	TIME	Setting to a comma separated list generates other metrics. (e.g., TIME:linuxtimers:PAPI_FP_OPS:PAPI_NATIVE_<event>)



Generate a Callpath Profile

```
% export TAU_MAKEFILE=$TAULIBDIR/Makefile.tau-mpi-pdt
% export PATH=$TAUROOTDIR/x86_64/bin:$PATH
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)

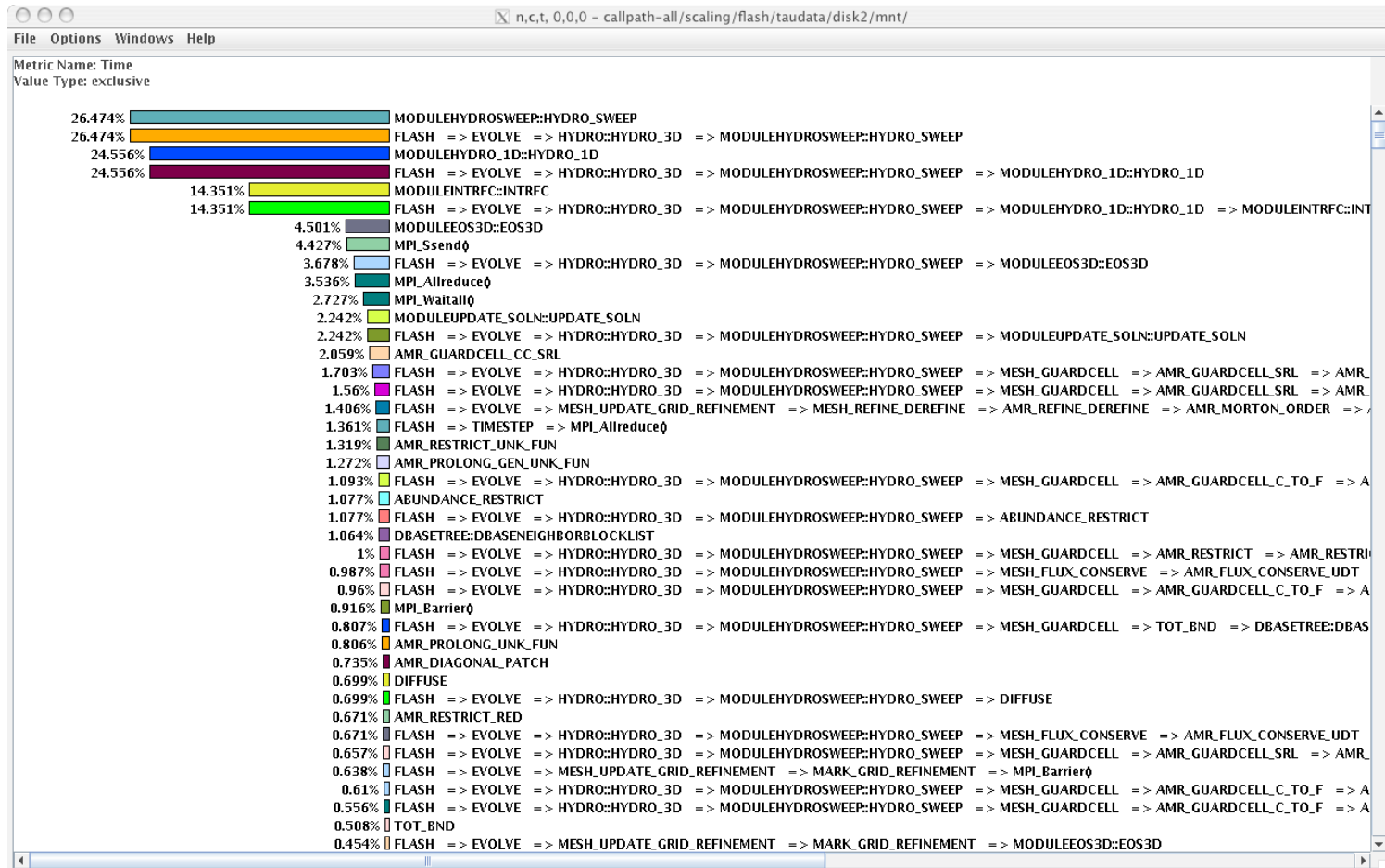
% qsub -I -l nodes=1:ppn=8 -X
% export TAU_CALLPATH=1
% export TAU_CALLPATH_DEPTH=100

% mpirun -np 8 ./a.out
% paraprof --pack app.ppk
    Move the app.ppk file to your desktop.
% paraprof app.ppk
(Windows -> Thread -> Call Graph)
```



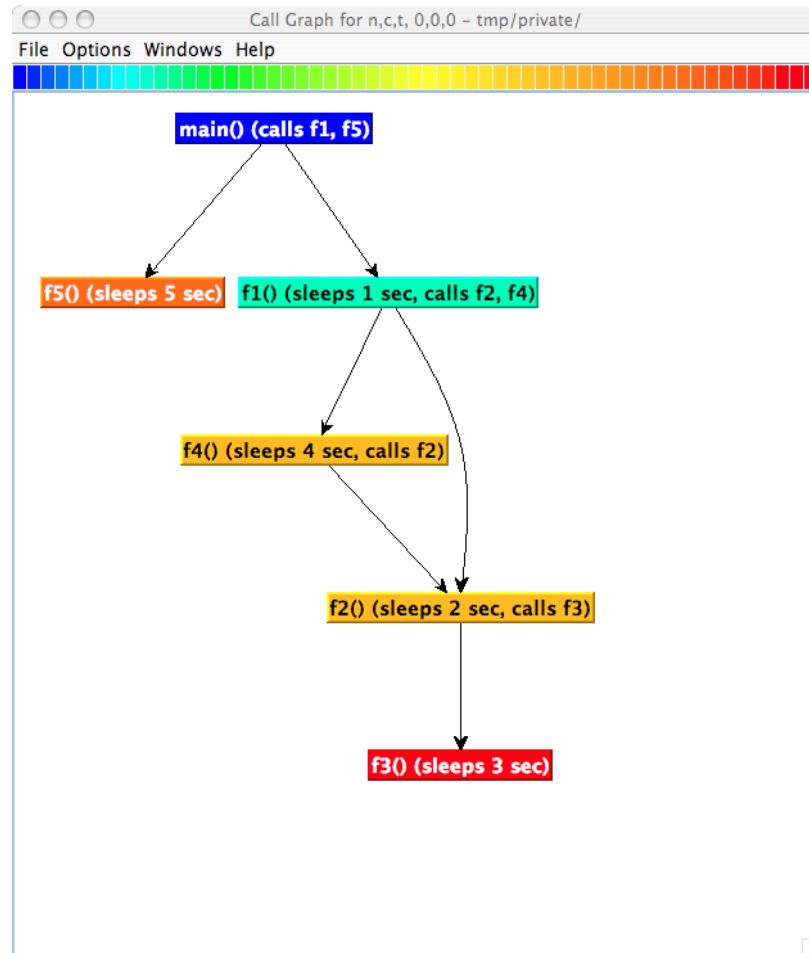
Usage Scenarios: Generating Callpath Profile

- Callpath profile for a given callpath depth:



Callpath Profile

- Generates program callgraph



Communication Matrix

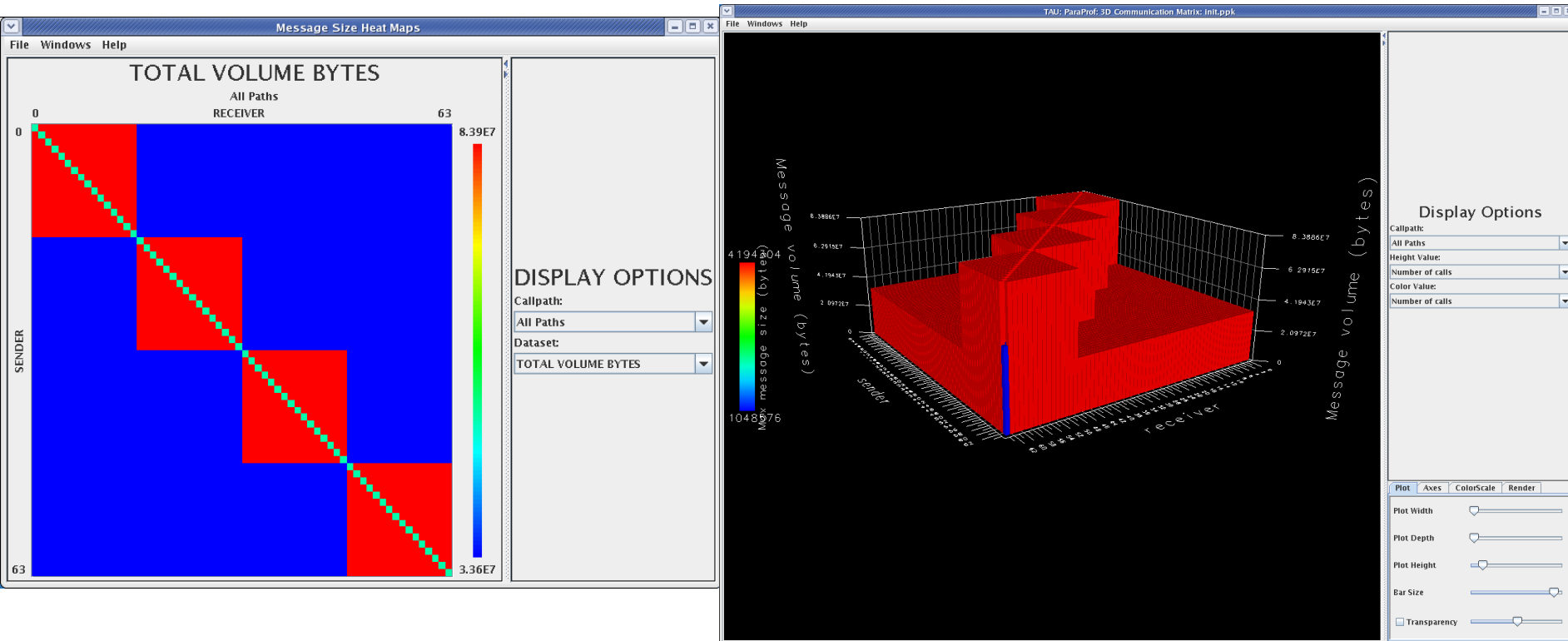
```
% export TAU_MAKEFILE=$TAULIBDIR/Makefile.tau-mpi-pdt
% export PATH=$TAUROOTDIR/x86_64/bin:$PATH
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)

% qsub -I -l nodes=1:ppn=8 -X
% export TAU_COMM_MATRIX=1
% mpirun -np 8 ./a.out (setting the environment variables)

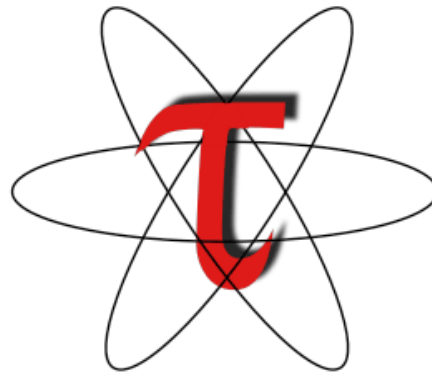
% paraprof
(Windows -> Communication Matrix)
```



