Parallel Performance Evaluation using the TAU Performance System Project

Workshop on Performance Tools for Petascale Computing

9:30 – 10:30am, Tuesday, July 17, 2007, Snowbird, UT

Sameer S. Shende

sameer@cs.uoregon.edu

http://www.cs.uoregon.edu/research/tau

Performance Research Laboratory

University of Oregon

UNIVERSITY
OF OREGON

Acknowledgements

- Dr. Allen D. Malony, Professor
- Alan Morris, Senior software engineer
- Wyatt Spear, Software engineer
- Scott Biersdorff, Software engineer
- Kevin Huck, Ph.D. student
- Aroon Nataraj, Ph.D. student
- Brad Davidson, Systems administrator

Outline

- Overview of features
- Instrumentation
- Measurement
- Analysis tools
  - Parallel profile analysis (ParaProf)
  - Performance data management (PerfDMF)
  - Performance data mining (PerfExplorer)
- Application examples
- Kernel monitoring and KTAU

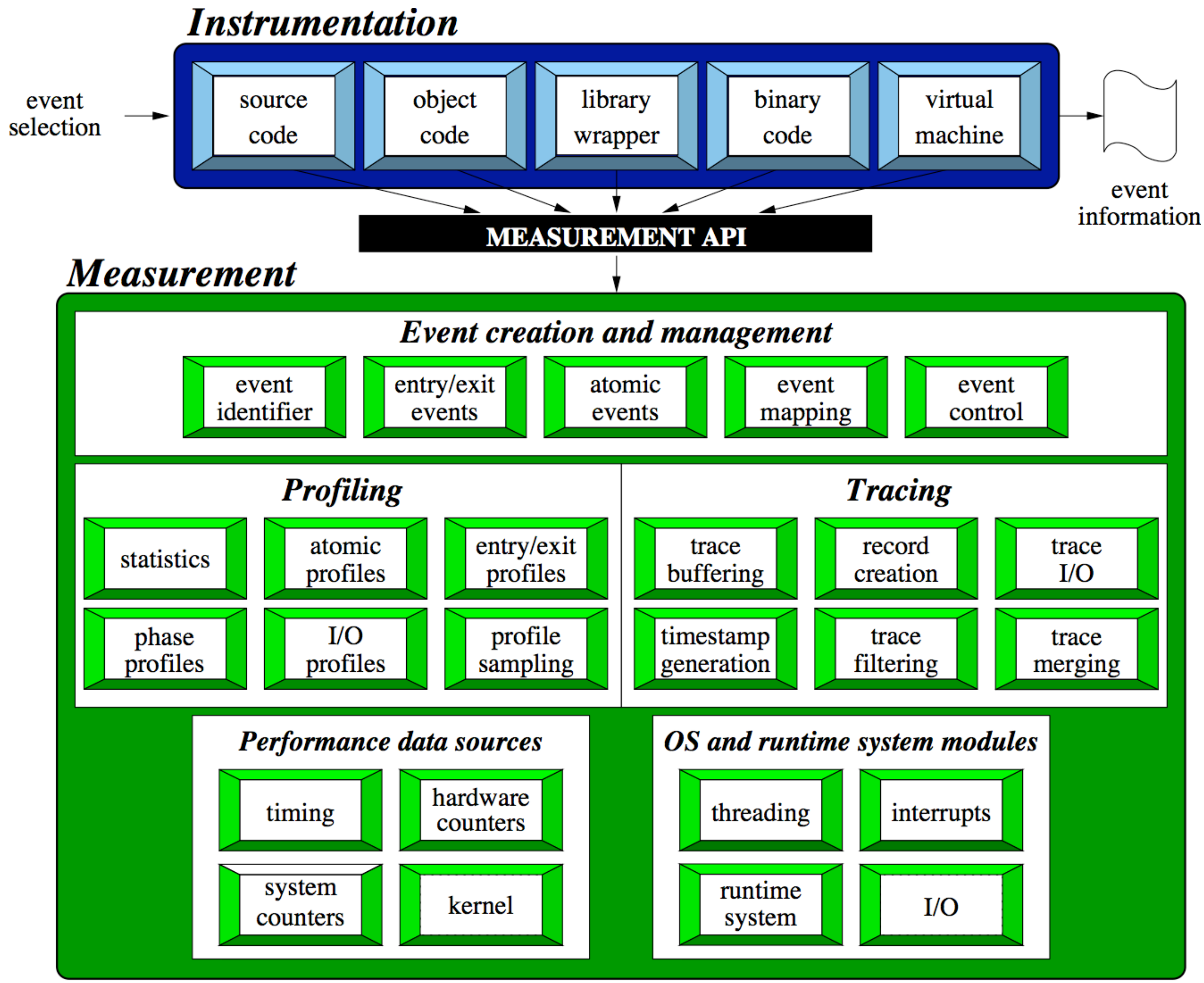
TAU Performance System

- ***T***uning and ***A***nalysis ***U***tilities (15+ year project effort)
- **Performance system framework for HPC systems**
  - Integrated, scalable, flexible, and parallel
- Targets a general complex system computation model
  - *Entities*: nodes / contexts / threads
  - *Multi-level*: system / software / parallelism
  - Measurement and analysis abstraction
- **Integrated toolkit for performance problem solving**
  - Instrumentation, measurement, analysis, and visualization
  - Portable performance profiling and tracing facility
  - Performance data management and data mining
- ***Partners***: LLNL, ANL, LANL, Research Center Jülich

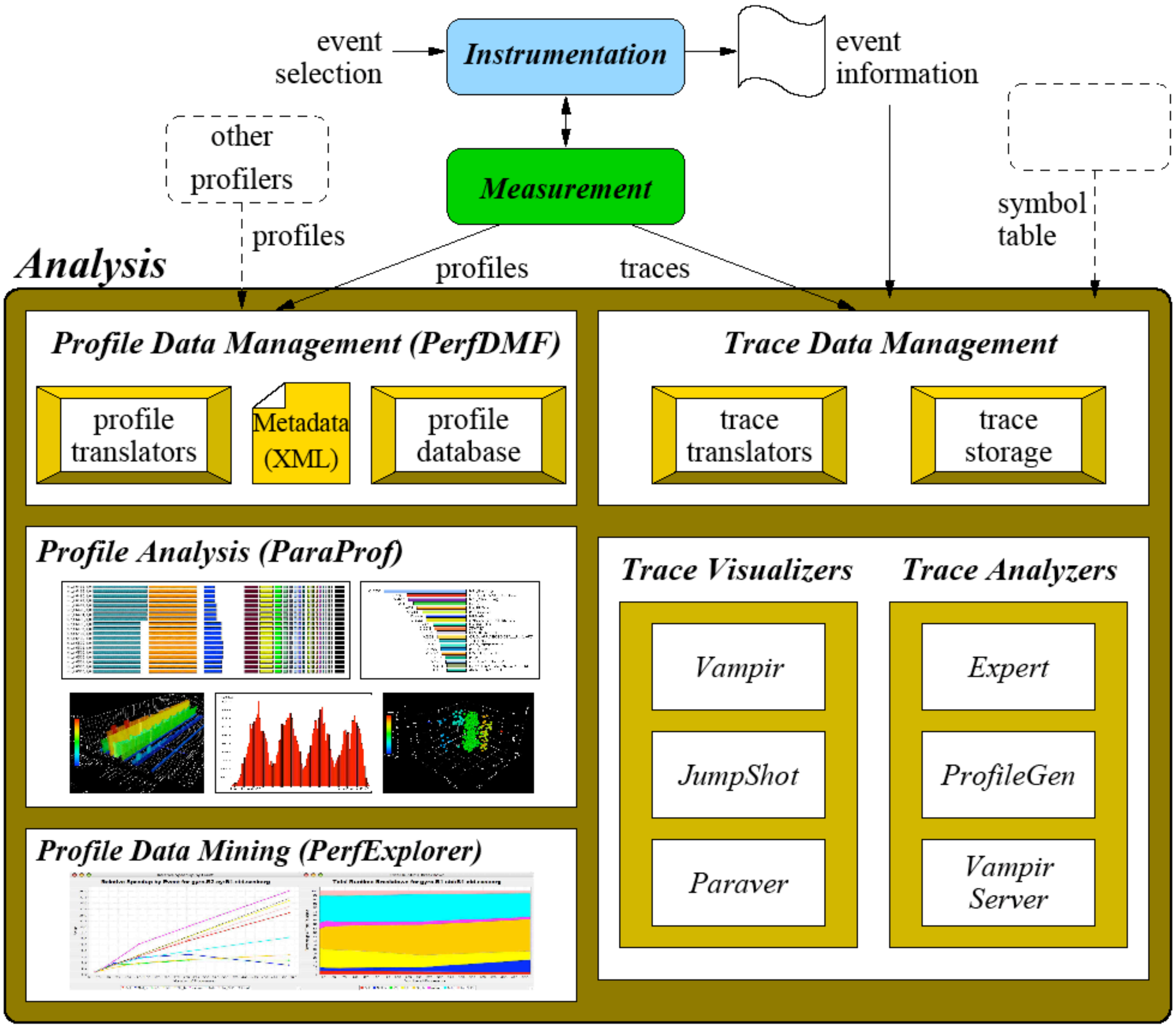
TAU Parallel Performance System Goals

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

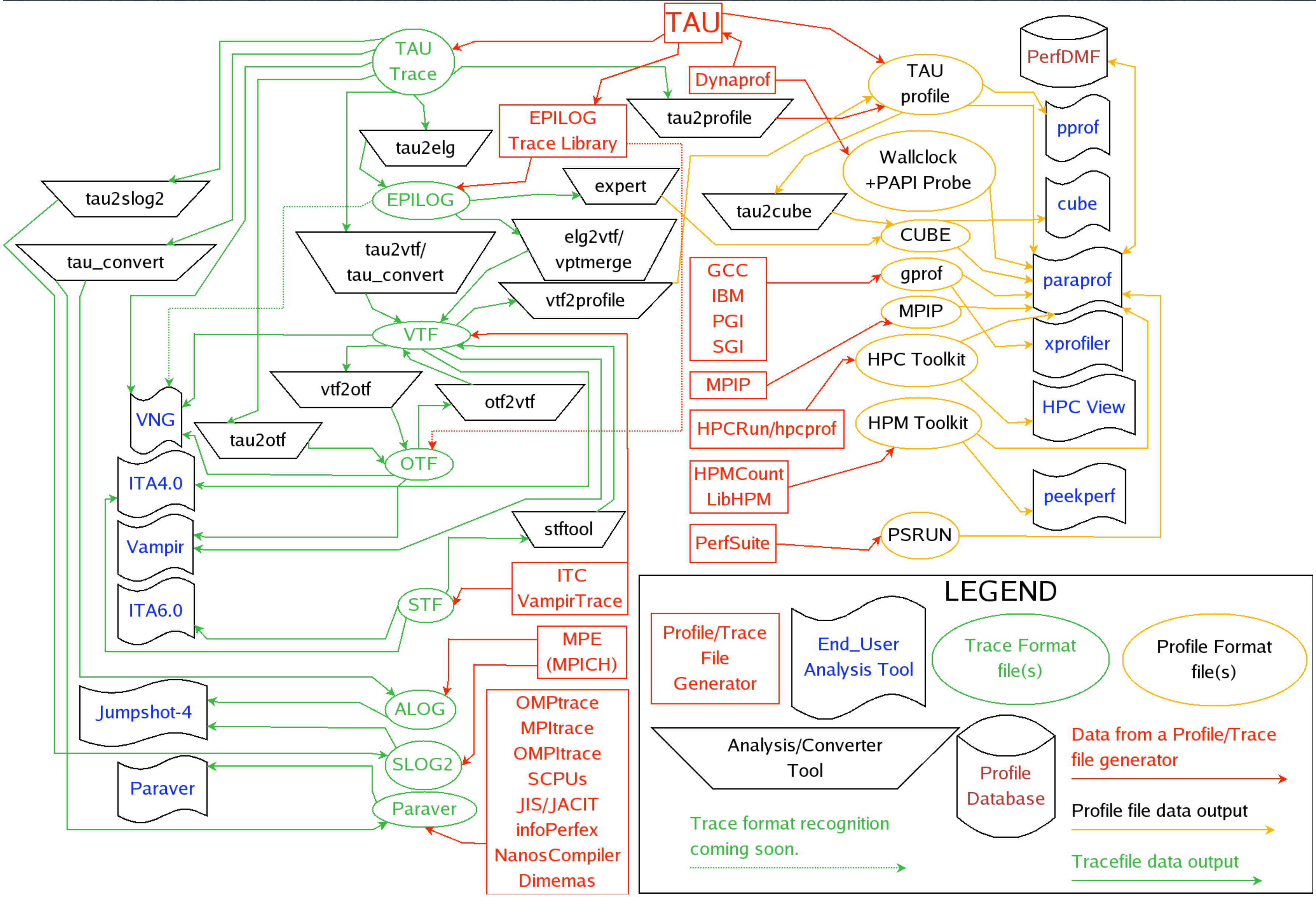
TAU Performance System Architecture



TAU Performance System Architecture



Building Bridges to Other Tools: TAU



TAU Instrumentation Approach

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

PAPI

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

TAU Instrumentation Mechanisms

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

Using TAU: A brief Introduction

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

  % setenv TAU_MAKEFILE /usr/tau-2.x/xt3/lib/Makefile.tau-mpi-pdt-pgi

  % 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

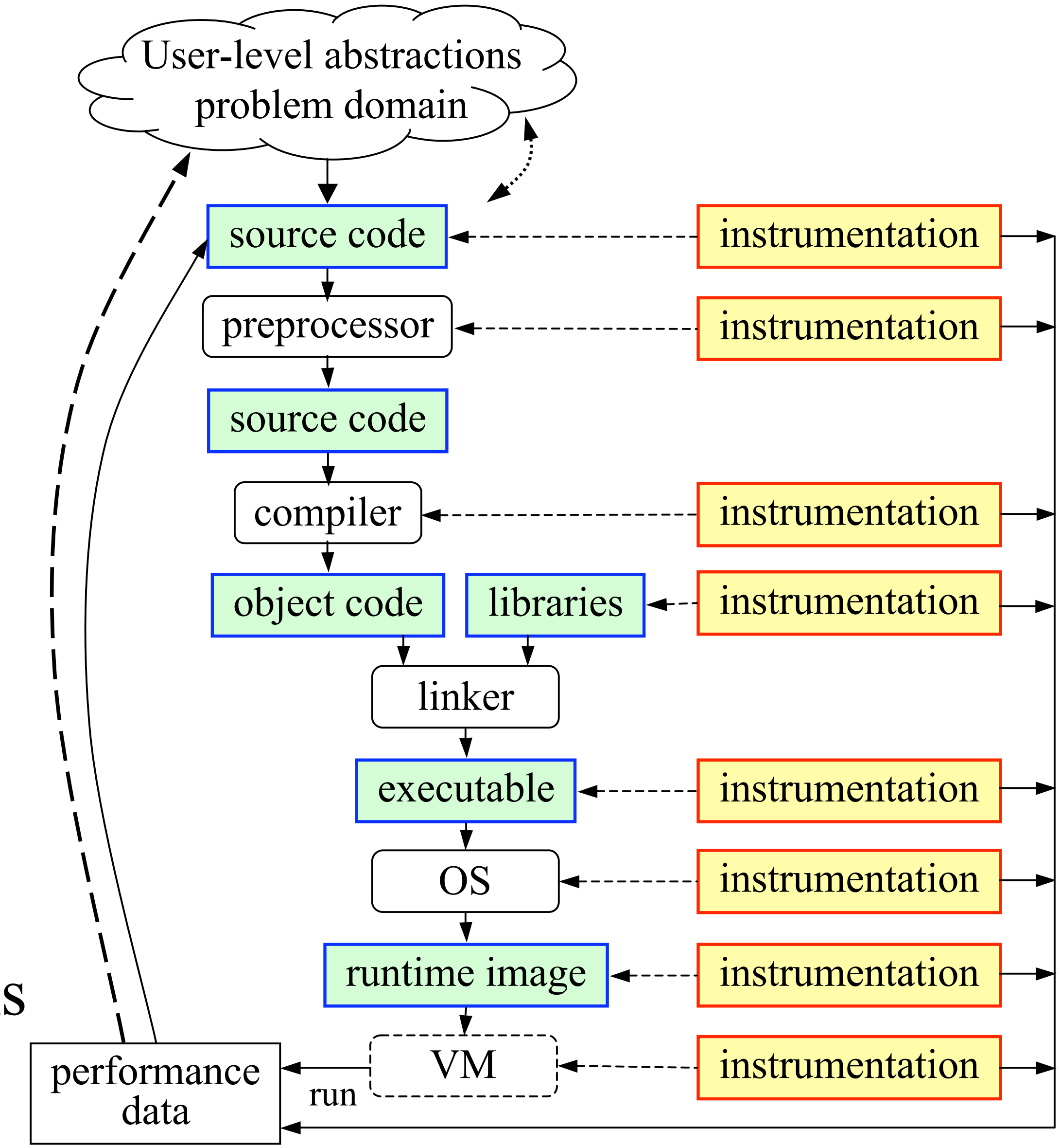
- Execute application and analyze performance data:

  % pprof (for text based profile display)

  % paraprof (for GUI)

Multi-Level Instrumentation and Mapping

- **Multiple interfaces**
- **Information sharing**
  - Between interfaces
- **Event selection**
  - Within/between levels
- **Mapping**
  - Associate performance data with high-level semantic abstractions



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 for CCA component software

TAU Measurement Mechanisms

- **Parallel profiling**
  - Function-level, block-level, statement-level
  - Supports user-defined events and mapping events
  - TAU parallel profile stored (dumped) during execution
  - Support for flat, callgraph/callpath, phase profiling
  - Support for memory profiling (headroom, malloc/leaks)
  - Support for tracking I/O (wrappers, Fortran instrumentation of read/write/print calls)
- **Tracing**
  - All profile-level events
  - Inter-process communication events
  - Inclusion of multiple counter data in traced events

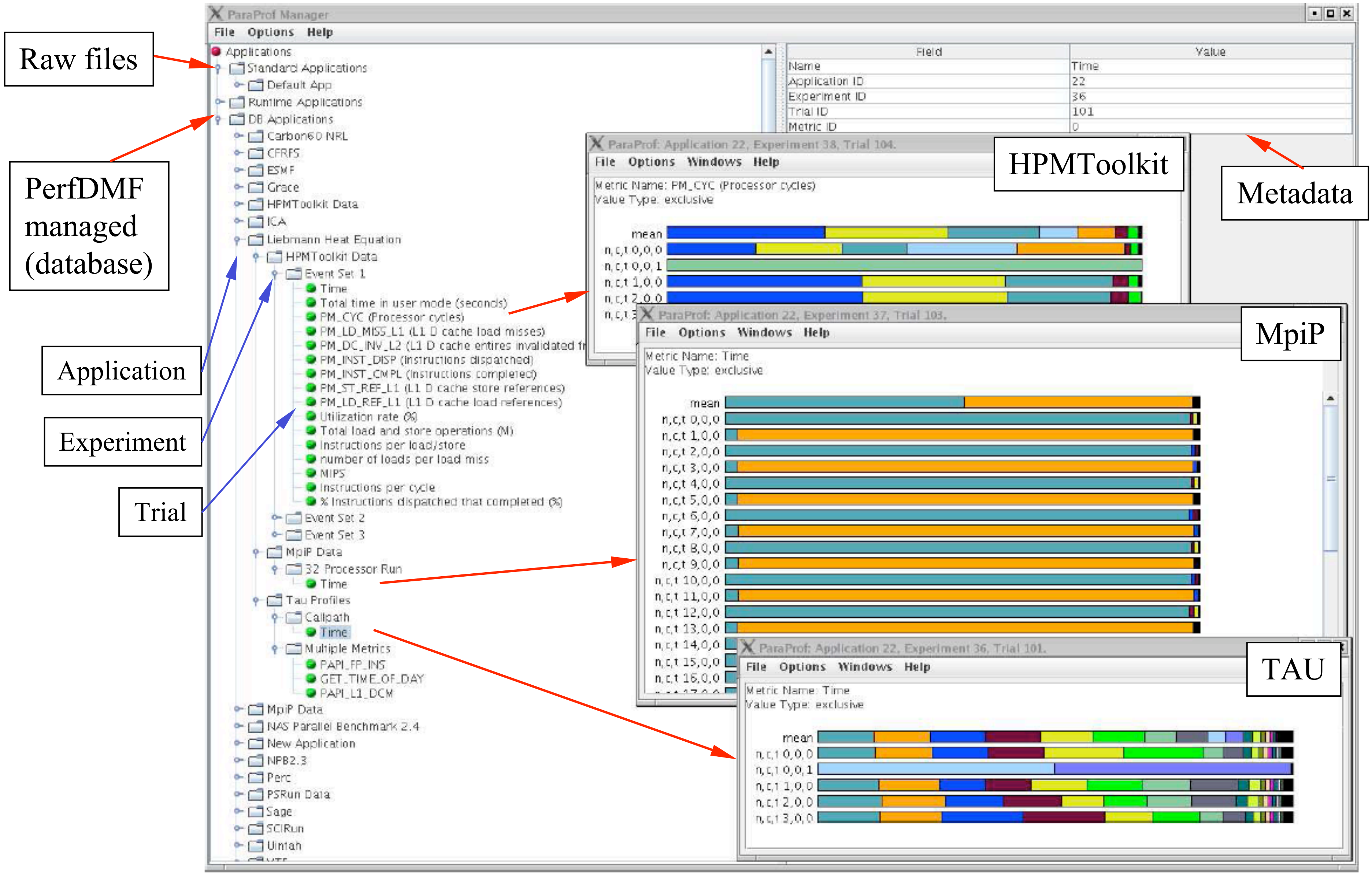
Types of Parallel Performance Profiling

- ***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

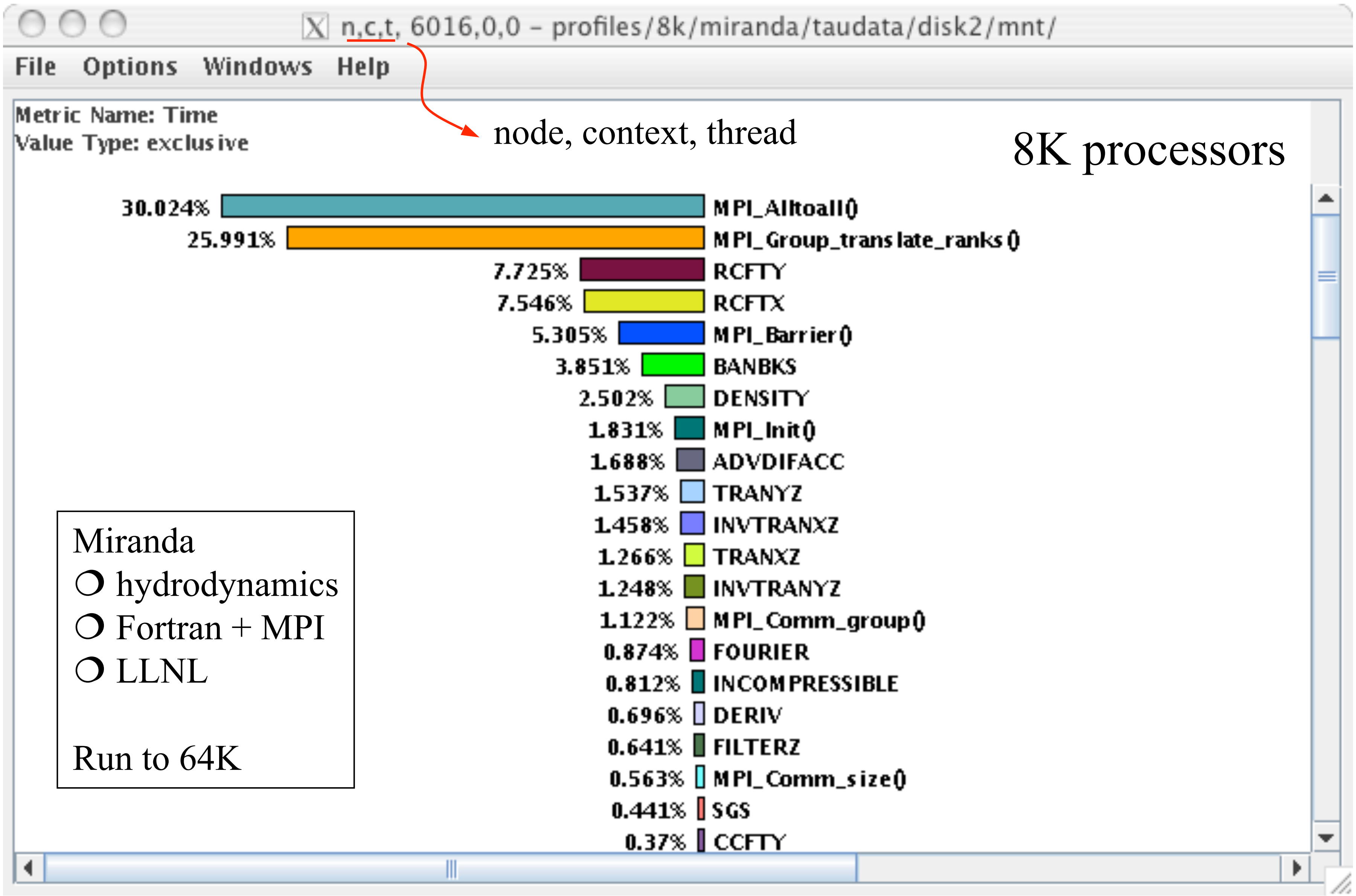
Performance Analysis and Visualization

- Analysis of parallel profile and trace measurement
- **Parallel profile analysis**
  - ***ParaProf***: parallel profile analysis and presentation
  - ***ParaVis***: parallel performance visualization package
  - Profile generation from trace data (***tau2profile***)
- Performance data management framework (***PerfDMF***)
- **Parallel trace analysis**
  - Translation to *VTF* (V3.0), *EPILOG, OTF* formats
  - Integration with *VNG* (Technical University of Dresden)
- Online parallel analysis and visualization
- Integration with *CUBE* browser (KOJAK, UTK, FZJ)

ParaProf Parallel Performance Profile Analysis



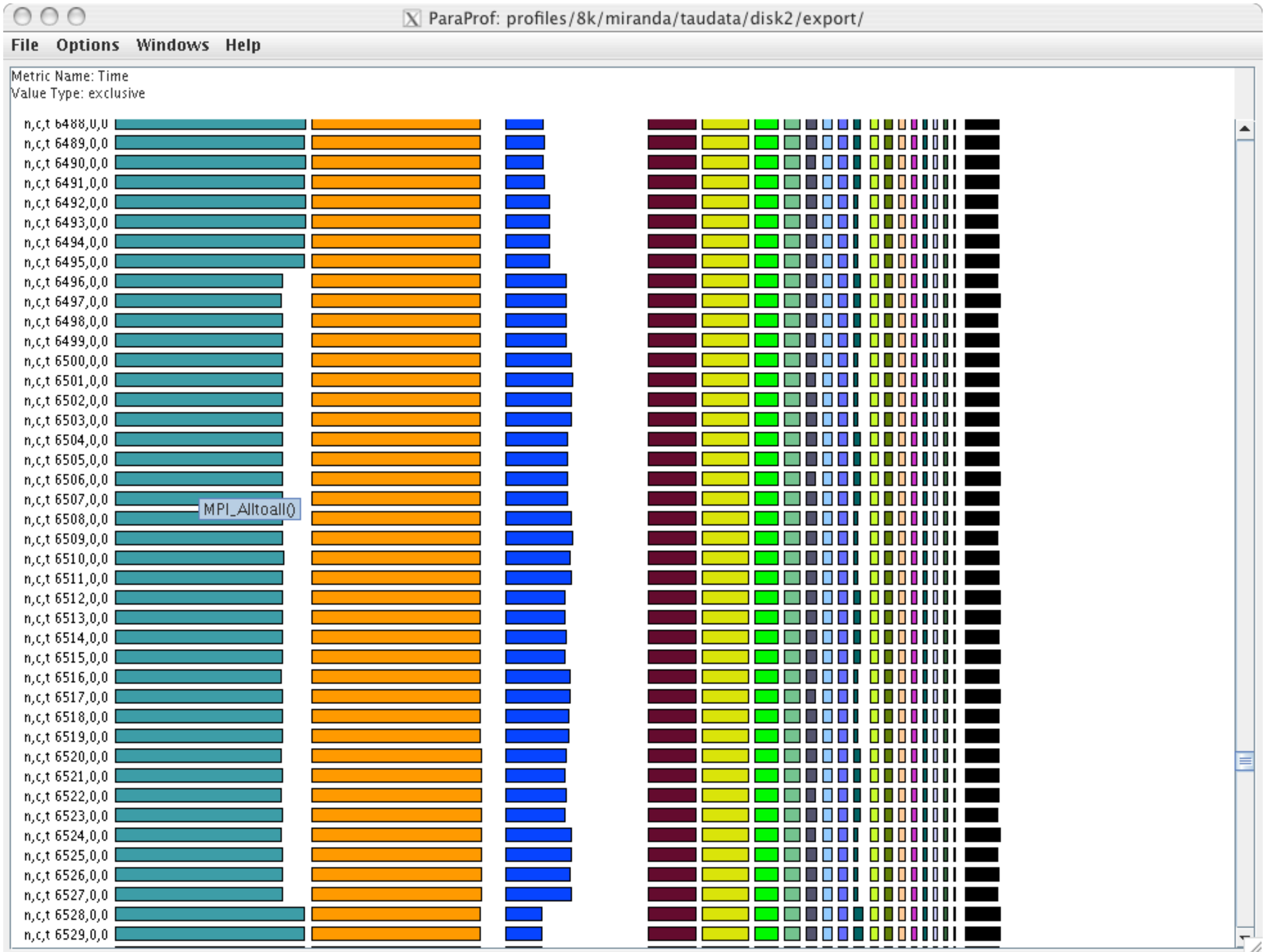
ParaProf – Flat Profile (Miranda, BG/L)



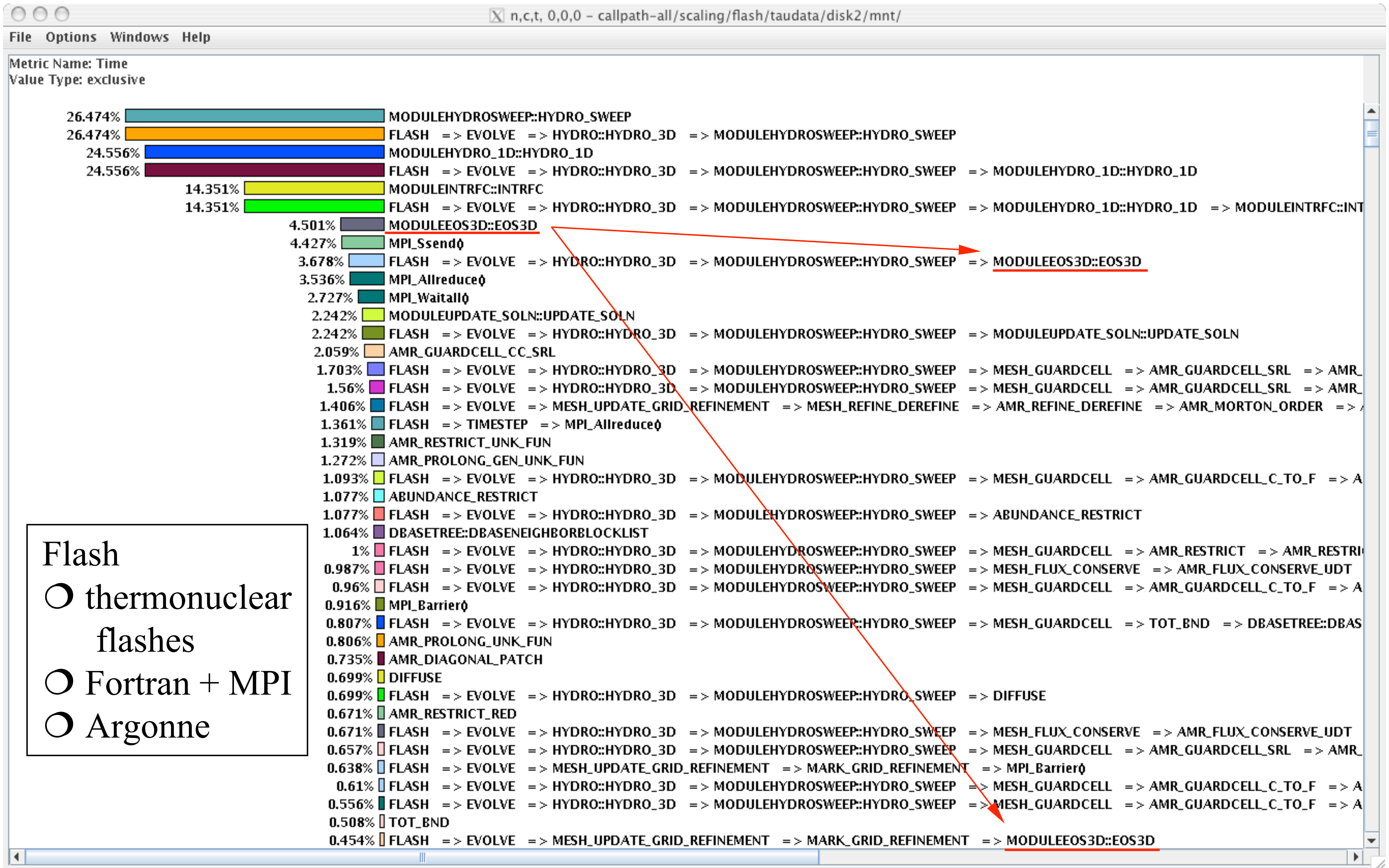
Miranda
- hydrodynamics
- Fortran + MPI
- LLNL

Run to 64K

ParaProf – Stacked View (Miranda)

