SC'14 Tutorial:
Hands-on Practical Hybrid Parallel Application Performance Engineering

Markus Geimer
Jülich Supercomputing Centre

Sameer Shende
University of Oregon

Bert Wesarg
Technische Universität Dresden

Brian Wylie
Jülich Supercomputing Centre

Agenda

Time	Topic	Presenter
08:30	Introduction to VI-HPS & parallel performance engineering	Wylie
09:15	VI-HPS Linux Live-ISO and MPI+OpenMP example code	Wylie / all
09:30	Instrumentation & measurement with **Score-P**	Wesarg
10:00	*Break*	
10:30	Profile examination with **CUBE**	Geimer
11:00	Configuration & customization of Score-P measurements	Geimer
11:30	Profile examination with **TAU** ParaProf	Shende
12:00	*Lunch*	
13:30	Automated trace analysis with **Scalasca**	Geimer
14:15	Interactive trace analysis with **Vampir**	Wesarg
15:00	*Break*	
15:30	Specialized Score-P measurements & analysis	Wesarg
16:00	Performance data management with **TAU** PerfExplorer	Shende
16:15	Finding typical parallel performance bottlenecks	Wesarg
16:45	Review & conclusion	Wylie
17:00	*Adjourn*	

Introduction to VI-HPS

Brian Wylie

Jülich Supercomputing Centre

Mission: Improve the quality and accelerate the development process of complex simulation codes running on highly-parallel computer systems

- Start-up funding (2006–2011) by Helmholtz Association of German Research Centres
- Activities
  - Development and integration of HPC programming tools
    - diagnose programming errors and optimization opportunities
  - Training & support to apply these tools
  - Academic workshops

HELMHOLTZ ASSOCIATION

http://www.vi-hps.org

VI-HPS partners (founders)

Forschungszentrum Jülich

- Jülich Supercomputing Centre

RWTH Aachen University

- Centre for Computing & Communication

Technische Universität Dresden

- Centre for Information Services & HPC

University of Tennessee (Knoxville)

- Innovative Computing Laboratory

VI-HPS partners (cont.)

Barcelona Supercomputing Center

- Centro Nacional de Supercomputación

German Research School

- Laboratory of Parallel Programming

Lawrence Livermore National Lab.

- Centre for Applied Scientific Computing

Technical University of Munich

- Chair for Computer Architecture

University of Oregon

- Performance Research Laboratory

University of Stuttgart

- HPC Centre

University of Versailles St-Quentin

- LRC ITACA

Allinea Software Ltd

Productivity tools

MUST

- MPI usage correctness checking

PAPI

- Interfacing to hardware performance counters

Periscope

- Automatic analysis via an on-line distributed search

Scalasca

- Large-scale parallel performance analysis

TAU

- Integrated parallel performance system

Vampir

- Interactive graphical trace visualization & analysis

Score-P

- Community instrumentation & measurement infrastructure

Productivity tools (cont.)

DDT/MAP/PR

- Parallel debugging & profiling

KCachegrind

- Callgraph-based cache analysis [x86 only]

MAQAO

- Assembly instrumentation & optimization [x86-64 only]

mpiP/mpiPview

- MPI profiling tool and analysis viewer

Open MPI

- Integrated memory checking

Open|Speedshop

- Integrated parallel performance analysis environment

Paraver/Dimemas/Extrae

- Event tracing and graphical trace visualization & analysis

Rubik

- Process mapping generation & optimization [BG only]

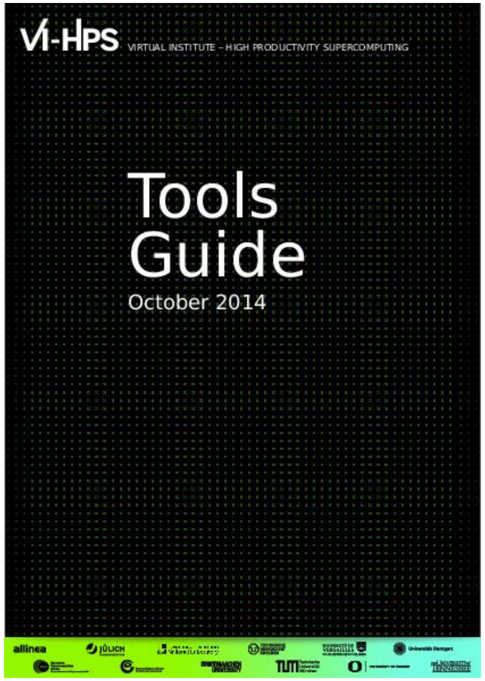
SIONlib/Spindle

- Optimized native parallel file I/O & library loading

STAT

- Stack trace analysis tools

For a brief overview of tools consult the VI-HPS Tools Guide:



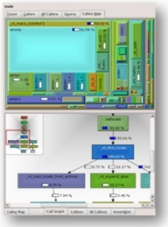
VI-HPS VIRTUAL INSTITUTE – HIGH PRODUCTIVITY SUPERCOMPUTING

Tools Guide

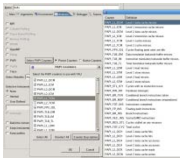
October 2014

Technologies and their integration

KCACHEGRIND

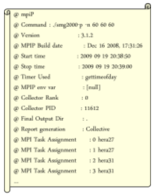


PAPI

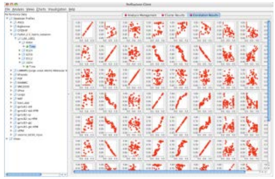


Hardware monitoring

LWM2 / MAP / MPIP / O|SS / MAQAO



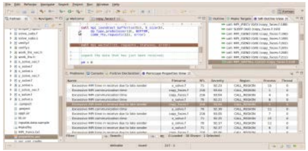
```
@ mpiP
@ Command : ./smg2000-p -n 60 60 60
@ Version                 : 3.1.2
@ MPIP Build date         : Dec 16 2008, 17:31:26
@ Start time              : 2009 09 19 20:38:50
@ Stop time               : 2009 09 19 20:39:00
@ Timer Used              : gettimeofday
@ MPIP env var            : [null]
@ Collector Rank          : 0
@ Collector PID           : 11612
@ Final Output Dir        : .
@ Report generation       : Collective
@ MPI Task Assignment     : 0 hera27
@ MPI Task Assignment     : 1 hera27
@ MPI Task Assignment     : 2 hera31
@ MPI Task Assignment     : 3 hera31
...
```



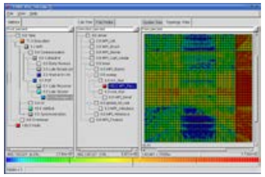
TAU

SCORE-P

Automatic profile & trace analysis



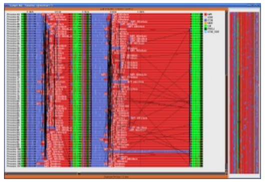
PERISCOPE



SCALASCA

Visual trace analysis

VAMPIR / PARAVER



Optimization

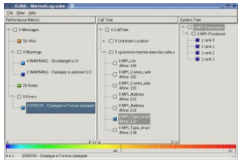
RUBIK / MAQAO

Execution

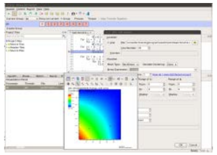
SYSMON / SPINDLE / SIONLIB / OPENMPI

Debugging, error & anomaly detection

MUST



DDT



STAT

Disclaimer

Tools will ***not*** automatically make you, your applications or computer systems more *productive*.

However, they can help you understand ***how*** your parallel code executes and ***when / where*** it's necessary to work on *correctness* and *performance* issues.

VI-HPS training & Tuning Workshops

- Goals
  - Give an overview of the programming tools suite
  - Explain the functionality of individual tools
  - Teach how to use the tools effectively
  - Offer hands-on experience and expert assistance using tools
  - Receive feedback from users to guide future development
- For best results, bring & analyze/tune your own code(s)!

- VI-HPS Hands-on Tutorial series
  - SC'08, ICCS'09, SC'09, Cluster'10, SC'10, SC'11, EuroMPI'12, XSEDE'13, SC'13, ***SC'14 (New Orleans)***
- VI-HPS Tuning Workshop series
  - 2008 (Aachen & Dresden), 2009 (Jülich & Bremen),
    2010 (Garching & Amsterdam/NL), 2011 (Stuttgart & Aachen),
    2012 (St-Quentin/F & Garching), 2013 (Saclay/F & Jülich)
    2014 (Barcelona/Spain, Kobe/Japan, Saclay/France, Edinburgh/UK)

Upcoming events

- 17th VI-HPS Tuning Workshop (23-27 February 2015)
  - Hosted by HLRS, Stuttgart, Germany
  - Using PRACE Tier-0 *Hornet* Cray XC40 system
  - VI-HPS and Cray tools to be presented
- Further events to be determined
  - (one-day) tutorials
    - With guided exercises usually using a Live-ISO
  - (multi-day) training workshops
    - With your own applications on actual HPC systems
- Check www.vi-hps.org/training for announced events
- Contact us if you might be interested in hosting an event

VI-HPS Linux Live DVD/ISO

- Bootable Linux installation on DVD (or USB memory stick)
- Includes everything needed to try out our parallel tools on an 64-bit x86-architecture notebook computer
  - VI-HPS tools: MUST, PAPI, Score-P, Periscope, Scalasca, TAU, Vampir*
  - Also: Eclipse/PTP, DDT*, TotalView*

    * time/capability-limited evaluation licences provided for commercial products
  - GCC (w/ OpenMP), OpenMPI
  - Manuals/User Guides
  - Tutorial exercises & examples
- Produced by U. Oregon PRL
  - Sameer Shende

PerfSuite
Tuning and Analysis Utilities
PAPI
scalasca
VAMPIR
Parallel Productivity Tools
Live DVD
Also includes: TotalView, DyninstAPI, PDT, Eclipse PTP, Berkeley UPC, ptoolsrte, Chapel, and much more...
Partners:
ParaTools, Inc.
University of Florida
University of Oregon
Totalview Technologies
RWTH Aachen University
HLRS / University of Stuttgart
Jülich Supercomputing Centre
Technische Universität Dresden
Technische Universität München
University of Wisconsin at Madison
Pittsburgh Supercomputing Center
University of Tennessee at Knoxville
National Center for Supercomputing Applications
November 2011
POINT VI-HPS
http://nic.uoregon.edu/point http://www.vi-hps.org

VI-HPS Linux Live ISO

- ISO image approximately 10GB
  - download latest version from website
  - http://www.vi-hps.org/training/live-iso/
  - optionally create bootable DVD or USB drive
- Boot directly from disk
  - enables hardware counter access and offers best performance, but no save/resume
- Boot within virtual machine (e.g., VirtualBox)
  - faster boot time and can save/resume state, but may not allow hardware counter access
- Boots into Linux environment for HPC
  - supports building and running provided MPI and/or OpenMP parallel application codes
  - and experimentation with VI-HPS (and third-party) tools

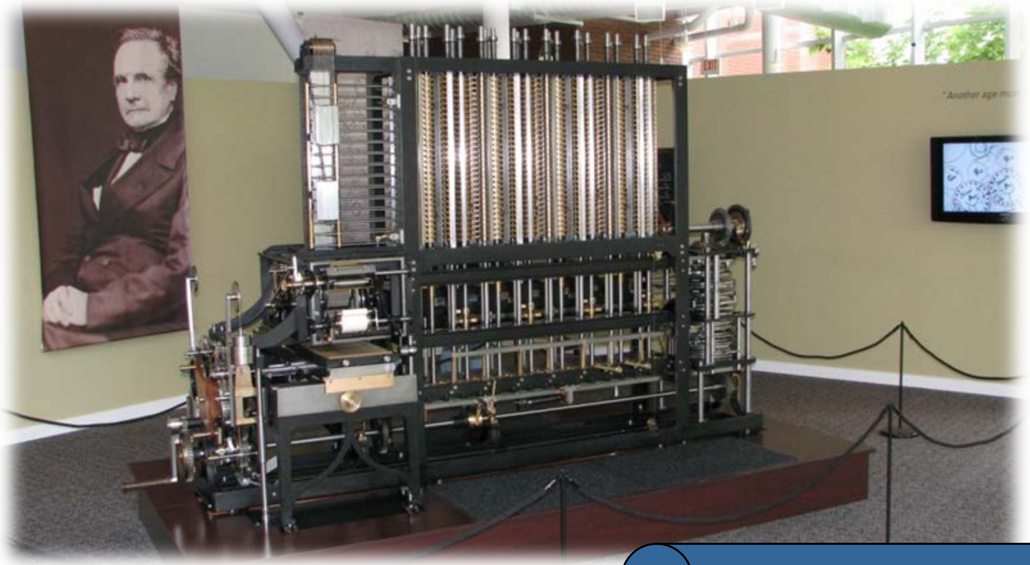
Introduction to Parallel Performance Engineering

Brian Wylie

Jülich Supercomputing Centre

(with content used with permission from tutorials by Bernd Mohr/JSC and Luiz DeRose/Cray)

Performance: an old problem



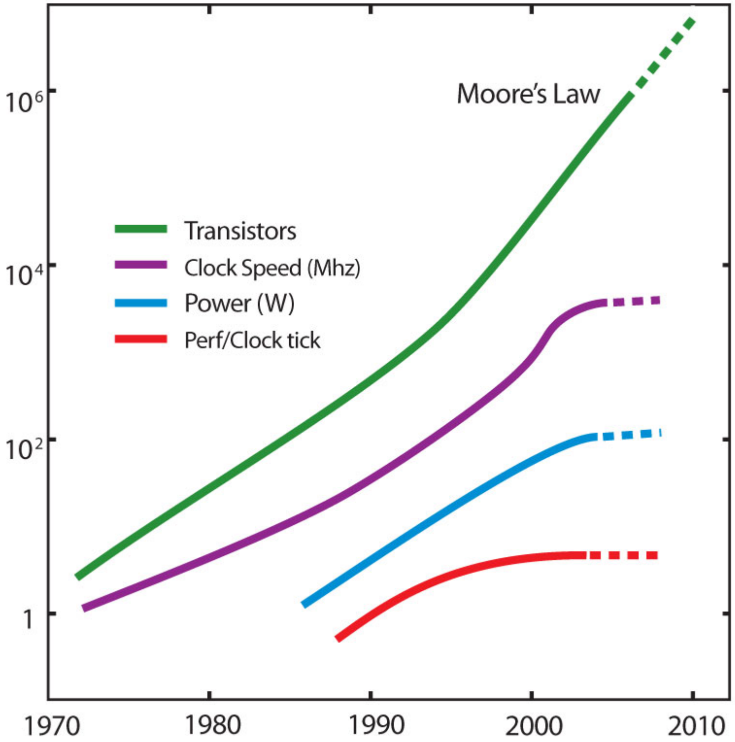
Difference Engine

> "The most constant difficulty in contriving the engine has arisen from the desire to reduce the time in which the calculations were executed to the shortest which is possible."
>
> Charles Babbage
> 1791 – 1871

Today: the “free lunch” is over

- Moore's law is still in charge, but
  - Clock rates no longer increase
  - Performance gains only through increased parallelism
- Optimizations of applications more difficult
  - Increasing application complexity
    - Multi-physics
    - Multi-scale
  - Increasing machine complexity
    - Hierarchical networks / memory
    - More CPUs / multi-core

☞Every doubling of scale reveals a new bottleneck!



Performance factors of parallel applications

- “Sequential” performance factors
  - Computation
    - ☞Choose right algorithm, use optimizing compiler
  - Cache and memory
    - ☞Tough! Only limited tool support, hope compiler gets it right
  - Input / output
    - ☞Often not given enough attention

- “Parallel” performance factors
  - Partitioning / decomposition
  - Communication (i.e., message passing)
  - Multithreading
  - Synchronization / locking
    - ☞More or less understood, good tool support

Tuning basics

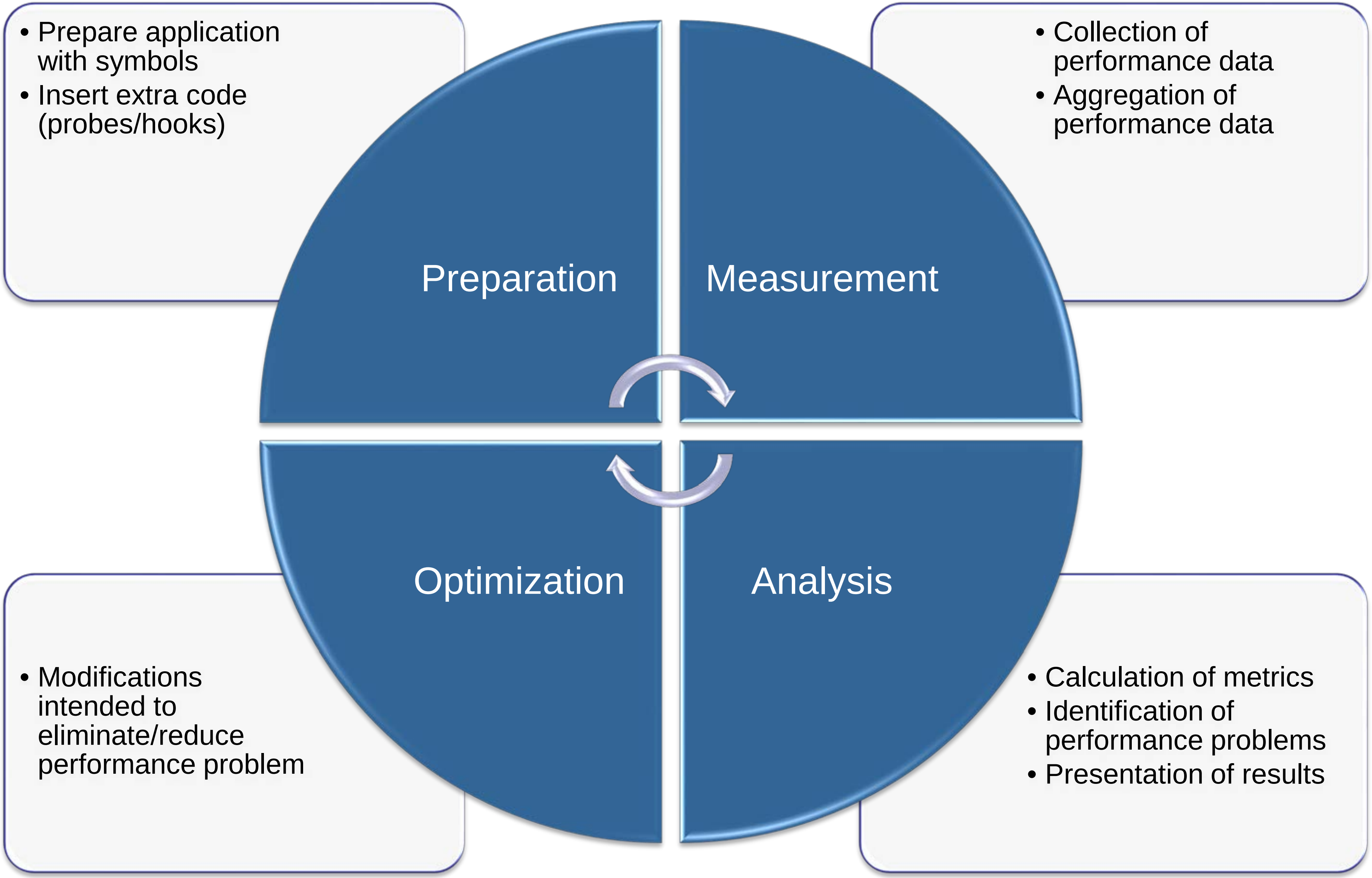
- Successful engineering is a combination of
  - The right algorithms and libraries
  - Compiler flags and directives
  - Thinking !!!
- Measurement is better than guessing
  - To determine performance bottlenecks
  - To compare alternatives
  - To validate tuning decisions and optimizations
    - ☞After each step!

However…

> “We should forget about small efficiencies, say 97% of the time: premature optimization is the root of all evil.”
>
> Charles A. R. Hoare

- It's easier to optimize a slow correct program than to debug a fast incorrect one
  - ☞ *Nobody cares how fast you can compute a wrong answer...*

Performance engineering workflow



- Prepare application with symbols
- Insert extra code (probes/hooks)

- Collection of performance data
- Aggregation of performance data

- Modifications intended to eliminate/reduce performance problem

- Calculation of metrics
- Identification of performance problems
- Presentation of results

The 80/20 rule

- Programs typically spend 80% of their time in 20% of the code
- Programmers typically spend 20% of their effort to get 80% of the total speedup possible for the application
  - ☞ *Know when to stop!*
- Don't optimize what does not matter
  - ☞ *Make the common case fast!*

> "If you optimize everything, you will always be unhappy."
>
> Donald E. Knuth

Metrics of performance

- What can be measured?
  - A **count** of how often an event occurs
    - E.g., the number of MPI point-to-point messages sent
  - The **duration** of some interval
    - E.g., the time spent these send calls
  - The **size** of some parameter
    - E.g., the number of bytes transmitted by these calls

- Derived metrics
  - E.g., rates / throughput
  - Needed for normalization

Example metrics

- Execution time
- Number of function calls
- CPI
  - CPU cycles per instruction
- FLOPS
  - Floating-point operations executed per second

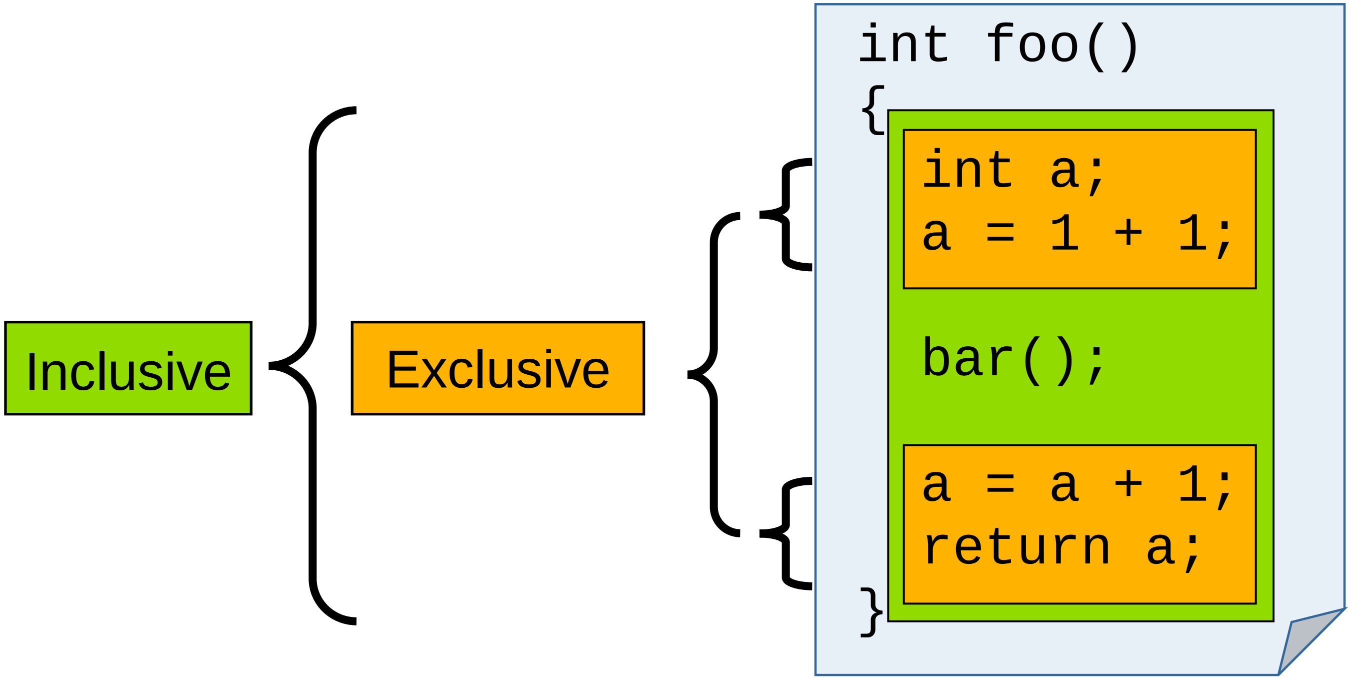
“math” Operations?
HW Operations?
HW Instructions?
32-/64-bit? …

Execution time

- Wall-clock time
  - Includes waiting time: I/O, memory, other system activities
  - In time-sharing environments also the time consumed by other applications
- CPU time
  - Time spent by the CPU to execute the application
  - Does not include time the program was context-switched out
    - Problem: Does not include inherent waiting time (e.g., I/O)
    - Problem: Portability? What is user, what is system time?
- Problem: Execution time is non-deterministic
  - Use mean or minimum of several runs

Inclusive vs. Exclusive values

- Inclusive
  - Information of all sub-elements aggregated into single value
- Exclusive
  - Information cannot be subdivided further



```
int foo()
{
  int a;
  a = 1 + 1;

  bar();

  a = a + 1;
  return a;
}
```

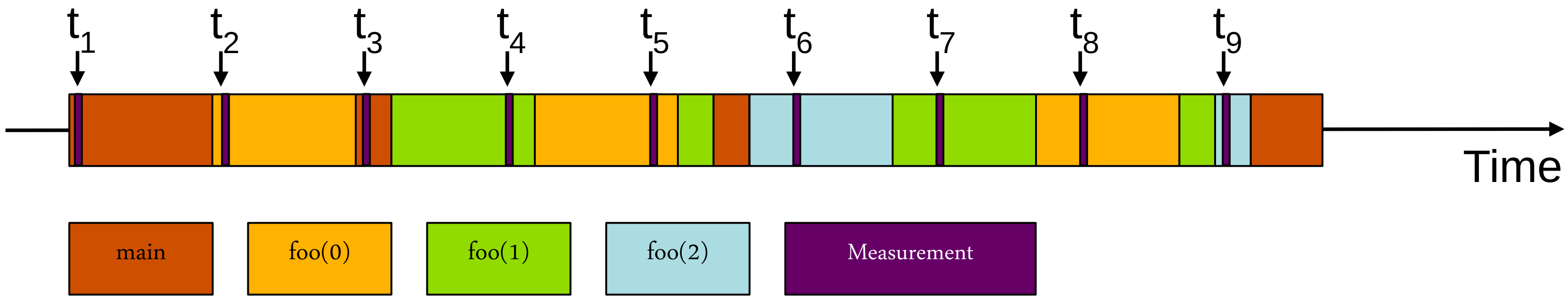
Classification of measurement techniques

- How are performance measurements triggered?
  - Sampling
  - Code instrumentation

- How is performance data recorded?
  - Profiling / Runtime summarization
  - Tracing

- How is performance data analyzed?
  - Online
  - Post mortem

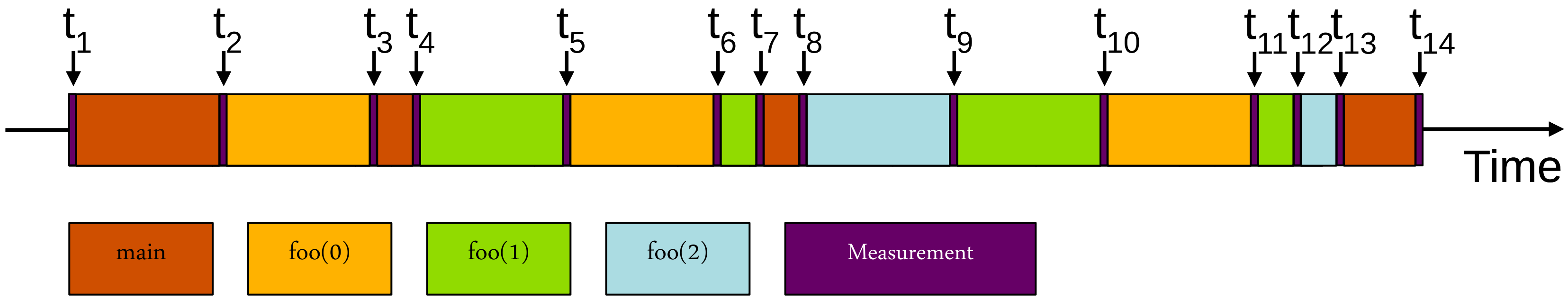
Sampling



```
int main()
{
  int i;

  for (i=0; i < 3; i++)
    foo(i);

  return 0;
}

void foo(int i)
{

  if (i > 0)
    foo(i – 1);

}
```

- Running program is periodically interrupted to take measurement
  - Timer interrupt, OS signal, or HWC overflow
  - Service routine examines return-address stack
  - Addresses are mapped to routines using symbol table information
- **Statistical** inference of program behavior
  - Not very detailed information on highly volatile metrics
  - Requires long-running applications
- Works with unmodified executables

Instrumentation



```
int main()
{
  int i;
  Enter(“main”);
  for (i=0; i < 3; i++)
    foo(i);
  Leave(“main”);
  return 0;
}

void foo(int i)
{
  Enter(“foo”);
  if (i > 0)
    foo(i – 1);
  Leave(“foo”);
}
```

- Measurement code is inserted such that every event of interest is captured **directly**
  - Can be done in various ways
- Advantage:
  - Much more detailed information
- Disadvantage:
  - Processing of source-code / executable necessary
  - Large relative overheads for small functions

Instrumentation techniques

- Static instrumentation
  - Program is instrumented prior to execution
- Dynamic instrumentation
  - Program is instrumented at runtime
- Code is inserted
  - Manually
  - Automatically
    - By a preprocessor / source-to-source translation tool
    - By a compiler
    - By linking against a pre-instrumented library / runtime system
    - By binary-rewrite / dynamic instrumentation tool

Critical issues

- Accuracy
  - Intrusion overhead
    - Measurement itself needs time and thus lowers performance
  - Perturbation
    - Measurement alters program behaviour
    - E.g., memory access pattern
  - Accuracy of timers & counters
- Granularity
  - How many measurements?
  - How much information / processing during each measurement?

☞ *Tradeoff: Accuracy vs. Expressiveness of data*

Classification of measurement techniques

- How are performance measurements triggered?
  - Sampling
  - Code instrumentation

- How is performance data recorded?
  - Profiling / Runtime summarization
  - Tracing

- How is performance data analyzed?
  - Online
  - Post mortem

Profiling / Runtime summarization

- Recording of aggregated information
  - Total, maximum, minimum, …
- For measurements
  - Time
  - Counts
    - Function calls
    - Bytes transferred
    - Hardware counters
- Over program and system entities
  - Functions, call sites, basic blocks, loops, …
  - Processes, threads

☞ *Profile = summarization of events over execution interval*

Types of profiles

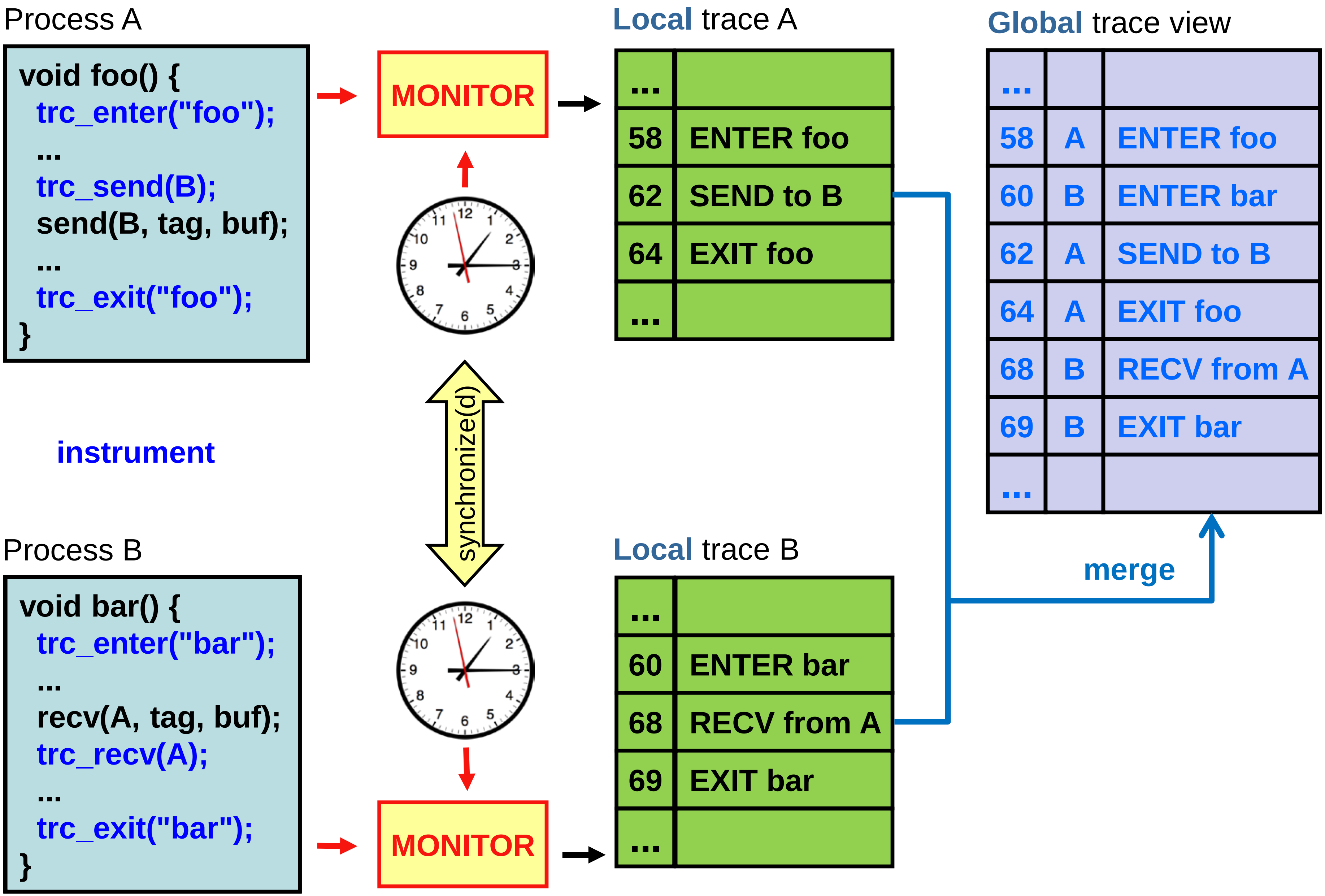
- Flat profile
    - Shows distribution of metrics per routine / instrumented region
    - Calling context is not taken into account
- Call-path profile
    - Shows distribution of metrics per executed call path
    - Sometimes only distinguished by partial calling context (e.g., two levels)
- Special-purpose profiles
    - Focus on specific aspects, e.g., MPI calls or OpenMP constructs
    - Comparing processes/threads

Tracing

- Recording detailed information about significant points (events) during execution of the program
  - Enter / leave of a region (function, loop, …)
  - Send / receive a message, …
- Save information in event record
  - Timestamp, location, event type
  - Plus event-specific information (e.g., communicator, sender / receiver, …)
- Abstract execution model on level of defined events

☞ *Event trace = Chronologically ordered sequence of event records*

Event tracing



Process A

```
void foo() {
  trc_enter("foo");
  ...
  trc_send(B);
  send(B, tag, buf);
  ...
  trc_exit("foo");
}
```

Process B

```
void bar() {
  trc_enter("bar");
  ...
  recv(A, tag, buf);
  trc_recv(A);
  ...
  trc_exit("bar");
}
```

Local trace A

...	
58	ENTER foo
62	SEND to B
64	EXIT foo
...	

Local trace B

...	
60	ENTER bar
68	RECV from A
69	EXIT bar
...	

Global trace view

...		
58	A	ENTER foo
60	B	ENTER bar
62	A	SEND to B
64	A	EXIT foo
68	B	RECV from A
69	B	EXIT bar
...		

Tracing Pros & Cons

- Tracing advantages
  - Event traces preserve the **temporal** and **spatial** relationships among individual events (☞ context)
  - Allows reconstruction of **dynamic** application behaviour on any required level of abstraction
  - Most general measurement technique
    - Profile data can be reconstructed from event traces
- Disadvantages
  - Traces can very quickly become extremely large
  - Writing events to file at runtime may causes perturbation

Classification of measurement techniques

- How are performance measurements triggered?
  - Sampling
  - Code instrumentation

- How is performance data recorded?
  - Profiling / Runtime summarization
  - Tracing

- How is performance data analyzed?
  - Online
  - Post mortem

Online analysis

- Performance data is processed during measurement run
  - Process-local profile aggregation
  - More sophisticated inter-process analysis using
    - “Piggyback” messages
    - Hierarchical network of analysis agents
- Inter-process analysis often involves application steering to interrupt and re-configure the measurement

Post-mortem analysis

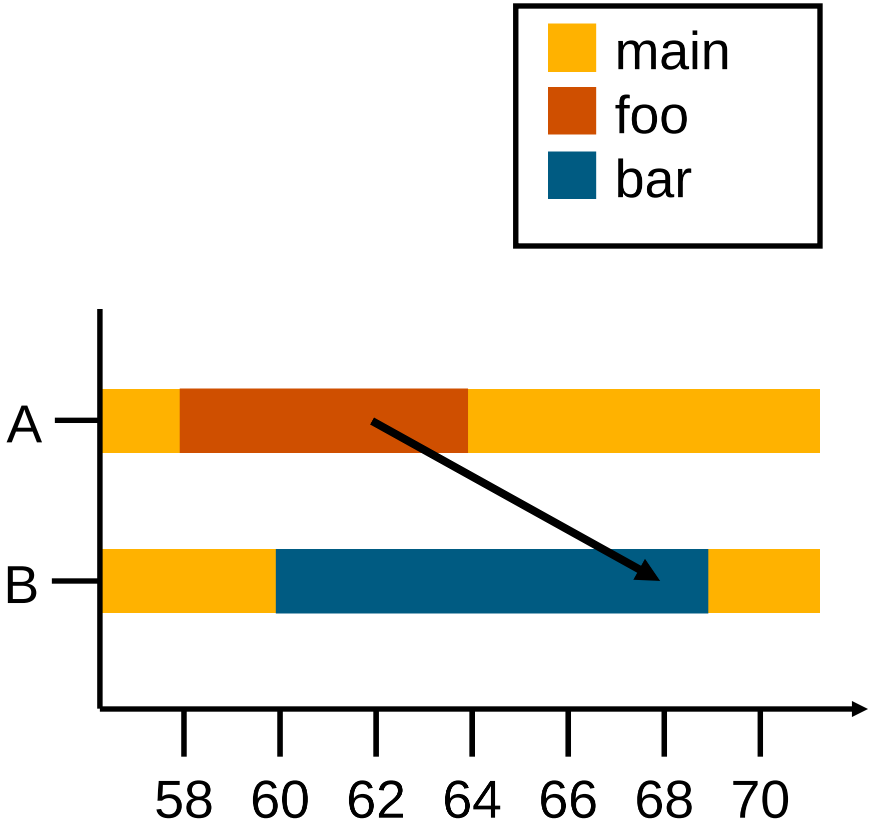
- Performance data is stored at end of measurement run
- Data analysis is performed afterwards
  - Automatic search for bottlenecks
  - Visual trace analysis
  - Calculation of statistics

Example: Time-line visualization

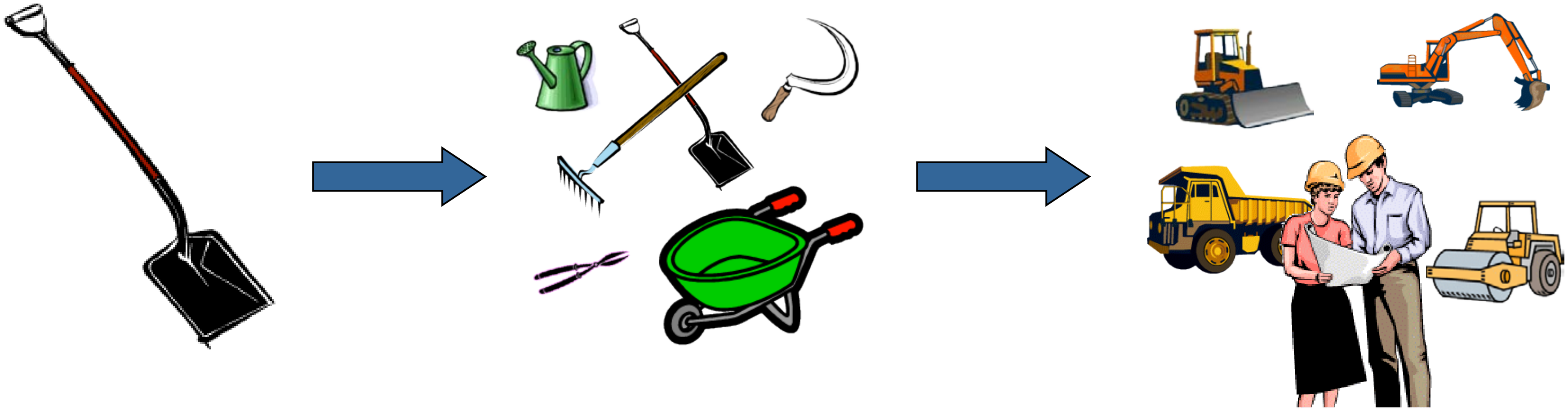
Global trace view

...		
58	A	ENTER foo
60	B	ENTER bar
62	A	SEND to B
64	A	EXIT foo
68	B	RECV from A
69	B	EXIT bar
...		

Post-Mortem → **Analysis**



No single solution is sufficient!



☞ *A combination of different methods, tools and techniques is typically needed!*

- Analysis
  - Statistics, visualization, automatic analysis, data mining, ...
- Measurement
  - Sampling / instrumentation, profiling / tracing, ...
- Instrumentation
  - Source code / binary, manual / automatic, ...

Typical performance analysis procedure

- Do I have a performance problem at all?
  - Time / speedup / scalability measurements
- What is the key bottleneck (computation / communication)?
  - MPI / OpenMP / flat profiling
- Where is the key bottleneck?
  - Call-path profiling, detailed basic block profiling
- Why is it there?
  - Hardware counter analysis, trace selected parts to keep trace size manageable
- Does the code have scalability problems?
  - Load imbalance analysis, compare profiles at various sizes function-by-function

Hands-on example code: NPB-MZ-MPI / BT (on Live-ISO/DVD)

VI-HPS Team

Performance Analysis Steps

1. Reference preparation for validation
2. Program instrumentation
3. Summary measurement collection
4. Summary analysis report examination
5. Summary experiment scoring
6. Summary measurement collection with filtering
7. Filtered summary analysis report examination
8. Event trace collection
9. Event trace examination & analysis

NPB-MZ-MPI suite

- The NAS Parallel Benchmark suite (MPI+OpenMP version)
  - Available from

    http://www.nas.nasa.gov/Software/NPB
  - 3 benchmarks in Fortran77
  - Configurable for various sizes & classes
- Move into the NPB3.3-MZ-MPI root directory

```
% cd Tutorial; ls
bin/     common/  jobscript/  Makefile  README.install   SP-MZ/
BT-MZ/   config/  LU-MZ/      README    README.tutorial  sys/
```

- Subdirectories contain source code for each benchmark
  - plus additional configuration and common code
- The provided distribution has already been configured for the tutorial, such that it's ready to “make” one or more of the benchmarks and install them into a (tool-specific) “bin” subdirectory

Building an NPB-MZ-MPI benchmark

- Type “make” for instructions

```
% make
  ===========================================
  =      NAS PARALLEL BENCHMARKS 3.3         =
  =      MPI+OpenMP Multi-Zone Versions      =
  =      F77                                 =
  ===========================================


  To make a NAS multi-zone benchmark type

         make <benchmark-name> CLASS=<class> NPROCS=<nprocs>

  where <benchmark-name> is “bt-mz”, “lu-mz”, or “sp-mz”
        <class>          is “S”, “W”, “A” through “F”
        <nprocs>         is number of processes

  [...]

 *******************************************
 * Custom build configuration is specified in config/make.def   *
 * Suggested tutorial exercise configuration for LiveISO/DVD:   *
 *       make bt-mz CLASS=W NPROCS=4                            *
 ****************************************************************
```

Hint: the recommended build configuration is available via

```
% make suite
```

Building an NPB-MZ-MPI benchmark

- Specify the benchmark configuration
  - benchmark name: **bt-mz**, lu-mz, sp-mz
  - the number of MPI processes: NPROCS=**4**
  - the benchmark class (S, W, A, B, C, D, E): CLASS=**W**

```
% make bt-mz CLASS=W NPROCS=4
cd BT-MZ; make CLASS=W NPROCS=4 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c
../sys/setparams bt-mz 4 W
mpif77 -c  -O3 -fopenmp bt.f
 [...]
cd ../common;  mpif77 -c  -O3 -fopenmp timers.f
mpif77 -O3 -fopenmp -o ../bin/bt-mz_W.4 \
bt.o initialize.o exact_solution.o exact_rhs.o set_constants.o \
adi.o rhs.o zone_setup.o x_solve.o y_solve.o exch_qbc.o \
solve_subs.o z_solve.o add.o error.o verify.o mpi_setup.o \
../common/print_results.o ../common/timers.o
Built executable ../bin/bt-mz_W.4
make: Leaving directory 'BT-MZ'
```

NPB-MZ-MPI / BT (Block Tridiagonal solver)

- What does it do?
  - Solves a discretized version of unsteady, compressible Navier-Stokes equations in three spatial dimensions
  - Performs 200 time-steps on a regular 3-dimensional grid
- Implemented in 20 or so Fortran77 source modules
- Uses MPI & OpenMP in combination
  - 4 processes with 4 threads each should be reasonable
    - don't expect to see speed-up when run on a laptop!
  - bt-mz_W.4 should run in around 5 to 12 seconds on a laptop
  - bt-mz_B.4 is more suitable for dedicated HPC compute nodes
    - Each class step takes around 10-15x longer

NPB-MZ-MPI / BT reference execution

- Launch as a hybrid MPI+OpenMP application

```
% cd bin
% OMP_NUM_THREADS=4  mpiexec -np 4 ./bt-mz_W.4
 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
 Number of zones:   4 x   4
 Iterations:  200   dt:   0.000800
 Number of active processes:      4
 Total number of threads:      16  (  4.0 threads/process)

 Time step    1
 Time step   20
 Time step   40
  [...]
 Time step  160
 Time step  180
 Time step  200
 Verification Successful

 BT-MZ Benchmark Completed.
 Time in seconds = 5.57
```

Alternatively execute script:

```
% sh ../jobscript/ISO/run.sh
```

Hint: save the benchmark output (or note the run time) to be able to refer to it later

Virtual Institute – High Productivity Supercomputing

VI-HPS

Score-P – A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir

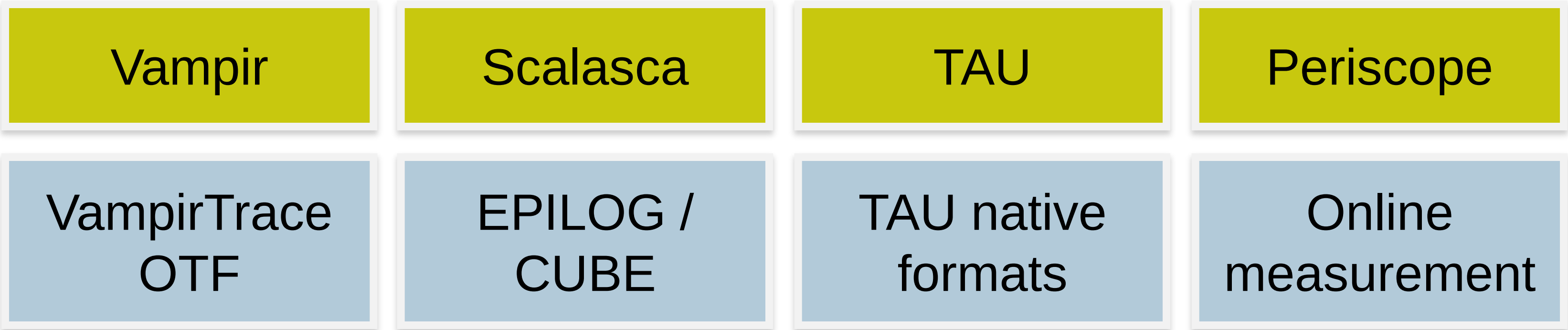
Markus Geimer[2], Bert Wesarg[1], Brian Wylie[2]

With contributions from
Andreas Knüpfer[1] and Christian Rössel[2]

[1]ZIH TU Dresden , [2]FZ Jülich

Fragmentation of Tools Landscape

- Several performance tools co-exist
- Separate measurement systems and output formats
- Complementary features and overlapping functionality
- Redundant effort for development and maintenance
- Limited or expensive interoperability
- Complications for user experience, support, training



Vampir	Scalasca	TAU	Periscope
VampirTrace OTF	EPILOG / CUBE	TAU native formats	Online measurement

SILC Project Idea

- Start a community effort for a common infrastructure
  - Score-P instrumentation and measurement system
  - Common data formats OTF2 and CUBE4
- Developer perspective:
  - Save manpower by sharing development resources
  - Invest in new analysis functionality and scalability
  - Save efforts for maintenance, testing, porting, support, training
- User perspective:
  - Single learning curve
  - Single installation, fewer version updates
  - Interoperability and data exchange
- SILC project funded by BMBF
- Close collaboration PRIMA project funded by DOE

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Partners

- Forschungszentrum Jülich, Germany
- German Research School for Simulation Sciences, Aachen, Germany
- Gesellschaft für numerische Simulation mbH Braunschweig, Germany
- RWTH Aachen, Germany
- Technische Universität Dresden, Germany
- Technische Universität München, Germany
- University of Oregon, Eugene, USA

Score-P Functionality

- Provide typical functionality for HPC performance tools
- Support all fundamental concepts of partner’s tools

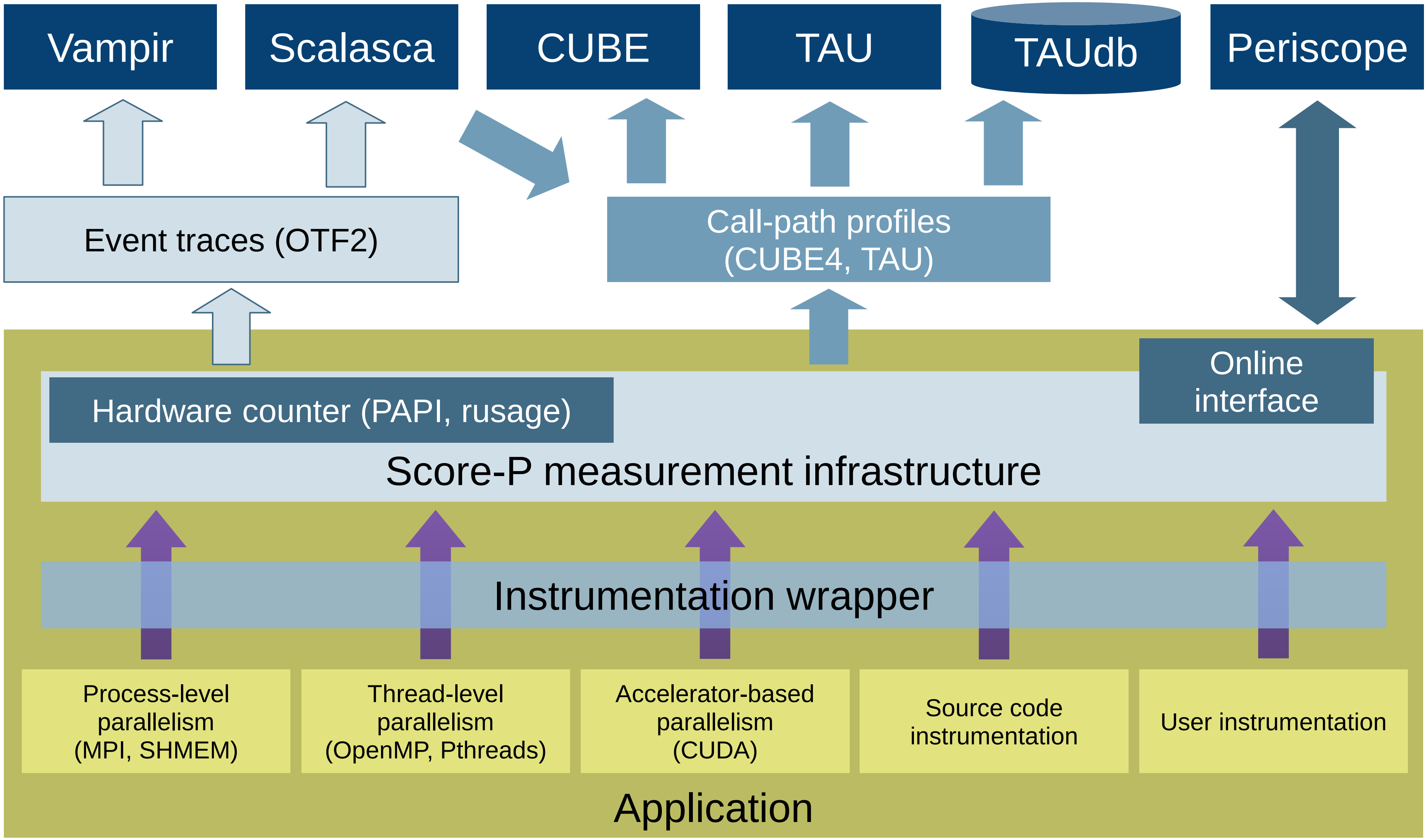
- Instrumentation (various methods)
- Flexible measurement without re-compilation:
  - Basic and advanced profile generation
  - Event trace recording
  - Online access to profiling data

- MPI/SHMEM, OpenMP/Pthreads, and hybrid parallelism (and serial)
- Enhanced functionality (OpenMP 3.0, CUDA, highly scalable I/O)

Design Goals

- Functional requirements
  - Generation of call-path profiles and event traces
  - Using direct instrumentation, later also sampling
  - Recording time, visits, communication data, hardware counters
  - Access and reconfiguration also at runtime
  - Support for MPI, OpenMP, basic CUDA, and all combinations
    - Later also OpenCL/OpenACC/…
- Non-functional requirements
  - Portability: all major HPC platforms
  - Scalability: petascale
  - Low measurement overhead
  - Easy and uniform installation through UNITE framework
  - Robustness
  - Open Source: New BSD License

Score-P Architecture


Vampir
Scalasca
CUBE
TAU
TAUdb
Periscope
Event traces (OTF2)
Call-path profiles (CUBE4, TAU)
Hardware counter (PAPI, rusage)
Online interface
Score-P measurement infrastructure
Instrumentation wrapper
Process-level parallelism (MPI, SHMEM)
Thread-level parallelism (OpenMP, Pthreads)
Accelerator-based parallelism (CUDA)
Source code instrumentation
User instrumentation
Application

Future Features and Management

- Scalability to maximum available CPU core count
- Support for OpenCL, OpenACC, Intel MIC
- Support for sampling, binary instrumentation
- Support for new programming models, e.g., PGAS
- Support for new architectures

- Ensure a single official release version at all times which will always work with the tools
- Allow experimental versions for new features or research

- Commitment to joint long-term cooperation

Score-P hands-on: NPB-MZ-MPI / BT

Performance Analysis Steps

1. Reference preparation for validation
2. Program instrumentation
3. Summary measurement collection
4. Summary analysis report examination
5. Summary experiment scoring
6. Summary measurement collection with filtering
7. Filtered summary analysis report examination
8. Event trace collection
9. Event trace examination & analysis

NPB-MZ-MPI / BT Instrumentation

- Change back to directory containing NPB BT-MZ

```
% cd ..
```

- Edit config/make.def to adjust build configuration
  - Modify specification of compiler/linker: MPIF77

```
...
#---------------------------------------------------------------------
# The Fortran compiler used for MPI programs
#---------------------------------------------------------------------
#MPIF77 = mpif77

# Alternative variants to perform instrumentation
...
MPIF77 = scorep mpif77

# This links MPI Fortran programs; usually the same as ${MPIF77}
FLINK   = $(MPIF77)
...
```

Uncomment the Score-P compiler wrapper specification

NPB-MZ-MPI / BT Instrumented Build

- Return to root directory and clean-up

```
% make clean
```

- Re-build executable using Score-P instrumenter

```
% make bt-mz CLASS=W NPROCS=4
cd BT-MZ; make CLASS=W NPROCS=4 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c -lm
../sys/setparams bt-mz 4 W
scorep mpif77 -c  -O3 -fopenmp bt.f
 [...]
cd ../common;  scorep mpif77 -c  -O3 -fopenmp timers.f
scorep mpif77 -O3 -fopenmp -o ../bin.scorep/bt-mz_W.4 \
bt.o initialize.o exact_solution.o exact_rhs.o set_constants.o \
adi.o rhs.o zone_setup.o x_solve.o y_solve.o exch_qbc.o \
solve_subs.o z_solve.o add.o error.o verify.o mpi_setup.o \
../common/print_results.o ../common/timers.o
Built executable ../bin.scorep/bt-mz_W.4
make: Leaving directory 'BT-MZ'
```

Performance Analysis Steps

1. Reference preparation for validation
2. Program instrumentation
3. Summary measurement collection
4. Summary analysis report examination
5. Summary experiment scoring
6. Summary measurement collection with filtering
7. Filtered summary analysis report examination
8. Event trace collection
9. Event trace examination & analysis