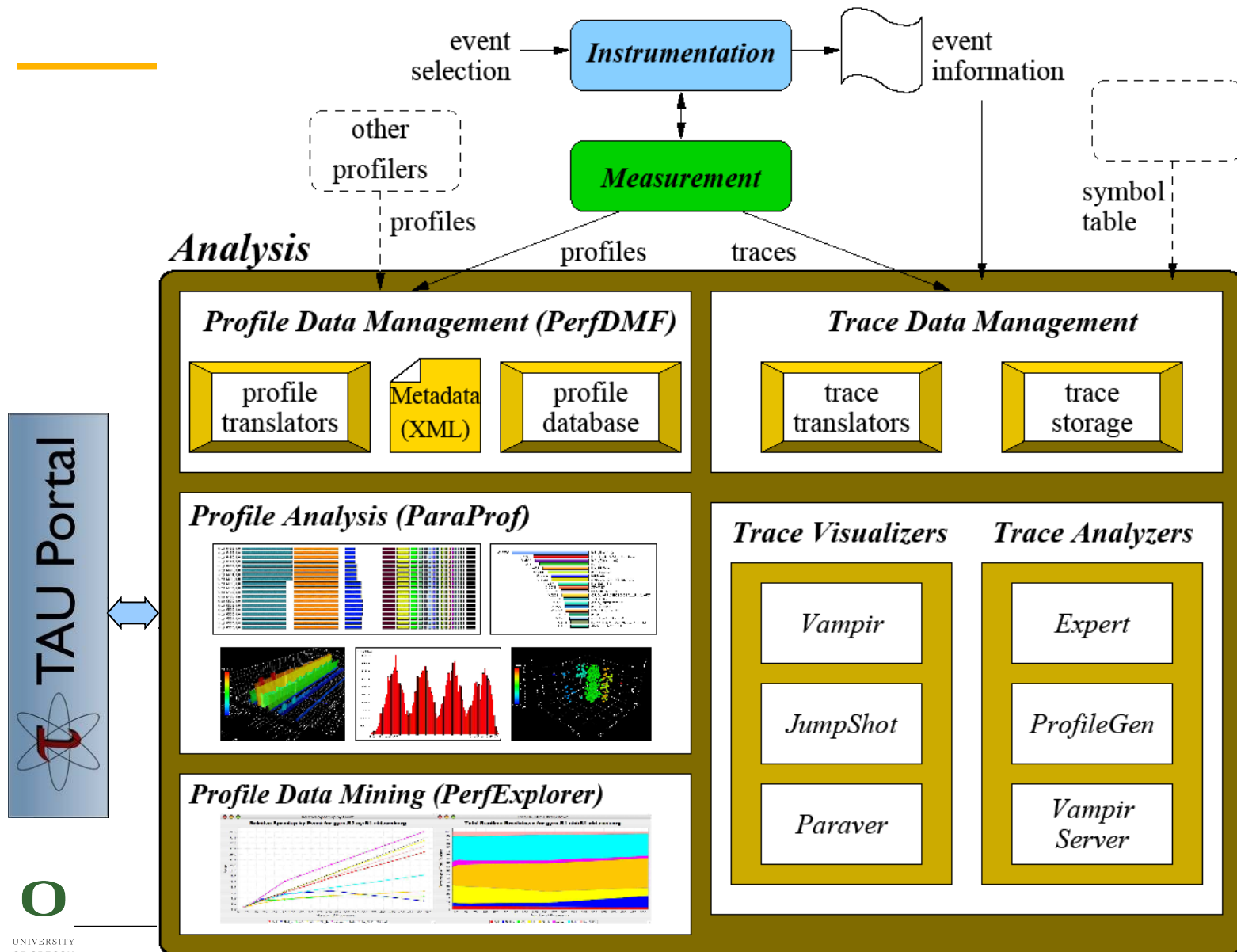
ParaProf: Communication Matrix Display



Analyzing performance data with ParaProf, PerfExplorer

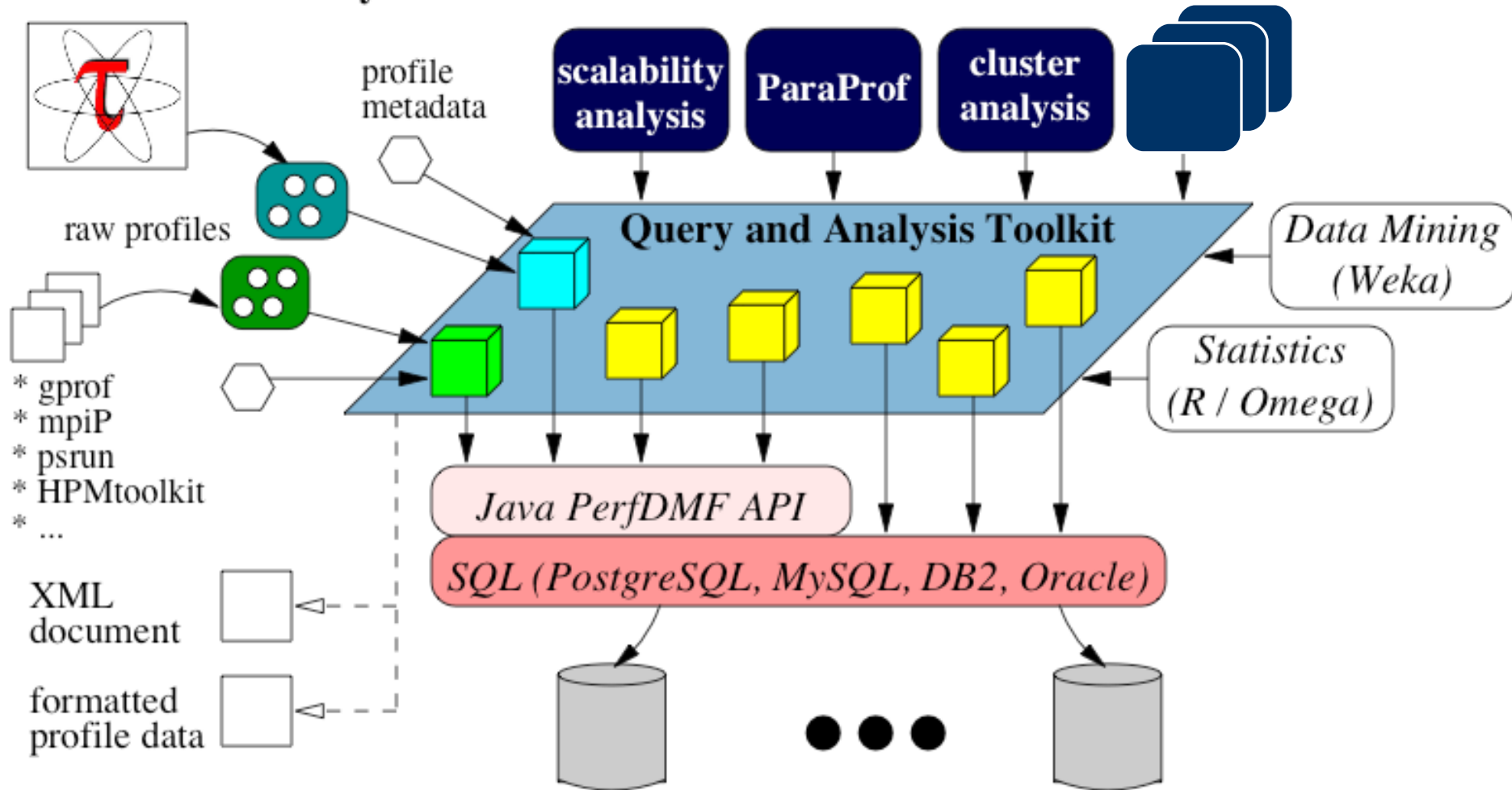


TAU Performance System Architecture

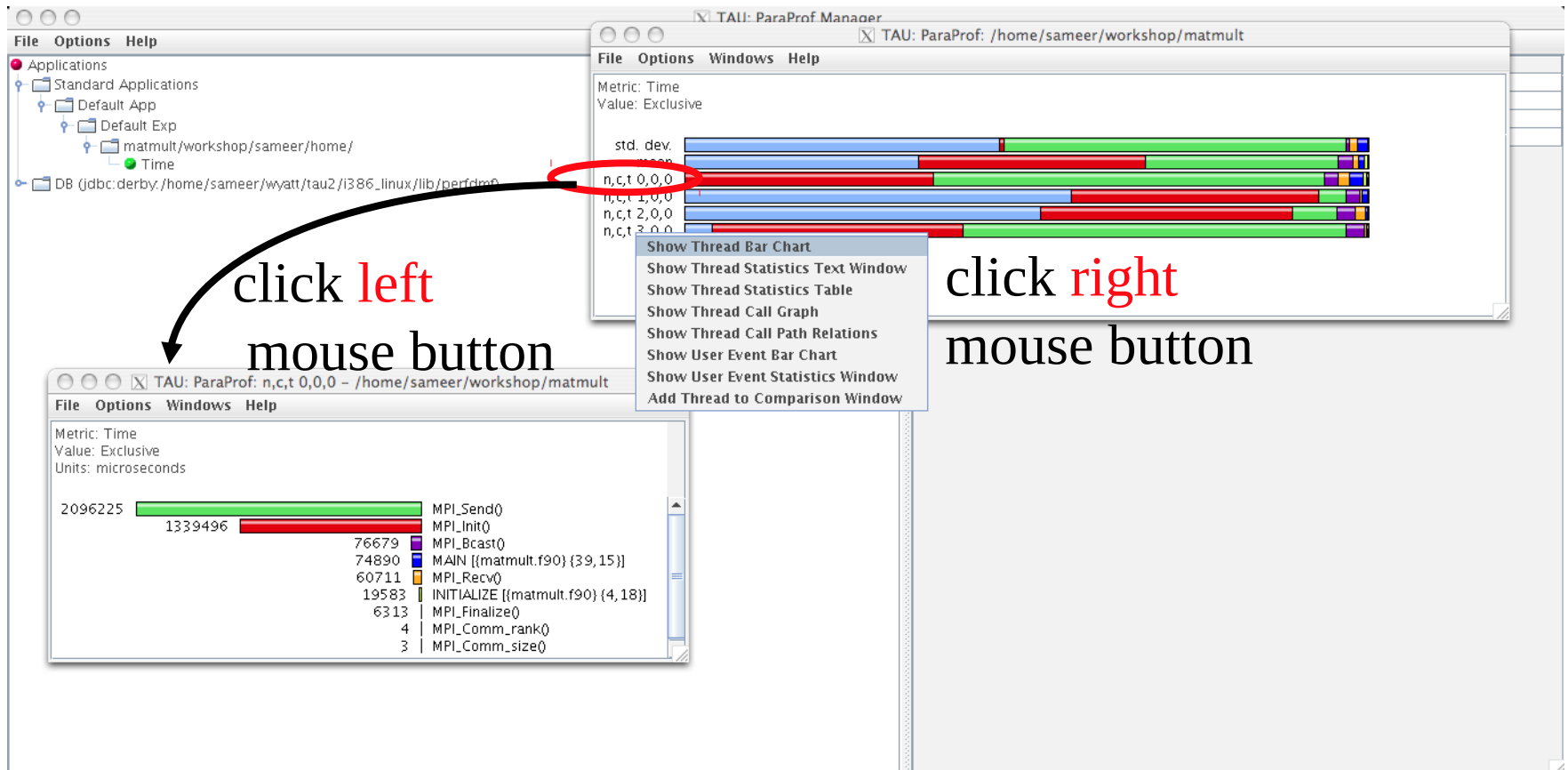


PerfDMF: Performance Data Mgmt. Framework

TAU Performance System



ParaProf Main Window



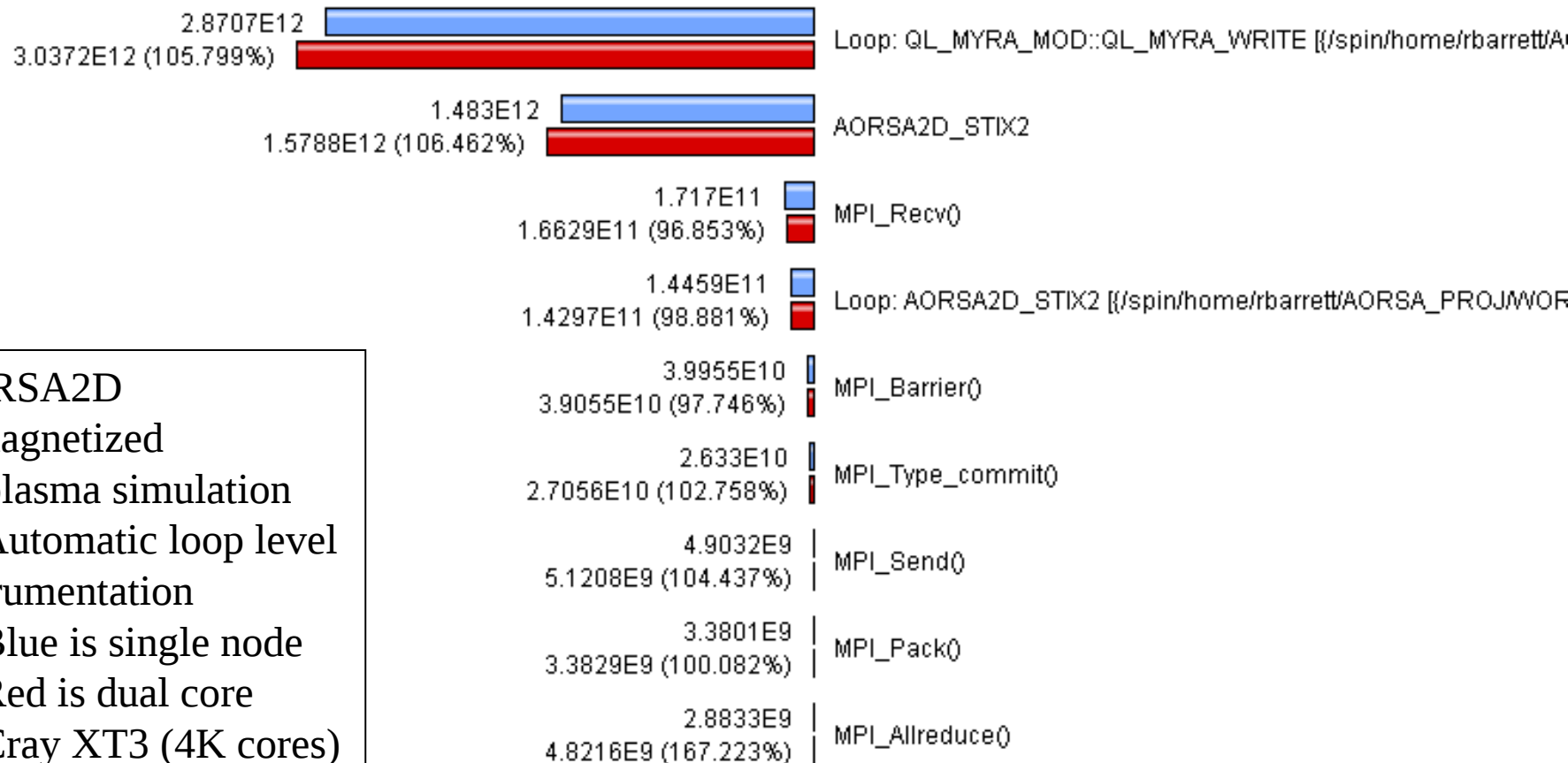
% paraprof matmult.ppk



Comparing Effects of Multi-Core Processors

Metric: PAPI_RES_STL
 Value: Exclusive
 Units: counts

C:\iter.350x350.4096pes.sn.loops.BARRIER.ppk - Mean
 C:\iter.350x350.2048pes.dc.loops.BARRIER.ppk - Mean

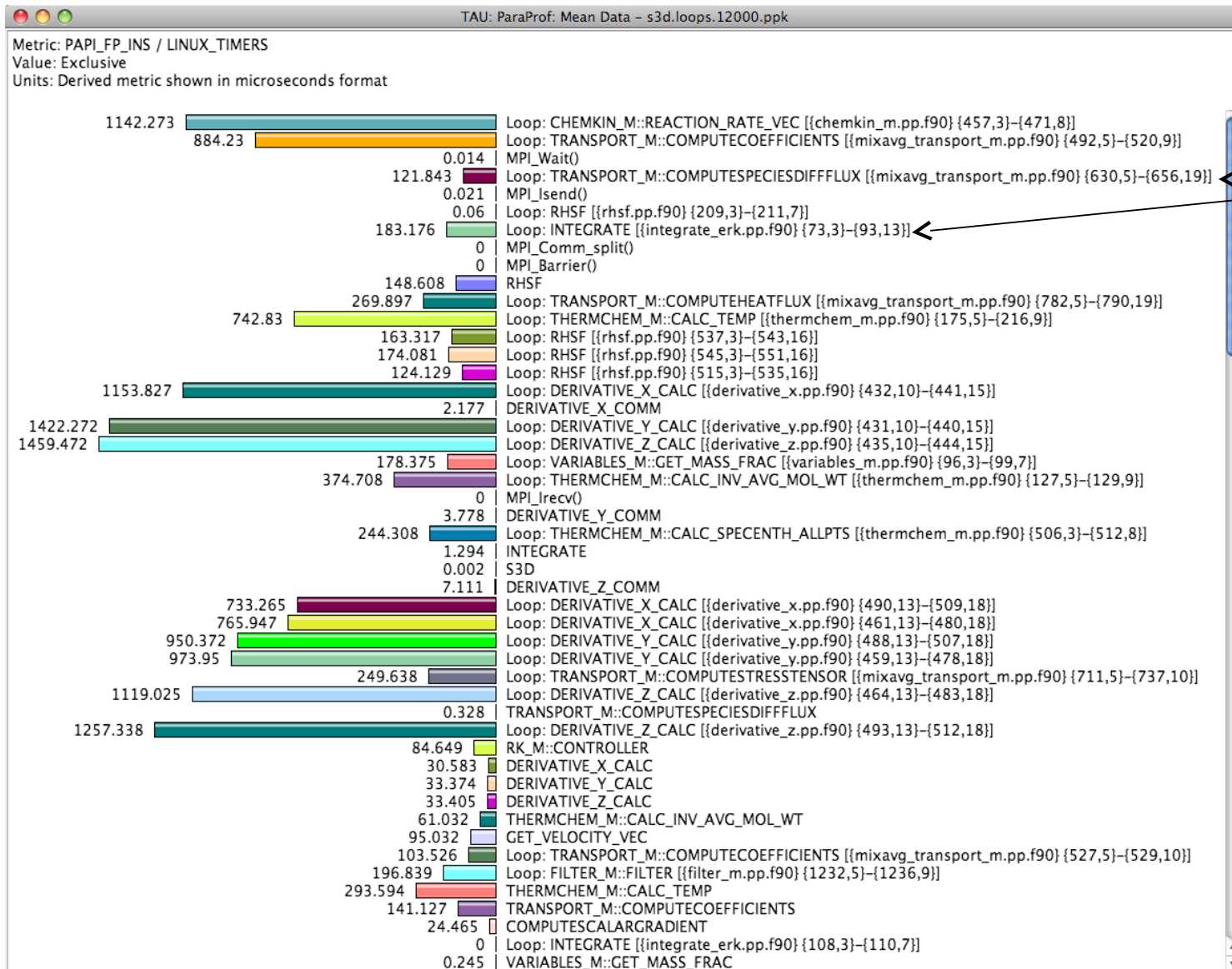


AORSA2D

- magnetized plasma simulation
- Automatic loop level instrumentation
- Blue is single node
- Red is dual core
- Cray XT3 (4K cores)

