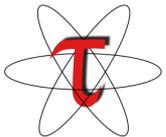


TAU PERFORMANCE SYSTEM

Wyatt Spear

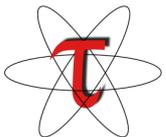
Sameer Shende, Allen D. Malony
Paratools Inc.



What Is Tau?

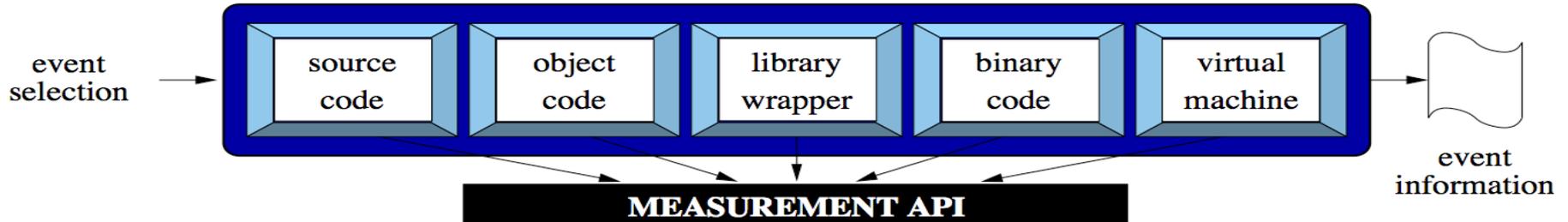


- Tuning and Analysis Utilities (15+ year project)
- Performance problem solving framework for HPC
 - Integrated, scalable, flexible, portable
 - Target all parallel programming / execution paradigms
- Integrated performance toolkit (open source)
 - Instrumentation, measurement, analysis, visualization
 - Widely-ported performance profiling / tracing system
 - Performance data management and data mining
- Broad application use (NSF, DOE, DOD, ...)

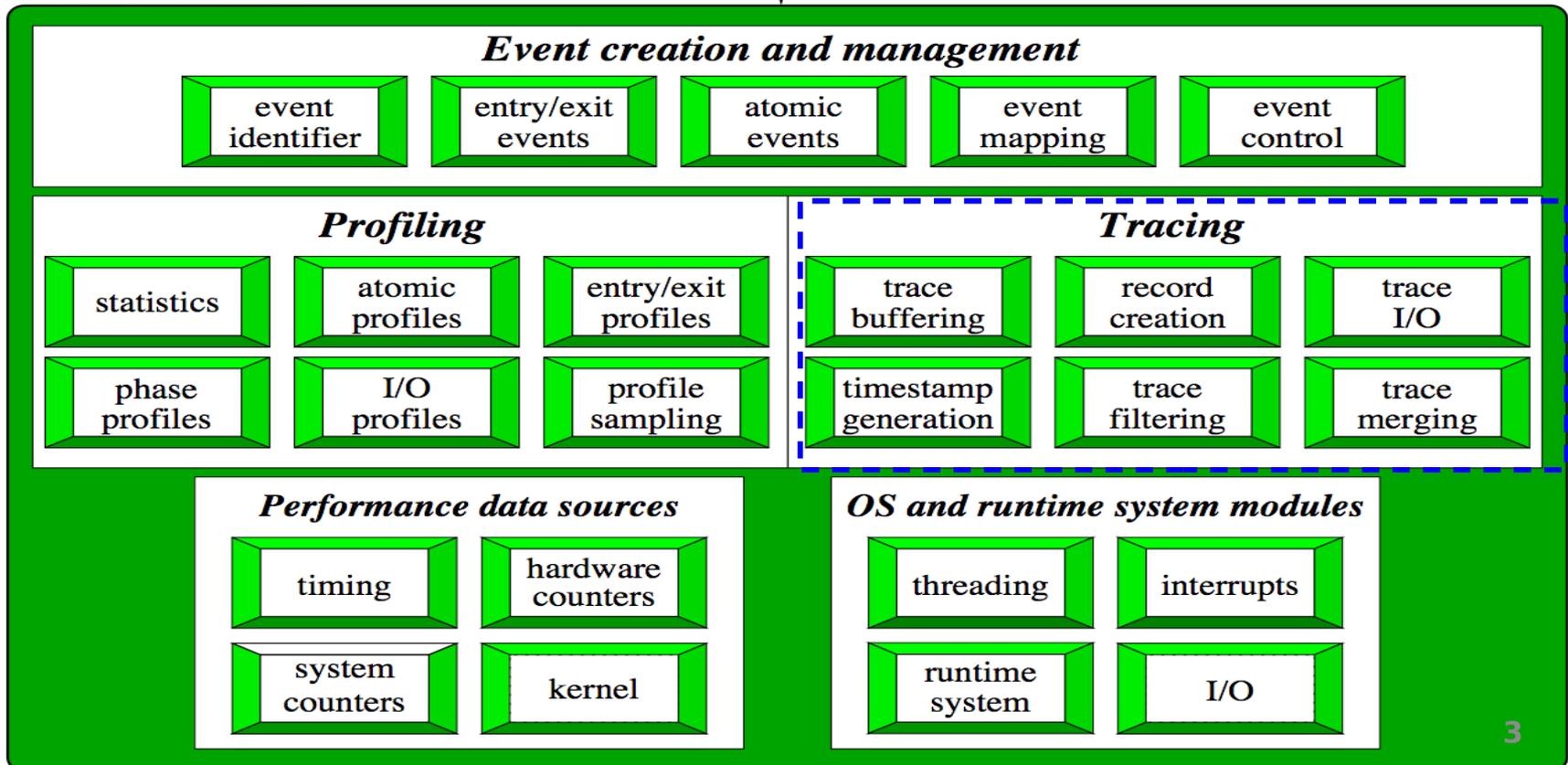


TAU Instrumentation / Measurement

Instrumentation

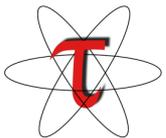


Measurement



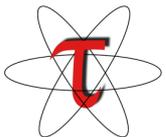
TAU Measurement Approach

- Portable and scalable parallel profiling solution
 - Multiple profiling types and options
 - Event selection and control (enabling/disabling, throttling)
 - Online profile access and sampling
 - Online performance profile overhead compensation
- Portable and scalable parallel tracing solution
 - Trace translation to OTF, EPILOG, Paraver, and SLOG2
 - Trace streams (OTF) and hierarchical trace merging
- Robust timing and hardware performance support
- Multiple counters (hardware, user-defined, system)
- Performance measurement of I/O and Linux kernel



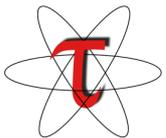
TAU Measurement Mechanisms

- Parallel profiling
 - Function-level, block-level, statement-level
 - Supports user-defined events and mapping events
 - Support for flat, callgraph/callpath, phase profiling
 - Support for parameter and context profiling
 - Support for tracking I/O and memory (library wrappers)
 - Parallel profile stored (dumped, shapshot) during execution
- Tracing
 - All profile-level events
 - Inter-process communication events
 - Inclusion of multiple counter data in traced events



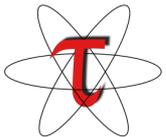
Types of Parallel Performance Profiling

- Flat profiles
 - Metric (e.g., time) spent in an event (callgraph nodes)
 - Exclusive/inclusive, # of calls, child calls
- Callpath profiles (Calldepth profiles)
 - Time spent along a calling path (edges in callgraph)
 - “main=> f1 => f2 => MPI_Send” (event name)
 - TAU_CALLPATH_DEPTH environment variable
- Phase profiles
 - Flat profiles under a phase (nested phases are allowed)
 - Default “main” phase
 - Supports static or dynamic (per-iteration) phases
 - Phase profiles may be generated from full callpath profiles in paraprof by choosing events as phases

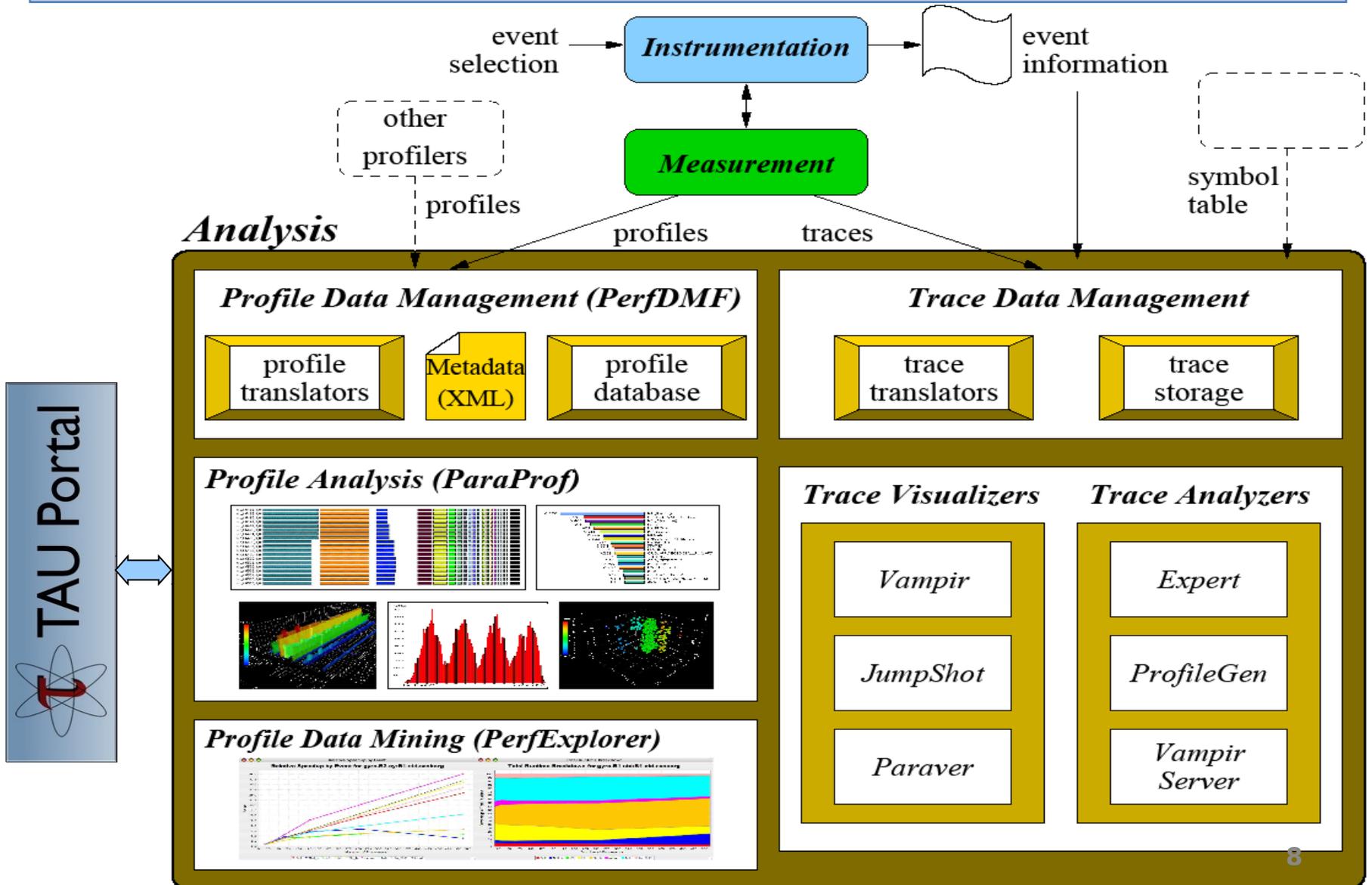


Direct Performance Observation

- Execution actions of interest exposed as events
 - In general, actions reflect some execution state
 - presence at a code location or change in data
 - occurrence in parallelism context (thread of execution)
 - Events encode actions for performance system to observe
- Observation is direct
 - Direct instrumentation of program (system) code (probes)
 - Instrumentation invokes performance measurement
 - Event measurement: performance data, meta-data, context
- Performance experiment
 - Actual events + performance measurements
- Contrast with (indirect) event-based sampling

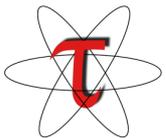


TAU Analysis

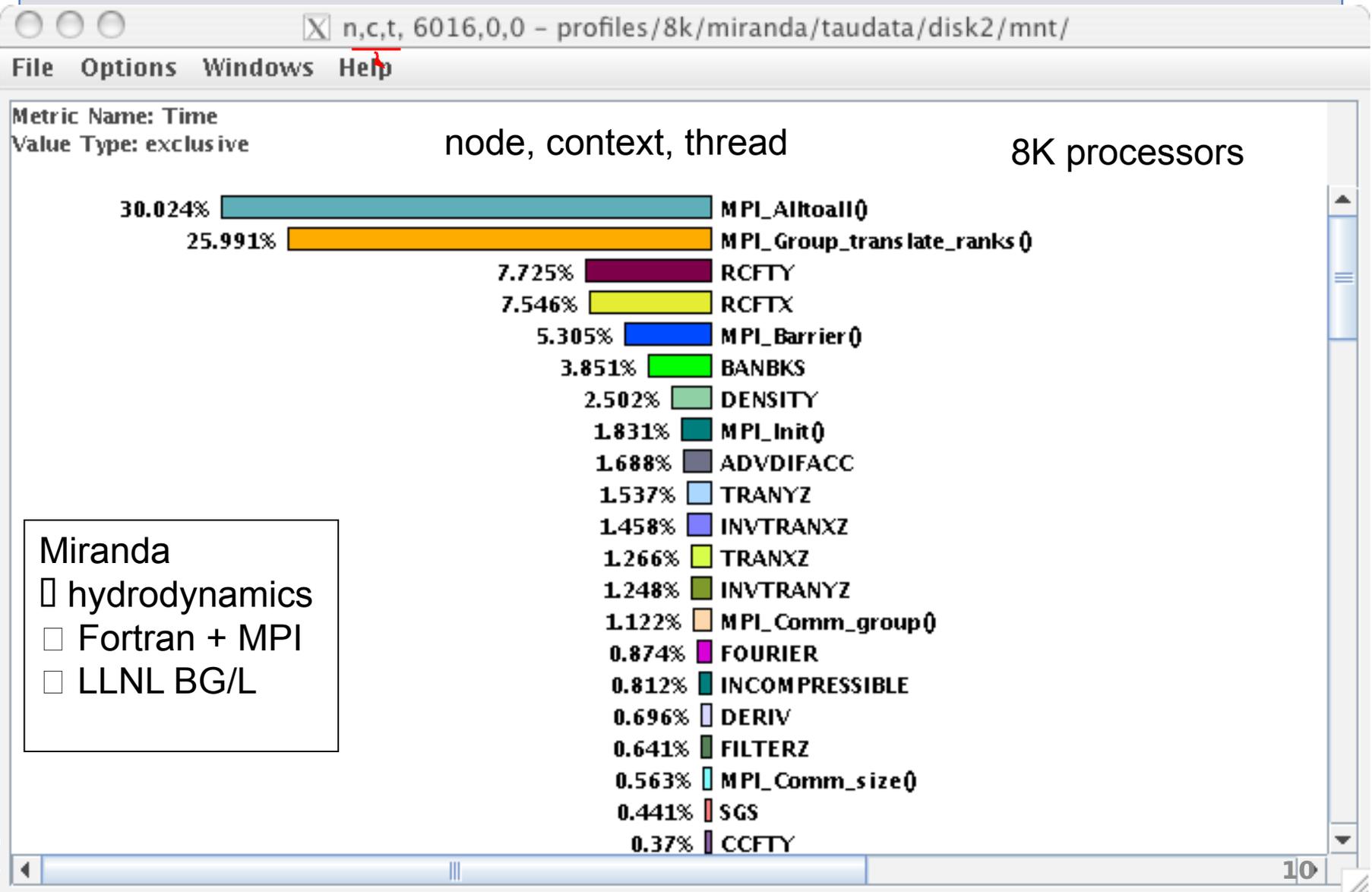


Performance Analysis

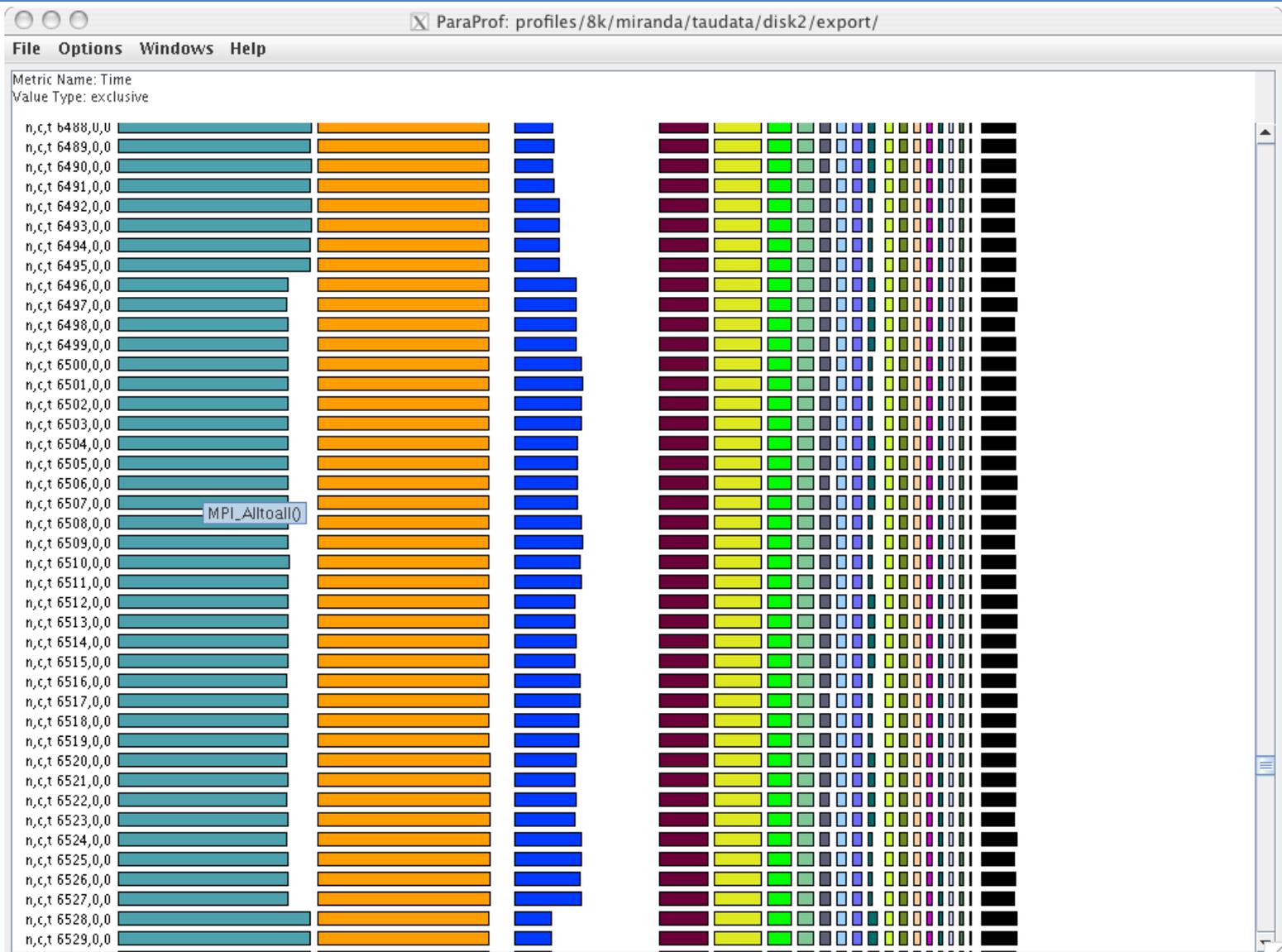
- Analysis of parallel profile and trace measurement
- Parallel profile analysis (ParaProf)
 - Java-based analysis and visualization tool
 - Support for large-scale parallel profiles
- Performance data management framework (PerfDMF)
- Parallel trace analysis
 - Translation to VTF (V3.0), EPILOG, OTF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



ParaProf - Flat Profile



ParaProf - Stacked View



Interval Events, Atomic Events in TAU

```

xterm
NODE 0;CONTEXT 0;THREAD 0:

```

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

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)

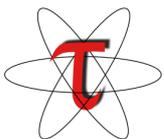
27.1

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()

Atomic event

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

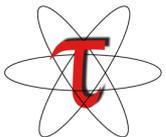
Context event = atomic event + executing context

```
% setenv TAU_CALLPATH_DEPTH 1
% setenv TAU_TRACK_HEAP 1
```



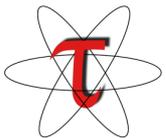
Configuring TAU

- TAU can measure several metrics with profiling and tracing approaches
- Different tools can also be invoked to instrument programs for TAU measurement
- Each configuration of TAU produces a measurement library for an architecture
- Each measurement configuration of TAU also creates a corresponding stub makefile that can be used to compile programs
- Typically configure multiple measurement libraries



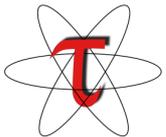
TAU Configuration - Examples

- Configure using PDT and MPI for BG/P
 - `./configure -arch=bgp -mpi`
 - pdt=/home/wspear/bin/pdtoolkit/ -BGPTIMERS
 - pdt_c++=xlc; make clean install
- Instrument hybrid OpenMP/MPI code with OPARI
 - `./configure -arch=bgp -mpi`
 - pdt=/home/wspear/bin/pdtoolkit/ -BGPTIMERS
 - pdt_c++=xlc -opari -openmp; make clean install
- Stub makefiles:
- `~/bin/tau2/Makefile.tau-bgptimers-mpi-pdt`
- `~/bin/Makefile.tau-bgptimers-mpi-pdt-openmp-opari`



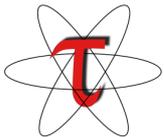
TAU Measurement Configuration

- `cd /soft/apps/tau/tau_latest/bgp/lib/; ls Makefile.tau-*`
- - `Makefile.tau-bgptimers-gnu-mpi-python-pdt`
- - `Makefile.tau-bgptimers-gnu-papi-mpi-pdt`
- - `Makefile.tau-bgptimers-gnu-papi-mpi-pdt-openmp-opari`
- - `Makefile.tau-bgptimers-mpi-pdt...`
- For an MPI+F90 application, you may want to start with:
 - `Makefile.tau-bgptimers-mpi-pdt`
 - Supports MPI instrumentation & PDT for automatic source instrumentation
- `export`
`TAU_MAKEFILE=/soft/apps/tau/tau_latest/bgp/lib/Makefile.ta`
`u-bgptimers-mpi-pdt`



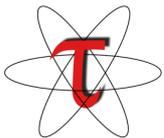
Using TAU: A brief Introduction

- To instrument source code using PDT
 - Choose an appropriate TAU stub makefile in <arch>/lib:
% setenv TAU_MAKEFILE
/opt/tau-2.19.1/x86_64/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)



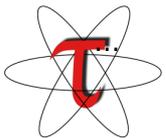
Application Build Environment

- Minimize impact on user's application build procedures
- Handle parsing, instrumentation, compilation, linking
- Dealing with Makefiles
 - Minimal change to application Makefile
 - Avoid changing compilation rules in application Makefile
 - No explicit inclusion of rules for process stages
- Some applications do not use Makefiles
 - Facilitate integration in whatever procedures used
- Two techniques:
 - TAU shell scripts (tau <compiler>.sh)
 - Invokes all PDT parser, TAU instrumenter, and compiler
 - TAU_COMPILER