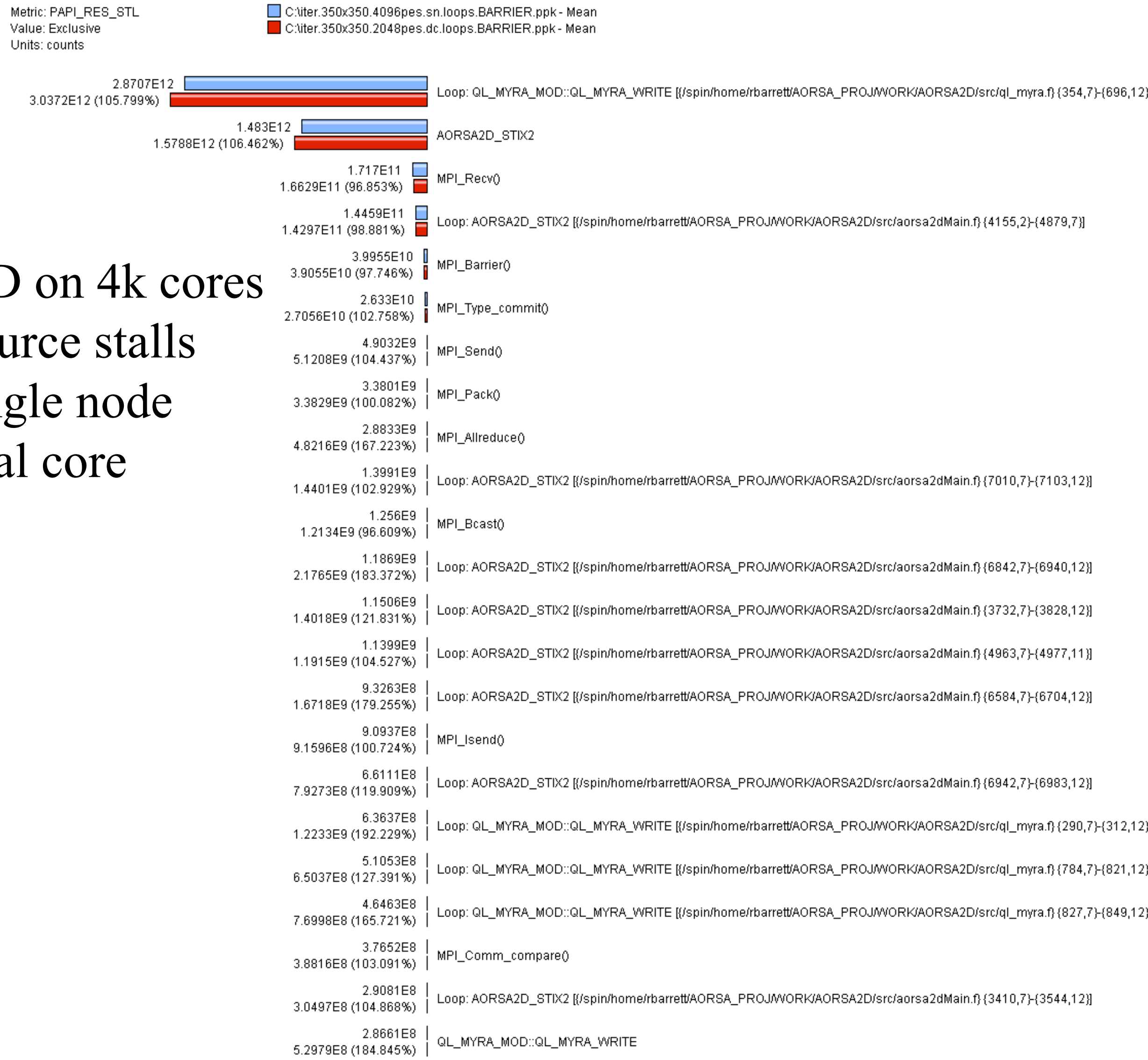


ParaProf – Callpath Profile (Flash)



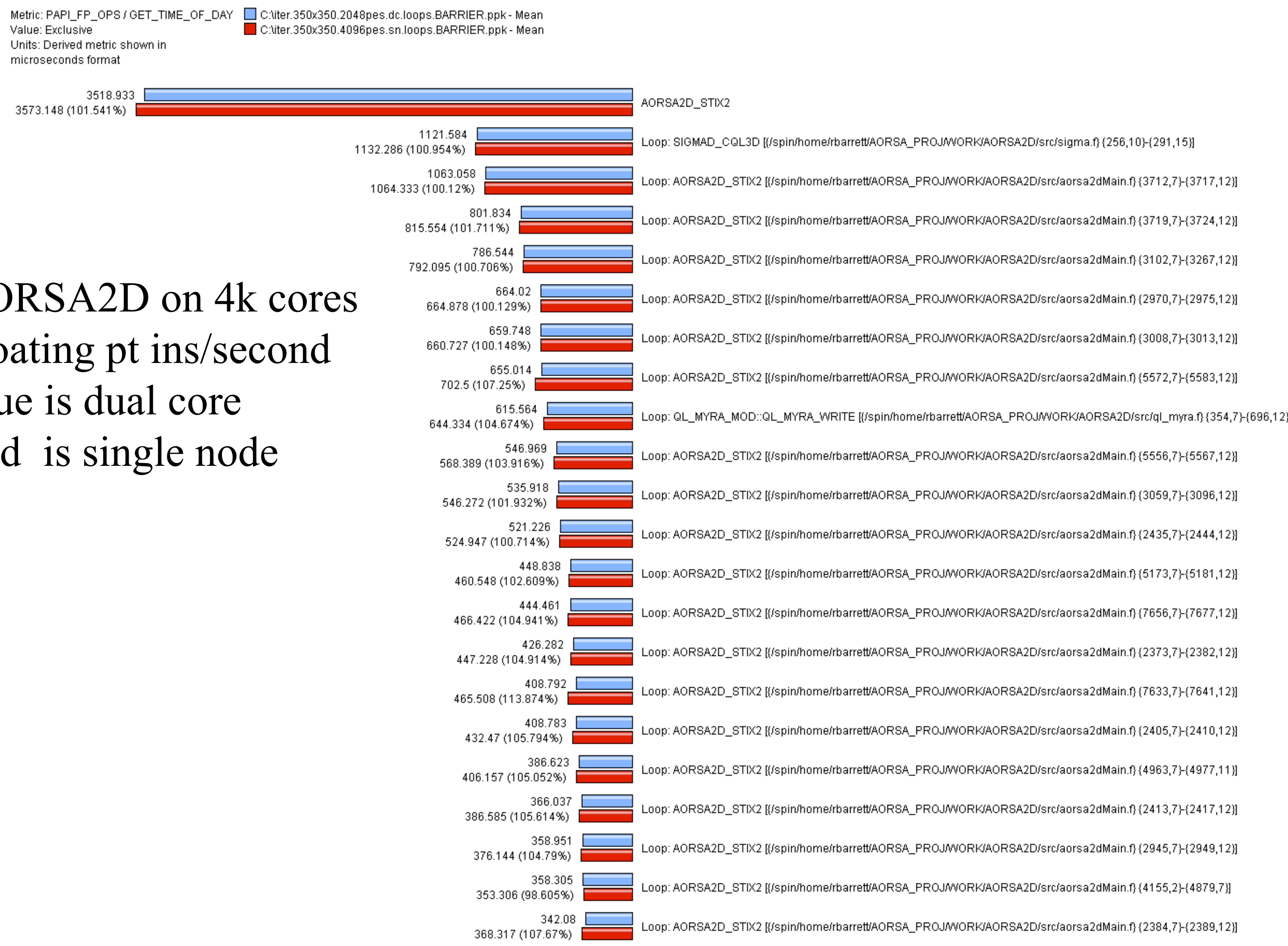
Flash
- thermonuclear flashes
- Fortran + MPI
- Argonne

Comparing Effects of MultiCore Processors



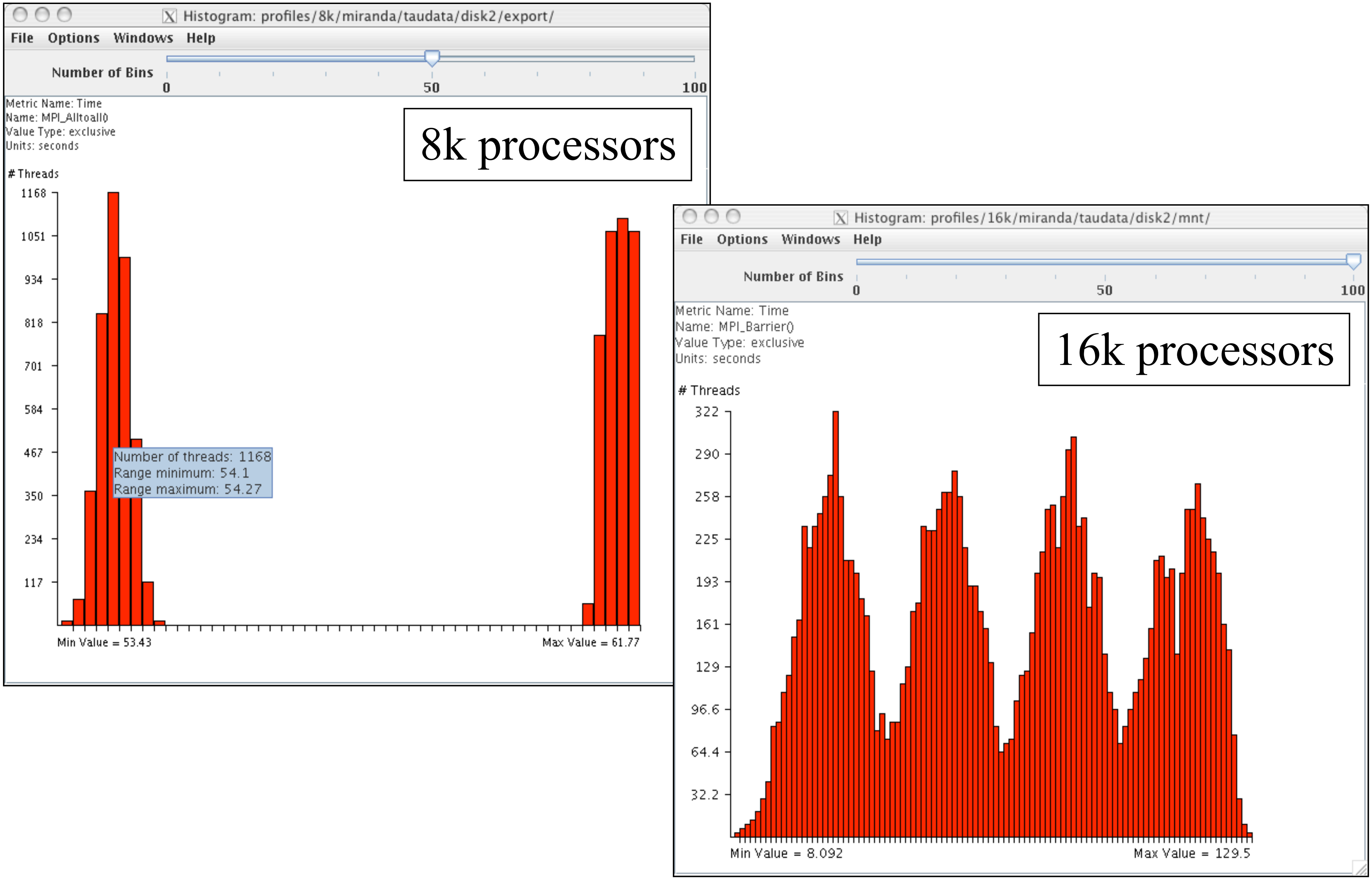
- AORSA2D on 4k cores
- PAPI resource stalls
- Blue is single node
- Red is dual core

Comparing FLOPS: MultiCore Processors

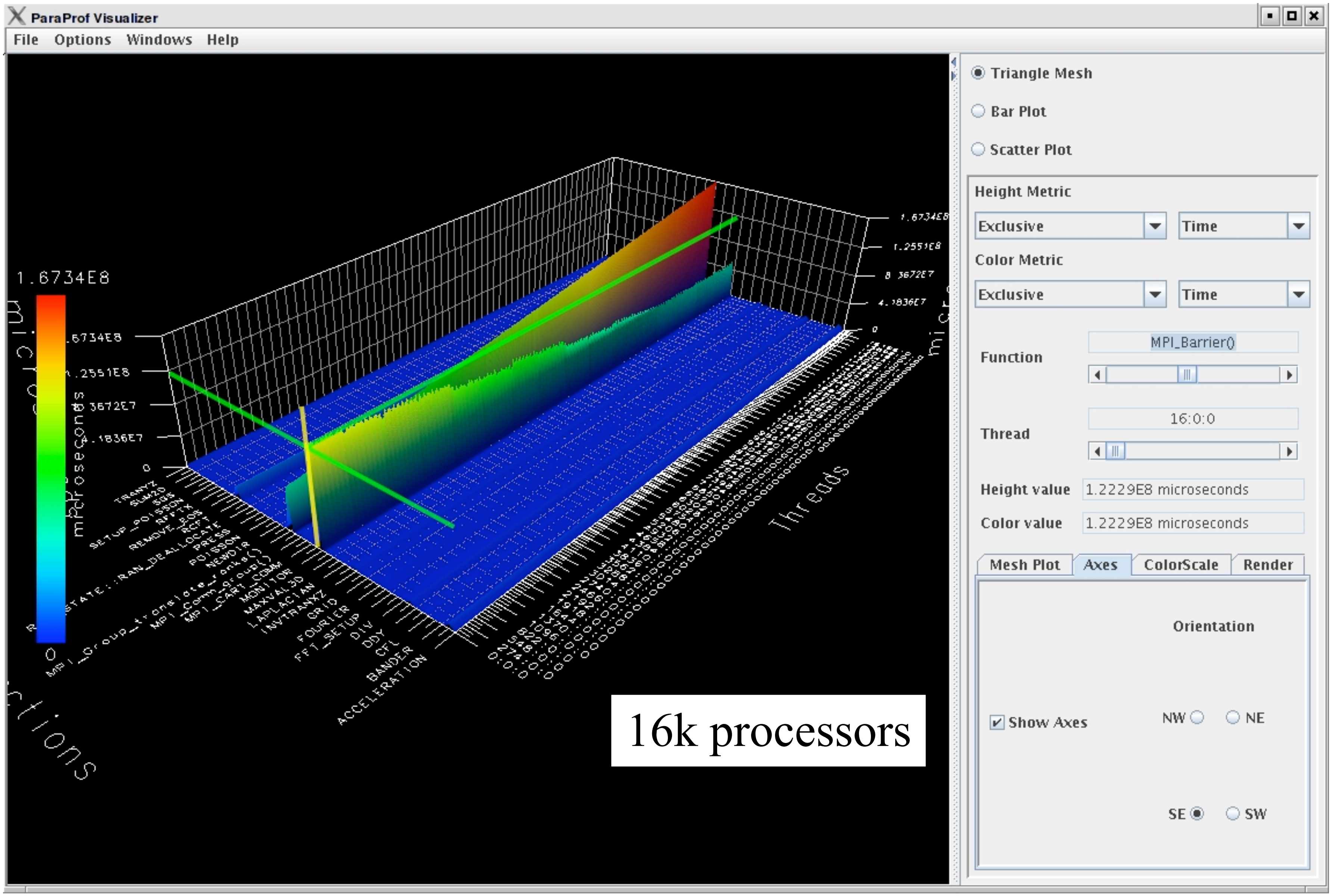


- AORSA2D on 4k cores
- Floating pt ins/second
- Blue is dual core
- Red is single node

ParaProf – Scalable Histogram View (Miranda)