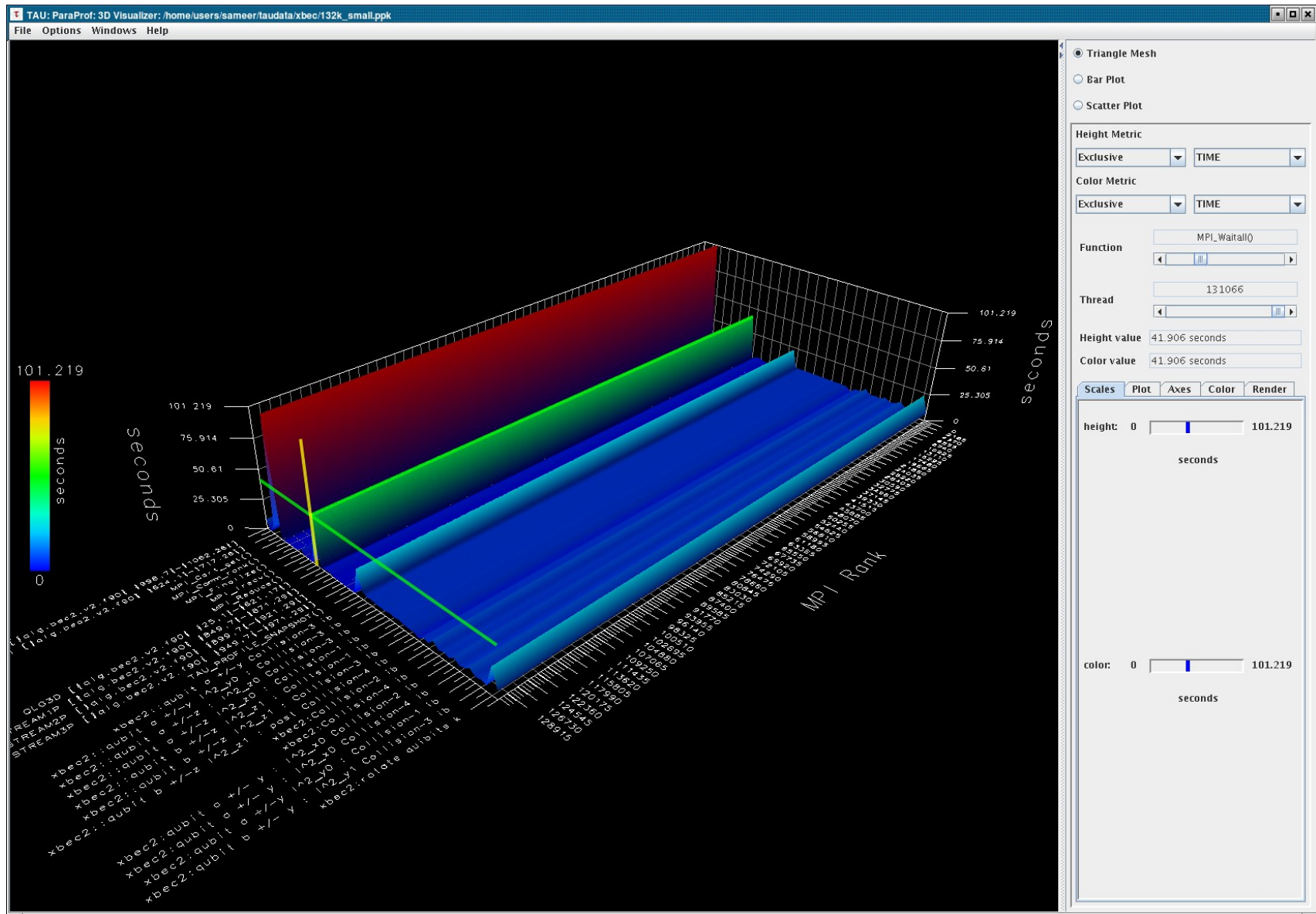


ParaProf: Mflops Sorted by Exclusive Time

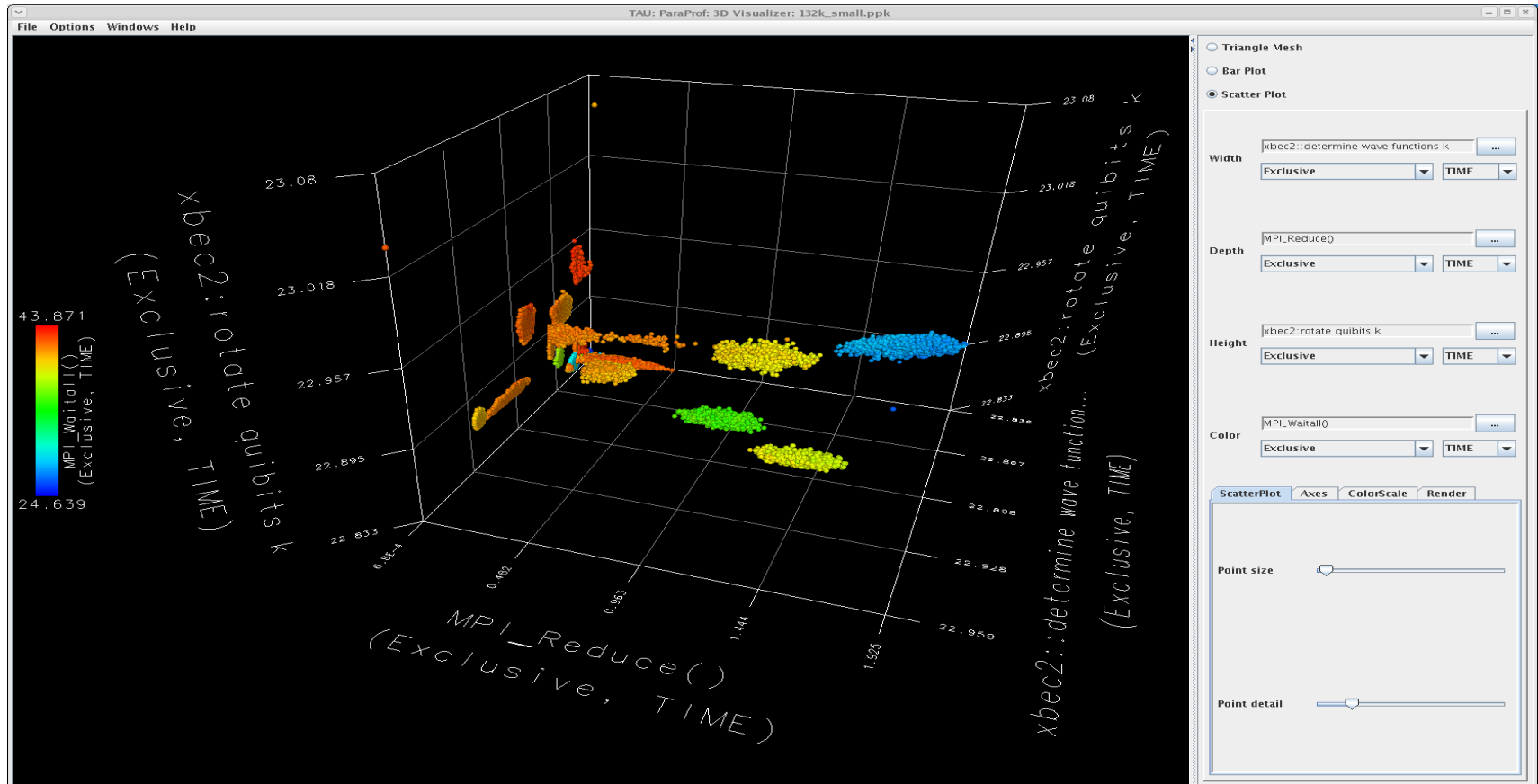


low mflops?

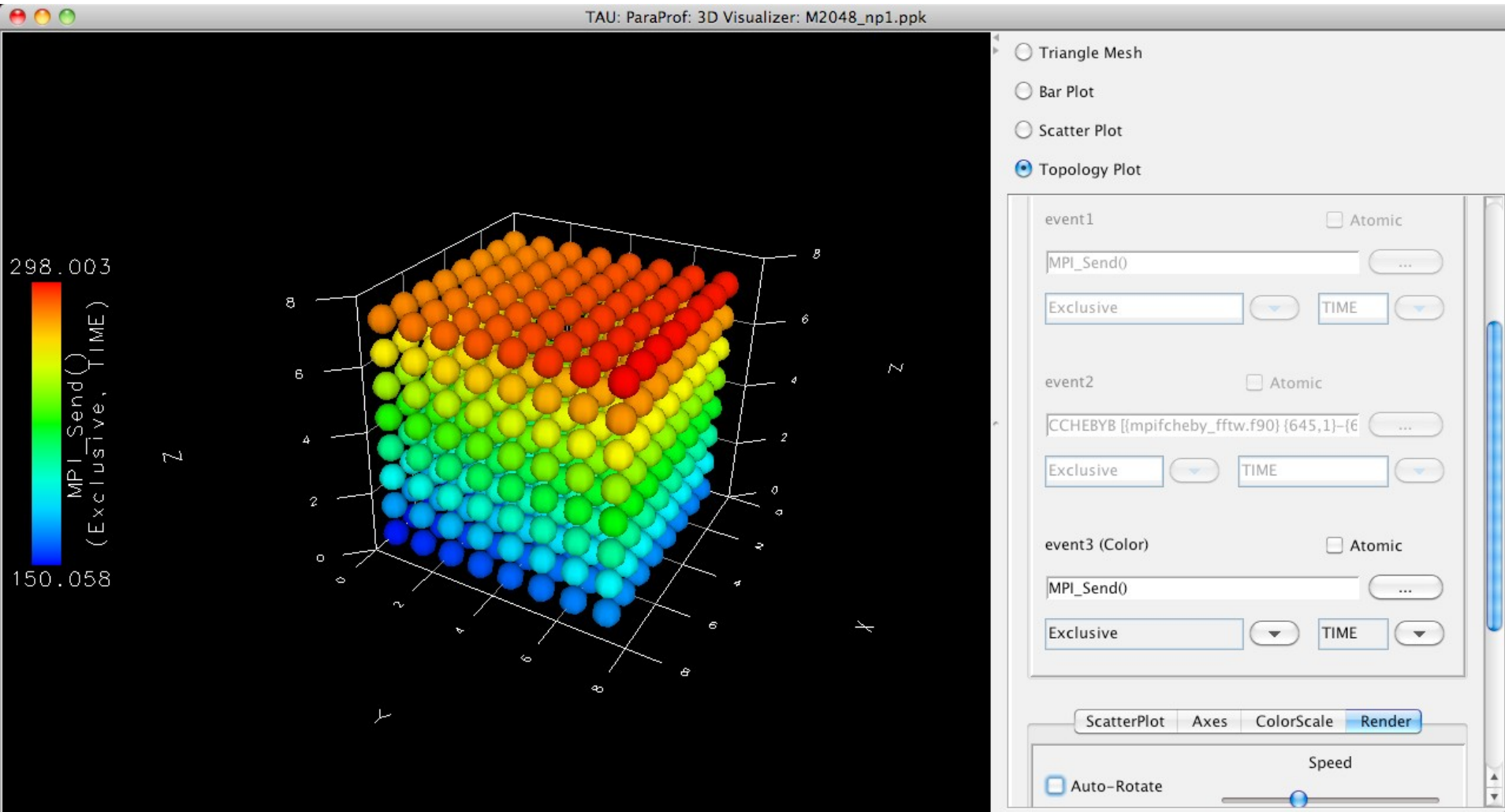
Scalable Visualization: ParaProf (128k cores)



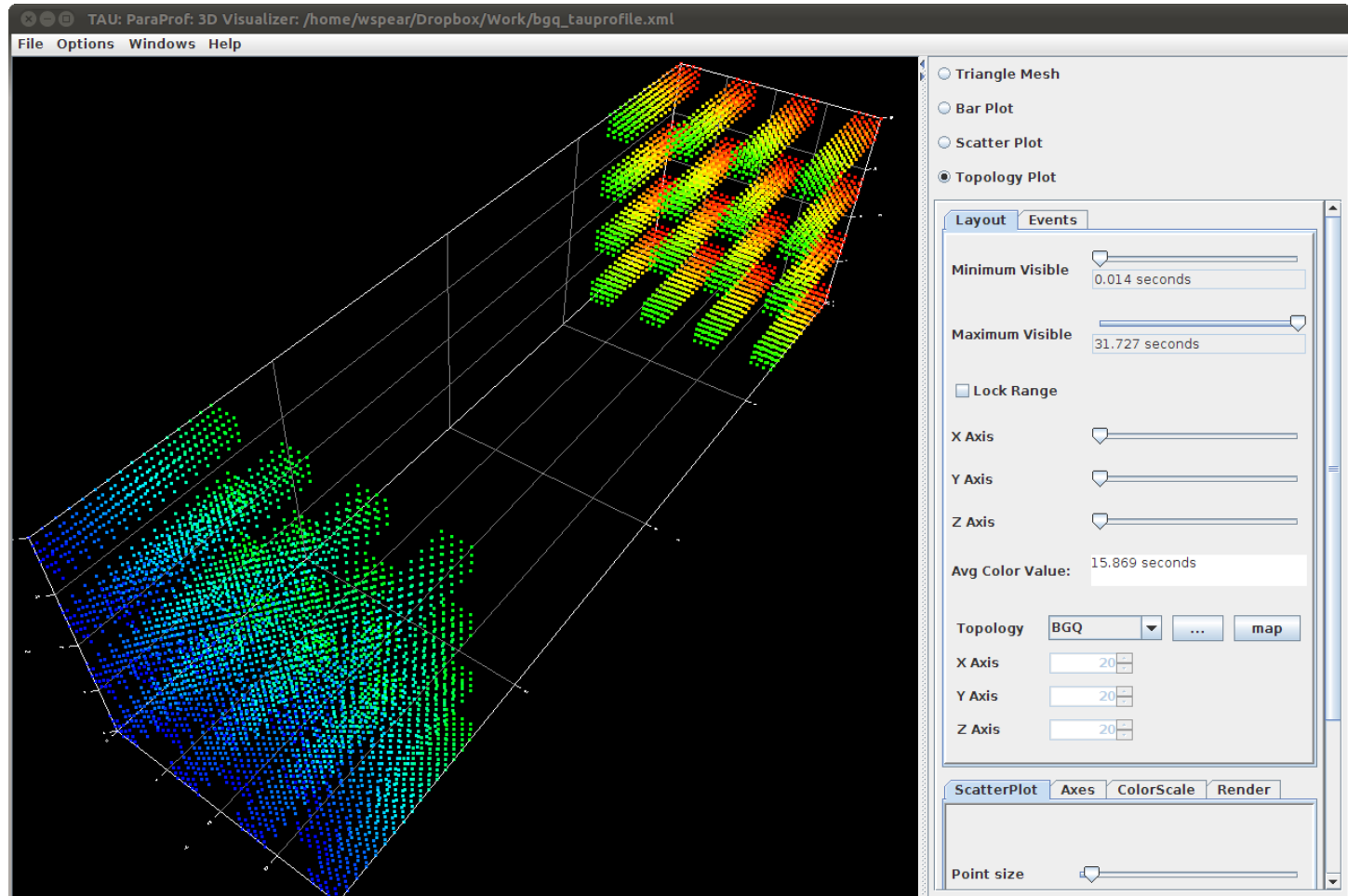
Scatter Plot: ParaProf (128k cores)



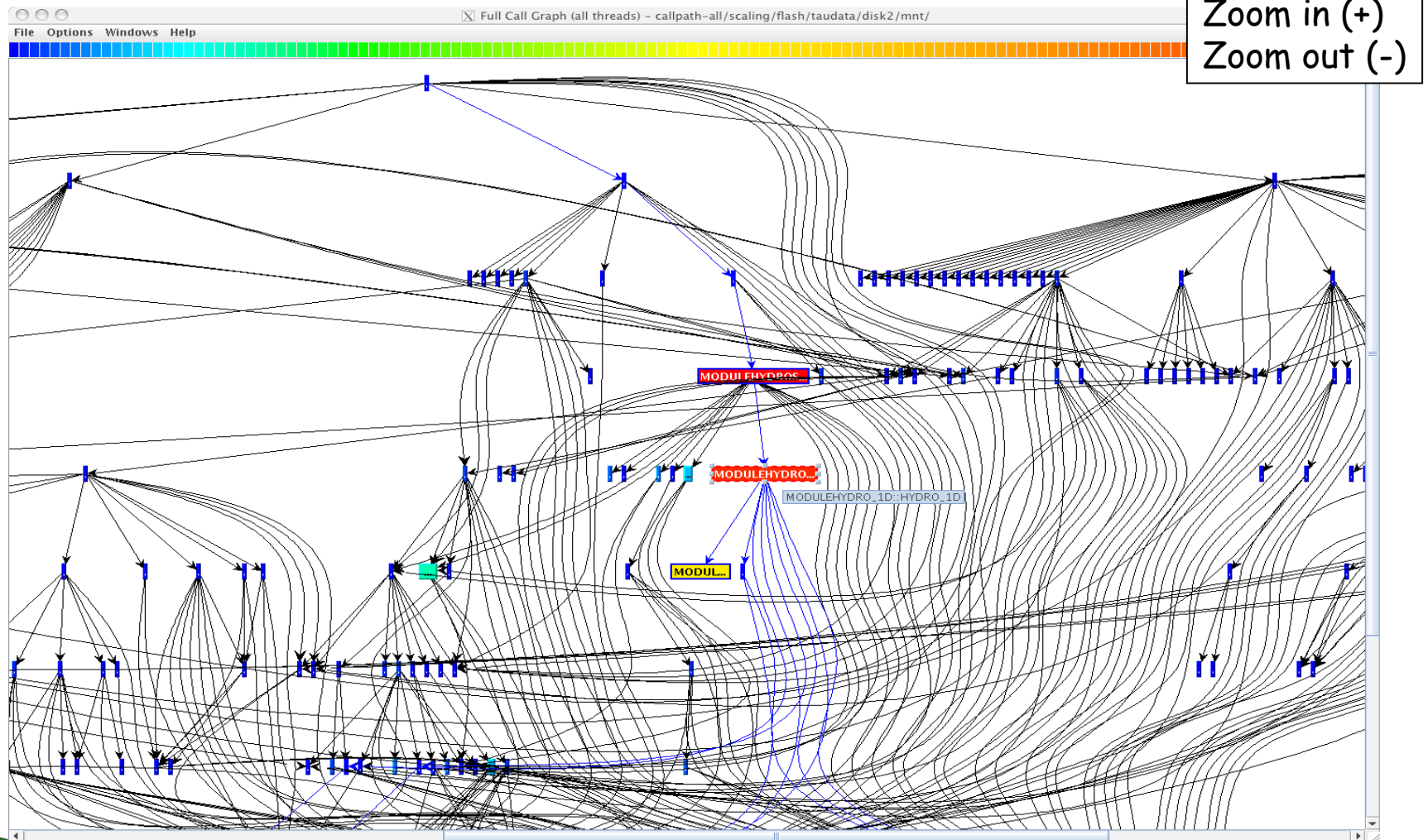
ParaProf: Topology View: MPI_Send on BG/P