Compile-Time Environment Variables

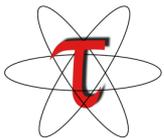
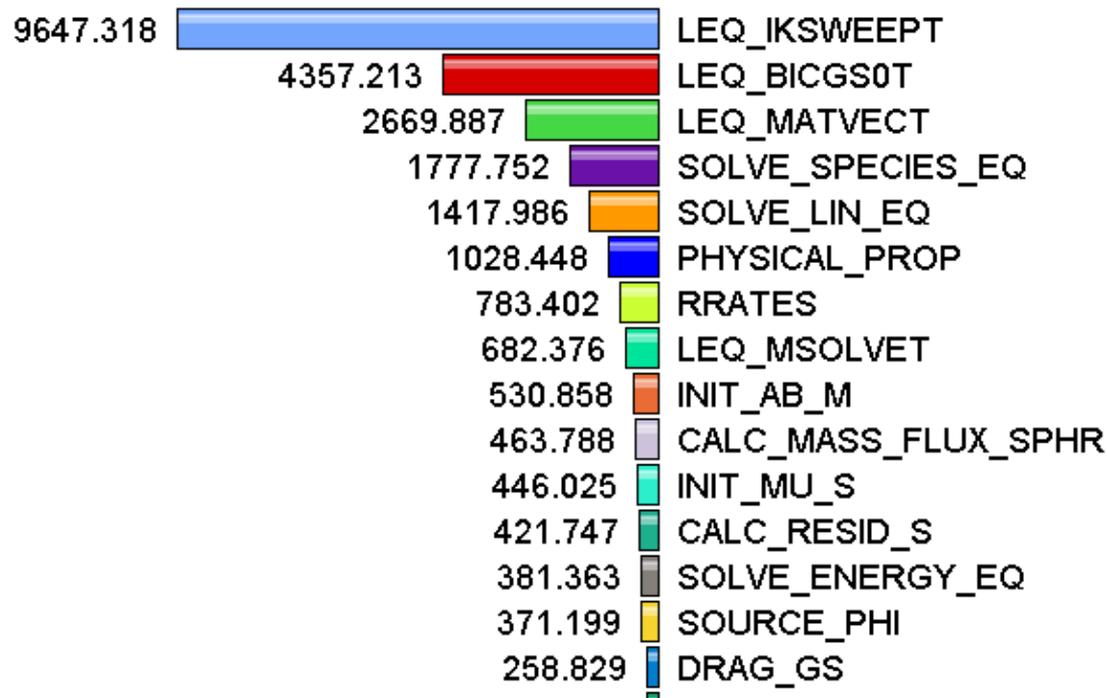
- Optional parameters for TAU_OPTIONS: [tau_compiler.sh -help]
- optVerbose Turn on verbose debugging messages
- optCompInst Use compiler based instrumentation
- optDetectMemoryLeaks Turn on debugging memory allocations/de-allocations to track leaks
- optKeepFiles Does not remove intermediate .pdb and .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)
- optTauSelectFile="" Specify selective instrumentation file for tau_instrumentor
- optNoCompInst Do not revert to compiler-based instrumentation if source instrumentation fails
- 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)



Usage Scenarios: Routine Level Profile

- **Goal: What routines account for the most time?**

How r Metric: P_VIRTUAL_TIME
Value: Exclusive
Units: seconds



Solution: Generating a flat profile with MPI

```
% setenv TAU_MAKEFILE /opt/tau-2.19.1/x86_64  
          /lib/Makefile.tau-mpi-pdt  
% set path=(/opt/tau-2.19.1/x86_64/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
```

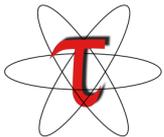
```
% paraprof
```

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
```



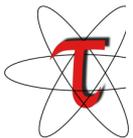
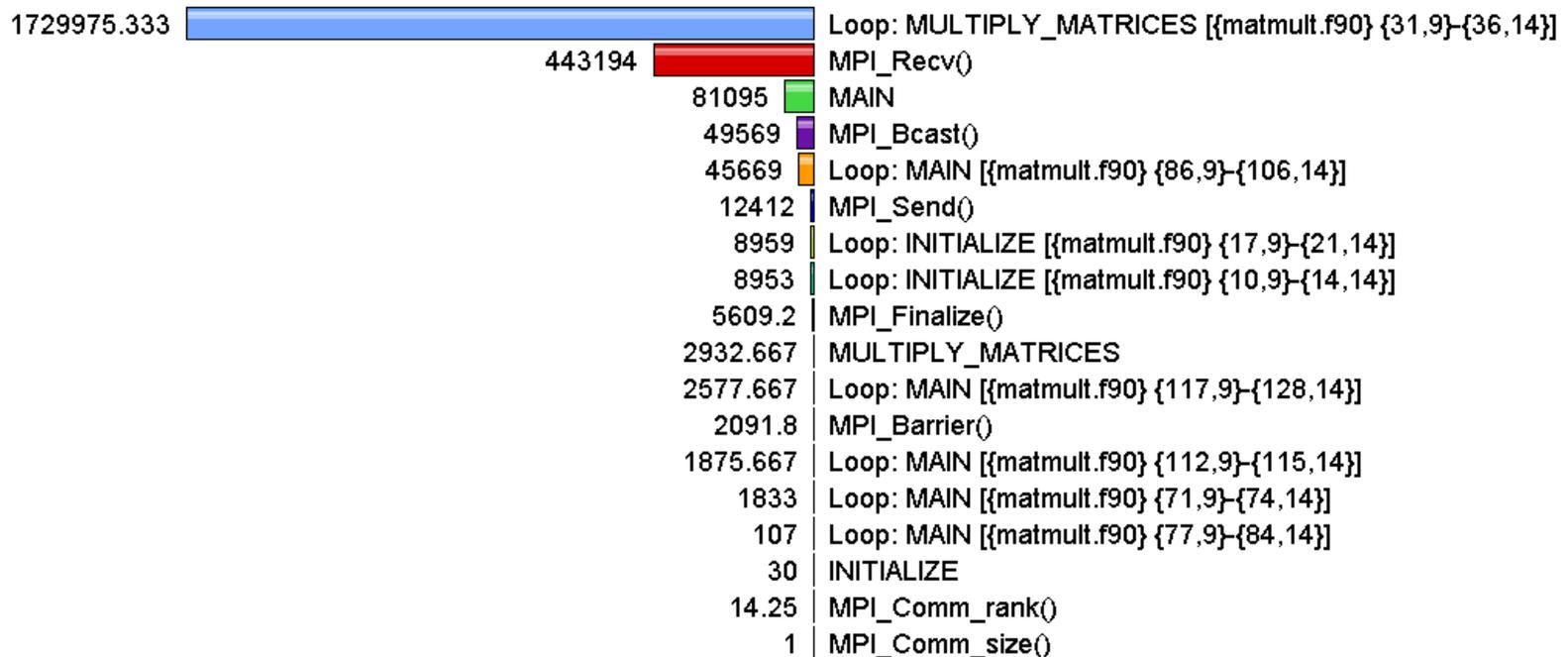
Usage Scenarios: Loop Level Instrumentation

- 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

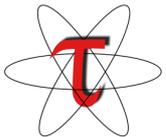


Solution: Generating a loop level profile

```
% setenv TAU_MAKEFILE /opt/tau-2.19.1/x86_64
                        /lib/Makefile.tau-mpi-pdt
% setenv TAU_OPTIONS '-optTauSelectFile=select.tau -optVerbose'
% cat select.tau
BEGIN_INSTRUMENT_SECTION
loops routine="#"
END_INSTRUMENT_SECTION

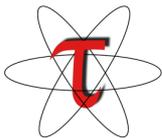
% module load tau
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% qsub run.job
% paraprof --pack app.ppk
Move the app.ppk file to your desktop.

% paraprof app.ppk
```



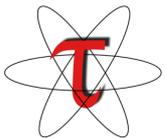
TAU Instrumentation Approach

- 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)
 - Flexible selection of event statistics
- Provides static events and dynamic events
- Enables “semantic” mapping
- Specification of event groups (aggregation, selection)
- Instrumentation optimization

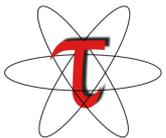
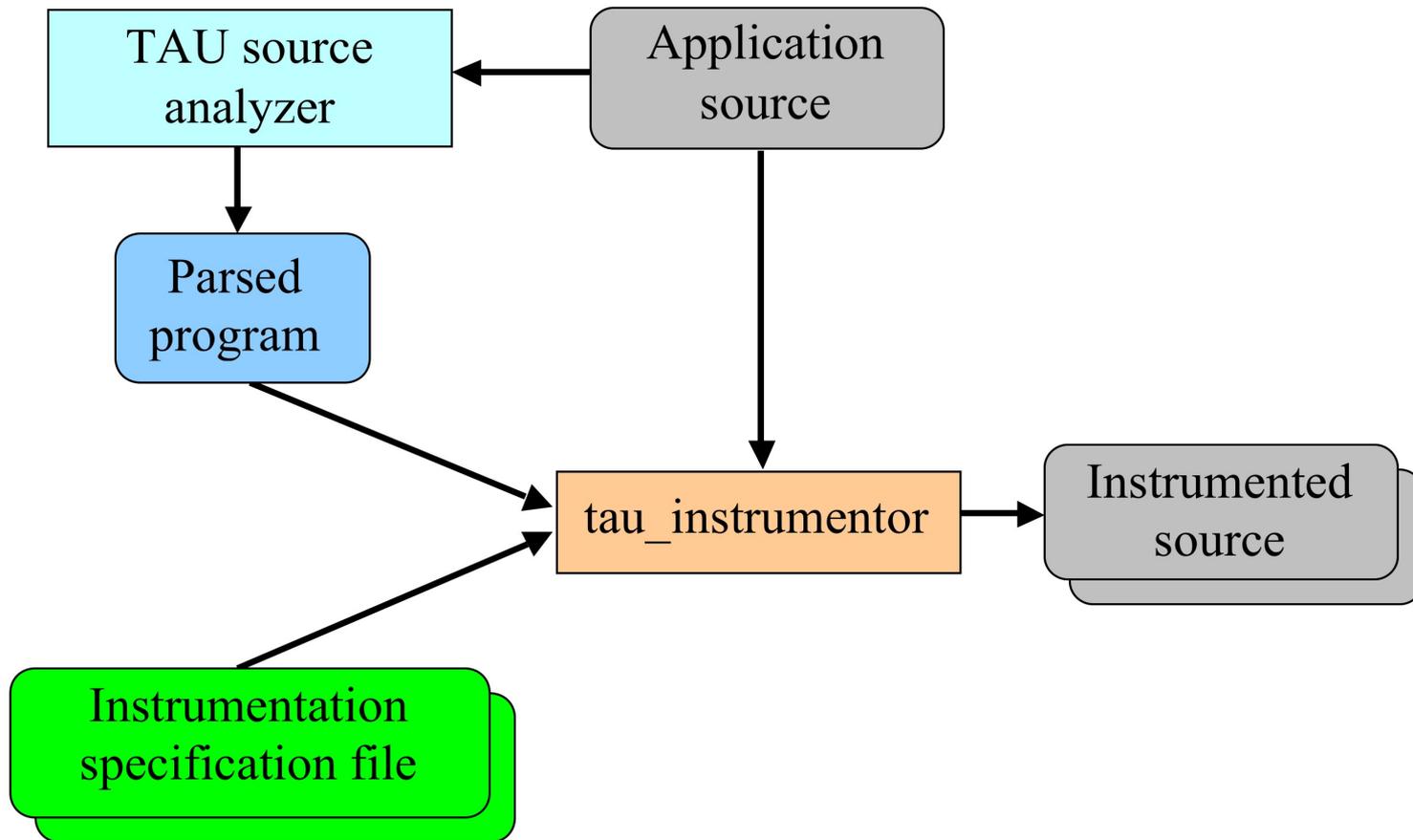


TAU Instrumentation Mechanisms

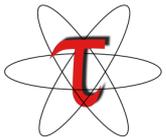
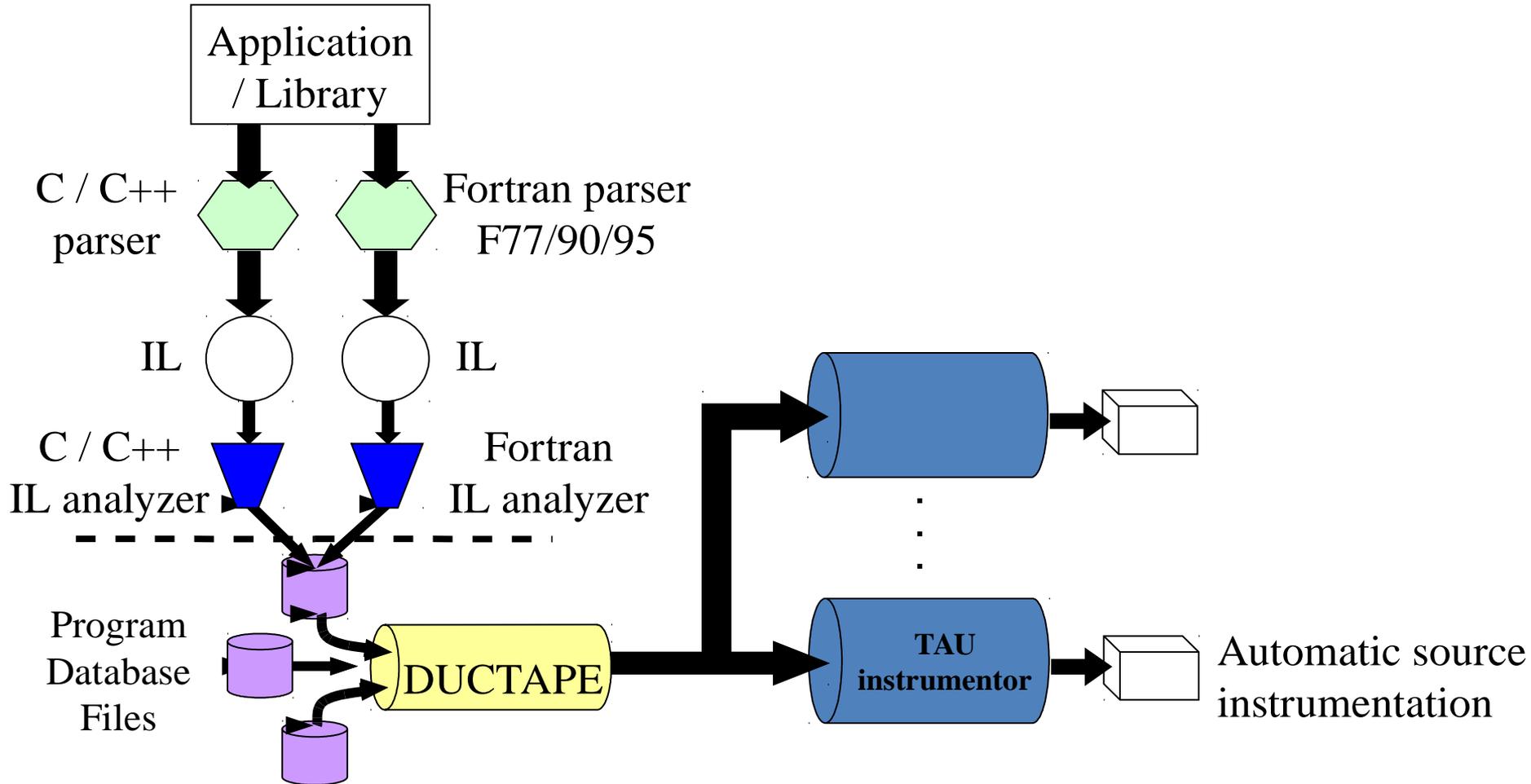
- Source code
 - Manual (TAU API, TAU component API)
 - Automatic (robust)
 - C, C++, F77/90/95 (Program Database Toolkit (PDT))
 - OpenMP (directive rewriting (Opari), POMP2 spec)
 - Library header wrapping
- Object code
 - Pre-instrumented libraries (e.g., MPI using PMPI)
 - Statically- and dynamically-linked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



Automatic Source-level Instrumentation



Program Database Toolkit (PDT)

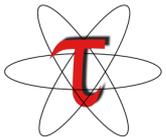


Selective Instrumentation File

- Specify a list of events to exclude or include
- # is a wildcard in a routine name

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

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



Selective Instrumentation File

- Optionally specify a list of files
- * and ? may be used as wildcard characters

```
BEGIN_FILE_EXCLUDE_LIST
```

```
f*.f90
```

```
Foo?.cpp
```

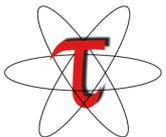
```
END_FILE_EXCLUDE_LIST
```

```
BEGIN_FILE_INCLUDE_LIST
```

```
main.cpp
```

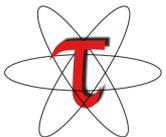
```
foo.f90
```

```
END_FILE_INCLUDE_LIST
```



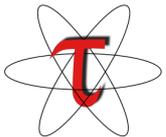
Selective Instrumentation File

- User instrumentation commands
 - Placed in INSTRUMENT section
 - 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
```



# MPI Wrapper Interposition Library

- Uses standard MPI Profiling Interface
  - Provides name shifted interface
  - MPI\_Send = PMPI\_Send
  - Weak bindings
- Create TAU instrumented MPI library
  - Interpose between MPI and TAU
  - Done during program link
  - -Impi replaced by -ITauMpi -lpmpi -Impi
  - No change to the source code!
  - Just re-link application to generate performance data



## MPI Shared Library Instrumentation

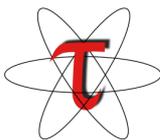
- Interpose the MPI wrapper library for applications that have already been compiled
  - Avoid re-compilation or re-linking
- Requires shared library MPI
  - Uses LD\_PRELOAD for Linux
  - On AIX use MPI\_EUILIB / MPI\_EUILIBPATH
  - Does not work on XT3
- Approach will work with other shared libraries
- Use TAU tauex
  - % mpirun -np 4 tauex a.out



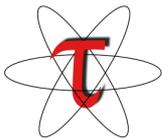
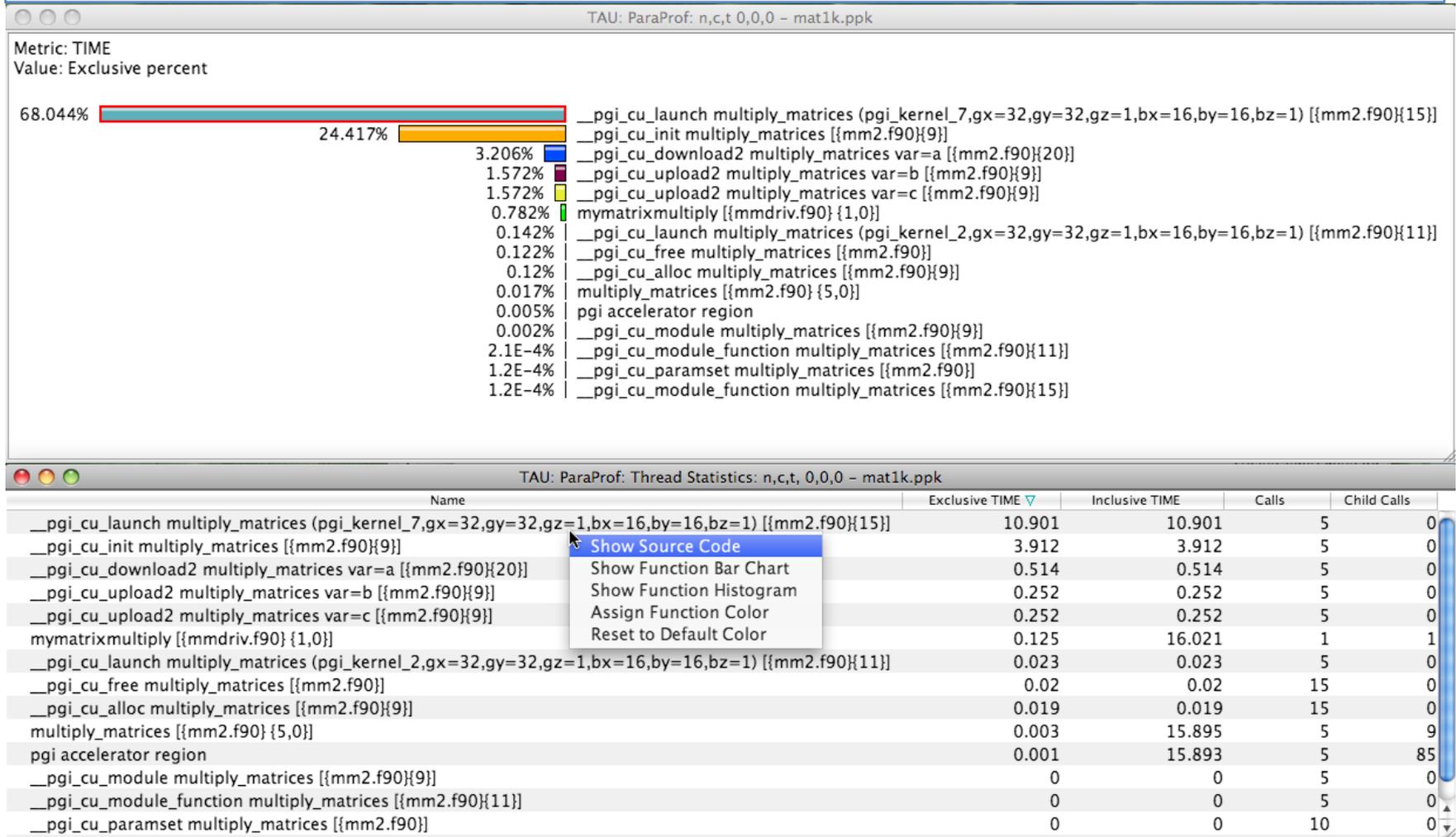
# Using tau\_exec

```
xterm
> cd ~/workshop-point/matmult
> mpif90 matmult.f90 -o matmult
> mpirun -np 4 ./matmult
>
> # To use tau_exec to measure the I/O and memory usage:
> mpirun -np 4 tau_exec -io -memory ./matmult
>
> # To measure memory leaks and get complete callpaths
> setenv TAU_TRACK_MEMORY_LEAKS 1
> setenv TAU_CALLPATH_DEPTH 100
> mpirun -np 4 tau_exec -io -memory ./matmult
> paraprof
> # Right click on a given rank (e.g., "node 2") and choose "Show Context Event
> # Window" and expand the ".TAU Application" node to see the callpath
> # To use a different configuration (e.g., Makefile.tau-papi-mpi-pdt)
> setenv TAU_METRICS TIME:PAPI_FP_INS:PAPI_L1_DCM
> mpirun -np 4 tau_exec -io -memory -T papi,mpi,pdt ./matmult
> # Using tau_exec with DyninstAPI:
> tau_run matmult -o matmult.i
> mpirun -np 4 tau_exec -io -memory ./matmult.i
>
> tau_run -XrunTAUsh-papi-mpi-pdt matmult -o matmult.i
> mpirun -np 4 tau_exec -io -memory -T papi,mpi,pdt ./matmult.i
> paraprof █
```