Measurement Configuration: scorep-info

- Score-P measurements are configured via environment variables:

```
% scorep-info config-vars --full
SCOREP_ENABLE_PROFILING
  Description: Enable profiling
 [...]
SCOREP_ENABLE_TRACING
  Description: Enable tracing
 [...]
SCOREP_TOTAL_MEMORY
  Description: Total memory in bytes for the measurement system
 [...]
SCOREP_EXPERIMENT_DIRECTORY
  Description: Name of the experiment directory
 [...]
SCOREP_FILTERING_FILE
  Description: A file name which contain the filter rules
 [...]
SCOREP_METRIC_PAPI
  Description: PAPI metric names to measure
 [...]
SCOREP_METRIC_RUSAGE
  Description: Resource usage metric names to measure
 [... More configuration variables ...]
```

Summary Measurement Collection

- Change to the directory containing the new executable adjust configuration and run application

```
% cd bin.scorep
% export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_W_4x4_sum
% OMP_NUM_THREADS=4  mpiexec -np 4 ./bt-mz_W.4
 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
 Number of zones:   4 x    4
 Iterations:  200   dt:   0.000800
 Number of active processes:     4
 Use the default load factors with threads
 Total number of threads:     16  (  4.0 threads/process)
 Use the default load factors with threads

 Time step    1
 Time step   20
  [...]
 Time step  180
 Time step  200
 Verification Successful

 BT-MZ Benchmark Completed.
 Time in seconds = 54.39
```

BT-MZ Summary Analysis Report Examination

- Creates experiment directory ./scorep_bt-mz_W_4x4_sum containing
  - a record of the measurement configuration (scorep.cfg)
  - the analysis report that was collated after measurement (profile.cubex)

```
% ls
...  scorep_bt-mz_W_4x4_sum
% ls scorep_bt-mz_W_4x4_sum
profile.cubex  scorep.cfg
```

- Interactive exploration with CUBE / ParaProf

```
% cube scorep_bt-mz_W_4x4_sum/profile.cubex

              [CUBE GUI showing summary analysis report]

% paraprof scorep_bt-mz_W_4x4_sum/profile.cubex

          [TAU ParaProf GUI showing summary analysis report]
```

Analysis report examination with CUBE

Markus Geimer

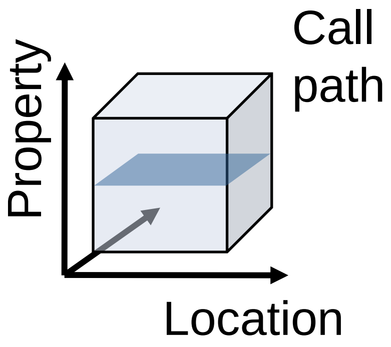
Jülich Supercomputing Centre

CUBE

- Parallel program analysis report exploration tools
  - Libraries for XML report reading & writing
  - Algebra utilities for report processing
  - GUI for interactive analysis exploration
    - requires Qt4
- Originally developed as part of Scalasca toolset
- Now available as a separate component
  - Can be installed independently of Score-P, e.g., on laptop or desktop
  - Latest release: CUBE 4.2.3 (June 2014)

Analysis presentation and exploration

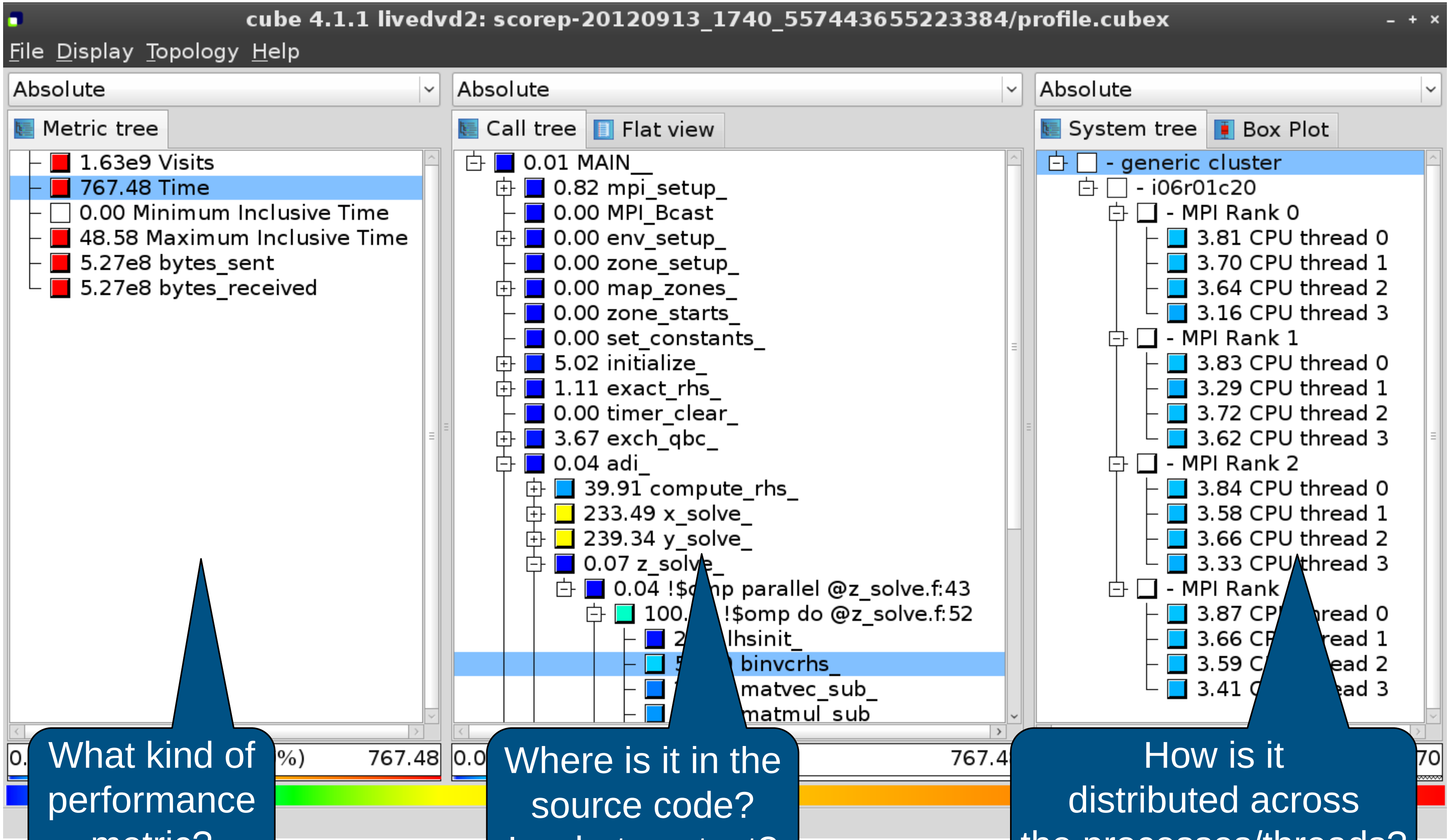
- Representation of values (severity matrix) on three hierarchical axes
  - Performance property (metric)
  - Call path (program location)
  - System location (process/thread)



- Three coupled tree browsers

- CUBE displays severities
  - As value: for precise comparison
  - As colour: for easy identification of hotspots
  - Inclusive value when closed & exclusive value when expanded
  - Customizable via display modes

Analysis presentation



cube 4.1.1 livedvd2: scorep-20120913_1740_557443655223384/profile.cubex

```
1.63e9 Visits
767.48 Time
0.00 Minimum Inclusive Time
48.58 Maximum Inclusive Time
5.27e8 bytes_sent
5.27e8 bytes_received
```

```
0.01 MAIN__
  0.82 mpi_setup_
  0.00 MPI_Bcast
  0.00 env_setup_
  0.00 zone_setup_
  0.00 map_zones_
  0.00 zone_starts_
  0.00 set_constants_
  5.02 initialize_
  1.11 exact_rhs_
  0.00 timer_clear_
  3.67 exch_qbc_
  0.04 adi_
    39.91 compute_rhs_
    233.49 x_solve_
    239.34 y_solve_
    0.07 z_solve_
      0.04 !$omp parallel @z_solve.f:43
        100. !$omp do @z_solve.f:52
          2 lhsinit_
          5 binvcrhs_
          matvec_sub_
          matmul_sub
```

```
- generic cluster
  - i06r01c20
    - MPI Rank 0
      3.81 CPU thread 0
      3.70 CPU thread 1
      3.64 CPU thread 2
      3.16 CPU thread 3
    - MPI Rank 1
      3.83 CPU thread 0
      3.29 CPU thread 1
      3.72 CPU thread 2
      3.62 CPU thread 3
    - MPI Rank 2
      3.84 CPU thread 0
      3.58 CPU thread 1
      3.66 CPU thread 2
      3.33 CPU thread 3
    - MPI Rank
      3.87 CP read 0
      3.66 C read 1
      3.59 C ead 2
      3.41 C ad 3
```

0. %) 767.48 0.0 767.4 70

What kind of performance metric?

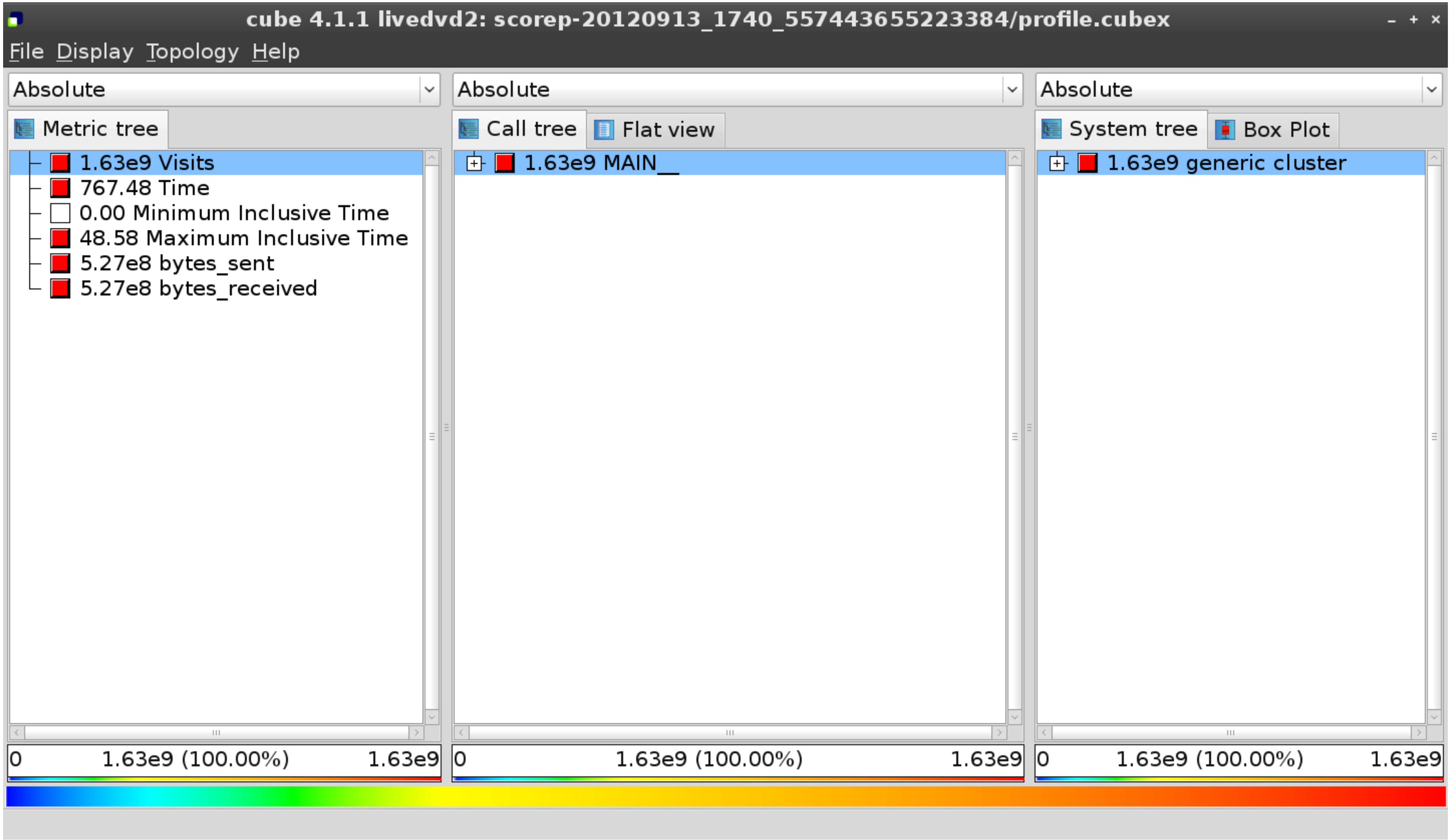
Where is it in the source code? In what context?

How is it distributed across the processes/threads?

Performance Analysis Steps

1. Reference preparation for validation
2. Program instrumentation
3. Summary measurement collection
4. Summary analysis report examination
5. Summary experiment scoring
6. Summary measurement collection with filtering
7. Filtered summary analysis report examination
8. Event trace collection
9. Event trace examination & analysis

Analysis report exploration (opening view)



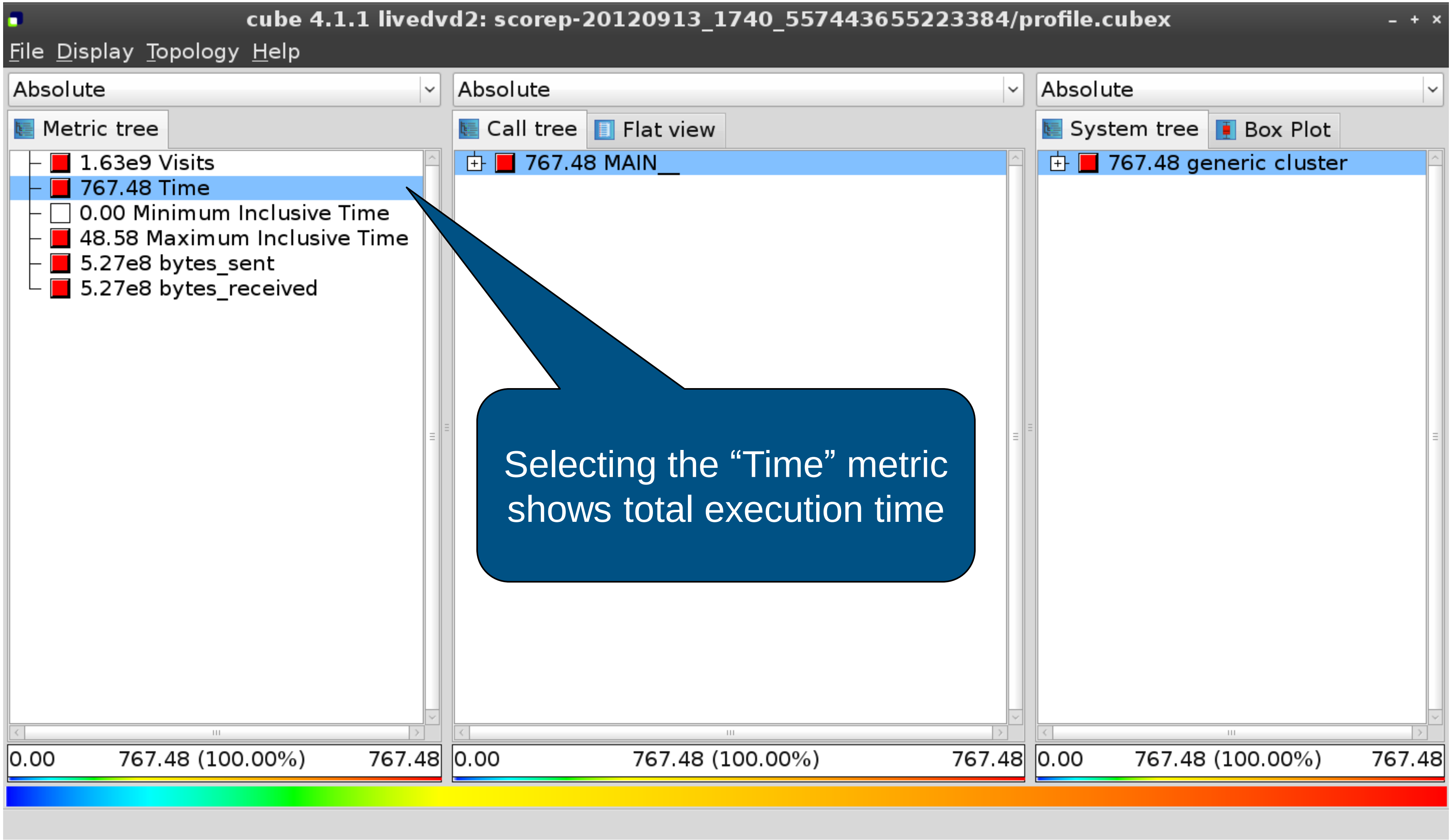
cube 4.1.1 livedvd2: scorep-20120913_1740_557443655223384/profile.cubex

- 🟥 1.63e9 Visits
- 🟥 767.48 Time
- □ 0.00 Minimum Inclusive Time
- 🟥 48.58 Maximum Inclusive Time
- 🟥 5.27e8 bytes_sent
- 🟥 5.27e8 bytes_received

⊞ 🟥 1.63e9 MAIN__

⊞ 🟥 1.63e9 generic cluster

Metric selection



cube 4.1.1 livedvd2: scorep-20120913_1740_557443655223384/profile.cubex

🟥 1.63e9 Visits
🟥 767.48 Time
□ 0.00 Minimum Inclusive Time
🟥 48.58 Maximum Inclusive Time
🟥 5.27e8 bytes_sent
🟥 5.27e8 bytes_received

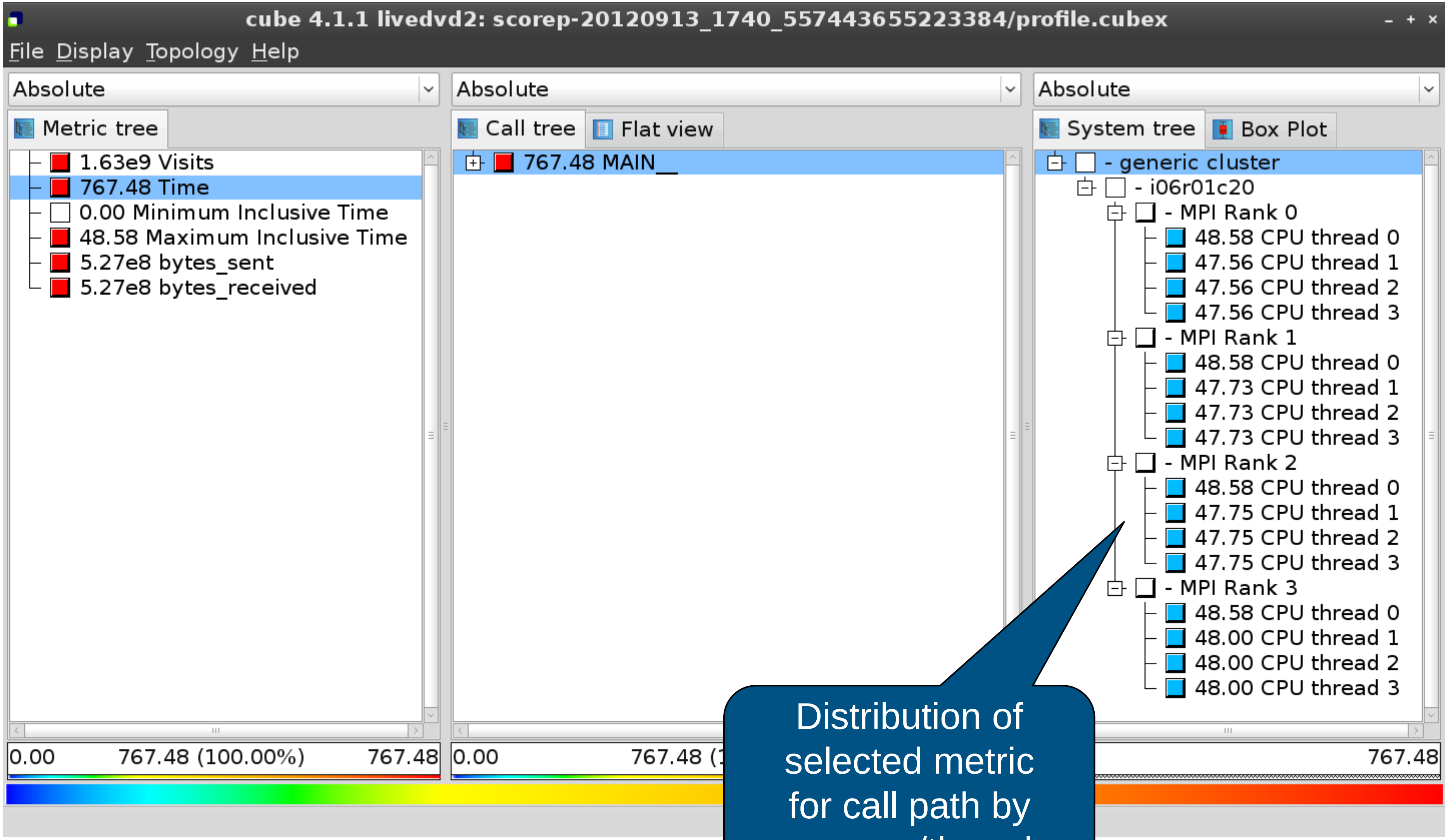
⊞ 🟥 767.48 MAIN__

⊞ 🟥 767.48 generic cluster

0.00	767.48 (100.00%)	767.48
0.00	767.48 (100.00%)	767.48
0.00	767.48 (100.00%)	767.48

Selecting the “Time” metric shows total execution time

Expanding the system tree



cube 4.1.1 livedvd2: scorep-20120913_1740_557443655223384/profile.cubex

Metric tree:

- 🟥 1.63e9 Visits
- 🟥 767.48 Time
- □ 0.00 Minimum Inclusive Time
- 🟥 48.58 Maximum Inclusive Time
- 🟥 5.27e8 bytes_sent
- 🟥 5.27e8 bytes_received

Call tree:

- ⊞ 🟥 767.48 MAIN__

System tree:

- □ - generic cluster
  - □ - i06r01c20
    - □ - MPI Rank 0
      - 🟦 48.58 CPU thread 0
      - 🟦 47.56 CPU thread 1
      - 🟦 47.56 CPU thread 2
      - 🟦 47.56 CPU thread 3
    - □ - MPI Rank 1
      - 🟦 48.58 CPU thread 0
      - 🟦 47.73 CPU thread 1
      - 🟦 47.73 CPU thread 2
      - 🟦 47.73 CPU thread 3
    - □ - MPI Rank 2
      - 🟦 48.58 CPU thread 0
      - 🟦 47.75 CPU thread 1
      - 🟦 47.75 CPU thread 2
      - 🟦 47.75 CPU thread 3
    - □ - MPI Rank 3
      - 🟦 48.58 CPU thread 0
      - 🟦 48.00 CPU thread 1
      - 🟦 48.00 CPU thread 2
      - 🟦 48.00 CPU thread 3

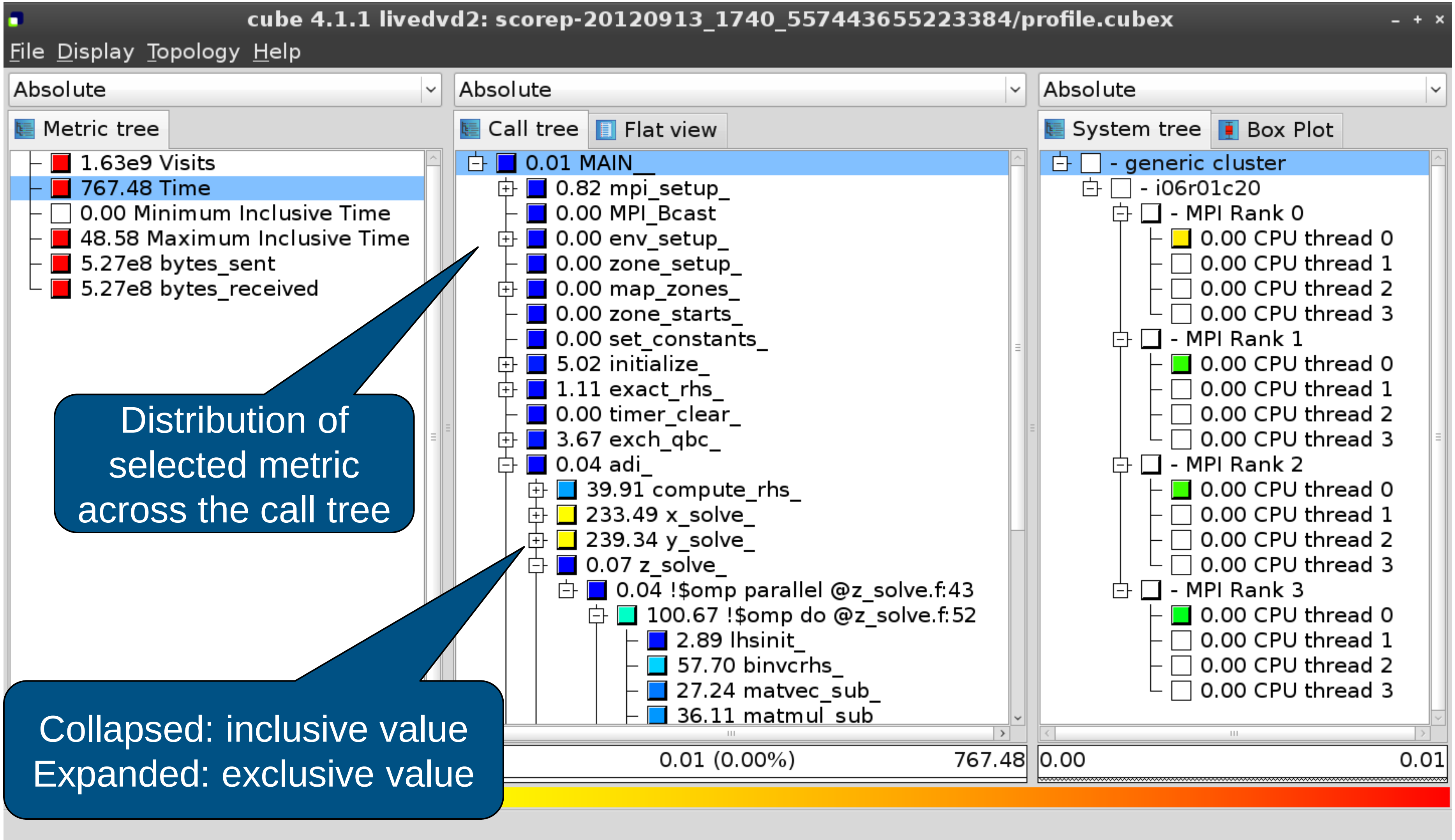
0.00 767.48 (100.00%) 767.48

0.00 767.48 (1

767.48

Distribution of selected metric for call path by process/thread

Expanding the call tree



cube 4.1.1 livedvd2: scorep-20120913_1740_557443655223384/profile.cubex

Metric tree:

- 1.63e9 Visits
- 767.48 Time
- 0.00 Minimum Inclusive Time
- 48.58 Maximum Inclusive Time
- 5.27e8 bytes_sent
- 5.27e8 bytes_received

Call tree:

```
0.01 MAIN__
  0.82 mpi_setup_
  0.00 MPI_Bcast
  0.00 env_setup_
  0.00 zone_setup_
  0.00 map_zones_
  0.00 zone_starts_
  0.00 set_constants_
  5.02 initialize_
  1.11 exact_rhs_
  0.00 timer_clear_
  3.67 exch_qbc_
  0.04 adi_
    39.91 compute_rhs_
    233.49 x_solve_
    239.34 y_solve_
    0.07 z_solve_
      0.04 !$omp parallel @z_solve.f:43
        100.67 !$omp do @z_solve.f:52
          2.89 lhsinit_
          57.70 binvcrhs_
          27.24 matvec_sub_
          36.11 matmul_sub
```

System tree:

```
- generic cluster
  - i06r01c20
    - MPI Rank 0
      0.00 CPU thread 0
      0.00 CPU thread 1
      0.00 CPU thread 2
      0.00 CPU thread 3
    - MPI Rank 1
      0.00 CPU thread 0
      0.00 CPU thread 1
      0.00 CPU thread 2
      0.00 CPU thread 3
    - MPI Rank 2
      0.00 CPU thread 0
      0.00 CPU thread 1
      0.00 CPU thread 2
      0.00 CPU thread 3
    - MPI Rank 3
      0.00 CPU thread 0
      0.00 CPU thread 1
      0.00 CPU thread 2
      0.00 CPU thread 3
```

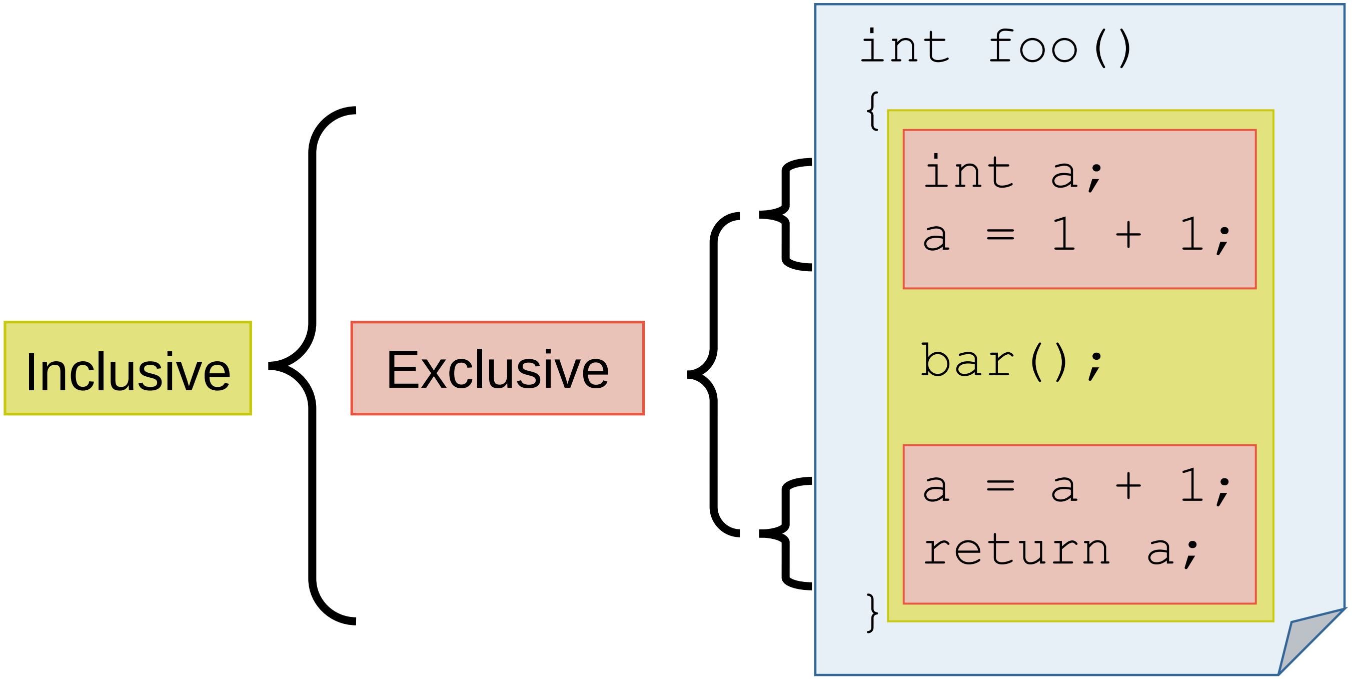
0.01 (0.00%) 767.48 | 0.00 0.01

Distribution of selected metric across the call tree

Collapsed: inclusive value
Expanded: exclusive value

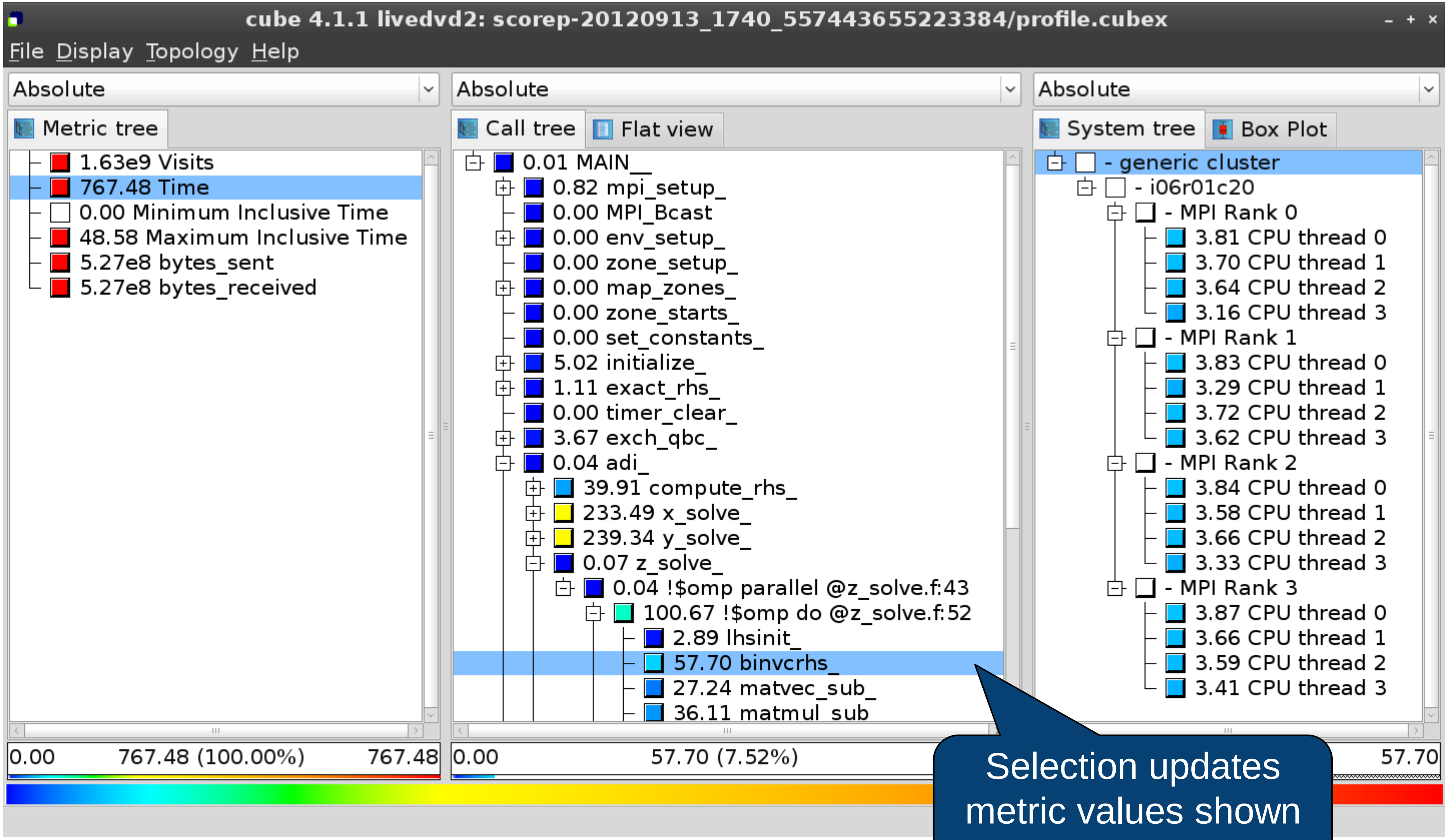
Inclusive vs. Exclusive values

- Inclusive
  - Information of all sub-elements aggregated into single value
- Exclusive
  - Information cannot be subdivided further



```
int foo()
{
  int a;
  a = 1 + 1;

  bar();

  a = a + 1;
  return a;
}
```

Selecting a call path



cube 4.1.1 livedvd2: scorep-20120913_1740_557443655223384/profile.cubex

File Display Topology Help

Absolute | Absolute | Absolute

Metric tree

```
🟥 1.63e9 Visits
🟥 767.48 Time
☐ 0.00 Minimum Inclusive Time
🟥 48.58 Maximum Inclusive Time
🟥 5.27e8 bytes_sent
🟥 5.27e8 bytes_received
```

Call tree | Flat view

```
⊟ 🟦 0.01 MAIN__
    ⊞ 🟦 0.82 mpi_setup_
      🟦 0.00 MPI_Bcast
    ⊞ 🟦 0.00 env_setup_
      🟦 0.00 zone_setup_
    ⊞ 🟦 0.00 map_zones_
      🟦 0.00 zone_starts_
      🟦 0.00 set_constants_
    ⊞ 🟦 5.02 initialize_
    ⊞ 🟦 1.11 exact_rhs_
      🟦 0.00 timer_clear_
    ⊞ 🟦 3.67 exch_qbc_
    ⊟ 🟦 0.04 adi_
        ⊞ 🟦 39.91 compute_rhs_
        ⊞ 🟨 233.49 x_solve_
        ⊞ 🟨 239.34 y_solve_
        ⊟ 🟦 0.07 z_solve_
            ⊟ 🟦 0.04 !$omp parallel @z_solve.f:43
                ⊟ 🟩 100.67 !$omp do @z_solve.f:52
                      🟦 2.89 lhsinit_
                      🟦 57.70 binvcrhs_
                      🟦 27.24 matvec_sub_
                      🟦 36.11 matmul_sub_
```

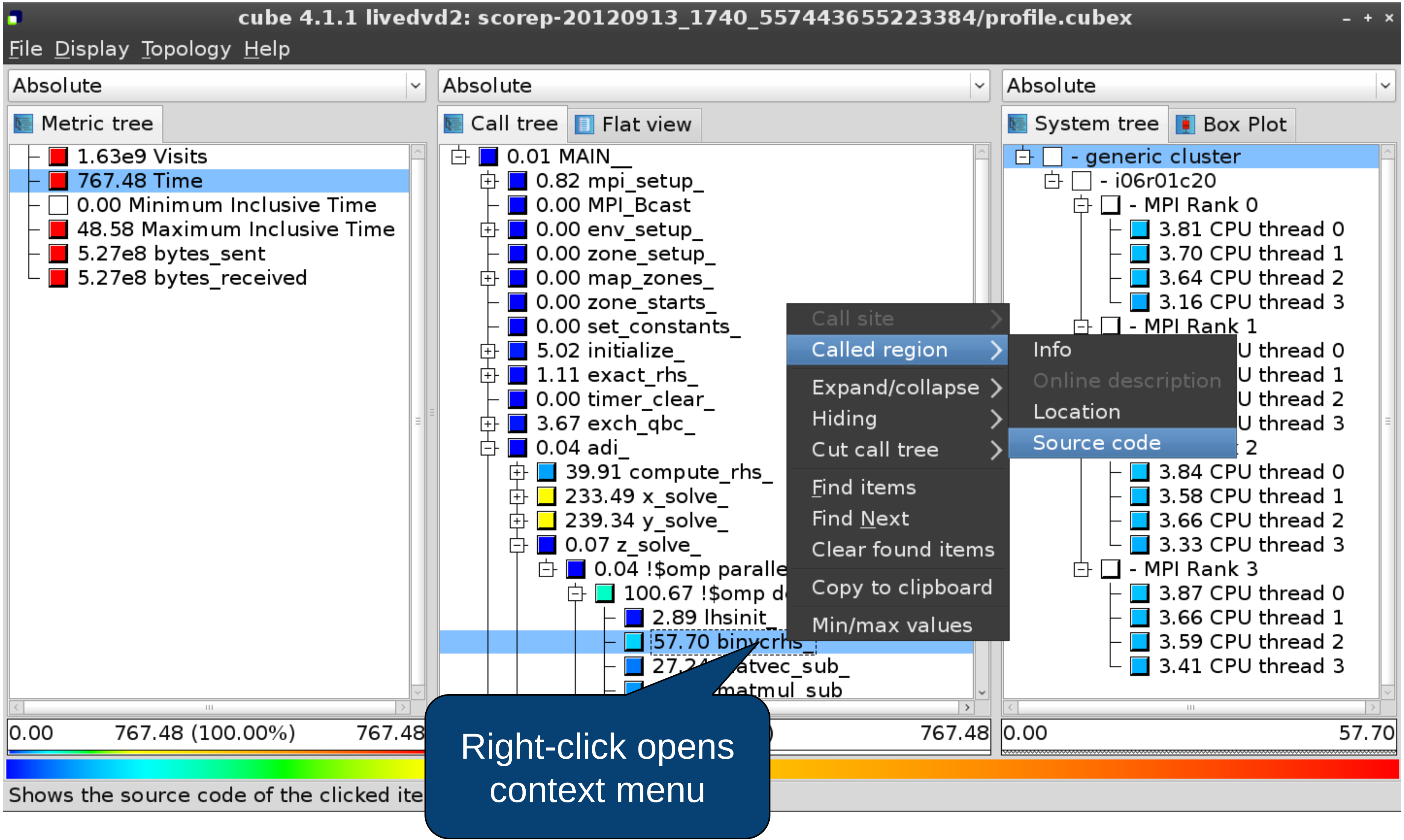
System tree | Box Plot

```
⊟ ☐ - generic cluster
    ⊟ ☐ - i06r01c20
        ⊟ ☐ - MPI Rank 0
              🟦 3.81 CPU thread 0
              🟦 3.70 CPU thread 1
              🟦 3.64 CPU thread 2
              🟦 3.16 CPU thread 3
        ⊟ ☐ - MPI Rank 1
              🟦 3.83 CPU thread 0
              🟦 3.29 CPU thread 1
              🟦 3.72 CPU thread 2
              🟦 3.62 CPU thread 3
        ⊟ ☐ - MPI Rank 2
              🟦 3.84 CPU thread 0
              🟦 3.58 CPU thread 1
              🟦 3.66 CPU thread 2
              🟦 3.33 CPU thread 3
        ⊟ ☐ - MPI Rank 3
              🟦 3.87 CPU thread 0
              🟦 3.66 CPU thread 1
              🟦 3.59 CPU thread 2
              🟦 3.41 CPU thread 3
```

0.00 767.48 (100.00%) 767.48 | 0.00 57.70 (7.52%) | 57.70

Selection updates metric values shown in columns to right

Source-code view via context menu



```
1.63e9 Visits
767.48 Time
0.00 Minimum Inclusive Time
48.58 Maximum Inclusive Time
5.27e8 bytes_sent
5.27e8 bytes_received
```

```
0.01 MAIN__
  0.82 mpi_setup_
  0.00 MPI_Bcast
  0.00 env_setup_
  0.00 zone_setup_
  0.00 map_zones_
  0.00 zone_starts_
  0.00 set_constants_
  5.02 initialize_
  1.11 exact_rhs_
  0.00 timer_clear_
  3.67 exch_qbc_
  0.04 adi_
    39.91 compute_rhs_
    233.49 x_solve_
    239.34 y_solve_
    0.07 z_solve_
      0.04 !$omp paralle
        100.67 !$omp d
          2.89 lhsinit_
          57.70 binvcrhs_
          27.2 atvec_sub_
               matmul_sub
```

```
- generic cluster
  - i06r01c20
    - MPI Rank 0
      3.81 CPU thread 0
      3.70 CPU thread 1
      3.64 CPU thread 2
      3.16 CPU thread 3
    - MPI Rank 1
      U thread 0
      U thread 1
      U thread 2
      U thread 3
       2
      3.84 CPU thread 0
      3.58 CPU thread 1
      3.66 CPU thread 2
      3.33 CPU thread 3
    - MPI Rank 3
      3.87 CPU thread 0
      3.66 CPU thread 1
      3.59 CPU thread 2
      3.41 CPU thread 3
```

0.00 767.48 (100.00%) 767.48

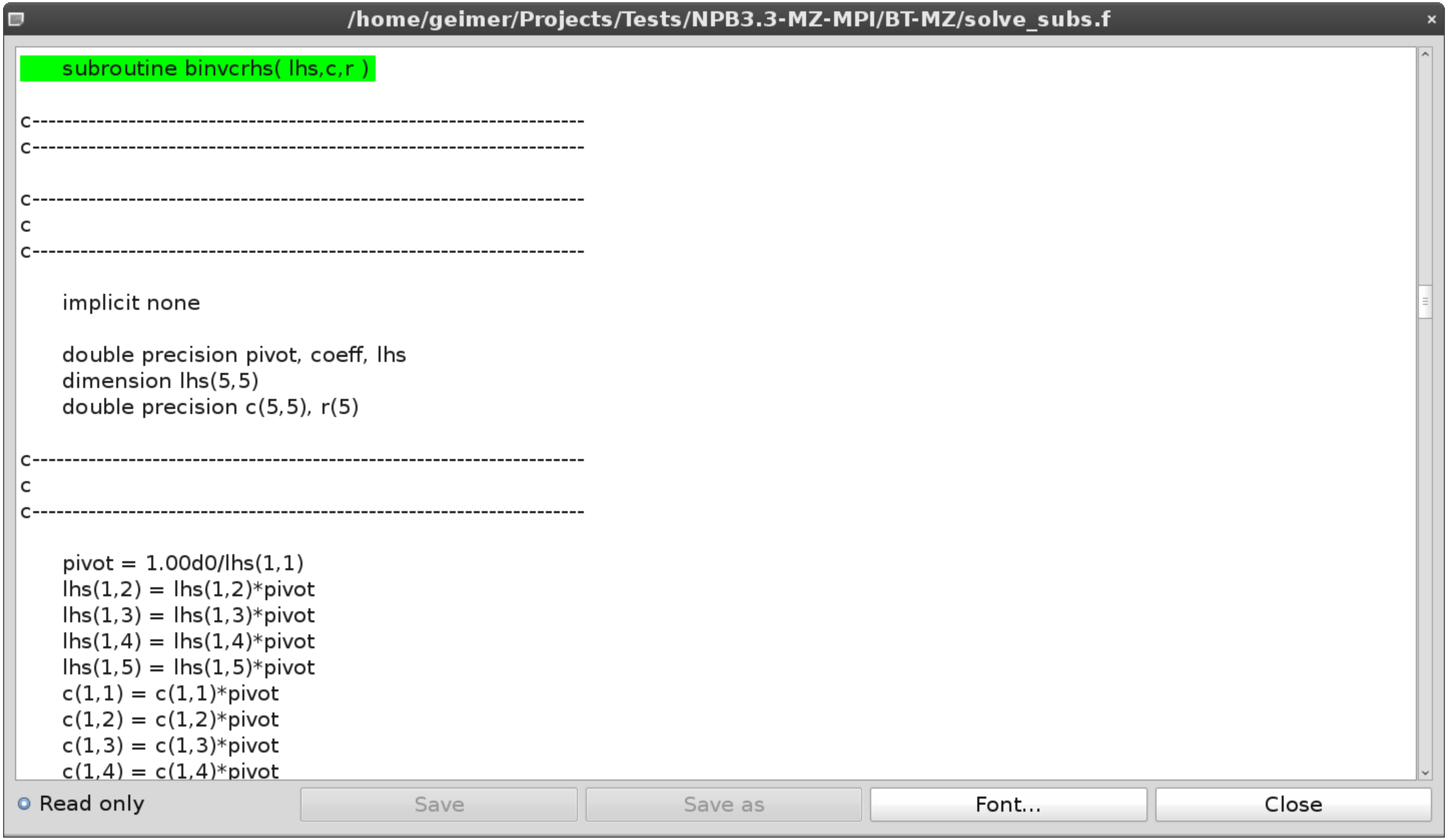
767.48

0.00 57.70

Shows the source code of the clicked ite

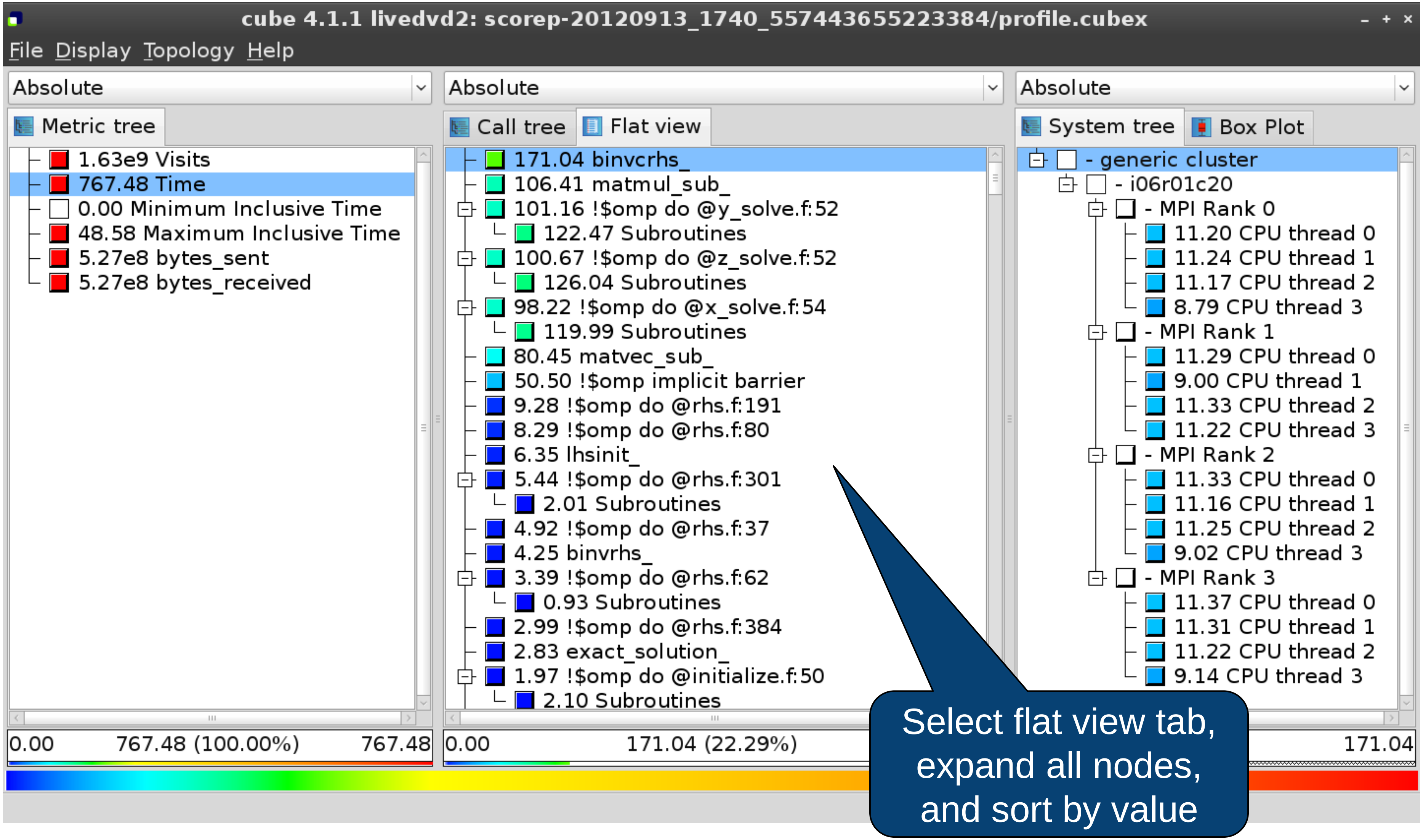
Right-click opens context menu

Source-code view


/home/geimer/Projects/Tests/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f
○ Read only
Save
Save as
Font...
Close

```
      subroutine binvcrhs( lhs,c,r )

c---------------------------------------------------------------------
c---------------------------------------------------------------------

c---------------------------------------------------------------------
c
c---------------------------------------------------------------------

      implicit none

      double precision pivot, coeff, lhs
      dimension lhs(5,5)
      double precision c(5,5), r(5)

c---------------------------------------------------------------------
c
c---------------------------------------------------------------------

      pivot = 1.00d0/lhs(1,1)
      lhs(1,2) = lhs(1,2)*pivot
      lhs(1,3) = lhs(1,3)*pivot
      lhs(1,4) = lhs(1,4)*pivot
      lhs(1,5) = lhs(1,5)*pivot
      c(1,1) = c(1,1)*pivot
      c(1,2) = c(1,2)*pivot
      c(1,3) = c(1,3)*pivot
      c(1,4) = c(1,4)*pivot
```

Flat profile view



cube 4.1.1 livedvd2: scorep-20120913_1740_557443655223384/profile.cubex

```
1.63e9 Visits
767.48 Time
0.00 Minimum Inclusive Time
48.58 Maximum Inclusive Time
5.27e8 bytes_sent
5.27e8 bytes_received
```

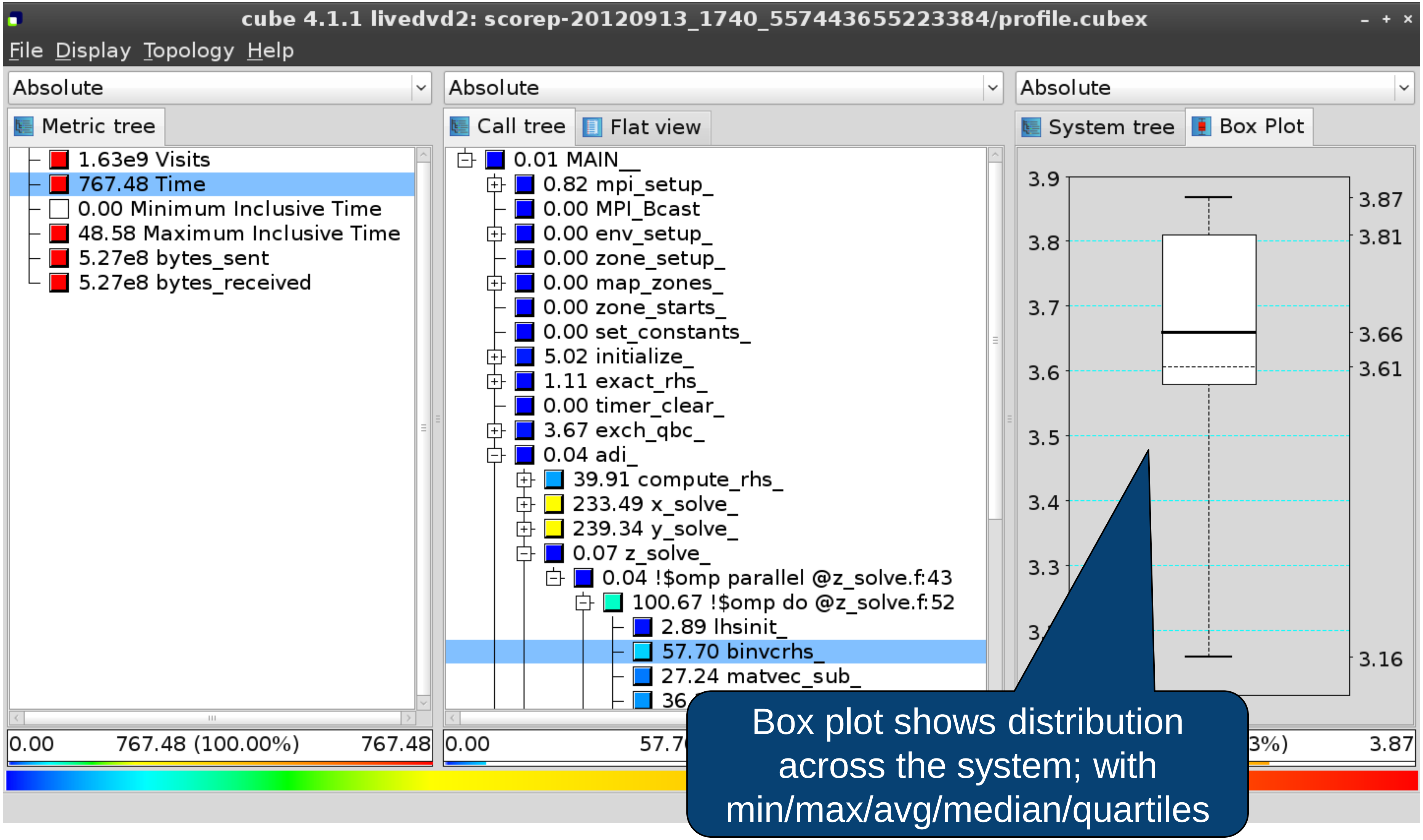
```
171.04 binvcrhs_
106.41 matmul_sub_
101.16 !$omp do @y_solve.f:52
    122.47 Subroutines
100.67 !$omp do @z_solve.f:52
    126.04 Subroutines
98.22 !$omp do @x_solve.f:54
    119.99 Subroutines
80.45 matvec_sub_
50.50 !$omp implicit barrier
9.28 !$omp do @rhs.f:191
8.29 !$omp do @rhs.f:80
6.35 lhsinit_
5.44 !$omp do @rhs.f:301
    2.01 Subroutines
4.92 !$omp do @rhs.f:37
4.25 binvrhs_
3.39 !$omp do @rhs.f:62
    0.93 Subroutines
2.99 !$omp do @rhs.f:384
2.83 exact_solution_
1.97 !$omp do @initialize.f:50
    2.10 Subroutines
```

```
- generic cluster
  - i06r01c20
    - MPI Rank 0
        11.20 CPU thread 0
        11.24 CPU thread 1
        11.17 CPU thread 2
        8.79 CPU thread 3
    - MPI Rank 1
        11.29 CPU thread 0
        9.00 CPU thread 1
        11.33 CPU thread 2
        11.22 CPU thread 3
    - MPI Rank 2
        11.33 CPU thread 0
        11.16 CPU thread 1
        11.25 CPU thread 2
        9.02 CPU thread 3
    - MPI Rank 3
        11.37 CPU thread 0
        11.31 CPU thread 1
        11.22 CPU thread 2
        9.14 CPU thread 3
```

```
0.00    767.48 (100.00%)    767.48
0.00    171.04 (22.29%)
171.04
```