ParaProf: Topology View: BG/Q Core Layout



ParaProf – Callgraph Zoomed (Flash)



ParaProf - Thread Statistics Table (GSI)

Thread Statistics: n,c,t, 0,0,0 - comp.ppk/

File Options Windows Help

Name	Inclusive Time	Exclusive Time ▾	Calls	Child Calls
▼ GSI	5,223.564	0.098	1	30
SPECMOD::INIT_SPEC_VARS	0.26	0.26	1	0
▶ MPI_Init()	0.056	0.054	1	1
▼ GSISUB	5,223.094	0.012	1	13
▶ RADINFO::RADINFO_READ	0.103	0.101	1	1,196
PCPINFO::PCPINFO_READ	0.042	0.042	1	0
▼ GLBSOI	5,212.171	0.024	1	12
MPI_Finalize()	1.004	1.004	1	0
▶ OBS_PARA	3.635	0.181	1	56
JFUNC::CREATE_JFUNC	0.142	0.142	1	0
GUESS_GRIDS::CREATE_GES_BIAS_GRIDS	0.059	0.059	1	0
▶ READ_GUESS	1,406.412	0.023	1	8
▼ READ_OBS	3,770.188	0.016	1	6
MPI_Allreduce()	3,725.802	3,725.802	3	0
▶ READ_BUFRTOVS	44.369	0.254	1	871,535
SATTHIN::MAKEGVALS	0	0	1	0
▶ W3FS21	0	0	1	1
BINARY_FILE_UTILITY::OPEN_BINARY_FILE	0.025	0.012	1	3
INITIALIZE::INITIALIZE_RTM	0.099	0.001	1	2
GUESS_GRIDS::CREATE_SFC_GRIDS	0	0	1	0
▶ M_FVANAGRID::ALLGETLIST_	30.582	0	1	10
ERROR_HANDLER::DISPLAY_MESSAGE	0	0	1	0
JFUNC::SET_POINTER	0	0	1	0
OZINFO::OZINFO_READ	0.016	0.016	1	0
DETER_SUBDOMAIN	0.008	0.008	1	0
GRIDMOD::CREATE_MAPPING	0.005	0.005	1	0
INIT_COMMVARS	0.004	0.004	1	0
▶ M_FVANAGRID::ALLGETLIST_	10.711	0	1	1
GRIDMOD::CREATE_GRID_VARS	0	0	1	0



ParaProf - Callpath Thread Relations Window

Call Path Data n,c,t, 0,0,0 - comp.ppk/

File Options Windows Help

Metric Name: Time
Sorted By: Exclusive
Units: seconds

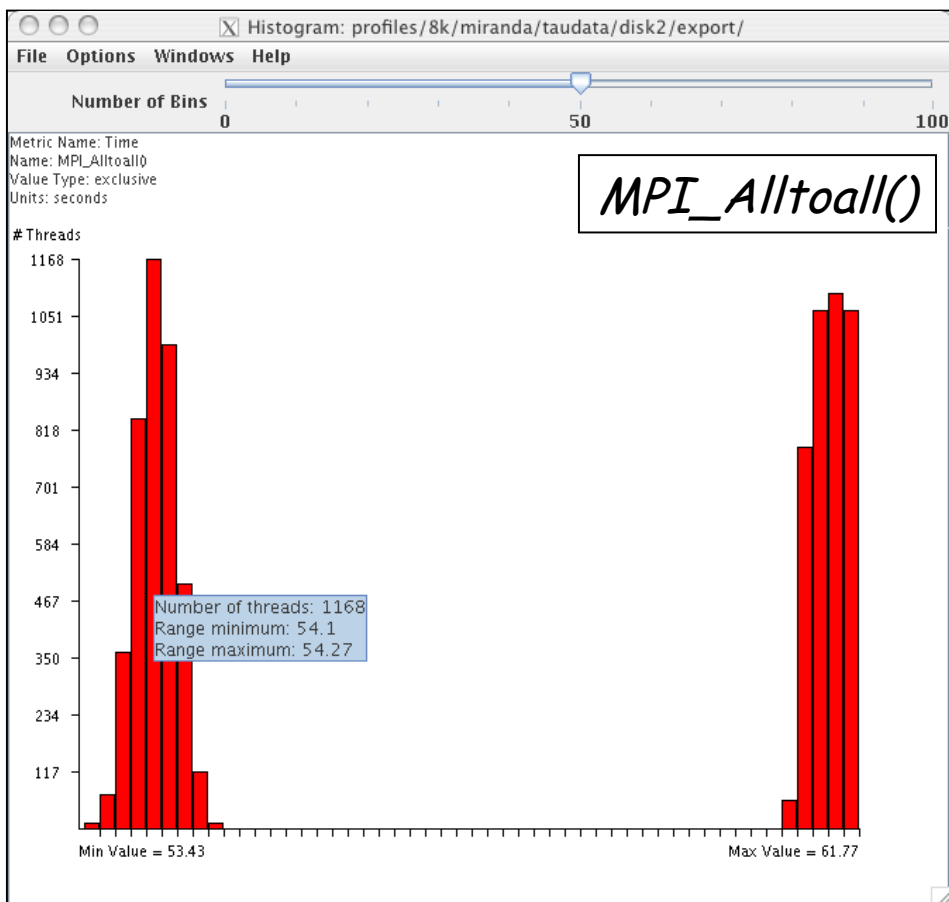
Exclusive	Inclusive	Calls/Tot.Calls	Name[id]
0.023	0.023	3/430	COMPUTE_DERIVED[55]
2.02	2.02	104/430	DPRODMOD::DPROD[66]
0.33	0.33	104/430	INTALLMOD::INTALL[1708]
0.003	0.003	1/430	M_FVANAGRID::ALLGETLIST_[1773]
1.639	1.639	1/430	OBS_PARA[1802]
3725.802	3725.802	3/430	READ_OBS[1860]
214.294	214.294	6/430	SETUPRHSALL[1900]
20.069	20.069	208/430	STPCALCMOD::STPCALC[1942]
--> 3964.18	3964.18	430	MPI_Allreduce()[1762]
2.6E-4	30.582	1/15	GLBSOI[93]
0.007	0.036	1/15	GS[107]
2.7E-4	10.711	1/15	GSISUB[1690]
31.273	1347.703	3/15	M_FVANAGRID::ALLGETLIST_[1773]
0.412	0.412	1/15	PREWGT[1831]
70.198	1406.389	4/15	READ_GUESS[1857]
0.952	0.952	3/15	SATTHIN::GETSFC_GLOBAL[1882]
86.937	95.933	1/15	WRITE_ALL[2004]
--> 196.61	1575.595	15	M_FVANAGRID::ALLGETLIST_[1773]
6.2E-5	6.2E-5	1/1	BALMOD::CREATE_BALANCE_VARS[7]
4.6E-5	4.6E-5	1/1	BALMOD::DESTROY_BALANCE_VARS[8]
3.494	3.494	1/1	BALMOD::PREBAL[9]
0.017	0.017	1/1	BERROR::CREATE_BERROR_VARS[11]
2.0E-4	2.0E-4	1/1	BERROR::DESTROY_BERROR_VARS[12]
8.6E-5	8.6E-5	1/1	BERROR::SET_PREDICTORS_VAR[16]
5.7E-5	5.7E-5	1/1	COMPACT_DIFFS::CREATE_CDIFF_COEFS[34]
4.9E-5	4.9E-5	1/1	COMPACT_DIFFS::DESTROY_CDIFF_COEFS[35]
0.015	0.042	1/1	COMPACT_DIFFS::INISPH[41]
0.052	8.196	3/3	COMPUTE_DERIVED[55]
1.4E-4	3.1E-4	3/3	GETLIST_:MOVDATE_[89]
4.2E-5	4.2E-5	1/1	GRIDMOD::DESTROY_GRID_VARS[98]
8.2E-5	8.2E-5	1/1	GRIDMOD::DESTROY_MAPPING[99]
0.169	0.169	3/3	GUESS_GRIDS::CREATE_ATM_GRIDS[1692]
3.3E-4	3.3E-4	3/3	GUESS_GRIDS::DESTROY_ATM_GRIDS[1695]
9.1E-5	9.1E-5	1/1	GUESS_GRIDS::DESTROY_GES_BIAS_GRIDS[1696]
2.2E-4	2.2E-4	1/1	GUESS_GRIDS::DESTROY_SFC_GRIDS[1697]
6.6E-5	6.4E-4	1/1	INITIALIZE::DESTROY_RTM[1705]
5.8E-5	5.8E-5	1/1	JFUNC::DESTROY_JFUNC[1739]
0.003	1/430	1/430	MPI_Allreduce()[1762]
0.017	0.017	68/116	MPI_Bcast()[1764]
0.004	0.004	297/409	MPI_Comm_rank()[1765]

Parent

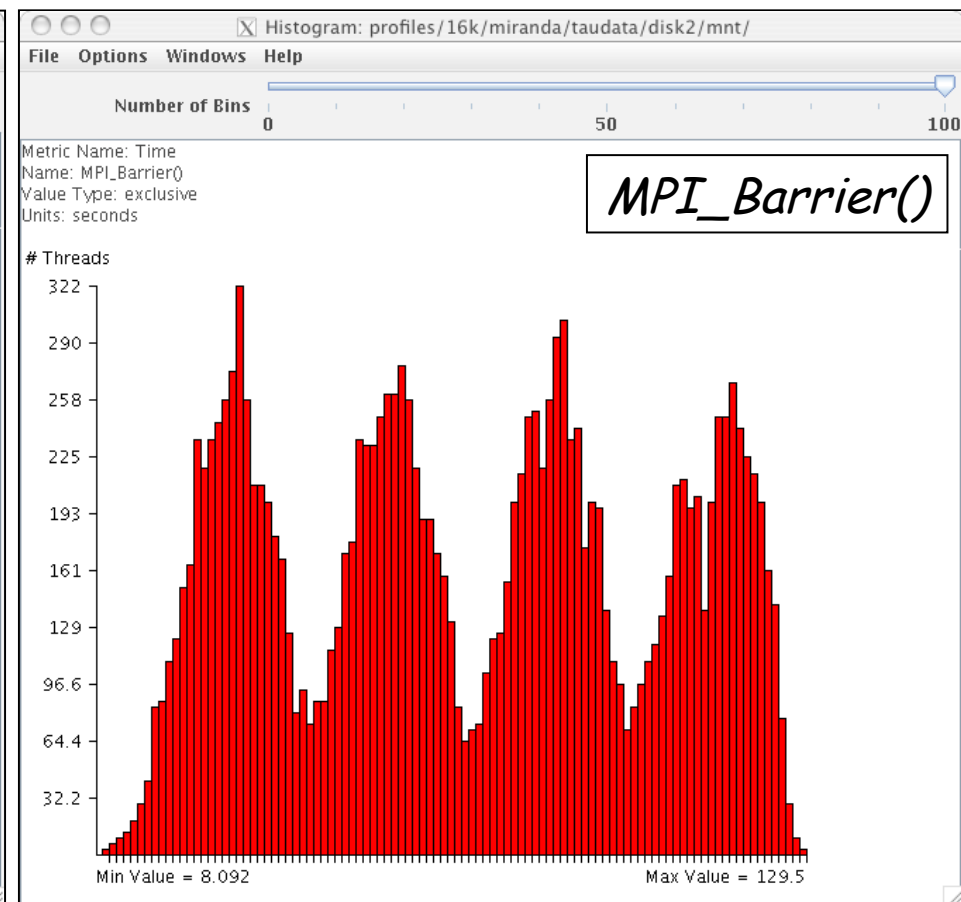
Routine

Children

ParaProf – Histogram View (Miranda)



8k processors



16k processors



Using Performance Database (PerfDMF)

- **Configure PerfDMF (Done by each user)**
% perfdmf_configure --create-default
 - Choose derby, PostgreSQL, MySQL, Oracle or DB2
 - Hostname
 - Username
 - Password
 - Say yes to downloading required drivers (we are not allowed to distribute these)
 - Stores parameters in your ~/.ParaProf/perfdmf.cfg file
- **Configure PerfExplorer (Done by each user)**
% perfexplorer_configure
- **Execute PerfExplorer**
% perfexplorer



PerfDMF and the TAU Portal

- Development of the TAU portal
 - Common repository for collaborative data sharing
 - Profile uploading, downloading, user management
 - Paraprof, PerfExplorer can be launched from the portal using Java Web Start (no TAU installation required)
- Portal URL
<http://tau.nic.uoregon.edu>



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
 - Client-server based system architecture
- Technology integration
 - Java API and toolkit for portability
 - PerfDMF
 - R-project/Omegahat, Octave/Matlab statistical analysis
 - WEKA data mining package
 - JFreeChart for visualization, vector output (EPS, SVG)

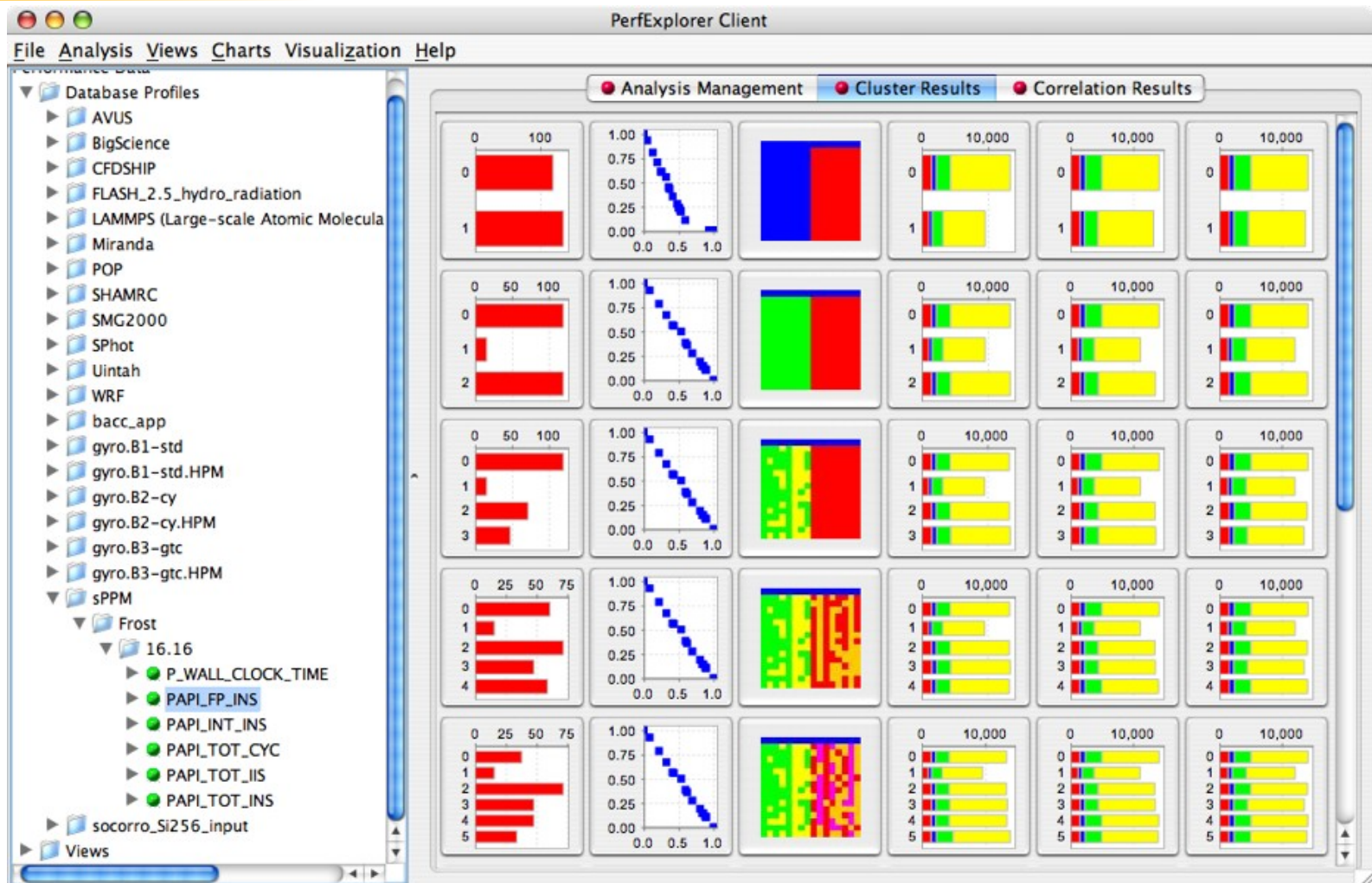


PerfExplorer - Cluster Analysis

- Performance data represented as vectors - each dimension is the cumulative time for an event
- *k*-means: *k* random centers are selected and instances are grouped with the "closest" (Euclidean) center
- New centers are calculated and the process repeated until stabilization or max iterations
- Dimension reduction necessary for meaningful results
- Virtual topology, summaries constructed