//



# Measuring Performance of PGI Accelerator Code



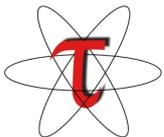
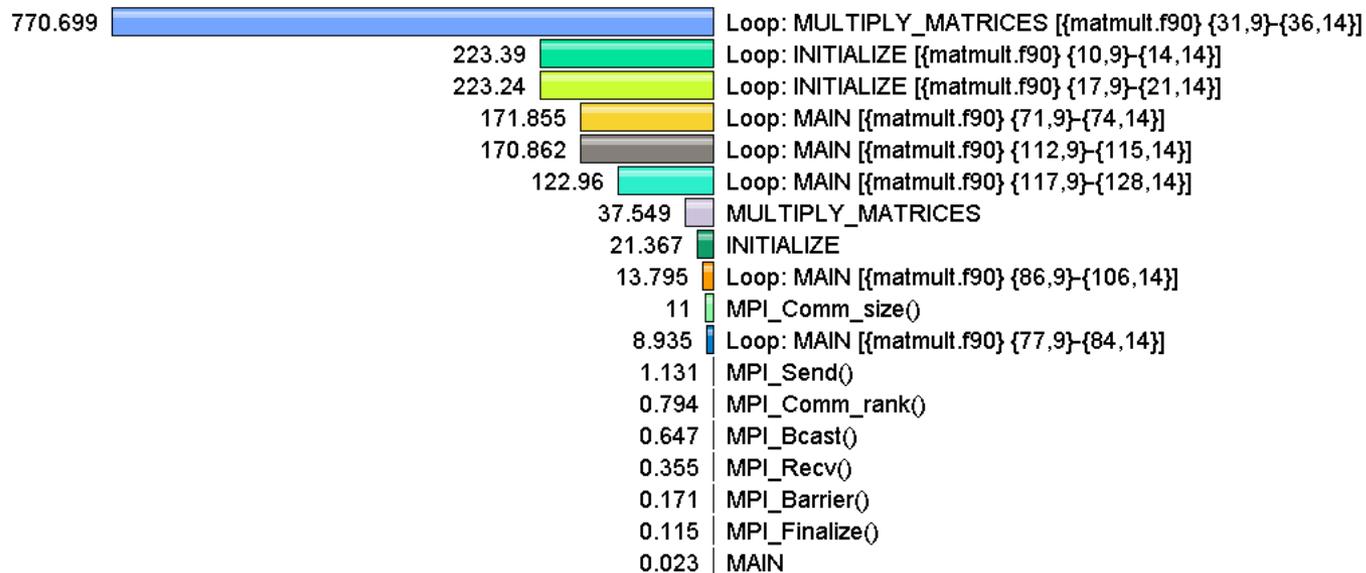
# Usage Scenarios: MFlops in Loops

- Goal: What execution rate do my application loops get in mflops?
- Flat profile with PAPI\_FP\_INS/OPS and time (-papi) with

Metric: PAPI\_FP\_INS / GET\_TIME\_OF\_DAY

Value: Exclusive

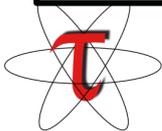
Units: Derived metric shown in microseconds format



# Generate a PAPI profile with 2 or more counters

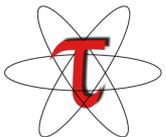
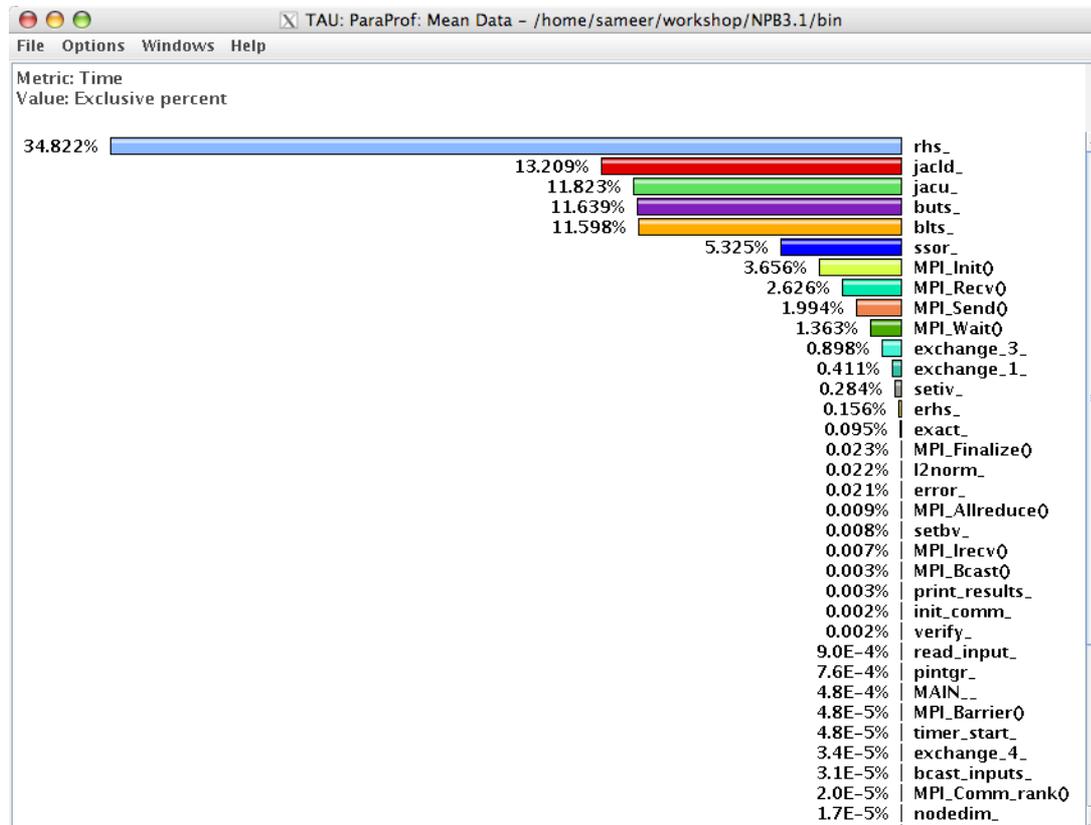
```
% setenv TAU_MAKEFILE /opt/tau-2.19.1/x86_64
 /lib/Makefile.tau-papi-mpi-pdt
% setenv TAU_OPTIONS '-optTauSelectFile=select.tau -optVerbose'
% cat select.tau
BEGIN_INSTRUMENT_SECTION
loops routine="#"
END_INSTRUMENT_SECTION

% 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

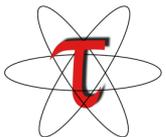
Goal: Easily generate routine level performance data using the compiler instead of PDT for parsing the source code



# Use Compiler-Based Instrumentation

```
% setenv TAU_MAKEFILE /opt/tau-2.19.1/x86_64
 /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
```



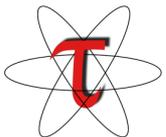
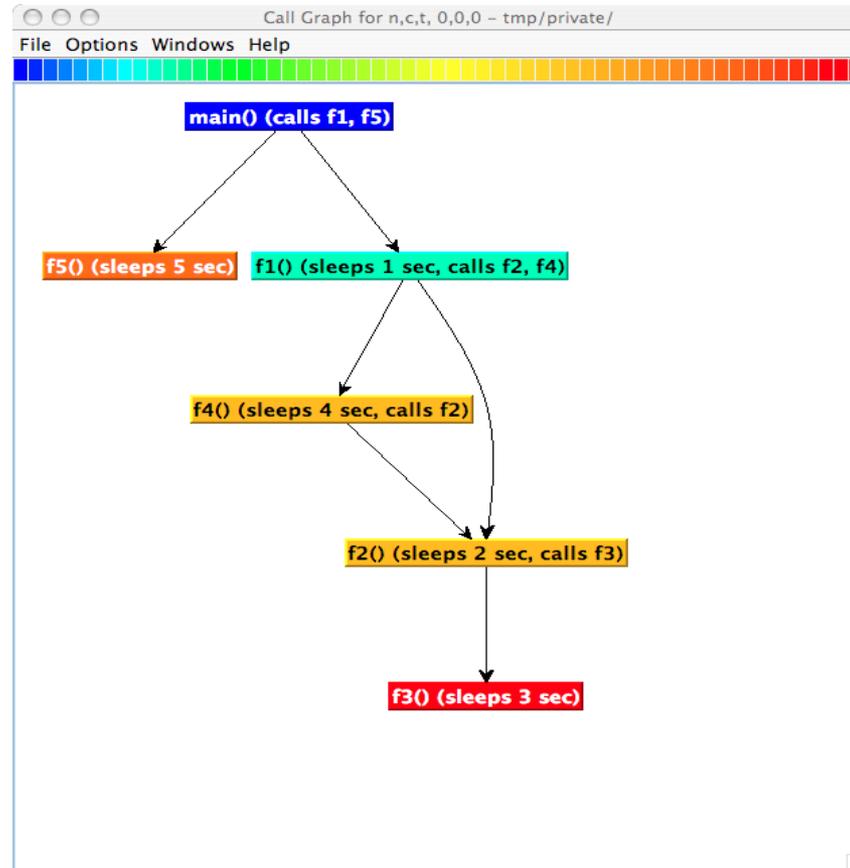
# -PROFILECALLPATH Option

- Generates profiles that show the calling order (edges and nodes in callgraph)
  - $A \Rightarrow B \Rightarrow C$  shows the time spent in C when it was called by B and B was called by A
  - Control the depth of callpath using `TAU_CALLPATH_DEPTH` environment variable
  - `-callpath` in the name of the stub Makefile name or setting `TAU_CALLPATH= 1` at runtime (TAU v2.18.1+)

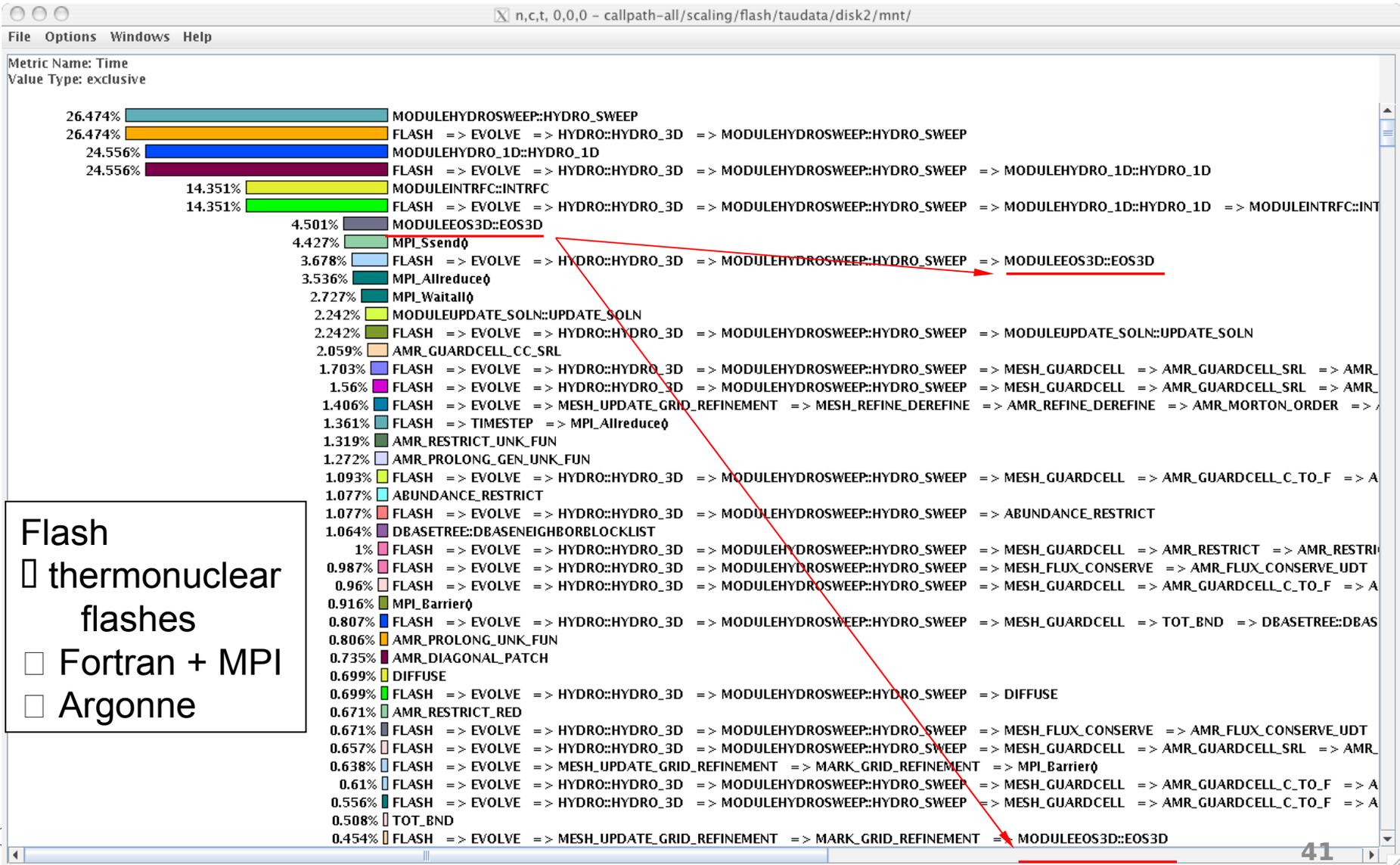


# Callpath Profile

- Generates program callgraph



# ParaProf - Callpath Profile



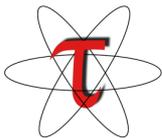
Flash  
 thermonuclear  
 flashes  
 Fortran + MPI  
 Argonne

# Generate a Callpath Profile

```
% setenv TAU_MAKEFILE /opt/tau-2.19.1/x86_64
 /lib/Makefile.tau-mpi-pdt
% set path=(/opt/tau-2.19.1/x86_64/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
```

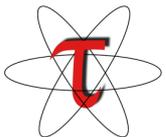
to generate the callpath profiles without any recompilation.

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



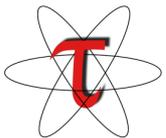
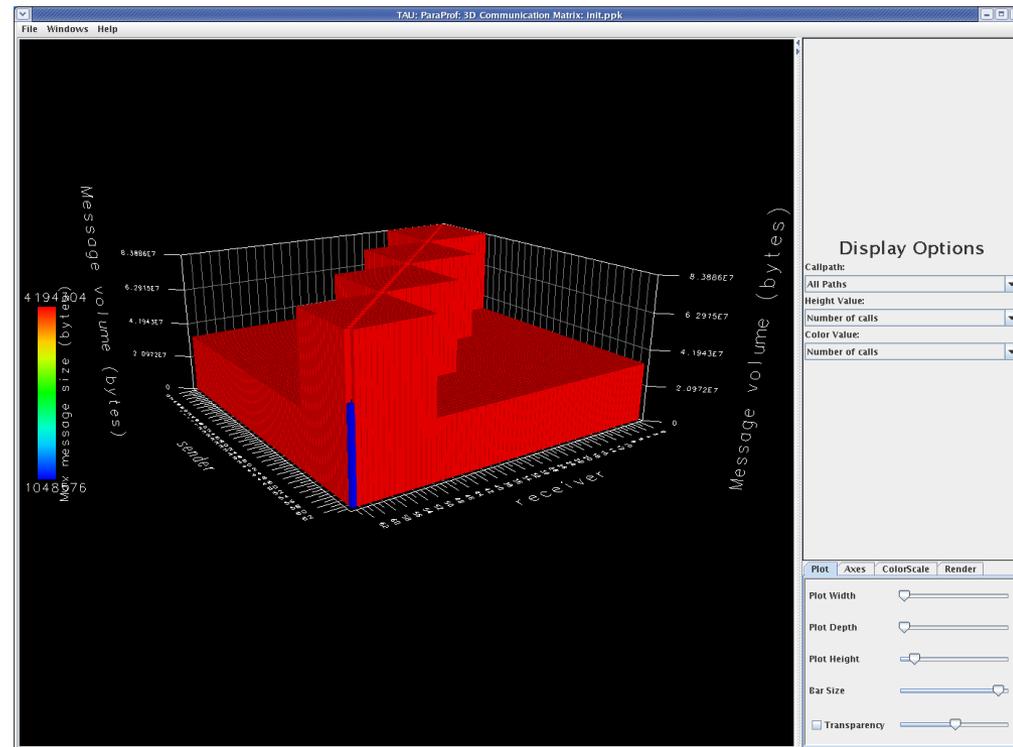
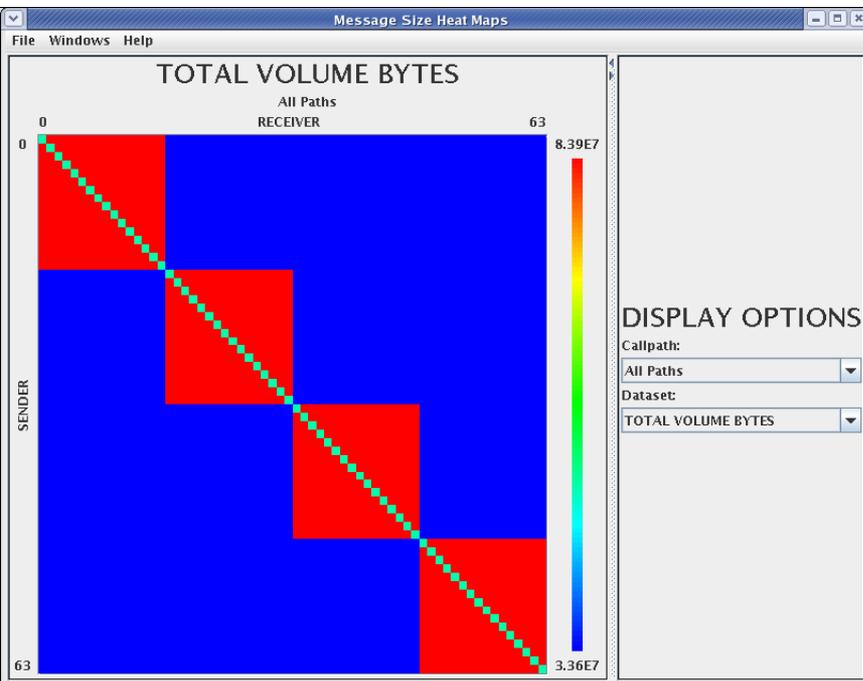
# -DEPTHLIMIT Option

- Allows users to enable instrumentation at runtime based on the depth of a calling routine on a callstack
  - Disables instrumentation in all routines a certain depth away from the root in a callgraph
- TAU\_DEPTH\_LIMIT environment variable specifies depth
  - % setenv TAU\_DEPTH\_LIMIT 1
    - enables instrumentation in only “main”
  - % setenv TAU\_DEPTH\_LIMIT 2
    - enables instrumentation in main and routines that are directly called by main
- Stub makefile has -depthlimit in its name:
  - setenv TAU\_MAKEFILE <taudir>/<arch>/lib/Makefile.tau-mpi-depthlimit-pdt



# Communication Matrix Display

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

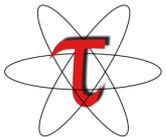


# Communication Matrix Profiles

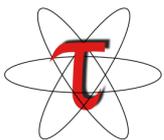
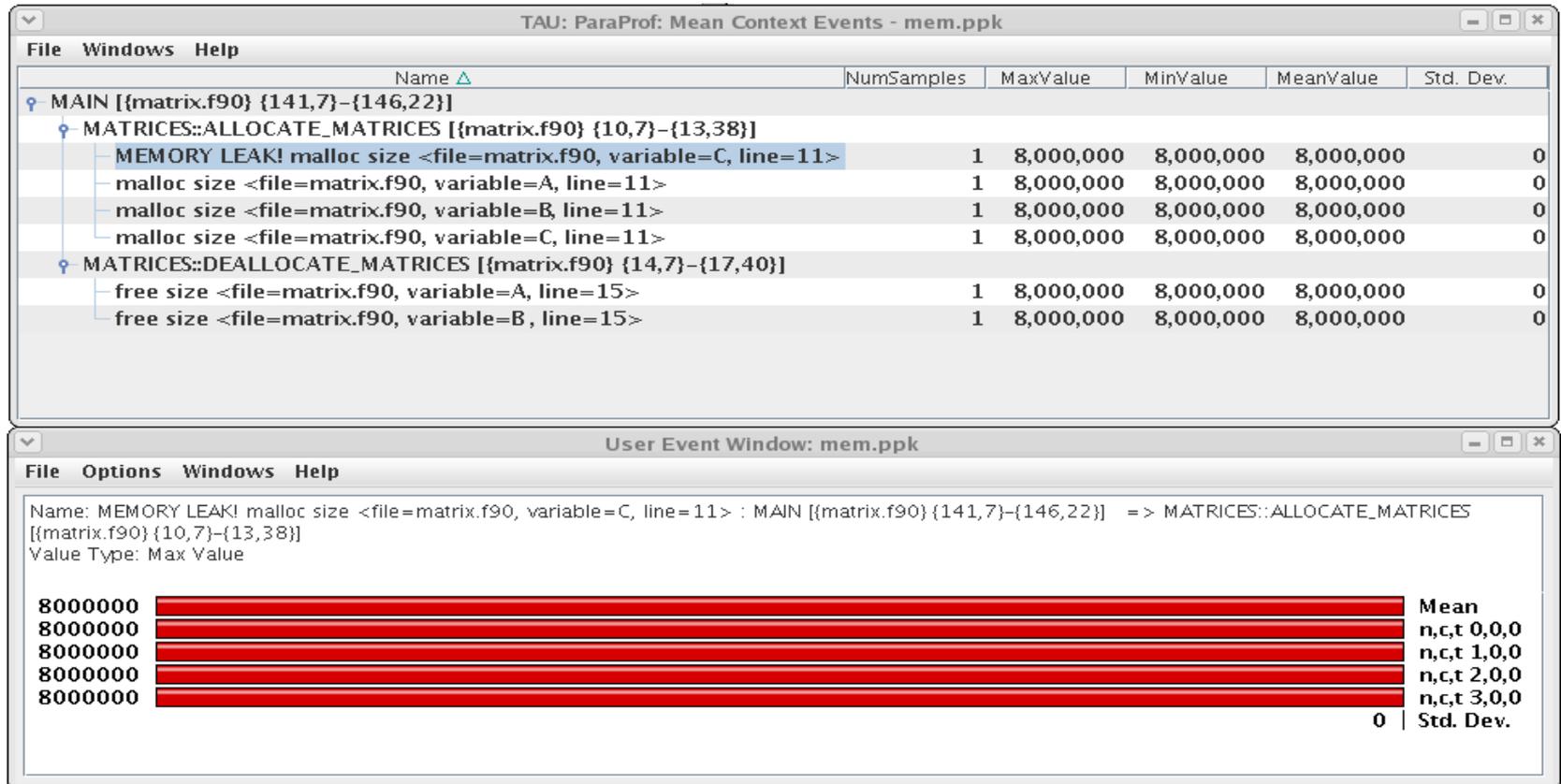
```
% export TAU_MAKEFILE = $TAU/Makefile.tau-mpi-pdt
% export PATH=/usr/local/packages/tau-2.19.1/x86_64/bin=$PATH
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% export TAU_COMM_MATRIX = 1

% qsub run.job (setting the environment variables)