Select flat view tab, expand all nodes, and sort by value

Box plot view



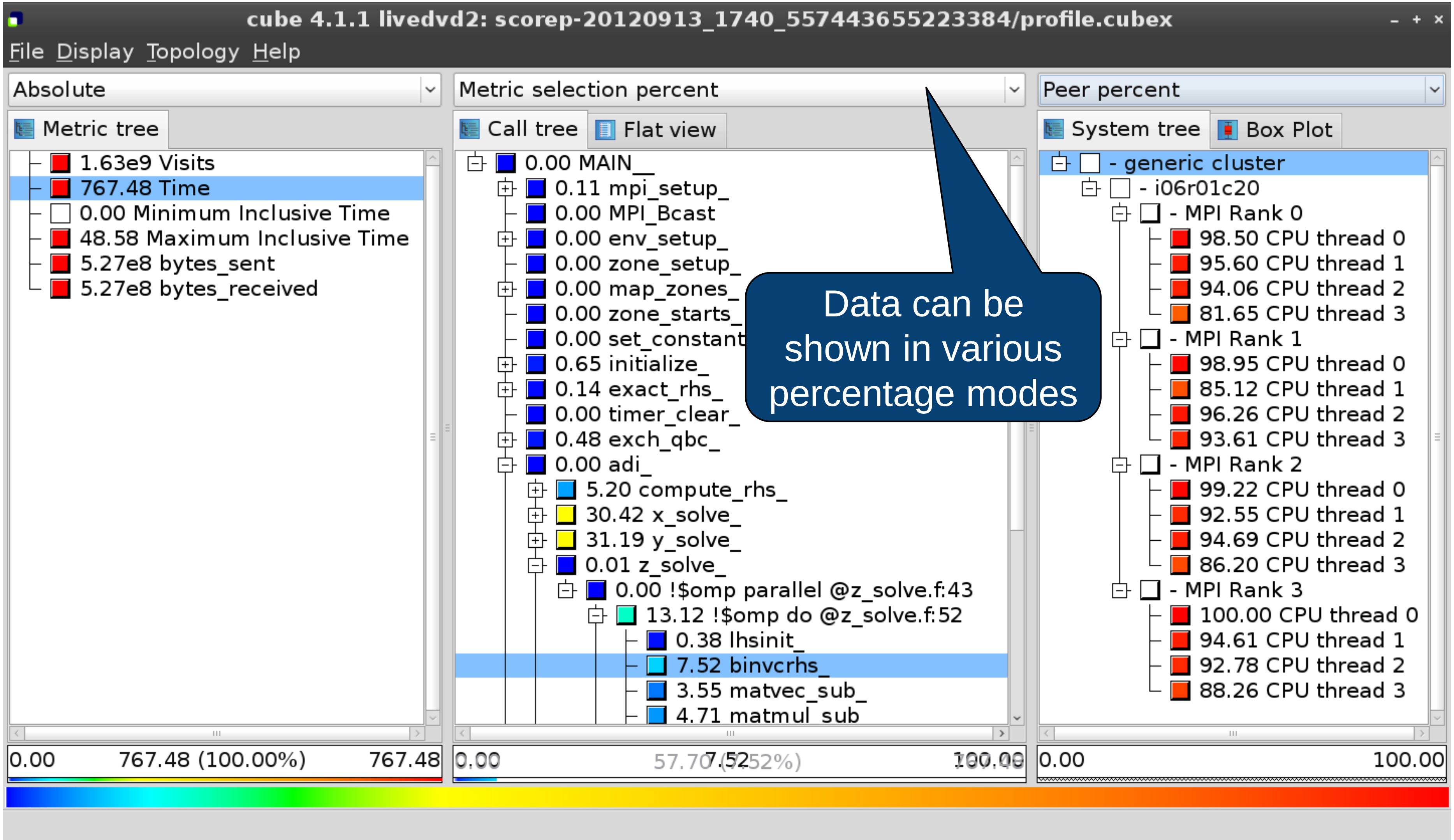
```
🟥 1.63e9 Visits
🟥 767.48 Time
□ 0.00 Minimum Inclusive Time
🟥 48.58 Maximum Inclusive Time
🟥 5.27e8 bytes_sent
🟥 5.27e8 bytes_received
```

```
⊟ 0.01 MAIN__
  ⊞ 0.82 mpi_setup_
    0.00 MPI_Bcast
  ⊞ 0.00 env_setup_
    0.00 zone_setup_
  ⊞ 0.00 map_zones_
    0.00 zone_starts_
    0.00 set_constants_
  ⊞ 5.02 initialize_
  ⊞ 1.11 exact_rhs_
    0.00 timer_clear_
  ⊞ 3.67 exch_qbc_
  ⊟ 0.04 adi_
    ⊞ 39.91 compute_rhs_
    ⊞ 233.49 x_solve_
    ⊞ 239.34 y_solve_
    ⊟ 0.07 z_solve_
      ⊟ 0.04 !$omp parallel @z_solve.f:43
        ⊟ 100.67 !$omp do @z_solve.f:52
            2.89 lhsinit_
            57.70 binvcrhs_
            27.24 matvec_sub_
            36
```

```
0.00      767.48 (100.00%)      767.48
0.00      57.7
3%)      3.87
```

Box plot shows distribution across the system; with min/max/avg/median/quartiles

Alternative display modes



cube 4.1.1 livedvd2: scorep-20120913_1740_557443655223384/profile.cubex

Absolute | Metric selection percent | Peer percent

```
- 🟥 1.63e9 Visits
- 🟥 767.48 Time
- ☐ 0.00 Minimum Inclusive Time
- 🟥 48.58 Maximum Inclusive Time
- 🟥 5.27e8 bytes_sent
- 🟥 5.27e8 bytes_received
```

```
⊟ 🟦 0.00 MAIN__
  ⊞ 🟦 0.11 mpi_setup_
  - 🟦 0.00 MPI_Bcast
  ⊞ 🟦 0.00 env_setup_
  - 🟦 0.00 zone_setup_
  ⊞ 🟦 0.00 map_zones_
  - 🟦 0.00 zone_starts_
  - 🟦 0.00 set_constant
  ⊞ 🟦 0.65 initialize_
  ⊞ 🟦 0.14 exact_rhs_
  - 🟦 0.00 timer_clear_
  ⊞ 🟦 0.48 exch_qbc_
  ⊟ 🟦 0.00 adi_
    ⊞ 🟦 5.20 compute_rhs_
    ⊞ 🟨 30.42 x_solve_
    ⊞ 🟨 31.19 y_solve_
    ⊟ 🟦 0.01 z_solve_
      ⊟ 🟦 0.00 !$omp parallel @z_solve.f:43
        ⊟ 🟩 13.12 !$omp do @z_solve.f:52
          - 🟦 0.38 lhsinit_
          - 🟦 7.52 binvcrhs_
          - 🟦 3.55 matvec_sub_
          - 🟦 4.71 matmul_sub_
```

```
⊟ ☐ - generic cluster
  ⊟ ☐ - i06r01c20
    ⊟ ☐ - MPI Rank 0
      - 🟥 98.50 CPU thread 0
      - 🟥 95.60 CPU thread 1
      - 🟥 94.06 CPU thread 2
      - 🟧 81.65 CPU thread 3
    ⊟ ☐ - MPI Rank 1
      - 🟥 98.95 CPU thread 0
      - 🟧 85.12 CPU thread 1
      - 🟥 96.26 CPU thread 2
      - 🟥 93.61 CPU thread 3
    ⊟ ☐ - MPI Rank 2
      - 🟥 99.22 CPU thread 0
      - 🟥 92.55 CPU thread 1
      - 🟥 94.69 CPU thread 2
      - 🟧 86.20 CPU thread 3
    ⊟ ☐ - MPI Rank 3
      - 🟥 100.00 CPU thread 0
      - 🟥 94.61 CPU thread 1
      - 🟥 92.78 CPU thread 2
      - 🟧 88.26 CPU thread 3
```

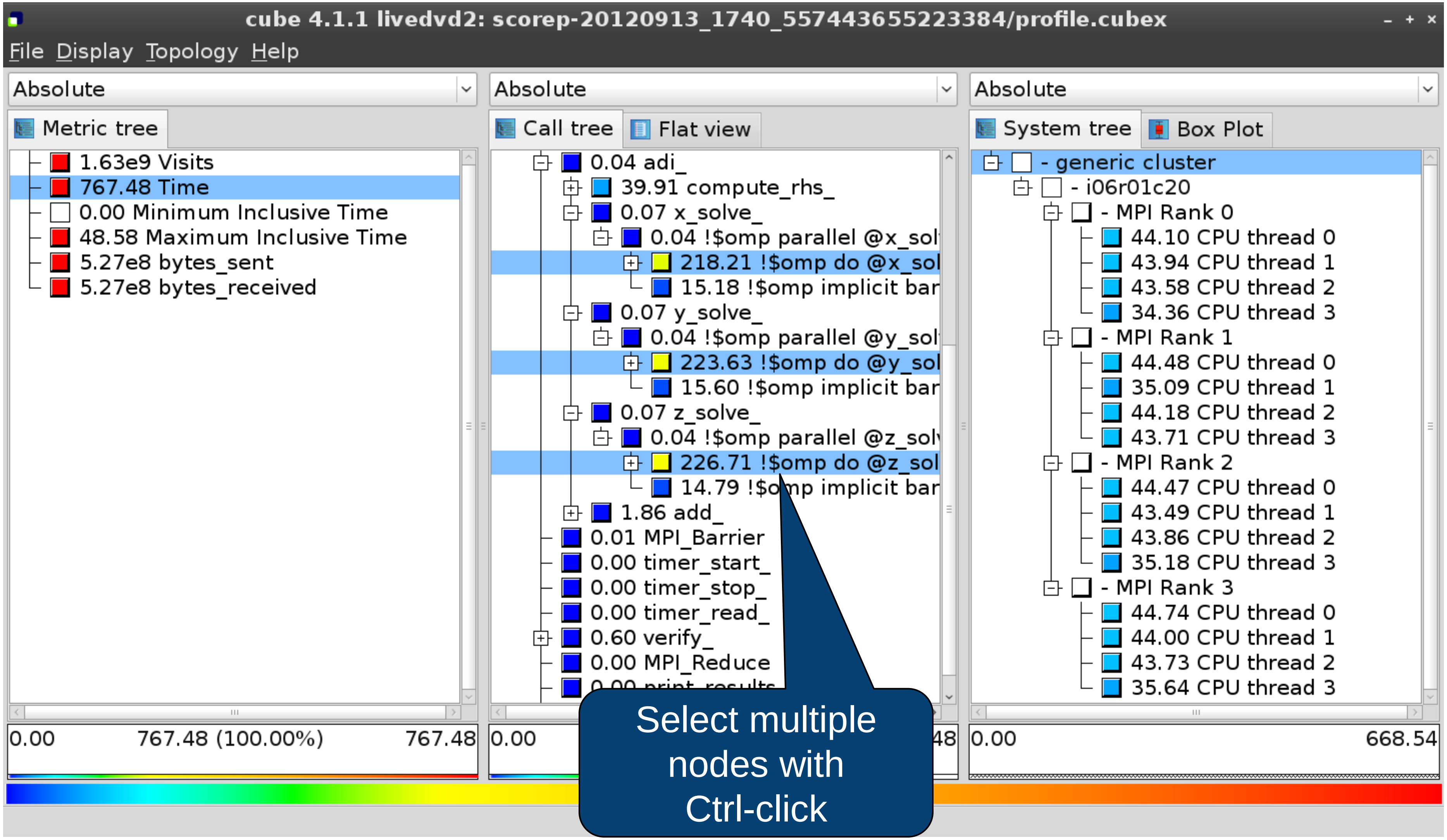
0.00 767.48 (100.00%) 767.48 | 0.00 57.70 7.52 (7.52%) 100.00 | 0.00 100.00

Data can be shown in various percentage modes

Important display modes

- Absolute
  - – Absolute value shown in seconds/bytes/counts
- Selection percent
  - – Value shown as percentage w.r.t. the selected node “on the left“ (metric/call path)
- Peer percent (system tree only)
  - – Value shown as percentage relative to the maximum peer value

Multiple selection



cube 4.1.1 livedvd2: scorep-20120913_1740_557443655223384/profile.cubex

Metric tree:

```
1.63e9 Visits
767.48 Time
0.00 Minimum Inclusive Time
48.58 Maximum Inclusive Time
5.27e8 bytes_sent
5.27e8 bytes_received
```

0.00 767.48 (100.00%) 767.48

Call tree:

```
0.04 adi_
  39.91 compute_rhs_
  0.07 x_solve_
    0.04 !$omp parallel @x_sol
      218.21 !$omp do @x_sol
      15.18 !$omp implicit bar
  0.07 y_solve_
    0.04 !$omp parallel @y_sol
      223.63 !$omp do @y_sol
      15.60 !$omp implicit bar
  0.07 z_solve_
    0.04 !$omp parallel @z_solv
      226.71 !$omp do @z_sol
      14.79 !$omp implicit bar
  1.86 add_
0.01 MPI_Barrier
0.00 timer_start_
0.00 timer_stop_
0.00 timer_read_
0.60 verify_
0.00 MPI_Reduce
0.00 print_results
```

0.00 48

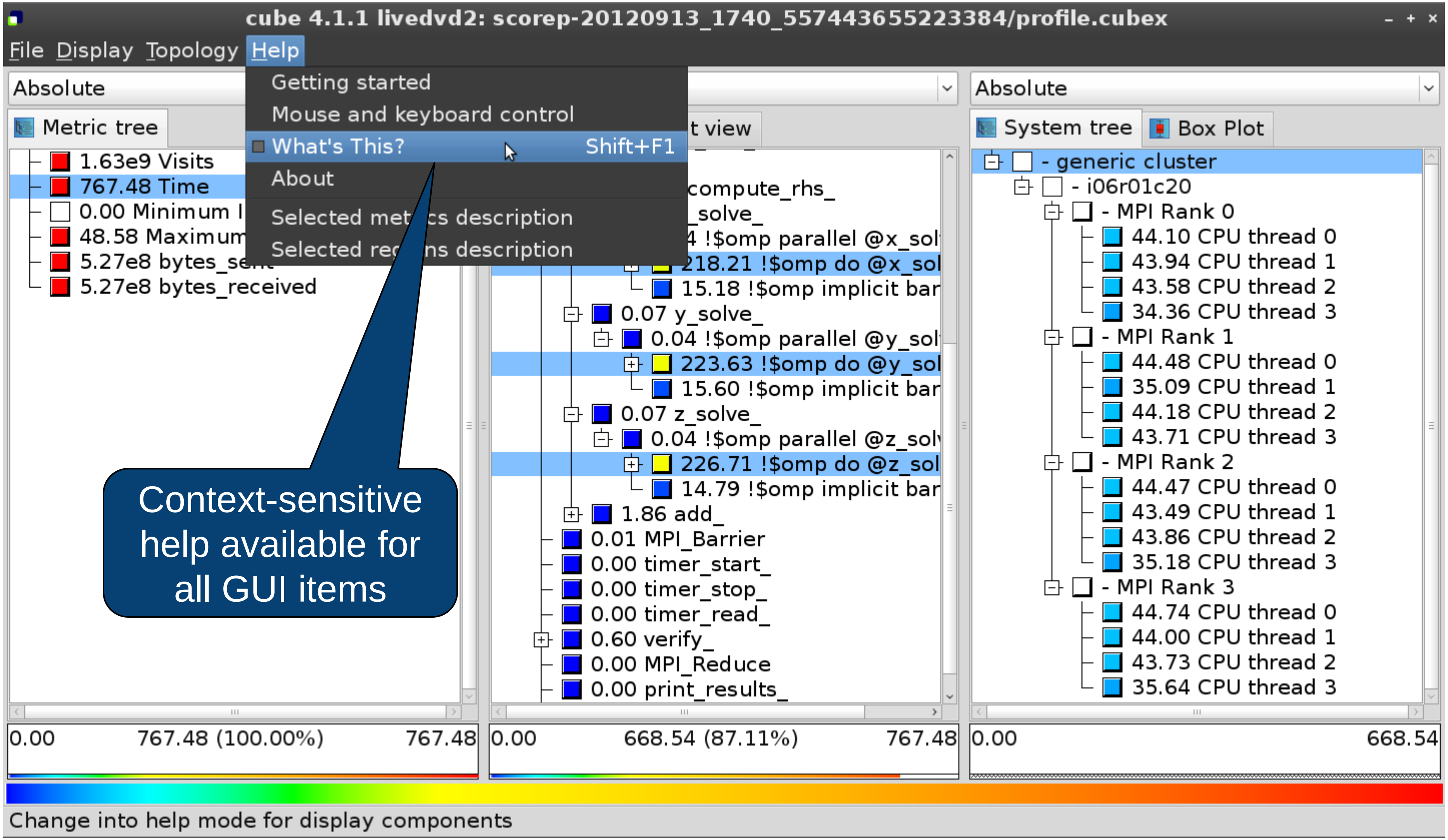
System tree:

```
- generic cluster
  - i06r01c20
    - MPI Rank 0
      44.10 CPU thread 0
      43.94 CPU thread 1
      43.58 CPU thread 2
      34.36 CPU thread 3
    - MPI Rank 1
      44.48 CPU thread 0
      35.09 CPU thread 1
      44.18 CPU thread 2
      43.71 CPU thread 3
    - MPI Rank 2
      44.47 CPU thread 0
      43.49 CPU thread 1
      43.86 CPU thread 2
      35.18 CPU thread 3
    - MPI Rank 3
      44.74 CPU thread 0
      44.00 CPU thread 1
      43.73 CPU thread 2
      35.64 CPU thread 3
```

0.00 668.54

Select multiple nodes with Ctrl-click

Context-sensitive help



cube 4.1.1 livedvd2: scorep-20120913_1740_557443655223384/profile.cubex
File Display Topology Help
Getting started
Mouse and keyboard control
What's This? Shift+F1
About
Selected metrics description
Selected regions description
Absolute
Absolute
Metric tree
t view
System tree
Box Plot

```
1.63e9 Visits
767.48 Time
0.00 Minimum I
48.58 Maximum
5.27e8 bytes_sent
5.27e8 bytes_received
```

```
compute_rhs_
_solve_
4 !$omp parallel @x_sol
218.21 !$omp do @x_sol
15.18 !$omp implicit bar
0.07 y_solve_
0.04 !$omp parallel @y_sol
223.63 !$omp do @y_sol
15.60 !$omp implicit bar
0.07 z_solve_
0.04 !$omp parallel @z_sol
226.71 !$omp do @z_sol
14.79 !$omp implicit bar
1.86 add_
0.01 MPI_Barrier
0.00 timer_start_
0.00 timer_stop_
0.00 timer_read_
0.60 verify_
0.00 MPI_Reduce
0.00 print_results_
```

```
- generic cluster
  - i06r01c20
    - MPI Rank 0
      44.10 CPU thread 0
      43.94 CPU thread 1
      43.58 CPU thread 2
      34.36 CPU thread 3
    - MPI Rank 1
      44.48 CPU thread 0
      35.09 CPU thread 1
      44.18 CPU thread 2
      43.71 CPU thread 3
    - MPI Rank 2
      44.47 CPU thread 0
      43.49 CPU thread 1
      43.86 CPU thread 2
      35.18 CPU thread 3
    - MPI Rank 3
      44.74 CPU thread 0
      44.00 CPU thread 1
      43.73 CPU thread 2
      35.64 CPU thread 3
```

```
0.00    767.48 (100.00%)    767.48
0.00    668.54 (87.11%)    767.48
0.00    668.54
```

Change into help mode for display components

Context-sensitive help available for all GUI items

CUBE algebra utilities

- Extracting solver sub-tree from analysis report

```
% cube_cut  -r '<<ITERATION>>'  scorep_bt-mz_W_4x4_sum/profile.cubex
Writing cut.cubex... done.
```

- Calculating difference of two reports

```
% cube_diff  scorep_bt-mz_W_4x4_sum/profile.cubex  cut.cubex
Writing diff.cubex... done.
```

- Additional utilities for merging, calculating mean, etc.
  - Default output of cube_*utility* is a new report *utility*.cubex
- Further utilities for report scoring & statistics
- Run utility with “-h” (or no arguments) for brief usage info

Further information

- CUBE
  - Parallel program analysis report exploration tools
    - Libraries for XML report reading & writing
    - Algebra utilities for report processing
    - GUI for interactive analysis exploration
  - Available under New BSD open-source license
  - Documentation & sources:
    - http://www.scalasca.org
  - User guide also part of installation:
    - `cube-config --cube-dir`/share/doc/CubeGuide.pdf
  - Contact:
    - mailto: scalasca@fz-juelich.de

Score-P hands-on:
NPB-MZ-MPI / BT (filtered)

Performance Analysis Steps

1. Reference preparation for validation
2. Program instrumentation
3. Summary measurement collection
4. Summary analysis report examination
5. Summary experiment scoring
6. Summary measurement collection with filtering
7. Filtered summary analysis report examination
8. Event trace collection
9. Event trace examination & analysis

Congratulations!?

- If you made it this far, you successfully used Score-P to
  - instrument the application
  - analyze its execution with a summary measurement, and
  - examine it with one the interactive analysis report explorer GUIs
- ... revealing the call-path profile annotated with
  - the “Time” metric
  - Visit counts
  - MPI message statistics (bytes sent/received)
- ... but how ***good*** was the measurement?
  - The measured execution produced the desired valid result
  - however, the execution took rather longer than expected!
    - even when ignoring measurement start-up/completion, therefore
    - it was probably dilated by instrumentation/measurement overhead

BT-MZ Summary Analysis Result Scoring

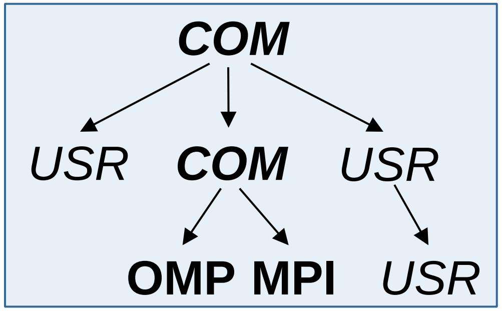
- Report scoring as textual output

```
% scorep-score scorep_bt-mz_W_4x4_sum/profile.cubex
Estimated aggregate size of event trace:                   1025MB
Estimated requirements for largest trace buffer (max_buf): 265MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       273MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=273MB to avoid intermedi
 or reduce requirements using USR regions

flt      type  max_buf[B]      visits time[s] time[%] time/visit[us]  region
          ALL 277,799,918 41,157,533   91.76   100.0           2.23  ALL
          USR 274,792,492 40,418,321   11.38    12.4           0.28  USR
          OMP   6,882,860    685,952   51.42    56.0          74.96  OMP
          COM     371,956     45,944   15.20    16.6         330.81  COM
          MPI     102,286      7,316   13.76    15.0        1880.84
```

1 GB total memory (265 MB per rank)!

High visit count, but very low time/visit ration!

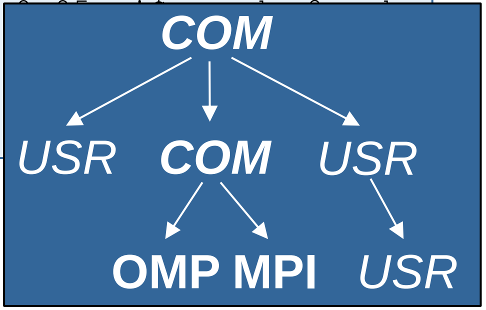
- Region/callpath classification
  - MPI (pure MPI library functions)
  - OMP (pure OpenMP functions/regions)
  - USR (user-level source local computation)
  - COM (“combined” USR + OpenMP/MPI)
  - ANY/ALL (aggregate of all region types)



- Score report breakdown by region

```
% scorep-score -r scorep_bt-mz_W_4x4_sum/profile.cubex
   [...]
flt       type   max_buf[B]       visits  time[s]  time[%]  time/visit[us]    region
          USR   85,774,338  12,516,672     3.56      3.9             0.28    matmul_sub_
          USR   85,774,338  12,516,672     2.87      3.1             0.23    matvec_sub_
          USR   85,774,338  12,516,672     4.15      4.5             0.33    binvcrhs_
          USR    7,974,876   1,170,624     0.34      0.4             0.29    lhsinit_
          USR    7,974,876   1,170,624     0.32      0.3             0.27    binvrhs
          USR    3,473,912     526,848     0.14      0.1             0.26    exact_solution…
          OMP      410,040      25,728     0.01      0.0             0.50    !$omp parallel…
          OMP      410,040      25,728     0.01      0.0             0.49    !$omp parallel…
          OMP      410,040         728     0.01      0.0             0.48    !$omp parallel…
          OMP      410,040                           0.0             0.47    !$omp parallel…
          OMP      209,040                           0.0             0.98    !$omp do @exch…
          OMP      209,040                           0.0             0.97    !$omp do @exch…
          OMP      209,040                           0.3             9.69    !$omp implicit…
          OMP      209,040                           0.3             9.66    !$omp implicit…
          OMP      209,040                           0.0
          OMP      209,040      25,728     0.24      0.3
          OMP      209,040      25,728     0.02      0.0
   [...]
```

More than 270 MB just for these 6 regions



BT-MZ Summary Analysis Score

- Summary measurement analysis score reveals
  - Total size of event trace would be ~1 GB
  - Maximum trace buffer size would be ~265 MB per rank
    - smaller buffer would require flushes to disk during measurement resulting in substantial perturbation
  - 99.8% of the trace requirements are for USR regions
    - purely computational routines never found on COM call-paths common to communication routines or OpenMP parallel regions
  - These USR regions contribute around 12.4% of total time
    - however, much of that is very likely to be measurement overhead for frequently-executed small routines (high visit count but very low time/visit ratio)
- Advisable to tune measurement configuration
  - Specify an adequate trace buffer size
  - Specify a filter file listing (USR) regions not to be measured

Performance Analysis Steps

1. Reference preparation for validation
2. Program instrumentation
3. Summary measurement collection
4. Summary analysis report examination
5. Summary experiment scoring
6. Summary measurement collection with filtering
7. Filtered summary analysis report examination
8. Event trace collection
9. Event trace examination & analysis

BT-MZ Summary Analysis Report Filtering

- Report scoring with prospective filter listing
  6 USR regions

```
% cat ../config/scorep.filt
SCOREP_REGION_NAMES_BEGIN EXCLUDE
binvcrhs*
matmul_sub*
matvec_sub*
exact_solution*
binvrhs*
lhs*init*
timer_*

% scorep-score -f ../config/scorep.filt scorep_bt-mz_W_4x4_sum/profile.cubex
Estimated aggregate size of event trace:                   23MB
Estimated requirements for largest trace buffer (max_buf): 8MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       16MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=16MB to avoid inte   ediate flushes
 or reduce requirements using USR regions filters.)
```

23 MB of memory in total,
8 MB per rank!

BT-MZ Summary Analysis Report Filtering

- Score report breakdown by region

```
% scorep-score -r -f ../config/scorep.filt \
> scorep_bt-mz_W_4x4_sum/profile.cubex
flt     type  max_buf[B]     visits time[s] time[%] time/visit[us]  region
 -       ALL 277,799,918 41,157,533   91.76   100.0           2.23  ALL
 -       USR 274,792,492 40,418,321   11.38    12.4           0.28  USR
 -       OMP   6,882,860    685,952   51.42    56.0          74.96  OMP
 -       COM     371,956     45,944   15.20    16.6         330.81  COM
 -       MPI     102,286      7,316   13.76    15.0        1880.84  MPI

 *       ALL   7,357,804    739,321   80.38    87.6         108.72  ALL-FLT
 +       FLT 274,791,764 40,418,212   11.37    12.4           0.28  FLT
 -       OMP   6,882,860    685,952   51.42    56.0          74.96  OMP-FLT
 *       COM     371,956     45,944   15.20    16.6         330.81  COM-FLT
 -       MPI     102,286      7,316   13.76    15.0        1880.84  MPI-FLT
 *       USR         728        109    0.00     0.0           2.38  USR-FLT
 [...]
```

Filtered routines marked with ‘+’

BT-MZ Filtered Summary Measurement

- Set new experiment directory and re-run measurement with new filter configuration

```
% export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_W_4x4_sum_filtered
% export SCOREP_FILTERING_FILE=../config/scorep.filt
% OMP_NUM_THREADS=4  mpiexec -np 4 ./bt-mz_W.4
 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
 Number of zones:   4 x   4
 Iterations:  200   dt:   0.000800
 Number of active processes:     4
 Use the default load factors with threads
 Total number of threads:     16  (  4.0 threads/process)
 Use the default load factors with threads

 Time step    1
 Time step   20
  [...]
 Time step  180
 Time step  200
 Verification Successful

 BT-MZ Benchmark Completed.
 Time in seconds = 8.11
```

BT-MZ Tuned Summary Analysis Report Score

- Scoring of new analysis report as textual output

```
% scorep-score scorep_bt-mz_W_4x4_sum_filtered/profile.cubex
Estimated aggregate size of event trace:                   23MB
Estimated requirements for largest trace buffer (max_buf): 8MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       16MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=16MB to avoid intermediate flushes
 or reduce requirements using USR regions filters.)

flt     type max_buf[B]  visits time[s] time[%] time/visit[us]  region
         ALL  7,357,804 739,321   25.32   100.0          34.25  ALL
         OMP  6,882,860 685,952   16.64    65.7          24.26  OMP
         COM    371,956  45,944    3.90    15.4          84.87  COM
         MPI    102,286   7,316    4.78    18.9         653.21  MPI
         USR        728     109    0.00     0.0           2.41  USR
```

- Significant reduction in runtime (measurement overhead)
  - Not only reduced time for USR regions, but MPI/OMP reduced too!
- Further measurement tuning (filtering) may be appropriate
  - e.g., use “timer_*” to filter timer_start_, timer_read_, etc.

Virtual Institute – High Productivity Supercomputing

Profile Examination with TAU ParaProf

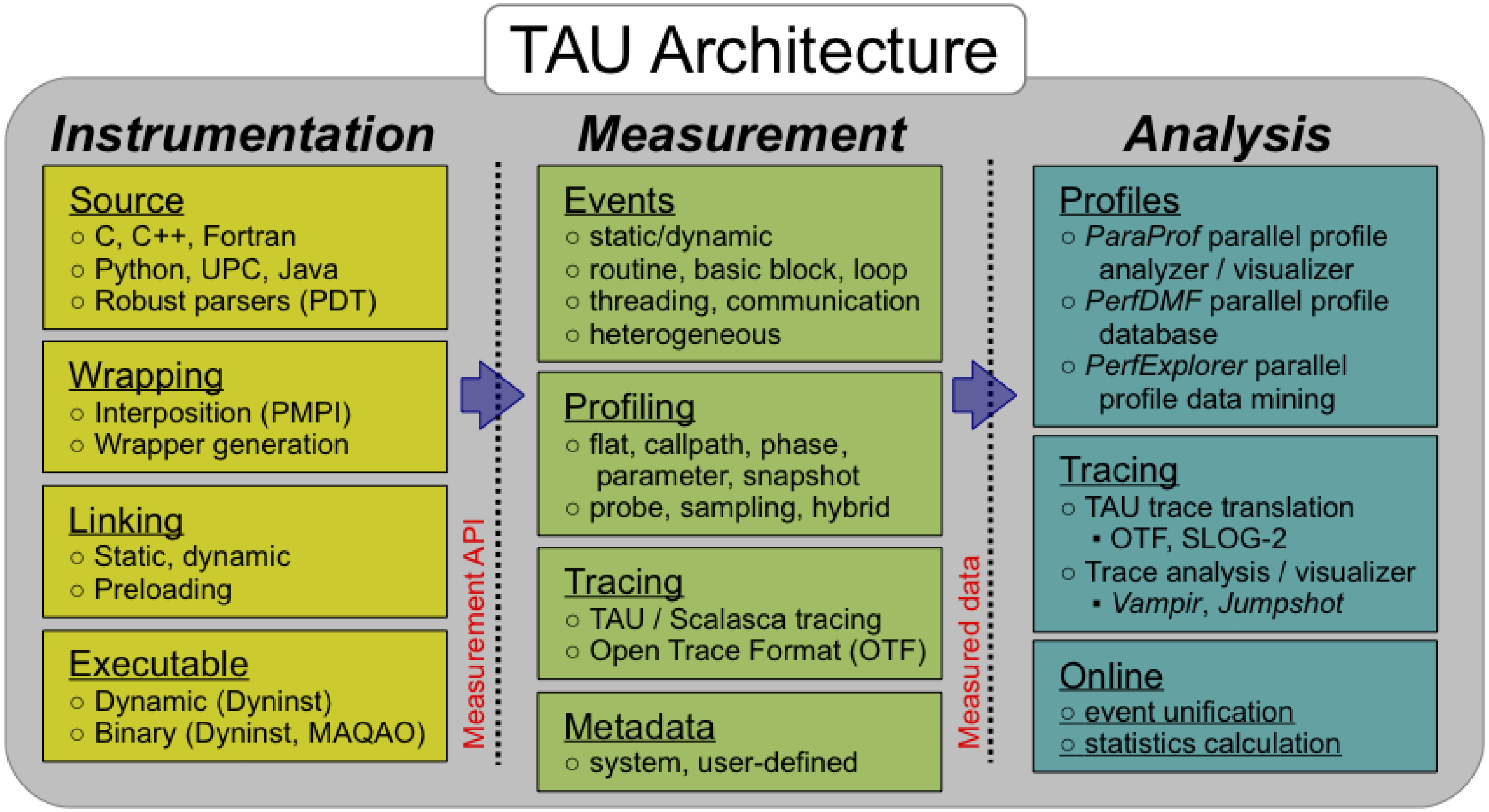
Sameer Shende

Performance Research Lab, University of Oregon

http://TAU.uoregon.edu

TAU Performance System®

- Parallel performance framework and toolkit
  - Supports all HPC platforms, compilers, runtime system
  - Provides portable instrumentation, measurement, analysis



TAU Architecture

Instrumentation

Source
- C, C++, Fortran
- Python, UPC, Java
- Robust parsers (PDT)

Wrapping
- Interposition (PMPI)
- Wrapper generation

Linking
- Static, dynamic
- Preloading

Executable
- Dynamic (Dyninst)
- Binary (Dyninst, MAQAO)

Measurement API

Measurement

Events
- static/dynamic
- routine, basic block, loop
- threading, communication
- heterogeneous

Profiling
- flat, callpath, phase, parameter, snapshot
- probe, sampling, hybrid

Tracing
- TAU / Scalasca tracing
- Open Trace Format (OTF)

Metadata
- system, user-defined

Measured data

Analysis

Profiles
- *ParaProf* parallel profile analyzer / visualizer
- *PerfDMF* parallel profile database
- *PerfExplorer* parallel profile data mining

Tracing
- TAU trace translation
  - OTF, SLOG-2
- Trace analysis / visualizer
  - *Vampir*, *Jumpshot*

Online
- event unification
- statistics calculation

TAU Performance System®

- Instrumentation
  - Fortran, C++, C, UPC, Java, Python, Chapel
  - Automatic instrumentation
- Measurement and analysis support
  - MPI, OpenSHMEM, ARMCI, PGAS, DMAPP
  - pthreads, OpenMP, hybrid, other thread models
  - GPU, CUDA, OpenCL, OpenACC
  - Parallel profiling and tracing
  - Use of Score-P for native OTF2 and CUBEX generation
  - Efficient callpath proflles and trace generation using Score-P
- Analysis
  - Parallel profile analysis (ParaProf), data mining (PerfExplorer)
  - Performance database technology (TAUdb)
  - 3D profile browser

TAU

- TAU supports both sampling and direct instrumentation
- Memory debugging as well as I/O performance evaluation
- Profiling as well as tracing
- Interfaces with Score-P for more efficient measurements
- TAU’s instrumentation covers:
  - Runtime library interposition (tau_exec)
  - Compiler-based instrumentation
  - PDT based Source level instrumentation: routine & loop
  - Event based sampling (TAU_SAMPLING=1)
  - Callstack unwinding with sampling (TAU_EBS_UNWIND=1)
  - OpenMP Tools Interface (OMPT, tau_exec –T ompt)
  - CUDA CUPTI, OpenCL (tau_exec -T cupti   -cupti)

Understanding Application Performance using TAU

- How much time is spent in each application routine and outer *loops*? Within loops, what is the contribution of each *statement*?
- How many instructions are executed in these code regions? Floating point, Level 1 and 2 *data cache misses*, hits, branches taken?
- What is the memory usage of the code? When and where is memory allocated/de-allocated? Are there any memory leaks?
- What are the I/O characteristics of the code? What is the peak read and write *bandwidth* of individual calls, total volume?
- What is the contribution of each *phase* of the program? What is the time wasted/spent waiting for collectives, and I/O operations in Initialization, Computation, I/O phases?
- How does the application *scale*? What is the efficiency, runtime breakdown of performance across different core counts?

Using TAU

- TAU supports several measurement and thread options

  Phase profiling, profiling with hardware counters, MPI library, CUDA…

  Each measurement configuration of TAU corresponds to a unique stub makefile (configuration file) and library that is generated when you configure it

- To instrument source code automatically using PDT

  Choose an appropriate TAU stub makefile in <arch>/lib:

  % export TAU_MAKEFILE=$TAU/Makefile.tau-mpi-pdt

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

  % export PATH=$TAU_ROOT/x86_64/bin:$PATH

  % export TAU=$TAU_ROOT/x86_64/lib

  Use tau_f90.sh, tau_cxx.sh, tau_upc.sh, or tau_cc.sh as F90, C++, UPC, or C compilers respectively:

  % mpif90 foo.f90 changes to

  % tau_f90.sh foo.f90

- Set runtime environment variables, execute application and analyze performance data:

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

Automatic Source Instrumentation using PDT

```
% module load openmpi tau; ls $TAU/Makefile.*
Makefile.tau-icpc-papi-mpi-pdt
Makefile.tau-icpc-papi-mpi-pthread-pdt
Makefile.tau-icpc-papi-ompt-mpi-pdt-openmp
Makefile.tau-mpi-pdt
Makefile.tau-papi-mpi-pdt
Makefile.tau-papi-mpi-pdt-openmp-opari-scorep
Makefile.tau-papi-mpi-pdt-scorep
Makefile.tau-papi-mpi-pthread-pdt
Makefile.tau-papi-pthread-pdt
Makefile.tau-papi-shmem-mpi-pdt
```

- For an MPI+F90 application with Intel MPI, you may choose

Makefile.tau-mpi-pdt

  - Supports MPI instrumentation & PDT for automatic source instrumentation

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-mpi-pdt
% tau_f90.sh matmult.f90 -o matmult
% mpirun -np 4 ./matmult
% paraprof
```

Using TAU with Score-P

```
% export TAU=$TAU_ROOT/x86_64/lib
% export TAU_MAKEFILE=$TAU/Makefile.tau-papi-mpi-pdt-openmp-opari-scorep
% export OMP_NUM_THREADS=10
% make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh

% mpirun -np 4  ./matmult

% cd score*; paraprof profile.cubex &
```

Installing and Configuring TAU

- Installing PDT:
  - wget http://tau.uoregon.edu/pdt_lite.tgz
  - ./configure –prefix=<dir>; make ; make install
- Installing TAU:
  - wget http://tau.uoregon.edu/tau.tgz
  - ./configure –arch=x86_64 -bfd=download -pdt=<dir> -papi=<dir> ...
  - For MIC:
  - ./configure –arch=mic_linux –pdt=<dir> -pdt_c++=g++ -papi=dir …
  - make install
- Using TAU:
  - export TAU_MAKEFILE=<taudir>/x86_64/
    lib/Makefile.tau-<TAGS>
  - make CC=tau_cc.sh   CXX=tau_cxx.sh   F90=tau_f90.sh

Compile-Time Options

Optional parameters for the TAU_OPTIONS environment variable:

% tau_compiler.sh

-optVerbose	Turn on verbose debugging messages
-optCompInst	Use compiler based instrumentation
-optNoCompInst	Do not revert to compiler instrumentation if source instrumentation fails.
-optTrackIO	Wrap POSIX I/O call and calculates vol/bw of I/O operations (Requires TAU to be configured with –*iowrapper*)
-optTrackGOMP	Enable tracking GNU OpenMP runtime layer (used without –opari)
-optMemDbg	Enable runtime bounds checking (see TAU_MEMDBG_* env vars)
-optKeepFiles	Does not remove intermediate .pdb and .inst.* files
-optPreProcess	Preprocess sources (OpenMP, Fortran) before instrumentation
-optTauSelectFile=”<file>"	Specify selective instrumentation file for *tau_instrumentor*
-optTauWrapFile=”<file>"	Specify path to *link_options.tau* generated by *tau_gen_wrapper*
-optHeaderInst	Enable Instrumentation of headers
-optTrackUPCR	Track UPC runtime layer routines (used with tau_upc.sh)
-optLinking=""	Options passed to the linker. Typically $(TAU_MPI_FLIBS) $(TAU_LIBS) $(TAU_CXXLIBS)
-optCompile=""	Options passed to the compiler. Typically $(TAU_MPI_INCLUDE) $(TAU_INCLUDE) $(TAU_DEFS)
-optPdtF95Opts=""	Add options for Fortran parser in PDT (f95parse/gfparse) …

Compile-Time Options (contd.)

•Optional parameters for the TAU_OPTIONS environment variable:

% tau_compiler.sh

-optMICOffload	Links code for Intel MIC offloading, requires both host and MIC TAU libraries
-optShared (default)	Use TAU’s shared library (libTAU.so) instead of static library
-optPdtCxxOpts=“”	Options for C++ parser in PDT (cxxparse).
-optPdtF90Parser=“”	Specify a different Fortran parser
-optPdtCleanscapeParser	Specify the Cleanscape Fortran parser instead of GNU gfparser
-optTau=“”	Specify options to the tau_instrumentor
-optTrackDMAPP	Enable instrumentation of low-level DMAPP API calls on Cray
-optTrackPthread	Enable instrumentation of pthread calls

See tau_compiler.sh for a full list of TAU_OPTIONS.

…

Compiling Fortran Codes with TAU

- If your Fortran code uses free format in .f files (fixed is default for .f), you may use:

```
% export TAU_OPTIONS= ‘-optPdtF95Opts=“-R free” -optVerbose ’
```

- To use the compiler based instrumentation instead of PDT (source-based):

```
% export TAU_OPTIONS= ‘-optCompInst -optVerbose’
```

- If your Fortran code uses C preprocessor directives (#include, #ifdef, #endif):

```
% export TAU_OPTIONS= ‘-optPreProcess -optVerbose -optDetectMemoryLeaks’
```

- To use an instrumentation specification file:

```
% export TAU_OPTIONS= ‘-optTauSelectFile=select.tau -optVerbose -optPreProcess’
% cat select.tau
BEGIN_INSTRUMENT_SECTION
loops  routine=“#”
# this statement instruments all outer loops in all routines. # is wildcard as well as comment in first column.
END_INSTRUMENT_SECTION
```

Runtime Environment Variables

Environment Variable	Default	Description
TAU_TRACE	0	Setting to 1 turns on tracing
TAU_CALLPATH	0	Setting to 1 turns on callpath profiling
TAU_TRACK_MEMORY_LEAKS	0	Setting to 1 turns on leak detection (for use with –optMemDbg or tau_exec)
TAU_MEMDBG_PROTECT_ABOVE	0	Setting to 1 turns on bounds checking for dynamically allocated arrays. (Use with –optMemDbg or tau_exec –memory_debug).
TAU_CALLPATH_DEPTH	2	Specifies depth of callpath. Setting to 0 generates no callpath or routine information, setting to 1 generates flat profile and context events have just parent information (e.g., Heap Entry: foo)
TAU_SAMPLING	1	Setting to 1 enables event-based sampling.
TAU_TRACK_SIGNALS	0	Setting to 1 generate debugging callstack info when a program crashes
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,P_VIRTUAL_TIME,PAPI_FP_INS,PAPI_NATIVE_<event>:<subevent>)

Runtime Environment Variables (contd.)

Environment Variable	Default	Description
TAU_TRACK_MEMORY_LEAKS	0	Tracks allocates that were not de-allocated (needs –optMemDbg or tau_exec –memory)
TAU_EBS_SOURCE	TIME	Allows using PAPI hardware counters for periodic interrupts for EBS (e.g., TAU_EBS_SOURCE=PAPI_TOT_INS when TAU_SAMPLING=1)
TAU_EBS_PERIOD	100000	Specifies the overflow count for interrupts
TAU_MEMDBG_ALLOC_MIN/MAX	0	Byte size minimum and maximum subject to bounds checking (used with TAU_MEMDBG_PROTECT_*)
TAU_MEMDBG_OVERHEAD	0	Specifies the number of bytes for TAU's memory overhead for memory debugging.
TAU_MEMDBG_PROTECT_BELOW/ABOVE	0	Setting to 1 enables tracking runtime bounds checking below or above the array bounds (requires –optMemDbg while building or tau_exec –memory)
TAU_MEMDBG_ZERO_MALLOC	0	Setting to 1 enables tracking zero byte allocations as invalid memory allocations.
TAU_MEMDBG_PROTECT_FREE	0	Setting to 1 detects invalid accesses to deallocated memory that should not be referenced until it is reallocated (requires –optMemDbg or tau_exec –memory)
TAU_MEMDBG_ATTEMPT_CONTINUE	0	Setting to 1 allows TAU to record and continue execution when a memory error occurs at runtime.
TAU_MEMDBG_FILL_GAP	Undefined	Initial value for gap bytes
TAU_MEMDBG_ALINGMENT	Sizeof(int)	Byte alignment for memory allocations
TAU_EVENT_THRESHOLD	0.5	Define a threshold value (e.g., .25 is 25%) to trigger marker events for min/max

Simplifying TAU’s usage (tau_exec)

- Uninstrumented execution
  - % mpirun -np 4 ./a.out
- Track MPI performance
  - % mpirun -np 4 tau_exec ./a.out
- Track POSIX I/O and MPI performance (MPI enabled by default)
  - % mpirun -np 4 tau_exec –T mpi,pdt –io ./a.out
- Track memory operations
  - % export TAU_TRACK_MEMORY_LEAKS=1
  - % mpirun –np 8 tau_exec –memory_debug ./a.out (bounds check)
- Use event based sampling (compile with –g)
  - % mpirun –np 8 tau_exec –ebs ./a.out
  - Also –ebs_source=<PAPI_COUNTER> -ebs_period=<overflow_count>
- Load wrapper interposition library
  - % mpirun –np 8 tau_exec –loadlib=<path/libwrapper.so> ./a.out
- Track GPGPU operations
  - % mpirun –np 8 tau_exec –cupti ./a.out
  - % mpirun –np 8 tau_exec –opencl ./a.out

Binary Rewriting Instrumentation

- Support for both static and dynamic executables
- Specify a list of routines to instrument
- Specify the TAU measurement library to be injected
- MAQAO:

```
% tau_rewrite -T [tags] a.out -o a.inst
```

- Dyninst:

```
% tau_run -T [tags] a.out -o a.inst
```

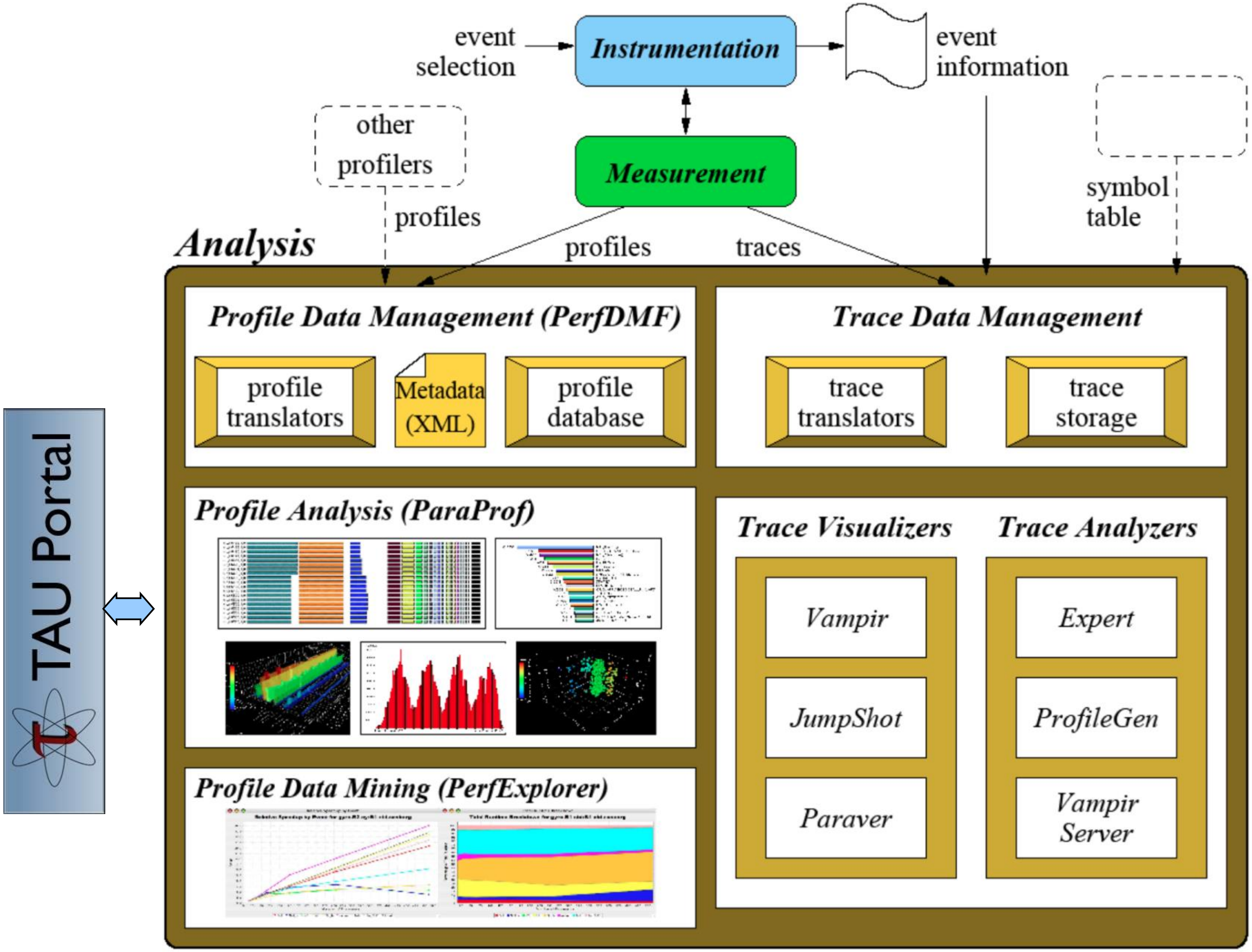
- Pebil:

```
% tau_pebil_rewrite -T [tags] a.out \
  -o a.inst
```

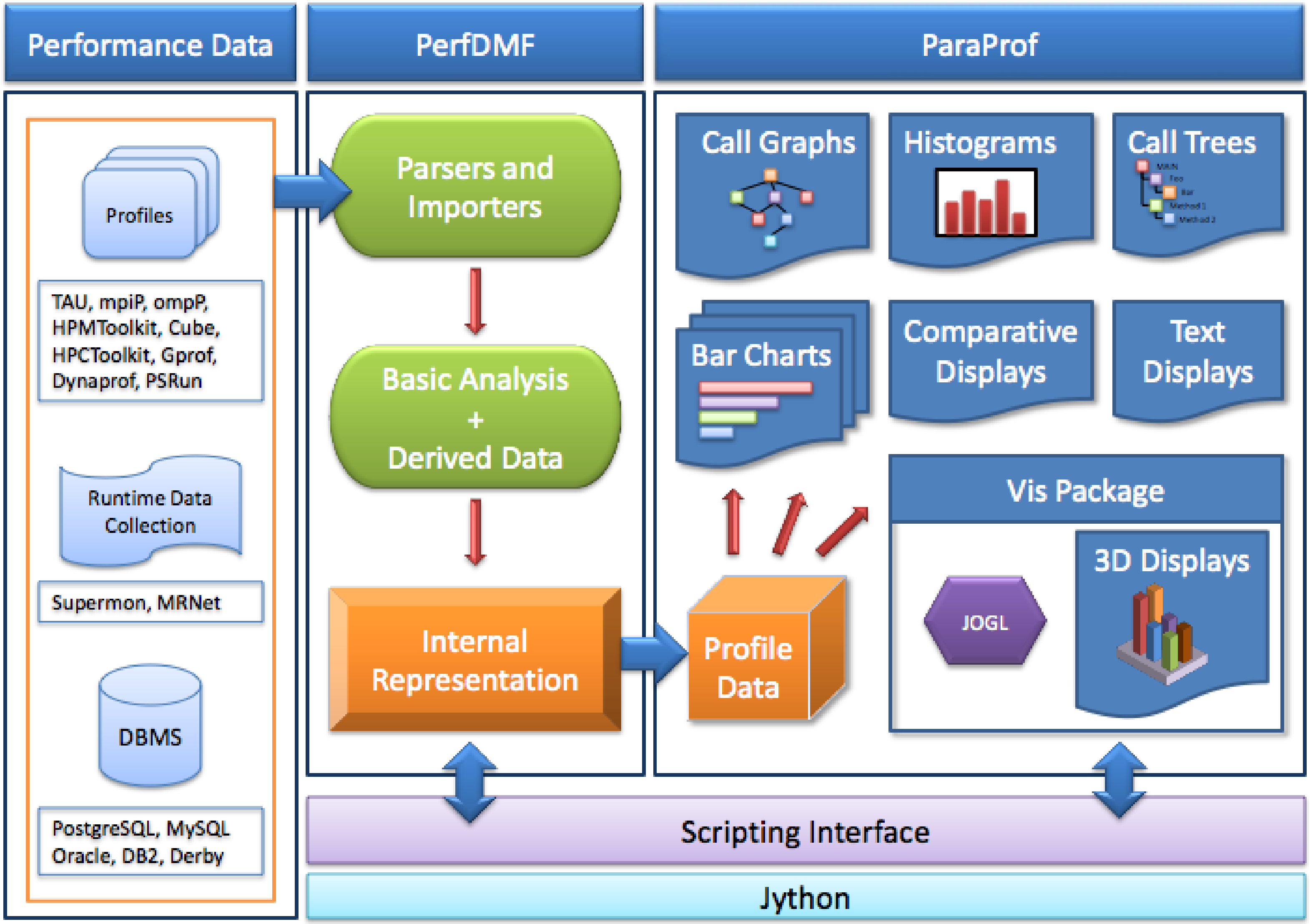
- Execute the application to get measurement data:

```
% mpirun -np 256 ./a.inst
```

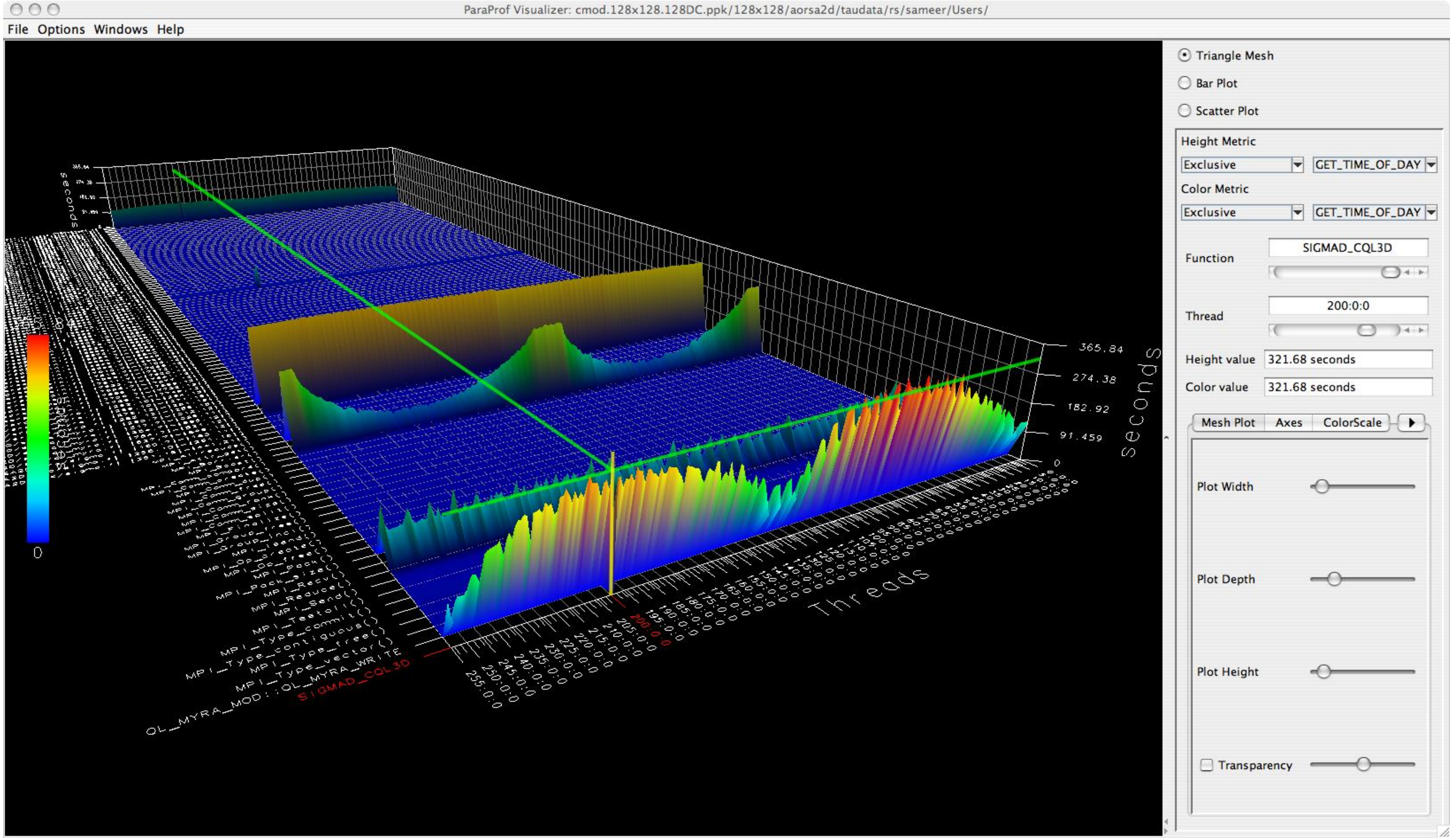
TAU Analysis



ParaProf Profile Analysis Framework



Parallel Profile Visualization: ParaProf



ParaProf Visualizer: cmod.128x128.128DC.ppk/128x128/aorsa2d/taudata/rs/sameer/Users/

● Triangle Mesh
○ Bar Plot
○ Scatter Plot

Height Metric: Exclusive — GET_TIME_OF_DAY

Color Metric: Exclusive — GET_TIME_OF_DAY

Function: SIGMAD_CQL3D

Thread: 200:0:0

Height value: 321.68 seconds

Color value: 321.68 seconds

Mesh Plot | Axes | ColorScale

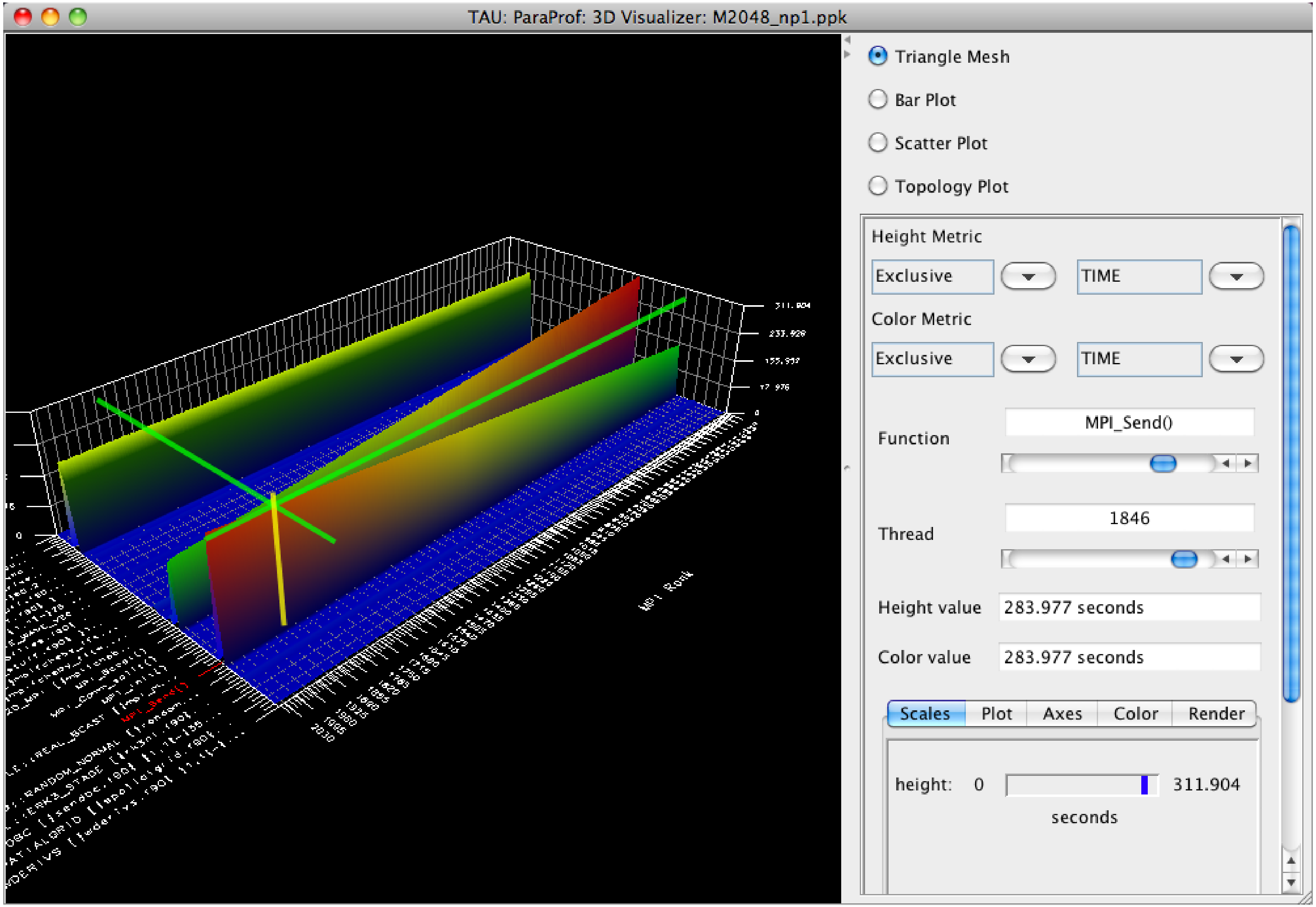
Plot Width

Plot Depth

Plot Height

□ Transparency

Parallel Profile Visualization: ParaProf



TAU: ParaProf: 3D Visualizer: M2048_np1.ppk

● Triangle Mesh

○ Bar Plot

○ Scatter Plot

○ Topology Plot

Height Metric: Exclusive, TIME

Color Metric: Exclusive, TIME

Function: MPI_Send()

Thread: 1846

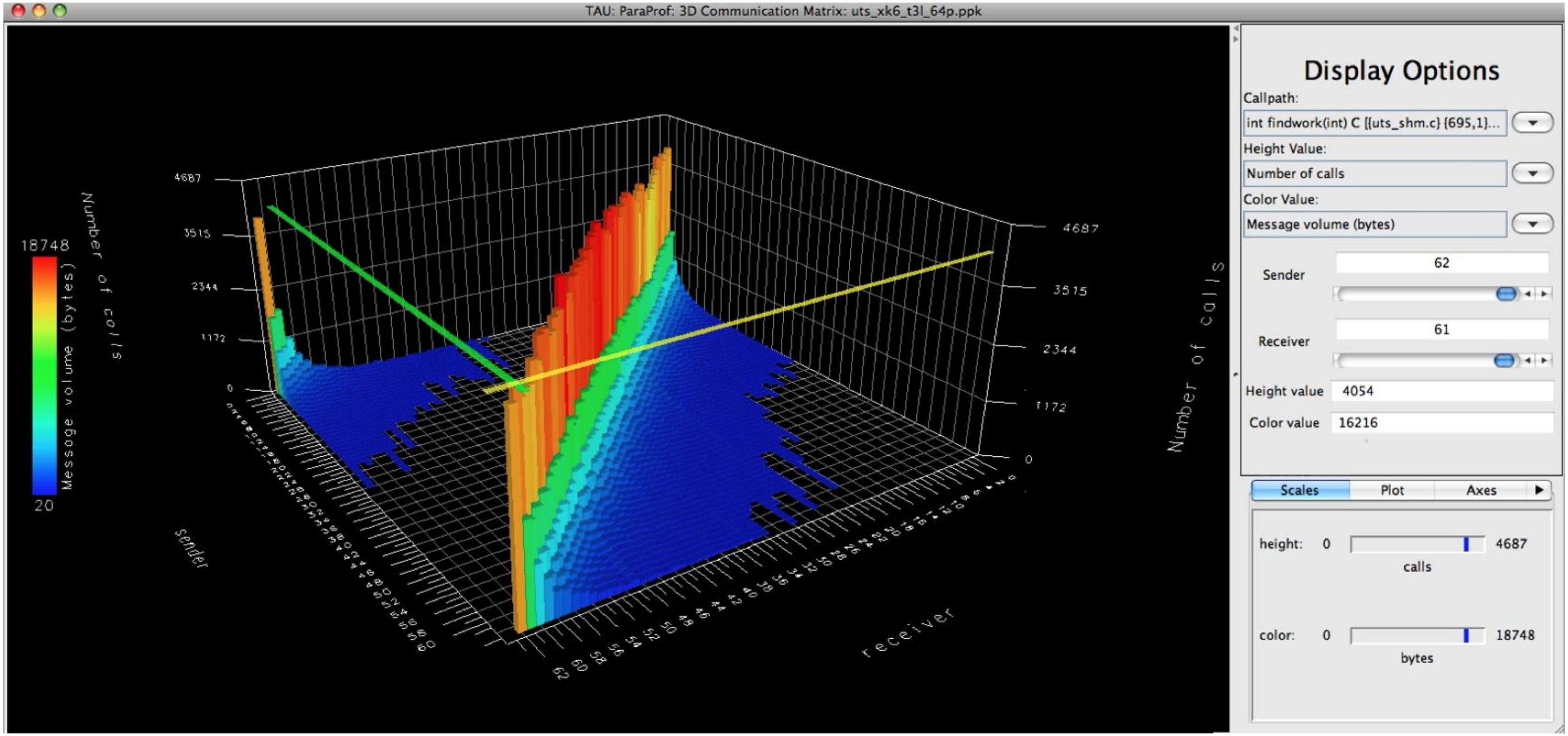
Height value: 283.977 seconds

Color value: 283.977 seconds

Scales | Plot | Axes | Color | Render

height: 0 — 311.904 seconds

ParaProf 3D Communication Matrix



TAU: ParaProf: 3D Communication Matrix: uts_xk6_t3l_64p.ppk

Display Options

Callpath: int findwork(int) C [{uts_shm.c} {695,1}...

Height Value: Number of calls

Color Value: Message volume (bytes)

Sender: 62

Receiver: 61

Height value: 4054

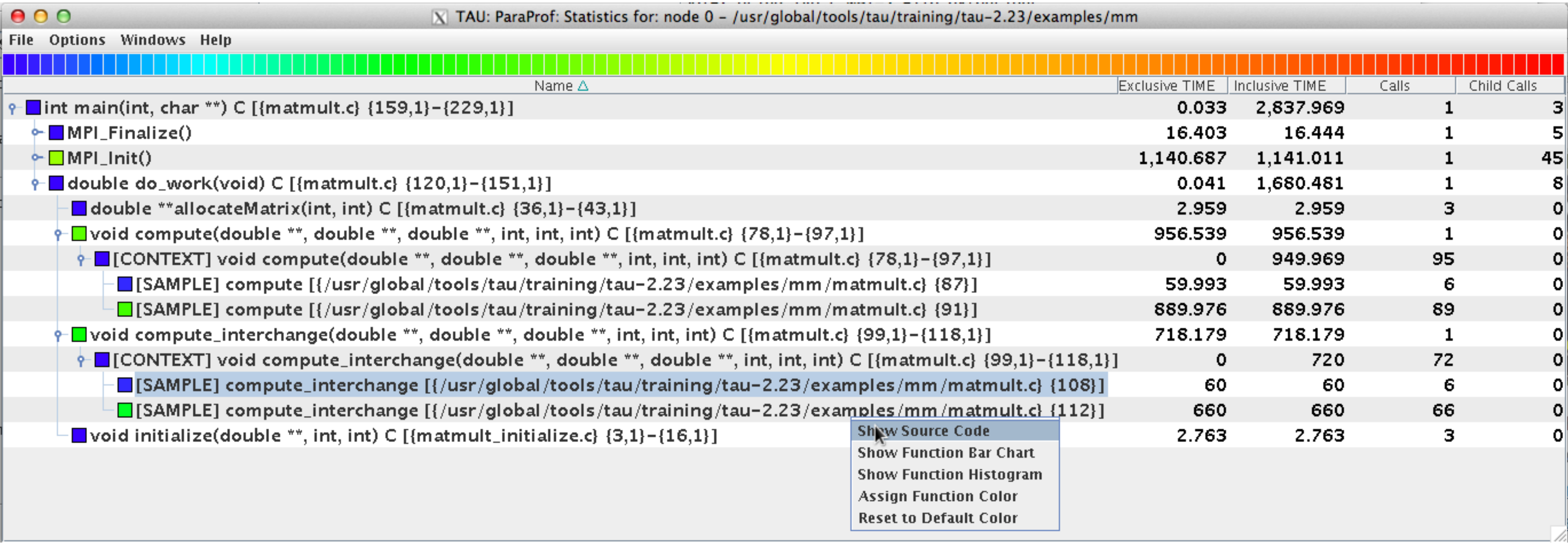
Color value: 16216

height: 0 – 4687 calls

color: 0 – 18748 bytes

% export TAU_COMM_MATRIX=1

Event Based Sampling in TAU

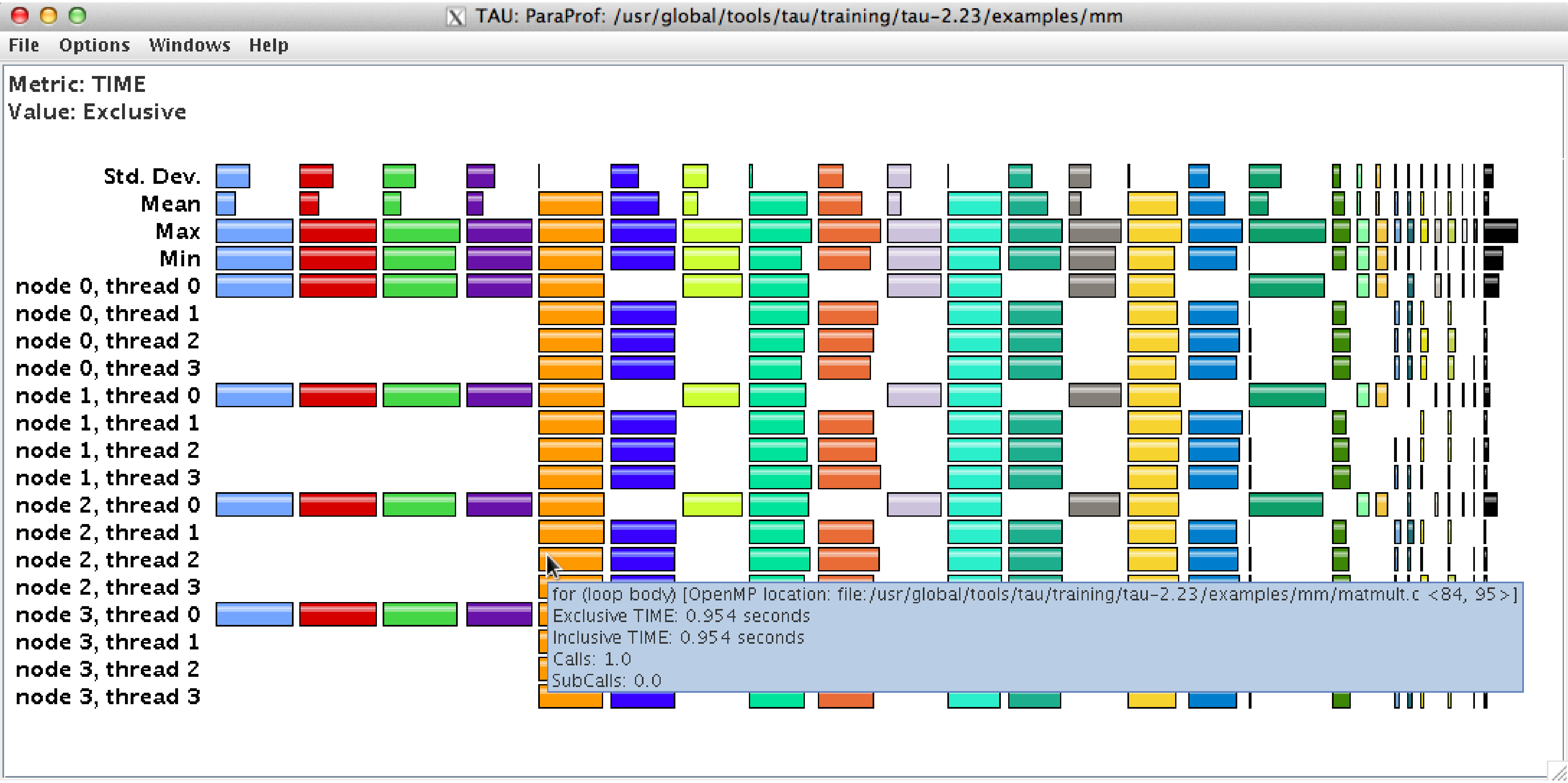


TAU: ParaProf: Statistics for: node 0 – /usr/global/tools/tau/training/tau-2.23/examples/mm

Name △	Exclusive TIME	Inclusive TIME	Calls	Child Calls
🟦 int main(int, char **) C [{matmult.c} {159,1}-{229,1}]	0.033	2,837.969	1	3
🟦 MPI_Finalize()	16.403	16.444	1	5
🟩 MPI_Init()	1,140.687	1,141.011	1	45
🟦 double do_work(void) C [{matmult.c} {120,1}-{151,1}]	0.041	1,680.481	1	8
🟦 double **allocateMatrix(int, int) C [{matmult.c} {36,1}-{43,1}]	2.959	2.959	3	0
🟩 void compute(double **, double **, double **, int, int, int) C [{matmult.c} {78,1}-{97,1}]	956.539	956.539	1	0
🟦 [CONTEXT] void compute(double **, double **, double **, int, int, int) C [{matmult.c} {78,1}-{97,1}]	0	949.969	95	0
🟦 [SAMPLE] compute [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {87}]	59.993	59.993	6	0
🟩 [SAMPLE] compute [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {91}]	889.976	889.976	89	0
🟩 void compute_interchange(double **, double **, double **, int, int, int) C [{matmult.c} {99,1}-{118,1}]	718.179	718.179	1	0
🟦 [CONTEXT] void compute_interchange(double **, double **, double **, int, int, int) C [{matmult.c} {99,1}-{118,1}]	0	720	72	0
🟦 [SAMPLE] compute_interchange [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {108}]	60	60	6	0
🟩 [SAMPLE] compute_interchange [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {112}]	660	660	66	0
🟦 void initialize(double **, int, int) C [{matmult_initialize.c} {3,1}-{16,1}]	2.763	2.763	3	0

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% make CC=tau_cc.sh CXX=tau_cxx.sh
% export TAU_SAMPLING=1
% mpirun –np 256 ./a.out
% paraprof
```

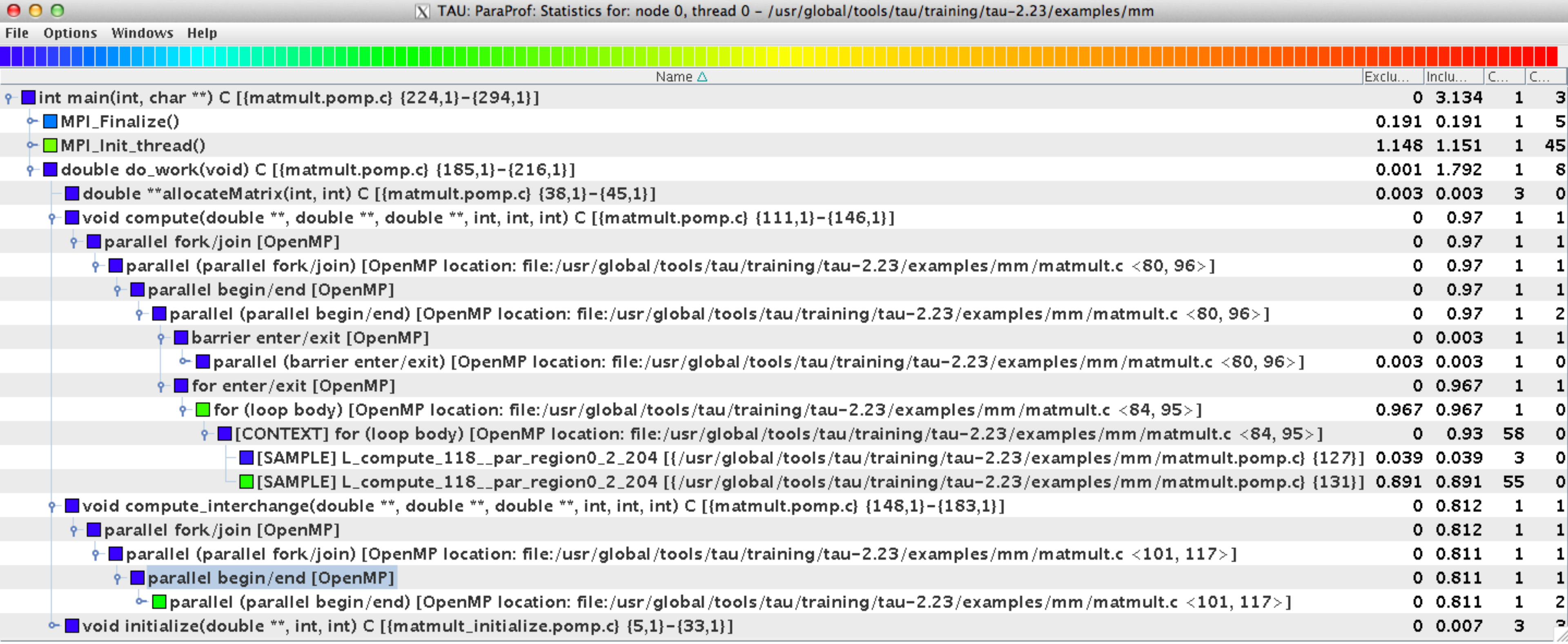
Mixed MPI and OpenMP Instrumentation



Metric: TIME
Value: Exclusive

for (loop body) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <84, 95>]
Exclusive TIME: 0.954 seconds
Inclusive TIME: 0.954 seconds
Calls: 1.0
SubCalls: 0.0

Opari OpenMP Instrumentation with Sampling

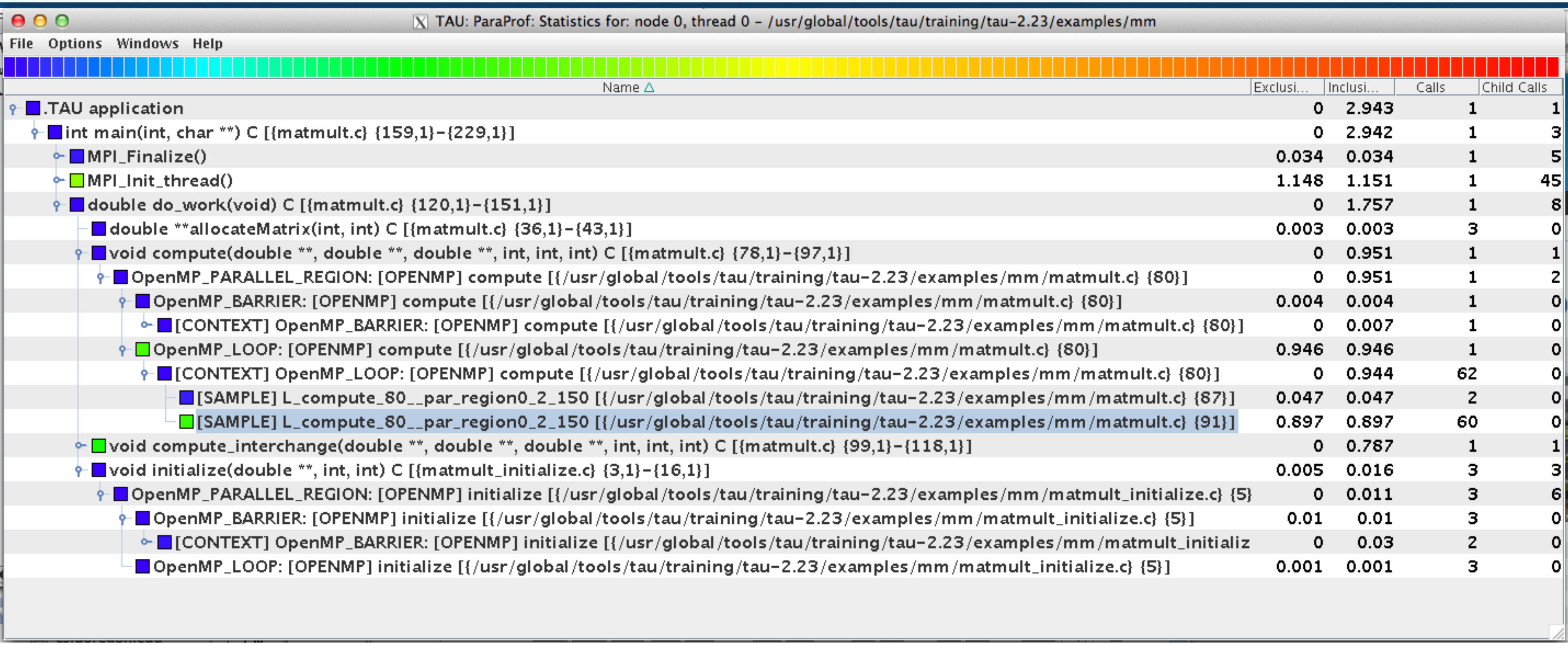


TAU: ParaProf: Statistics for: node 0, thread 0 – /usr/global/tools/tau/training/tau-2.23/examples/mm

Name △	Exclu...	Inclu...	C...	C...
int main(int, char **) C [{matmult.pomp.c} {224,1}-{294,1}]	0	3.134	1	3
MPI_Finalize()	0.191	0.191	1	5
MPI_Init_thread()	1.148	1.151	1	45
double do_work(void) C [{matmult.pomp.c} {185,1}-{216,1}]	0.001	1.792	1	8
double **allocateMatrix(int, int) C [{matmult.pomp.c} {38,1}-{45,1}]	0.003	0.003	3	0
void compute(double **, double **, double **, int, int, int) C [{matmult.pomp.c} {111,1}-{146,1}]	0	0.97	1	1
parallel fork/join [OpenMP]	0	0.97	1	1
parallel (parallel fork/join) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <80, 96>]	0	0.97	1	1
parallel begin/end [OpenMP]	0	0.97	1	1
parallel (parallel begin/end) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <80, 96>]	0	0.97	1	2
barrier enter/exit [OpenMP]	0	0.003	1	1
parallel (barrier enter/exit) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <80, 96>]	0.003	0.003	1	0
for enter/exit [OpenMP]	0	0.967	1	1
for (loop body) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <84, 95>]	0.967	0.967	1	0
[CONTEXT] for (loop body) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <84, 95>]	0	0.93	58	0
[SAMPLE] L_compute_118__par_region0_2_204 [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.pomp.c} {127}]	0.039	0.039	3	0
[SAMPLE] L_compute_118__par_region0_2_204 [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.pomp.c} {131}]	0.891	0.891	55	0
void compute_interchange(double **, double **, double **, int, int, int) C [{matmult.pomp.c} {148,1}-{183,1}]	0	0.812	1	1
parallel fork/join [OpenMP]	0	0.812	1	1
parallel (parallel fork/join) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <101, 117>]	0	0.811	1	1
parallel begin/end [OpenMP]	0	0.811	1	1
parallel (parallel begin/end) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <101, 117>]	0	0.811	1	2
void initialize(double **, int, int) C [{matmult_initialize.pomp.c} {5,1}-{33,1}]	0	0.007	3	3

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt-opari-openmp
% make CC=tau_cc.sh CXX=tau_cxx.sh
% export TAU_SAMPLING=1; export OMP_NUM_THREADS=16
% mpirun –np 256 ./a.out
% paraprof
```

TAU’s Support for Intel OMPT

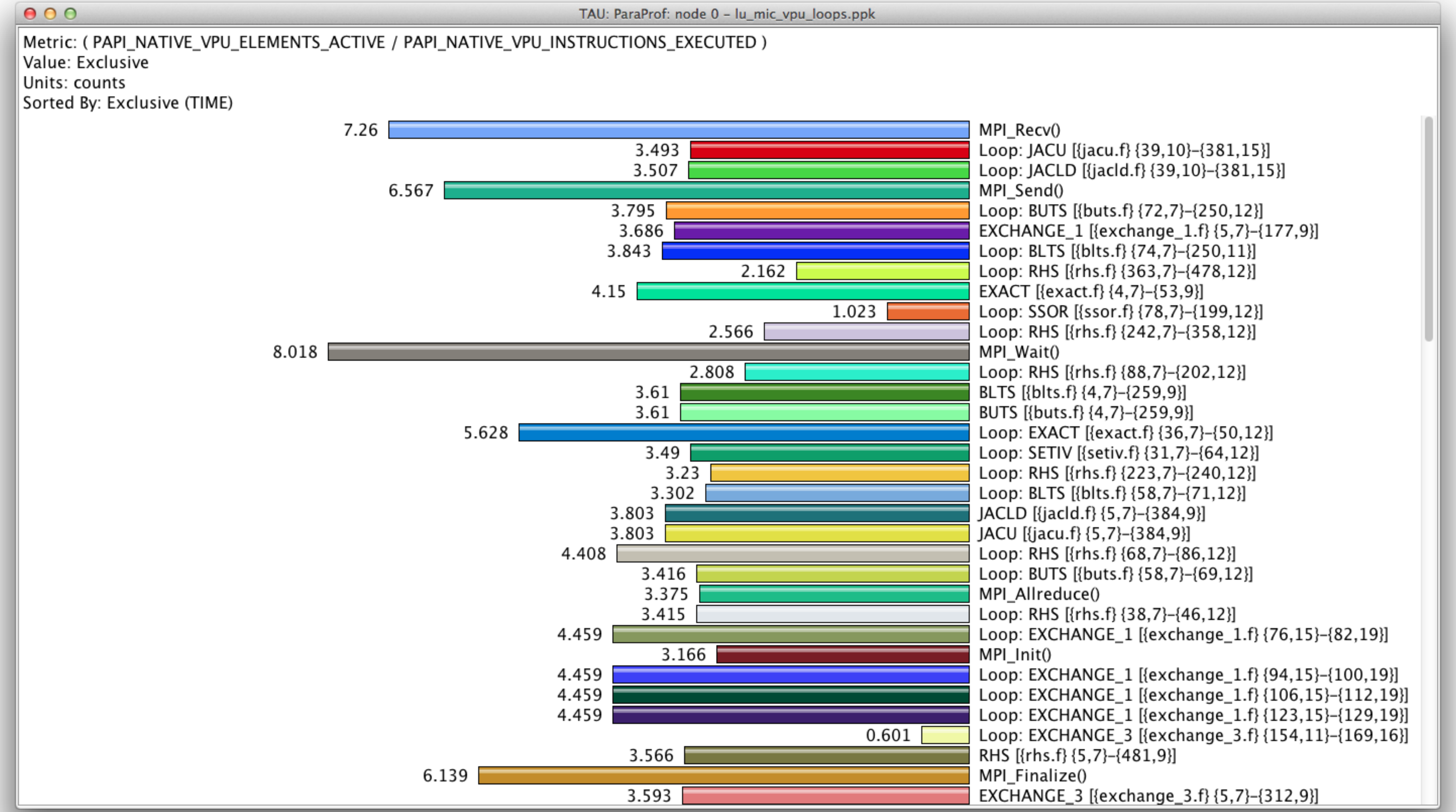


TAU: ParaProf: Statistics for: node 0, thread 0 – /usr/global/tools/tau/training/tau-2.23/examples/mm

Name △	Exclusi...	Inclusi...	Calls	Child Calls
🟦 .TAU application	0	2.943	1	1
🟦 int main(int, char **) C [{matmult.c} {159,1}-{229,1}]	0	2.942	1	3
🟦 MPI_Finalize()	0.034	0.034	1	5
🟩 MPI_Init_thread()	1.148	1.151	1	45
🟦 double do_work(void) C [{matmult.c} {120,1}-{151,1}]	0	1.757	1	8
🟦 double **allocateMatrix(int, int) C [{matmult.c} {36,1}-{43,1}]	0.003	0.003	3	0
🟦 void compute(double **, double **, double **, int, int, int) C [{matmult.c} {78,1}-{97,1}]	0	0.951	1	1
🟦 OpenMP_PARALLEL_REGION: [OPENMP] compute [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {80}]	0	0.951	1	2
🟦 OpenMP_BARRIER: [OPENMP] compute [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {80}]	0.004	0.004	1	0
🟦 [CONTEXT] OpenMP_BARRIER: [OPENMP] compute [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {80}]	0	0.007	1	0
🟩 OpenMP_LOOP: [OPENMP] compute [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {80}]	0.946	0.946	1	0
🟦 [CONTEXT] OpenMP_LOOP: [OPENMP] compute [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {80}]	0	0.944	62	0
🟦 [SAMPLE] L_compute_80__par_region0_2_150 [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {87}]	0.047	0.047	2	0
🟩 [SAMPLE] L_compute_80__par_region0_2_150 [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {91}]	0.897	0.897	60	0
🟩 void compute_interchange(double **, double **, double **, int, int, int) C [{matmult.c} {99,1}-{118,1}]	0	0.787	1	1
🟦 void initialize(double **, int, int) C [{matmult_initialize.c} {3,1}-{16,1}]	0.005	0.016	3	3
🟦 OpenMP_PARALLEL_REGION: [OPENMP] initialize [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult_initialize.c} {5}]	0	0.011	3	6
🟦 OpenMP_BARRIER: [OPENMP] initialize [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult_initialize.c} {5}]	0.01	0.01	3	0
🟦 [CONTEXT] OpenMP_BARRIER: [OPENMP] initialize [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult_initializ	0	0.03	2	0
🟦 OpenMP_LOOP: [OPENMP] initialize [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult_initialize.c} {5}]	0.001	0.001	3	0

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt-ompt-openmp
% make CC=tau_cc.sh CXX=tau_cxx.sh
% export TAU_SAMPLING=1; export OMP_NUM_THREADS=16
% mpirun –np 256 tau_exec –T ompt –loadlib=$TAU/libiomp5.so  ./a.out
% paraprof
```

ParaProf Derived Metric Window: Intel MIC



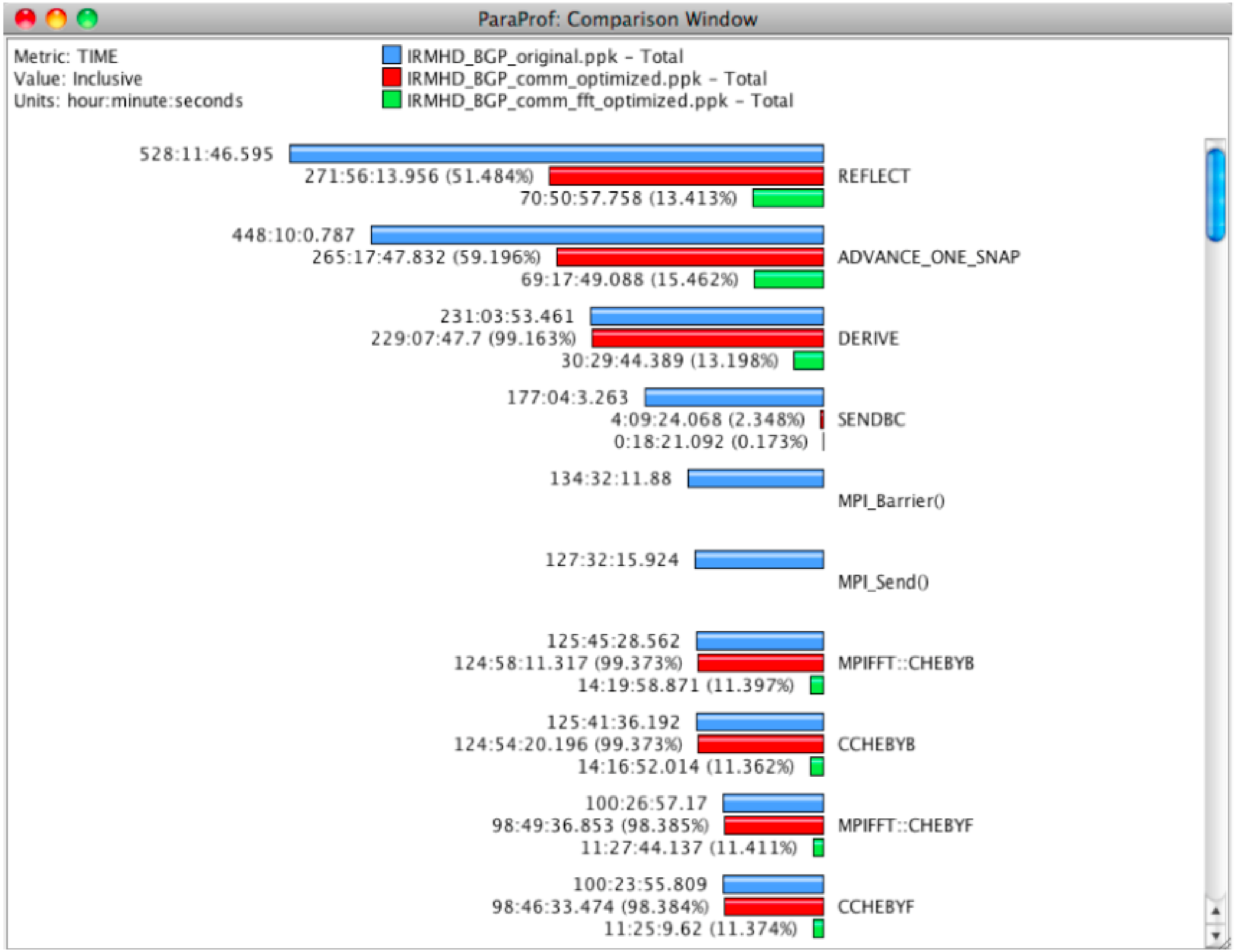
TAU: ParaProf: node 0 – lu_mic_vpu_loops.ppk

Metric: (PAPI_NATIVE_VPU_ELEMENTS_ACTIVE / PAPI_NATIVE_VPU_INSTRUCTIONS_EXECUTED)
Value: Exclusive
Units: counts
Sorted By: Exclusive (TIME)

7.26 MPI_Recv()
3.493 Loop: JACU [{jacu.f} {39,10}-{381,15}]
3.507 Loop: JACLD [{jacld.f} {39,10}-{381,15}]
6.567 MPI_Send()
3.795 Loop: BUTS [{buts.f} {72,7}-{250,12}]
3.686 EXCHANGE_1 [{exchange_1.f} {5,7}-{177,9}]
3.843 Loop: BLTS [{blts.f} {74,7}-{250,11}]
2.162 Loop: RHS [{rhs.f} {363,7}-{478,12}]
4.15 EXACT [{exact.f} {4,7}-{53,9}]
1.023 Loop: SSOR [{ssor.f} {78,7}-{199,12}]
2.566 Loop: RHS [{rhs.f} {242,7}-{358,12}]
8.018 MPI_Wait()
2.808 Loop: RHS [{rhs.f} {88,7}-{202,12}]
3.61 BLTS [{blts.f} {4,7}-{259,9}]
3.61 BUTS [{buts.f} {4,7}-{259,9}]
5.628 Loop: EXACT [{exact.f} {36,7}-{50,12}]
3.49 Loop: SETIV [{setiv.f} {31,7}-{64,12}]
3.23 Loop: RHS [{rhs.f} {223,7}-{240,12}]
3.302 Loop: BLTS [{blts.f} {58,7}-{71,12}]
3.803 JACLD [{jacld.f} {5,7}-{384,9}]
3.803 JACU [{jacu.f} {5,7}-{384,9}]
4.408 Loop: RHS [{rhs.f} {68,7}-{86,12}]
3.416 Loop: BUTS [{buts.f} {58,7}-{69,12}]
3.375 MPI_Allreduce()
3.415 Loop: RHS [{rhs.f} {38,7}-{46,12}]
4.459 Loop: EXCHANGE_1 [{exchange_1.f} {76,15}-{82,19}]
3.166 MPI_Init()
4.459 Loop: EXCHANGE_1 [{exchange_1.f} {94,15}-{100,19}]
4.459 Loop: EXCHANGE_1 [{exchange_1.f} {106,15}-{112,19}]
4.459 Loop: EXCHANGE_1 [{exchange_1.f} {123,15}-{129,19}]
0.601 Loop: EXCHANGE_3 [{exchange_3.f} {154,11}-{169,16}]
3.566 RHS [{rhs.f} {5,7}-{481,9}]
6.139 MPI_Finalize()
3.593 EXCHANGE_3 [{exchange_3.f} {5,7}-{312,9}]

```
% export TAU_MAKEFILE=$TAUROOT/mic_linux/lib/Makefile.tau-icpc-papi-mpi-pdt
% export TAU_METRICS=TIME,PAPI_NATIVE_VPU_ELEMENTS_ACTIVE,
                              PAPI_NATIVE_VPU_INSTRUCTIONS_EXECUTED
```

ParaProf Comparison Window



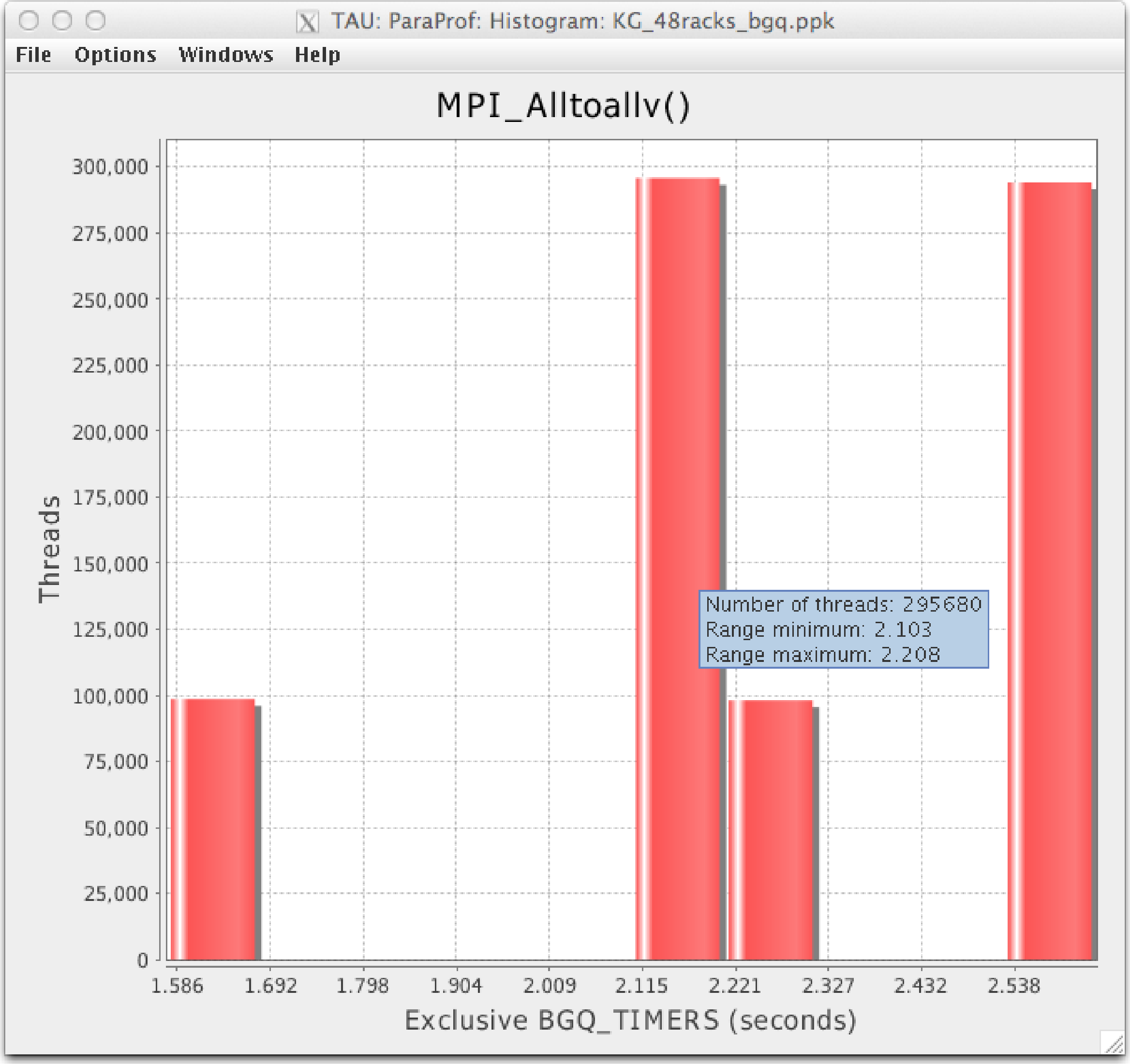
ParaProf: Comparison Window

Metric: TIME
Value: Inclusive
Units: hour:minute:seconds

🟦 IRMHD_BGP_original.ppk – Total
🟥 IRMHD_BGP_comm_optimized.ppk – Total
🟩 IRMHD_BGP_comm_fft_optimized.ppk – Total

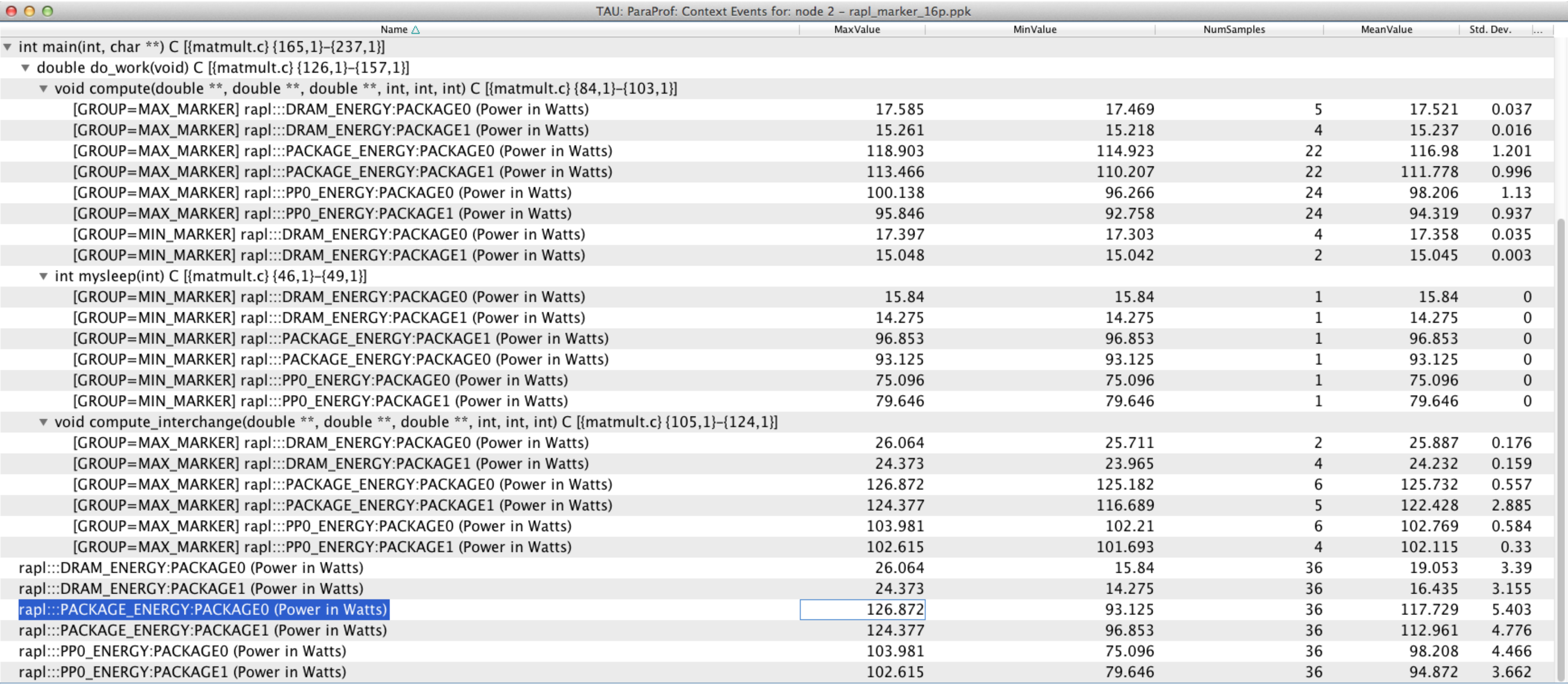
```
528:11:46.595
271:56:13.956 (51.484%)    REFLECT
70:50:57.758 (13.413%)

448:10:0.787
265:17:47.832 (59.196%)    ADVANCE_ONE_SNAP
69:17:49.088 (15.462%)

231:03:53.461
229:07:47.7 (99.163%)      DERIVE
30:29:44.389 (13.198%)

177:04:3.263
4:09:24.068 (2.348%)       SENDBC
0:18:21.092 (0.173%)

134:32:11.88
                           MPI_Barrier()

127:32:15.924
                           MPI_Send()

125:45:28.562
124:58:11.317 (99.373%)    MPIFFT::CHEBYB
14:19:58.871 (11.397%)

125:41:36.192
124:54:20.196 (99.373%)    CCHEBYB
14:16:52.014 (11.362%)

100:26:57.17
98:49:36.853 (98.385%)     MPIFFT::CHEBYF
11:27:44.137 (11.411%)

100:23:55.809
98:46:33.474 (98.384%)     CCHEBYF
11:25:9.62 (11.374%)
```

ParaProf Histogram Window



Number of threads: 295680
Range minimum: 2.103
Range maximum: 2.208

Marker Events in TAU Show Sudden Spikes

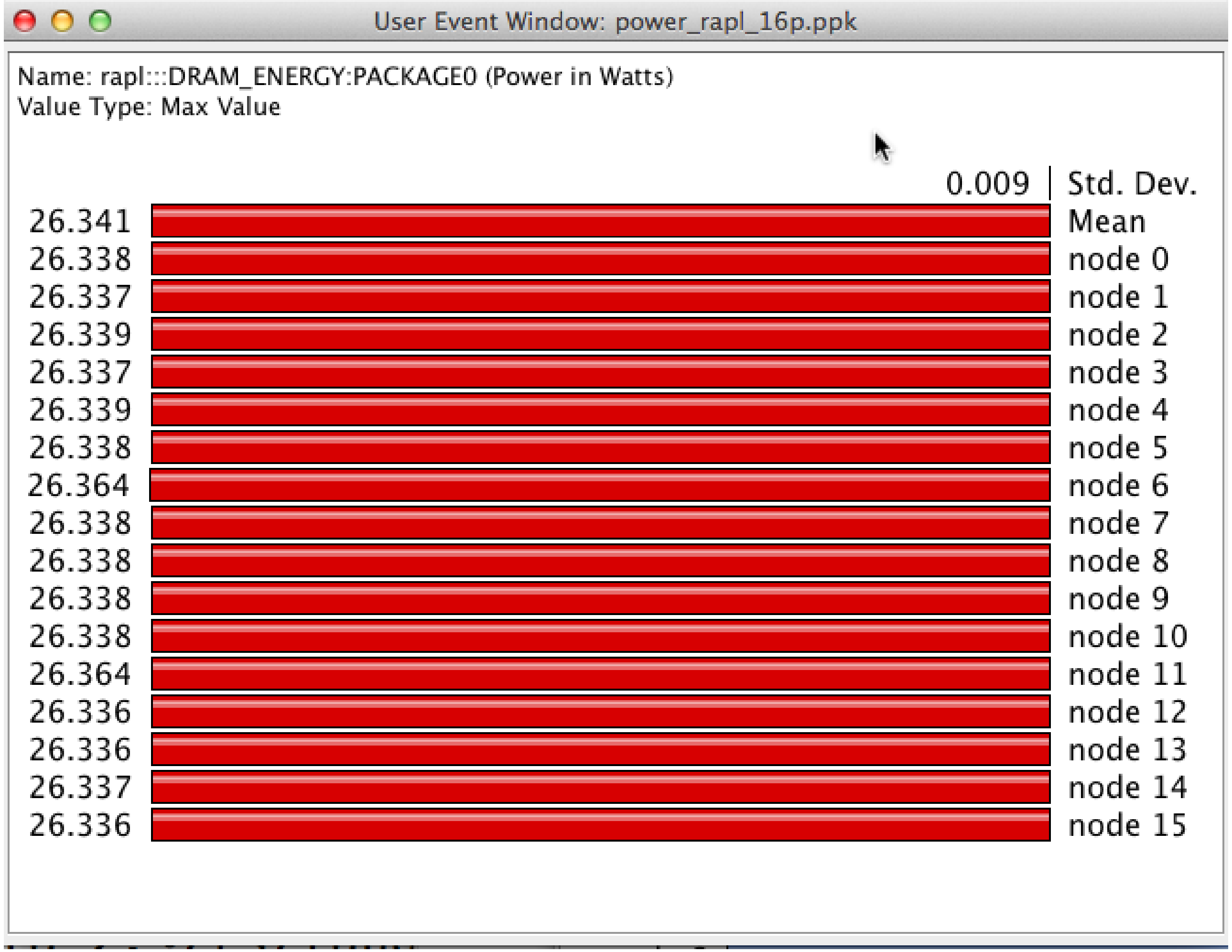


TAU: ParaProf: Context Events for: node 2 – rapl_marker_16p.ppk

Name △	MaxValue	MinValue	NumSamples	MeanValue	Std. Dev.
▼ int main(int, char **) C [{matmult.c} {165,1}-{237,1}]					
▼ double do_work(void) C [{matmult.c} {126,1}-{157,1}]					
▼ void compute(double **, double **, double **, int, int, int) C [{matmult.c} {84,1}-{103,1}]					
[GROUP=MAX_MARKER] rapl:::DRAM_ENERGY:PACKAGE0 (Power in Watts)	17.585	17.469	5	17.521	0.037
[GROUP=MAX_MARKER] rapl:::DRAM_ENERGY:PACKAGE1 (Power in Watts)	15.261	15.218	4	15.237	0.016
[GROUP=MAX_MARKER] rapl:::PACKAGE_ENERGY:PACKAGE0 (Power in Watts)	118.903	114.923	22	116.98	1.201
[GROUP=MAX_MARKER] rapl:::PACKAGE_ENERGY:PACKAGE1 (Power in Watts)	113.466	110.207	22	111.778	0.996
[GROUP=MAX_MARKER] rapl:::PP0_ENERGY:PACKAGE0 (Power in Watts)	100.138	96.266	24	98.206	1.13
[GROUP=MAX_MARKER] rapl:::PP0_ENERGY:PACKAGE1 (Power in Watts)	95.846	92.758	24	94.319	0.937
[GROUP=MIN_MARKER] rapl:::DRAM_ENERGY:PACKAGE0 (Power in Watts)	17.397	17.303	4	17.358	0.035
[GROUP=MIN_MARKER] rapl:::DRAM_ENERGY:PACKAGE1 (Power in Watts)	15.048	15.042	2	15.045	0.003
▼ int mysleep(int) C [{matmult.c} {46,1}-{49,1}]					
[GROUP=MIN_MARKER] rapl:::DRAM_ENERGY:PACKAGE0 (Power in Watts)	15.84	15.84	1	15.84	0
[GROUP=MIN_MARKER] rapl:::DRAM_ENERGY:PACKAGE1 (Power in Watts)	14.275	14.275	1	14.275	0
[GROUP=MIN_MARKER] rapl:::PACKAGE_ENERGY:PACKAGE1 (Power in Watts)	96.853	96.853	1	96.853	0
[GROUP=MIN_MARKER] rapl:::PACKAGE_ENERGY:PACKAGE0 (Power in Watts)	93.125	93.125	1	93.125	0
[GROUP=MIN_MARKER] rapl:::PP0_ENERGY:PACKAGE0 (Power in Watts)	75.096	75.096	1	75.096	0
[GROUP=MIN_MARKER] rapl:::PP0_ENERGY:PACKAGE1 (Power in Watts)	79.646	79.646	1	79.646	0
▼ void compute_interchange(double **, double **, double **, int, int, int) C [{matmult.c} {105,1}-{124,1}]					
[GROUP=MAX_MARKER] rapl:::DRAM_ENERGY:PACKAGE0 (Power in Watts)	26.064	25.711	2	25.887	0.176
[GROUP=MAX_MARKER] rapl:::DRAM_ENERGY:PACKAGE1 (Power in Watts)	24.373	23.965	4	24.232	0.159
[GROUP=MAX_MARKER] rapl:::PACKAGE_ENERGY:PACKAGE0 (Power in Watts)	126.872	125.182	6	125.732	0.557
[GROUP=MAX_MARKER] rapl:::PACKAGE_ENERGY:PACKAGE1 (Power in Watts)	124.377	116.689	5	122.428	2.885
[GROUP=MAX_MARKER] rapl:::PP0_ENERGY:PACKAGE0 (Power in Watts)	103.981	102.21	6	102.769	0.584
[GROUP=MAX_MARKER] rapl:::PP0_ENERGY:PACKAGE1 (Power in Watts)	102.615	101.693	4	102.115	0.33
rapl:::DRAM_ENERGY:PACKAGE0 (Power in Watts)	26.064	15.84	36	19.053	3.39
rapl:::DRAM_ENERGY:PACKAGE1 (Power in Watts)	24.373	14.275	36	16.435	3.155
rapl:::PACKAGE_ENERGY:PACKAGE0 (Power in Watts)	126.872	93.125	36	117.729	5.403
rapl:::PACKAGE_ENERGY:PACKAGE1 (Power in Watts)	124.377	96.853	36	112.961	4.776
rapl:::PP0_ENERGY:PACKAGE0 (Power in Watts)	103.981	75.096	36	98.208	4.466
rapl:::PP0_ENERGY:PACKAGE1 (Power in Watts)	102.615	79.646	36	94.872	3.662

```
% export TAU_EVENT_THRESHOLD 0.5
```