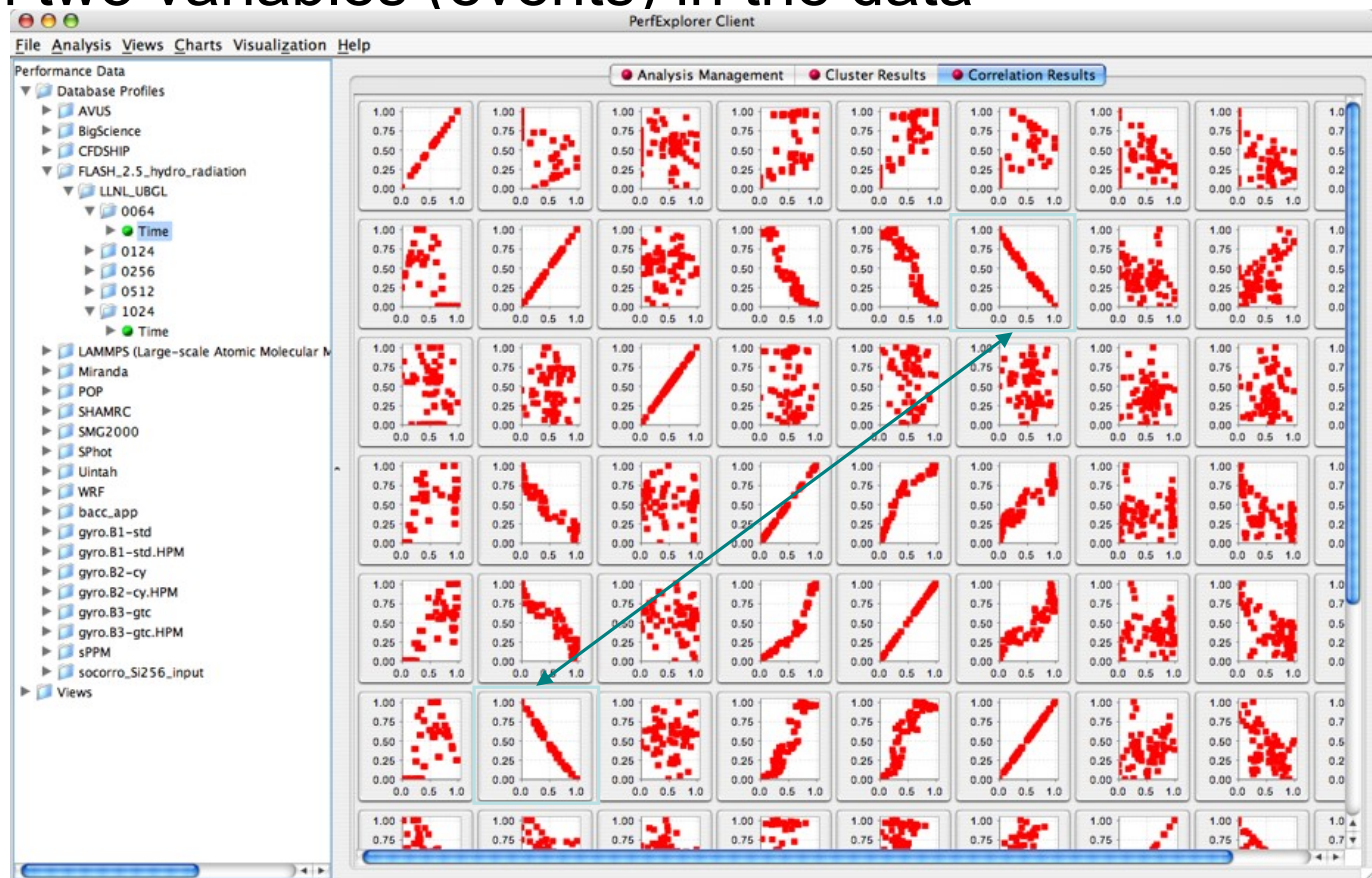


PerfExplorer - Cluster Analysis (sPPM)



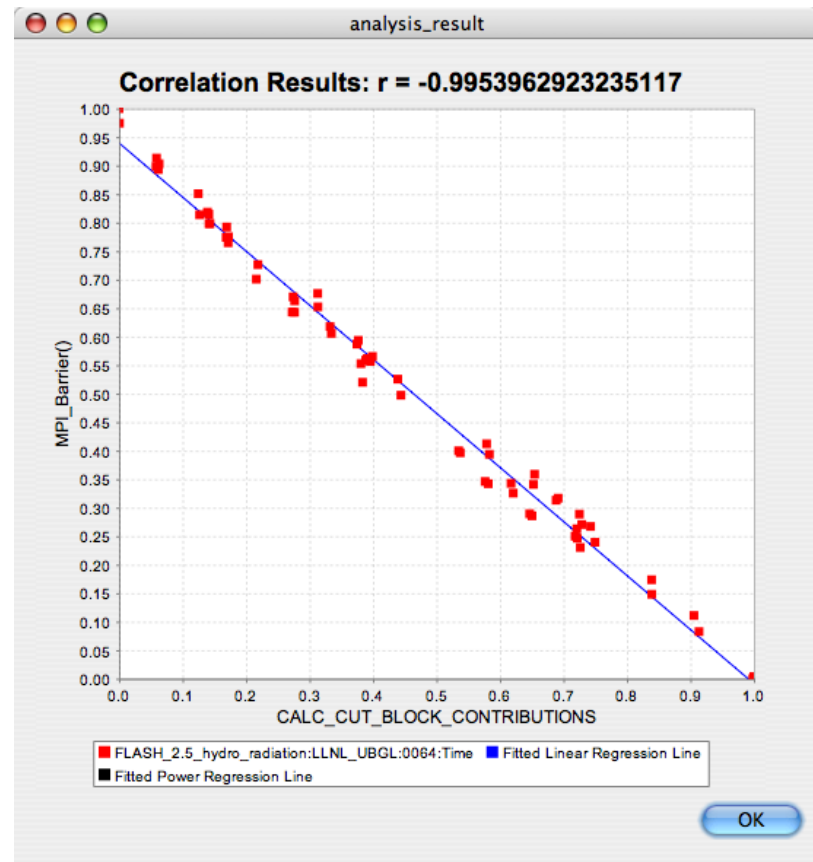
PerfExplorer - Correlation Analysis (Flash)

- Describes strength and direction of a linear relationship between two variables (events) in the data



PerfExplorer - Correlation Analysis (Flash)

- -0.995 indicates strong, negative relationship
- As CALC_CUT_BLOCK_CONTRIBUTIONS increases in execution time, MPI_Barrier() decreases



PerfExplorer - Comparative Analysis

- Relative speedup, efficiency
 - total runtime, by event, one event, by phase
- Breakdown of total runtime
- Group fraction of total runtime
- Correlating events to total runtime
- Timesteps per second
- Performance Evaluation Research Center (PERC)
 - PERC tools study (led by ORNL, Pat Worley)
 - In-depth performance analysis of select applications
 - Evaluation performance analysis requirements
 - Test tool functionality and ease of use



PerfExplorer - Interface

The screenshot shows the PerfExplorer Client interface. On the left is a file tree under 'gyro.B1-std' containing various subfolders like 'B1-std-hwpc.phoenix.0x002', 'B1-std-inst.phoenix.0x002', 'B1-std-nl2.cheetah.affnosng', 'B1-std-nl2.cheetah.noaffnosng', 'B1-std-nl2.phoenix.0x002', 'B1-std-nl2.phoenix.0x002scr', 'B1-std.53newest.phoenix.0x002', 'B1-std.cheetah.affnosng', 'B1-std.cheetah.affsng', 'B1-std.cheetah.noaffnosng', 'B1-std.hockney', 'B1-std.new.phoenix.0x002', 'B1-std.phoenix.0x002', 'B1-std.phoenix.0x002scr', 'B1-std.ram0x002.a', 'B1-std.ram0x002.b', 'B1-std.seaborg', 'B1-std.timing.seaborg.128', 'B1-std.timing.seaborg.16', 'B1-std.timing.seaborg.256', 'B1-std.timing.seaborg.32', 'B1-std.timing.seaborg.512', 'B1-std.timing.seaborg.64', 'B1-std.tg', 'gyro.B2-cy', and 'gyro.B3-gtc'. On the right is a table titled 'Analysis Management' and 'Performance Explorer' showing metadata for the selected experiment 'B1-std-nl2.cheetah.noaffnosng' with ID 16. The table lists fields like Name, Experiment ID, system_name, system_machine_type, system_arch, system_os, system_memory_size, system_processor_amt, system_l1_cache_size, system_l2_cache_size, system_userdata, compiler_cpp_name, and compiler_cpp_version. Red callouts highlight key features: 'Select experiments and trials of interest' points to the file tree, 'Experiment metadata' points to the table, and 'Data organized in application, experiment, trial structure (will allow arbitrary in future)' points to the file tree structure.

File Analysis Charts Help

gyro.B1-std

- B1-std-hwpc.phoenix.0x002
- B1-std-inst.phoenix.0x002
- B1-std-inst.phoenix.0x002.profil
- B1-std-nl2.cheetah.affnosng
- B1-std-nl2.cheetah.affsng
- B1-std-nl2.cheetah.noaffnosng
- B1-std-nl2.phoenix.0x002
- B1-std-nl2.phoenix.0x002scr
- B1-std.53newest.phoenix.0x002
- B1-std.cheetah.affnosng
- B1-std.cheetah.affsng
- B1-std.cheetah.noaffnosng
- B1-std.hockney
- B1-std.new.phoenix.0x002
- B1-std.phoenix.0x002
- B1-std.phoenix.0x002scr
- B1-std.ram0x002.a
- B1-std.ram0x002.b
- B1-std.seaborg
 - B1-std.timing.seaborg.128
 - B1-std.timing.seaborg.16
 - B1-std.timing.seaborg.256
 - B1-std.timing.seaborg.32
 - B1-std.timing.seaborg.512
 - B1-std.timing.seaborg.64
- B1-std.tg
- gyro.B2-cy
- gyro.B3-gtc

Analysis Management Performance Explorer

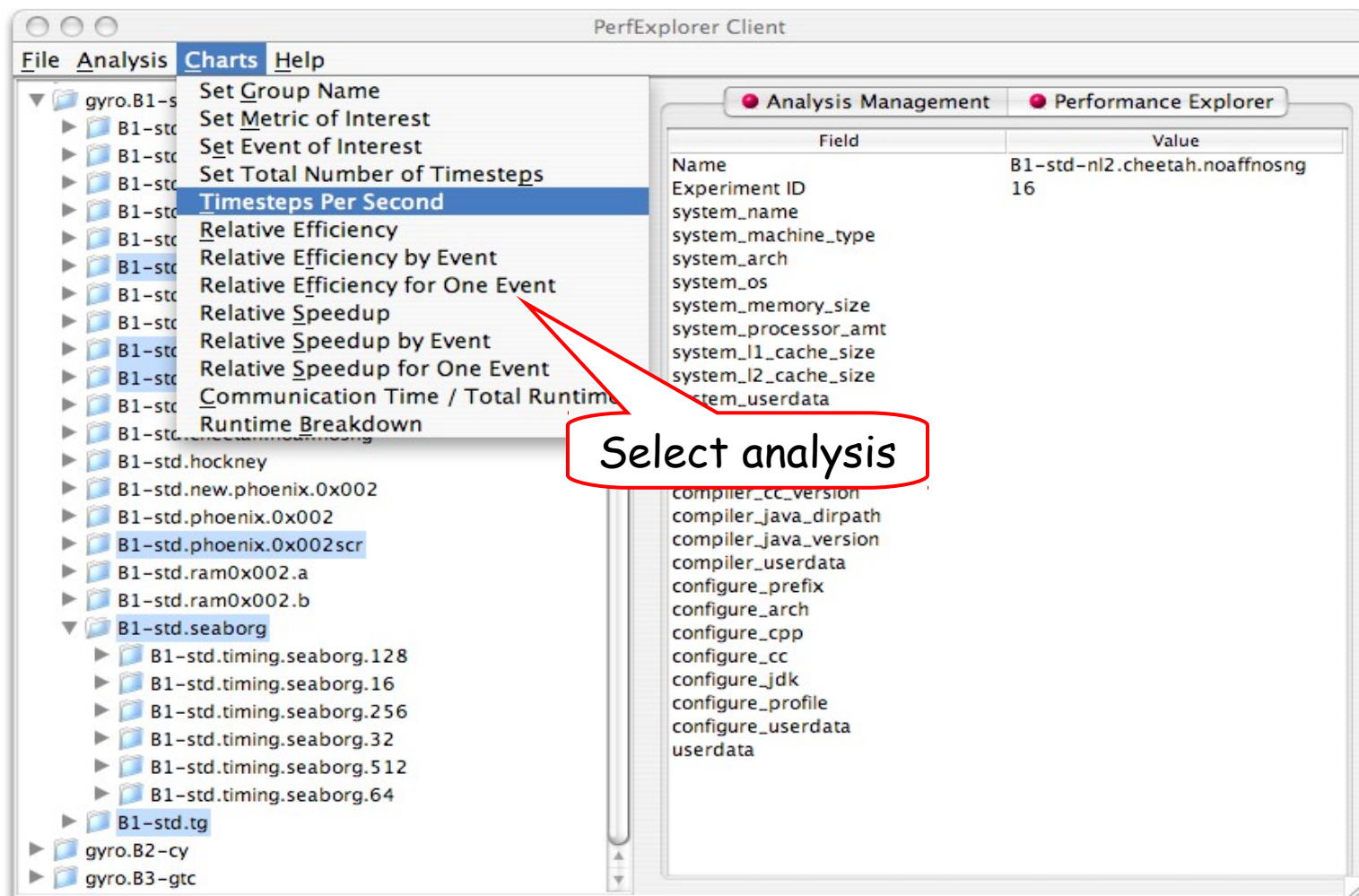
Field	Value
Name	B1-std-nl2.cheetah.noaffnosng
Experiment ID	16
system_name	
system_machine_type	
system_arch	
system_os	
system_memory_size	
system_processor_amt	
system_l1_cache_size	
system_l2_cache_size	
system_userdata	
compiler_cpp_name	
compiler_cpp_version	
configuration_prefix	
configuration_arch	