% paraprof
(Windows -> Communication Matrix)
(Windows -> 3D Communication Matrix)
```



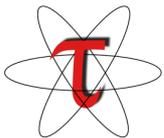
# Usage Scenario: Detect Memory Leaks



# Detect Memory Leaks

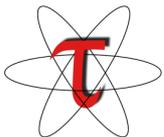
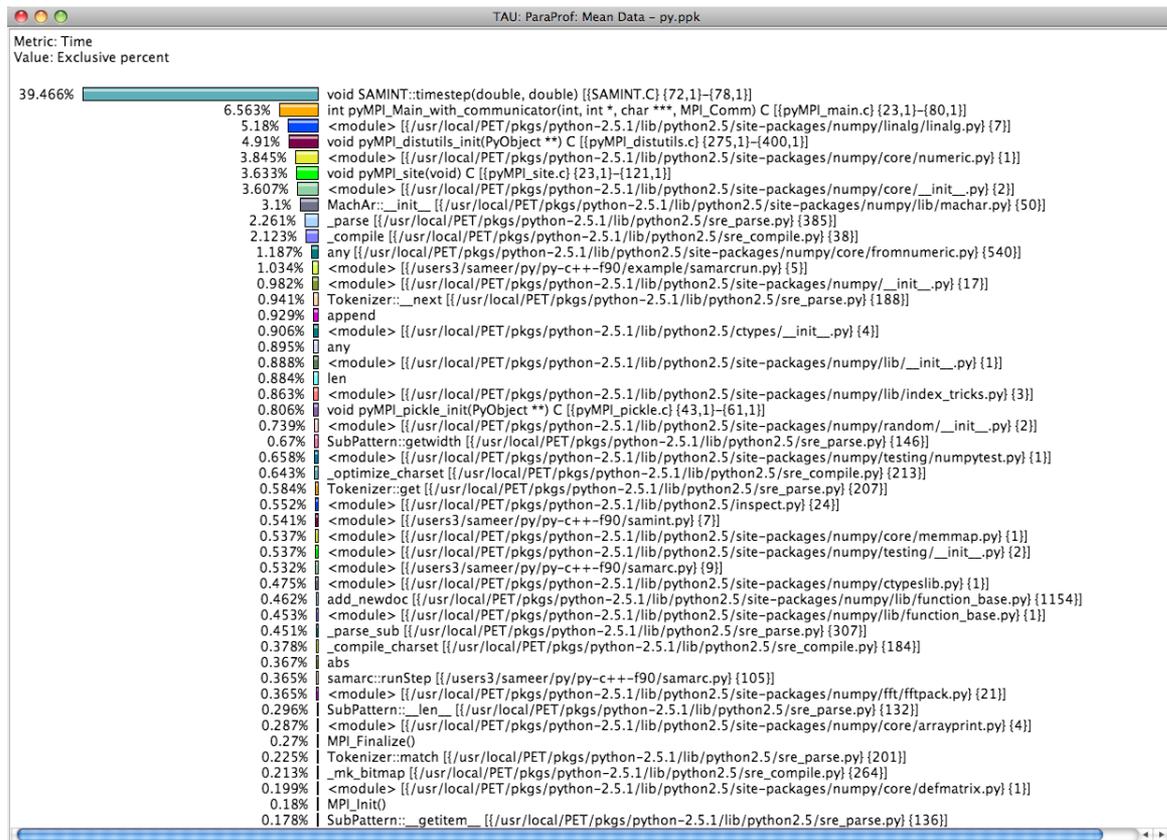
```
% setenv TAU_MAKEFILE /opt/tau-2.19.1/x86_64 /lib/Makefile.tau-mpi-pdt
% setenv TAU_OPTIONS '-optDetectMemoryLeaks -optVerbose'
% module load tau
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
% setenv TAU_CALLPATH_DEPTH 100

% 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)
NOTE: setenv TAU_TRACK_HEAP 1 and setenv TAU_TRACK_HEADROOM 1 may be used to track
heap and headroom utilization at the entry and exit of each routine.
TAU_CALLPATH_DEPTH=1 shows just the routine name, and 0 shows just one event for the
entire program.
```



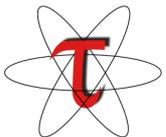
# Usage Scenarios: Mixed Python+F90+C+pyMPI

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



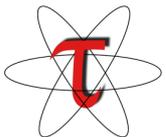
# Generate a Multi-Language Profile w/ Python

```
% setenv TAU_MAKEFILE /opt/tau-2.19.1/x86_64
 /lib/Makefile.tau-python-mpi-pdt
% set path=(/opt/tau-2.19.1/x86_64/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>/x86_64/lib/bindings-python-mpi-pdt-pgi
(same options string as TAU_MAKEFILE)
setenv LD_LIBRARY_PATH <taudir>/x86_64/lib/bindings-icpc-python-mpi-pdt-pgi\ :
$LD_LIBRARY_PATH
% poe <dir>/pyMPI-2.5b0-TAU/bin/pyMPI ./wrapper.py -procs 4
(Instrumented pyMPI with wrapper.py)
```



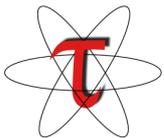
# Tracing in TAU

- Generates event-trace logs, rather than summary profiles
  - `setenv TAU_TRACE 1`
- Traces show when and where an event occurred in terms of location and the process that executed it
- Traces from multiple processes are merged:
  - `% tau_treemerge.pl`
  - generates `tau.trc` and `tau.edf` as merged trace and event definition file
- TAU traces can be converted to Vampir's OTF/VTF3, Jumpshot SLOG2, Paraver trace formats:
  - `% tau2otf tau.trc tau.edf app.otf`
  - `% tau2vtf tau.trc tau.edf app.vpt.gz`
  - `% tau2slog2 tau.trc tau.edf -o app.slog2`
  - `% tau_convert -paraver tau.trc tau.edf app.prv`

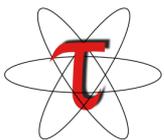
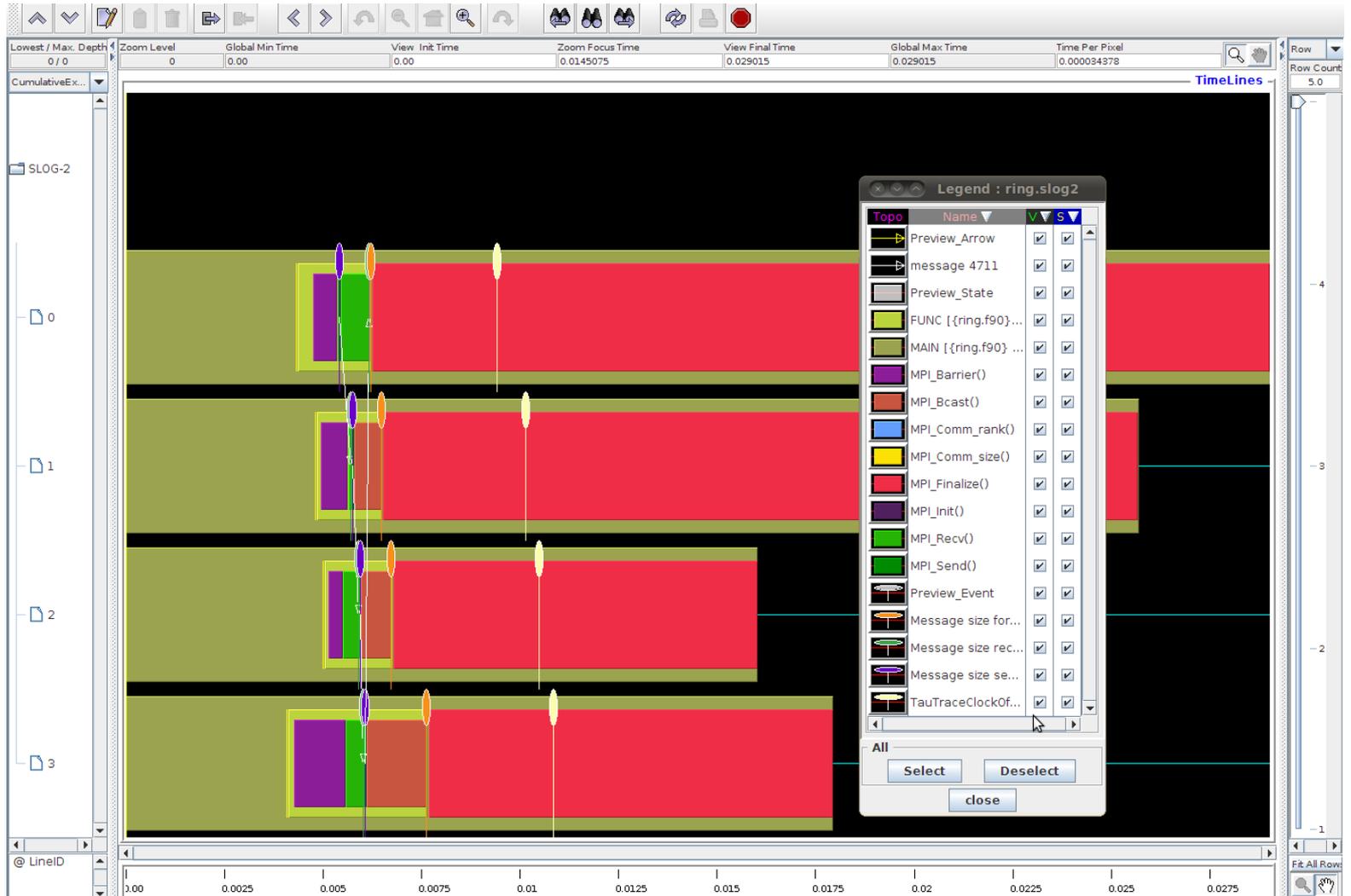


# Generate a Trace File