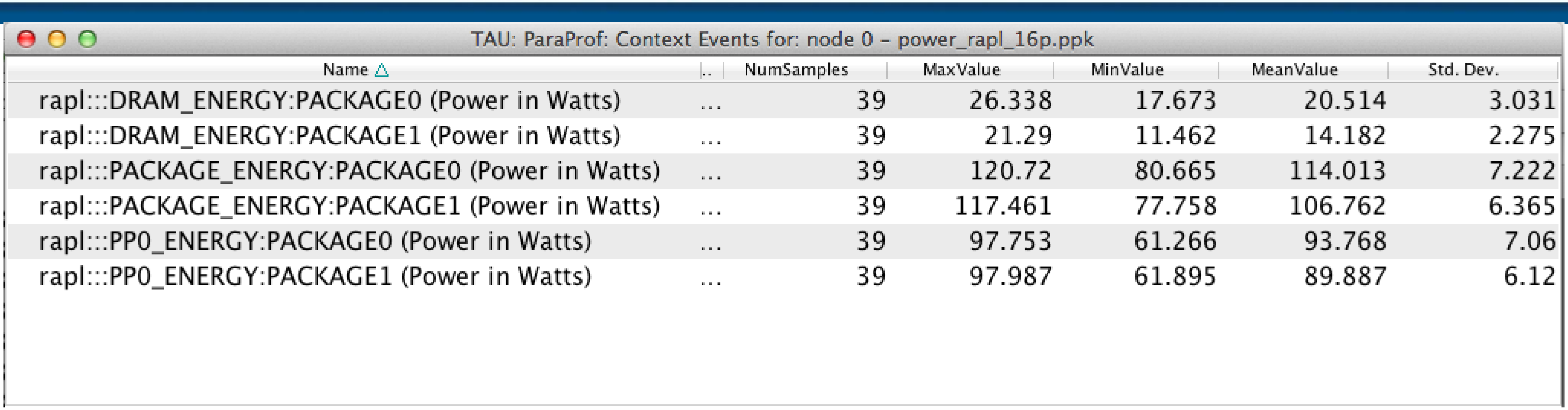
Energy Profiling in TAU



Name: rapl:::DRAM_ENERGY:PACKAGE0 (Power in Watts)
Value Type: Max Value

0.009	Std. Dev.
26.341	Mean
26.338	node 0
26.337	node 1
26.339	node 2
26.337	node 3
26.339	node 4
26.338	node 5
26.364	node 6
26.338	node 7
26.338	node 8
26.338	node 9
26.338	node 10
26.364	node 11
26.336	node 12
26.336	node 13
26.337	node 14
26.336	node 15

Profiling Power Using TAU with PAPI and RAPL



TAU: ParaProf: Context Events for: node 0 – power_rapl_16p.ppk

Name △	..	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
rapl:::DRAM_ENERGY:PACKAGE0 (Power in Watts)	...	39	26.338	17.673	20.514	3.031
rapl:::DRAM_ENERGY:PACKAGE1 (Power in Watts)	...	39	21.29	11.462	14.182	2.275
rapl:::PACKAGE_ENERGY:PACKAGE0 (Power in Watts)	...	39	120.72	80.665	114.013	7.222
rapl:::PACKAGE_ENERGY:PACKAGE1 (Power in Watts)	...	39	117.461	77.758	106.762	6.365
rapl:::PP0_ENERGY:PACKAGE0 (Power in Watts)	...	39	97.753	61.266	93.768	7.06
rapl:::PP0_ENERGY:PACKAGE1 (Power in Watts)	...	39	97.987	61.895	89.887	6.12

```
#include <TAU.h>

TAU_TRACK_POWER();   // In Fortran: call TAU_TRACK_POWER()

% sudo chmod –R go+r /dev/cpu/*/msr
% sudo /sbin/setcap cap_sys_rawio=ep ./a.out
% unset LD_LIBRARY_PATH
% ldd ./a.out
should have no “not found” entries, Use –Wl,-rpath,/path while linking
% ./a.out
% paraprof
```

Hands-on: Profile Report Extraction

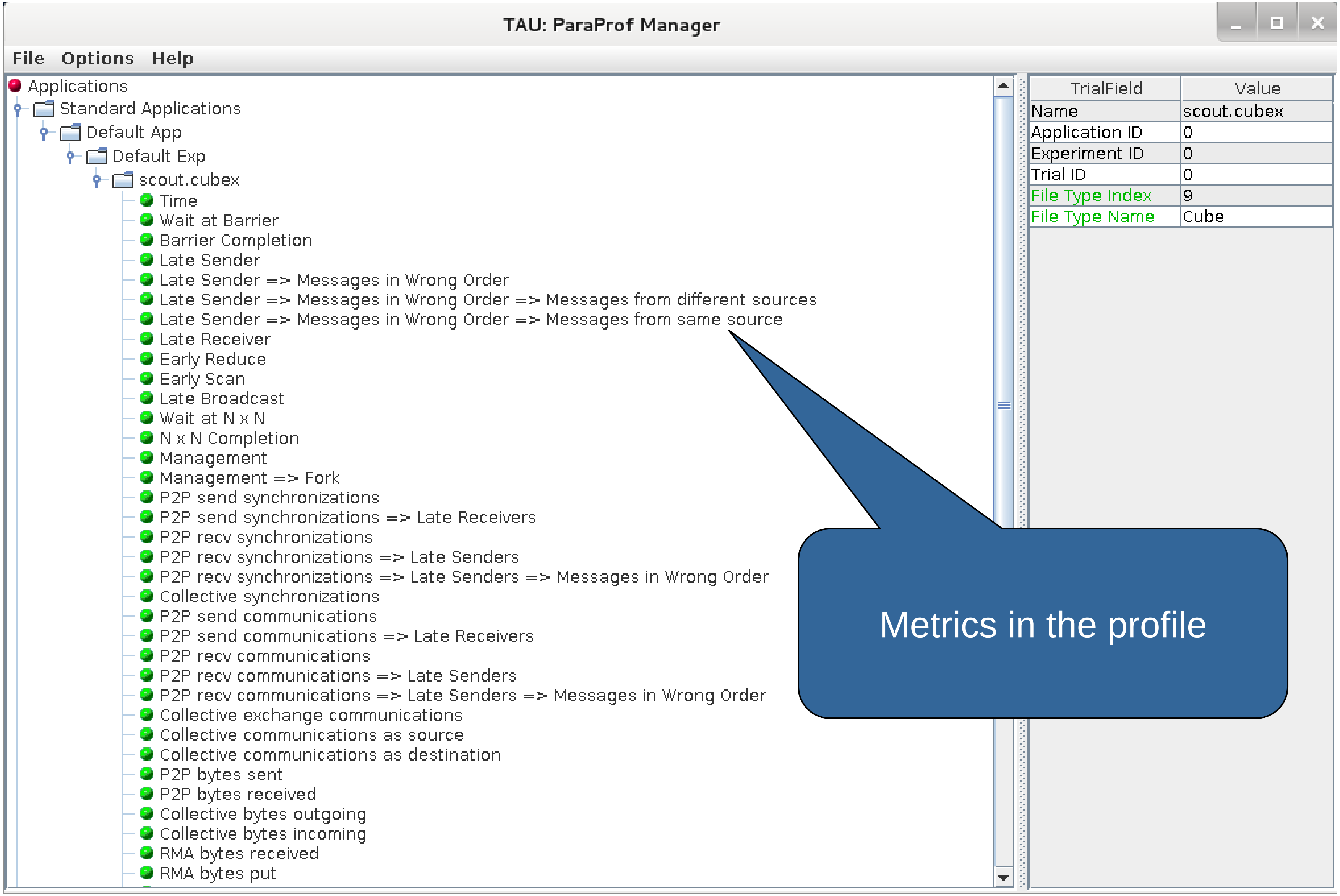
- The Tutorial contains Score-P experiments of BT-MZ
  - class “B“, 4 processes with 4 OpenMP threads each
  - collected on a dedicated node of the SuperMUC HPC system at Leibniz Rechenzentrum (LRZ), Munich, Germany

```
% cd
% ls
periscope-1.5                         scorep_bt-mz_B_4x4_sum
README                                scorep_bt-mz_B_4x4_sum+mets
run.out                               scorep_bt-mz_B_4x4_trace
scorep-20120913_1740_557443655223384
```

- Start TAU‘s paraprof GUI with default profile report

```
% paraprof scorep-20120913_1740_557443655223384/profile.cubex
OR
% paraprof scorep_bt-mz_B_4x4_trace/scout.cubex
```

ParaProf Manager Widow: scout.cubex

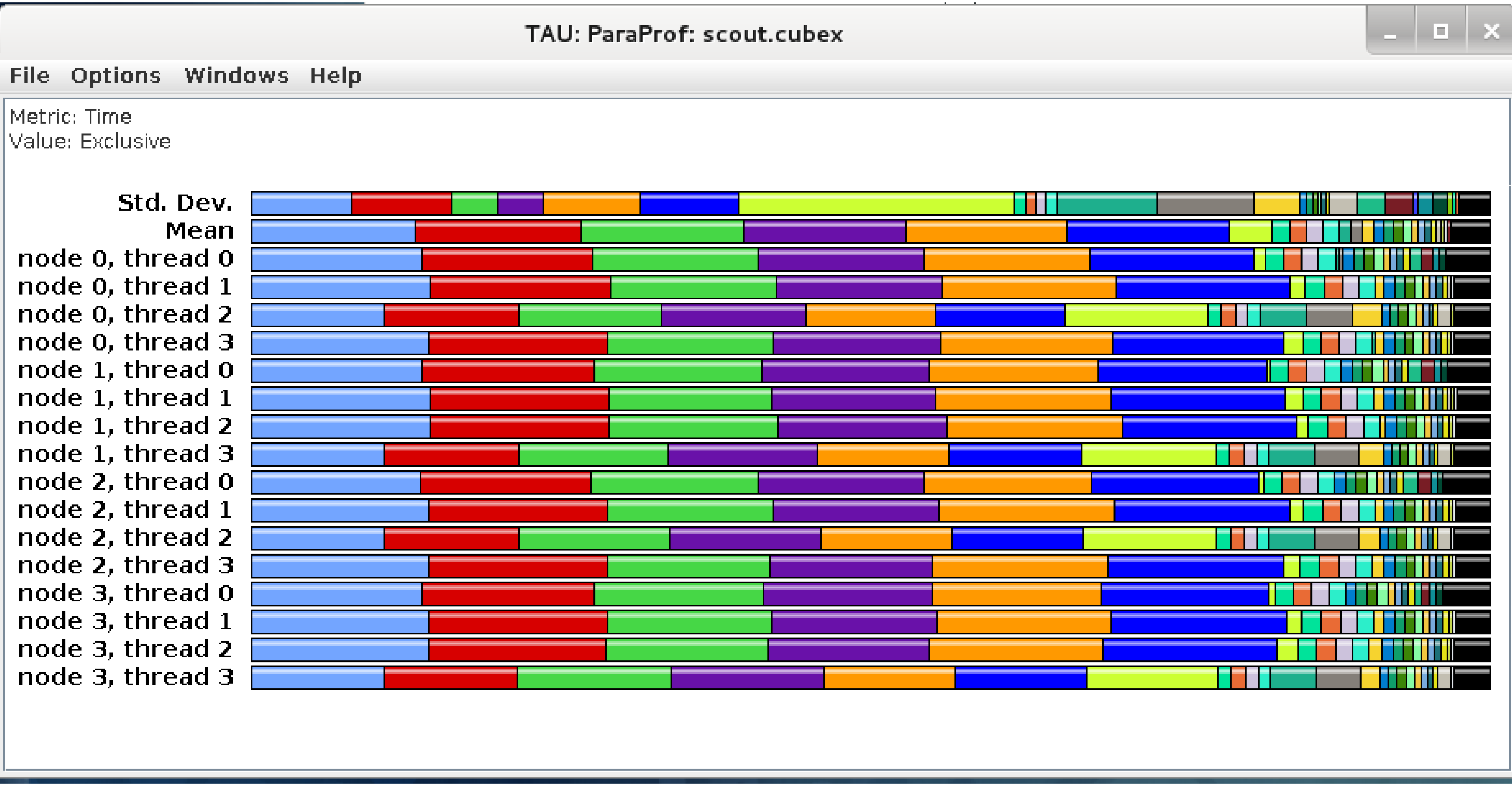


- Applications
  - Standard Applications
    - Default App
      - Default Exp
        - scout.cubex
          - Time
          - Wait at Barrier
          - Barrier Completion
          - Late Sender
          - Late Sender => Messages in Wrong Order
          - Late Sender => Messages in Wrong Order => Messages from different sources
          - Late Sender => Messages in Wrong Order => Messages from same source
          - Late Receiver
          - Early Reduce
          - Early Scan
          - Late Broadcast
          - Wait at N x N
          - N x N Completion
          - Management
          - Management => Fork
          - P2P send synchronizations
          - P2P send synchronizations => Late Receivers
          - P2P recv synchronizations
          - P2P recv synchronizations => Late Senders
          - P2P recv synchronizations => Late Senders => Messages in Wrong Order
          - Collective synchronizations
          - P2P send communications
          - P2P send communications => Late Receivers
          - P2P recv communications
          - P2P recv communications => Late Senders
          - P2P recv communications => Late Senders => Messages in Wrong Order
          - Collective exchange communications
          - Collective communications as source
          - Collective communications as destination
          - P2P bytes sent
          - P2P bytes received
          - Collective bytes outgoing
          - Collective bytes incoming
          - RMA bytes received
          - RMA bytes put

TrialField	Value
Name	scout.cubex
Application ID	0
Experiment ID	0
Trial ID	0
File Type Index	9
File Type Name	Cube

Metrics in the profile

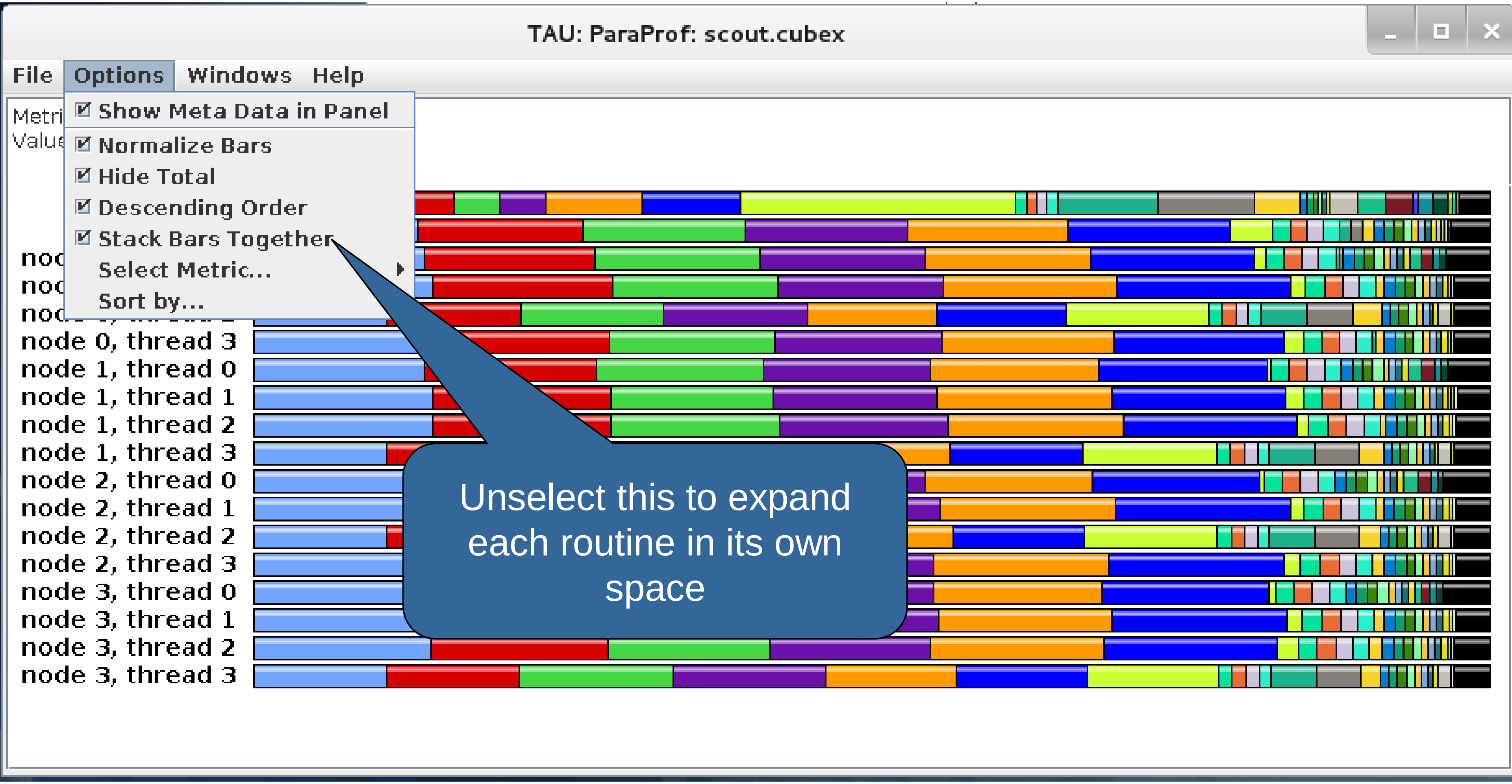
ParaProf: Main Window



TAU: ParaProf: scout.cubex

Metric: Time
Value: Exclusive

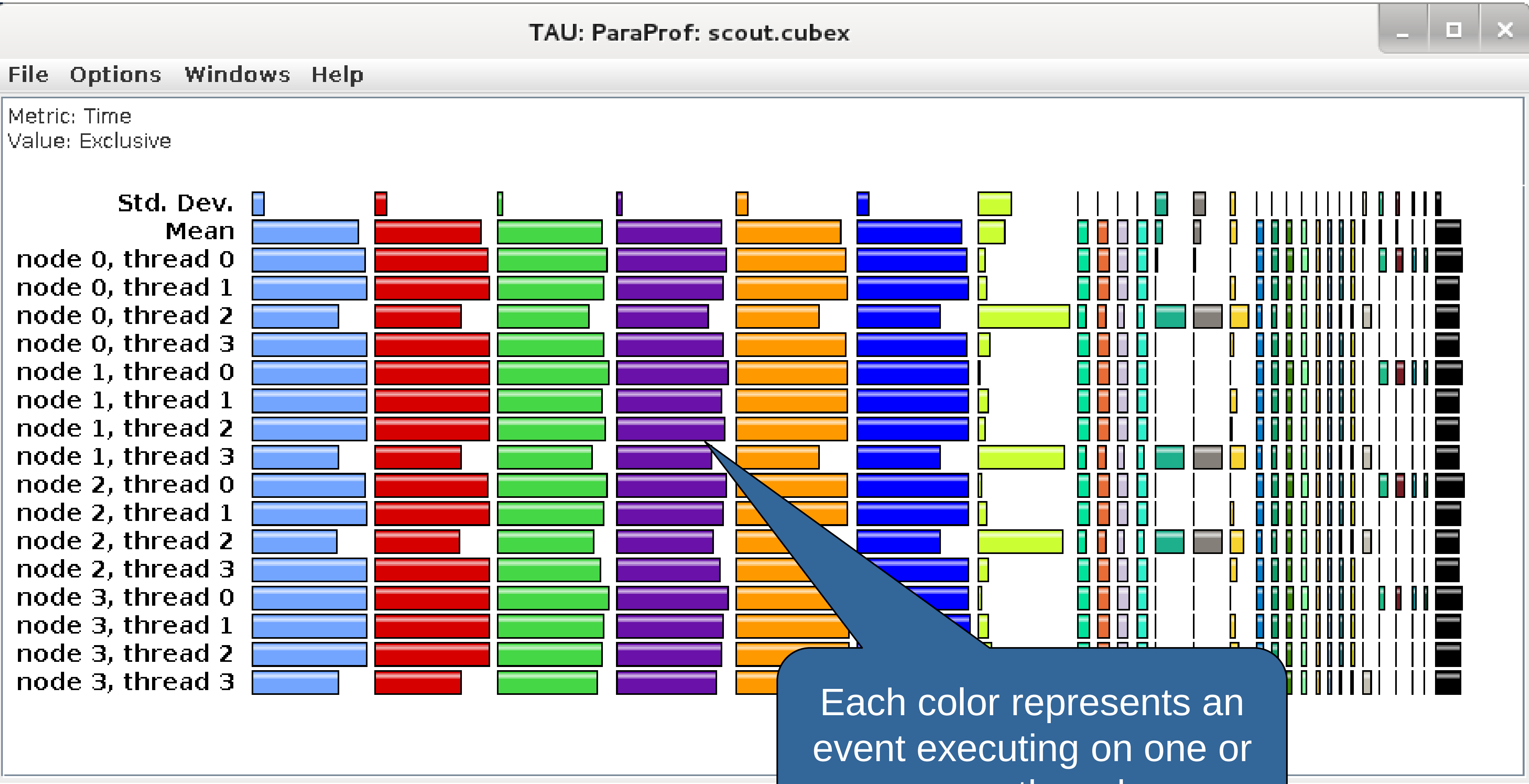
ParaProf: Options



TAU: ParaProf: scout.cubex

Unselect this to expand each routine in its own space

ParaProf: Unstack Bar Charts

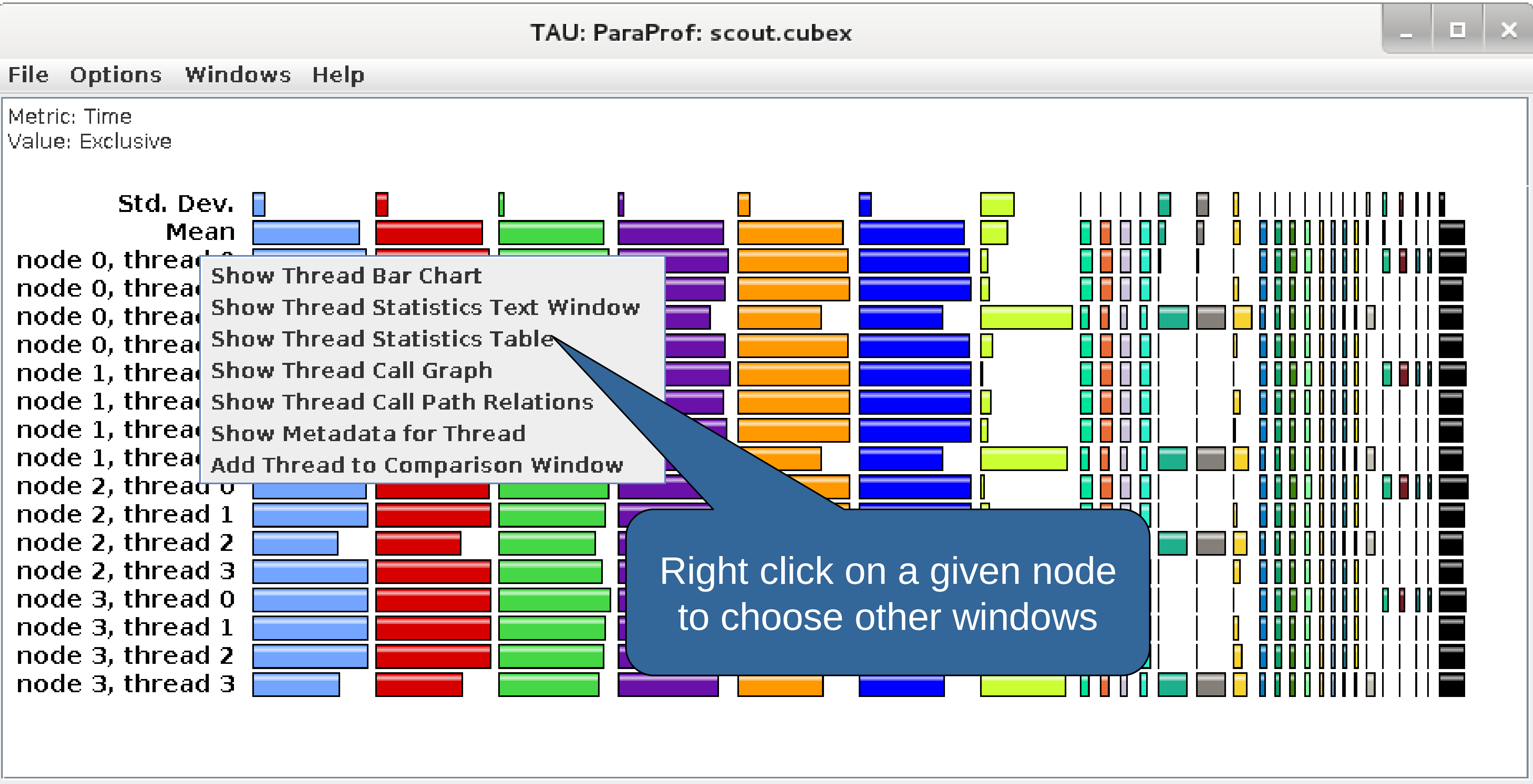


TAU: ParaProf: scout.cubex

Metric: Time
Value: Exclusive

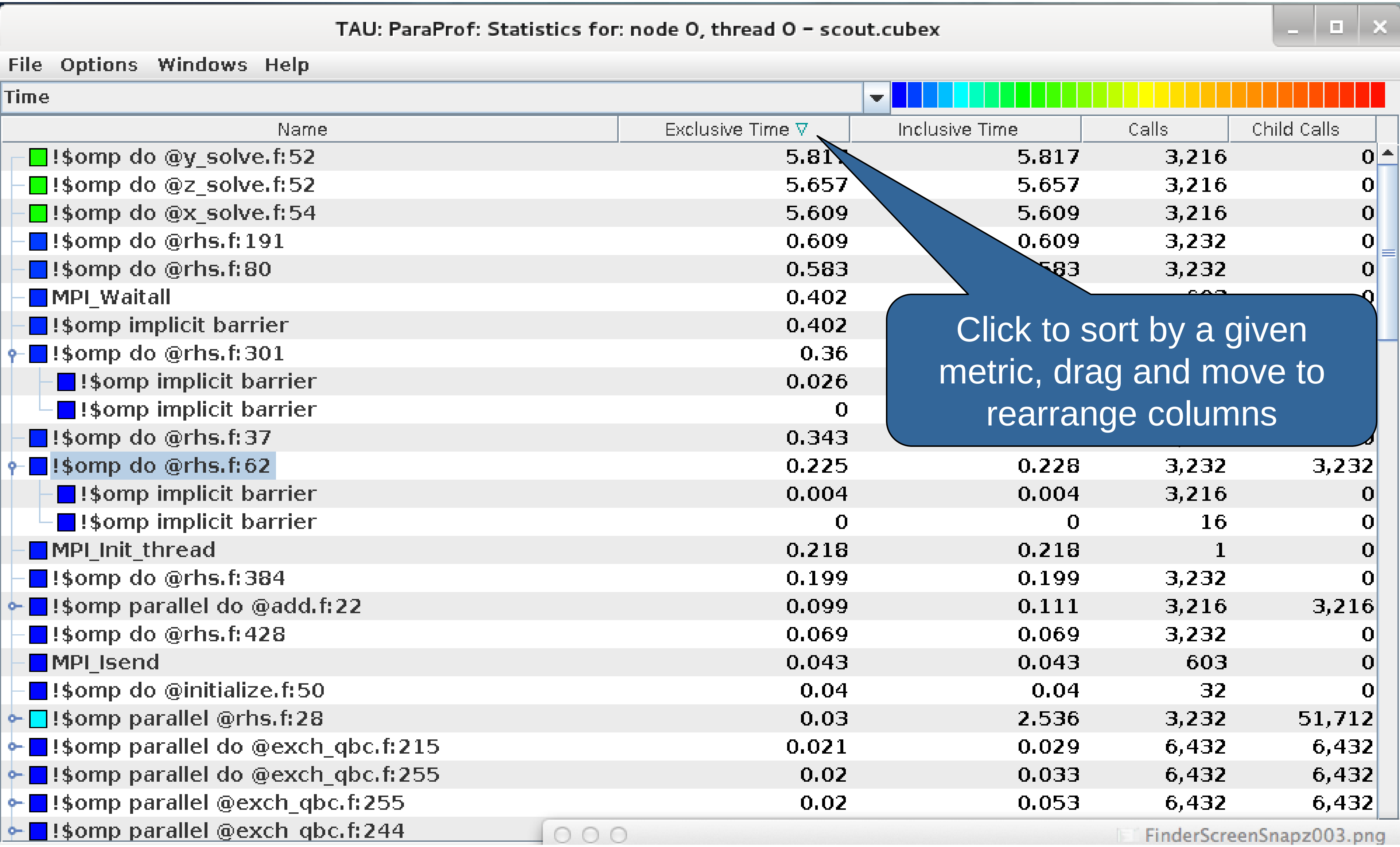
Each color represents an event executing on one or more threads

ParaProf: Windows



Right click on a given node to choose other windows

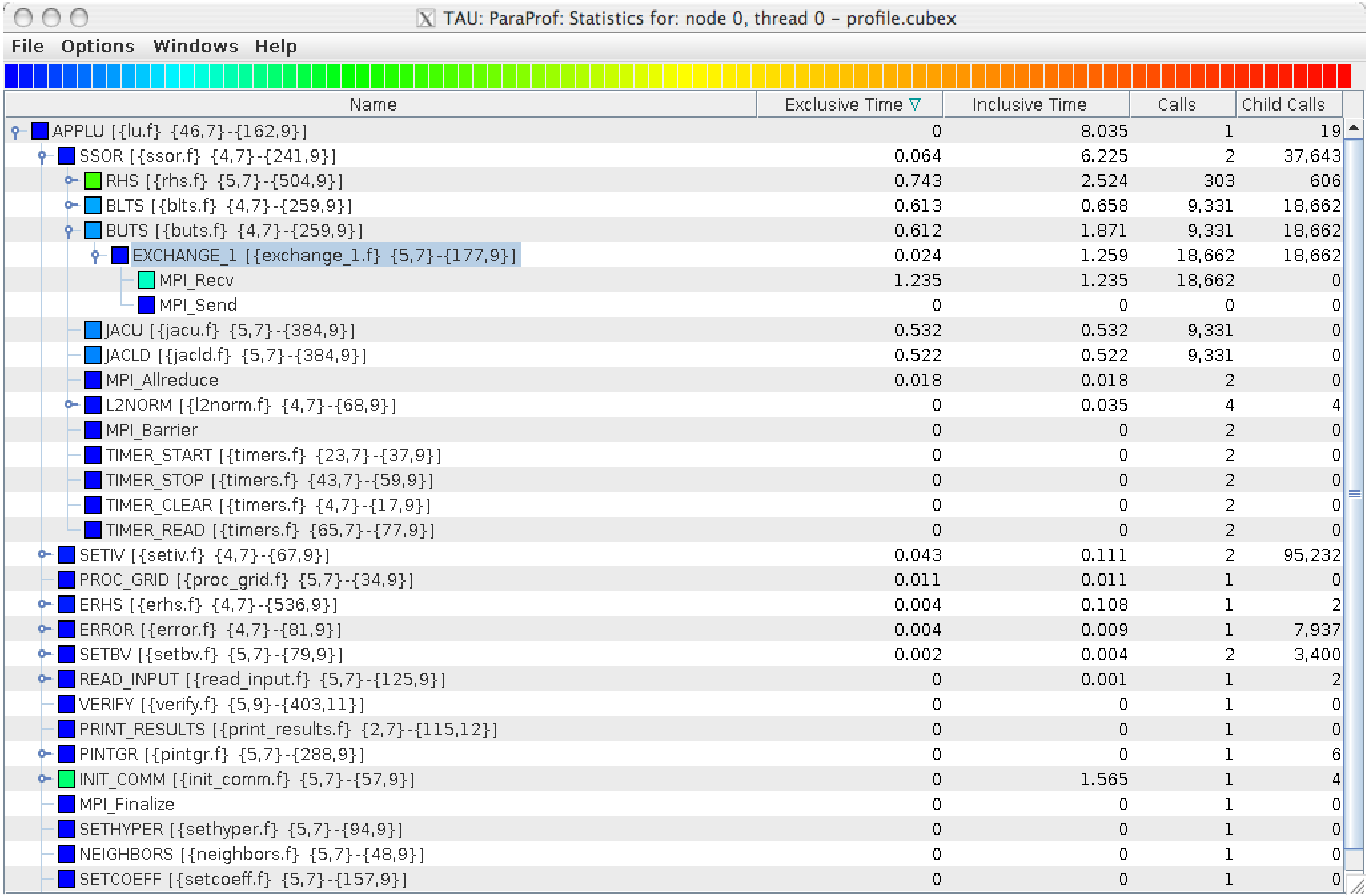
ParaProf: Thread Statistics Table


TAU: ParaProf: Statistics for: node 0, thread 0 - scout.cubex
File Options Windows Help
Time
FinderScreenSnapz003.png

Name	Exclusive Time ▽	Inclusive Time	Calls	Child Calls
🟩 !$omp do @y_solve.f:52	5.81[illegible]	5.817	3,216	0
🟩 !$omp do @z_solve.f:52	5.657	5.657	3,216	0
🟩 !$omp do @x_solve.f:54	5.609	5.609	3,216	0
🟦 !$omp do @rhs.f:191	0.609	0.609	3,232	0
🟦 !$omp do @rhs.f:80	0.583	[illegible]583	3,232	0
🟦 MPI_Waitall	0.402	[illegible]	[illegible]	0
🟦 !$omp implicit barrier	0.402	[illegible]	[illegible]	[illegible]
🟦 !$omp do @rhs.f:301	0.36	[illegible]	[illegible]	[illegible]
🟦 !$omp implicit barrier	0.026	[illegible]	[illegible]	[illegible]
🟦 !$omp implicit barrier	0	[illegible]	[illegible]	[illegible]
🟦 !$omp do @rhs.f:37	0.343	[illegible]	[illegible]	[illegible]
🟦 !$omp do @rhs.f:62	0.225	0.228	3,232	3,232
🟦 !$omp implicit barrier	0.004	0.004	3,216	0
🟦 !$omp implicit barrier	0	0	16	0
🟦 MPI_Init_thread	0.218	0.218	1	0
🟦 !$omp do @rhs.f:384	0.199	0.199	3,232	0
🟦 !$omp parallel do @add.f:22	0.099	0.111	3,216	3,216
🟦 !$omp do @rhs.f:428	0.069	0.069	3,232	0
🟦 MPI_Isend	0.043	0.043	603	0
🟦 !$omp do @initialize.f:50	0.04	0.04	32	0
🟦 !$omp parallel @rhs.f:28	0.03	2.536	3,232	51,712
🟦 !$omp parallel do @exch_qbc.f:215	0.021	0.029	6,432	6,432
🟦 !$omp parallel do @exch_qbc.f:255	0.02	0.033	6,432	6,432
🟦 !$omp parallel @exch_qbc.f:255	0.02	0.053	6,432	6,432
🟦 !$omp parallel @exch_qbc.f:244	[illegible]	[illegible]	[illegible]	[illegible]

Click to sort by a given metric, drag and move to rearrange columns

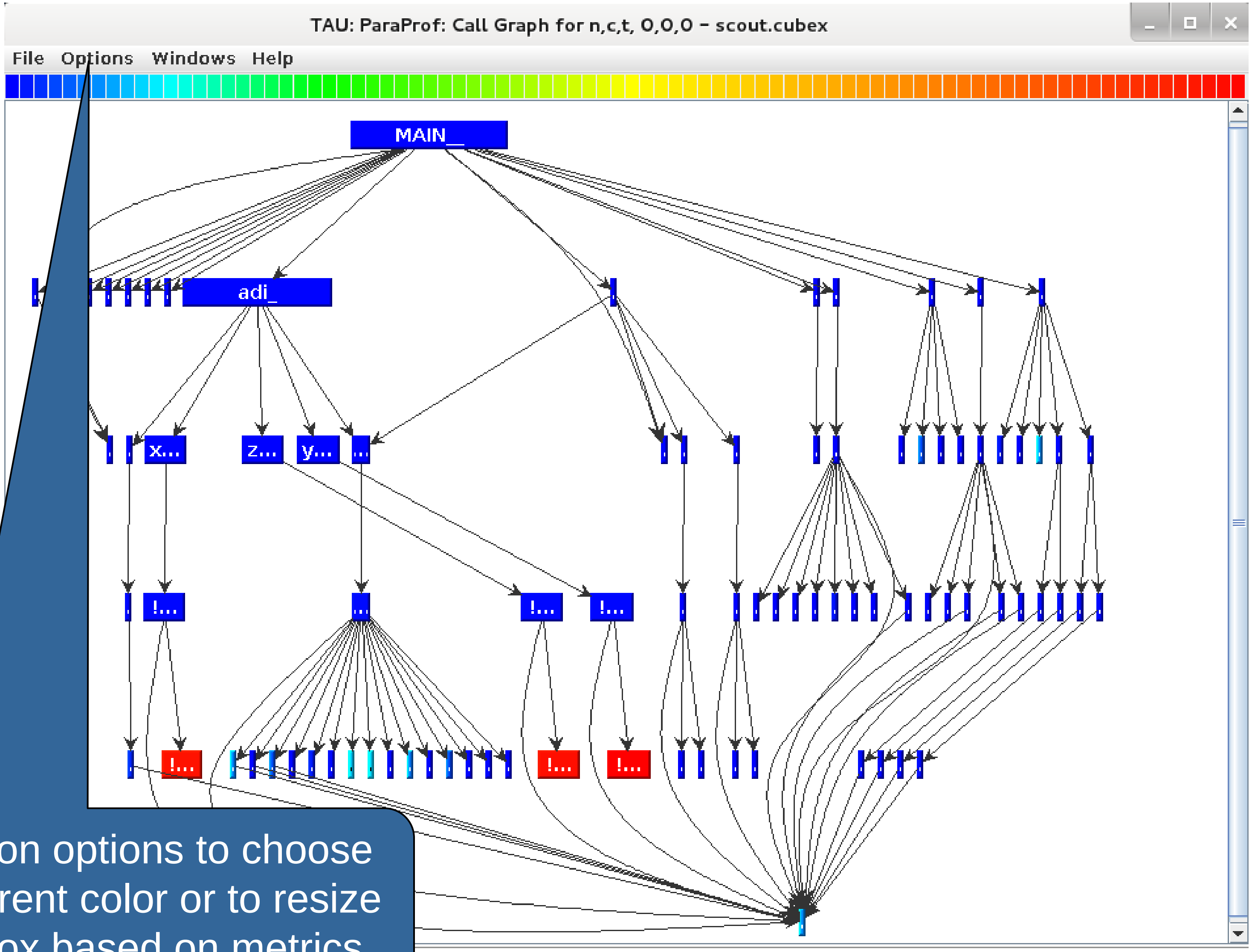
Example: Score-P with TAU (NPB LU)



TAU: ParaProf: Statistics for: node 0, thread 0 – profile.cubex

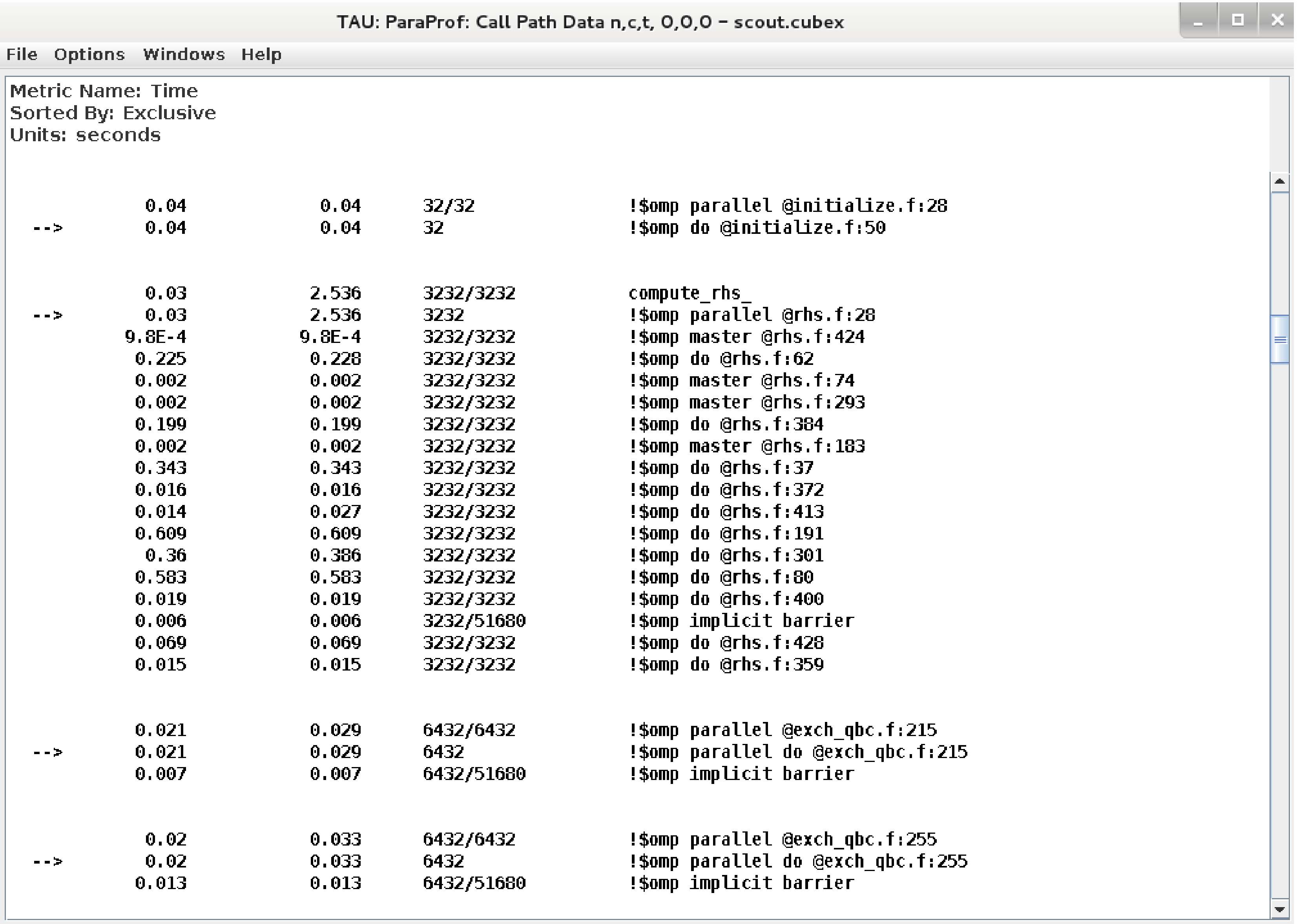
Name	Exclusive Time ▽	Inclusive Time	Calls	Child Calls
🟦 APPLU [{lu.f} {46,7}-{162,9}]	0	8.035	1	19
🟦 SSOR [{ssor.f} {4,7}-{241,9}]	0.064	6.225	2	37,643
🟩 RHS [{rhs.f} {5,7}-{504,9}]	0.743	2.524	303	606
🟦 BLTS [{blts.f} {4,7}-{259,9}]	0.613	0.658	9,331	18,662
🟦 BUTS [{buts.f} {4,7}-{259,9}]	0.612	1.871	9,331	18,662
🟦 EXCHANGE_1 [{exchange_1.f} {5,7}-{177,9}]	0.024	1.259	18,662	18,662
🟩 MPI_Recv	1.235	1.235	18,662	0
🟦 MPI_Send	0	0	0	0
🟦 JACU [{jacu.f} {5,7}-{384,9}]	0.532	0.532	9,331	0
🟦 JACLD [{jacld.f} {5,7}-{384,9}]	0.522	0.522	9,331	0
🟦 MPI_Allreduce	0.018	0.018	2	0
🟦 L2NORM [{l2norm.f} {4,7}-{68,9}]	0	0.035	4	4
🟦 MPI_Barrier	0	0	2	0
🟦 TIMER_START [{timers.f} {23,7}-{37,9}]	0	0	2	0
🟦 TIMER_STOP [{timers.f} {43,7}-{59,9}]	0	0	2	0
🟦 TIMER_CLEAR [{timers.f} {4,7}-{17,9}]	0	0	2	0
🟦 TIMER_READ [{timers.f} {65,7}-{77,9}]	0	0	2	0
🟦 SETIV [{setiv.f} {4,7}-{67,9}]	0.043	0.111	2	95,232
🟦 PROC_GRID [{proc_grid.f} {5,7}-{34,9}]	0.011	0.011	1	0
🟦 ERHS [{erhs.f} {4,7}-{536,9}]	0.004	0.108	1	2
🟦 ERROR [{error.f} {4,7}-{81,9}]	0.004	0.009	1	7,937
🟦 SETBV [{setbv.f} {5,7}-{79,9}]	0.002	0.004	2	3,400
🟦 READ_INPUT [{read_input.f} {5,7}-{125,9}]	0	0.001	1	2
🟦 VERIFY [{verify.f} {5,9}-{403,11}]	0	0	1	0
🟦 PRINT_RESULTS [{print_results.f} {2,7}-{115,12}]	0	0	1	0
🟦 PINTGR [{pintgr.f} {5,7}-{288,9}]	0	0	1	6
🟩 INIT_COMM [{init_comm.f} {5,7}-{57,9}]	0	1.565	1	4
🟦 MPI_Finalize	0	0	1	0
🟦 SETHYPER [{sethyper.f} {5,7}-{94,9}]	0	0	1	0
🟦 NEIGHBORS [{neighbors.f} {5,7}-{48,9}]	0	0	1	0
🟦 SETCOEFF [{setcoeff.f} {5,7}-{157,9}]	0	0	1	0

ParaProf: Thread Callgraph Window



Click on options to choose a different color or to resize the box based on metrics

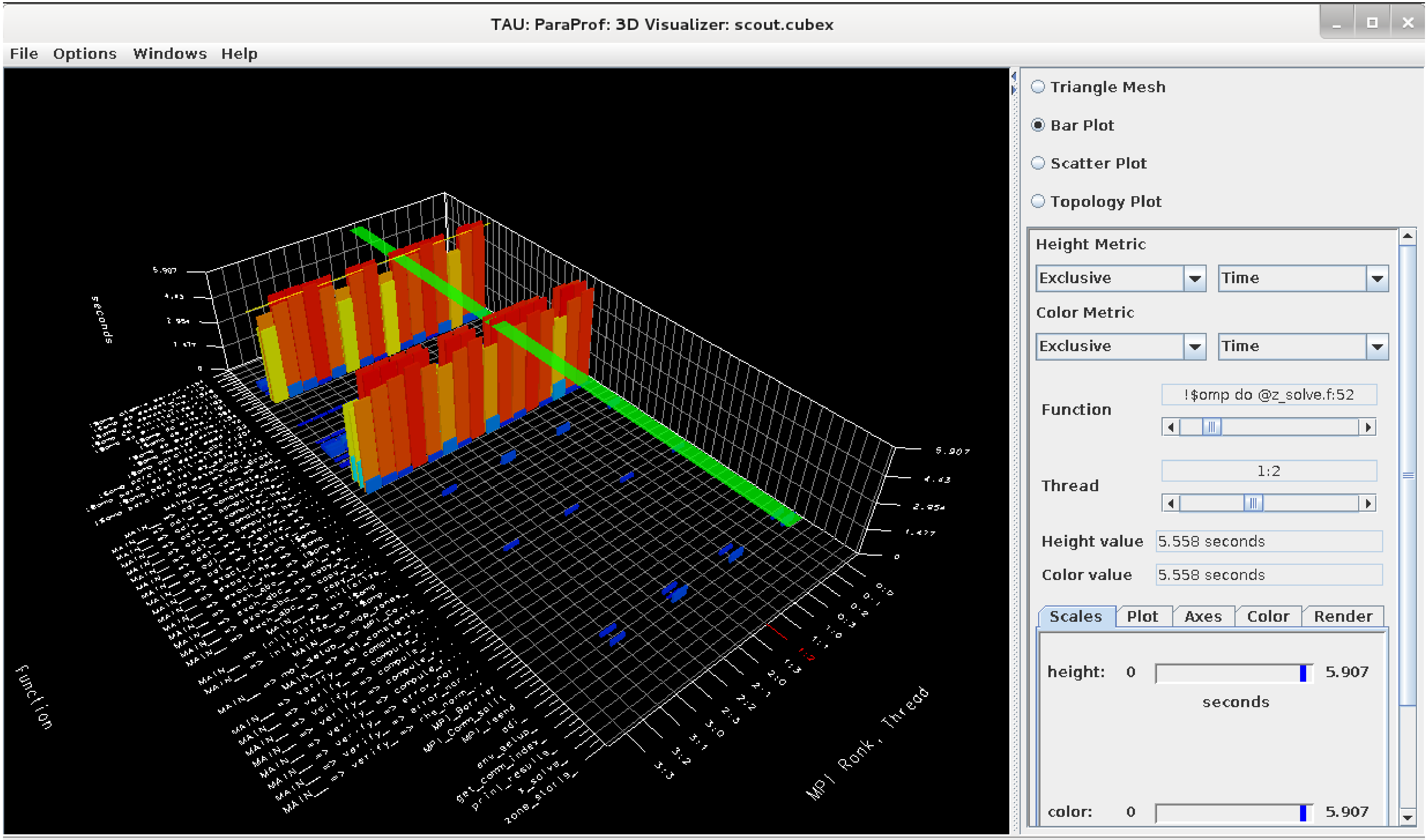
ParaProf: Callpath Thread Relations Window



TAU: ParaProf: Call Path Data n,c,t, 0,0,0 - scout.cubex
File Options Windows Help

```
Metric Name: Time
Sorted By: Exclusive
Units: seconds


             0.04              0.04      32/32               !$omp parallel @initialize.f:28
   -->       0.04              0.04      32                  !$omp do @initialize.f:50


             0.03             2.536      3232/3232           compute_rhs_
   -->       0.03             2.536      3232                !$omp parallel @rhs.f:28
           9.8E-4            9.8E-4      3232/3232           !$omp master @rhs.f:424
            0.225             0.228      3232/3232           !$omp do @rhs.f:62
            0.002             0.002      3232/3232           !$omp master @rhs.f:74
            0.002             0.002      3232/3232           !$omp master @rhs.f:293
            0.199             0.199      3232/3232           !$omp do @rhs.f:384
            0.002             0.002      3232/3232           !$omp master @rhs.f:183
            0.343             0.343      3232/3232           !$omp do @rhs.f:37
            0.016             0.016      3232/3232           !$omp do @rhs.f:372
            0.014             0.027      3232/3232           !$omp do @rhs.f:413
            0.609             0.609      3232/3232           !$omp do @rhs.f:191
             0.36             0.386      3232/3232           !$omp do @rhs.f:301
            0.583             0.583      3232/3232           !$omp do @rhs.f:80
            0.019             0.019      3232/3232           !$omp do @rhs.f:400
            0.006             0.006      3232/51680          !$omp implicit barrier
            0.069             0.069      3232/3232           !$omp do @rhs.f:428
            0.015             0.015      3232/3232           !$omp do @rhs.f:359


            0.021             0.029      6432/6432           !$omp parallel @exch_qbc.f:215
   -->      0.021             0.029      6432                !$omp parallel do @exch_qbc.f:215
            0.007             0.007      6432/51680          !$omp implicit barrier


             0.02             0.033      6432/6432           !$omp parallel @exch_qbc.f:255
   -->       0.02             0.033      6432                !$omp parallel do @exch_qbc.f:255
            0.013             0.013      6432/51680          !$omp implicit barrier
```

ParaProf: Windows -> 3D Visualization -> Bar Plot



TAU: ParaProf: 3D Visualizer: scout.cubex

○ Triangle Mesh

● Bar Plot

○ Scatter Plot

○ Topology Plot

Height Metric: Exclusive, Time

Color Metric: Exclusive, Time

Function: !$omp do @z_solve.f:52

Thread: 1:2

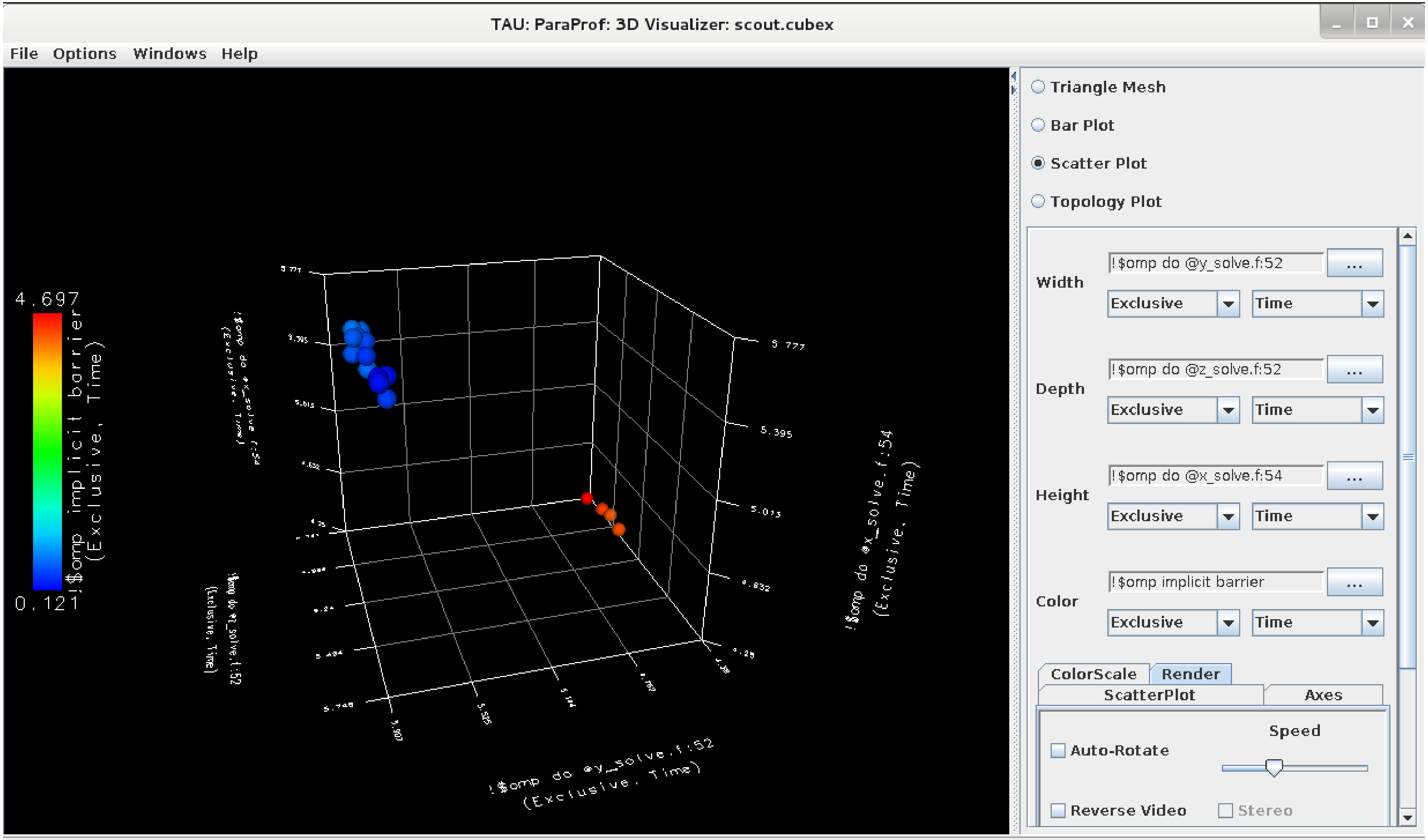
Height value: 5.558 seconds

Color value: 5.558 seconds

height: 0 5.907 seconds

color: 0 5.907

ParaProf: 3D Scatter Plot



TAU: ParaProf: 3D Visualizer: scout.cubex
File
Options
Windows
Help
4.697
0.121
!$omp implicit barrier (Exclusive, Time)
5.777
5.395
5.013
4.632
4.25
5.535
5.760
!$omp do @y_solve.f:52 (Exclusive, Time)
!$omp do @x_solve.f:54 (Exclusive, Time)
!$omp do @z_solve.f:52 (Exclusive, Time)
...
ColorScale
Render
ScatterPlot
Axes
Speed