Experiment metadata

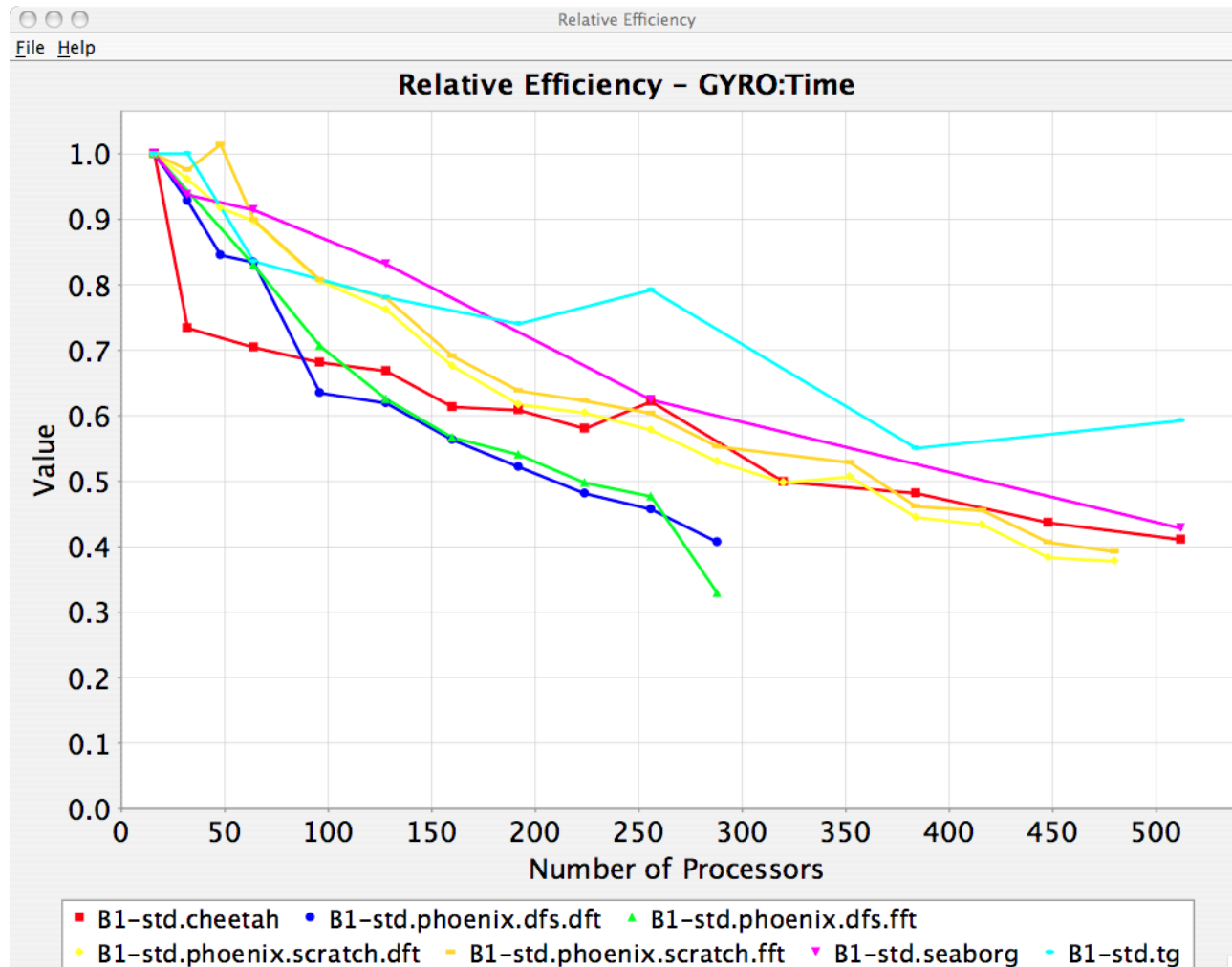
Select experiments and trials of interest

Data organized in application, experiment, trial structure (will allow arbitrary in future)

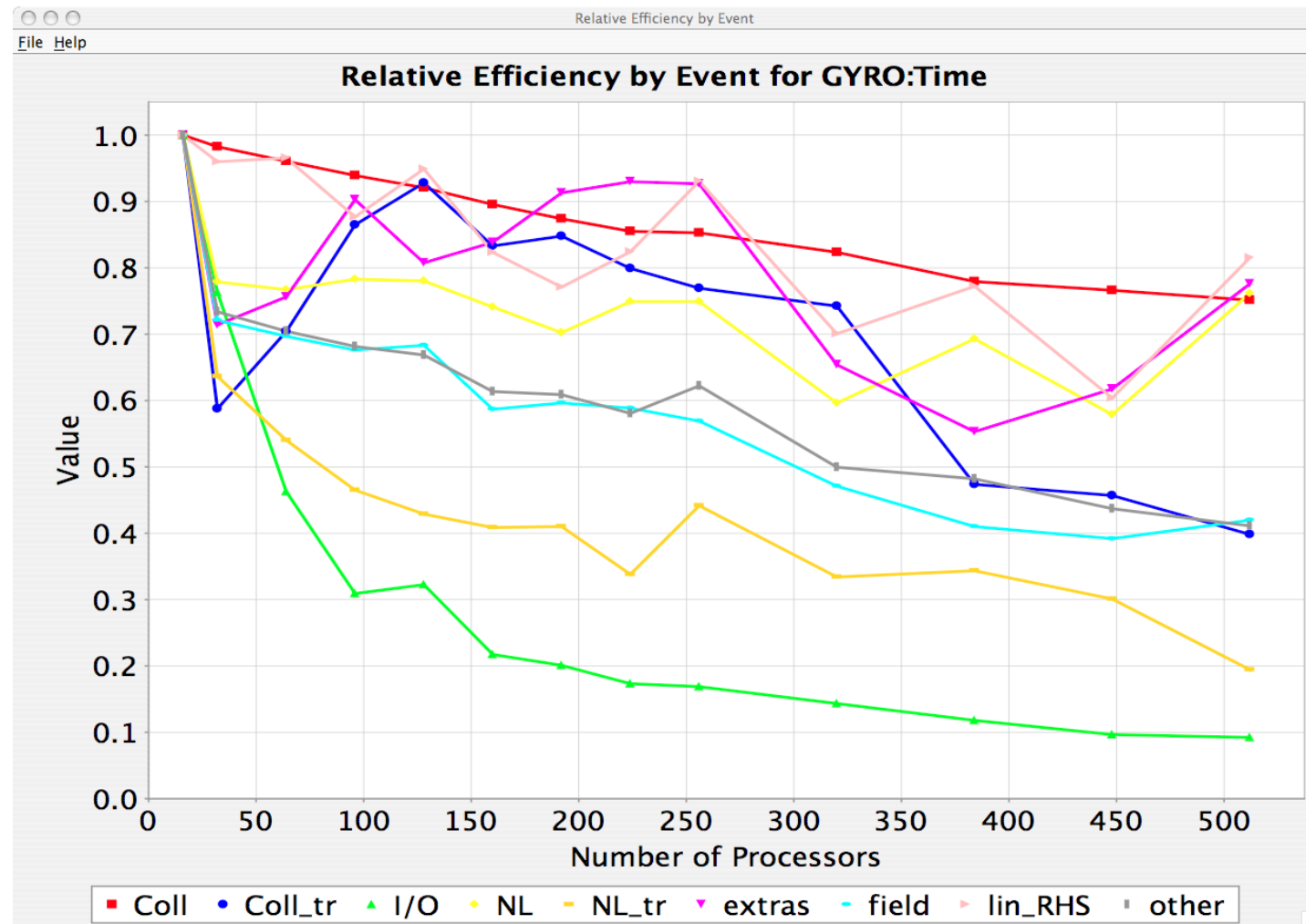
PerfExplorer - Interface



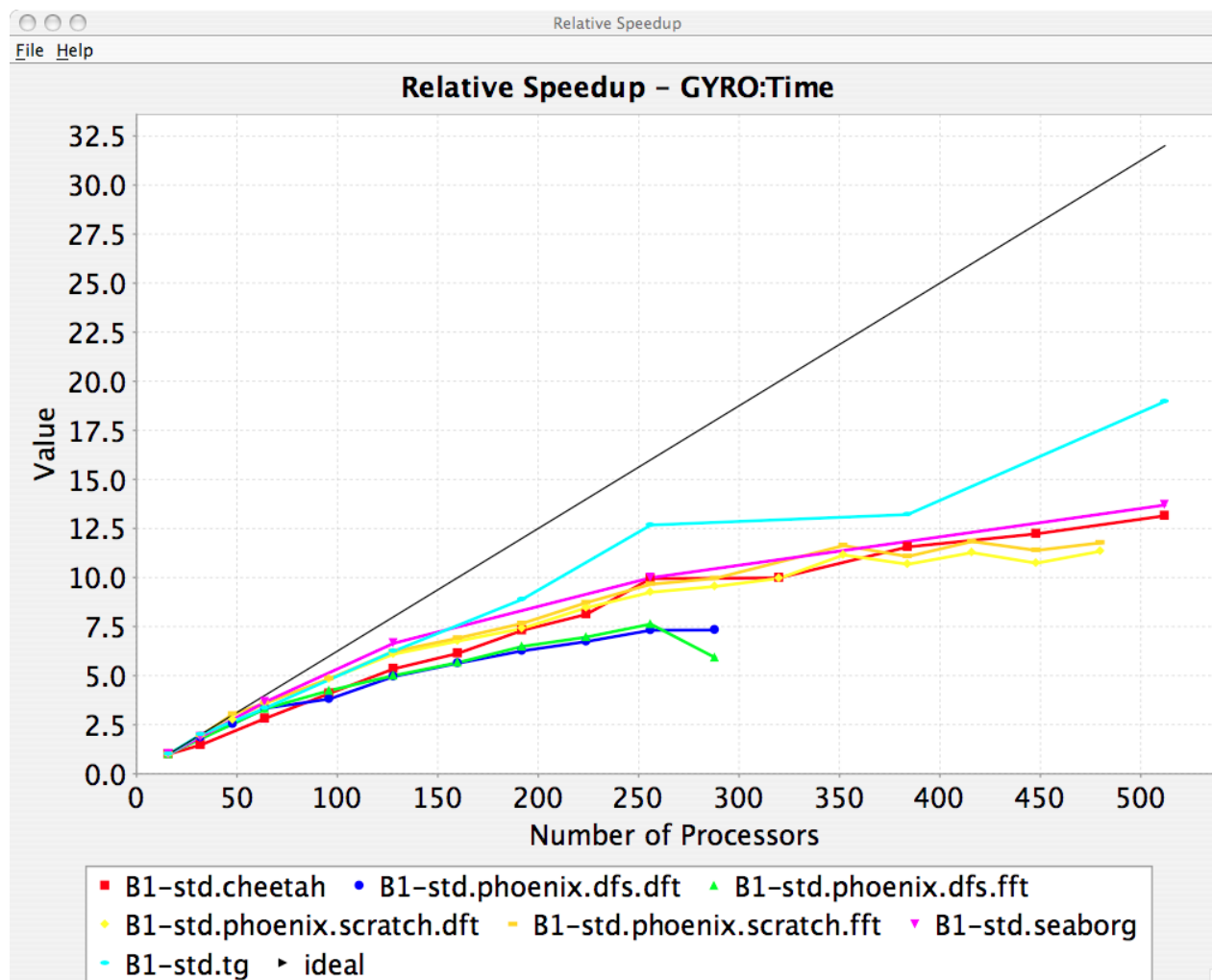
PerfExplorer - Relative Efficiency Plots



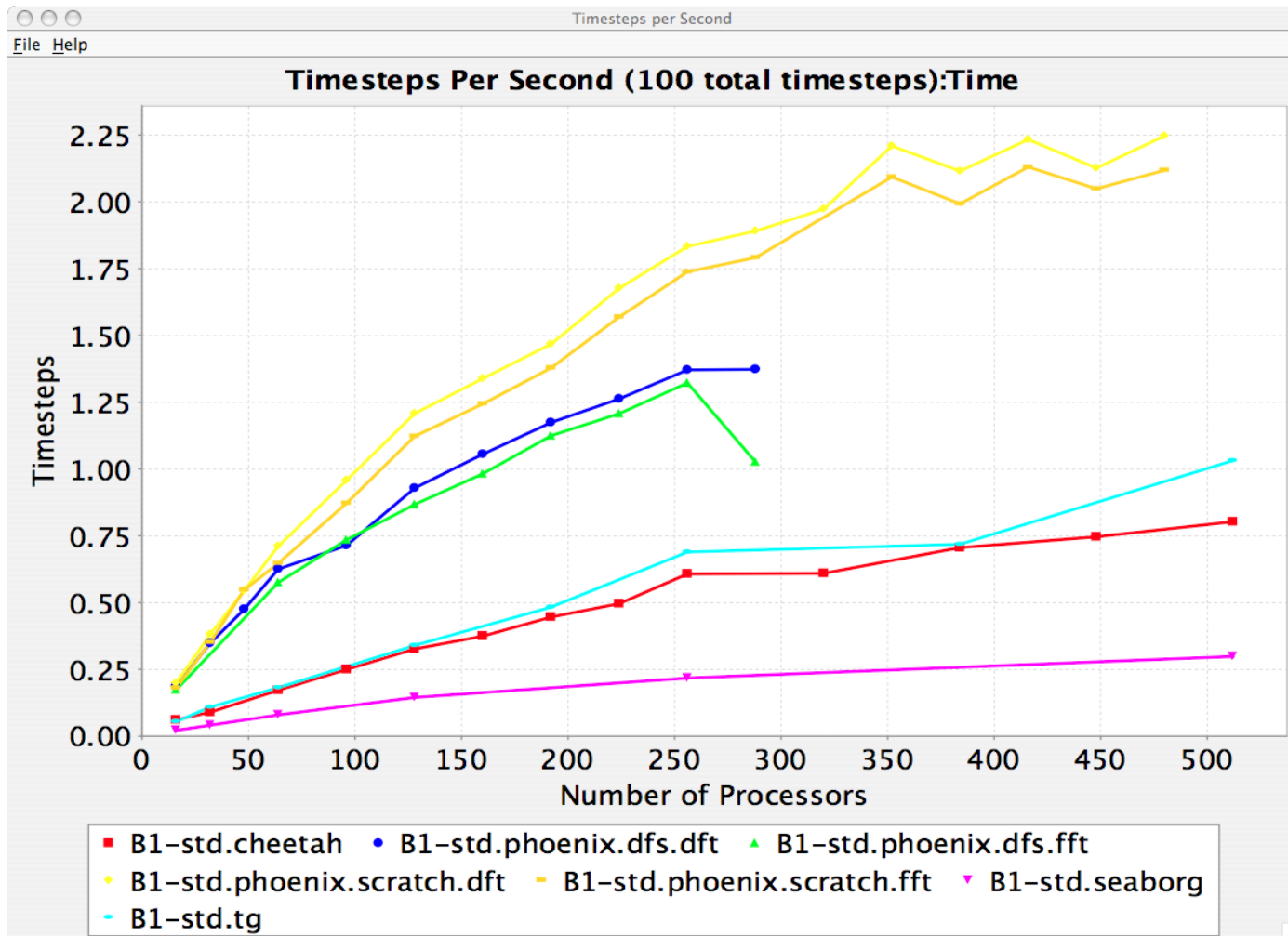
PerfExplorer - Relative Efficiency by Routine