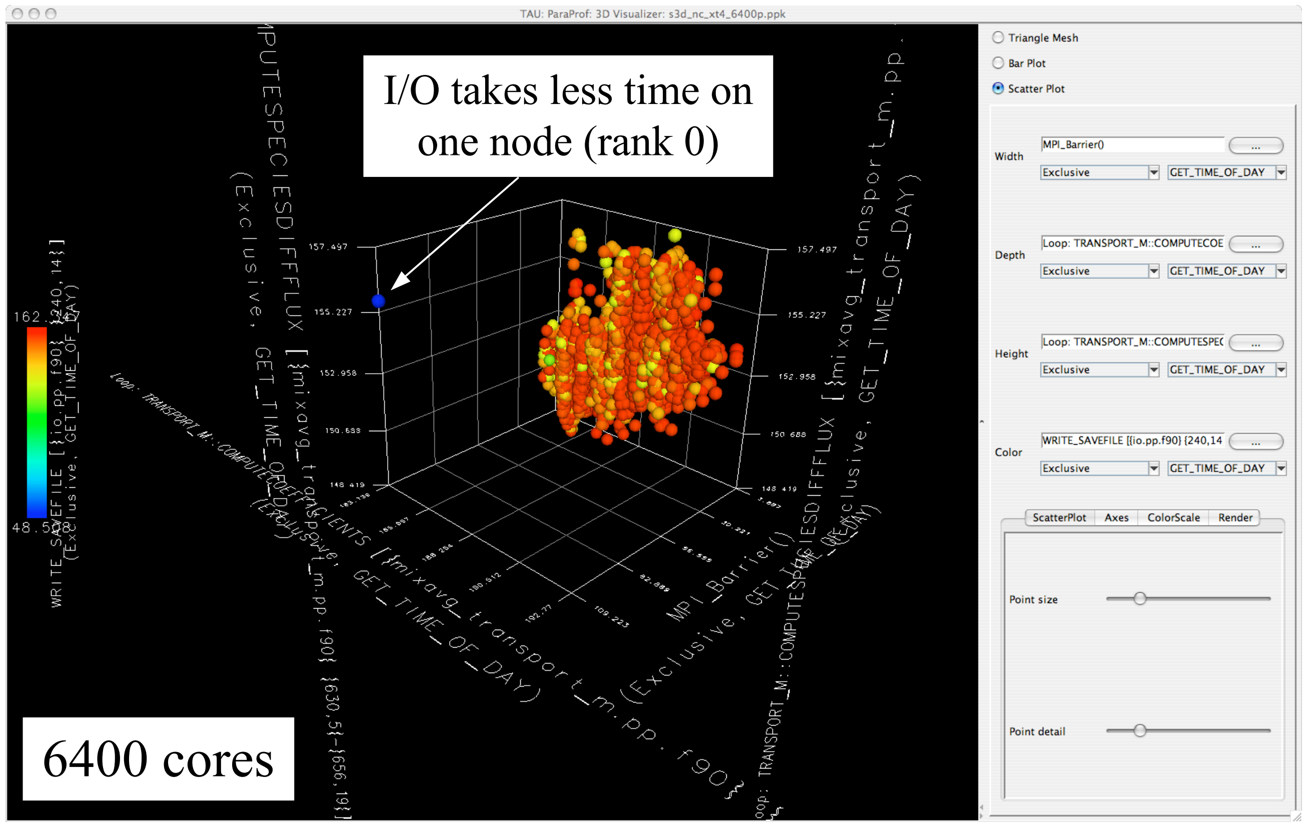


ParaProf – 3D Full Profile (Miranda)



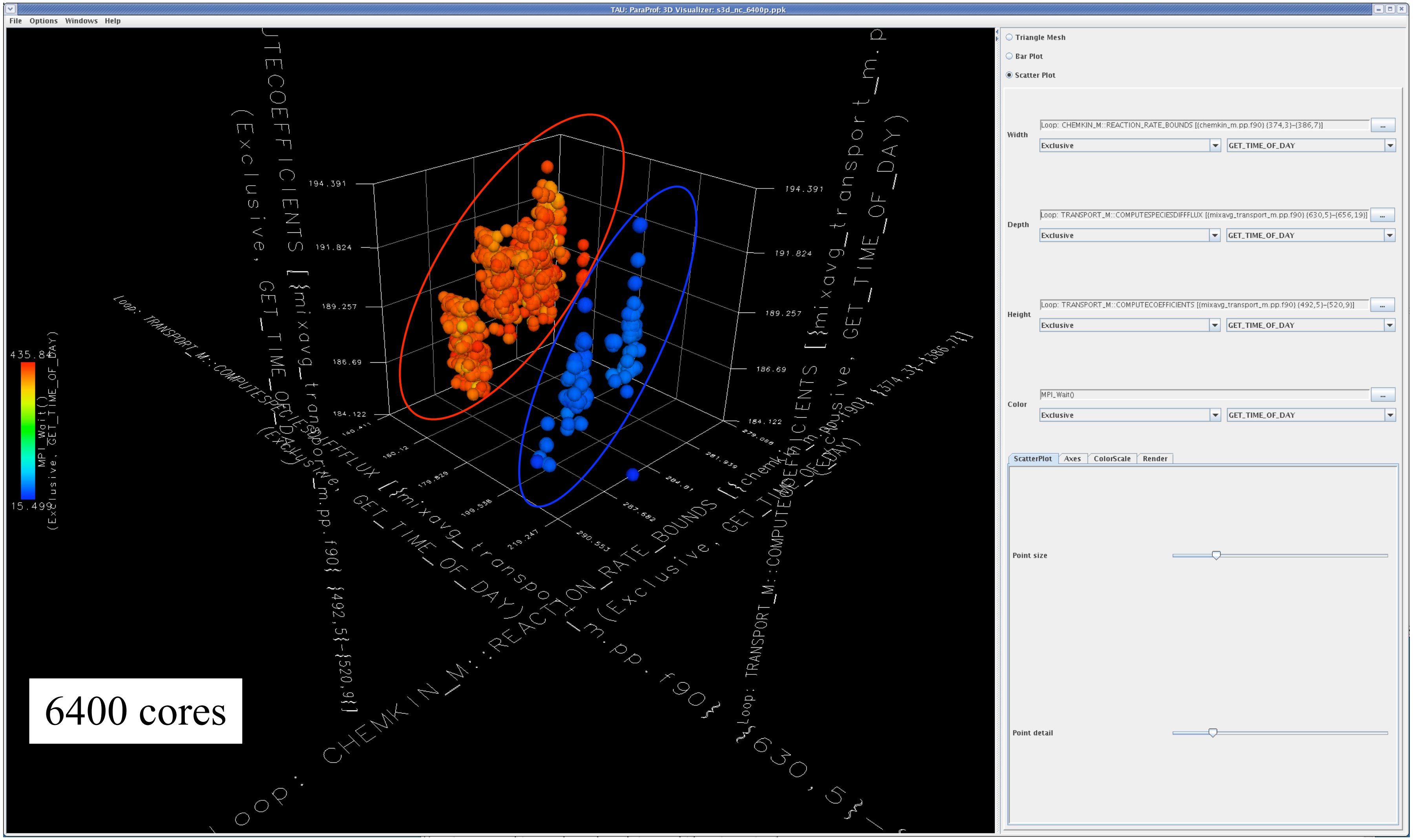
ParaProf – 3D Scatterplot (S3D – XT4 only)

- Each point is a "thread" of execution
- A total of four metrics shown in relation
- ParaVis 3D profile visualization library
  - JOGL



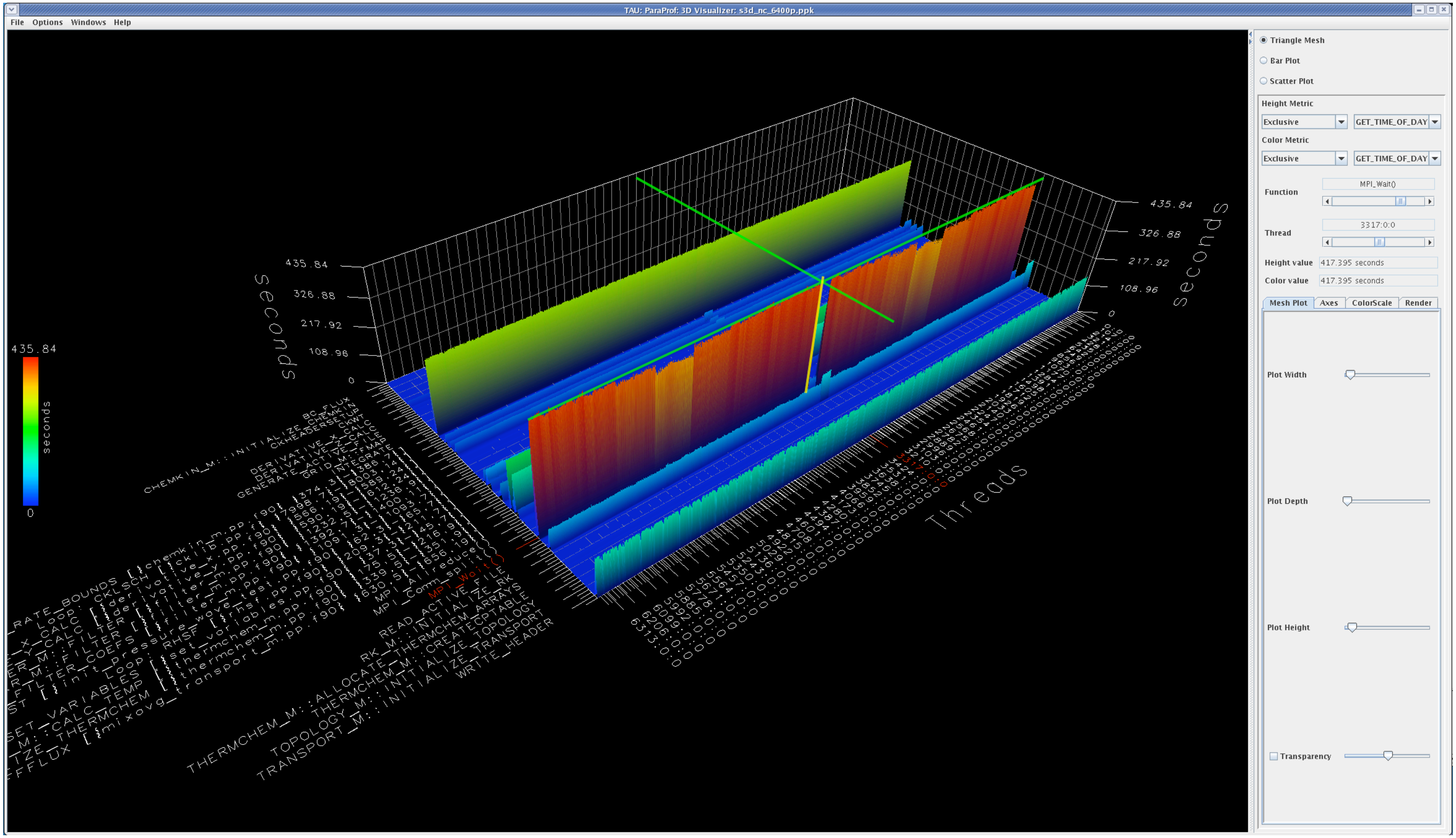
- Events (exclusive time metric)
  - MPI_Barrier(), two loops
  - write operation

S3D Scatter Plot: Visualizing Hybrid XT3+XT4



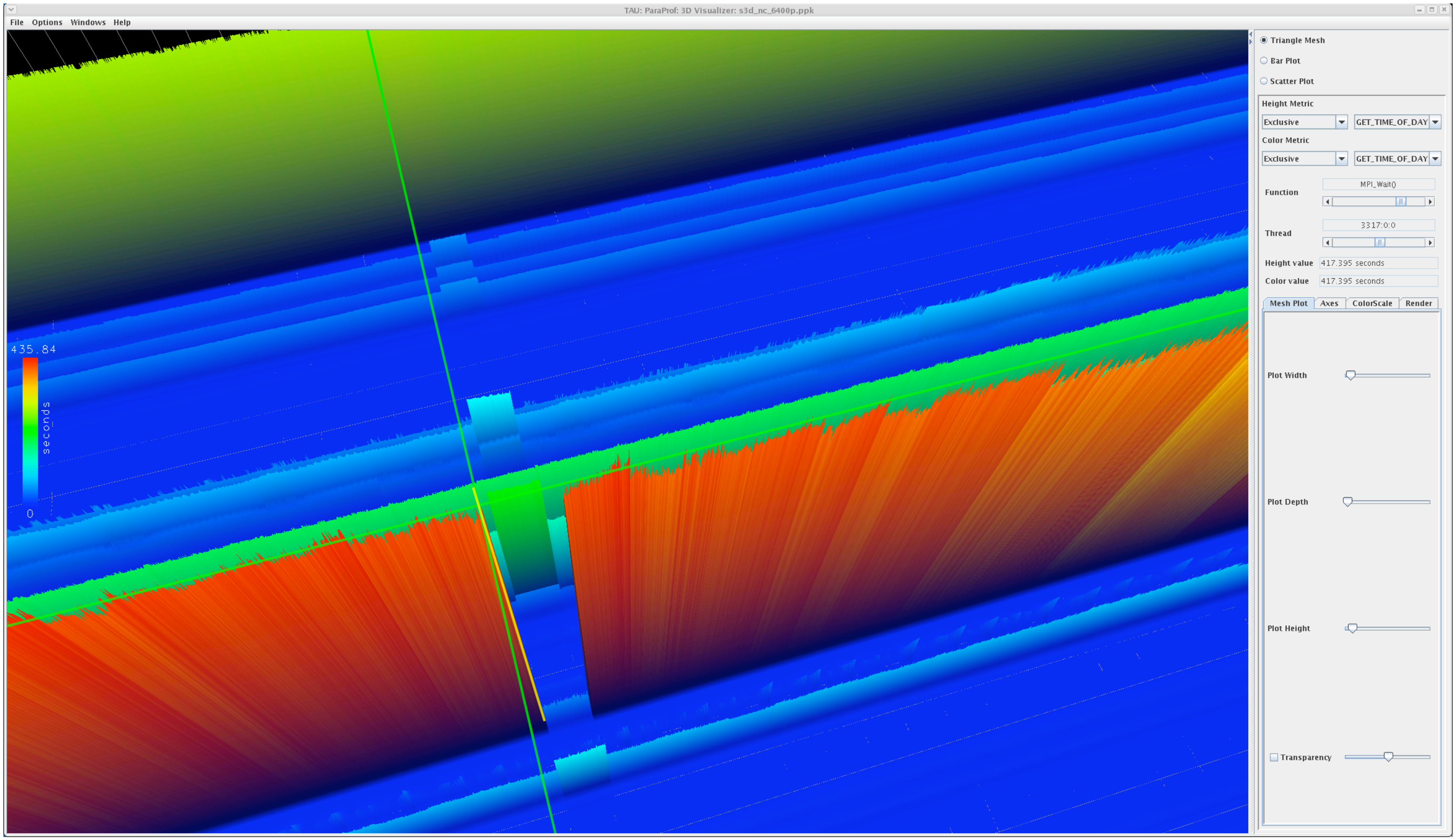
- Red nodes are XT4, blue are XT3

S3D: 6400 cores on XT3+XT4 System (Jaguar)