○ Triangle Mesh
○ Bar Plot
● Scatter Plot
○ Topology Plot

Width: !$omp do @y_solve.f:52 — Exclusive — Time

Depth: !$omp do @z_solve.f:52 — Exclusive — Time

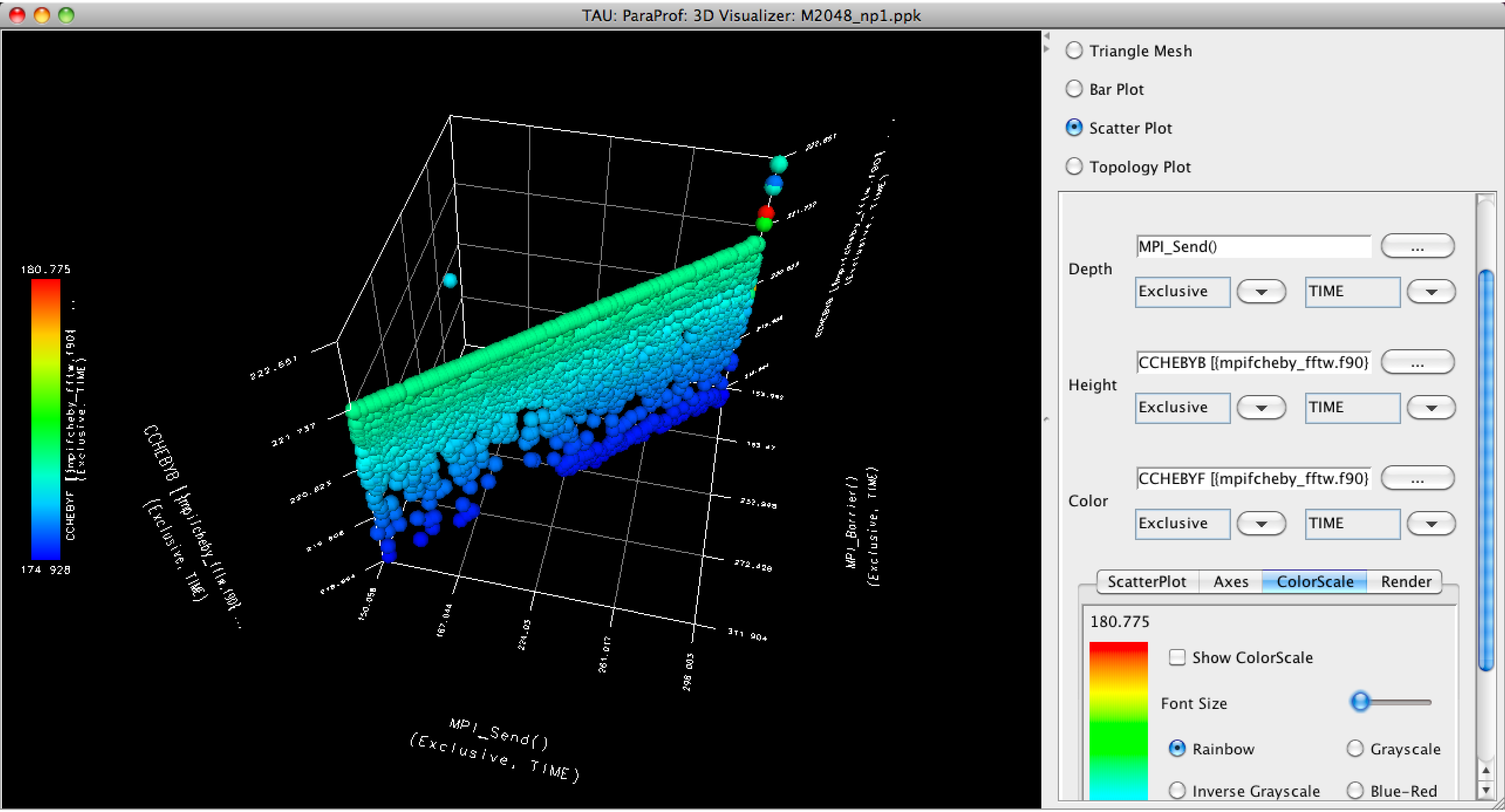
Height: !$omp do @x_solve.f:54 — Exclusive — Time

Color: !$omp implicit barrier — Exclusive — Time

□ Auto-Rotate

□ Reverse Video □ Stereo

ParaProf: Scatter Plot


TAU: ParaProf: 3D Visualizer: M2048_np1.ppk
180.775
174.928
CCHEBYF [{mpifcheby_fftw.f90} (Exclusive, TIME)
CCHEBYB [{mpifcheby_fftw.f90} (Exclusive, TIME)
222.681
221.737
220.823
150.058
187.044
224.03
261.017
298.003
311.904
272.428
252.949
MPI_Barrier() (Exclusive, TIME)
MPI_Send() (Exclusive, TIME)

○ Triangle Mesh

○ Bar Plot

● Scatter Plot

○ Topology Plot

Depth: MPI_Send() ... Exclusive TIME

Height: CCHEBYB [{mpifcheby_fftw.f90} ... Exclusive TIME

Color: CCHEBYF [{mpifcheby_fftw.f90} ... Exclusive TIME

ScatterPlot | Axes | ColorScale | Render

180.775

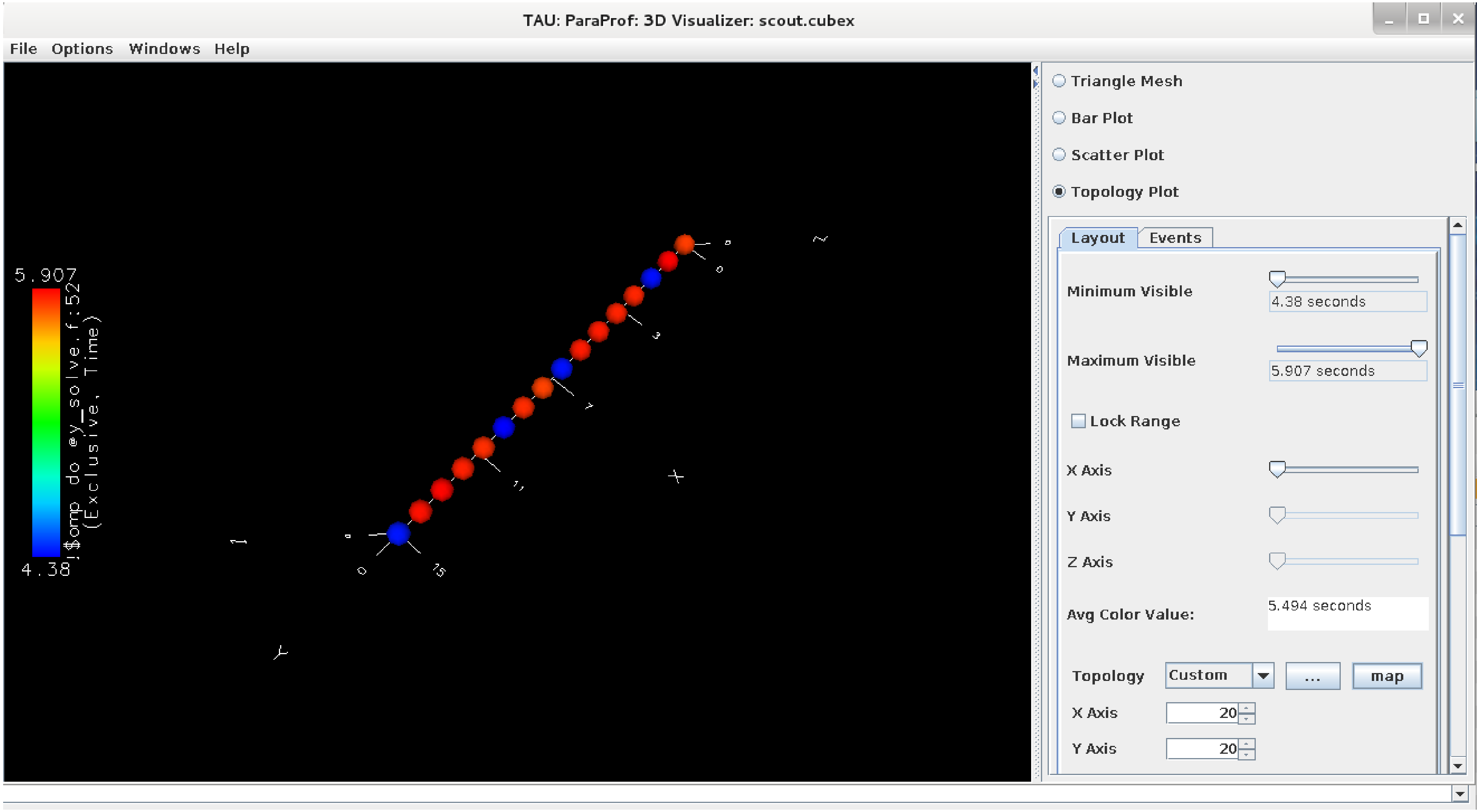
□ Show ColorScale

Font Size

● Rainbow ○ Grayscale

○ Inverse Grayscale ○ Blue-Red

ParaProf: Topology 3D View



TAU: ParaProf: 3D Visualizer: scout.cubex

File Options Windows Help

○ Triangle Mesh

○ Bar Plot

○ Scatter Plot

● Topology Plot

Layout | Events

Minimum Visible: 4.38 seconds

Maximum Visible: 5.907 seconds

□ Lock Range

X Axis

Y Axis

Z Axis

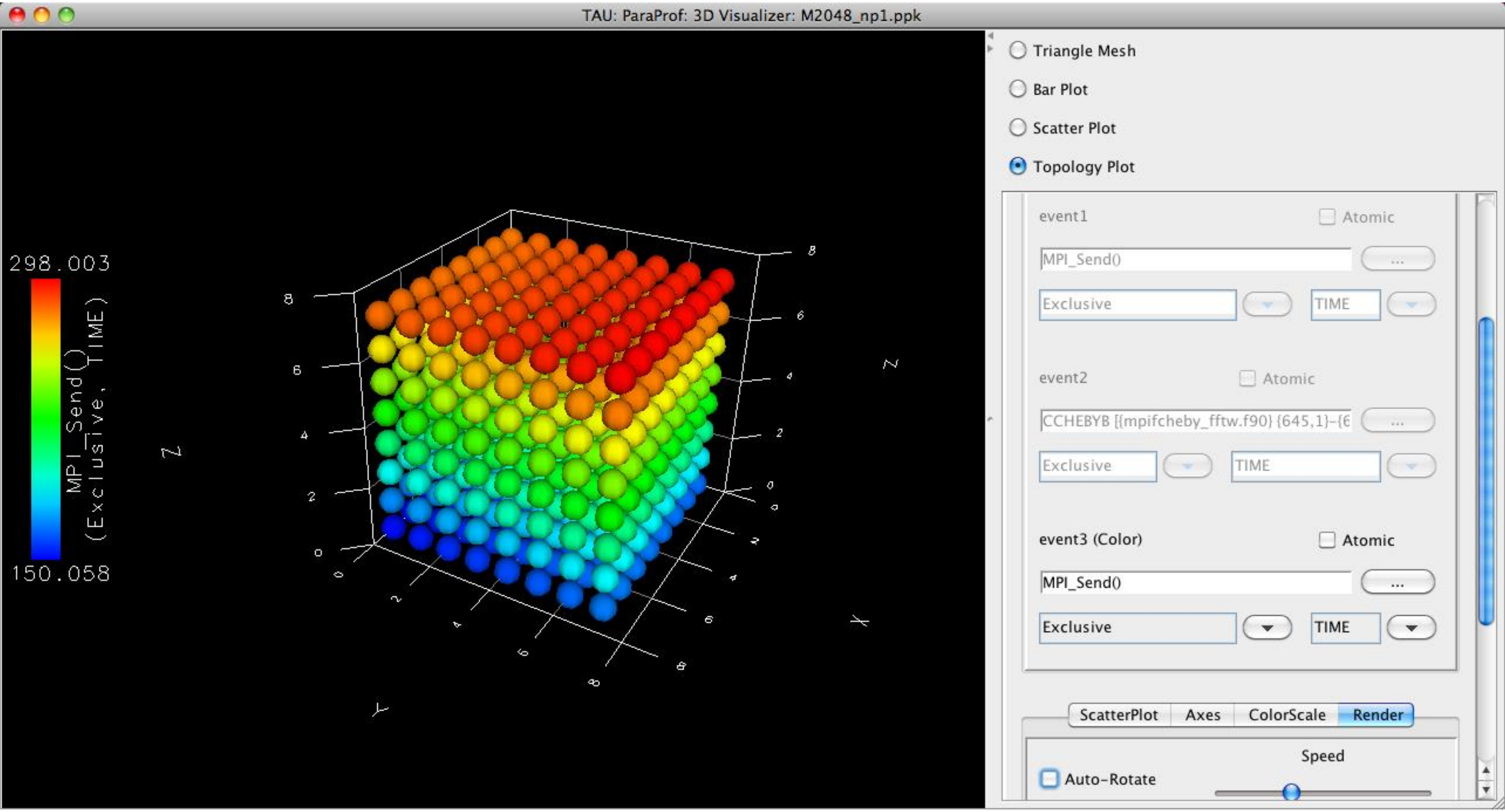
Avg Color Value: 5.494 seconds

Topology: Custom ... map

X Axis: 20

Y Axis: 20

ParaProf: Topology 3D View (IBM BG/P)



○ Triangle Mesh

○ Bar Plot

○ Scatter Plot

● Topology Plot

event1 □ Atomic

MPI_Send() ...

Exclusive TIME

event2 □ Atomic

CCHEBYB [{mpifcheby_fftw.f90} {645,1}-{6 ...

Exclusive TIME

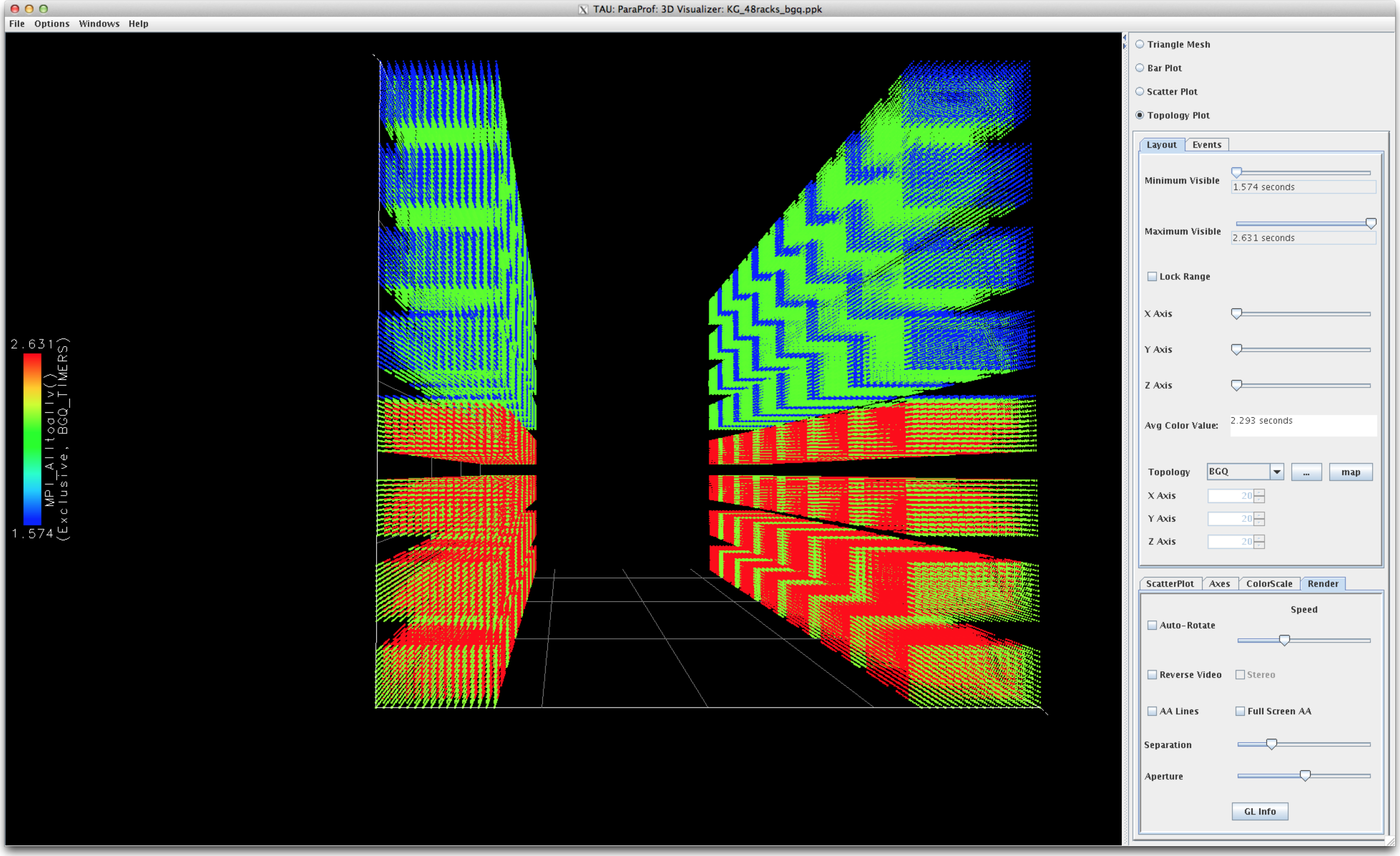
event3 (Color) □ Atomic

MPI_Send() ...

Exclusive TIME

□ Auto-Rotate

ParaProf: Topology View Torus (IBM BG/Q)



TAU: ParaProf: 3D Visualizer: KG_48racks_bgq.ppk

○ Triangle Mesh

○ Bar Plot

○ Scatter Plot

● Topology Plot

Minimum Visible: 1.574 seconds

Maximum Visible: 2.631 seconds

□ Lock Range

X Axis

Y Axis

Z Axis

Avg Color Value: 2.293 seconds

Topology: BGQ

X Axis: 20

Y Axis: 20

Z Axis: 20

Speed

□ Auto-Rotate

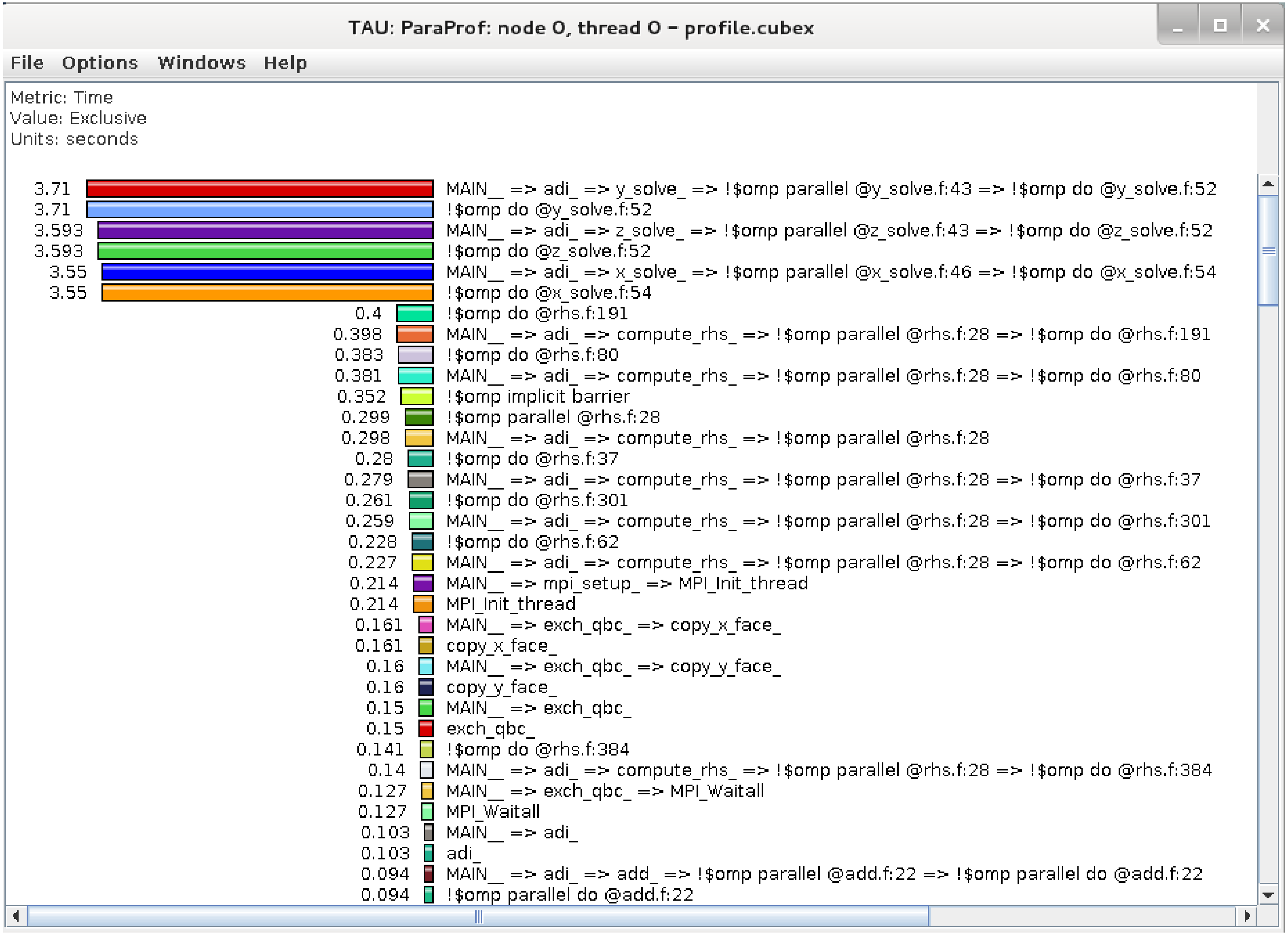
□ Reverse Video □ Stereo

□ AA Lines □ Full Screen AA

Separation

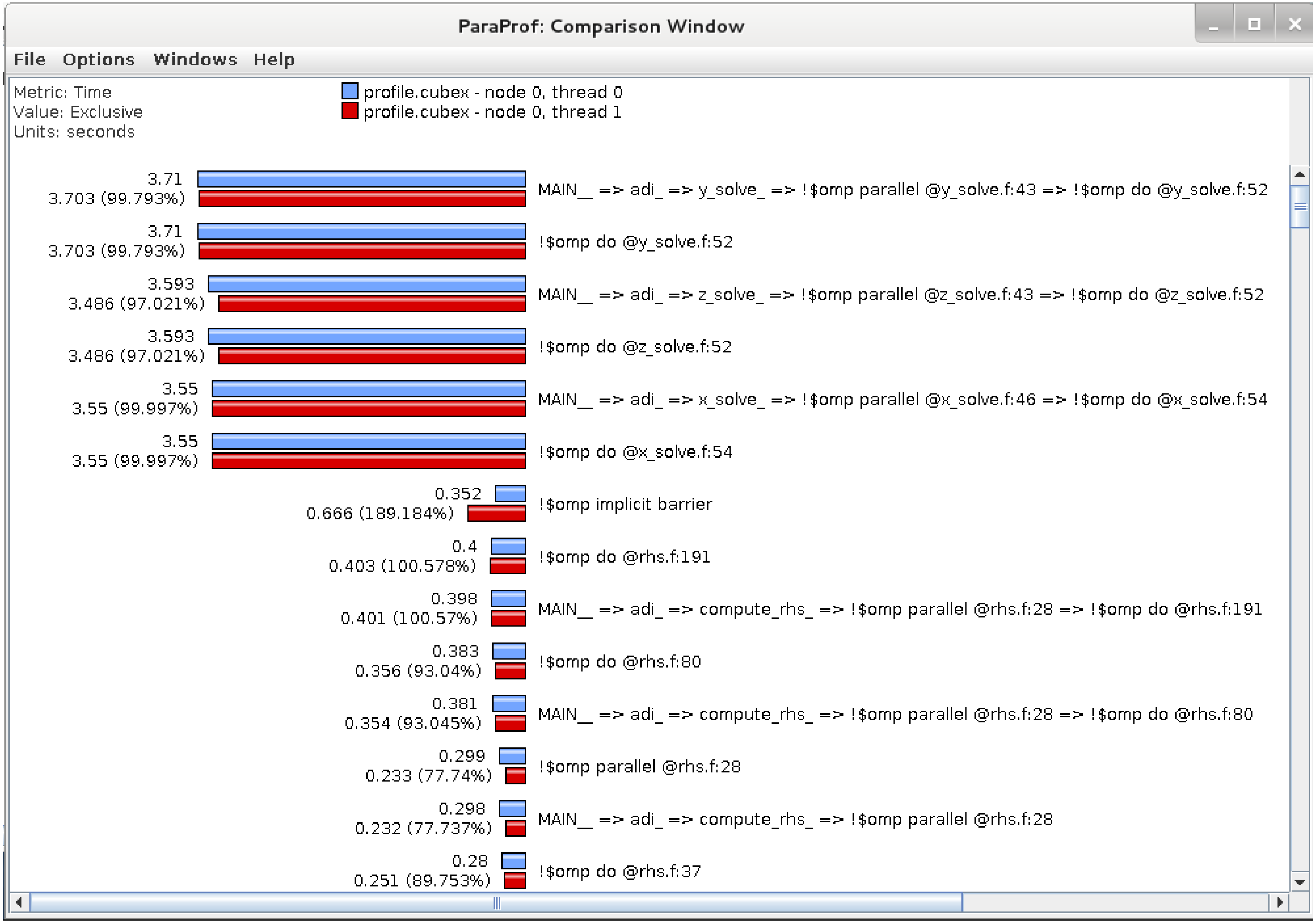
Aperture

ParaProf: Node View



```
Metric: Time
Value: Exclusive
Units: seconds

3.71   MAIN__ => adi_ => y_solve_ => !$omp parallel @y_solve.f:43 => !$omp do @y_solve.f:52
3.71   !$omp do @y_solve.f:52
3.593  MAIN__ => adi_ => z_solve_ => !$omp parallel @z_solve.f:43 => !$omp do @z_solve.f:52
3.593  !$omp do @z_solve.f:52
3.55   MAIN__ => adi_ => x_solve_ => !$omp parallel @x_solve.f:46 => !$omp do @x_solve.f:54
3.55   !$omp do @x_solve.f:54
0.4    !$omp do @rhs.f:191
0.398  MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:191
0.383  !$omp do @rhs.f:80
0.381  MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:80
0.352  !$omp implicit barrier
0.299  !$omp parallel @rhs.f:28
0.298  MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28
0.28   !$omp do @rhs.f:37
0.279  MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:37
0.261  !$omp do @rhs.f:301
0.259  MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:301
0.228  !$omp do @rhs.f:62
0.227  MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:62
0.214  MAIN__ => mpi_setup_ => MPI_Init_thread
0.214  MPI_Init_thread
0.161  MAIN__ => exch_qbc_ => copy_x_face_
0.161  copy_x_face_
0.16   MAIN__ => exch_qbc_ => copy_y_face_
0.16   copy_y_face_
0.15   MAIN__ => exch_qbc_
0.15   exch_qbc_
0.141  !$omp do @rhs.f:384
0.14   MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:384
0.127  MAIN__ => exch_qbc_ => MPI_Waitall
0.127  MPI_Waitall
0.103  MAIN__ => adi_
0.103  adi_
0.094  MAIN__ => adi_ => add_ => !$omp parallel @add.f:22 => !$omp parallel do @add.f:22
0.094  !$omp parallel do @add.f:22
```

ParaProf: Add Thread to Comparison Window

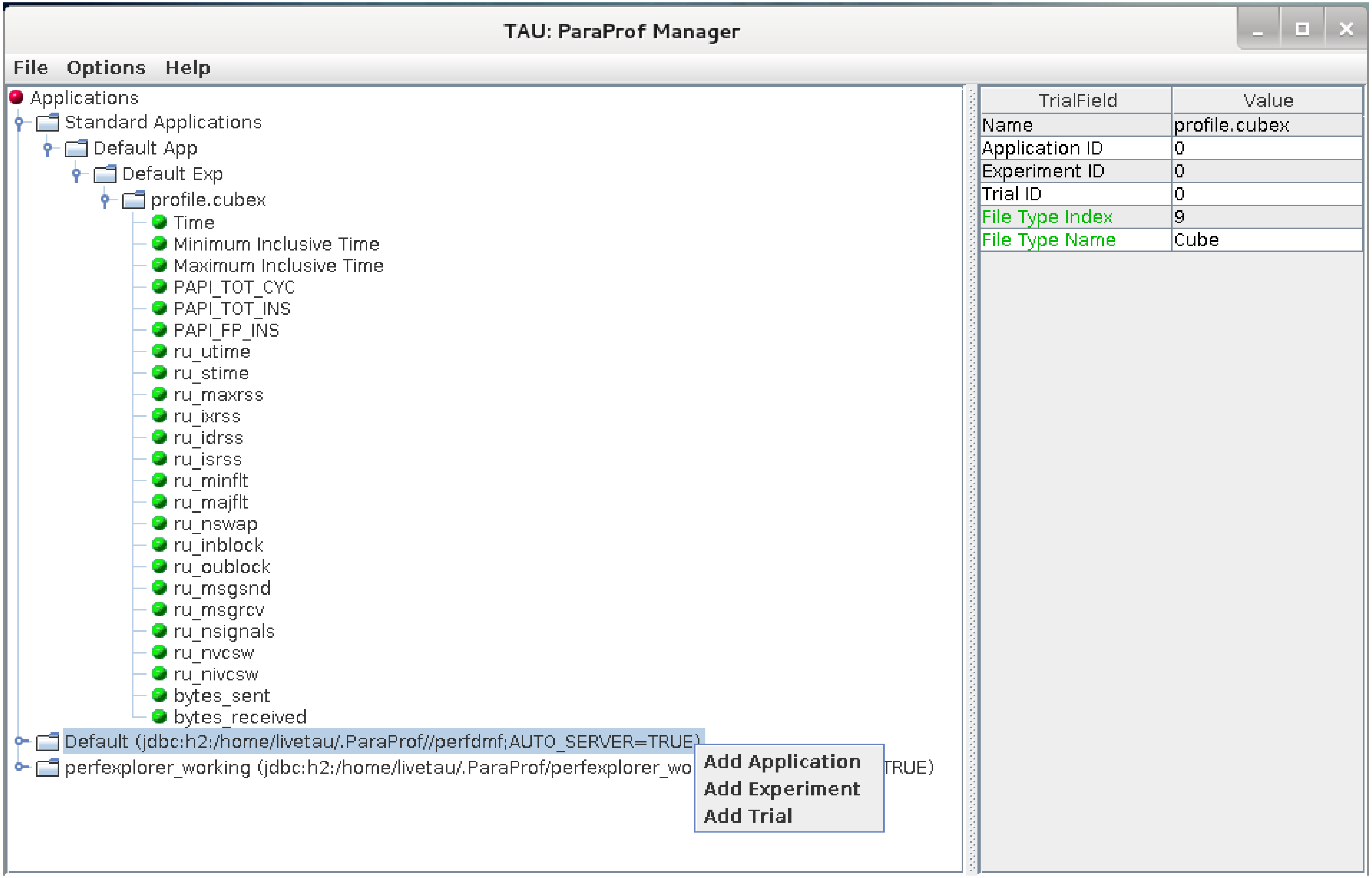


Metric: Time
Value: Exclusive
Units: seconds

🟦 profile.cubex - node 0, thread 0
🟥 profile.cubex - node 0, thread 1

```
3.71
3.703 (99.793%)    MAIN__ => adi_ => y_solve_ => !$omp parallel @y_solve.f:43 => !$omp do @y_solve.f:52
3.71
3.703 (99.793%)    !$omp do @y_solve.f:52
3.593
3.486 (97.021%)    MAIN__ => adi_ => z_solve_ => !$omp parallel @z_solve.f:43 => !$omp do @z_solve.f:52
3.593
3.486 (97.021%)    !$omp do @z_solve.f:52
3.55
3.55 (99.997%)     MAIN__ => adi_ => x_solve_ => !$omp parallel @x_solve.f:46 => !$omp do @x_solve.f:54
3.55
3.55 (99.997%)     !$omp do @x_solve.f:54
0.352
0.666 (189.184%)   !$omp implicit barrier
0.4
0.403 (100.578%)   !$omp do @rhs.f:191
0.398
0.401 (100.57%)    MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:191
0.383
0.356 (93.04%)     !$omp do @rhs.f:80
0.381
0.354 (93.045%)    MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:80
0.299
0.233 (77.74%)     !$omp parallel @rhs.f:28
0.298
0.232 (77.737%)    MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28
0.28
0.251 (89.753%)    !$omp do @rhs.f:37
```

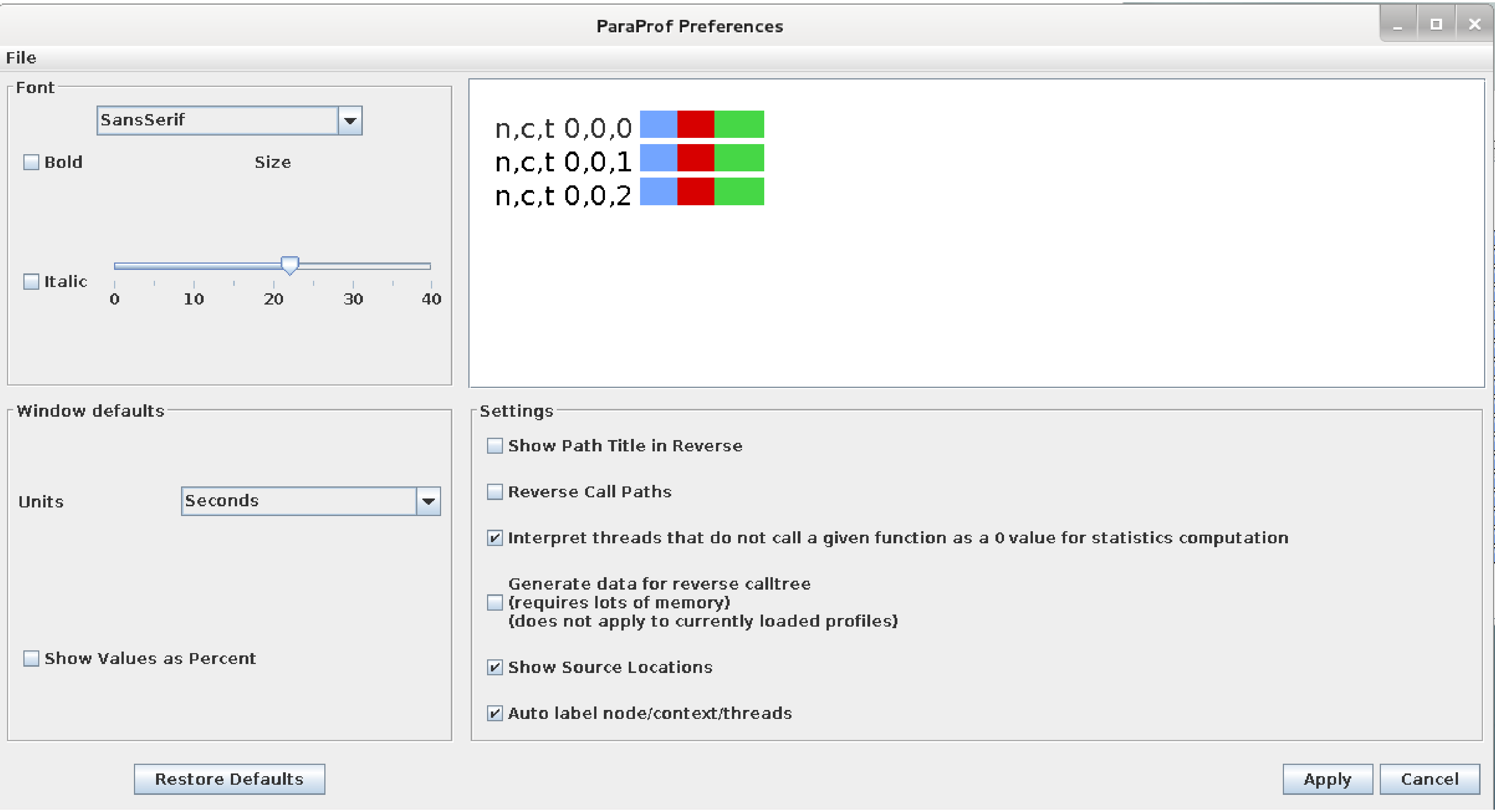
ParaProf: Score-P Profile Files, Database



- Applications
  - Standard Applications
    - Default App
      - Default Exp
        - profile.cubex
          - Time
          - Minimum Inclusive Time
          - Maximum Inclusive Time
          - PAPI_TOT_CYC
          - PAPI_TOT_INS
          - PAPI_FP_INS
          - ru_utime
          - ru_stime
          - ru_maxrss
          - ru_ixrss
          - ru_idrss
          - ru_isrss
          - ru_minflt
          - ru_majflt
          - ru_nswap
          - ru_inblock
          - ru_oublock
          - ru_msgsnd
          - ru_msgrcv
          - ru_nsignals
          - ru_nvcsw
          - ru_nivcsw
          - bytes_sent
          - bytes_received
  - Default (jdbc:h2:/home/livetau/.ParaProf//perfdmf;AUTO_SERVER=TRUE)
  - perfexplorer_working (jdbc:h2:/home/livetau/.ParaProf/perfexplorer_wo TRUE)

TrialField	Value
Name	profile.cubex
Application ID	0
Experiment ID	0
Trial ID	0
File Type Index	9
File Type Name	Cube

ParaProf: File Preferences Window



ParaProf Preferences

Font

SansSerif

□ Bold

Size

□ Italic

n,c,t 0,0,0

n,c,t 0,0,1

n,c,t 0,0,2

Window defaults

Units: Seconds

□ Show Values as Percent

Settings

□ Show Path Title in Reverse

□ Reverse Call Paths

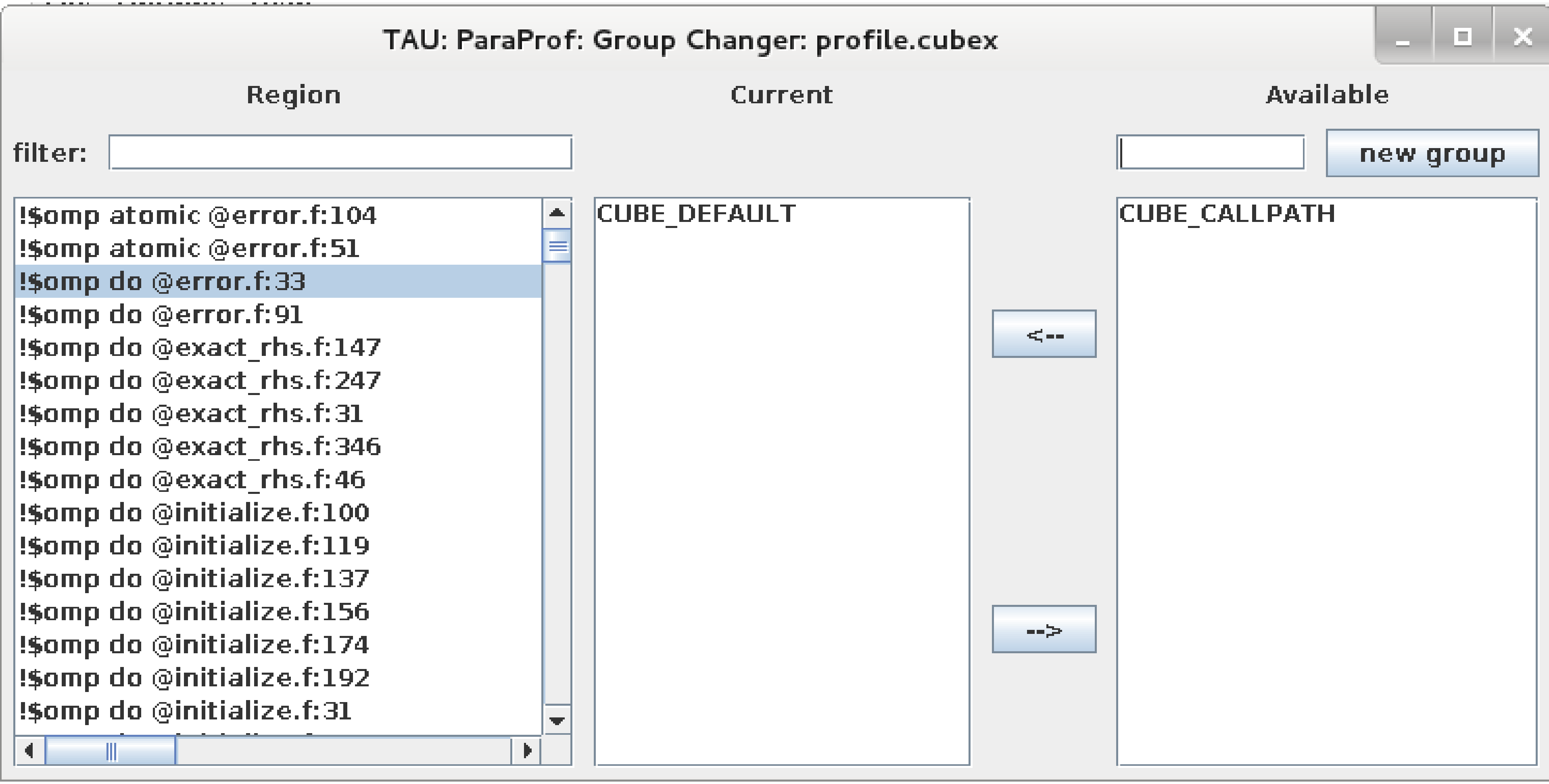
☑ Interpret threads that do not call a given function as a 0 value for statistics computation

□ Generate data for reverse calltree
(requires lots of memory)
(does not apply to currently loaded profiles)

☑ Show Source Locations

☑ Auto label node/context/threads

ParaProf: Group Changer Window



TAU: ParaProf: Group Changer: profile.cubex

Region

filter:

!$omp atomic @error.f:104
!$omp atomic @error.f:51
!$omp do @error.f:33
!$omp do @error.f:91
!$omp do @exact_rhs.f:147
!$omp do @exact_rhs.f:247
!$omp do @exact_rhs.f:31
!$omp do @exact_rhs.f:346
!$omp do @exact_rhs.f:46
!$omp do @initialize.f:100
!$omp do @initialize.f:119
!$omp do @initialize.f:137
!$omp do @initialize.f:156
!$omp do @initialize.f:174
!$omp do @initialize.f:192
!$omp do @initialize.f:31

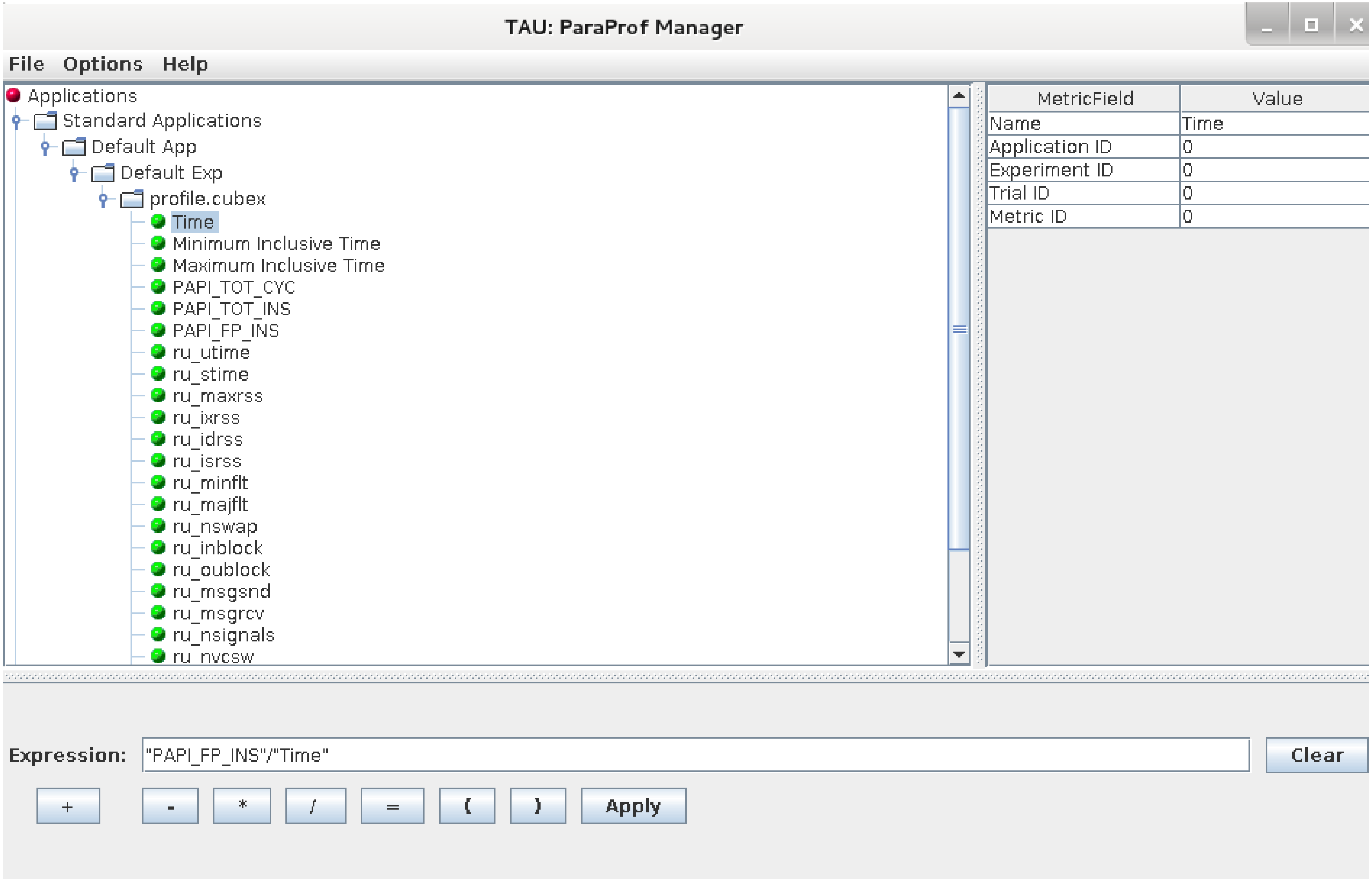
Current

CUBE_DEFAULT

Available

CUBE_CALLPATH

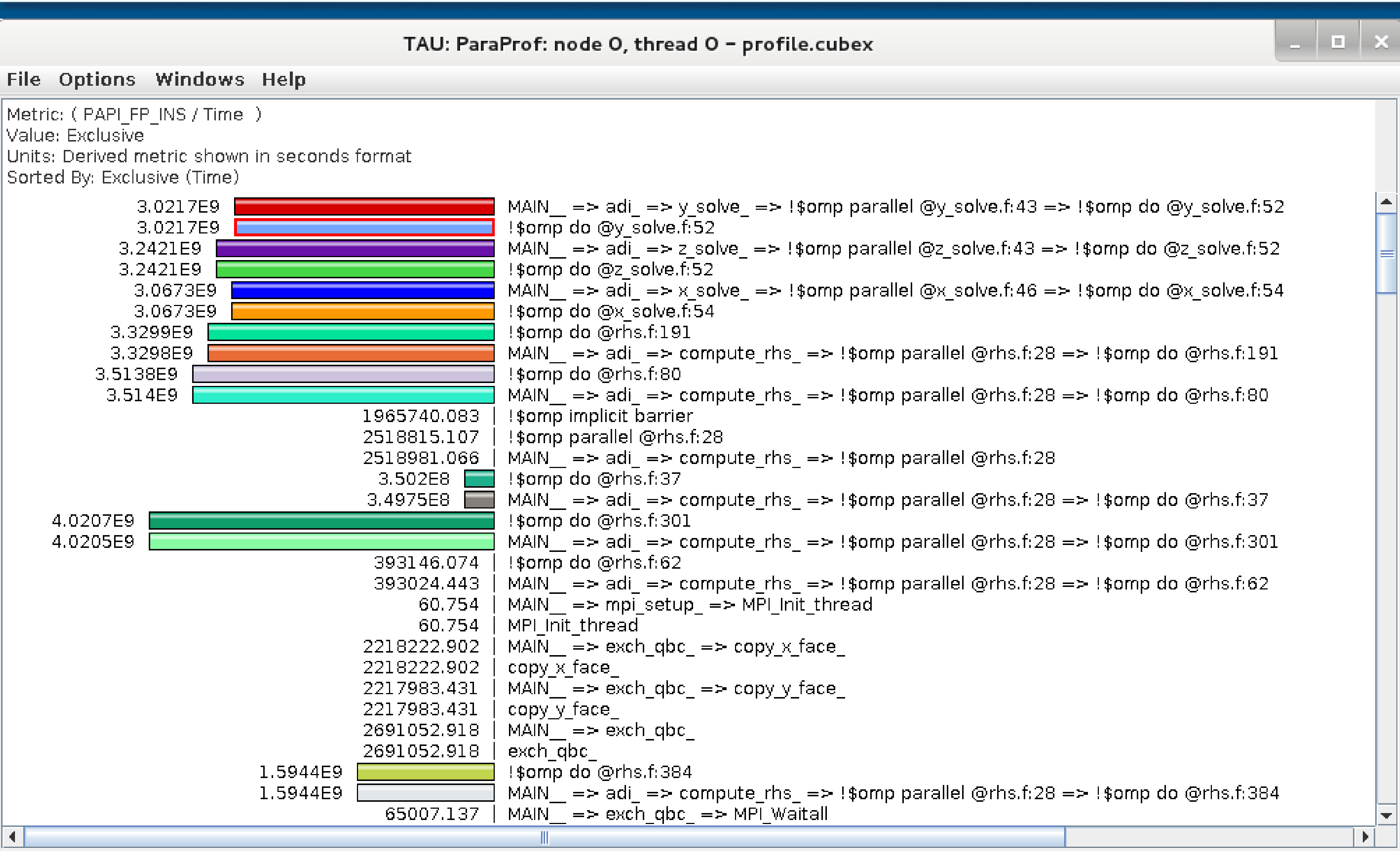
ParaProf: Derived Metric Panel in Manager Window



MetricField	Value
Name	Time
Application ID	0
Experiment ID	0
Trial ID	0
Metric ID	0

Expression: "PAPI_FP_INS"/"Time"

Sorting Derived FLOPS metric by Exclusive Time



```
Metric: ( PAPI_FP_INS / Time  )
Value: Exclusive
Units: Derived metric shown in seconds format
Sorted By: Exclusive (Time)

     3.0217E9   MAIN__ => adi_ => y_solve_ => !$omp parallel @y_solve.f:43 => !$omp do @y_solve.f:52
     3.0217E9   !$omp do @y_solve.f:52
     3.2421E9   MAIN__ => adi_ => z_solve_ => !$omp parallel @z_solve.f:43 => !$omp do @z_solve.f:52
     3.2421E9   !$omp do @z_solve.f:52
     3.0673E9   MAIN__ => adi_ => x_solve_ => !$omp parallel @x_solve.f:46 => !$omp do @x_solve.f:54
     3.0673E9   !$omp do @x_solve.f:54
     3.3299E9   !$omp do @rhs.f:191
     3.3298E9   MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:191
     3.5138E9   !$omp do @rhs.f:80
      3.514E9   MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:80
  1965740.083 | !$omp implicit barrier
  2518815.107 | !$omp parallel @rhs.f:28
  2518981.066 | MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28
     3.502E8    !$omp do @rhs.f:37
    3.4975E8    MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:37
    4.0207E9    !$omp do @rhs.f:301
    4.0205E9    MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:301
   393146.074 | !$omp do @rhs.f:62
   393024.443 | MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:62
       60.754 | MAIN__ => mpi_setup_ => MPI_Init_thread
       60.754 | MPI_Init_thread
  2218222.902 | MAIN__ => exch_qbc_ => copy_x_face_
  2218222.902 | copy_x_face_
  2217983.431 | MAIN__ => exch_qbc_ => copy_y_face_
  2217983.431 | copy_y_face_
  2691052.918 | MAIN__ => exch_qbc_
  2691052.918 | exch_qbc_
    1.5944E9    !$omp do @rhs.f:384
    1.5944E9    MAIN__ => adi_ => compute_rhs_ => !$omp parallel @rhs.f:28 => !$omp do @rhs.f:384
    65007.137 | MAIN__ => exch_qbc_ => MPI_Waitall
```

Download TAU from U. Oregon

http://tau.uoregon.edu

http://www.hpclinux.com [LiveDVD, OVA]

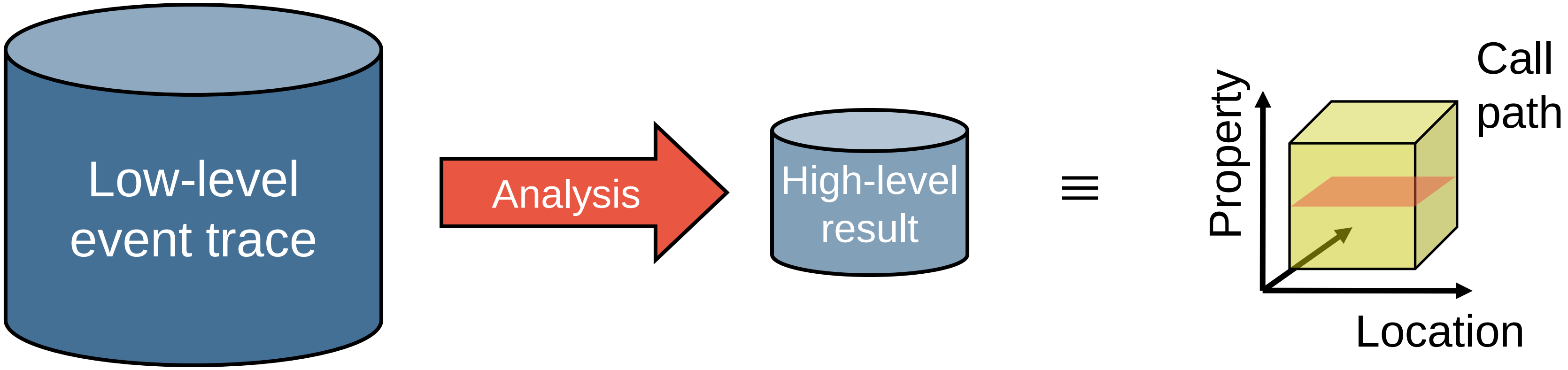
Free download, open source, BSD license

Automatic trace analysis with Scalasca

Markus Geimer
Jülich Supercomputing Centre

Automatic trace analysis

- Idea
  - Automatic search for patterns of inefficient behavior
  - Classification of behavior & quantification of significance



  - Guaranteed to cover the entire event trace
  - Quicker than manual/visual trace analysis
  - Parallel replay analysis exploits available memory & processors to deliver scalability