PerfExplorer - Relative Speedup

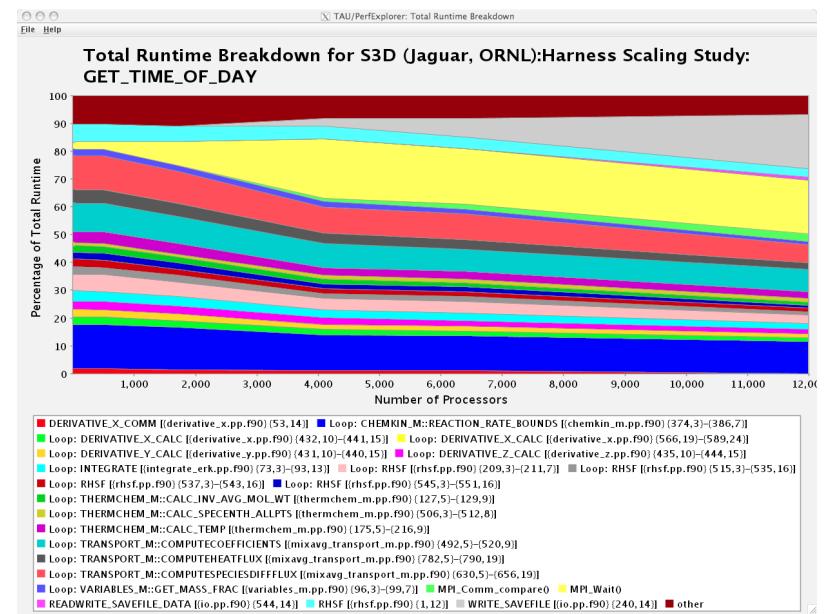
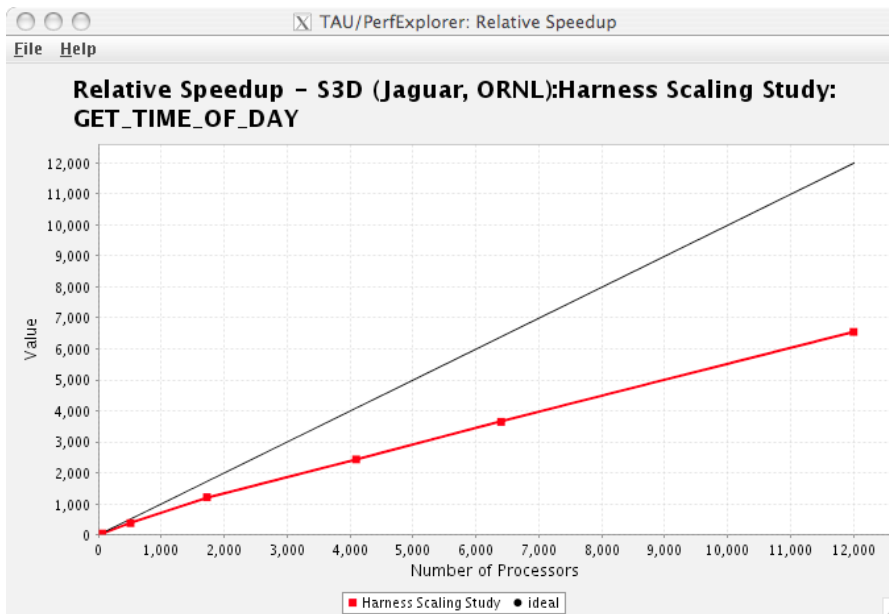


PerfExplorer - Timesteps Per Second

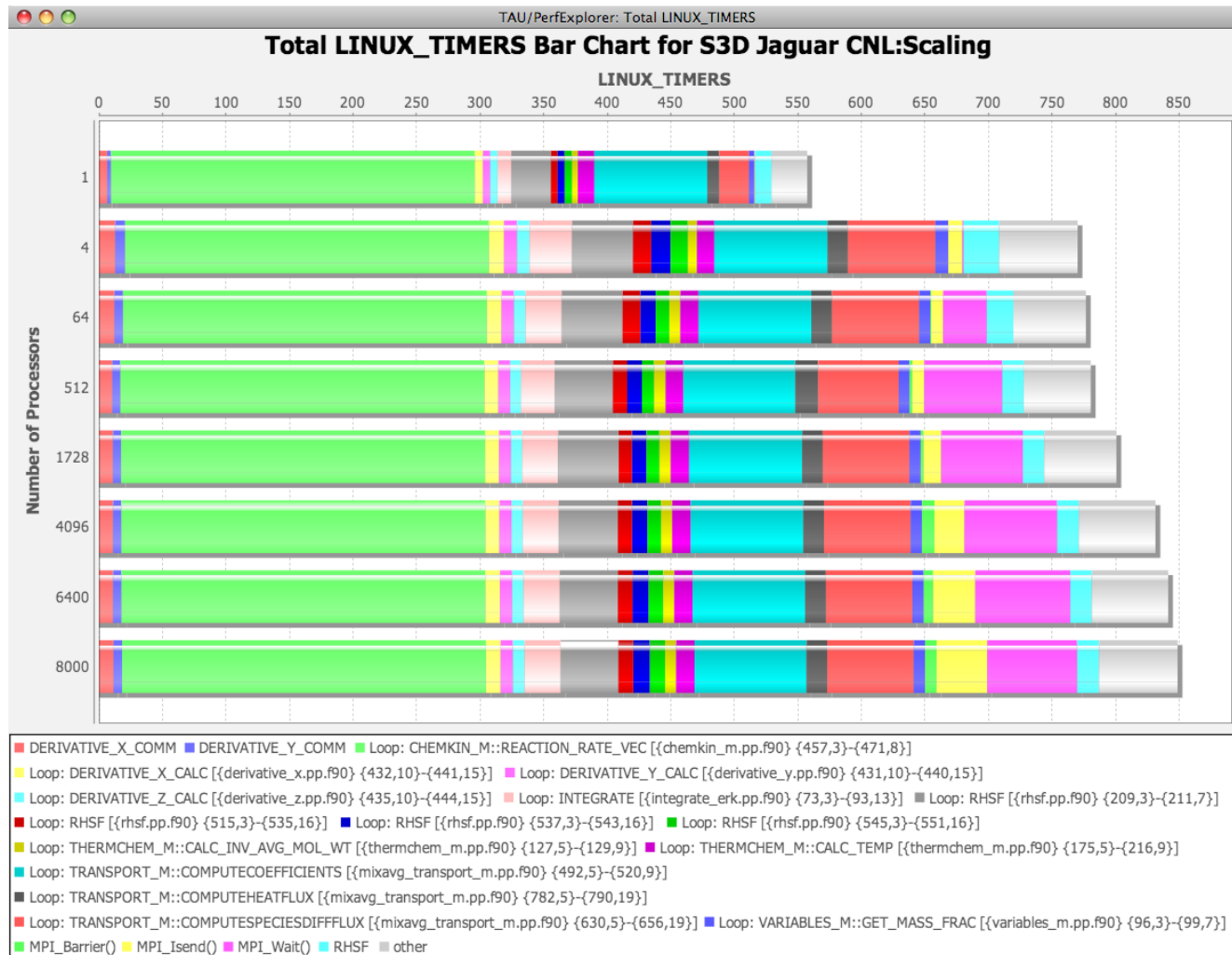


Usage Scenarios: Evaluate Scalability

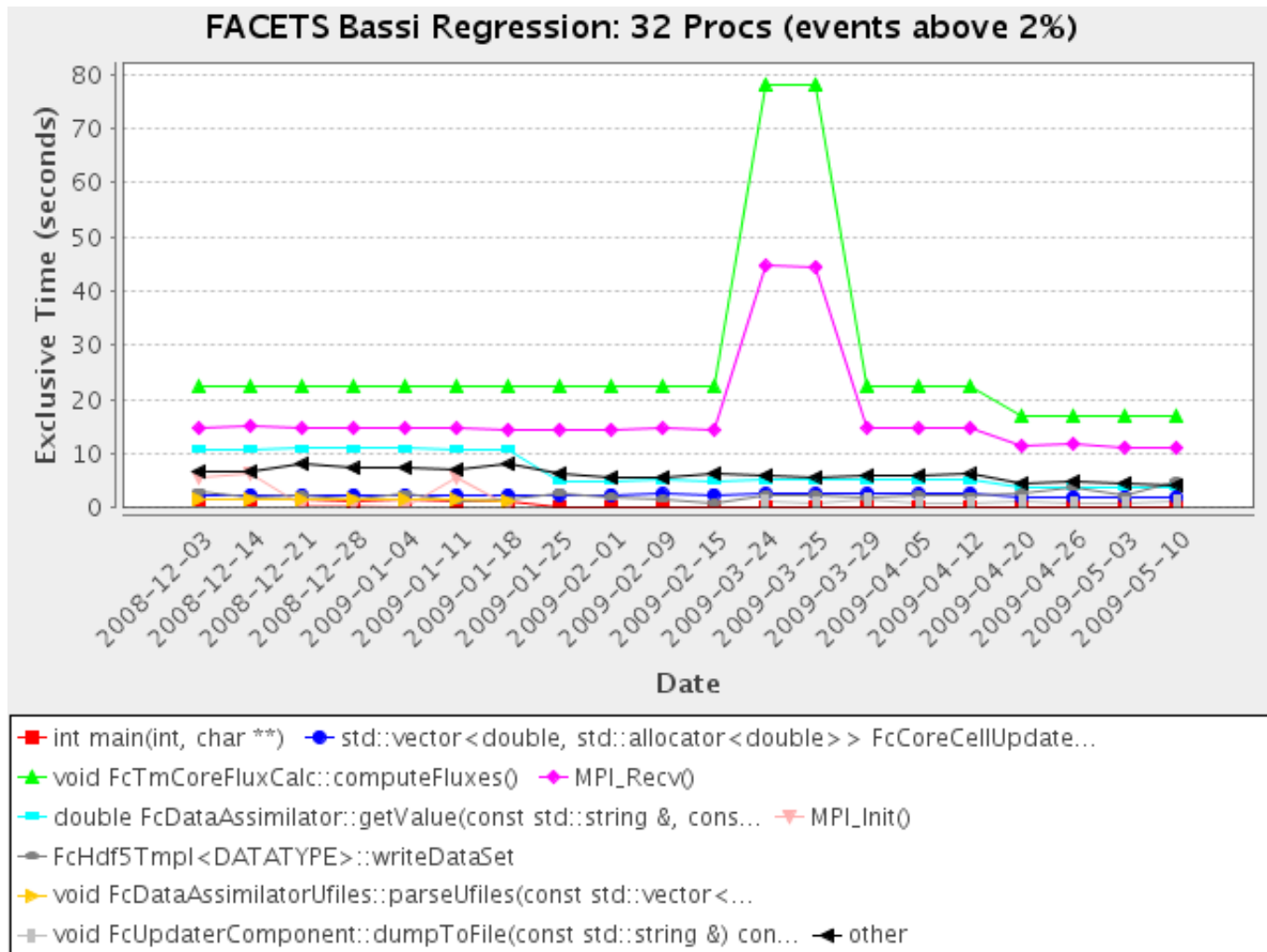
- Goal: How does my application scale? What bottlenecks occur at what core counts?
- Load profiles in PerfDMF database and examine with PerfExplorer



Usage Scenarios: Evaluate Scalability



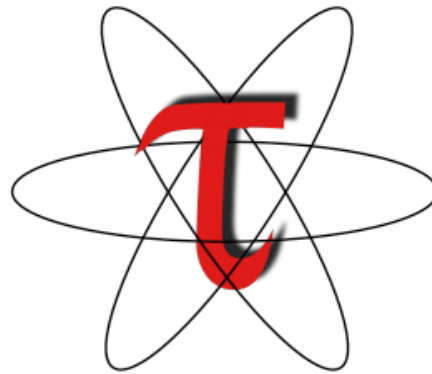
Performance Regression Testing



Evaluate Scalability using PerfExplorer Charts

```
% export TAU_MAKEFILE=$TAU_ROOT
                                /lib/Makefile.tau-mpi-pdt
% export PATH=$TAUROOTDIR/x86_64/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:
% perfdmf_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
    perfdmf_loadtrial
Then,
% perfexplorer
(Select experiment, Menu: Charts -> Speedup)
```

PAPI hardware counters



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]: 0x4000801b |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
[...]
```

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
```

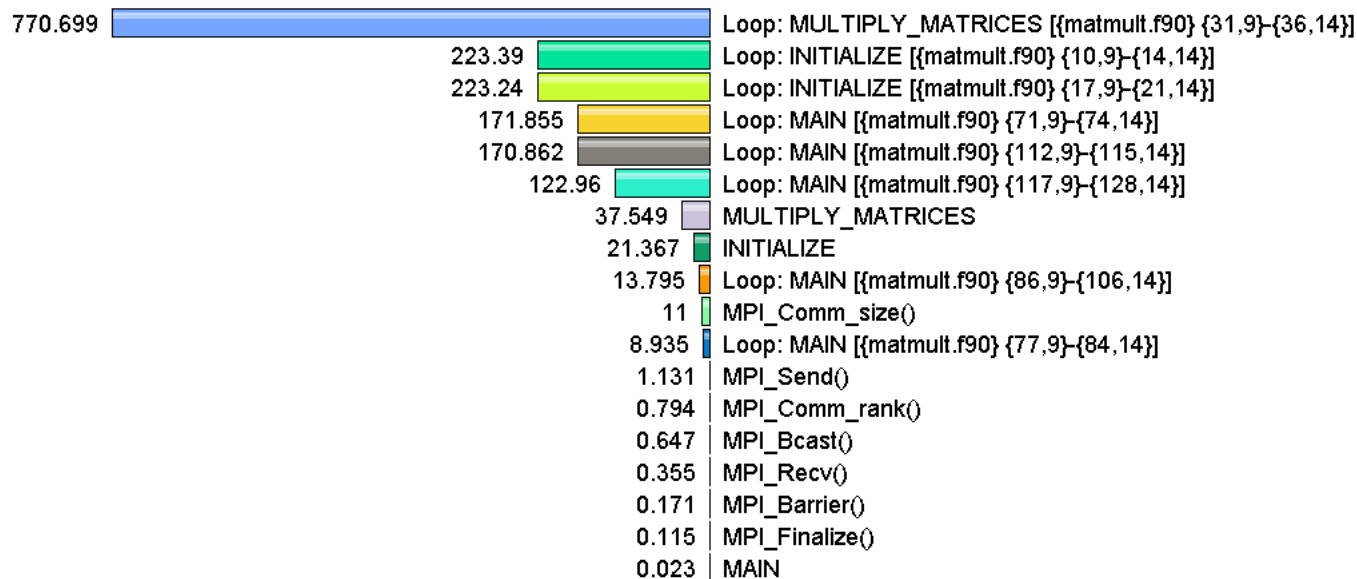
Usage Scenarios: Calculate mflops in Loops