- Gap represents XT3 nodes

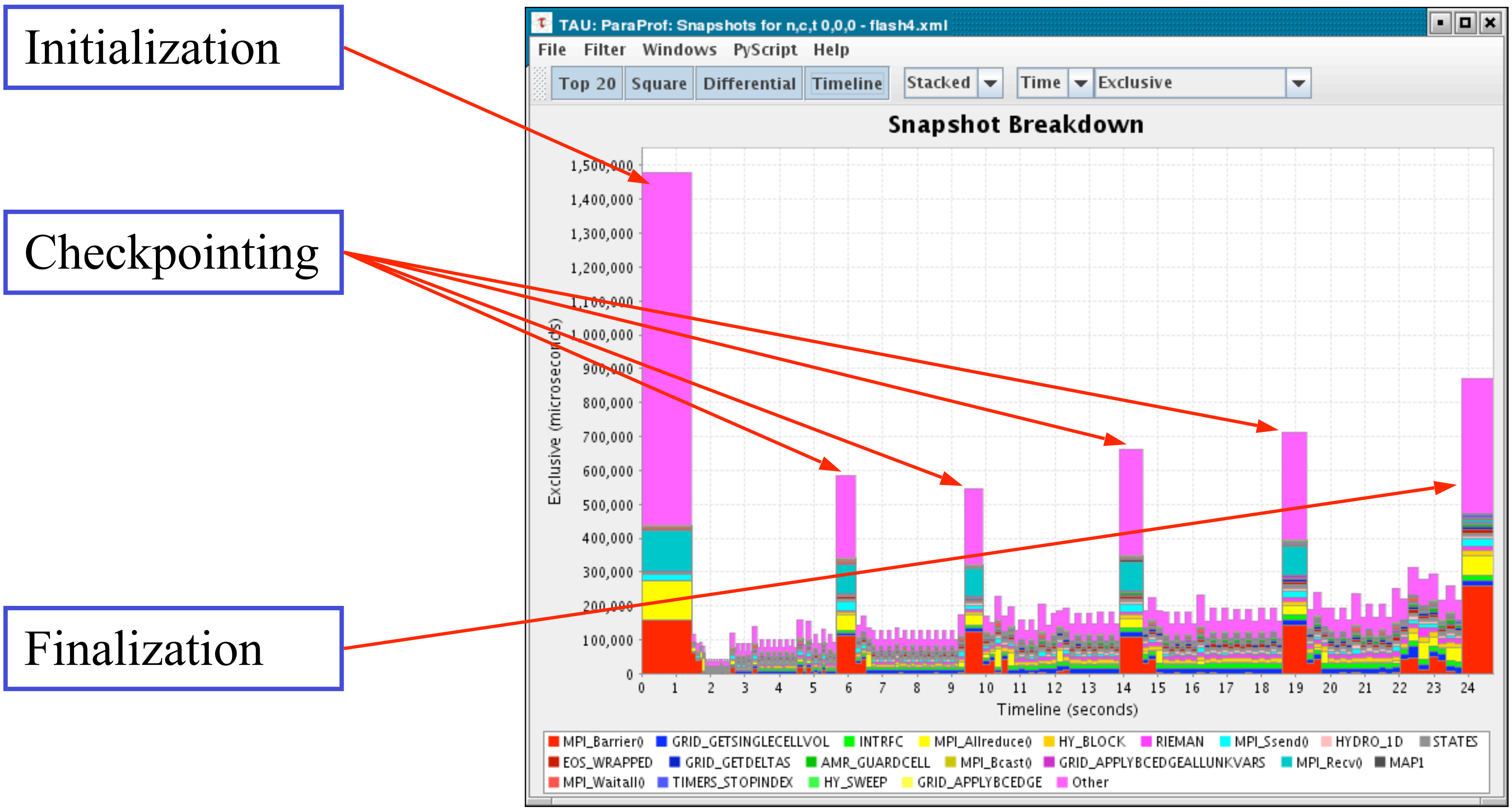
Visualizing S3D Profiles in ParaProf



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

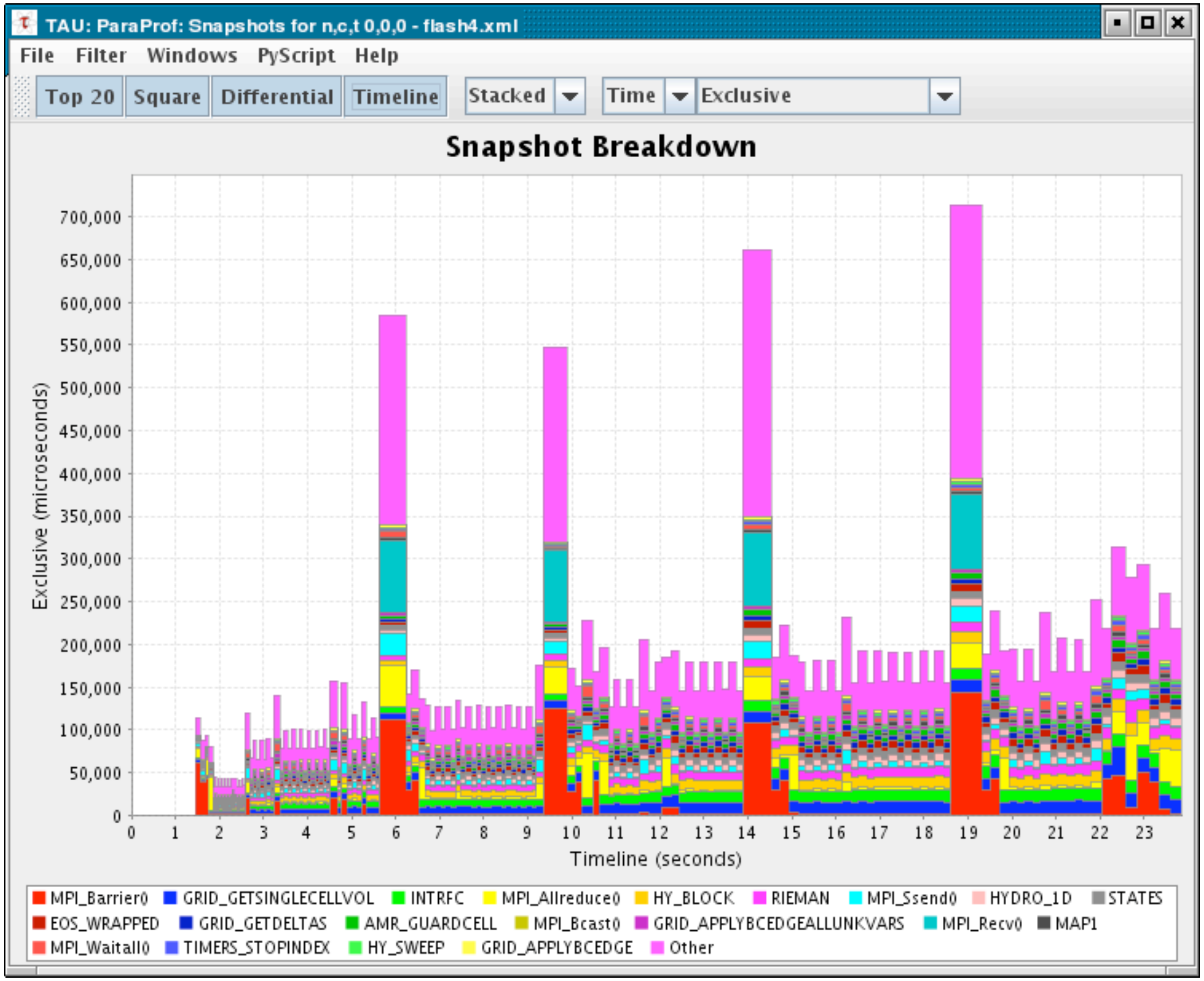
Profile Snapshots in ParaProf

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



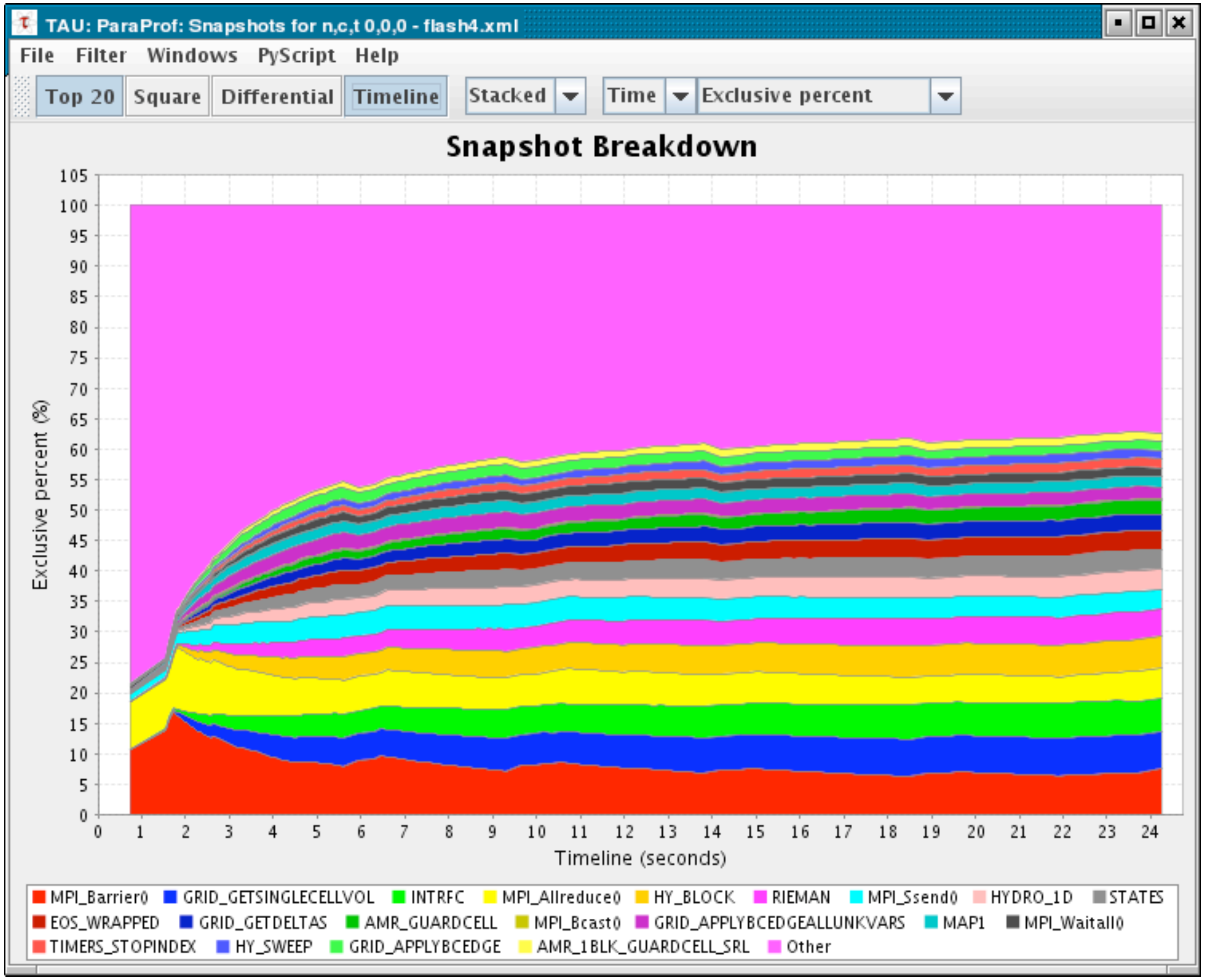
Profile Snapshots in ParaProf

- Filter snapshots (only show main loop iterations)