```
% setenv TAU_MAKEFILE /opt/tau-2.19.1/x86_64
 /lib/Makefile.tau-mpi-pdt
% set path=(/opt/tau-2.19.1/x86_64/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)
```

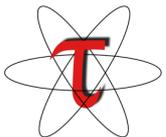


# Trace Visualization

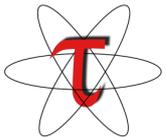
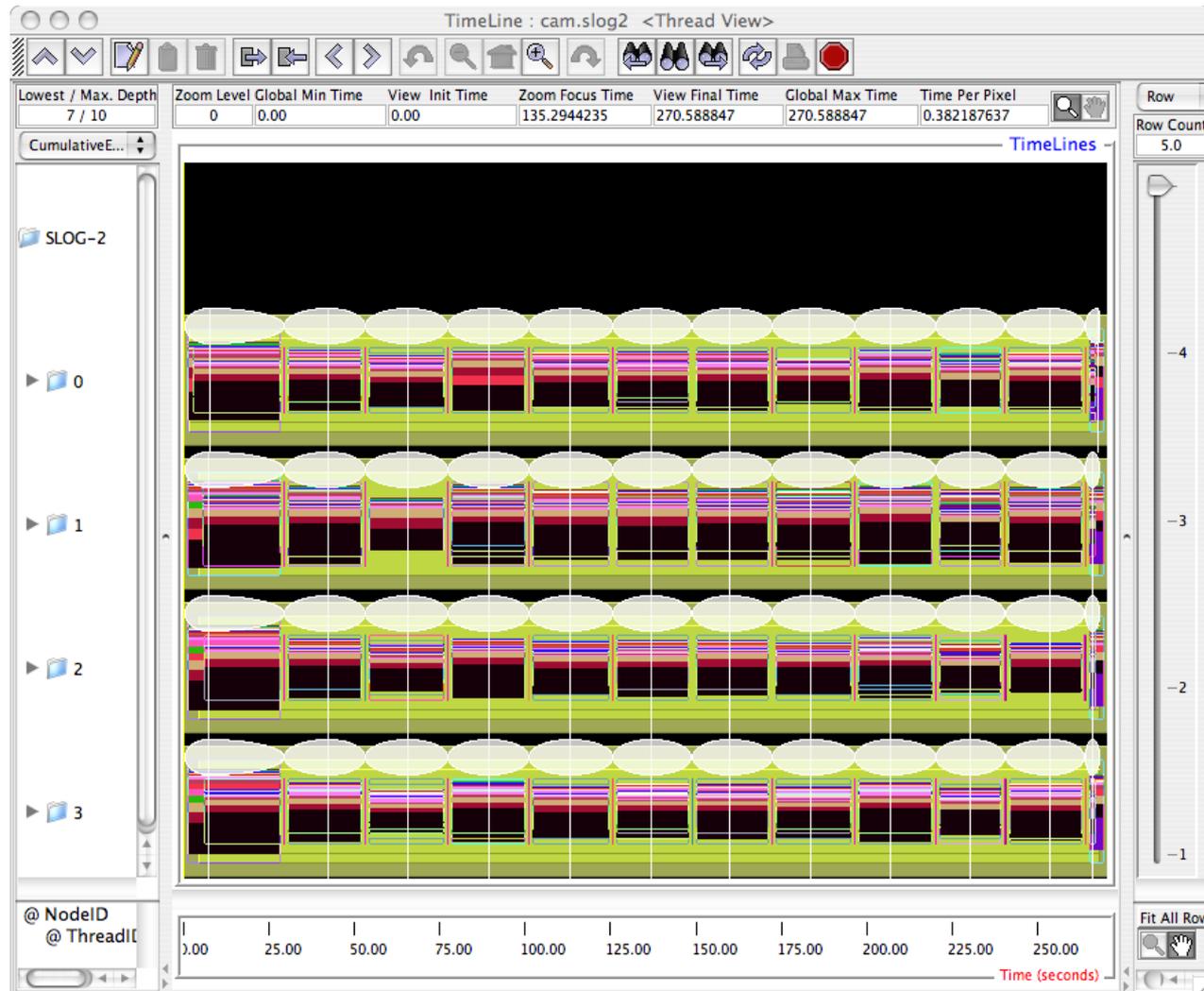


# Jumpshot

- <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
  - Installed on IBM BG/P
  - 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

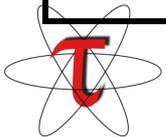


# Jumpshot



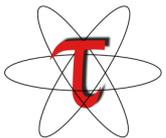
# Environment Variables in TAU

| 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_HEAP or<br>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.                                                                                                                 |
| TAU_SYNCHRONIZE_CLOCKS                  | 1       | Synchronize clocks across nodes to correct timestamps in traces                                                                              |
| 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>)                          |



# How to explain performance?

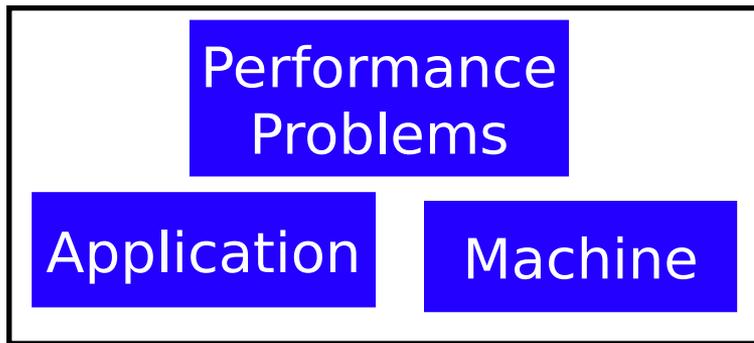
- Should not just redescribe the performance results
- Should explain performance phenomena
  - What are the causes for performance observed?
  - What are the factors and how do they interrelate?
  - Performance analytics, forensics, and decision support
- Need to add knowledge to do more intelligent things
  - Automated analysis needs good informed feedback
  - iterative tuning, performance regression testing
  - Performance model generation requires interpretation
- We need better methods and tools for
  - Integrating meta-information
  - Knowledge-based performance problem solving



# Role of Metadata and Knowledge Role

You have to capture these...

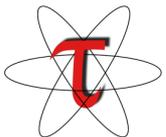
Context Knowledge



Performance Knowledge

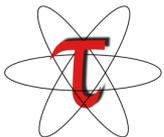
...to understand this

Performance Result



# Performance Data Management

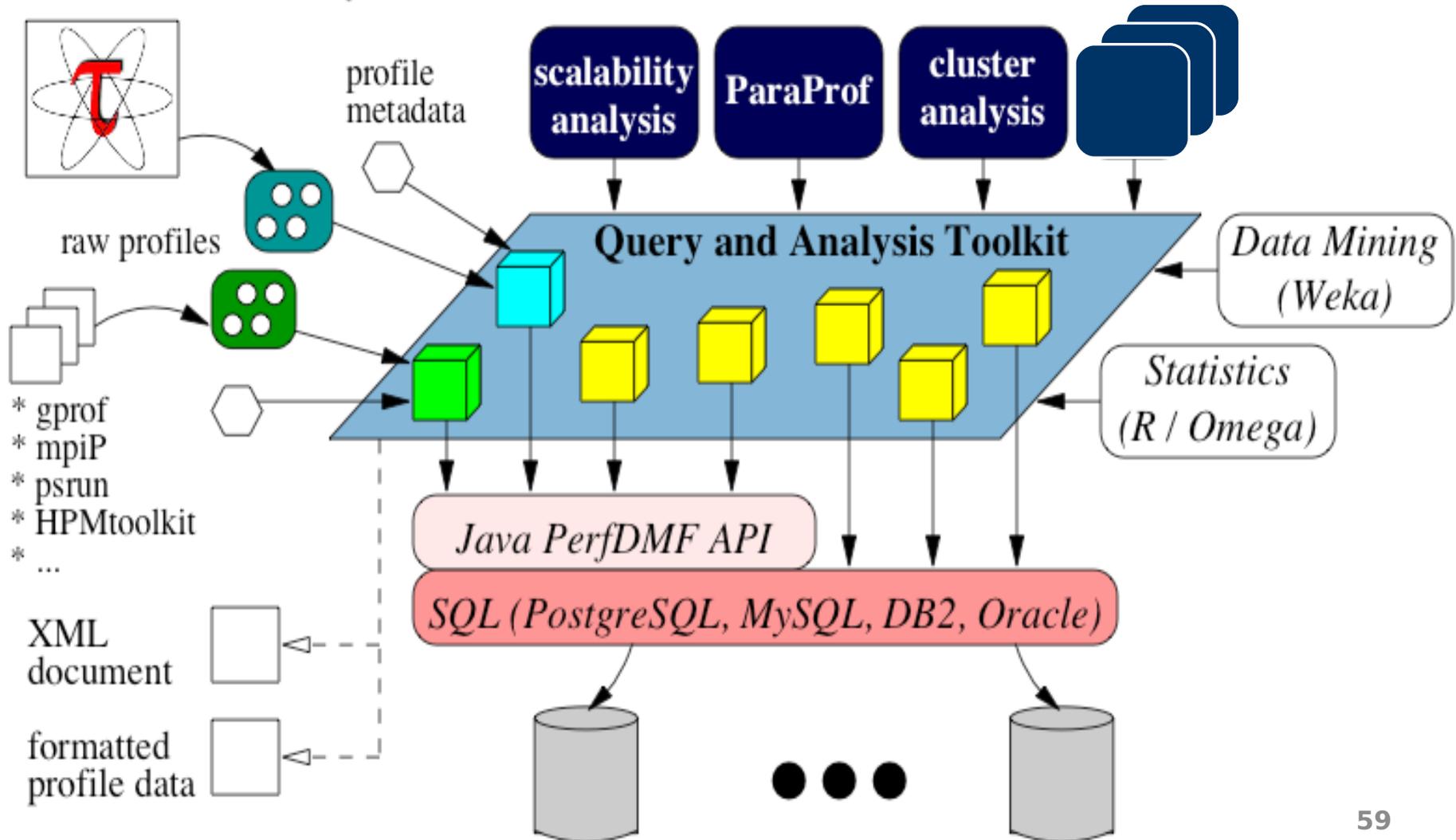
- Provide an open, flexible framework to support common data management tasks
  - Foster multi-experiment performance evaluation
- Extensible toolkit to promote integration and reuse across available performance tools (PerfDMF)
  - Originally designed to address critical TAU requirements
  - Supported profile formats:  
TAU, CUBE (Scalasca), HPC Toolkit (Rice), HPM Toolkit (IBM), gprof, mpiP, pstrun (PerfSuite), Open|SpeedShop, ...
  - Supported DBMS:  
PostgreSQL, MySQL, Oracle, DB2, Derby/Cloudscape
  - Profile query and analysis API
- Reference implementation for PERI-DB project



# PerfDMF Architecture

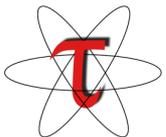
## TAU Performance System

## Performance Analysis Programs



# Metadata Collection

- Integration of XML metadata for each parallel 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



# Parallel Profile Analysis – pprof

emacs@neutron.cs.uoregon.edu

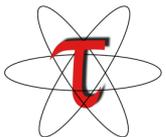
Buffers Files Tools Edit Search Mule Help

Reading Profile files in profile.\*

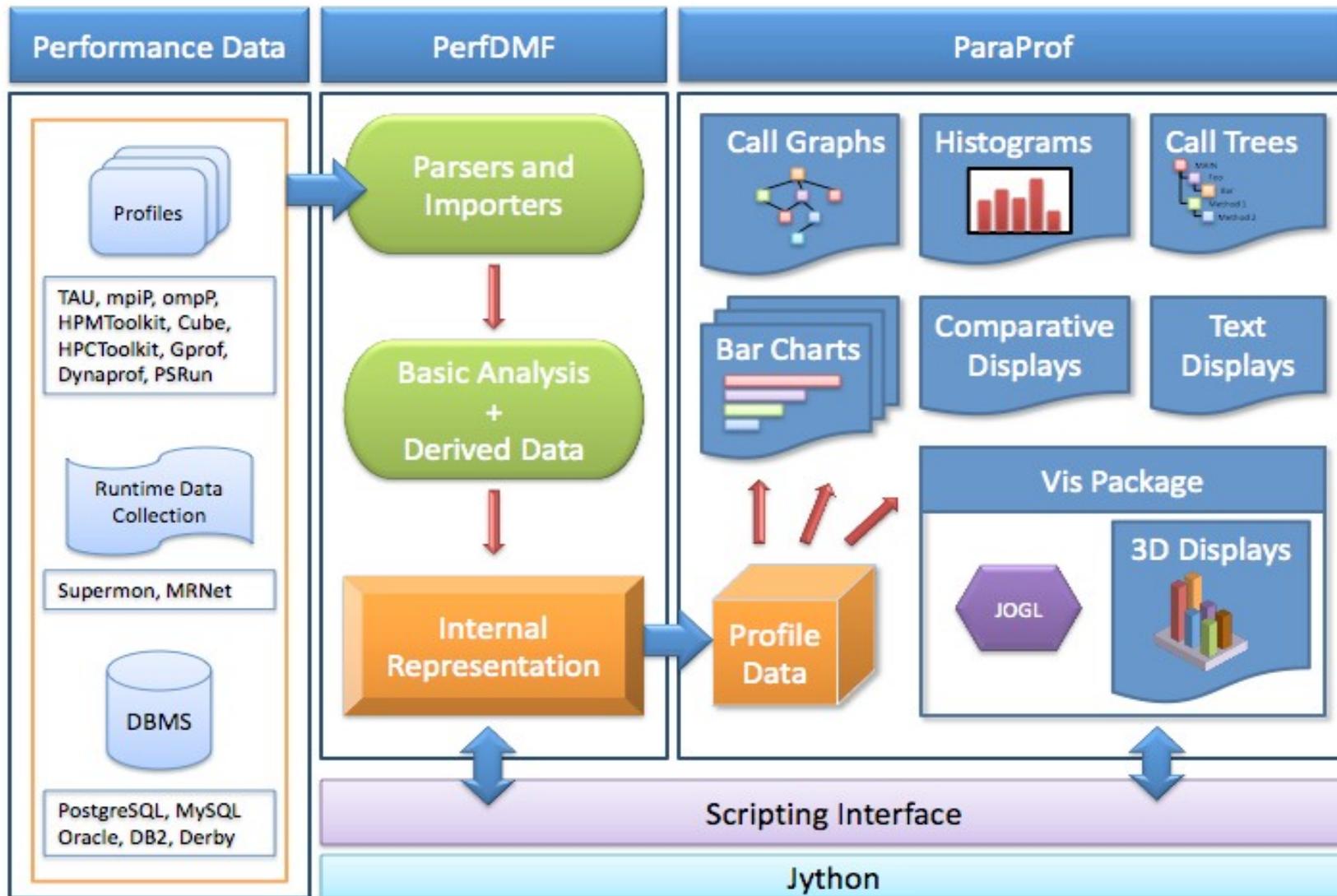
NODE 0;CONTEXT 0;THREAD 0:

| %Time | Exclusive<br>msec | Inclusive<br>total msec | #Call | #Subrs | Inclusive<br>usec/call | Name                  |
|-------|-------------------|-------------------------|-------|--------|------------------------|-----------------------|
| 100.0 | 1                 | 3:11.293                | 1     | 15     | 191293269              | applu                 |
| 99.6  | 3,667             | 3:10.463                | 3     | 37517  | 63487925               | bcast_inputs          |
| 67.1  | 491               | 2:08.326                | 37200 | 37200  | 3450                   | exchange_1            |
| 44.5  | 6,461             | 1:25.159                | 9300  | 18600  | 9157                   | butts                 |
| 41.0  | 1:18.436          | 1:18.436                | 18600 | 0      | 4217                   | MPI_Recv()            |
| 29.5  | 6,778             | 56,407                  | 9300  | 18600  | 6065                   | blts                  |
| 26.2  | 50,142            | 50,142                  | 19204 | 0      | 2611                   | MPI_Send()            |
| 16.2  | 24,451            | 31,031                  | 301   | 602    | 103096                 | rhs                   |
| 3.9   | 7,501             | 7,501                   | 9300  | 0      | 807                    | jaclcd                |
| 3.4   | 838               | 6,594                   | 604   | 1812   | 10918                  | exchange_3            |
| 3.4   | 6,590             | 6,590                   | 9300  | 0      | 709                    | jacu                  |
| 2.6   | 4,989             | 4,989                   | 608   | 0      | 8206                   | MPI_Wait()            |
| 0.2   | 0.44              | 400                     | 1     | 4      | 400081                 | init_comm             |
| 0.2   | 398               | 399                     | 1     | 39     | 399634                 | MPI_Init()            |
| 0.1   | 140               | 247                     | 1     | 47616  | 247086                 | setiv                 |
| 0.1   | 131               | 131                     | 57252 | 0      | 2                      | exact                 |
| 0.1   | 89                | 103                     | 1     | 2      | 103168                 | erhs                  |
| 0.1   | 0.966             | 96                      | 1     | 2      | 96458                  | read_input            |
| 0.0   | 95                | 95                      | 9     | 0      | 10603                  | MPI_Bcast()           |
| 0.0   | 26                | 44                      | 1     | 7937   | 44878                  | error                 |
| 0.0   | 24                | 24                      | 608   | 0      | 40                     | MPI_Irecv()           |
| 0.0   | 15                | 15                      | 1     | 5      | 15630                  | MPI_Finalize()        |
| 0.0   | 4                 | 12                      | 1     | 1700   | 12335                  | setbv                 |
| 0.0   | 7                 | 8                       | 3     | 3      | 2893                   | l2norm                |
| 0.0   | 3                 | 3                       | 8     | 0      | 491                    | MPI_Allreduce()       |
| 0.0   | 1                 | 3                       | 1     | 6      | 3874                   | pintgr                |
| 0.0   | 1                 | 1                       | 1     | 0      | 1007                   | MPI_Barrier()         |
| 0.0   | 0.116             | 0.837                   | 1     | 4      | 837                    | exchange_4            |