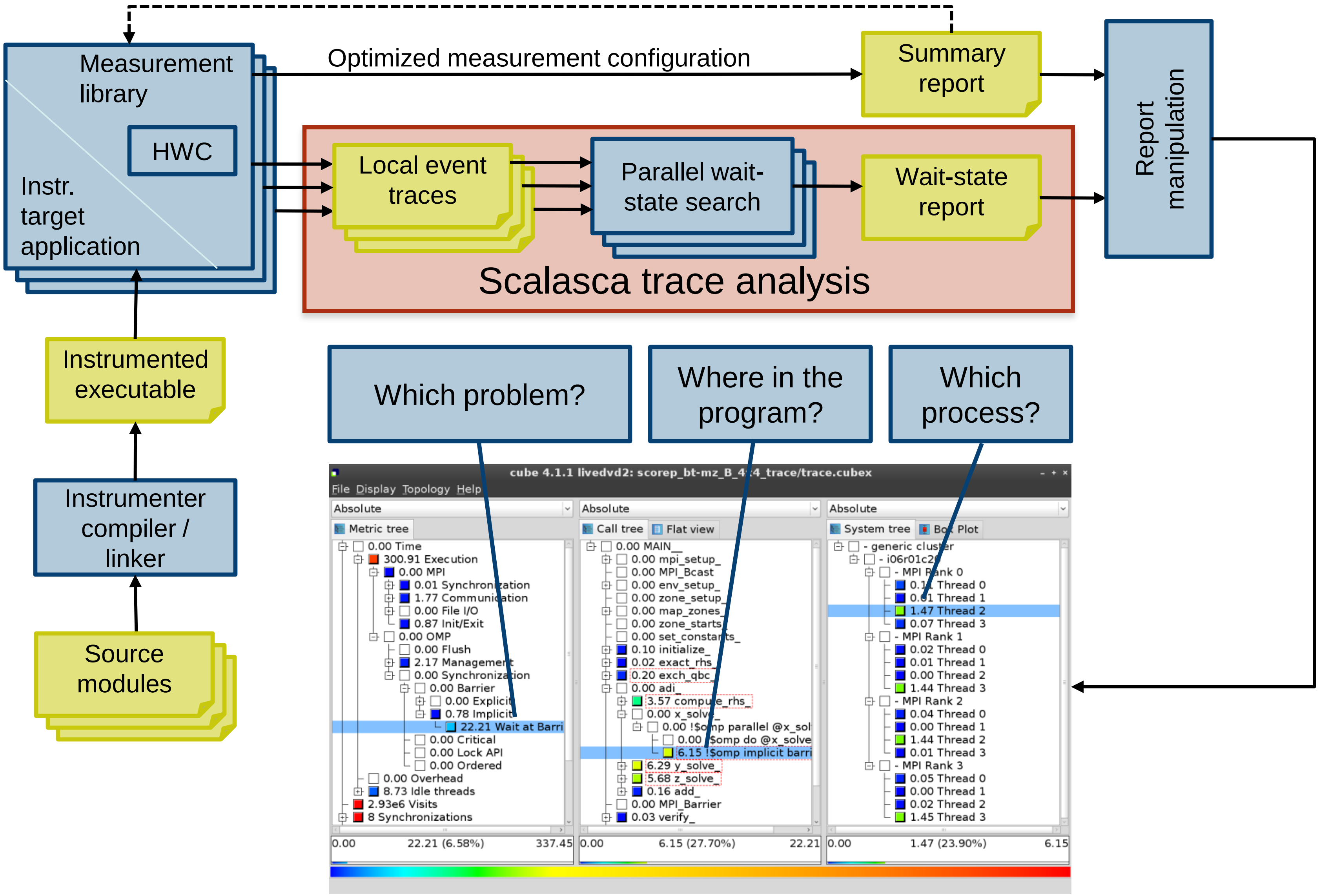
The Scalasca project

- Project started in 2006
  - Follow-up to pioneering KOJAK project (started 1998)
- Joint development of
  - Jülich Supercomputing Centre
  - German Research School for Simulation Sciences
- Development of a **scalable** performance analysis toolset for most popular parallel programming paradigms
- Specifically targeting **large-scale** parallel applications
  - such as those running on IBM BlueGene or Cray XT systems with one million or more processes/threads
- Latest release:
  - Scalasca v2.1 (August 2014)

Scalasca 2.1 features

- Open source, BSD 3-clause license
- Fairly portable
  - IBM Blue Gene, IBM SP & blade clusters, Cray XT/XE/XK/XC, SGI Altix, Solaris & Linux clusters, Fujitsu FX10 & K computer, ...
- Uses Score-P instrumenter & measurement libraries
  - Scalasca 2.1 core package focuses on trace-based analyses
  - Supports common data formats
    - Reads event traces in OTF2 format
    - Writes analysis reports in CUBE4 format
- Current limitations:
  - No support for nested OpenMP parallelism and tasking
  - Unable to handle OTF2 traces containing CUDA events

Scalasca workflow



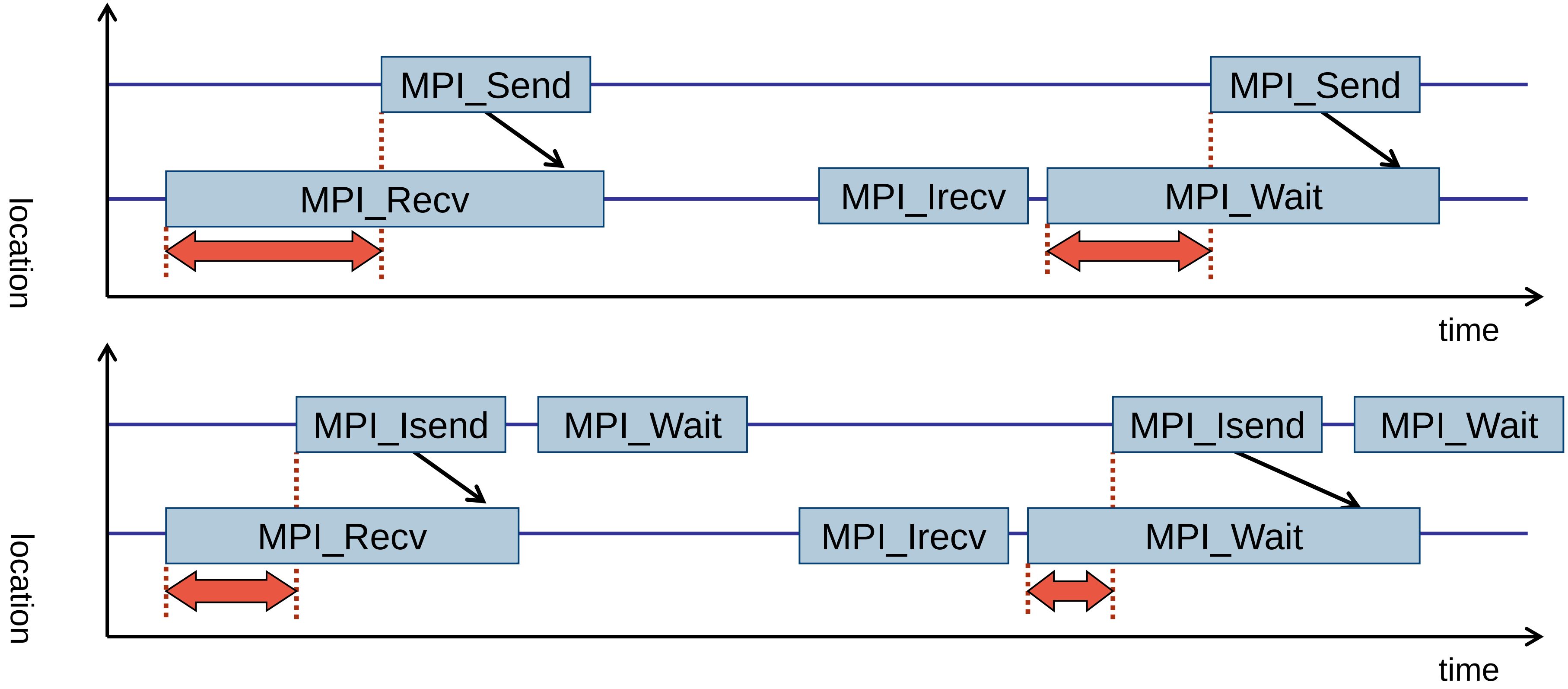
```
cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex
File  Display  Topology  Help
```

```
Absolute
Metric tree
0.00 Time
  300.91 Execution
    0.00 MPI
      0.01 Synchronization
      1.77 Communication
      0.00 File I/O
      0.87 Init/Exit
    0.00 OMP
      0.00 Flush
      2.17 Management
      0.00 Synchronization
        0.00 Barrier
          0.00 Explicit
          0.78 Implicit
            22.21 Wait at Barri
        0.00 Critical
        0.00 Lock API
        0.00 Ordered
    0.00 Overhead
    8.73 Idle threads
  2.93e6 Visits
  8 Synchronizations
0.00          22.21 (6.58%)          337.45
```

```
Absolute
Call tree   Flat view
0.00 MAIN__
  0.00 mpi_setup_
  0.00 MPI_Bcast
  0.00 env_setup_
  0.00 zone_setup_
  0.00 map_zones_
  0.00 zone_starts_
  0.00 set_constants_
  0.10 initialize_
  0.02 exact_rhs_
  0.20 exch_qbc_
  0.00 adi_
    3.57 compute_rhs_
    0.00 x_solve_
      0.00 !$omp parallel @x_sol
        0.00 !$omp do @x_solve
        6.15 !$omp implicit barri
    6.29 y_solve_
    5.68 z_solve_
    0.16 add_
  0.00 MPI_Barrier
  0.03 verify_
0.00          6.15 (27.70%)          22.21
```

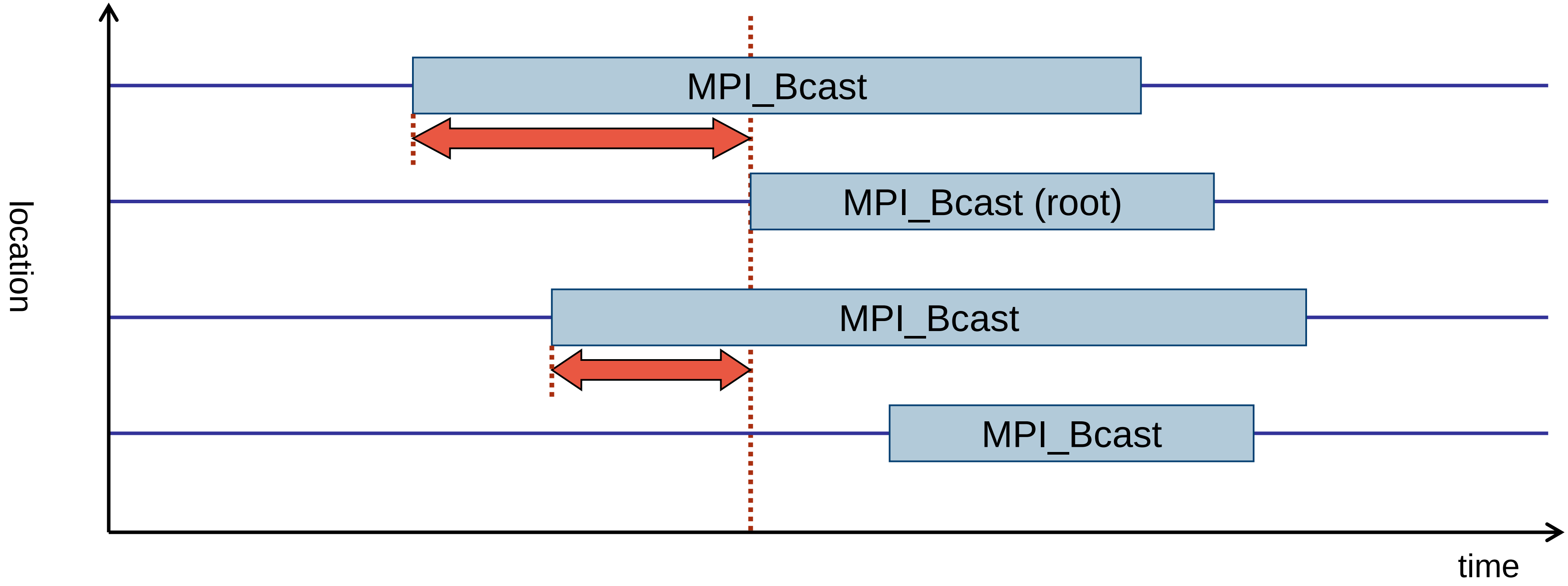
```
Absolute
System tree   Box Plot
- generic cluster
  - i06r01c20
    - MPI Rank 0
        0.11 Thread 0
        0.01 Thread 1
        1.47 Thread 2
        0.07 Thread 3
    - MPI Rank 1
        0.02 Thread 0
        0.01 Thread 1
        0.00 Thread 2
        1.44 Thread 3
    - MPI Rank 2
        0.04 Thread 0
        0.00 Thread 1
        1.44 Thread 2
        0.01 Thread 3
    - MPI Rank 3
        0.05 Thread 0
        0.00 Thread 1
        0.02 Thread 2
        1.45 Thread 3
0.00          1.47 (23.90%)          6.15
```

Example: Late Sender



- Waiting time caused by a blocking receive operation posted earlier than the corresponding send
- Applies to blocking as well as non-blocking communication

Example: Late Broadcast



- Waiting times if the destination processes of a collective 1-to-N operation enter the operation earlier than the source process (root)
- Applies to: MPI_Bcast, MPI_Scatter, MPI_Scatterv

Hands-on exercise: NPB-MZ-MPI / BT

scalasca

Scalasca command

- One command for (almost) everything…

```
% scalasca
Scalasca 2.1
Toolset for scalable performance analysis of large-scale applications
usage: scalasca [OPTION]... ACTION <argument>...
    1. prepare application objects and executable for measurement:
       scalasca -instrument <compile-or-link-command> # skin (using scorep)
    2. run application under control of measurement system:
       scalasca -analyze <application-launch-command> # scan
    3. interactively explore measurement analysis report:
       scalasca -examine <experiment-archive|report>  # square

   -c, --show-config  show configuration and exit
   -h, --help         show this help and exit
   -n, --dry-run      show actions without taking them
       --quickref     show quick reference guide and exit
   -v, --verbose      enable verbose commentary
   -V, --version      show version information and exit
```

Scalasca compatibility command: *skin*

- Scalasca application instrumenter

```
% skin
Scalasca 2.1: application instrumenter using scorep
usage: skin [-v] [-comp] [-pdt] [-pomp] [-user] <compile-or-link-cmd>
   -comp={all|none|...}: routines to be instrumented by compiler
          (... custom instrumentation specification for compiler)
   -pdt:  process source files with PDT instrumenter
   -pomp: process source files for POMP directives
   -user: enable EPIK user instrumentation API macros in source code
   -v:    enable verbose commentary when instrumenting

   --*:   options to pass to Score-P instrumenter
```

  – Deprecated command
    - Provides compatibility with Scalasca 1.x
    - Prints corresponding Score-P instrumenter command
    - Helps in transitioning existing configurations
  – Recommended: use Score-P instrumenter directly

Scalasca convenience command: *scan*

- Scalasca measurement collection & analysis nexus

```
% scan
Scalasca 2.1: measurement collection & analysis nexus
usage: scan {options} [launchcmd [launchargs]] target [targetargs]
      where {options} may include:
  -h    Help: show this brief usage message and exit.
  -v    Verbose: increase verbosity.
  -n    Preview: show command(s) to be launched but don't execute.
  -q    Quiescent: execution with neither summarization nor tracing.
  -s    Summary: enable runtime summarization. [Default]
  -t    Tracing: enable trace collection and analysis.
  -a    Analyze: skip measurement to (re-)analyze an existing trace.
  -e exptdir   : Experiment archive to generate and/or analyze.
                 (overrides default experiment archive title)
  -f filtfile  : File specifying measurement filter.
  -l lockfile  : File that blocks start of measurement.
```

Scalasca convenience command: *square*

- Scalasca analysis report explorer

```
% square
Scalasca 2.1: analysis report explorer
usage: square [-v] [-s] [-f filtfile] [-F] <experiment archive
                                            | cube file>
   -c <none|quick|full>: Level of sanity checks for newly created reports
   -F                  : Force remapping of already existing reports
   -f filtfile         : Use specified filter file when doing scoring
   -s                  : Skip display and output textual score report
   -v                  : Enable verbose mode
```

Automatic measurement configuration

- **`scan`** configures Score-P measurement by setting some environment variables automatically
  - e.g., experiment title, profiling/tracing mode, filter file, …
  - Precedence order:
    - Command-line arguments
    - Environment variables already set
    - Automatically determined values
- Also, **`scan`** includes consistency checks and prevents corrupting existing experiment directories
- For tracing experiments, after trace collection completes then automatic parallel trace analysis is initiated
  - uses identical launch configuration to that used for measurement (i.e., the same allocated compute resources)

BT-MZ summary measurement

- Run the application using the Scalasca measurement collection & analysis nexus prefixed to launch command

```
% export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_W_4x4_sum
% OMP_NUM_THREADS=4 scan mpiexec -np 4 ./bt-mz_W.4
S=C=A=N: Scalasca 2.1 runtime summarization
S=C=A=N: ./scorep_bt-mz_W_4x4_sum experiment archive
S=C=A=N: Thu Jun 12 18:05:17 2014: Collect start
mpiexec -np 4 ./bt-mz_W.4

 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark

 Number of zones:   8 x   8
 Iterations: 200    dt:   0.000300
 Number of active processes:     4

 [... More application output ...]

S=C=A=N: Thu Jun 12 18:05:39 2014: Collect done (status=0) 22s
S=C=A=N: ./scorep_bt-mz_W_4x4_sum complete.
```

- Creates experiment directory ./scorep_bt-mz_W_4x4_sum

BT-MZ summary analysis report examination

- Score summary analysis report

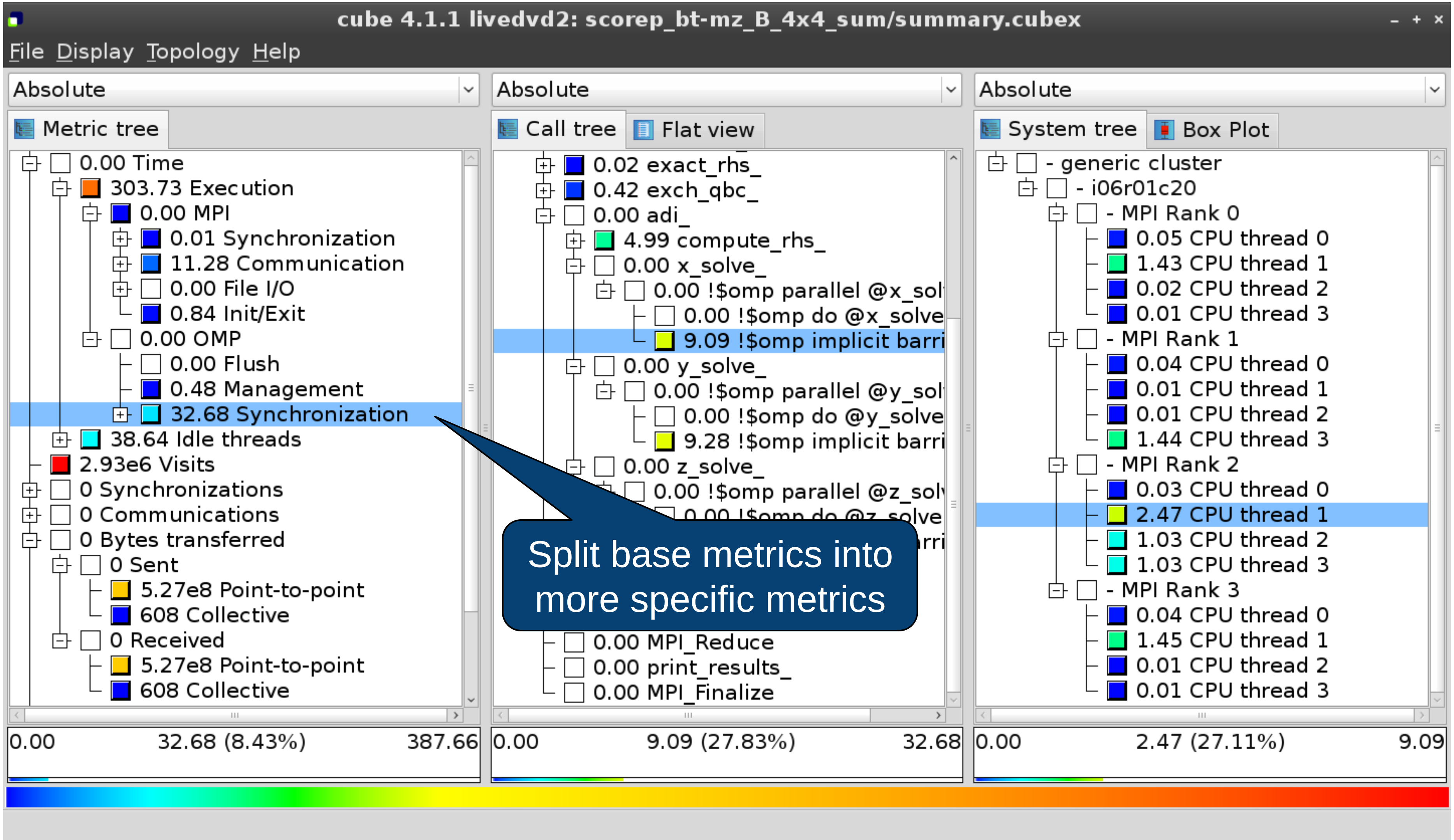
```
% square -s  scorep_bt-mz_W_4x4_sum
INFO: Post-processing runtime summarization result...
INFO: Score report written to ./scorep_bt-mz_W_4x4_sum/scorep.score
```

- Post-processing and interactive exploration with CUBE

```
% square  scorep_bt-mz_W_4x4_sum
INFO: Displaying ./scorep_bt-mz_W_4x4_sum/summary.cubex...

                  [GUI showing summary analysis report]
```

- The post-processing derives additional metrics and generates a structured metric hierarchy

Post-processed summary analysis report



cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_sum/summary.cubex

```
0.00 Time
  303.73 Execution
    0.00 MPI
      0.01 Synchronization
      11.28 Communication
      0.00 File I/O
      0.84 Init/Exit
    0.00 OMP
      0.00 Flush
      0.48 Management
      32.68 Synchronization
  38.64 Idle threads
2.93e6 Visits
0 Synchronizations
0 Communications
0 Bytes transferred
  0 Sent
    5.27e8 Point-to-point
    608 Collective
  0 Received
    5.27e8 Point-to-point
    608 Collective
```

```
0.00                32.68 (8.43%)              387.66
```

```
0.02 exact_rhs_
0.42 exch_qbc_
0.00 adi_
  4.99 compute_rhs_
  0.00 x_solve_
    0.00 !$omp parallel @x_sol
      0.00 !$omp do @x_solve
      9.09 !$omp implicit barri
  0.00 y_solve_
    0.00 !$omp parallel @y_sol
      0.00 !$omp do @y_solve
      9.28 !$omp implicit barri
  0.00 z_solve_
    0.00 !$omp parallel @z_solv
      0.00 !$omp do @z_solve
                           rri
0.00 MPI_Reduce
0.00 print_results_
0.00 MPI_Finalize
```

```
0.00                9.09 (27.83%)              32.68
```

```
- generic cluster
  - i06r01c20
    - MPI Rank 0
      0.05 CPU thread 0
      1.43 CPU thread 1
      0.02 CPU thread 2
      0.01 CPU thread 3
    - MPI Rank 1
      0.04 CPU thread 0
      0.01 CPU thread 1
      0.01 CPU thread 2
      1.44 CPU thread 3
    - MPI Rank 2
      0.03 CPU thread 0
      2.47 CPU thread 1
      1.03 CPU thread 2
      1.03 CPU thread 3
    - MPI Rank 3
      0.04 CPU thread 0
      1.45 CPU thread 1
      0.01 CPU thread 2
      0.01 CPU thread 3
```

```
0.00                2.47 (27.11%)              9.09
```

Split base metrics into more specific metrics

Performance Analysis Steps

1. Reference preparation for validation
2. Program instrumentation
3. Summary measurement collection
4. Summary analysis report examination
5. Summary experiment scoring
6. Summary measurement collection with filtering
7. Filtered summary analysis report examination
8. Event trace collection
9. Event trace examination & analysis

BT-MZ trace measurement collection...

- Re-run the application using Scalasca nexus with **“-t”** flag

```
% export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_W_4x4_trace
% OMP_NUM_THREADS=4 scan -t mpiexec -np 4 ./bt-mz_W.4
S=C=A=N: Scalasca 2.1 trace collection and analysis
S=C=A=N: ./scorep_bt-mz_W_4x4_trace experiment archive
S=C=A=N: Thu Jun 12 18:05:39 2014: Collect start
mpiexec -np 4 ./bt-mz_B.4
 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark

 Number of zones:   8 x   8
 Iterations: 200    dt:   0.000300
 Number of active processes:     4

 [... More application output ...]

S=C=A=N: Thu Jun 12 18:05:58 2014: Collect done (status=0) 19s
 [... continued ...]
```

BT-MZ trace measurement ... analysis

- Continues with automatic (parallel) analysis of trace files

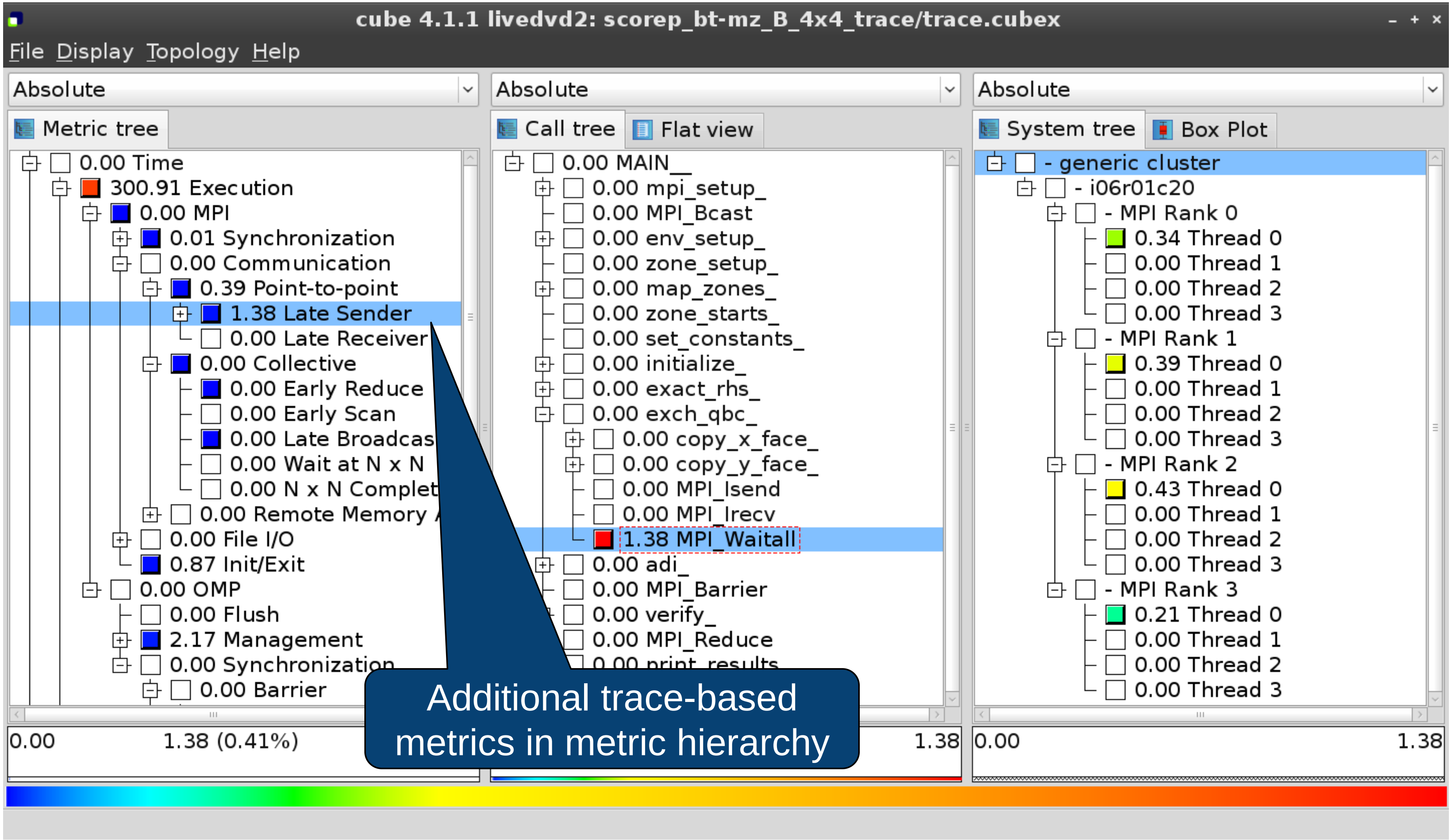
```
S=C=A=N: Thu Jun 12 18:05:58 2014: Analyze start
mpiexec -np 4 scout.hyb ./scorep_bt-mz_W_4x4_trace/traces.otf2
SCOUT   Copyright (c) 1998-2012 Forschungszentrum Juelich GmbH
        Copyright (c) 2009-2012 German Research School for Simulation
                                Sciences GmbH

Analyzing experiment archive ./scorep_bt-mz_W_4x4_trace/traces.otf2

Opening experiment archive ... done (0.002s).
Reading definition data    ... done (0.004s).
Reading event trace data   ... done (0.130s).
Preprocessing              ... done (0.259s).
Analyzing trace data       ...
  Wait-state detection (fwd)      (1/4) ... done (0.575s).
  Wait-state detection (bwd)      (2/4) ... done (0.138s).
  Synchpoint exchange             (3/4) ... done (0.358s).
  Critical-path analysis          (4/4) ... done (0.288s).
done (1.360s).
Writing analysis report    ... done (0.121s).

Total processing time     : 1.924s
S=C=A=N: Thu Jun 12 18:06:00 2014: Analyze done (status=0) 2s
```

BT-MZ trace analysis report exploration

- Produces trace analysis report in experiment directory containing trace-based wait-state metrics

```
% square  scorep_bt-mz_W_4x4_trace
INFO: Post-processing runtime summarization result...
INFO: Post-processing trace analysis report...
INFO: Displaying ./scorep_bt-mz_W_4x4_trace/trace.cubex...

                [GUI showing trace analysis report]
```

Post-processed trace analysis report



cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex

Metric tree:

```
□ 0.00 Time
  🟥 300.91 Execution
    🟦 0.00 MPI
      🟦 0.01 Synchronization
      □ 0.00 Communication
        🟦 0.39 Point-to-point
          🟦 1.38 Late Sender
          □ 0.00 Late Receiver
        🟦 0.00 Collective
          🟦 0.00 Early Reduce
          □ 0.00 Early Scan
          🟦 0.00 Late Broadcas
          □ 0.00 Wait at N x N
          □ 0.00 N x N Complet
        □ 0.00 Remote Memory /
      □ 0.00 File I/O
      🟦 0.87 Init/Exit
    □ 0.00 OMP
      □ 0.00 Flush
      🟦 2.17 Management
      □ 0.00 Synchronization
        □ 0.00 Barrier
```

Call tree:

```
□ 0.00 MAIN__
  □ 0.00 mpi_setup_
  □ 0.00 MPI_Bcast
  □ 0.00 env_setup_
  □ 0.00 zone_setup_
  □ 0.00 map_zones_
  □ 0.00 zone_starts_
  □ 0.00 set_constants_
  □ 0.00 initialize_
  □ 0.00 exact_rhs_
  □ 0.00 exch_qbc_
    □ 0.00 copy_x_face_
    □ 0.00 copy_y_face_
    □ 0.00 MPI_Isend
    □ 0.00 MPI_Irecv
    🟥 1.38 MPI_Waitall
  □ 0.00 adi_
  □ 0.00 MPI_Barrier
  □ 0.00 verify_
  □ 0.00 MPI_Reduce
  □ 0.00 print_results_
```

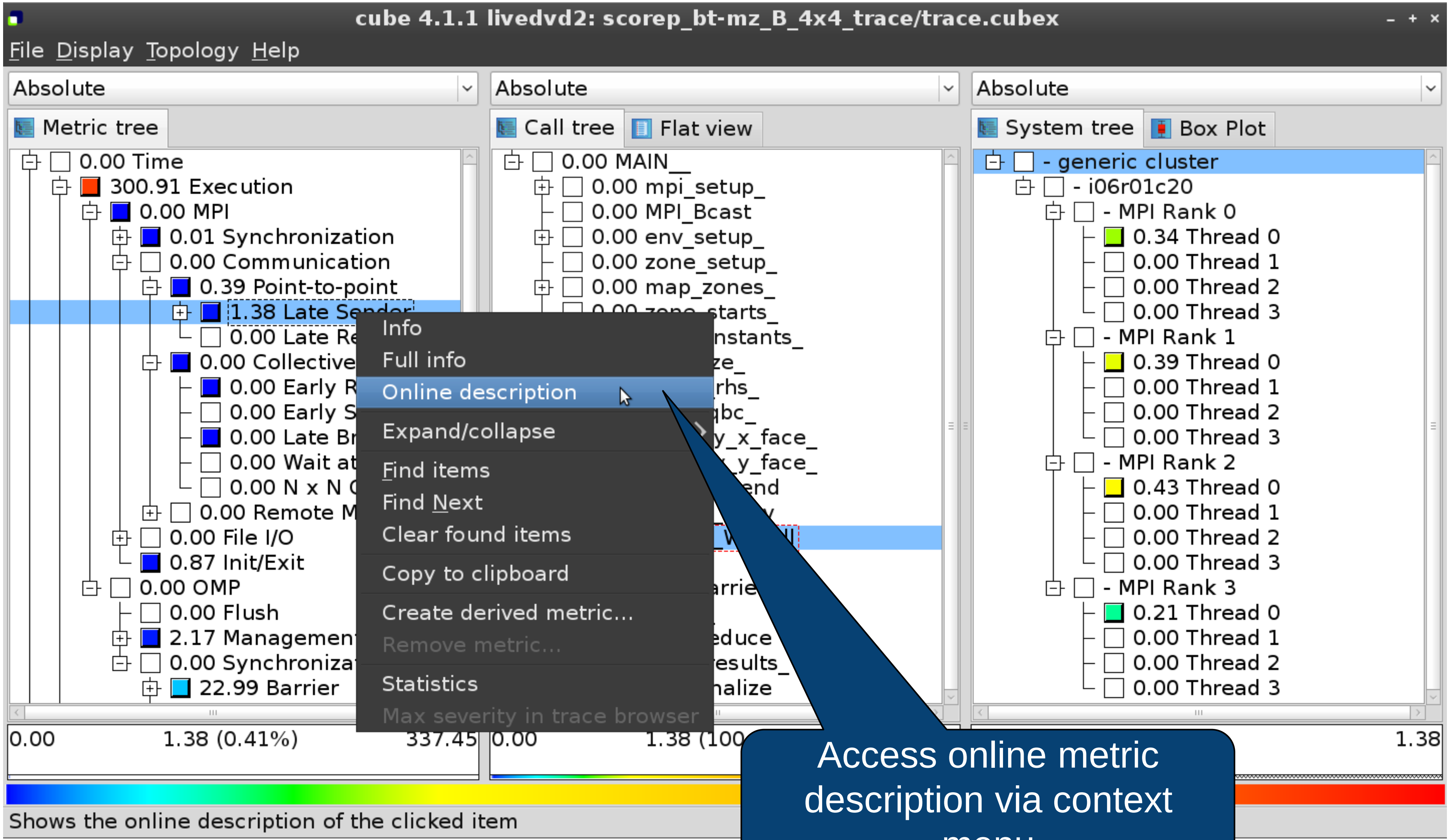
System tree:

```
□ - generic cluster
  □ - i06r01c20
    □ - MPI Rank 0
      🟩 0.34 Thread 0
      □ 0.00 Thread 1
      □ 0.00 Thread 2
      □ 0.00 Thread 3
    □ - MPI Rank 1
      🟨 0.39 Thread 0
      □ 0.00 Thread 1
      □ 0.00 Thread 2
      □ 0.00 Thread 3
    □ - MPI Rank 2
      🟨 0.43 Thread 0
      □ 0.00 Thread 1
      □ 0.00 Thread 2
      □ 0.00 Thread 3
    □ - MPI Rank 3
      🟩 0.21 Thread 0
      □ 0.00 Thread 1
      □ 0.00 Thread 2
      □ 0.00 Thread 3
```

```
0.00          1.38 (0.41%)          1.38    0.00          1.38
```

Additional trace-based metrics in metric hierarchy

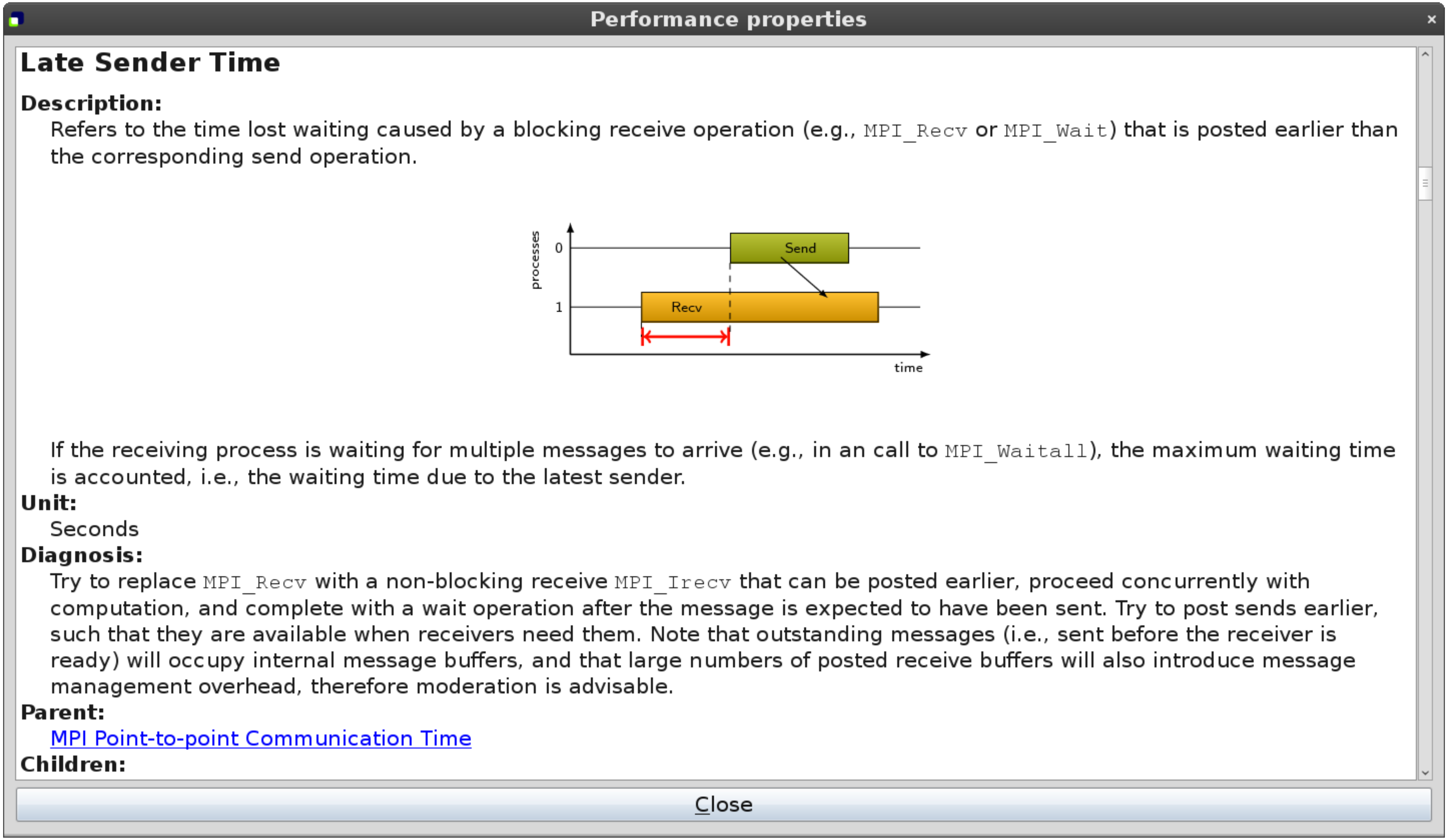
Online metric description



Shows the online description of the clicked item

Access online metric description via context menu

Online metric description



Late Sender Time

Description:

Refers to the time lost waiting caused by a blocking receive operation (e.g., `MPI_Recv` or `MPI_Wait`) that is posted earlier than the corresponding send operation.

If the receiving process is waiting for multiple messages to arrive (e.g., in an call to `MPI_Waitall`), the maximum waiting time is accounted, i.e., the waiting time due to the latest sender.

Unit:

Seconds

Diagnosis:

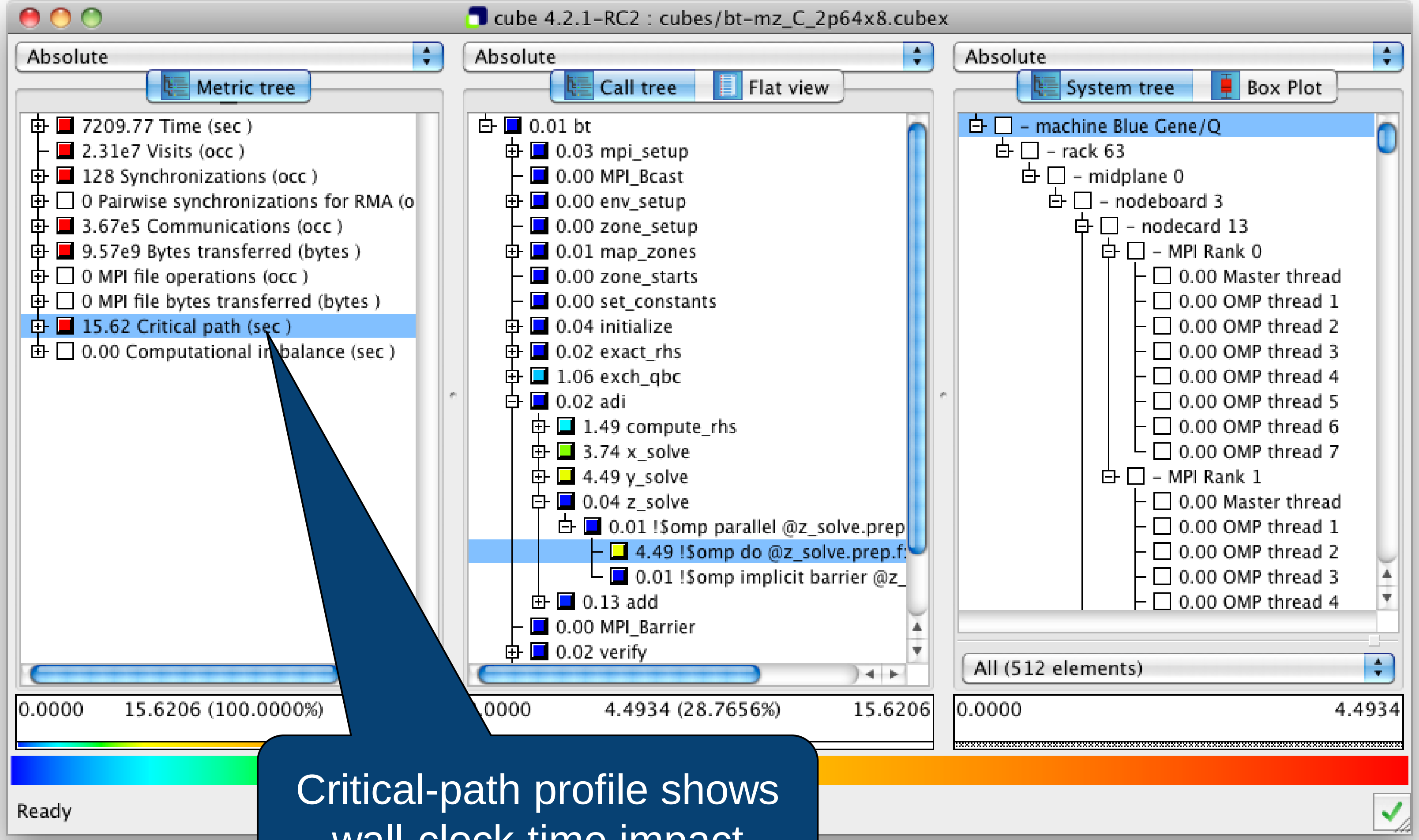
Try to replace `MPI_Recv` with a non-blocking receive `MPI_Irecv` that can be posted earlier, proceed concurrently with computation, and complete with a wait operation after the message is expected to have been sent. Try to post sends earlier, such that they are available when receivers need them. Note that outstanding messages (i.e., sent before the receiver is ready) will occupy internal message buffers, and that large numbers of posted receive buffers will also introduce message management overhead, therefore moderation is advisable.

Parent:

MPI Point-to-point Communication Time

Children:

Critical-path analysis



cube 4.2.1-RC2 : cubes/bt-mz_C_2p64x8.cubex

Absolute | Metric tree

```
⊞ 7209.77 Time (sec )
  2.31e7 Visits (occ )
⊞ 128 Synchronizations (occ )
⊞ 0 Pairwise synchronizations for RMA (o
⊞ 3.67e5 Communications (occ )
⊞ 9.57e9 Bytes transferred (bytes )
⊞ 0 MPI file operations (occ )
⊞ 0 MPI file bytes transferred (bytes )
⊞ 15.62 Critical path (sec )
⊞ 0.00 Computational imbalance (sec )
```

Absolute | Call tree | Flat view

```
⊟ 0.01 bt
   ⊞ 0.03 mpi_setup
     0.00 MPI_Bcast
   ⊞ 0.00 env_setup
     0.00 zone_setup
   ⊞ 0.01 map_zones
     0.00 zone_starts
     0.00 set_constants
   ⊞ 0.04 initialize
   ⊞ 0.02 exact_rhs
   ⊞ 1.06 exch_qbc
   ⊟ 0.02 adi
      ⊞ 1.49 compute_rhs
      ⊞ 3.74 x_solve
      ⊞ 4.49 y_solve
      ⊟ 0.04 z_solve
         ⊟ 0.01 !$omp parallel @z_solve.prep
              4.49 !$omp do @z_solve.prep.f:
              0.01 !$omp implicit barrier @z_
      ⊞ 0.13 add
     0.00 MPI_Barrier
   ⊞ 0.02 verify
```

Absolute | System tree | Box Plot

```
⊟ □ - machine Blue Gene/Q
   ⊟ □ - rack 63
      ⊟ □ - midplane 0
         ⊟ □ - nodeboard 3
            ⊟ □ - nodecard 13
               ⊟ □ - MPI Rank 0
                    □ 0.00 Master thread
                    □ 0.00 OMP thread 1
                    □ 0.00 OMP thread 2
                    □ 0.00 OMP thread 3
                    □ 0.00 OMP thread 4
                    □ 0.00 OMP thread 5
                    □ 0.00 OMP thread 6
                    □ 0.00 OMP thread 7
               ⊟ □ - MPI Rank 1
                    □ 0.00 Master thread
                    □ 0.00 OMP thread 1
                    □ 0.00 OMP thread 2
                    □ 0.00 OMP thread 3
                    □ 0.00 OMP thread 4
```

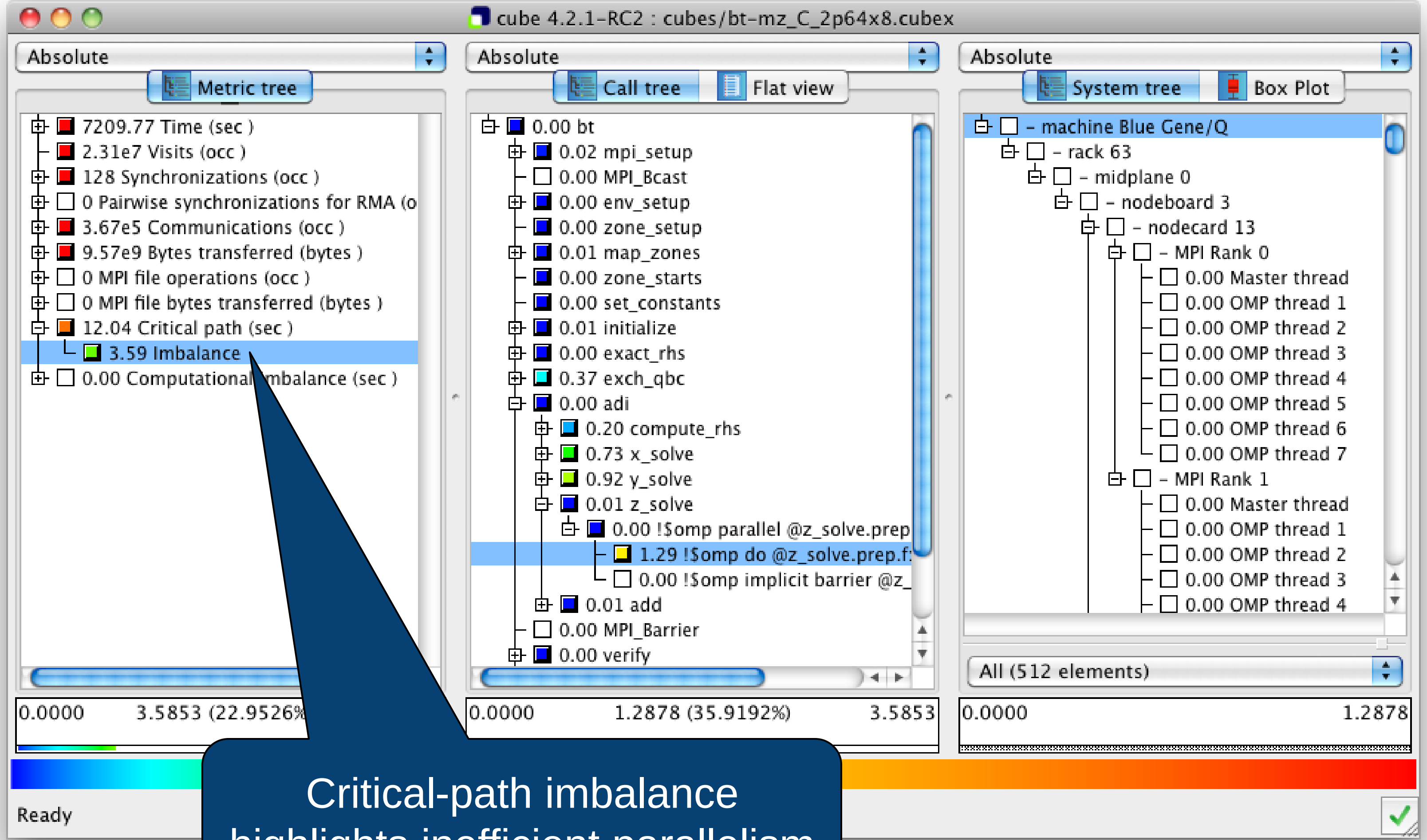
All (512 elements)

```
0.0000    15.6206 (100.0000%)
.0000     4.4934 (28.7656%)    15.6206
0.0000                         4.4934
```

Ready

Critical-path profile shows wall-clock time impact

Critical-path analysis

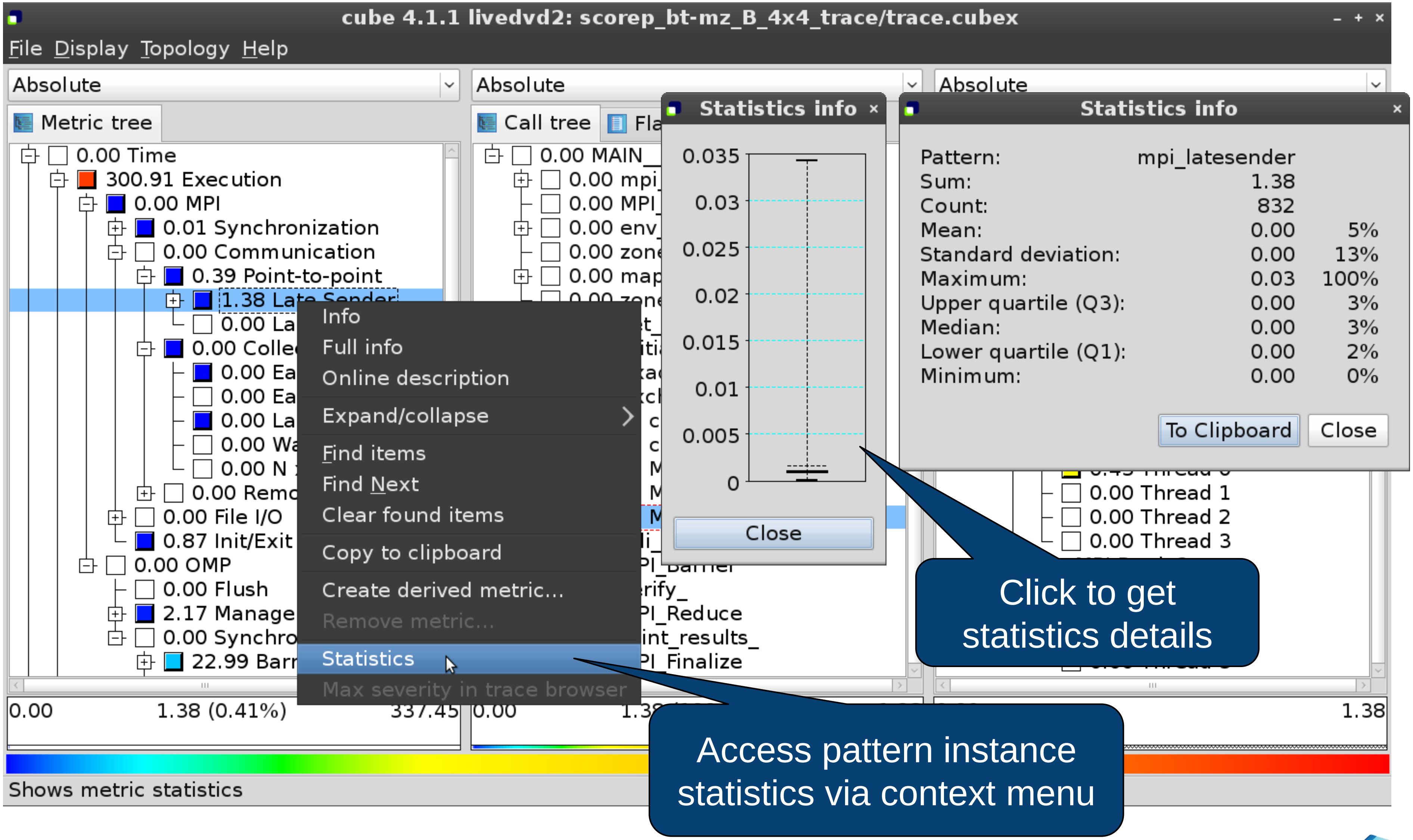


cube 4.2.1-RC2 : cubes/bt-mz_C_2p64x8.cubex

```
Absolute                                  Absolute                                        Absolute
Metric tree                               Call tree | Flat view                           System tree | Box Plot

⊞ 7209.77 Time (sec )                     ⊟ 0.00 bt                                       ⊟ □ - machine Blue Gene/Q
  2.31e7 Visits (occ )                      ⊞ 0.02 mpi_setup                                ⊟ □ - rack 63
⊞ 128 Synchronizations (occ )                 0.00 MPI_Bcast                                  ⊟ □ - midplane 0
⊞ □ 0 Pairwise synchronizations for RMA (o  ⊞ 0.00 env_setup                                    ⊟ □ - nodeboard 3
⊞ 3.67e5 Communications (occ )                0.00 zone_setup                                     ⊟ □ - nodecard 13
⊞ 9.57e9 Bytes transferred (bytes )         ⊞ 0.01 map_zones                                        ⊟ □ - MPI Rank 0
⊞ □ 0 MPI file operations (occ )              0.00 zone_starts                                          □ 0.00 Master thread
⊞ □ 0 MPI file bytes transferred (bytes )     0.00 set_constants                                        □ 0.00 OMP thread 1
⊟ 12.04 Critical path (sec )                ⊞ 0.01 initialize                                           □ 0.00 OMP thread 2
    3.59 Imbalance                          ⊞ 0.00 exact_rhs                                            □ 0.00 OMP thread 3
⊞ □ 0.00 Computational imbalance (sec )     ⊞ 0.37 exch_qbc                                             □ 0.00 OMP thread 4
                                            ⊟ 0.00 adi                                                  □ 0.00 OMP thread 5
                                              ⊞ 0.20 compute_rhs                                        □ 0.00 OMP thread 6
                                              ⊞ 0.73 x_solve                                            □ 0.00 OMP thread 7
                                              ⊞ 0.92 y_solve                                        ⊟ □ - MPI Rank 1
                                              ⊟ 0.01 z_solve                                            □ 0.00 Master thread
                                                ⊟ 0.00 !$omp parallel @z_solve.prep                     □ 0.00 OMP thread 1
                                                    1.29 !$omp do @z_solve.prep.f:                      □ 0.00 OMP thread 2
                                                    □ 0.00 !$omp implicit barrier @z_                   □ 0.00 OMP thread 3
                                              ⊞ 0.01 add                                                □ 0.00 OMP thread 4
                                              □ 0.00 MPI_Barrier
                                            ⊞ 0.00 verify
                                                                                          All (512 elements)

0.0000    3.5853 (22.9526%)               0.0000    1.2878 (35.9192%)    3.5853           0.0000                 1.2878

Ready
```