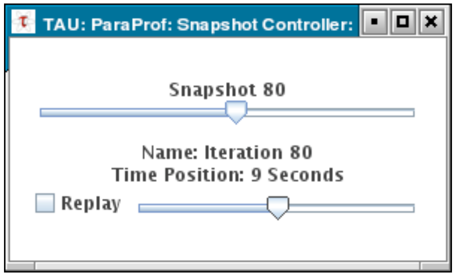


Profile Snapshots in ParaProf

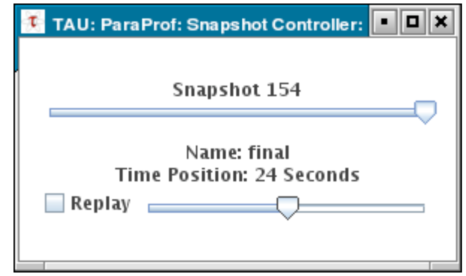
- Breakdown as a percentage

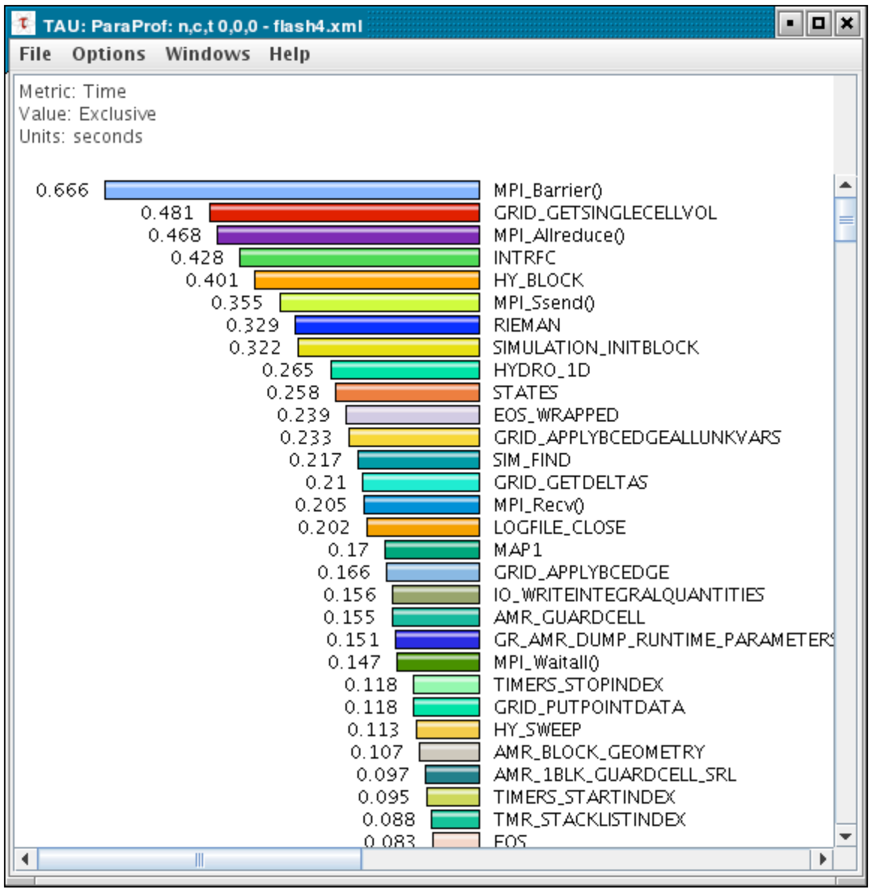


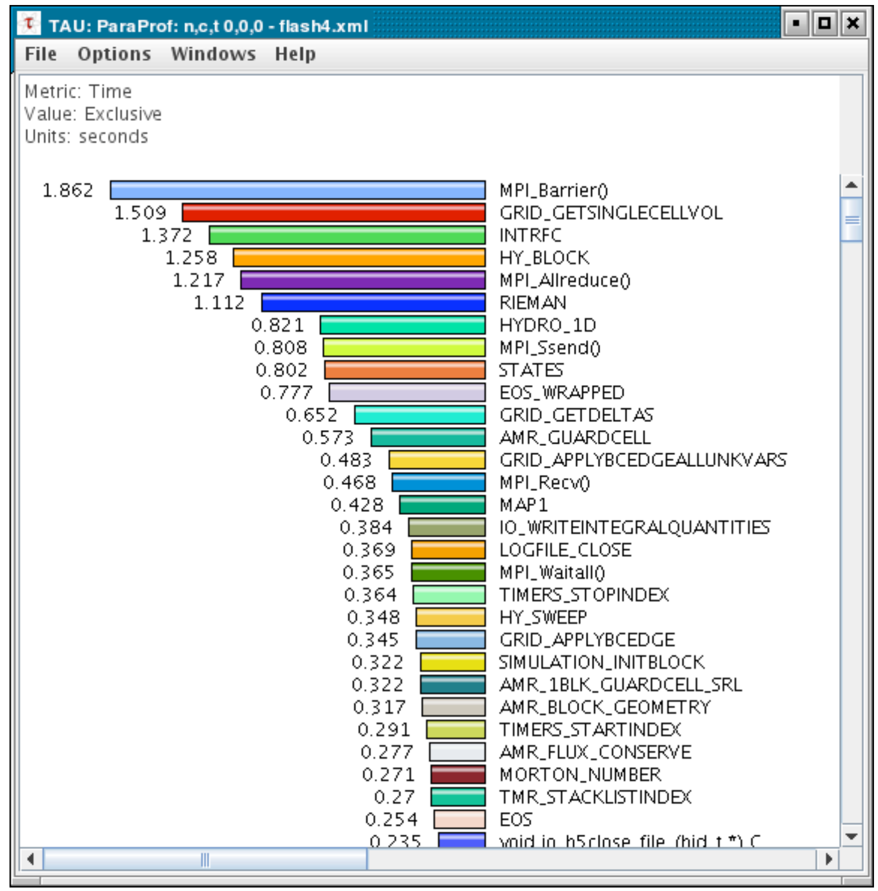
Snapshot replay in ParaProf



All windows dynamically update

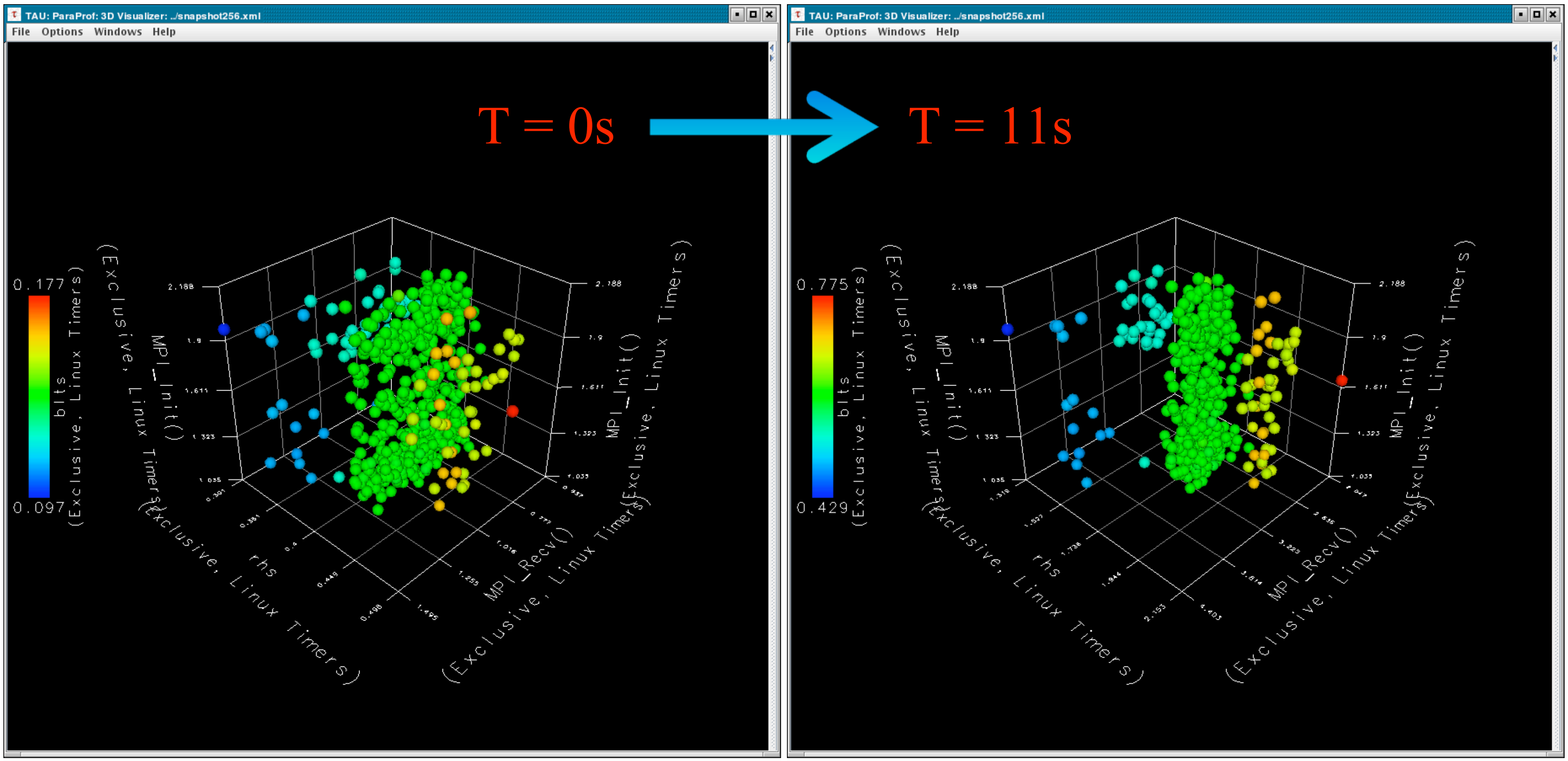




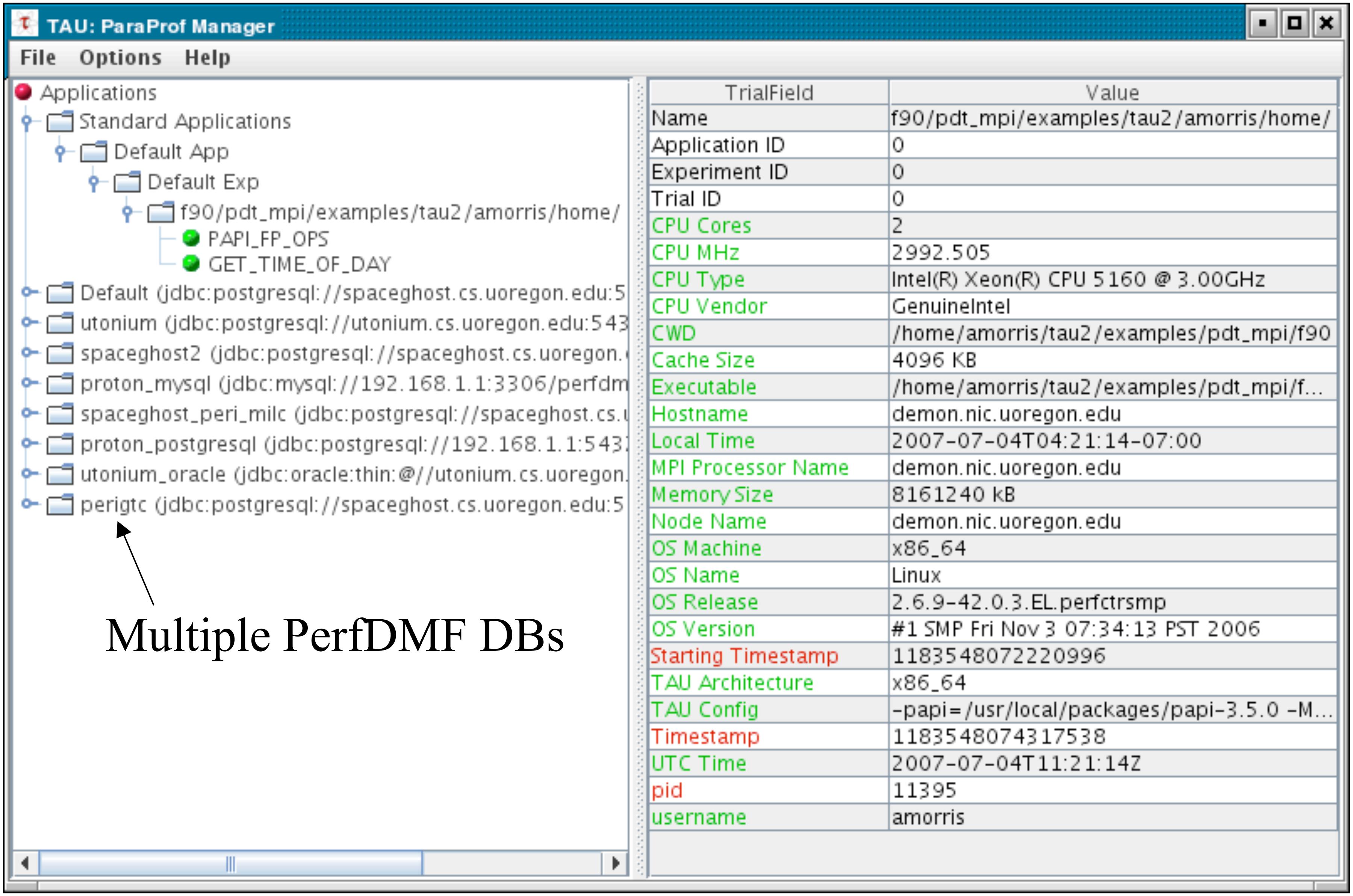


Profile Snapshots in ParaProf

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



New automated metadata collection



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

Multiple PerfDMF DBs

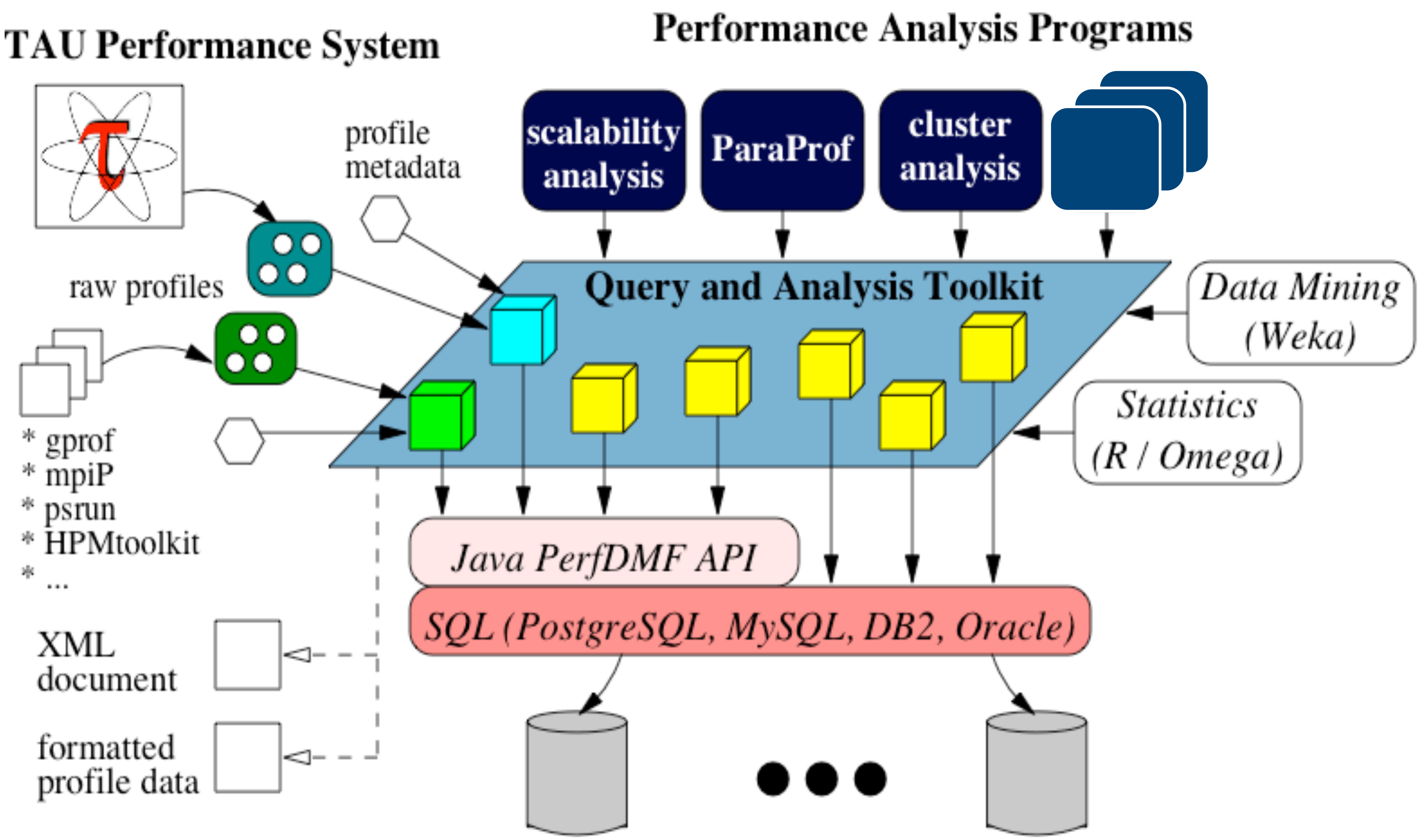
Performance Data Management: Motivation

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

PerfDMF Approach

- Performance Data Management Framework
- Originally designed to address critical TAU requirements
- Broader goal is to provide an open, flexible framework to support common data management tasks
- Extensible toolkit to promote integration and reuse across available performance tools
  - Supported profile formats:
    TAU, CUBE, Dynaprof, HPC Toolkit, HPM Toolkit, gprof, mpiP, psrun (PerfSuite), others in development
  - Supported DBMS:
    PostgreSQL, MySQL, Oracle, DB2, Derby/Cloudscape

PerfDMF Architecture



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

Recent PerfDMF Development

- Integration of XML metadata for each profile
  - Common Profile Attributes
  - Thread/process specific Profile Attributes
  - Automatic collection of runtime information
  - Any other data the user wants to collect can be added
    - Build information
    - Job submission information
  - Two methods for acquiring metadata:
    - TAU_METADATA() call from application
    - Optional XML file added when saving profile to PerfDMF
  - TAU Metadata XML schema is simple, easy to generate from scripting tools (no XML libraries required)

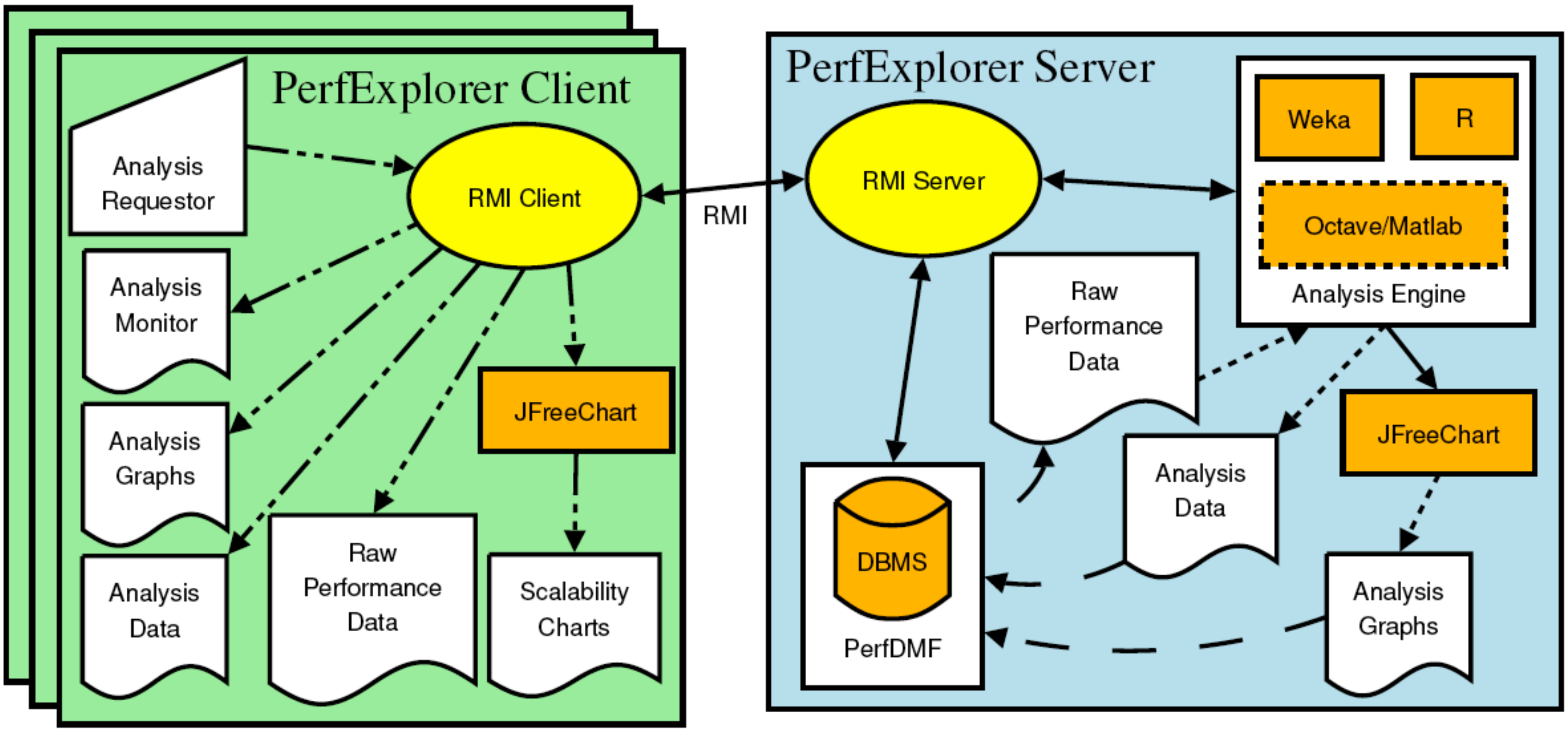
Performance Data Mining (Objectives)

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

Performance Data Mining (PerfExplorer)

- Performance knowledge discovery framework
  - Data mining analysis applied to parallel performance data
    - comparative, clustering, correlation, dimension reduction, …
  - Use the existing TAU infrastructure
    - TAU performance profiles, PerfDMF
  - 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)

Performance Data Mining (PerfExplorer)



K. Huck and A. Malony, "**PerfExplorer: A Performance Data Mining Framework For Large-Scale Parallel Computing**," SC 2005.

PerfExplorer Analysis Methods

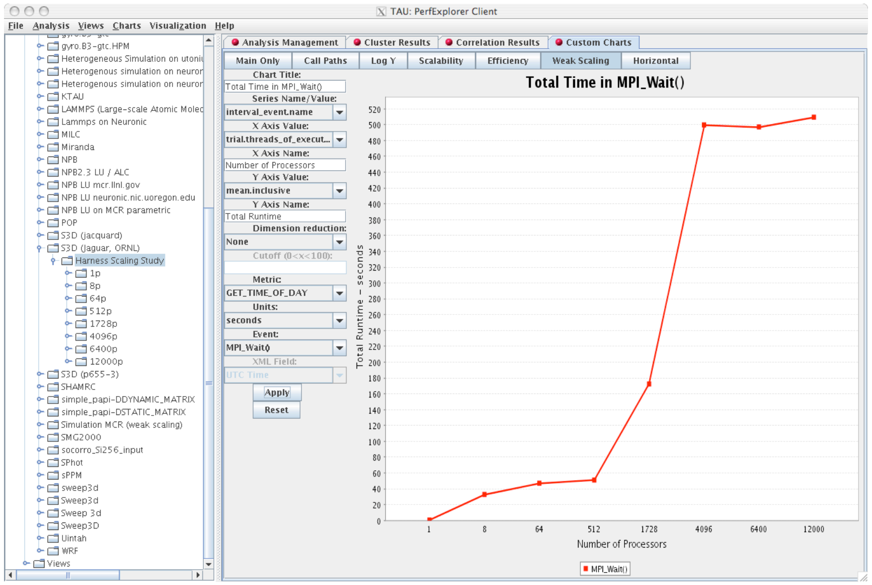
- Data summaries, distributions, scatterplots
- Clustering
  - *k*-means
  - Hierarchical
- Correlation analysis
- Dimension reduction
  - PCA
  - Random linear projection
  - Thresholds
- Comparative analysis
- Data management views