| 0.0   | 0.512             | 0.512                   | 1     | 0      | 512                    | MPI_Keyval_create()   |
| 0.0   | 0.121             | 0.353                   | 1     | 2      | 353                    | exchange_5            |
| 0.0   | 0.024             | 0.191                   | 1     | 2      | 191                    | exchange_6            |
| 0.0   | 0.103             | 0.103                   | 6     | 0      | 17                     | MPI_Type_contiguous() |

--:-- NPB\_LU.out (Fundamental)--L8--Top-----



# ParaProf Profile Analysis Framework



# Metadata for Each Experiment

TAU: ParaProf Manager

File Options Help

Applications

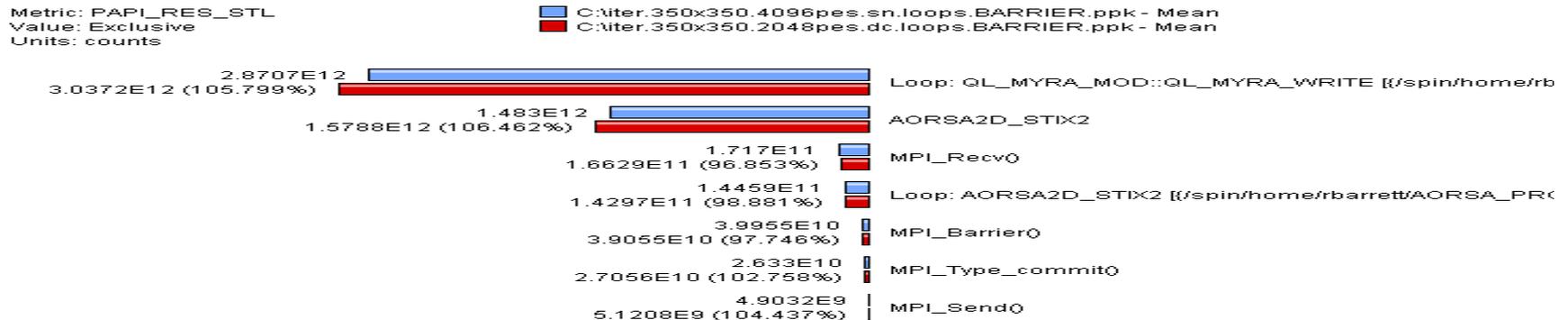
- Standard Applications
  - Default App
    - Default Exp
      - f90/pdt\_mpi/examples/tau2/amorris/home/
        - PAPI\_FP\_OPS
        - GET\_TIME\_OF\_DAY
- Default (jdbc:postgresql://spaceghost.cs.uoregon.edu:5432/)
- utionium (jdbc:postgresql://utionium.cs.uoregon.edu:5432/)
- spaceghost2 (jdbc:postgresql://spaceghost.cs.uoregon.edu:5432/)
- proton\_mysql (jdbc:mysql://192.168.1.1:3306/perfdm)
- spaceghost\_peri\_milc (jdbc:postgresql://spaceghost.cs.uoregon.edu:5432/)
- proton\_postgresql (jdbc:postgresql://192.168.1.1:5432/)
- utionium\_oracle (jdbc:oracle:thin:@//utionium.cs.uoregon.edu:1521/)
- perijtc (jdbc:postgresql://spaceghost.cs.uoregon.edu:5432/)

Multiple PerfDMF DBs

| 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.perfctrsm                  |
| 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                                    |

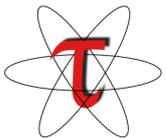
63

# Comparing Effects of Multi-Core Processors



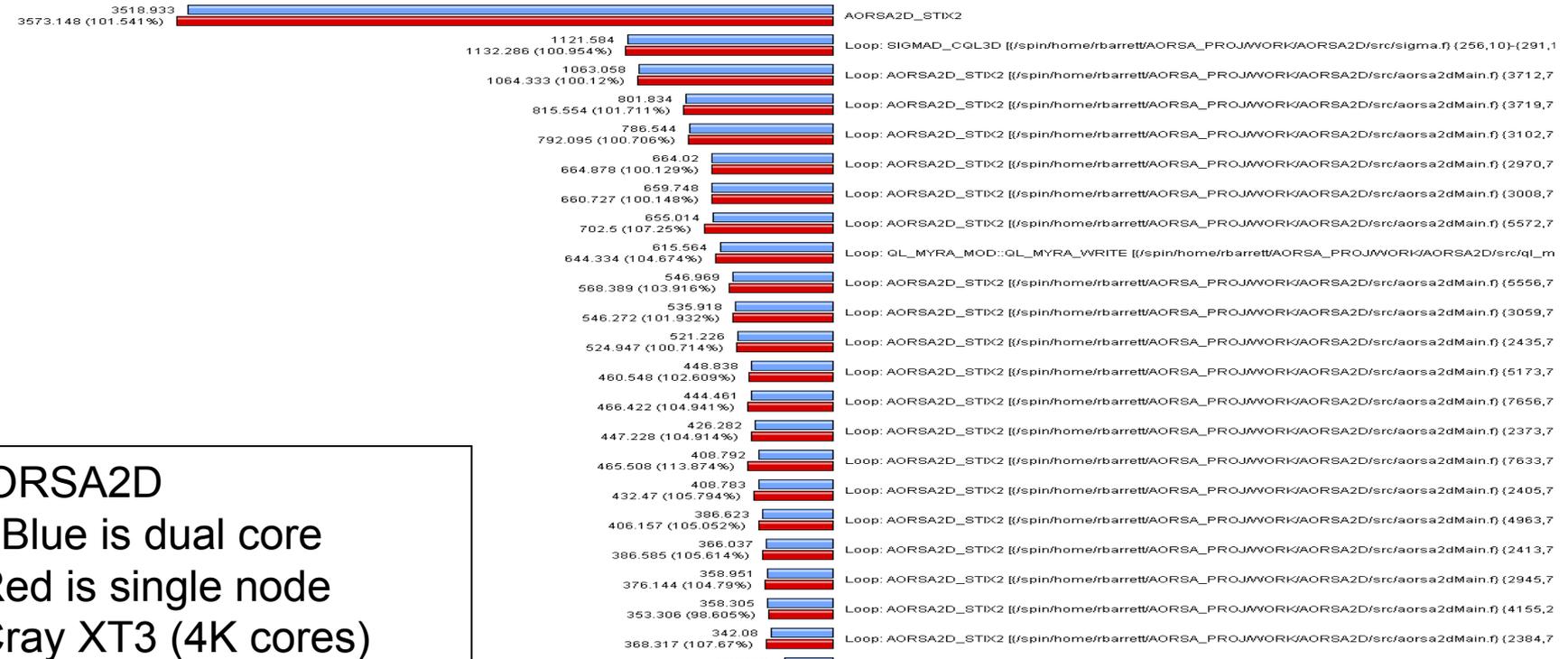
AORSA2D

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



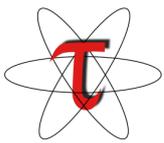
# Comparing FLOPS (AORSA2D, Cray XT3)

Metric: PAPI\_FP\_OPS / GET\_TIME\_OF\_DAY  
 Value: Exclusive  
 Units: Derived metric shown in microseconds format

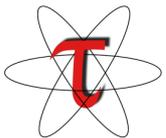
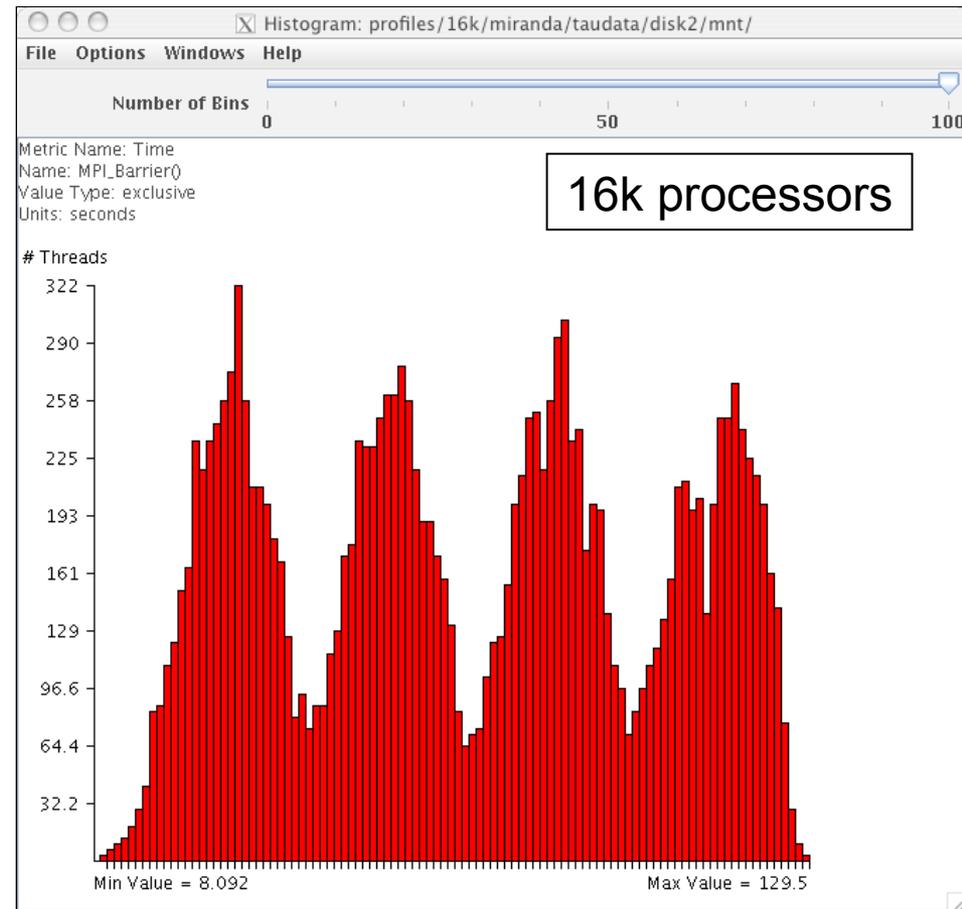
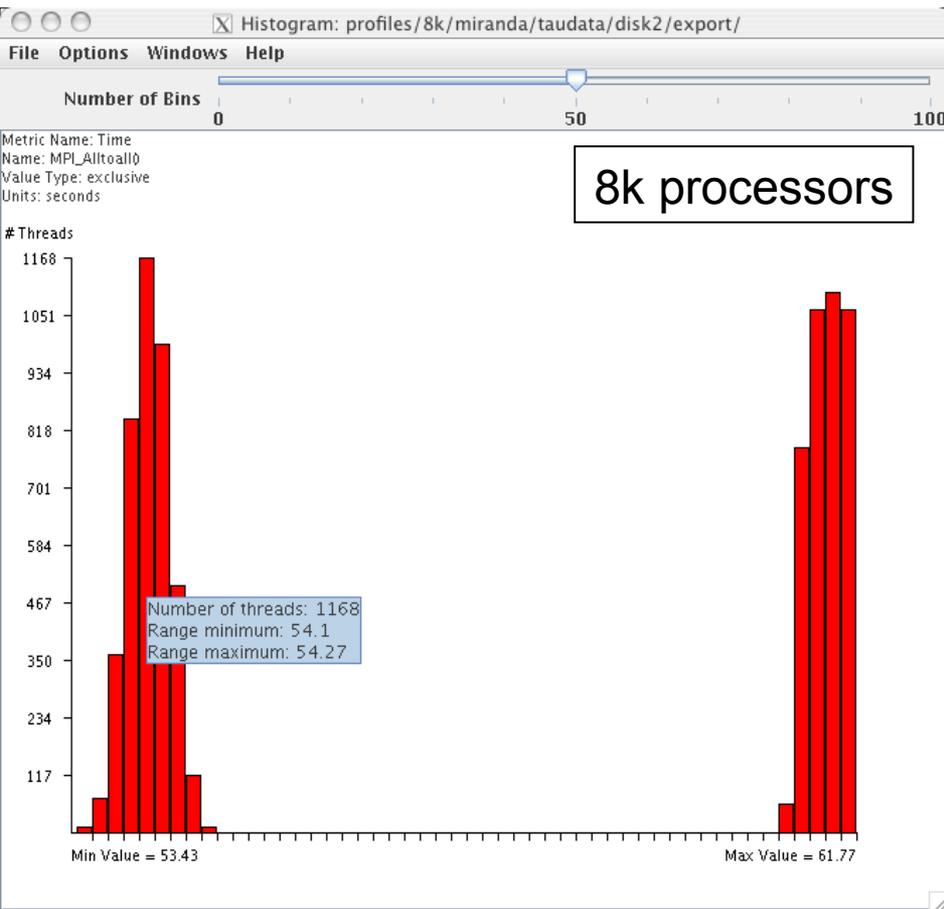


**AORSA2D**

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

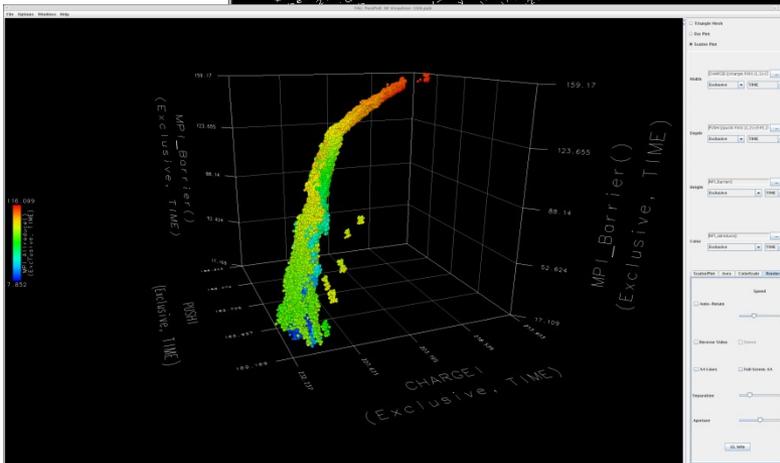
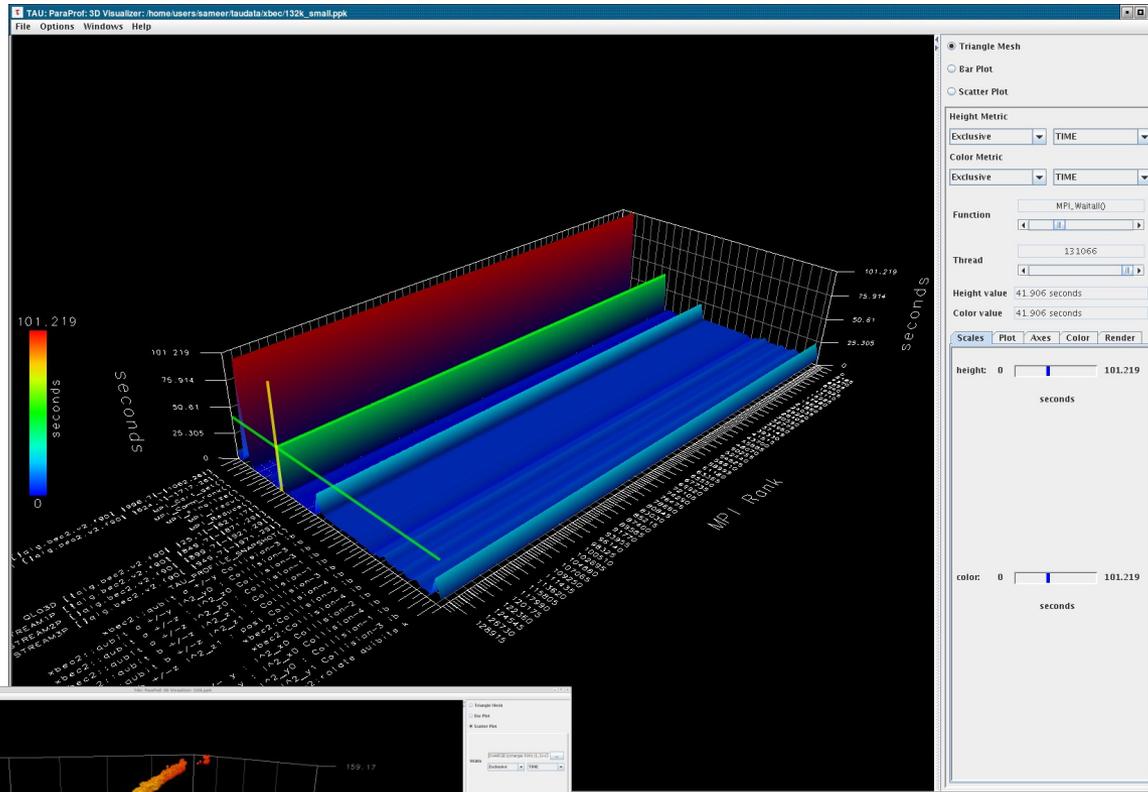


# ParaProf - Scalable Histogram



# ParaProf - 3D View (Full Profile)

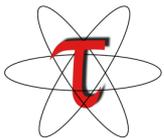
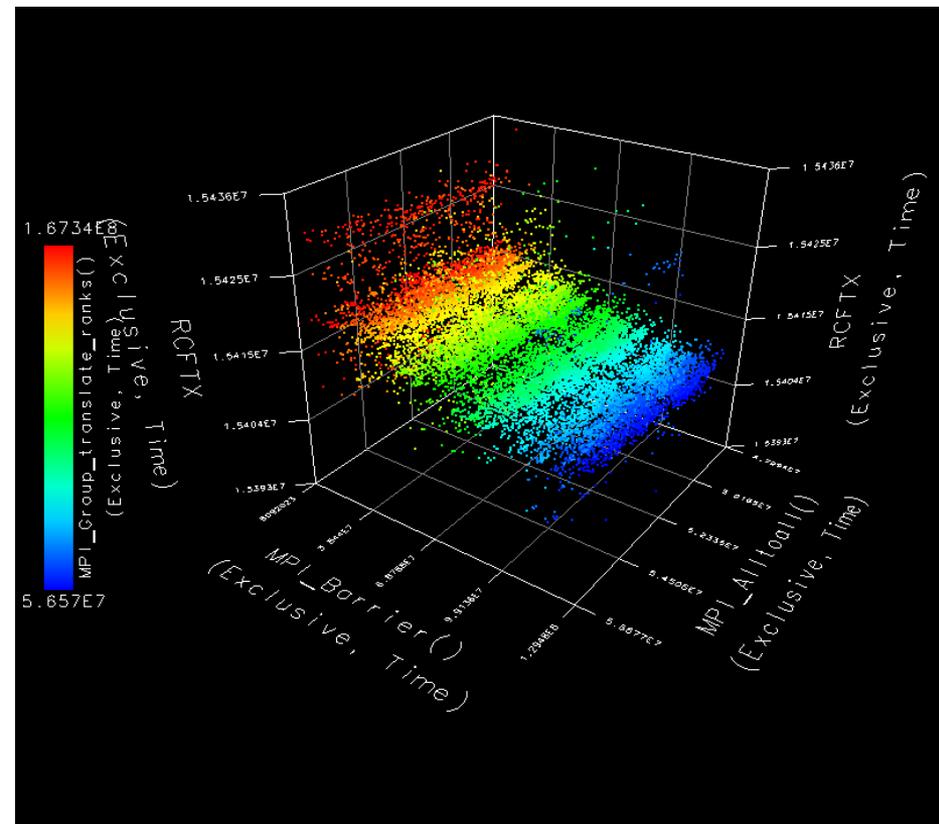
xbec



128k processors

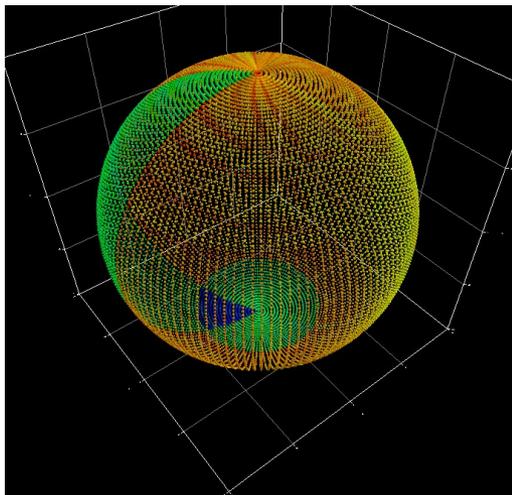
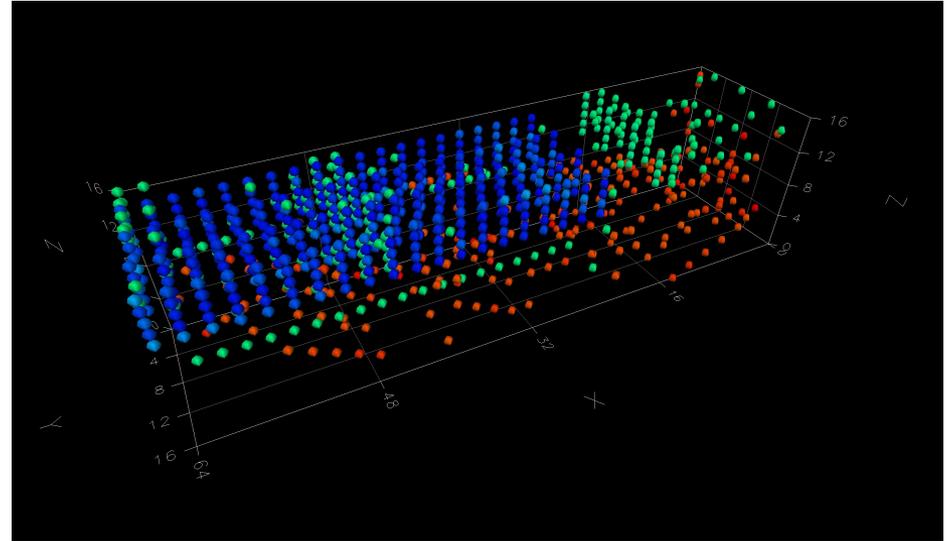
# ParaProf - 3D Scatterplot

- Each point is a “thread” of execution
- A total of four metrics shown in relation
- ParaProf’s visualization library
  - JOGL
- Miranda, 32k cores

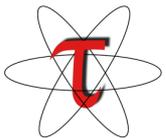


# ParaProf - 3D Topology View

- Thread/Node points from metadata or user defined
- Visibility selection by value and position



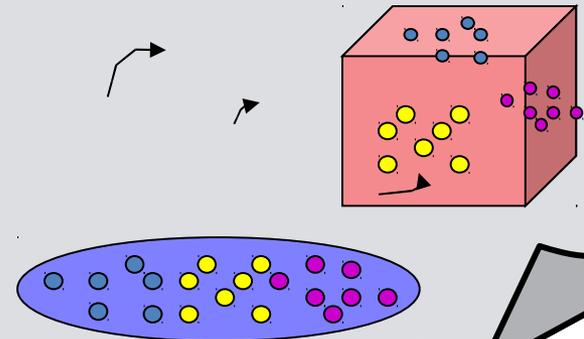
- Sweep3D 16k BGL Run
- Default Cartesian and User Defined Sphere Mappings



# Performance Mapping

- Example: Particles distributed on

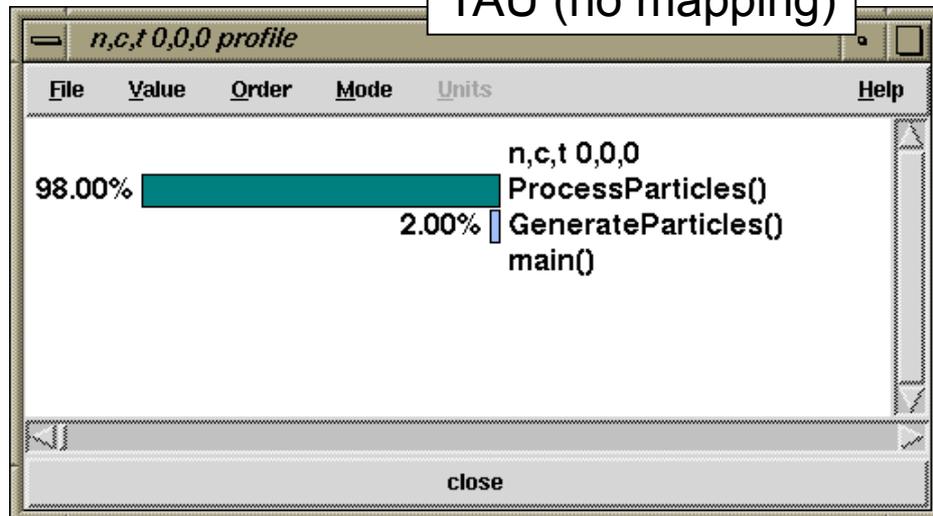
```
Particle* P[MAX]; /* Array of particles */
int GenerateParticles() {
 /* distribute particles over all faces of the cube */
 for (int face=0, last=0; face < 6; face++){
 /* particles on this face */
 int particles_on_this_face = num(face);
 for (int i=last; i < particles_on_this_face; i++) {
 /* particle properties are a function of face */
 P[i] = ... f(face);
 ...
 }
 last+= particles_on_this_face;
 }
}
```



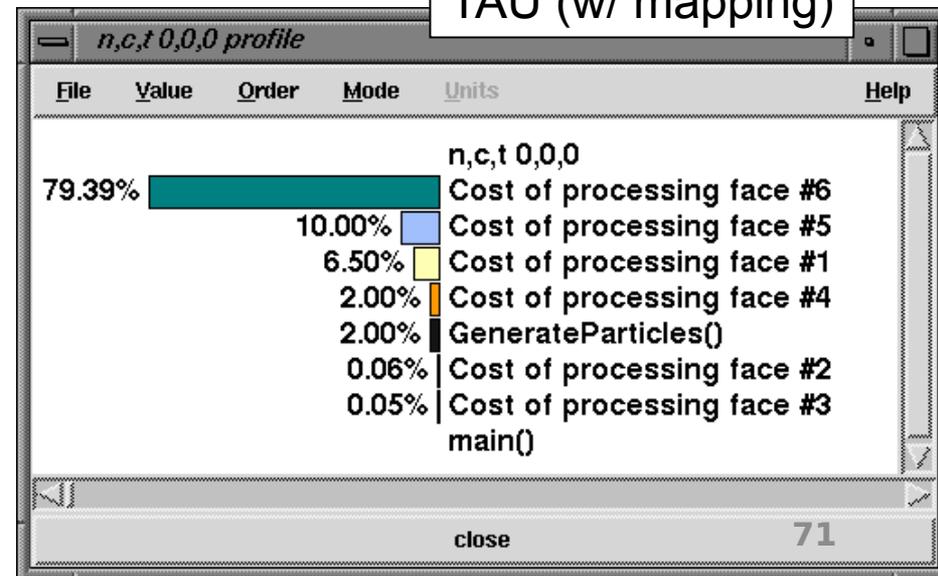
# No Mapping versus Mapping

- Typical performance tools report performance with respect to routines
- Does not provide support for mapping
- TAU's performance mapping can observe performance with respect to scientist's programming and problem abstractions

TAU (no mapping)



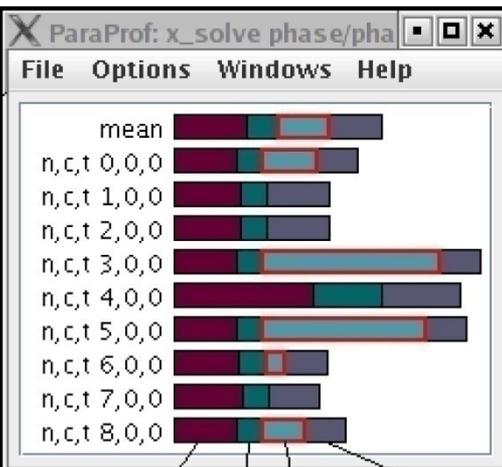
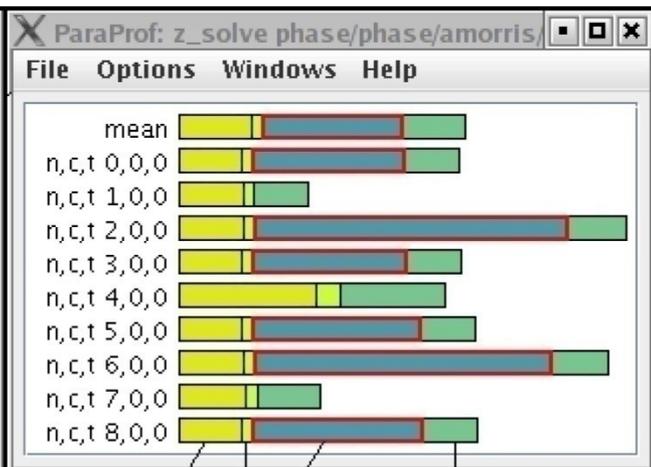
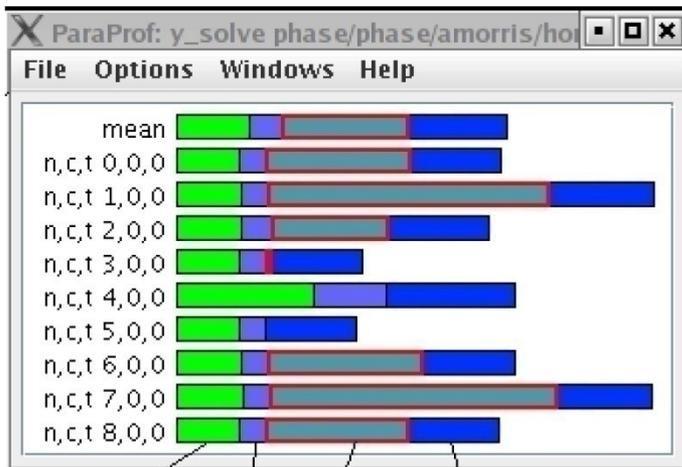
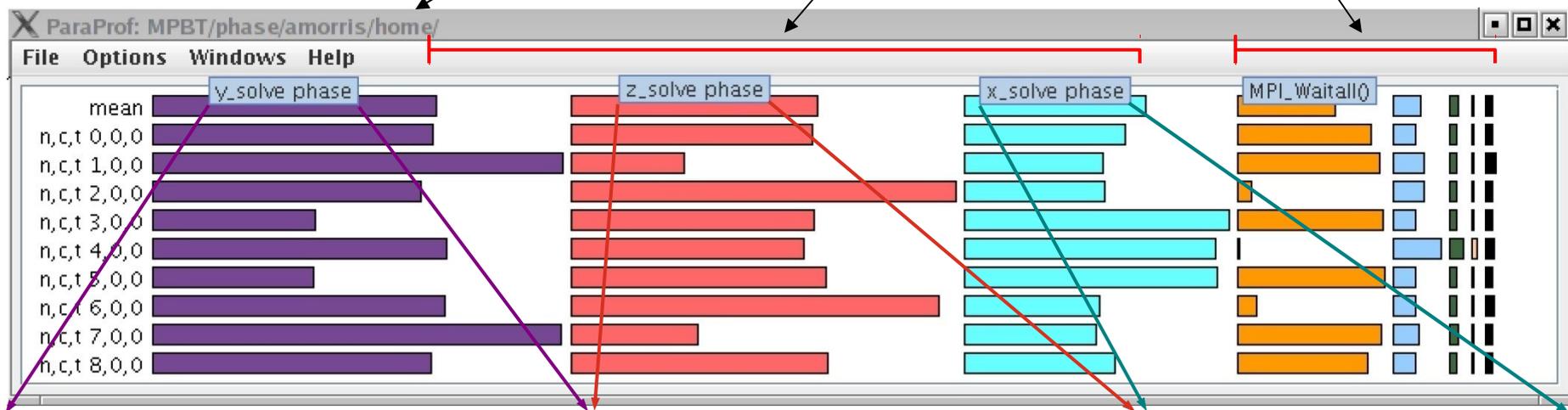
TAU (w/ mapping)





# NAS BT - Phase Profile

Main phase shows nested phases and immediate events



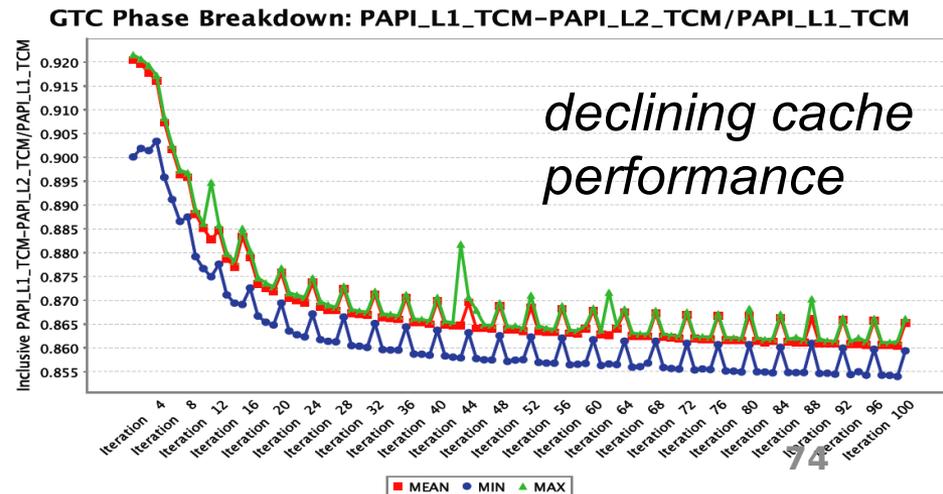
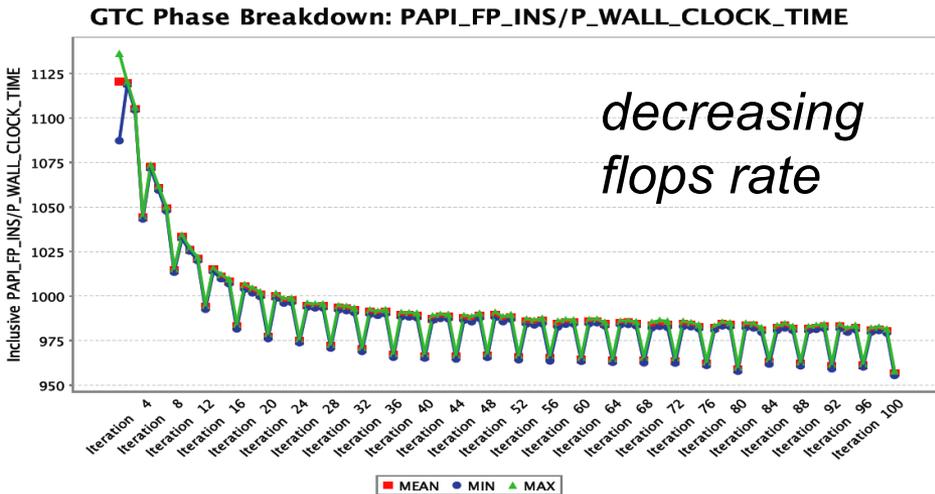
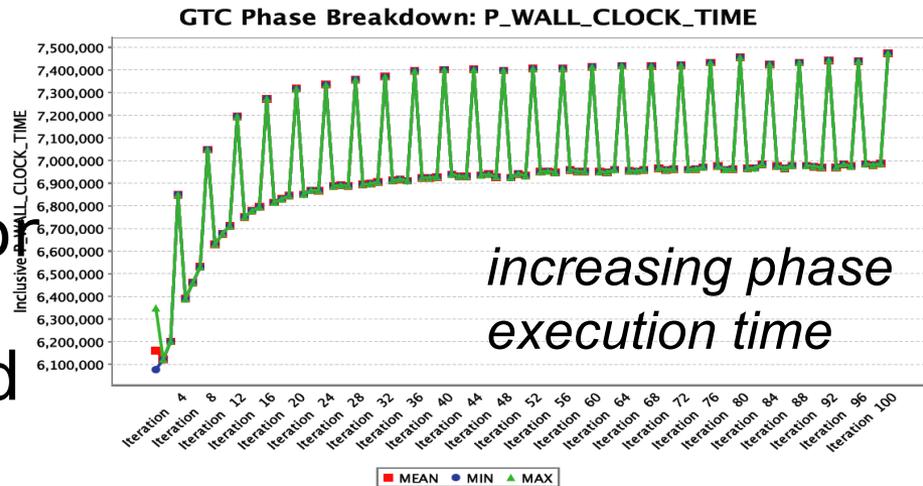
Y\_SOLVE\_CELL  
MPI\_Wait()  
LHSY  
Y\_BACKSUBSTITUTE

Z\_SOLVE\_CELL  
MPI\_Wait()  
LHSZ  
Z\_BACKSUBSTITUTE

X\_SOLVE\_CELL  
MPI\_Wait()  
LHSX  
X\_BACKSUBSTITUTE

# Phase Profiling of HW Counters

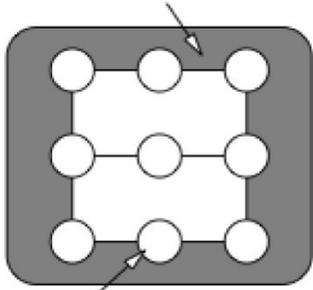
- GTC particle-in-cell simulation of fusion turbulence
- Phases assigned to iterations
- Poor temporal locality for one important data
- Automatically generated by PE2 python script



# Profile Snapshots in ParaProf

- Profile snapshots are parallel profiles recorded at runtime
- Shows performance profile dynamics (all types allowed)

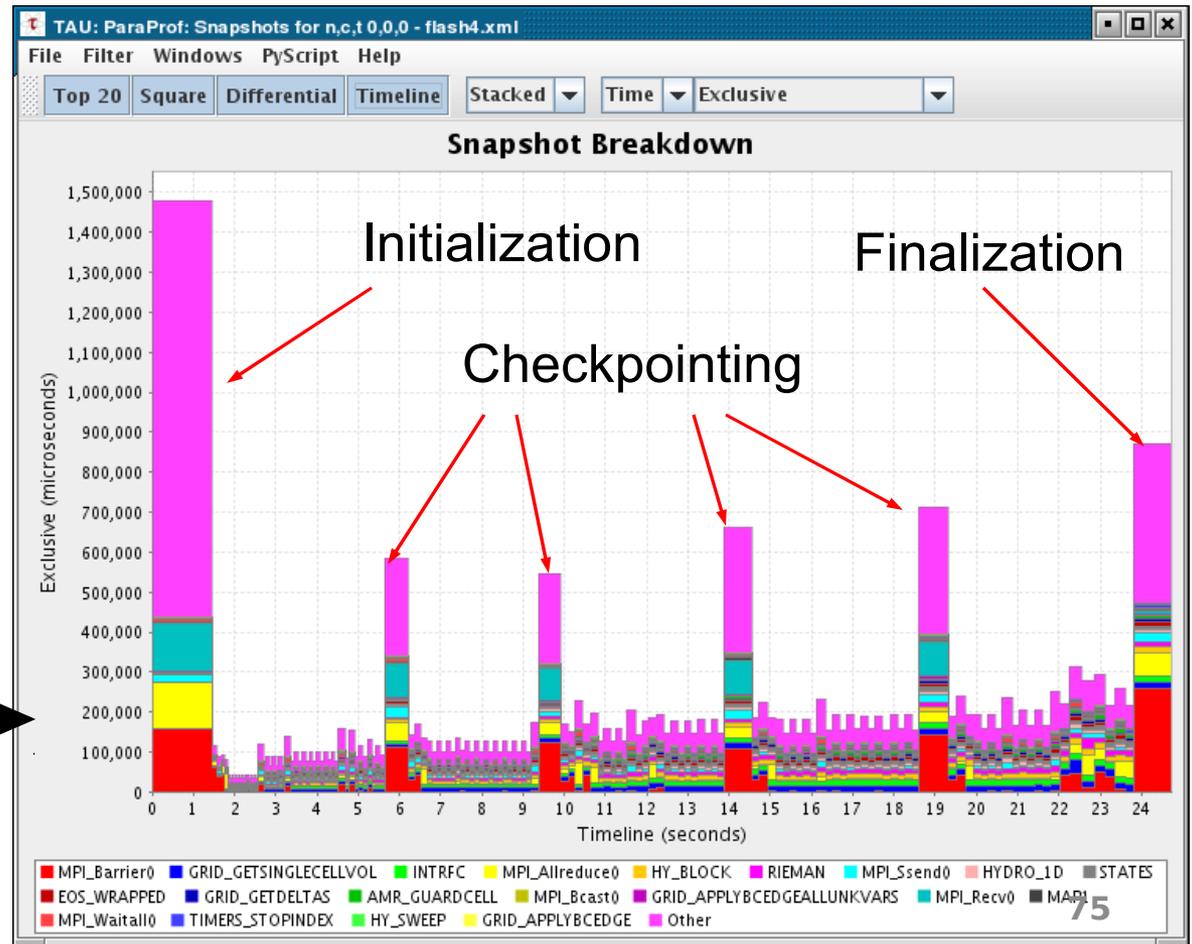
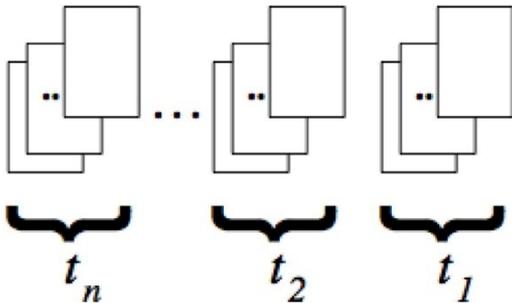
TAU measurement



application run  
on parallel system

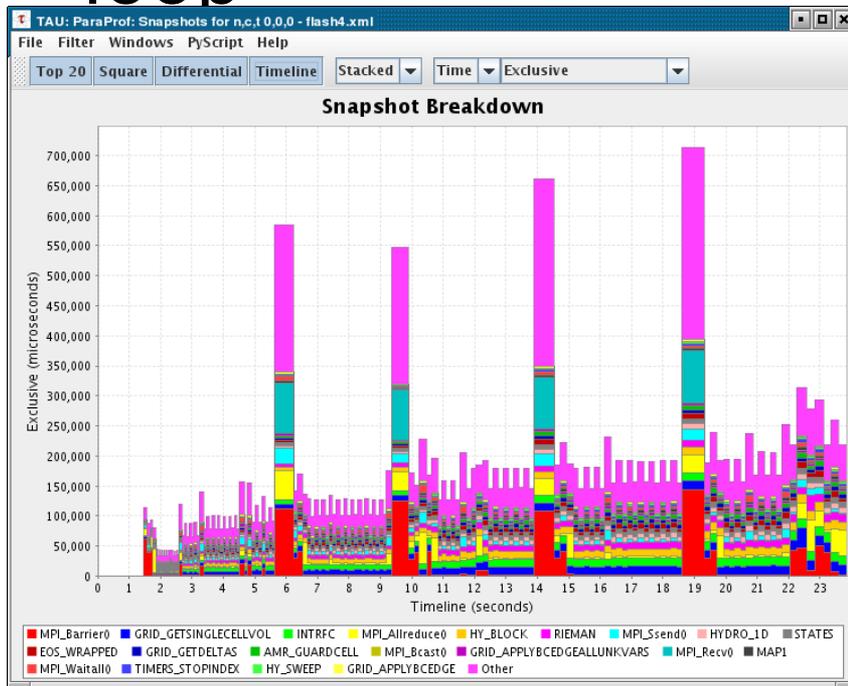


parallel profile snapshots

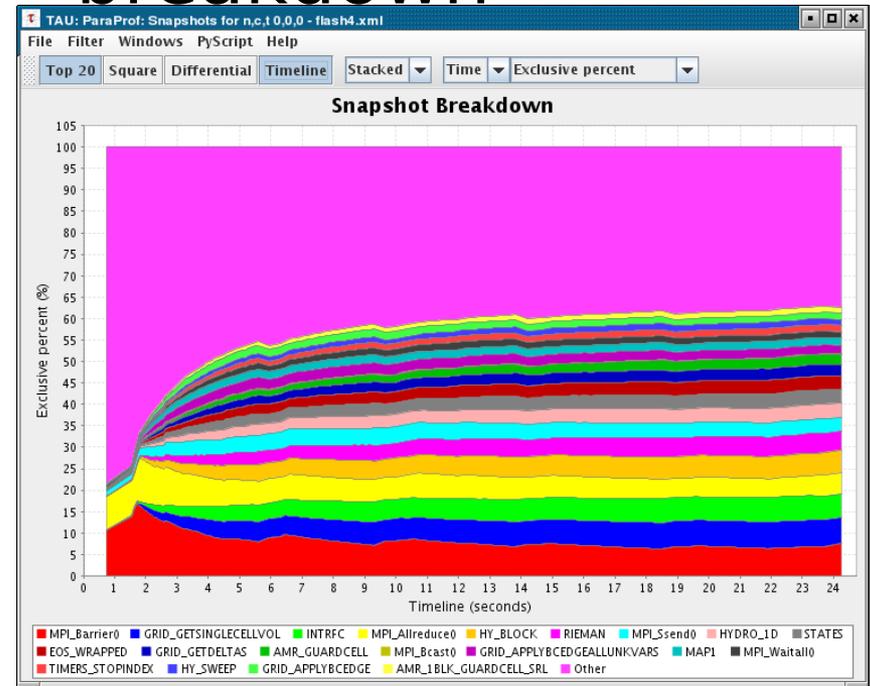


# Profile Snapshot Views

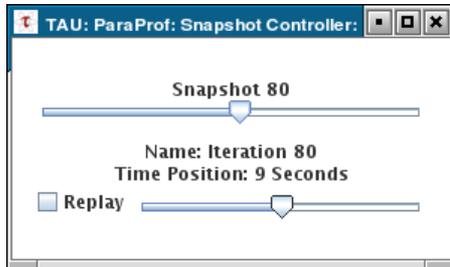
- Only show main loop



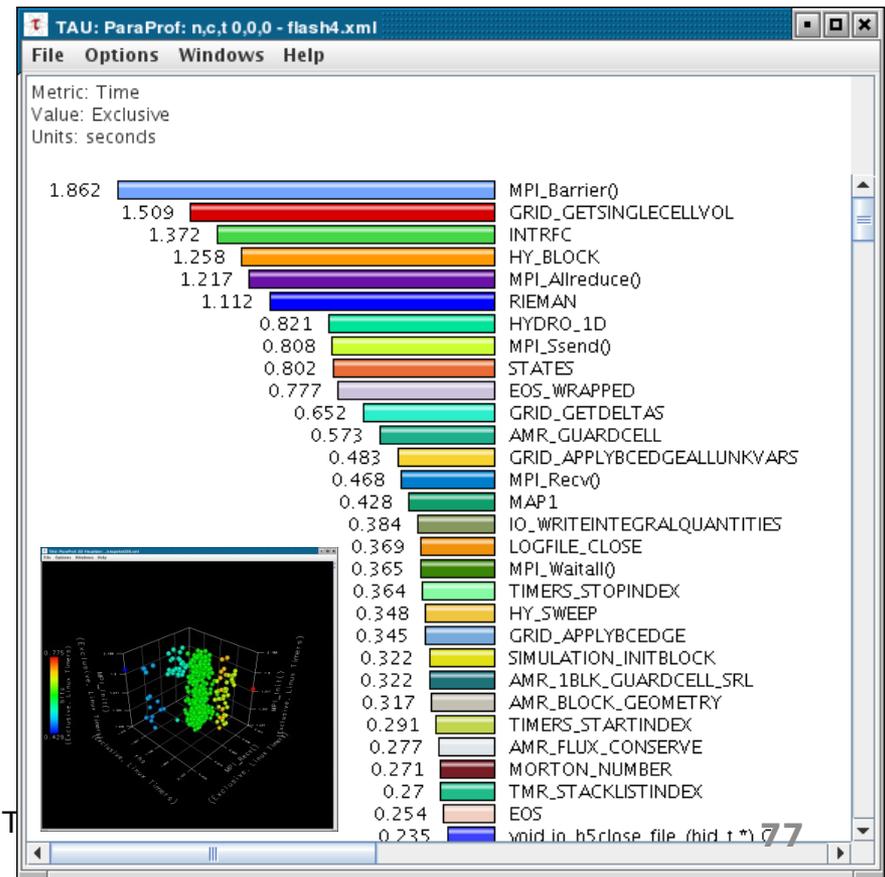
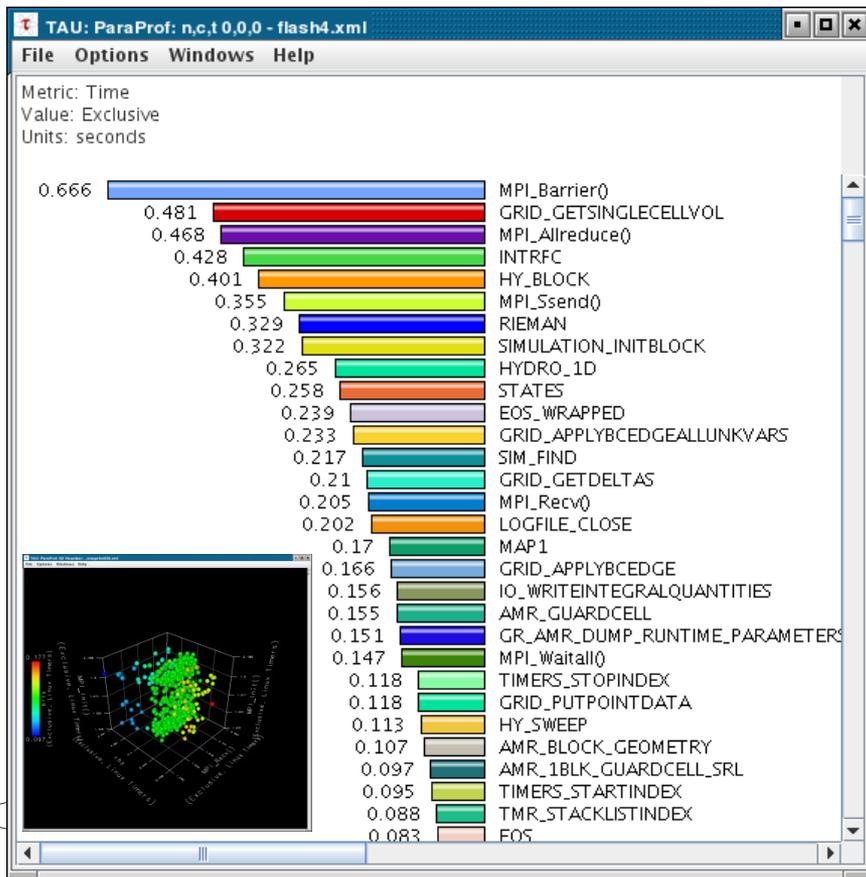
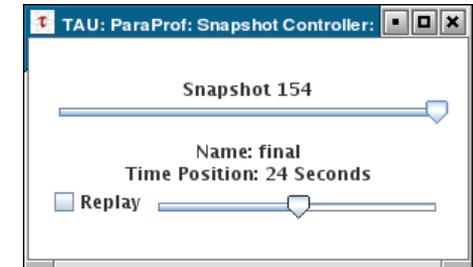
- Percentage breakdown



# Snapshot Replay in ParaProf

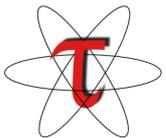


All windows dynamically update



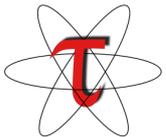
# Performance Data Mining / Analytics

- Conduct systematic and scalable analysis process
  - Multi-experiment performance analysis
  - Support automation, collaboration, and reuse
- Performance knowledge discovery framework
  - Data mining analysis applied to parallel performance data
  - comparative, clustering, correlation, dimension reduction, ...
  - Use the existing TAU infrastructure
- PerfExplorer v1 performance data mining framework
  - Multiple experiments and parametric studies
  - Integrate available statistics and data mining packages
  - Weka, R, Matlab / Octave
  - Apply data mining operations in interactive environment

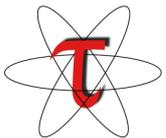
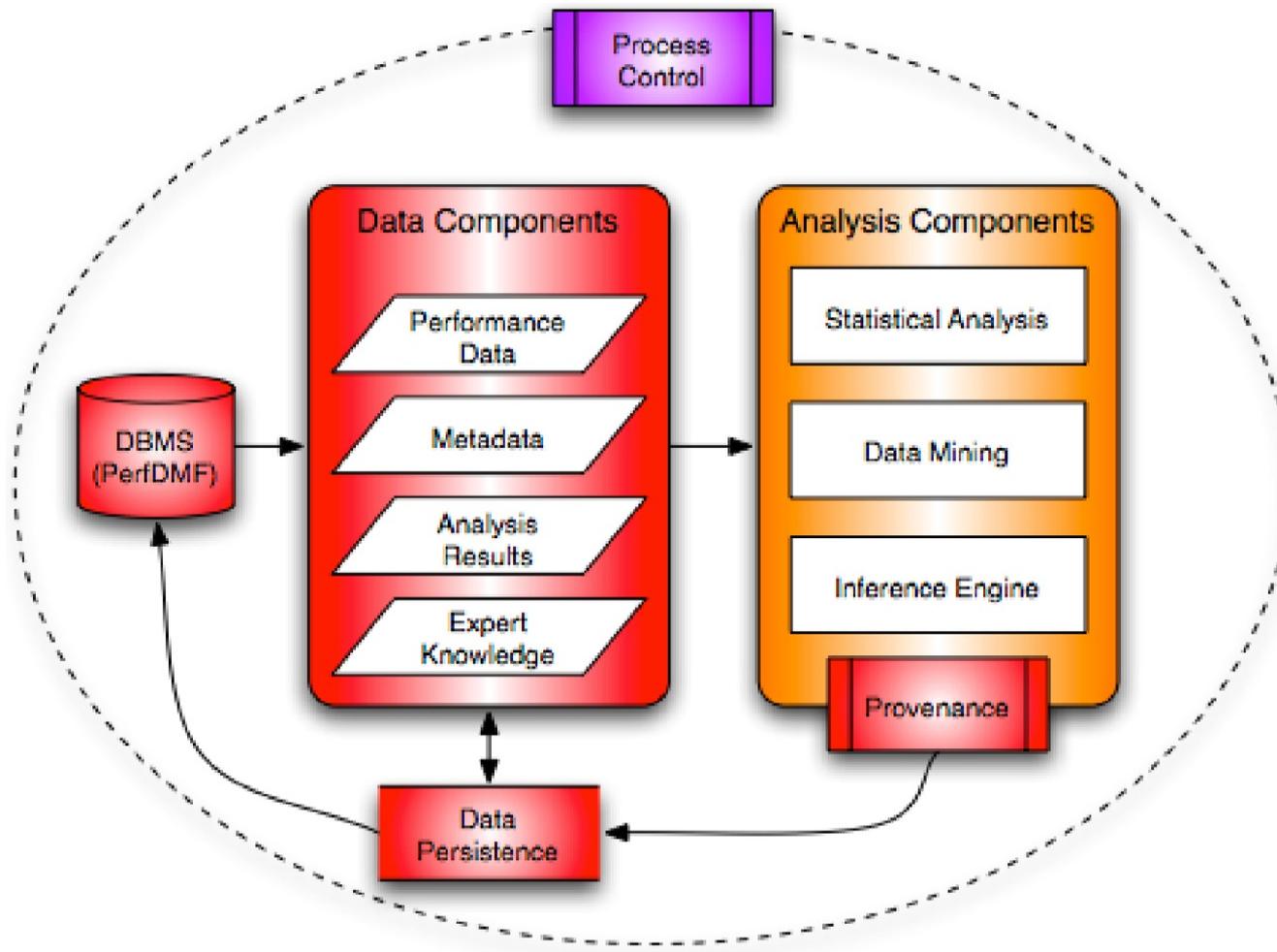


# PerfExplorer v2 - Requirements

- 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 analysis results
- Provenance
  - Provides historical record of analysis results

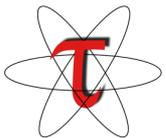
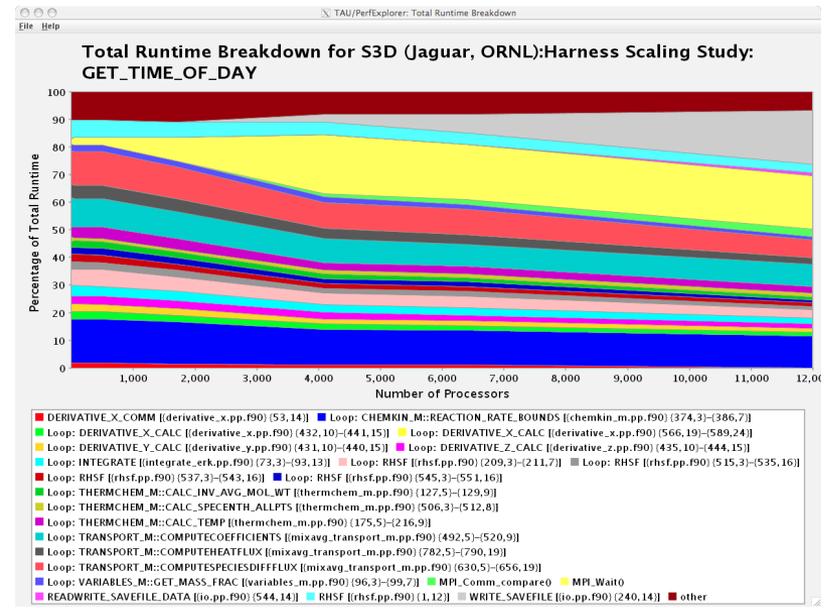
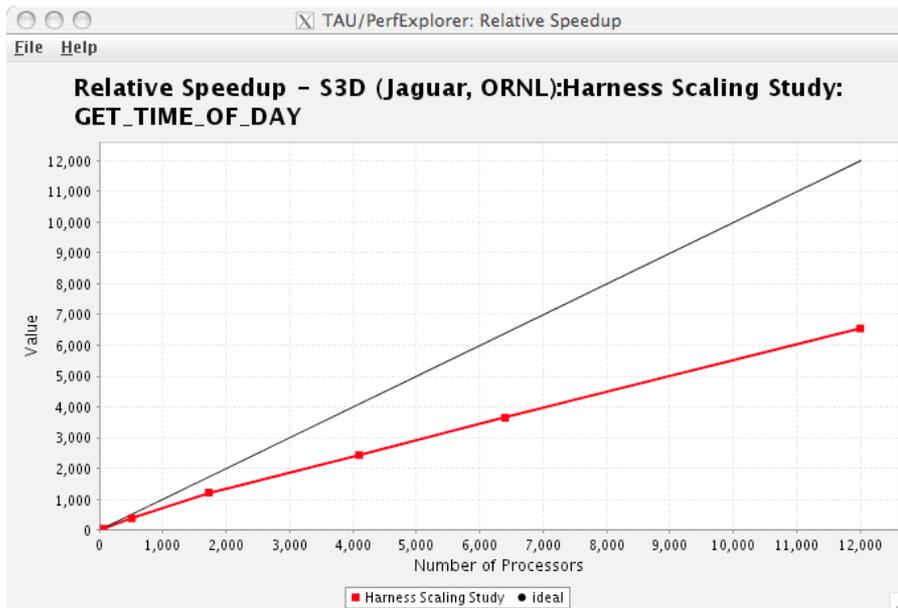


# PerfExplorer v2 Architecture

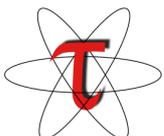
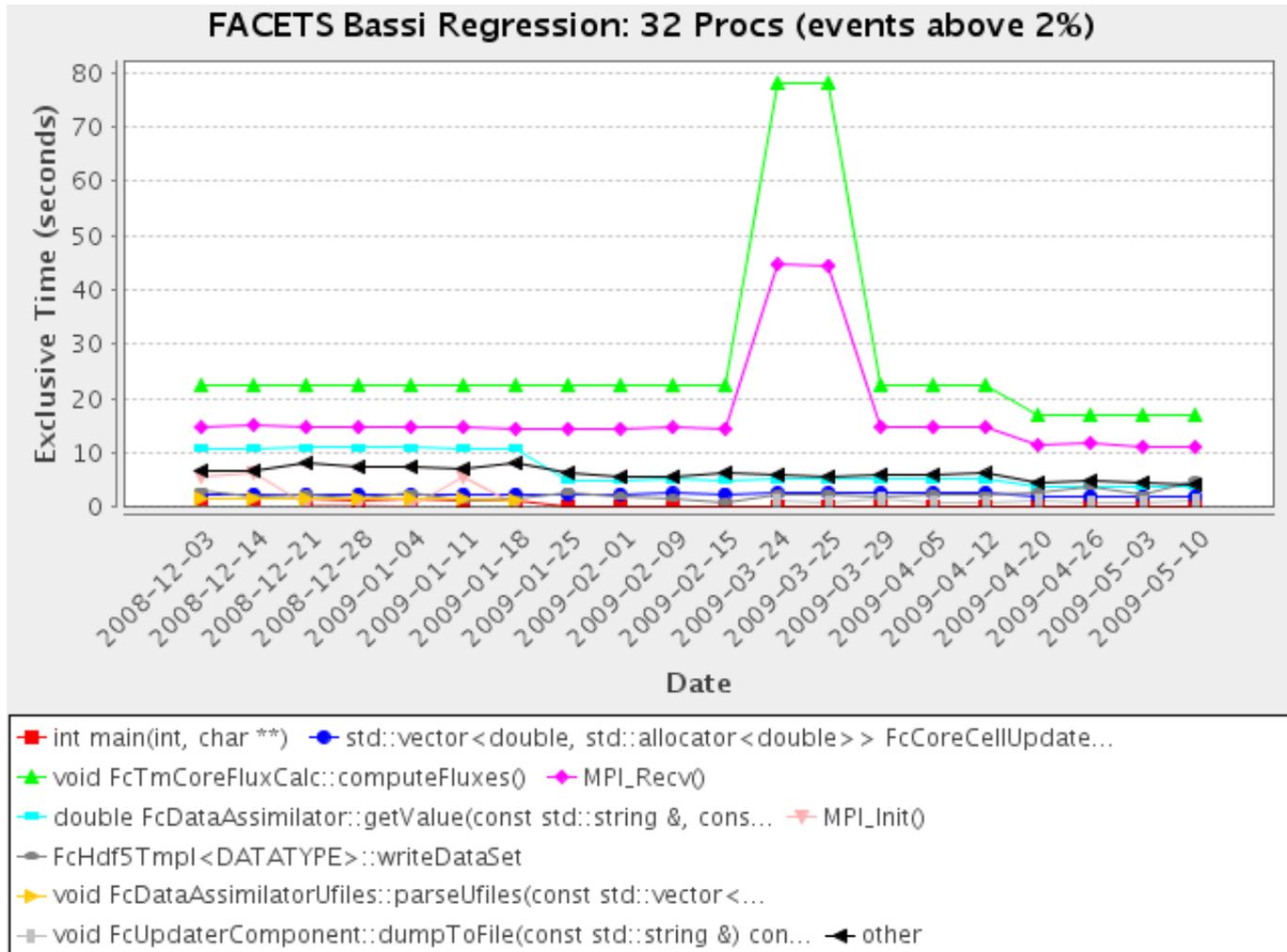


# Usage Scenarios: Evaluate Scalability

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



# Performance Regression Testing



# Evaluate Scalability using PerfExplorer Charts