- 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



Generate a PAPI profile with 2 or more counters

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

% export PATH=$TAUROOTDIR/x86_64/bin:$PATH
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% qsub -I -l nodes=1:ppn=8 -X
% export TAU_METRICS=TIME:PAPI_FP_INS:PAPI_L1_DCM
% mpirun -np 8 ./a.out
% paraprof --pack app.ppk
    Move the app.ppk file to your desktop.
% paraprof app.ppk
    Choose Options -> Show Derived Panel -> "PAPI_FP_INS", click "/", "TIME", click "Apply"
    choose.
```

Derived Metrics in ParaProf

TAU: ParaProf Manager

File Options Help

Applications

- Standard Applications
 - Default App
 - Default Exp
 - f90/root/
 - TIME
 - PAPI_L1_DCM
 - PAPI_FP_INS

TrialField	Value
Trial ID	0
CPU Cores	6
CPU MHz	2600.093
CPU Type	Six-Core AMD Opteron(tm) Processor 8435
CPU Vendor	AuthenticAMD
CWD	/root/f90
Cache Size	512 KB
Executable	/root/f90/ring.i
File Type Index	1
File Type Name	Tau profiles
Hostname	b2e4s5.corning.com
Local Time	2010-04-18T02:27:58-04:00
MPI Processor Name	b2e4s5.corning.com
MemorySize	66006592 KB
Node Name	b2e4s5.corning.com
OS Machine	x86_64
OS Name	Linux
OS Release	2.6.18-128.el5.perfctr
OS Version	#1 SMP Mon Jun 29 12:32:22 PDT 2009
Starting Timestamp	1271572078767980
TAU Architecture	x86_64
TAU Config	-c++=g++ -cc=gcc -fortran=gfortran -mpilib=/usr/nic/dev/intel/impi/3.2.2.006/lib64 -mpiinc=/usr/nic/dev/intel/impi/3.2.2.006/in...
TAU Makefile	/usr/corning/apps/paratools/tau-2.19.1/x86_64/lib/Makefile.tau-intelmpi-papi-mpi-pdt
TAU Version	tau-2.19.1
TAU_CALLPATH	off
TAU_CALLPATH_DEPTH	2
TAU_COMM_MATRIX	off
TAU_COMPENSATE	off
TAU_PROFILE	on
TAU_PROFILE_FORMAT	profile
TAU_THROTTLE	on
TAU_THROTTLE_NUMCALLS	100000
TAU_THROTTLE_PERCALL	10
TAU_TRACE	off
TAU_TRACK_HEADROOM	off
TAU_TRACK_HEAP	off
TAU_TRACK_MESSAGE	off
Timestamp	1271572078835079

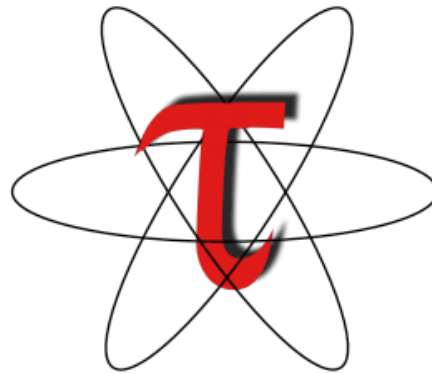
Expression: "PAPI_FP_INS"/"PAPI_L1_DCM"

Clear

+ - * / = () Apply



TAU IDE Integration



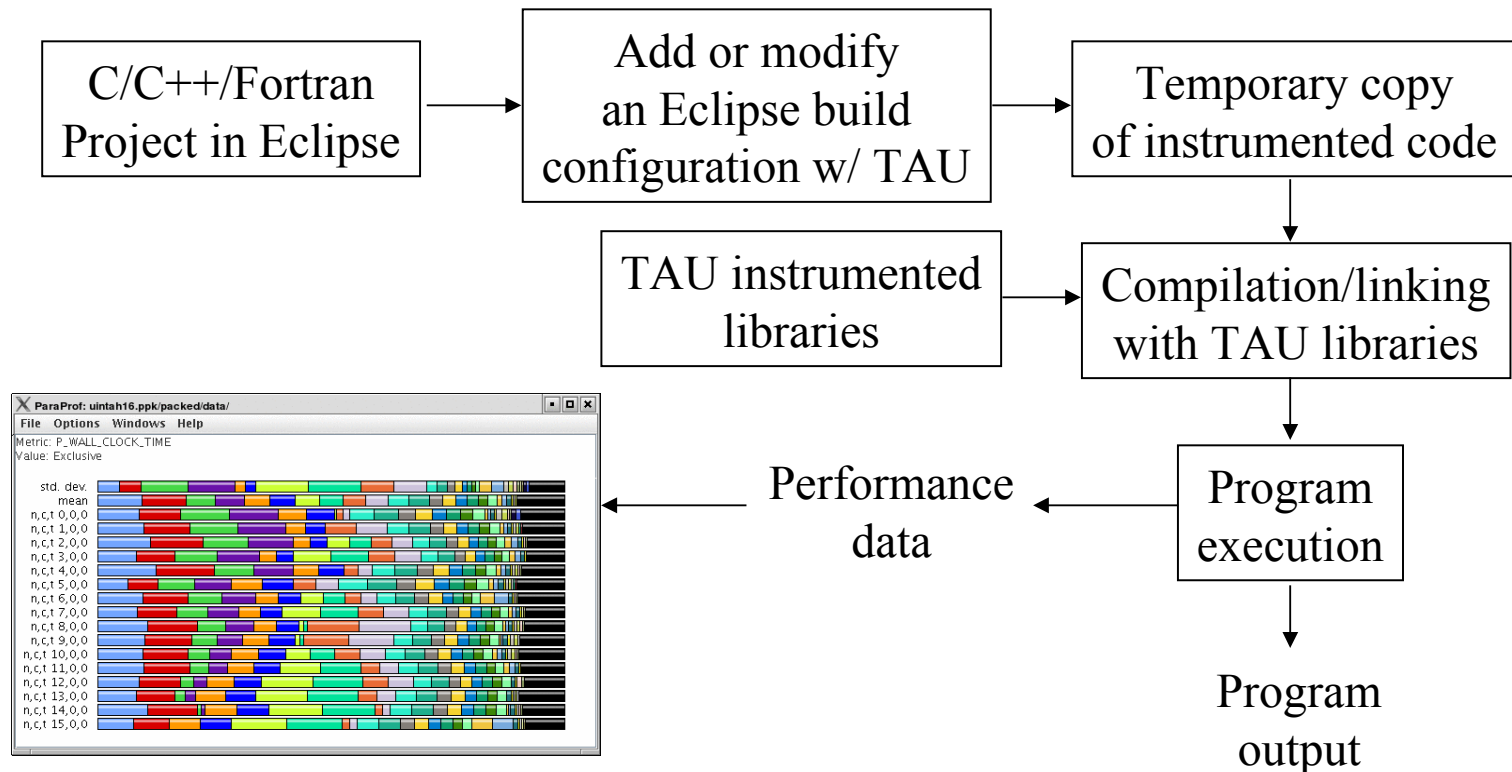
TAU IDE Integration

- 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

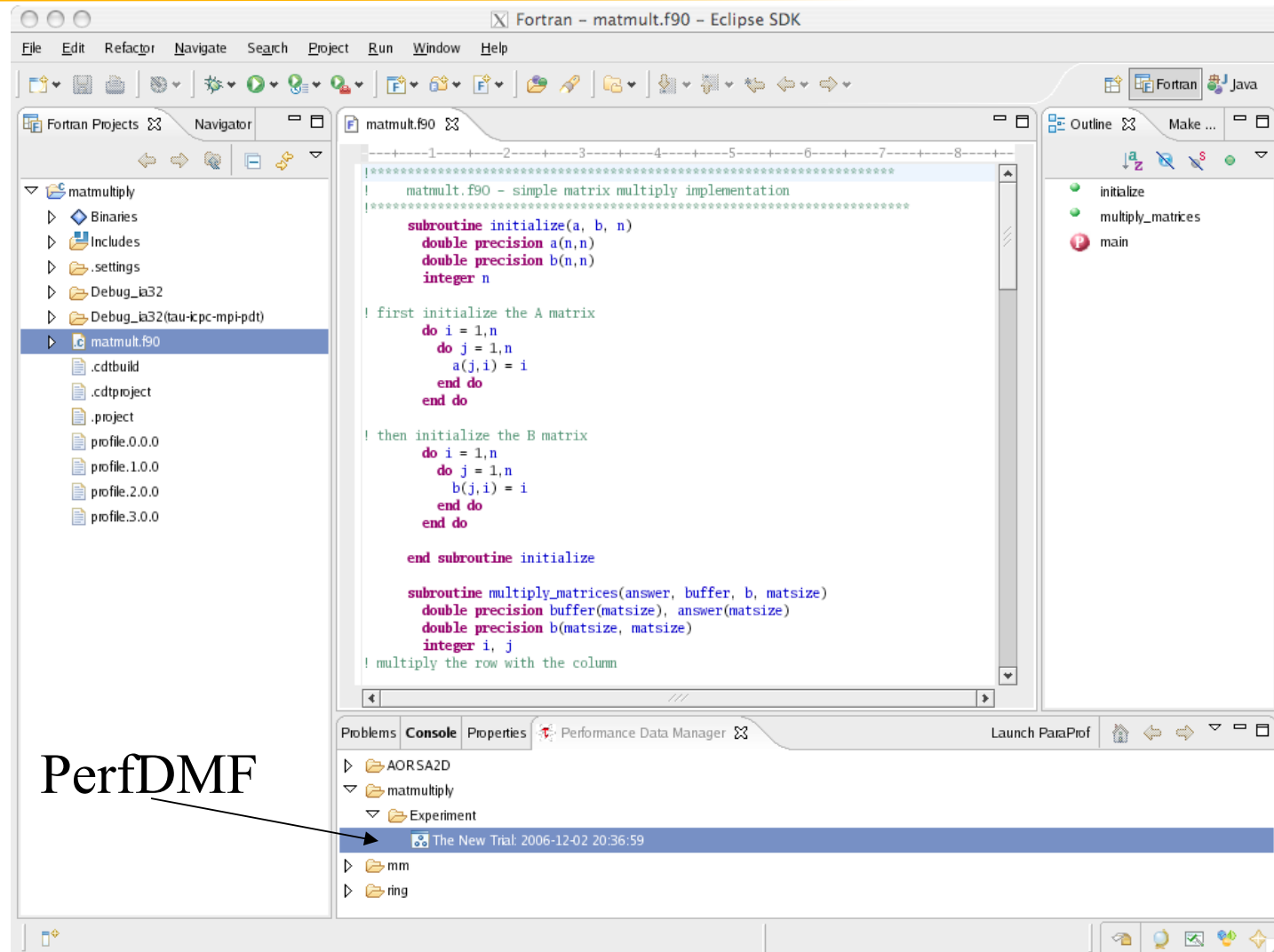


TAU and Eclipse

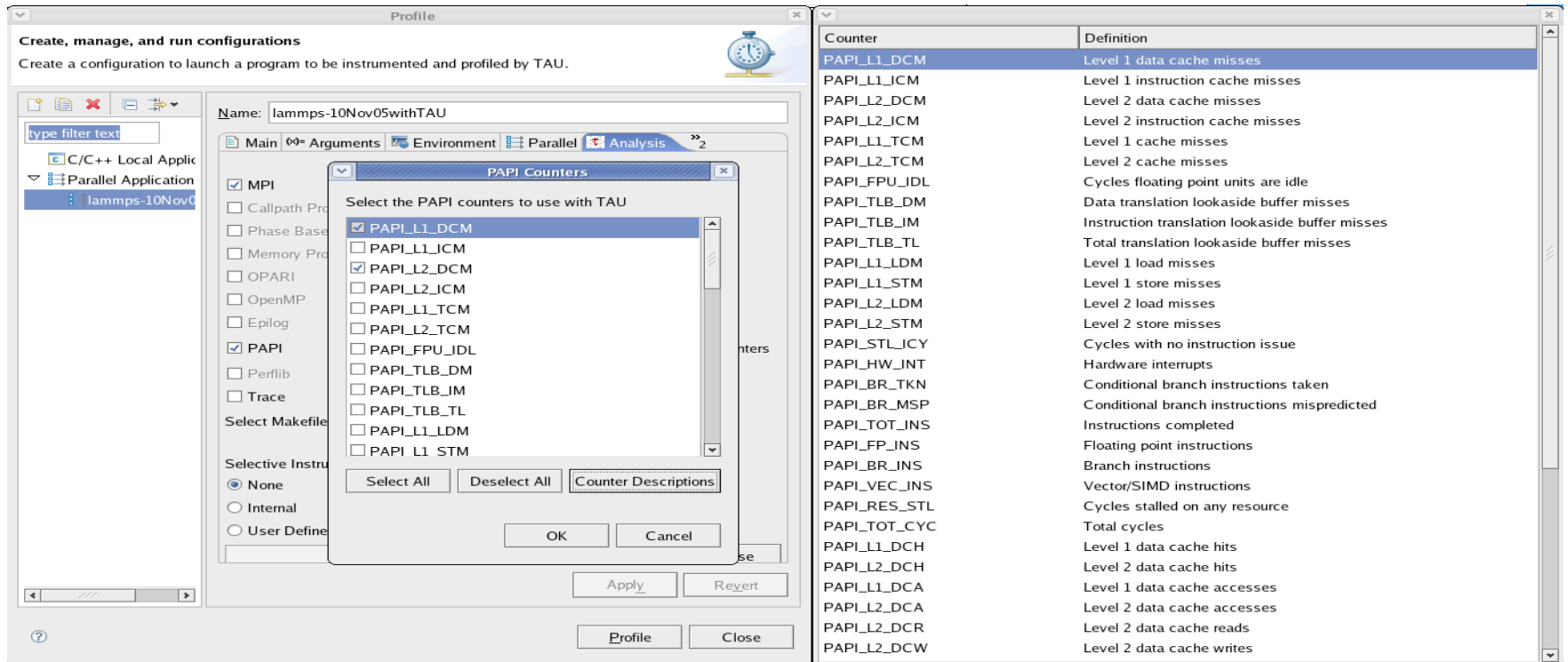
- Provide an interface for configuring TAU's automatic instrumentation within Eclipse's build system
- Manage runtime configuration settings and environment variables for execution of TAU instrumented programs



TAU and Eclipse



Choosing PAPI Counters with TAU in Eclipse



The screenshot shows the Eclipse IDE's 'Profile' window. The 'Name' field is set to 'lammps-10Nov05withTAU'. The 'Analysis' tab is selected. A 'PAPI Counters' dialog box is open, showing a list of counters to be selected. The 'PAPI' checkbox is checked. The 'PAPI Counters' dialog box has a list of counters with checkboxes. The 'PAPI' checkbox is checked. The 'PAPI Counters' dialog box has a list of counters with checkboxes. The 'PAPI' checkbox is checked.

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

% /soft/perftools/tau/eclipse/eclipse

Configuring Job Submission for Remote Execution on BGQ

Run Configurations
Create, manage, and run configurations
Create a configuration to launch a parallel application

Name: piBGQ

Resources | Application | Arguments | Environment | Synchronize | Common

Resource Manager: gov.anl.alcf.bgq.pbs.batch

Basic PBS Settings | Advanced PBS Settings | Import PBS Script

Name	Value	Description
Job Name:	ptp_job	The name assigned to the job by the qsub or qalter command.
Account:	TAU	Account to which to charge this job.
Queue:	default	Designation of the queue to which to submit the job.
Number of nodes:	1	Number and/or type of nodes to be reserved for exclusive use by the job.
Wallclock Time:	00:30:00	Maximum amount of real time during which the job can be in the running state.
BG/Q Mode:	c4	running mode (c1 - 1 MPI task/node, c2 - 2 MPI tasks/node, c4 - 4 MPI tasks/node)
Environment Variables:		Environment variables passed in the -env parameter to qsub

View Script | View Configuration | Restore Defaults

Filter matched 6 of 6 items

Apply | Revert

Close | Run



Job and System Display for BGQ Resource Manager

System Monitoring - piBGQ/src/piBGQ.f90 - Eclipse

File Edit Navigate Search Project Run Window Help

Resource Managers

gov.anl.alcf.bgq.pbs.batch (LML_JAXB)

Active Jobs

	step	owner	queue	wall	queuedate	dispatchdate	totalc	status
■	1473	hargrove	default	3600	2012-04-27 23:39:30	2012-04-27 23:39:44	2048	RUNNING
■	1473	hargrove	default	3600	2012-04-27 23:39:34	2012-04-27 23:39:57	2048	RUNNING
■	1473	hargrove	default	3600	2012-04-27 23:39:42	2012-04-27 23:40:23	2048	RUNNING

Further Information

Console Properties Problems Tasks

Key	Value
dispatchdate	2012-04-27 23:14:04
group	unknown
job_state	R
name	N512_R1_RRU_512_32
nodelist	R00-M0-N00-C00,R00-M0-N00-C01,R00-M0-N00-C04,R00-M0-N00-C05,R00-M0-N02-C00,R00-M0-N02-C01
owner	morozov
queue	default
queuedate	2012-04-27 23:13:53
state	Running

system: i2b.alcf.anl.g

R0

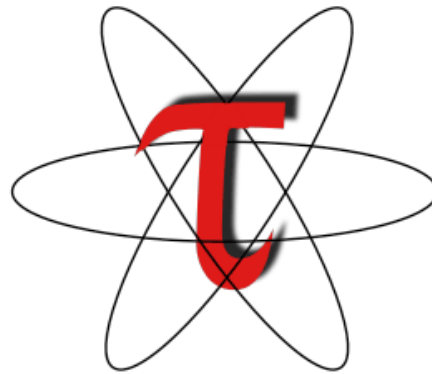
R00

R00-M1

R00-M0



Hands-on training with sample codes



Lab Instructions

Get `workshop.tar.gz` using:

```
% wget http://tau.uoregon.edu/workshop.tar.gz
```

Or

```
% tar xzf workshop.tar.gz
```

And follow the instructions in the README file.

For LiveDVD, see `~/workshop-point/README` and follow.



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
`% module load tau`
`% export TAU_MAKEFILE=`
`$TAULIBDIR/Makefile.tau-[options]`
3. If stub makefile has `-papi` in its name, set the `TAU_METRICS` environment variable:
`% export`
`TAU_METRICS=TIME:PAPI_L2_DCM:PAPI_TOT_CYC...`
4. Run:
`%qsub -I -l nodes=1:ppn=8 -X; mpirun -np 8 ./a.out`
5. Build and run workshop examples, then run
`O pprof/paraprof`

Support Acknowledgements

- Department of Energy (DOE)
 - Office of Science contracts
 - SciDAC contracts, LBL
 - LLNL-LANL-SNL ASC/NNSA contract
- Department of Defense (DoD)
 - PETTT, HPTi
- National Science Foundation (NSF)
 - POINT, SI-2
- University of Oregon
 - Dr. A. Malony, W. Spear, Dr. Lee, S. Biersdorff, S. Millstein, N. Chaimov
- University of Tennessee, Knoxville
 - Dr. Shirley Moore
- T.U. Dresden, GWT
 - Dr. Wolfgang Nagel and Dr. Andreas Knupfer
- Research Centre Juelich
 - Dr. Bernd Mohr, Dr. Felix Wolf