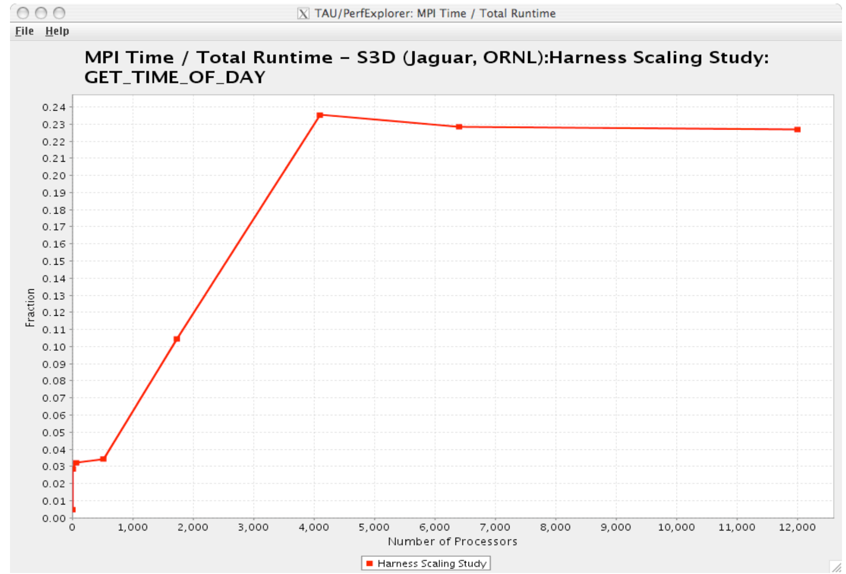
PerfDMF and the TAU Portal

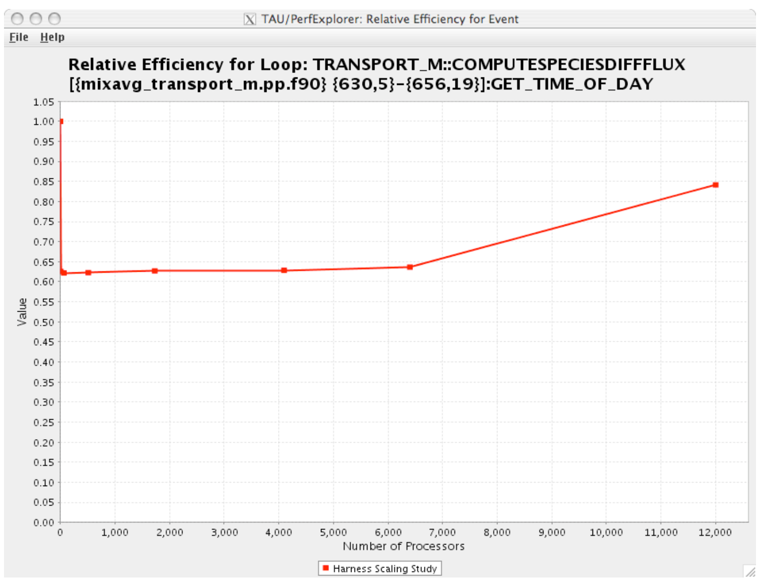
- Development of the TAU portal
  - Common repository for collaborative data sharing
  - Profile uploading, downloading, user management
  - Paraprof, PerfExplorer can be launched from the portal using Java Web Start (no TAU installation required)
- Portal URL

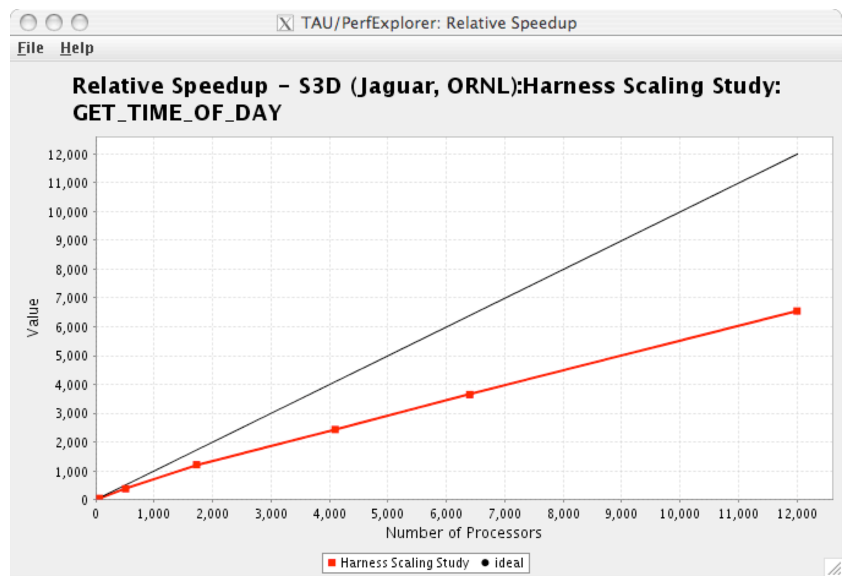
  http://tau.nic.uoregon.edu

PerfExplorer: Cross Experiment Analysis for S3D

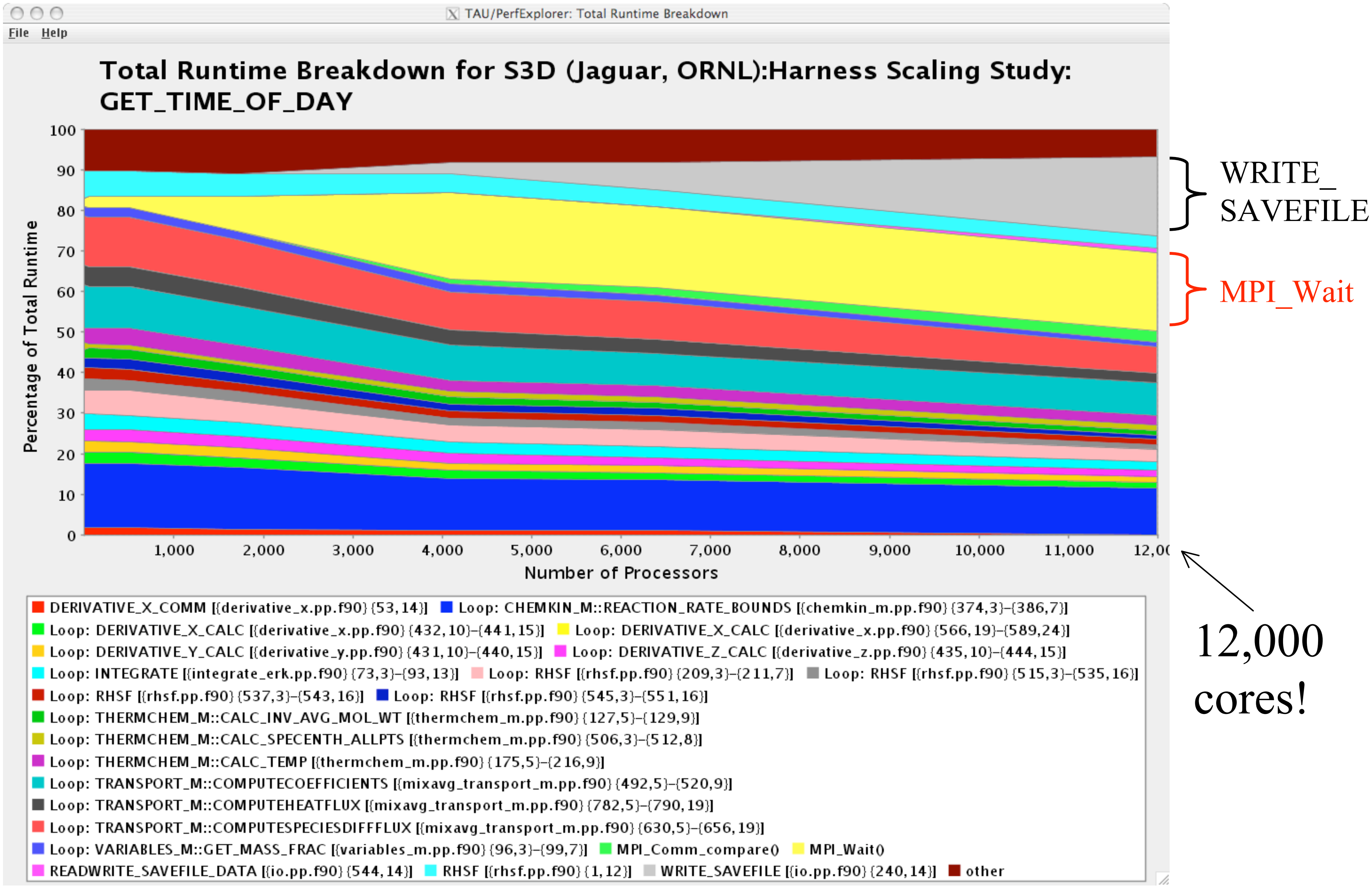








PerfExplorer: S3D Total Runtime Breakdown

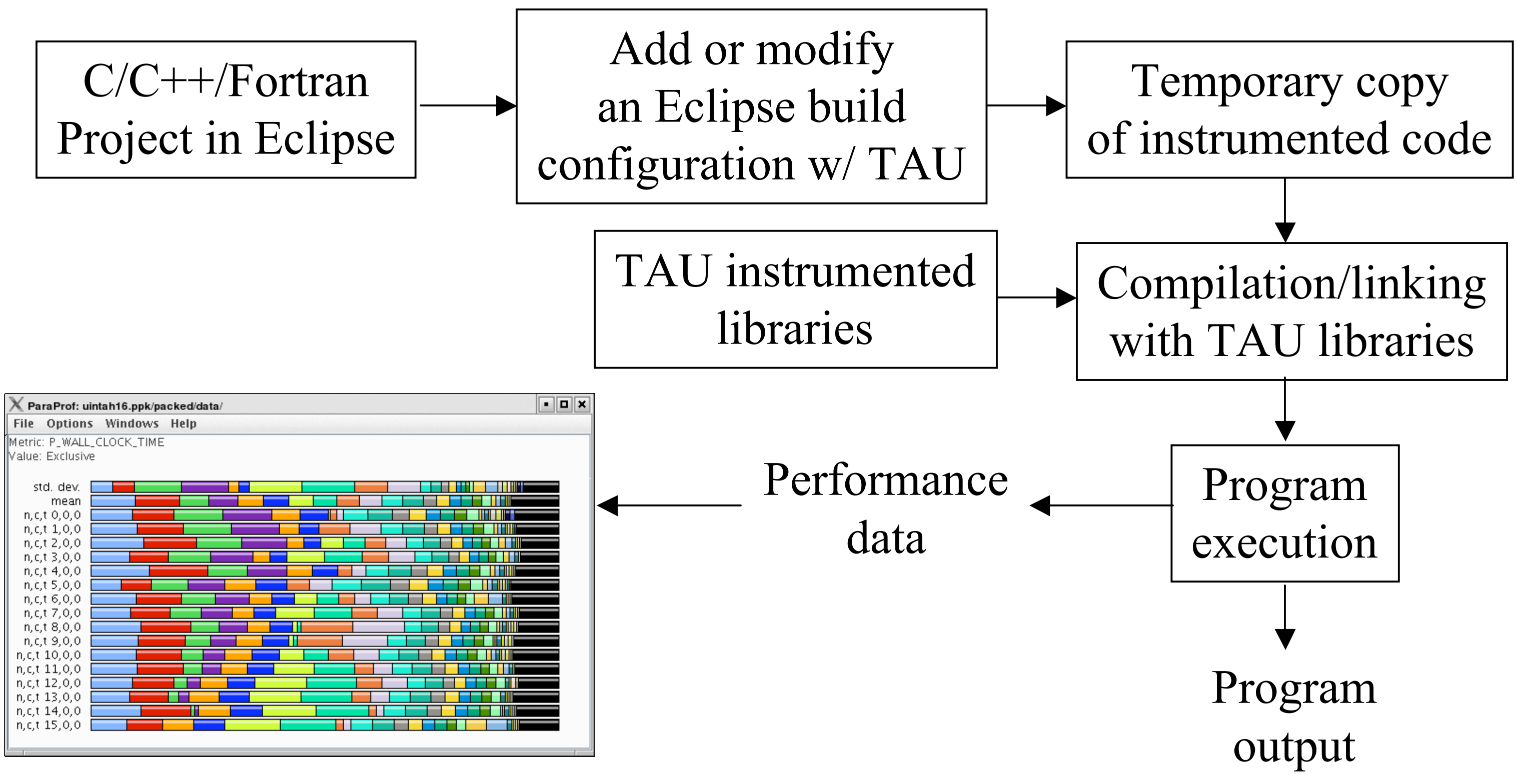


TAU Plug-Ins for Eclipse: Motivation

- 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

Adding TAU to Eclipse

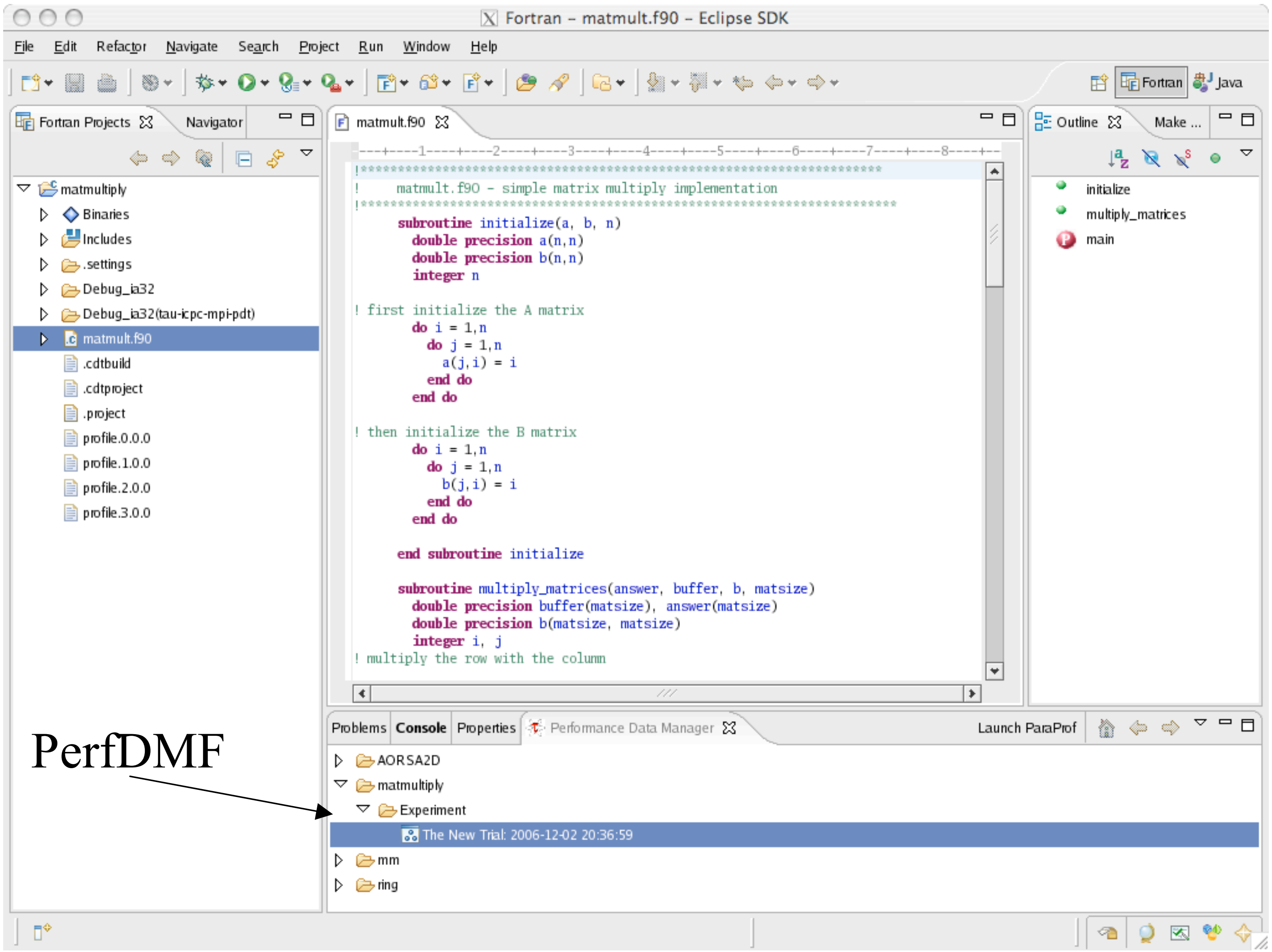
- Provide an interface for configuring TAU's automatic instrumentation within Eclipse's build system
- Manage runtime configuration settings and environment variables for execution of TAU instrumented programs