```
% setenv TAU_MAKEFILE /opt/tau-2.19.1/x86_64
 /lib/Makefile.tau-mpi-pdt
% set path=(/opt/tau-2.19.1/x86_64/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, OR use **perfdmf\_loadtrial** on the commandline)

```
% perfexplorer
```

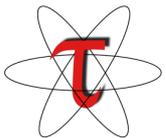
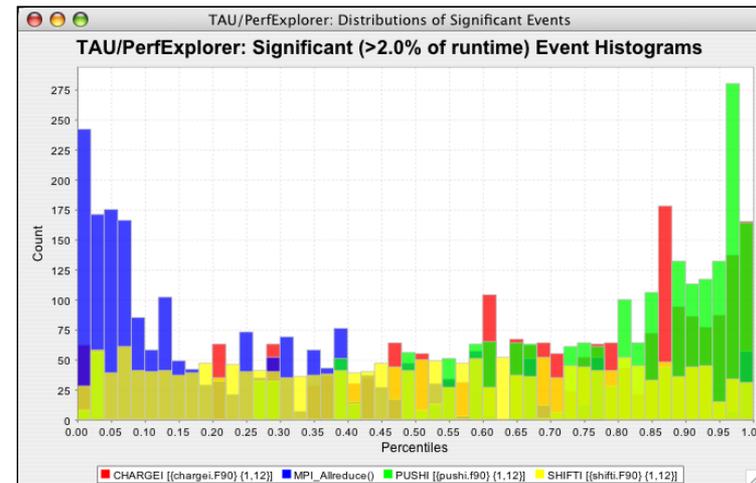
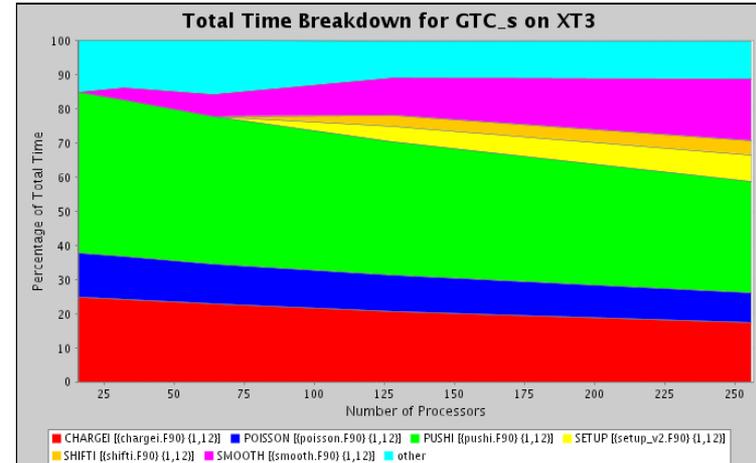
(Charts -> Speedup)