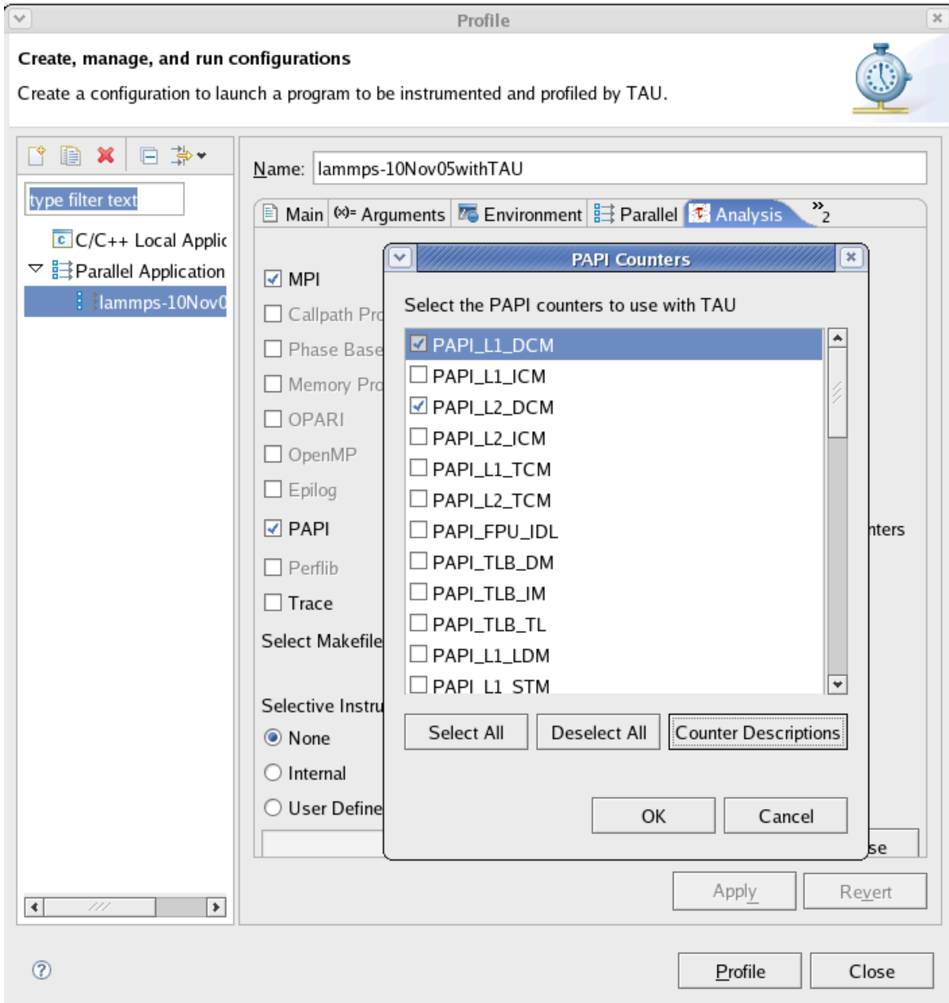
TAU Eclipse Plug-In Features

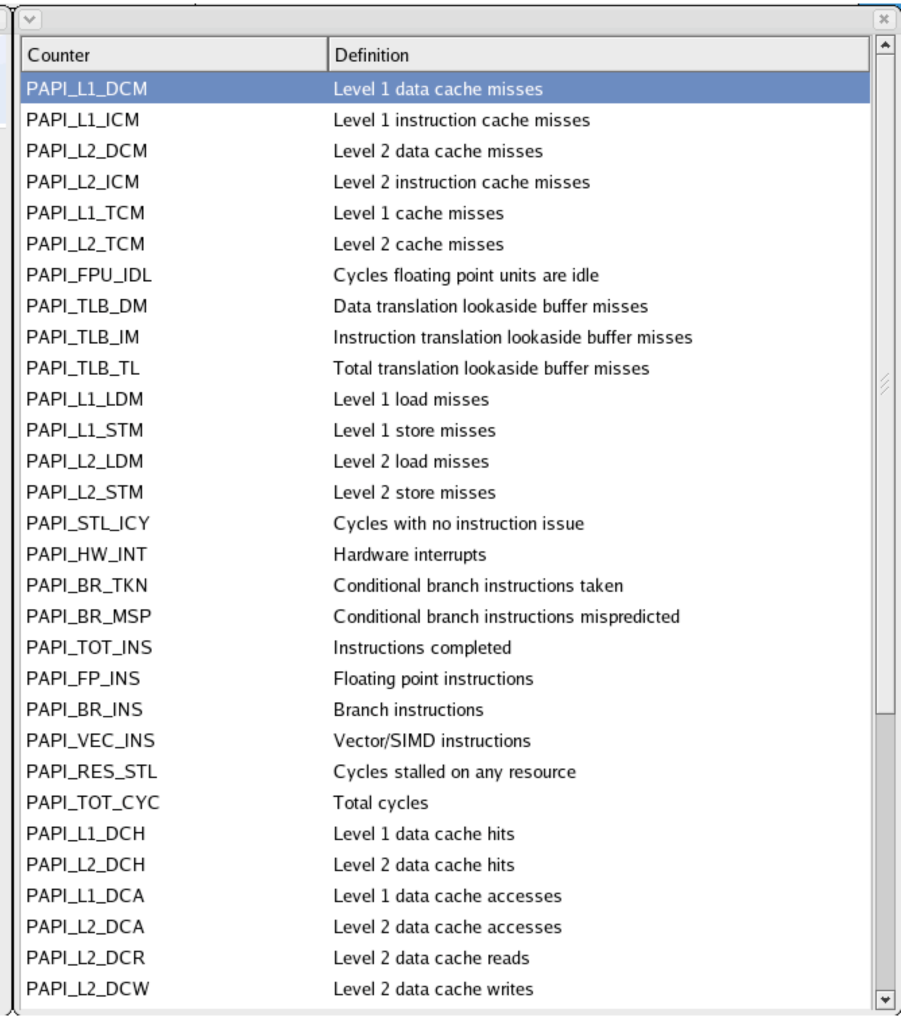
- Performance data collection
  - Graphical selection of TAU stub makefiles and compiler options
  - Automatic instrumentation, compilation and execution of target C, C++ or Fortran projects
  - Selective instrumentation via source editor and source outline views
  - Full integration with the Parallel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

TAU Eclipse Plug-In Features



Choosing PAPI Counters with TAU's in Eclipse





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

Future Plug-In Development

- Integration of additional TAU components
  - Automatic selective instrumentation based on previous experimental results
  - Trace format conversion from within Eclipse
- Trace and profile visualization within Eclipse
- Scalability testing interface
- Additional user interface enhancements

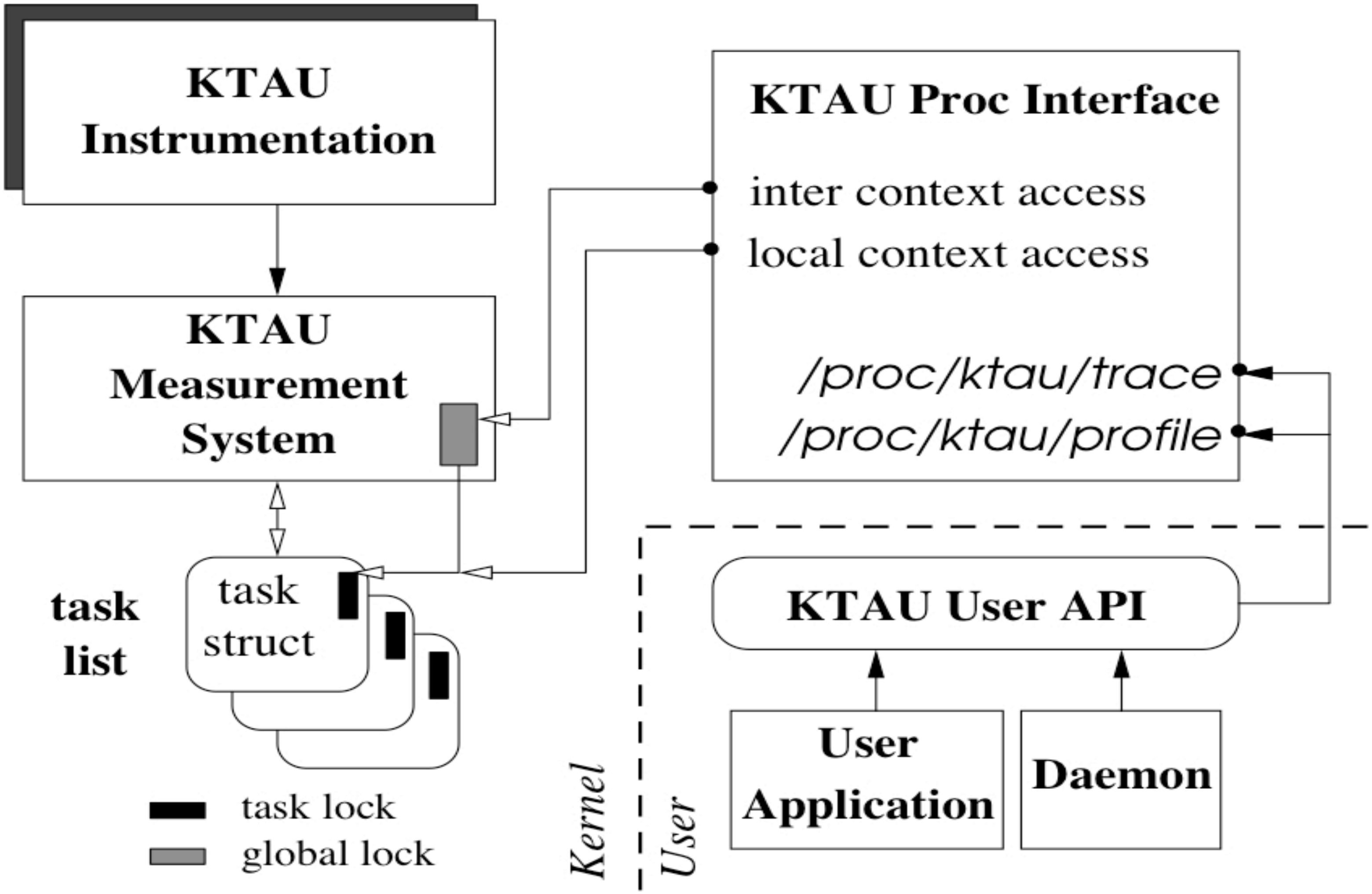
KTAU Project

- Trend toward Extremely Large Scales
  - System-level influences are increasingly dominant performance bottleneck contributors
  - Application sensitivity at scale to the system (e.g., OS noise)
  - Complex I/O path and subsystems another example
  - Isolating system-level factors non-trivial
- OS Kernel instrumentation and measurement is important to understanding system-level influences
- But can we closely correlate observed application and OS performance?
- KTAU / TAU (Part of the ANL/UO ZeptoOS Project)
  - Integrated methodology and framework to measure whole-system performance

Applying KTAU+TAU

- How does ***real*** OS-noise affect ***real*** applications on target platforms?
  - Requires a tightly coupled performance measurement & analysis approach provided by KTAU+TAU
  - Provides an estimate of application slowdown due to Noise (and in particular, different noise-components - IRQ, scheduling, etc)
  - Can empower both application and the middleware and OS communities.
  - A. Nataraj, A. Morris, A. Malony, M. Sottile, P. Beckman, "The Ghost in the Machine : Observing the Effects of Kernel Operation on Parallel Application Performance", SC'07.
- Measuring and analyzing complex, multi-component I/O subsystems in systems like BG(L/P) (work in progress).

KTAU System Architecture



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

TAU: Interoperability

- **What we can offer other tools:**
  - Automated source-level instrumentation (tau_instrumentor, PDT)
  - ParaProf 3D profile browser
  - PerfDMF database, PerfExplorer cross-experiment analysis tool
  - Eclipse/PTP plugins for performance evaluation tools
  - Conversion of trace and profile formats
  - Kernel-level performance tracking using KTAU
  - Support for most HPC platforms, compilers, MPI-1,2 wrappers

- **What help we need from other projects:**
  - Common API for compiler instrumentation
    - Scalasca/Kojak and VampirTrace compiler wrappers
    - Intel, Sun, GNU, Hitachi, PGI, …
  - Support for sampling for hybrid instrumentation/sampling measurement
    - HPCToolkit, PerfSuite
  - Portable, robust binary rewriting system that requires no root previleges
    - DyninstAPI
  - Scalable communication framework for runtime data analysis
    - MRNet, Supermon

Support Acknowledgements

- US 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
- US Department of Defense (DoD)
- NSF Software and Tools for High-End Computing
- Research Centre Juelich
- TU Dresden
- Los Alamos National Laboratory
- ParaTools, Inc.

TAU Transport Substrate - Motivations

- Transport Substrate
  - Enables movement of measurement-related data
  - TAU, in the past, has relied on shared file-system
- Some Modes of Performance Observation
  - Offline / Post-mortem observation and analysis
    - least requirements for a specialized transport
  - Online observation
    - long running applications, especially at scale
    - dumping to file-system can be suboptimal
  - Online observation with feedback into application
    - in addition, requires that the transport is bi-directional
- Performance observation problems and requirements are a function of the mode

Requirements

- Improve performance of transport
  - NFS can be slow and variable
  - Specialization and remoting of FS-operations to front-end
- Data Reduction
  - At scale, cost of moving data too high
  - Sample in different domain (node-wise, event-wise)
- Control
  - Selection of events, measurement technique, target nodes
  - What data to output, how often and in what form?
  - Feedback into the measurement system, feedback into application
- Online, distributed processing of generated performance data
  - Use compute resource of transport nodes
  - Global performance analyses within the topology
  - Distribute statistical analyses
- Scalability, most important - All of above at very large scales

Approach and Prototypes

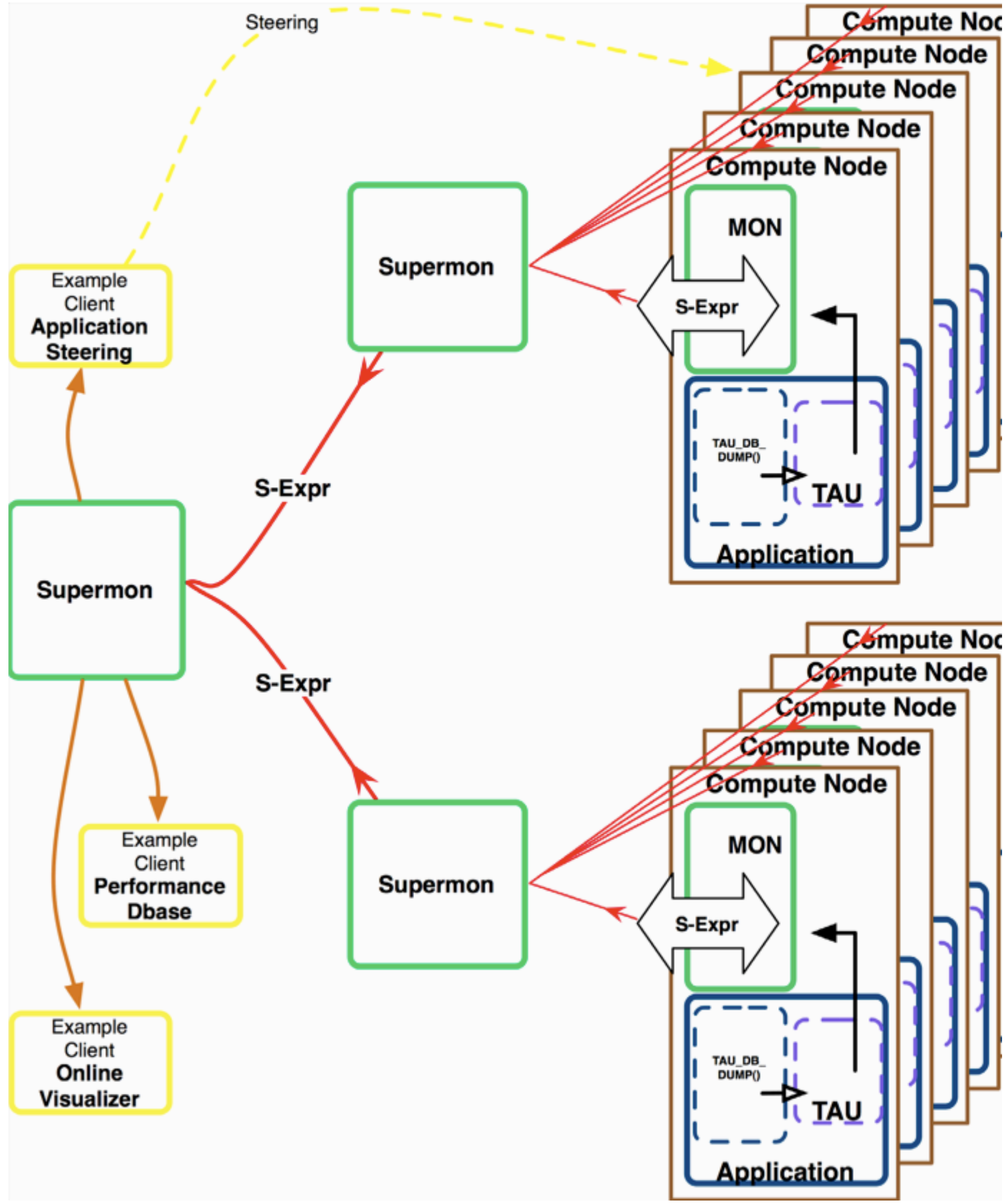
- Measurement and measured data transport de-coupled
  - Earlier, no such clear distinction in TAU
- Created abstraction to separate and hide transport
  - *TauOutput*
- Did not create a custom transport for TAU(as yet)
  - Use existing monitoring/transport capabilities
- TAUover: Supermon (Sottile and Minnich, LANL) and MRNET (Arnold and Miller, UWisc)
- A. Nataraj, M.Sottile, A. Morris, A. Malony, S. Shende "TAUoverSupermon: Low-overhead Online Parallel Performance Monitoring", Europar'07.

Rationale

- Moved away from NFS
- Separation of concerns
  - Scalability, portability, robustness
  - Addressed independent of TAU
- Re-use existing technologies where appropriate
- Multiple bindings
  - Use different solutions best suited to particular platform
- Implementation speed
  - Easy, fast to create adapter that binds to existing transport

Substrate Architecture - High-level

- Components
  - Front-End (FE)
  - Intermediate Nodes
  - Back-End (BE)
- NFS, Supermon, MRNet API
- Push-Pull model of data retrieval
- Figure shows ***ToS*** high-level view



Substrate Architecture - Back-End

- Application calls into TAU
  - Per-Iteration explicit call to output routine
  - Periodic calls using alarm
- TauOutput object invoked
  - Configuration specific: compile or runtime
  - One per thread
- TauOutput mimics subset of FS-style operations
  - Avoids changes to TAU code
  - If required rest of TAU can be made aware of output type
- Non-blocking *recv* for control
- Back-end pushes, Sink pulls