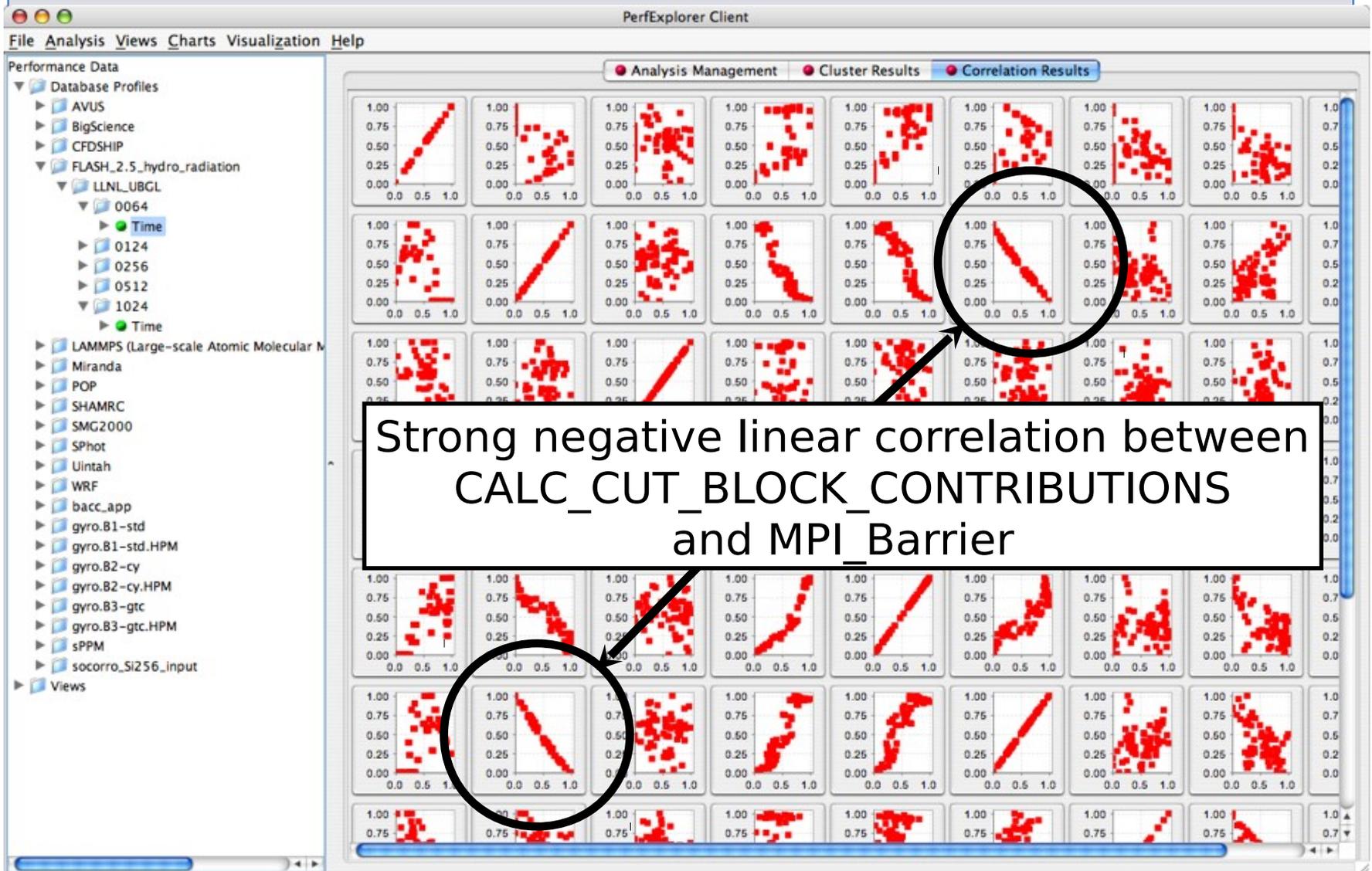


# PerfExplorer – Relative Comparisons

- 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

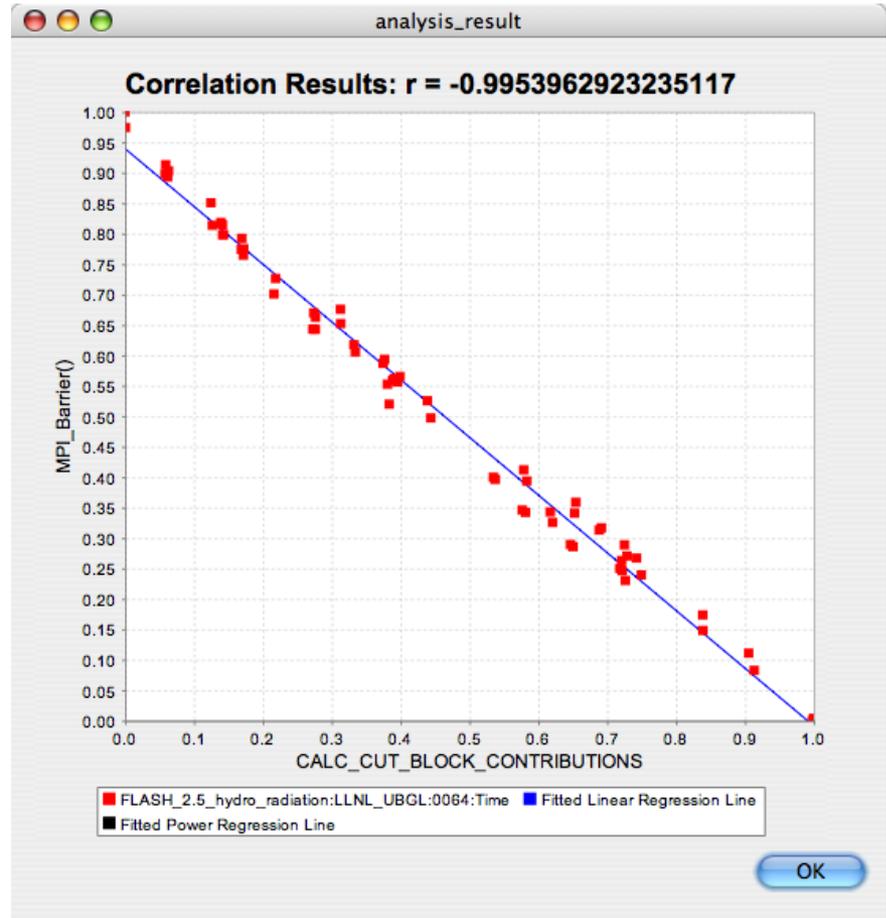


# PerfExplorer – Correlation Analysis



# PerfExplorer – Correlation Analysis

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

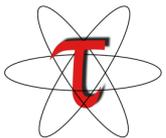
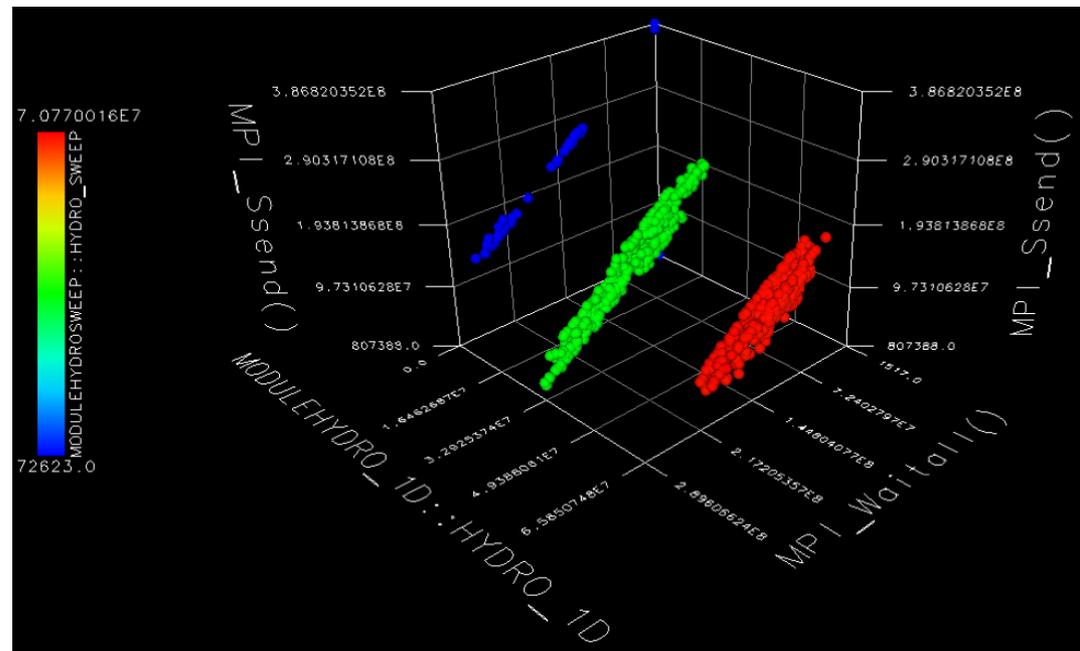


# PerfExplorer - Cluster Analysis

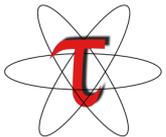
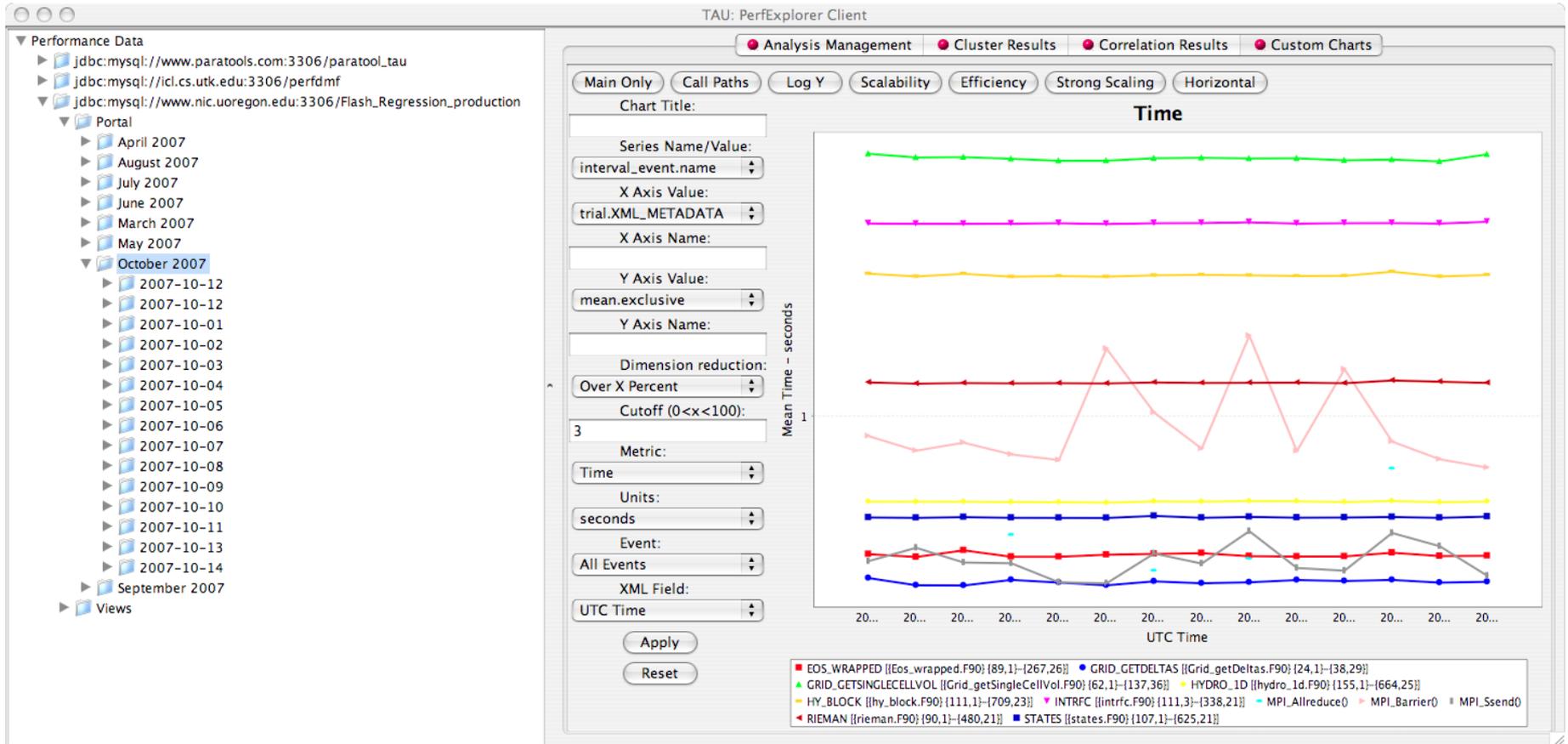


# PerfExplorer – Cluster Analysis

- Four significant events automatically selected
- Clusters and correlations are visible

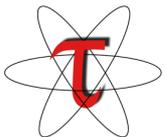


# PerfExplorer – Performance Regression



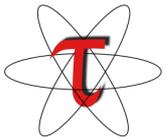
# Other Projects in TAU

- TAU Portal
  - Support collaborative performance study
- Kernel-level system measurements (KTAU)
  - Application to OS noise analysis and I/O system analysis
- TAU performance monitoring
  - TAUoverSupermon and TAUoverMRNet
- PerfExplorer integration and expert-based analysis
  - OpenUH compiler optimizations
  - Computational quality of service in CCA
- Eclipse CDT and PTP integration
- Performance tools integration (NSF POINT project)



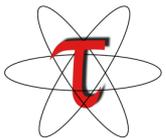
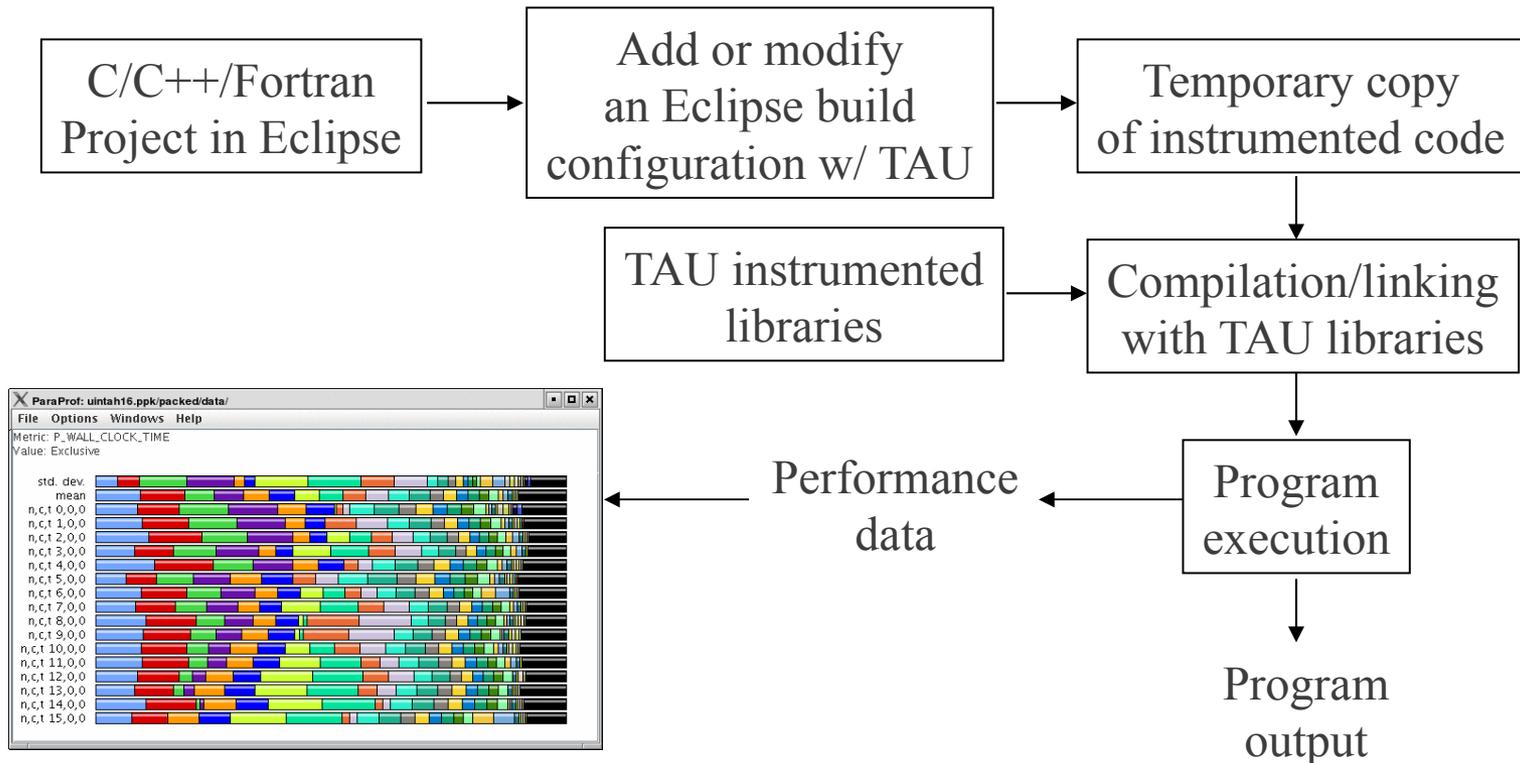
# TAU Integration with IDEs

- 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

The screenshot shows the Eclipse IDE interface for a Fortran project named 'matmult.f90'. The main editor displays the following code:

```
! 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

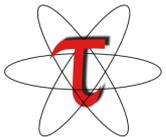
! multiply the row with the column
```

The left-hand side shows the Project Navigator with the 'matmult.f90' file selected. The right-hand side shows the Outline view with the following structure:

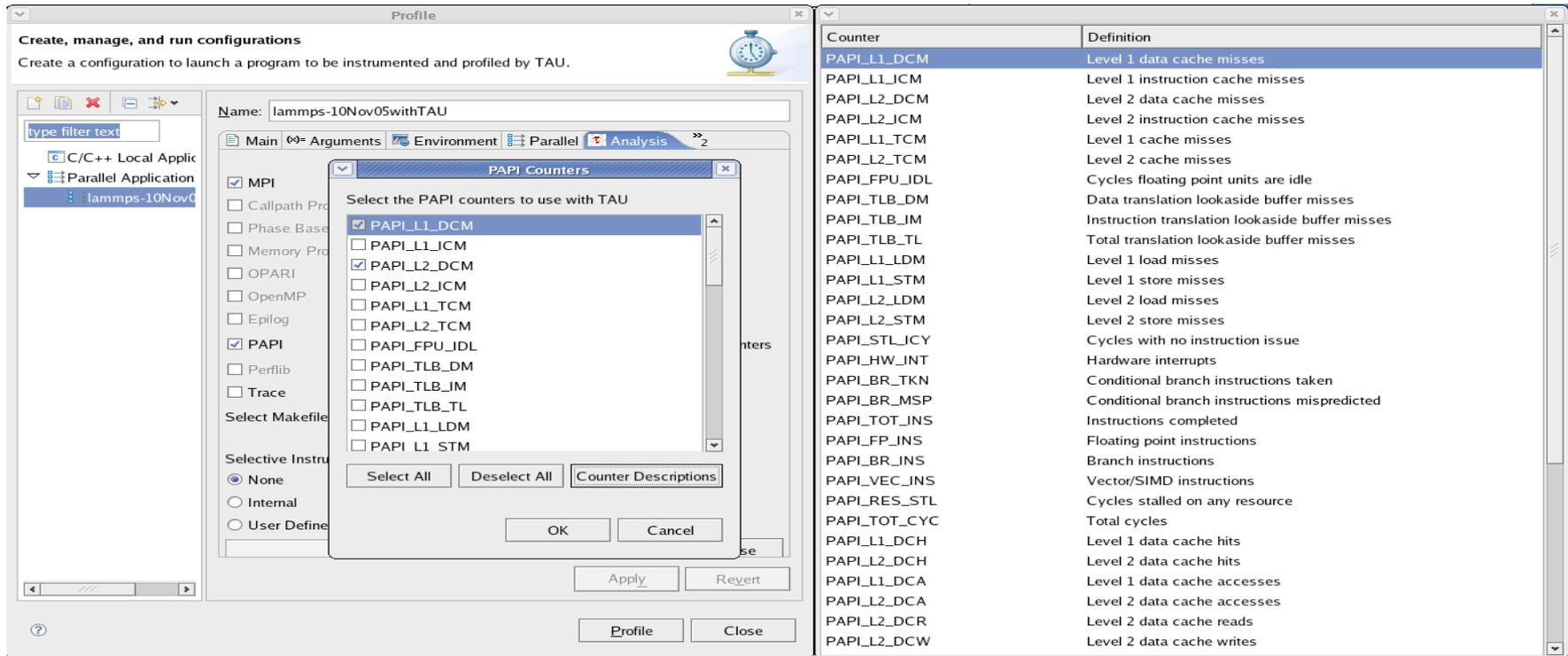
- initialize
- multiply\_matrices
- main

The bottom of the IDE features the Performance Data Manager (PerfDMF) view, which is currently empty. An arrow points from the text 'PerfDMF' to this view.

PerfDMF



# Choosing PAPI Counters with TAU in Eclipse

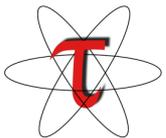


The screenshot shows the Eclipse IDE interface. The main window is titled "Profile" and contains a "PAPI Counters" dialog box. The dialog box has a title bar "PAPI Counters" and a subtitle "Select the PAPI counters to use with TAU". It contains a list of PAPI counter names with checkboxes next to them. The following table shows the list of counters and their definitions as seen in the screenshot:

| 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                       |

The dialog box also has buttons for "Select All", "Deselect All", "Counter Descriptions", "OK", and "Cancel". The main window has a "Name" field containing "lammps-10Nov05withTAU" and a "PAPI" checkbox that is checked. The "Analysis" tab is selected in the main window.

% /usr/local/packages/eclipse/eclipse

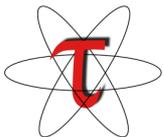


# Support Acknowledgements

- Department of Energy (DOE)
  - Office of Science
    - MICS, Argonne National Lab
  - ASC/NNSA
    - University of Utah ASC/NNSA Level 1
    - ASC/NNSA, Lawrence Livermore National Lab
- Department of Defense (DoD)
  - HPC Modernization Office (HPCMO)
- NSF Software Development for Cyberinfrastructure (SDCI)
- Research Centre Juelich
- ANL, NASA Ames, LANL, SNL
- TU Dresden
- ParaTools, Inc.
- University of Oregon Performance Research Lab



UNIVERSITY  
OF OREGON



# For more information

- TAU Website:  
<http://tau.uoregon.edu>
  - Software
  - Release notes
  - Documentation
- To use Paraprof on your local system without installing TAU:  
<http://tau.uoregon.edu/paraprof>
- Paratools: <http://www.paratools.com>