Critical-path imbalance highlights inefficient parallelism

Pattern instance statistics



Statistics info

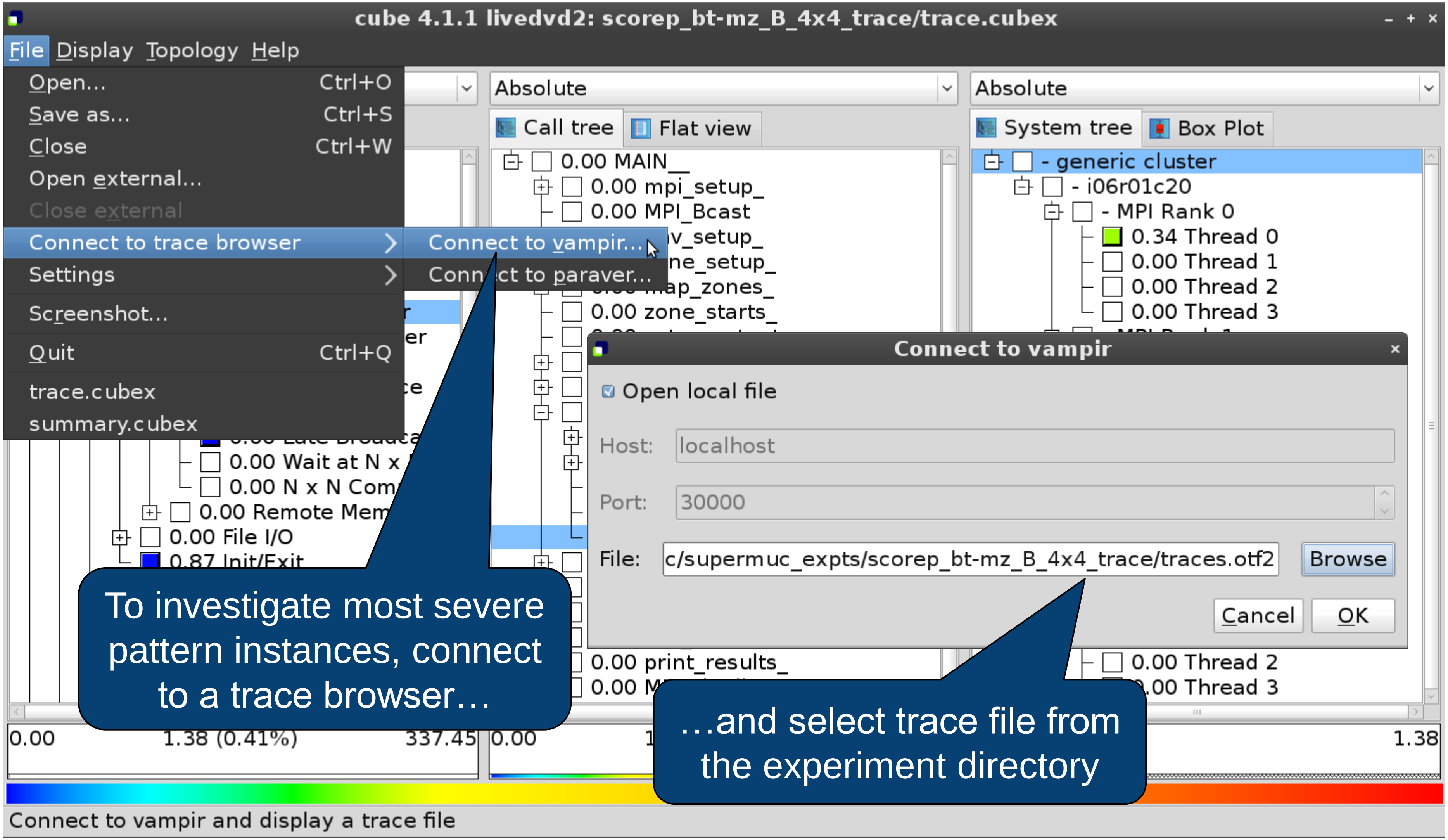
Pattern:	mpi_latesender	
Sum:	1.38	
Count:	832	
Mean:	0.00	5%
Standard deviation:	0.00	13%
Maximum:	0.03	100%
Upper quartile (Q3):	0.00	3%
Median:	0.00	3%
Lower quartile (Q1):	0.00	2%
Minimum:	0.00	0%

To Clipboard Close

Click to get statistics details

Access pattern instance statistics via context menu

Connect to Vampir trace browser



cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex

Connect to vampir

☑ Open local file

Host: localhost

Port: 30000

File: c/supermuc_expts/scorep_bt-mz_B_4x4_trace/traces.otf2

Browse

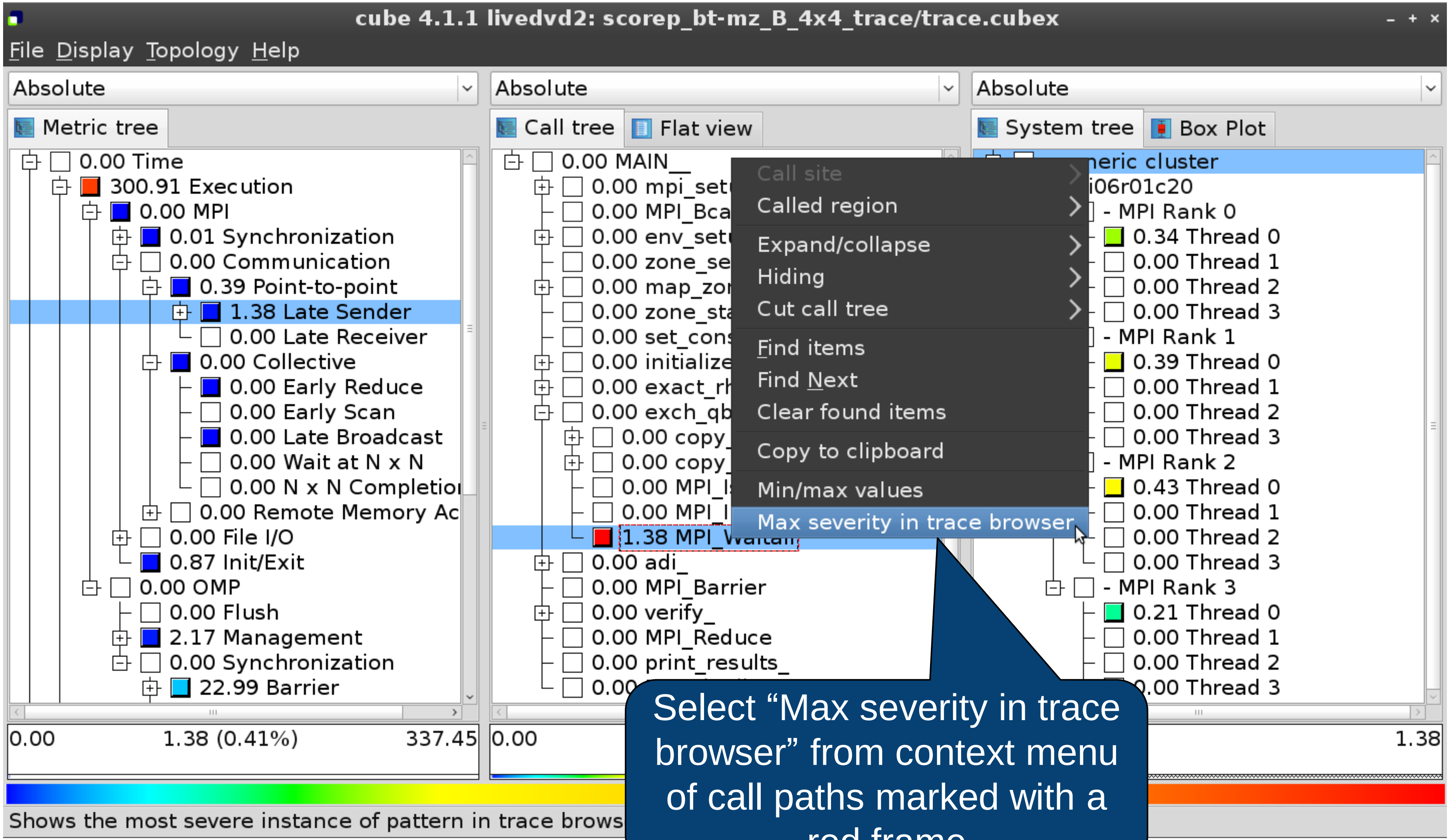
Cancel OK

Connect to vampir and display a trace file

To investigate most severe pattern instances, connect to a trace browser…

…and select trace file from the experiment directory

Show most severe pattern instances



cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex

```
0.00 Time
  300.91 Execution
    0.00 MPI
      0.01 Synchronization
      0.00 Communication
        0.39 Point-to-point
          1.38 Late Sender
          0.00 Late Receiver
        0.00 Collective
          0.00 Early Reduce
          0.00 Early Scan
          0.00 Late Broadcast
          0.00 Wait at N x N
          0.00 N x N Completio
        0.00 Remote Memory Ac
      0.00 File I/O
      0.87 Init/Exit
    0.00 OMP
      0.00 Flush
      2.17 Management
      0.00 Synchronization
        22.99 Barrier
```

```
0.00 MAIN__
  0.00 mpi_set
  0.00 MPI_Bca
  0.00 env_set
  0.00 zone_se
  0.00 map_zor
  0.00 zone_sta
  0.00 set_cons
  0.00 initialize
  0.00 exact_rh
  0.00 exch_qb
    0.00 copy_
    0.00 copy_
    0.00 MPI_I
    0.00 MPI_I
    1.38 MPI_Waitall
  0.00 adi_
  0.00 MPI_Barrier
  0.00 verify_
  0.00 MPI_Reduce
  0.00 print_results_
  0.00
```

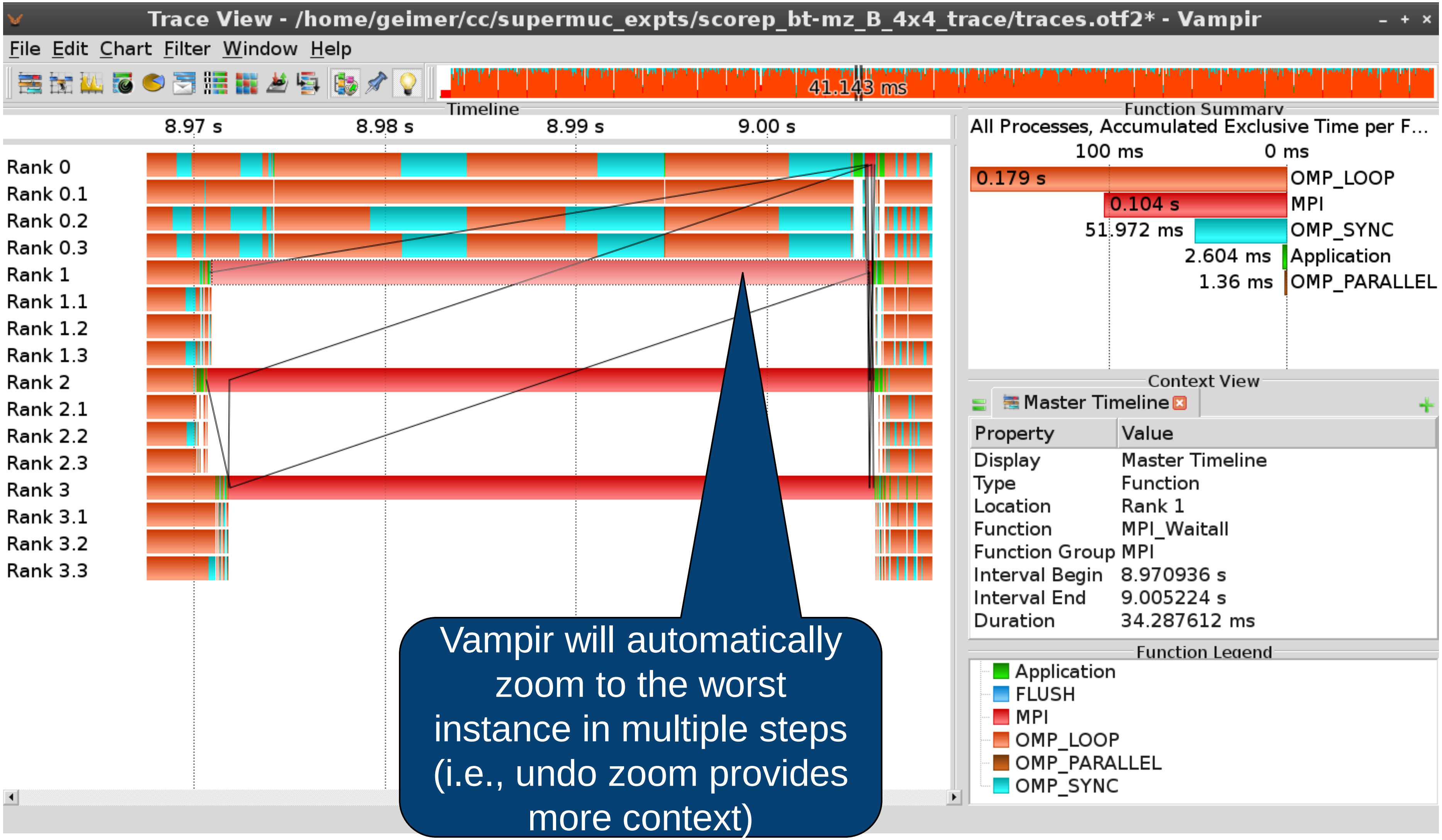
```
eric cluster
i06r01c20
  - MPI Rank 0
    0.34 Thread 0
    0.00 Thread 1
    0.00 Thread 2
    0.00 Thread 3
  - MPI Rank 1
    0.39 Thread 0
    0.00 Thread 1
    0.00 Thread 2
    0.00 Thread 3
  - MPI Rank 2
    0.43 Thread 0
    0.00 Thread 1
    0.00 Thread 2
    0.00 Thread 3
  - MPI Rank 3
    0.21 Thread 0
    0.00 Thread 1
    0.00 Thread 2
    0.00 Thread 3
```

0.00 1.38 (0.41%) 337.45 | 0.00 | 1.38

Shows the most severe instance of pattern in trace brows

Select “Max severity in trace browser” from context menu of call paths marked with a red frame

Investigate most severe instance in Vampir



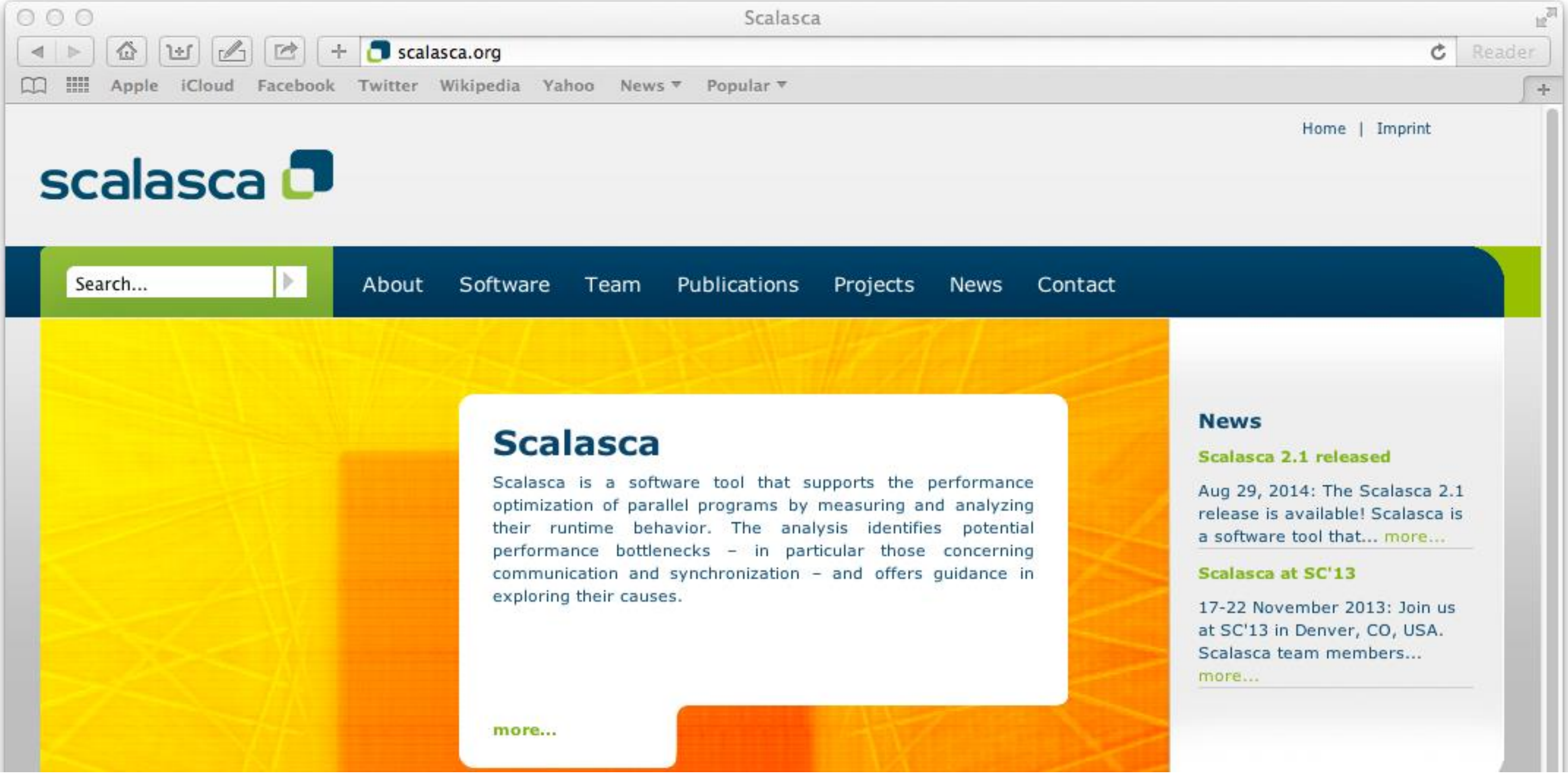
Property	Value
Display	Master Timeline
Type	Function
Location	Rank 1
Function	MPI_Waitall
Function Group	MPI
Interval Begin	8.970936 s
Interval End	9.005224 s
Duration	34.287612 ms

Vampir will automatically zoom to the worst instance in multiple steps (i.e., undo zoom provides more context)

Further information

Website: www.scalasca.org

User support: scalasca@fz-juelich.de



Scalasca
scalasca.org
Reader
Apple iCloud Facebook Twitter Wikipedia Yahoo News Popular
Home | Imprint
scalasca
Search...
About Software Team Publications Projects News Contact

Scalasca

Scalasca is a software tool that supports the performance optimization of parallel programs by measuring and analyzing their runtime behavior. The analysis identifies potential performance bottlenecks – in particular those concerning communication and synchronization – and offers guidance in exploring their causes.

more...

News

Scalasca 2.1 released

Aug 29, 2014: The Scalasca 2.1 release is available! Scalasca is a software tool that... more...

Scalasca at SC'13

17-22 November 2013: Join us at SC'13 in Denver, CO, USA. Scalasca team members... more...

Virtual Institute – High Productivity Supercomputing

VI-HPS

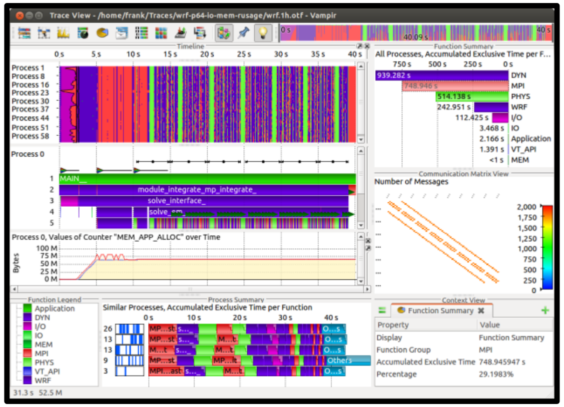
Performance Analysis with Vampir

Bert Wesarg, Andreas Knüpfer

ZIH, Technische Universität Dresden

Mission

- Visualization of dynamics of complex parallel processes
- Full details for arbitrary temporal and spatial levels
- Supplement to automatic analysis



Property	Value
Display	Function Summary
Function Group	MPI
Accumulated Exclusive Time	748.945947 s
Percentage	29.1983%

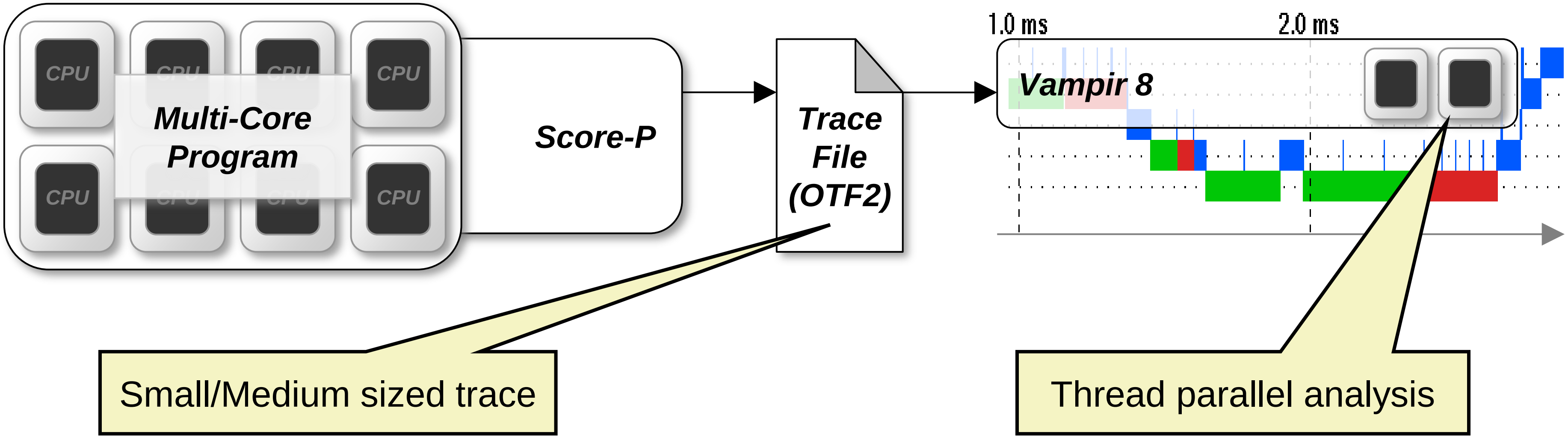
Typical questions that Vampir helps to answer:

- What happens in my application execution during a given time in a given process or thread?
- How do the communication patterns of my application execute on a real system?
- Are there any imbalances in computation, I/O or memory usage and how do they affect the parallel execution of my application?

Vampir – Visualization Modes (1)

- Directly on front end or local machine

```
% vampir
```



Small/Medium sized trace

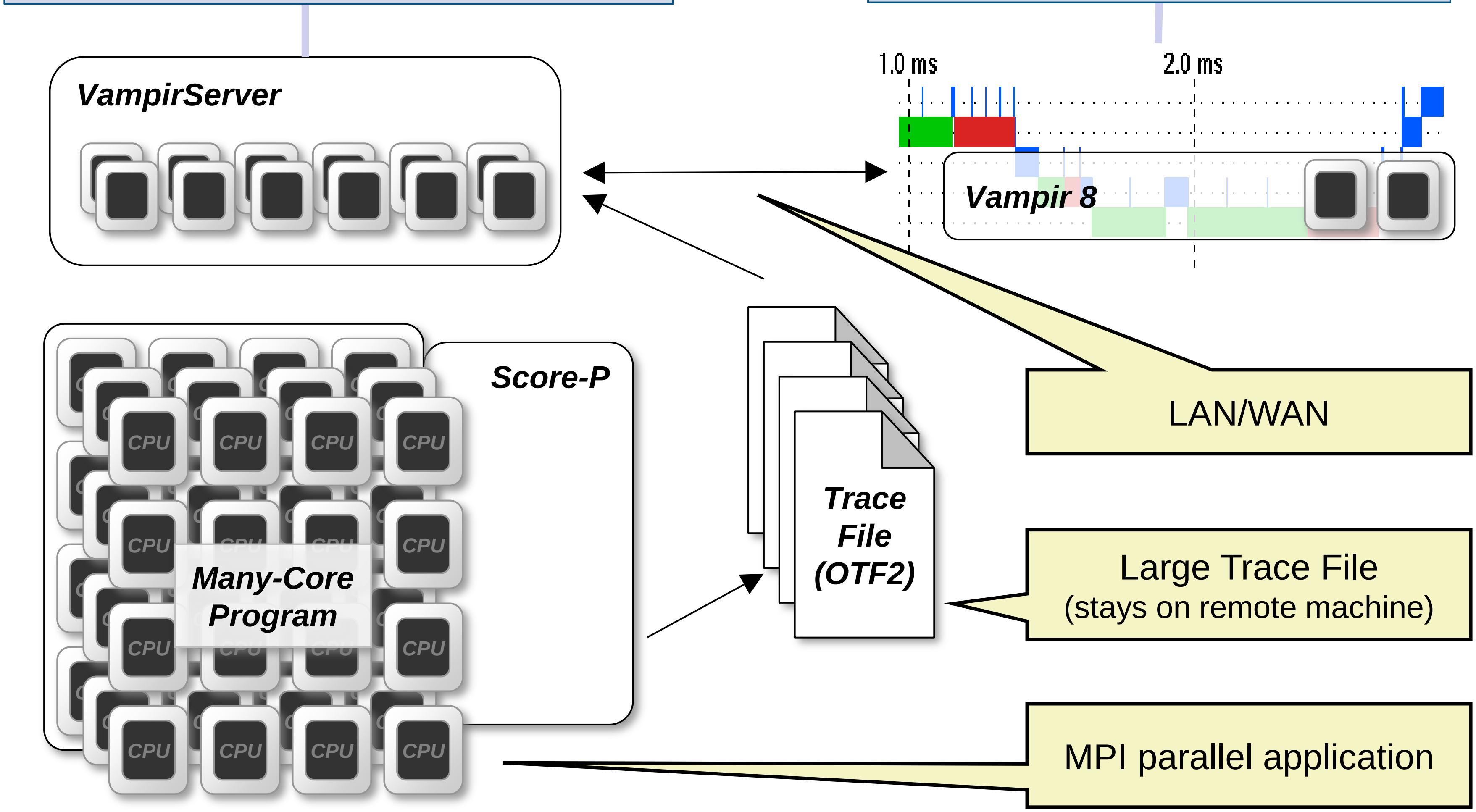
Thread parallel analysis

Vampir – Visualization Modes (2)

- On local machine with remote VampirServer

```
% vampirserver start –n 12
```

```
% vampir
```



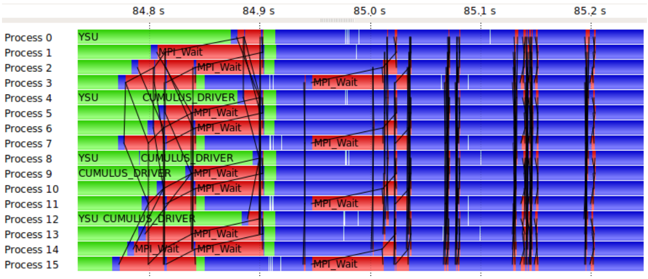
Product overview

- Vampir & VampirServer
  - Interactive trace visualization and analysis
  - Intuitive browsing and zooming
  - Scalable to large trace data sizes (20 TByte)
  - Scalable to high parallelism (200000 processes)

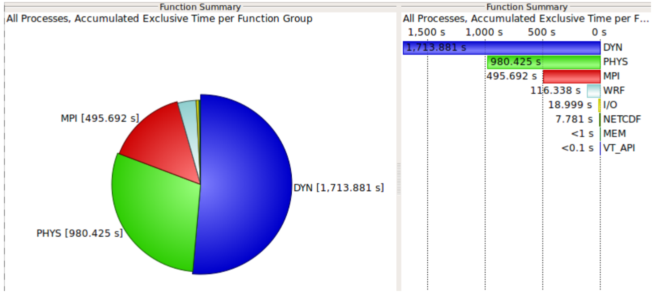
- Vampir is available for Linux, Windows and Mac OS X

The Main Displays of Vampir

- **Timeline Charts:**
  - Master Timeline
  - Process Timeline
  - Counter Data Timeline
  - Performance Radar
- **Summary Charts:**
  - Function Summary
  - Message Summary
  - Process Summary
  - Communication Matrix View



Show application activities and communication along a time axis



Provide quantitative results for the currently selected time interval

Vampir hands-on

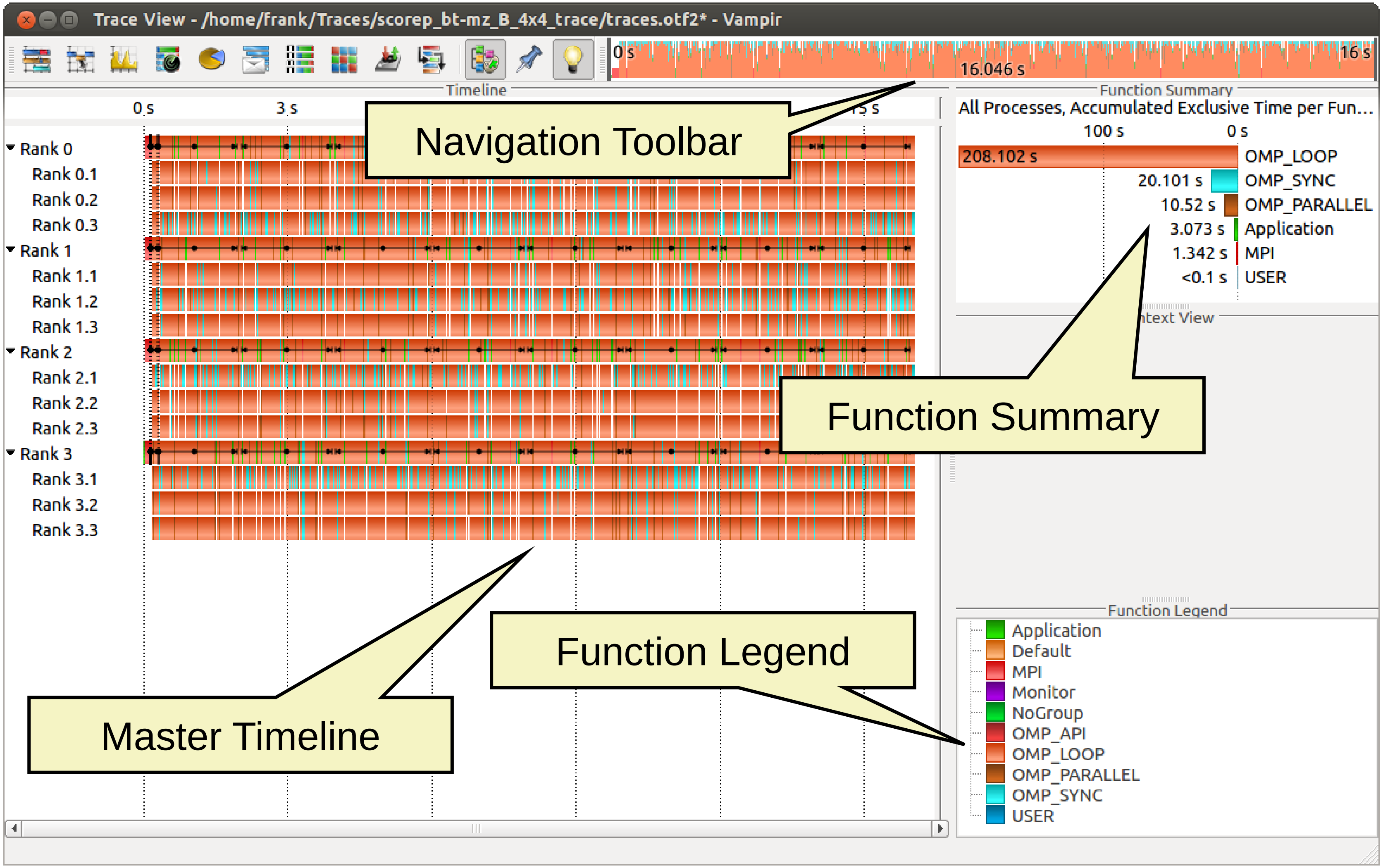
Visualizing and analyzing NPB-MZ-MPI / BT

Performance Analysis Steps

1. Reference preparation for validation
2. Program instrumentation
3. Summary measurement collection
4. Summary analysis report examination
5. Summary experiment scoring
6. Summary measurement collection with filtering
7. Filtered summary analysis report examination
8. Event trace collection
9. Event trace examination & analysis

Vampir: Visualization of the NPB-MZ-MPI / BT trace

```
% vampir scorep_bt-mz_B_4x4_trace
```



Trace View - /home/frank/Traces/scorep_bt-mz_B_4x4_trace/traces.otf2* - Vampir

All Processes, Accumulated Exclusive Time per Fun…

208.102 s	OMP_LOOP
20.101 s	OMP_SYNC
10.52 s	OMP_PARALLEL
3.073 s	Application
1.342 s	MPI
<0.1 s	USER

- 🟩 Application
- 🟧 Default
- 🟥 MPI
- 🟪 Monitor
- 🟩 NoGroup
- 🟥 OMP_API
- 🟧 OMP_LOOP
- 🟫 OMP_PARALLEL
- 🟦 OMP_SYNC
- 🟦 USER

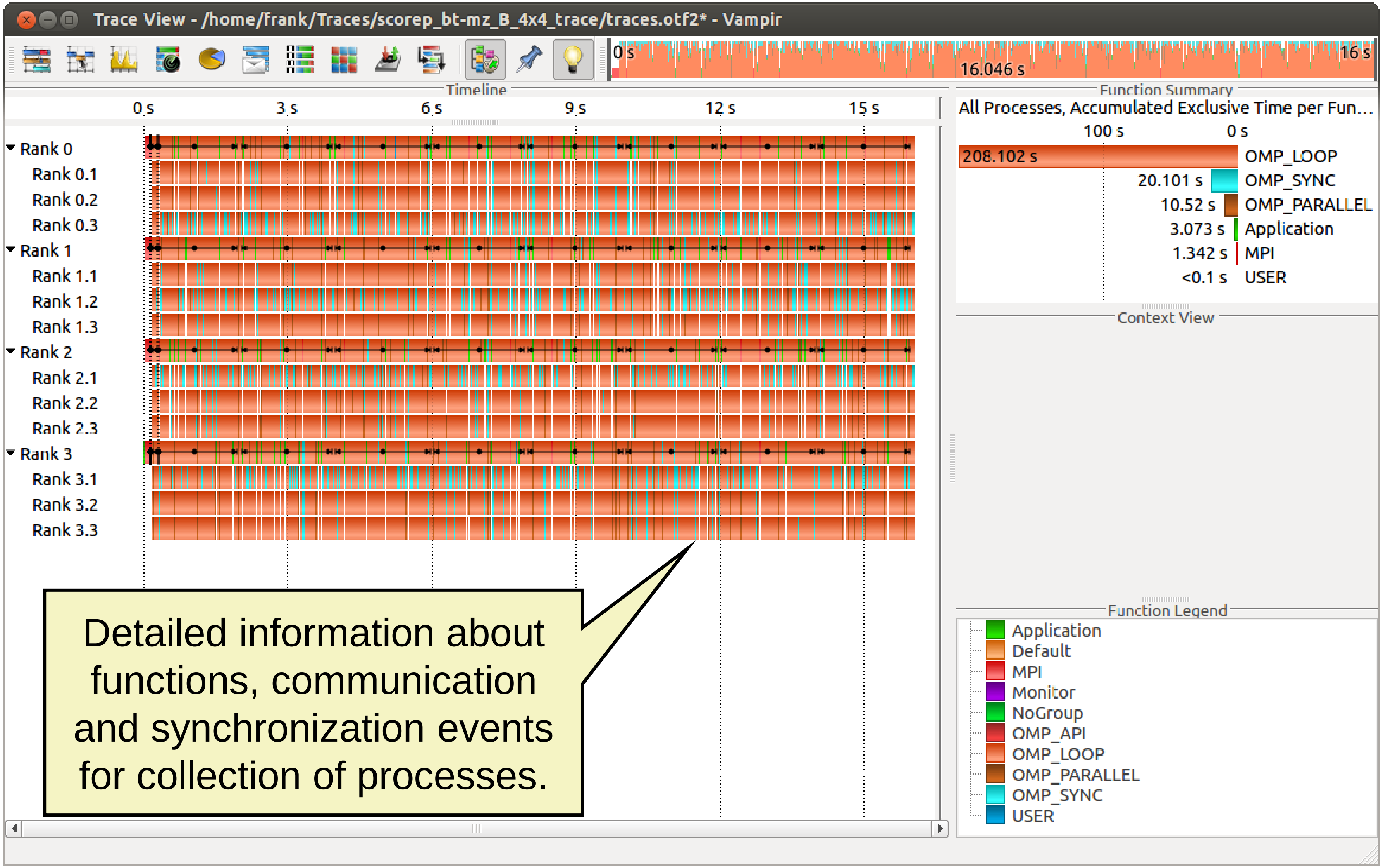
Navigation Toolbar

Function Summary

Function Legend

Master Timeline

Master Timeline



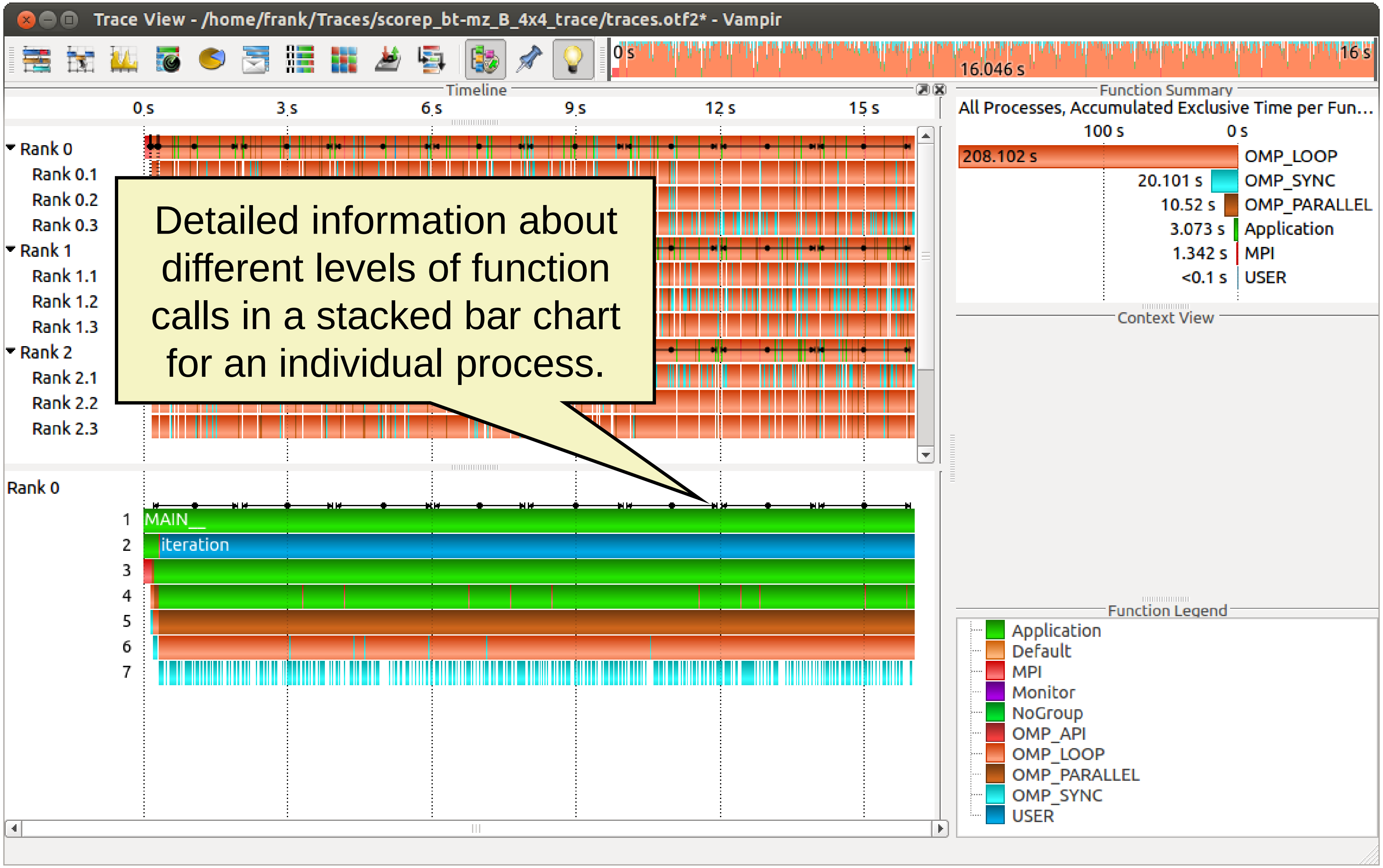
Trace View - /home/frank/Traces/scorep_bt-mz_B_4x4_trace/traces.otf2* - Vampir

208.102 s	OMP_LOOP
20.101 s	OMP_SYNC
10.52 s	OMP_PARALLEL
3.073 s	Application
1.342 s	MPI
<0.1 s	USER

- 🟩 Application
- 🟧 Default
- 🟥 MPI
- 🟪 Monitor
- 🟩 NoGroup
- 🟥 OMP_API
- 🟧 OMP_LOOP
- 🟫 OMP_PARALLEL
- 🟦 OMP_SYNC
- 🟦 USER

Detailed information about functions, communication and synchronization events for collection of processes.

Process Timeline



Trace View - /home/frank/Traces/scorep_bt-mz_B_4x4_trace/traces.otf2* - Vampir

All Processes, Accumulated Exclusive Time per Fun...

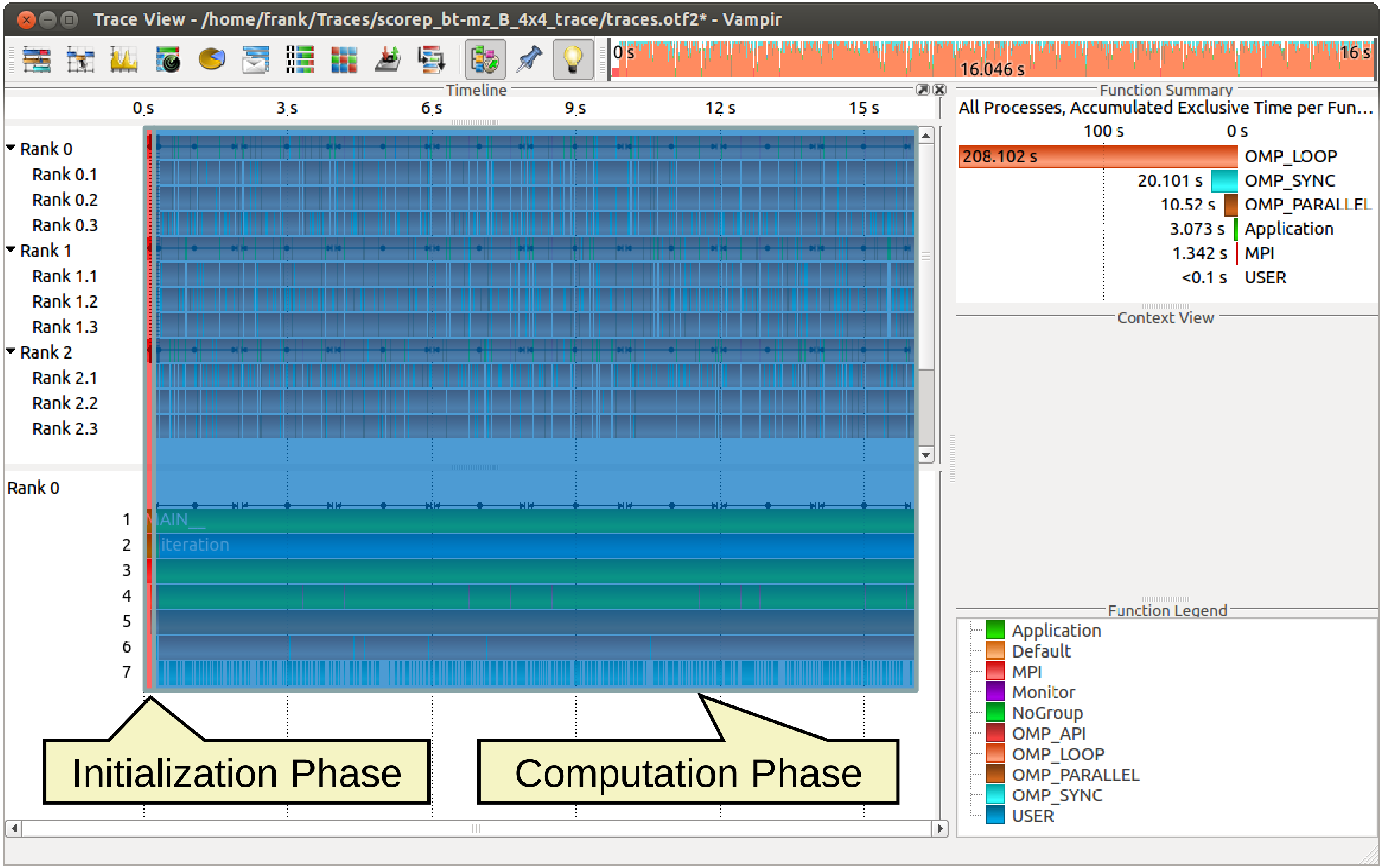
100 s	0 s
208.102 s	OMP_LOOP
20.101 s	OMP_SYNC
10.52 s	OMP_PARALLEL
3.073 s	Application
1.342 s	MPI
<0.1 s	USER

- 🟩 Application
- 🟧 Default
- 🟥 MPI
- 🟪 Monitor
- 🟩 NoGroup
- 🟥 OMP_API
- 🟧 OMP_LOOP
- 🟫 OMP_PARALLEL
- 🟦 OMP_SYNC
- 🟦 USER

Detailed information about different levels of function calls in a stacked bar chart for an individual process.

Vampir: Visualization of the NPB-MZ-MPI / BT trace

Typical program phases



Trace View - /home/frank/Traces/scorep_bt-mz_B_4x4_trace/traces.otf2* - Vampir

Function Summary

All Processes, Accumulated Exclusive Time per Fun...

Time	Function
208.102 s	OMP_LOOP
20.101 s	OMP_SYNC
10.52 s	OMP_PARALLEL
3.073 s	Application
1.342 s	MPI
<0.1 s	USER

Function Legend

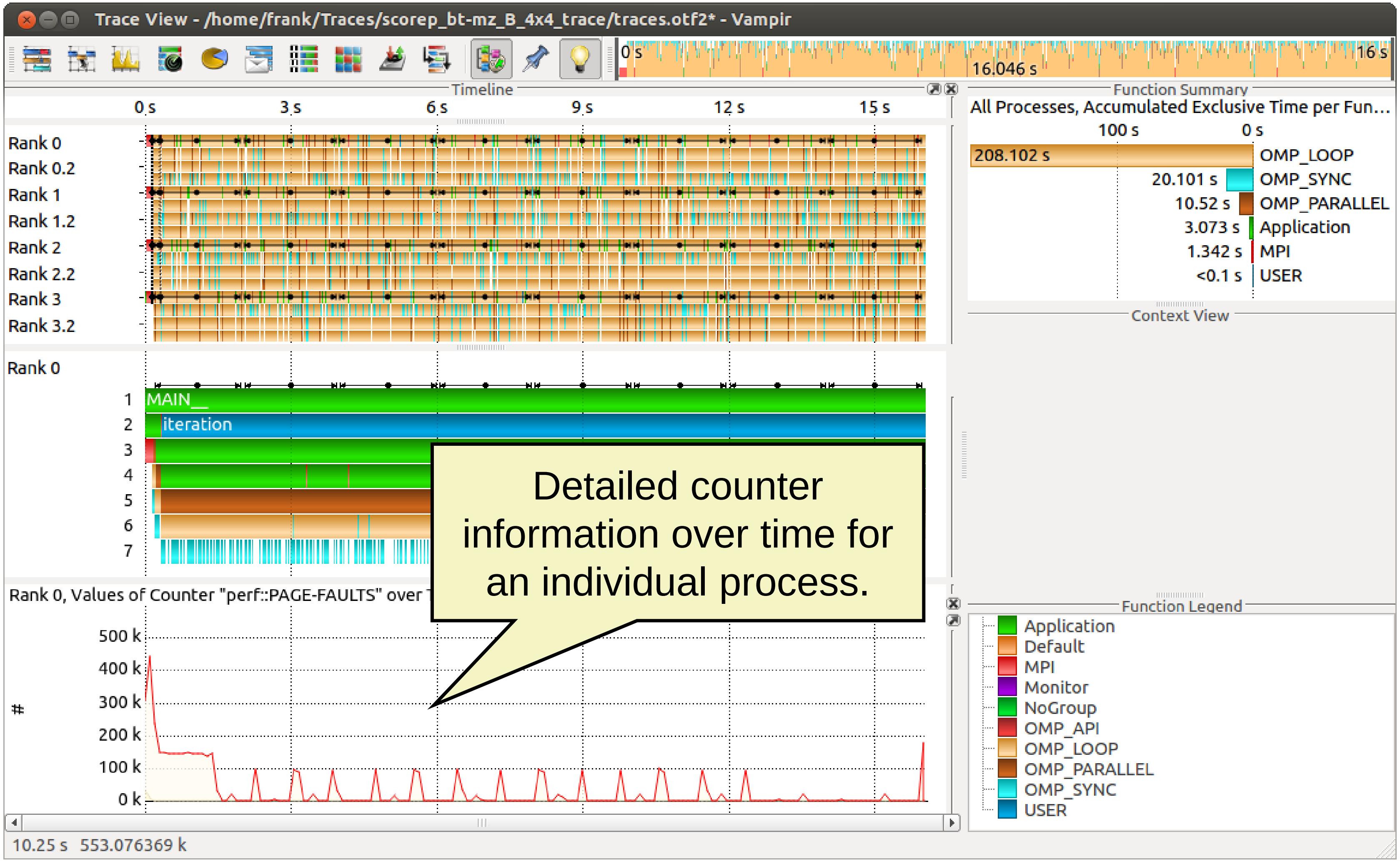
- 🟩 Application
- 🟧 Default
- 🟥 MPI
- 🟪 Monitor
- 🟩 NoGroup
- 🟥 OMP_API
- 🟧 OMP_LOOP
- 🟫 OMP_PARALLEL
- 🟦 OMP_SYNC
- 🟦 USER

Initialization Phase

Computation Phase

Vampir: Visualization of the NPB-MZ-MPI / BT trace

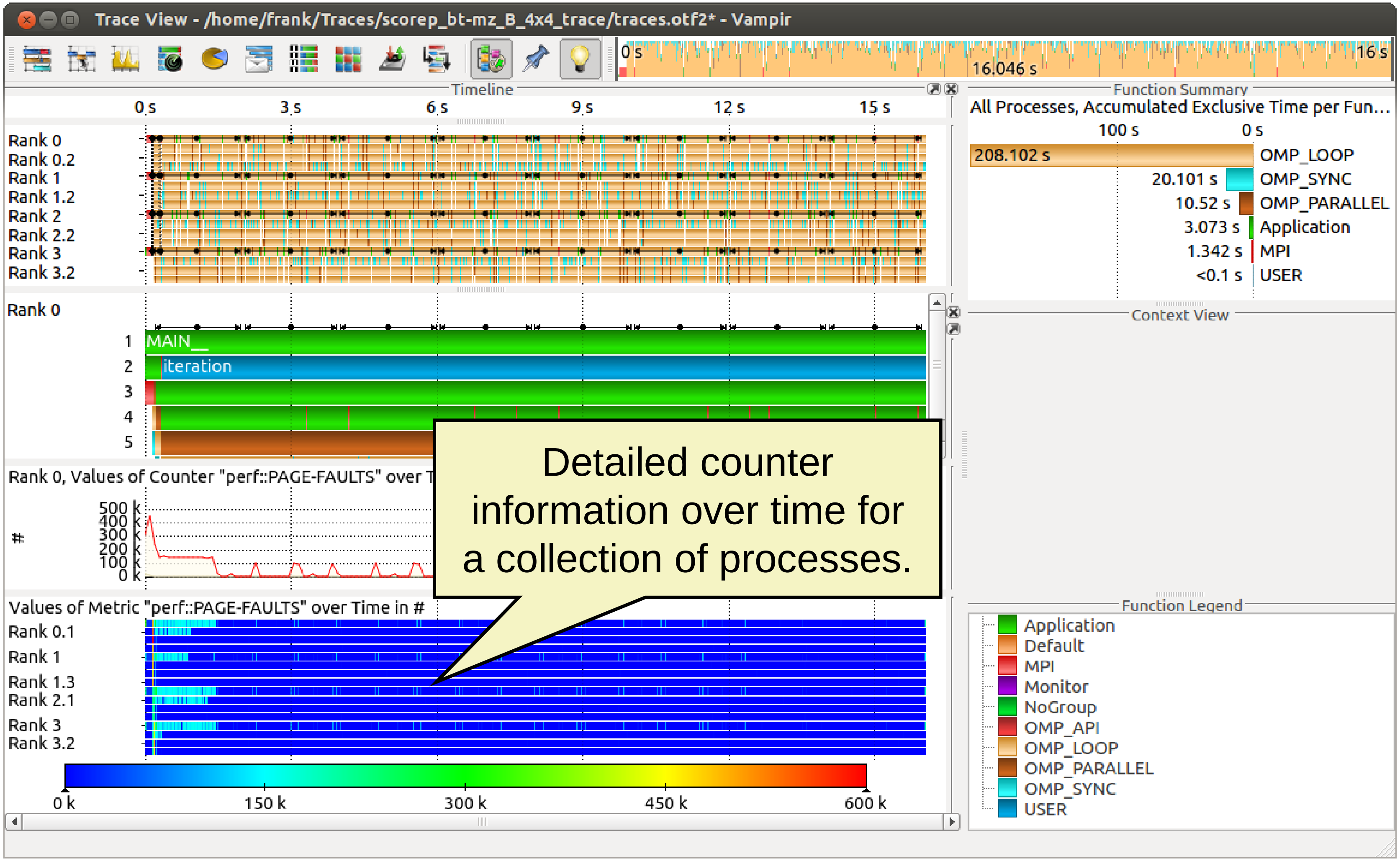
Counter Data Timeline



Detailed counter information over time for an individual process.

Vampir: Visualization of the NPB-MZ-MPI / BT trace

Performance Radar

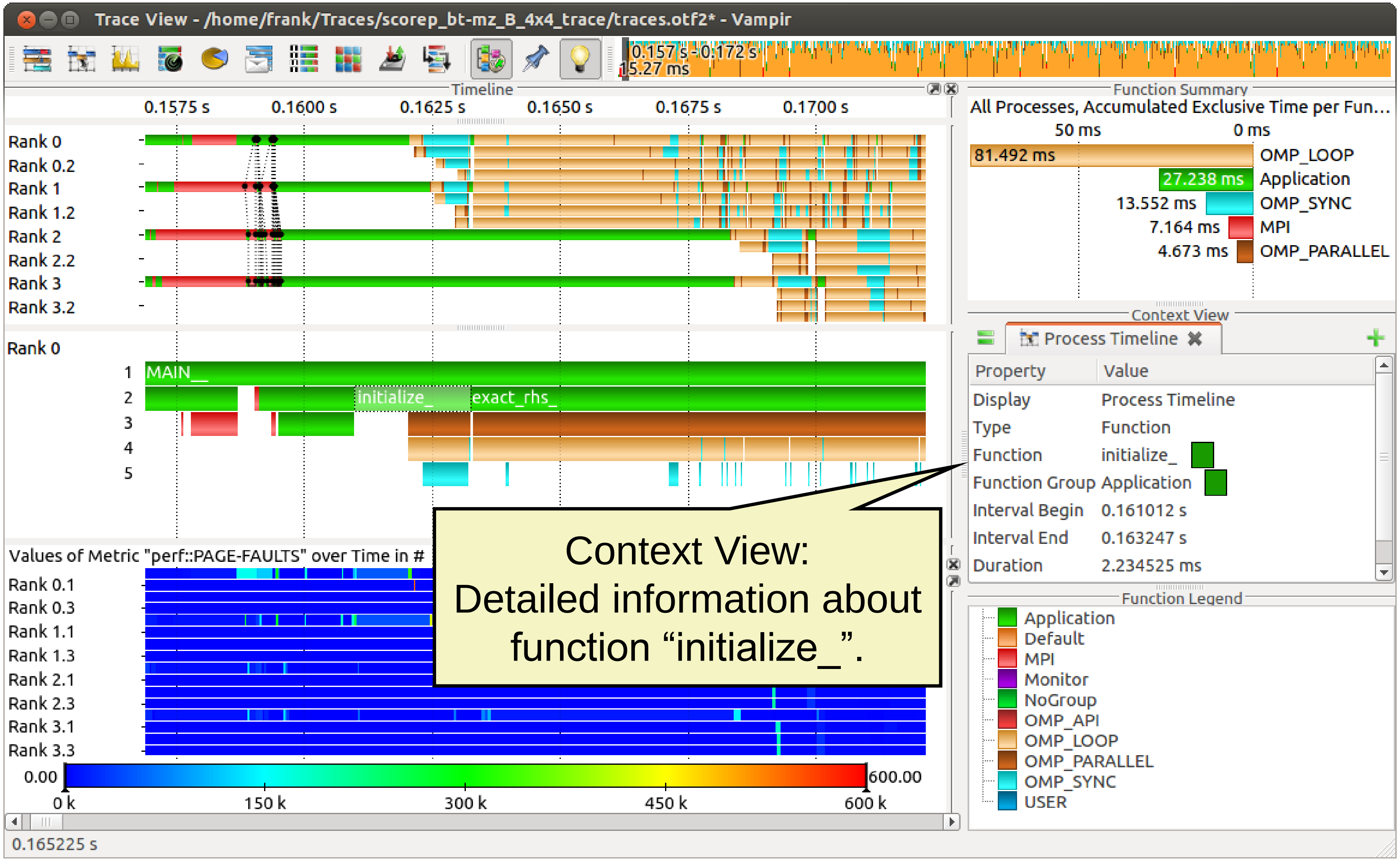


Trace View - /home/frank/Traces/scorep_bt-mz_B_4x4_trace/traces.otf2* - Vampir

Detailed counter information over time for a collection of processes.

Vampir: Visualization of the NPB-MZ-MPI / BT trace

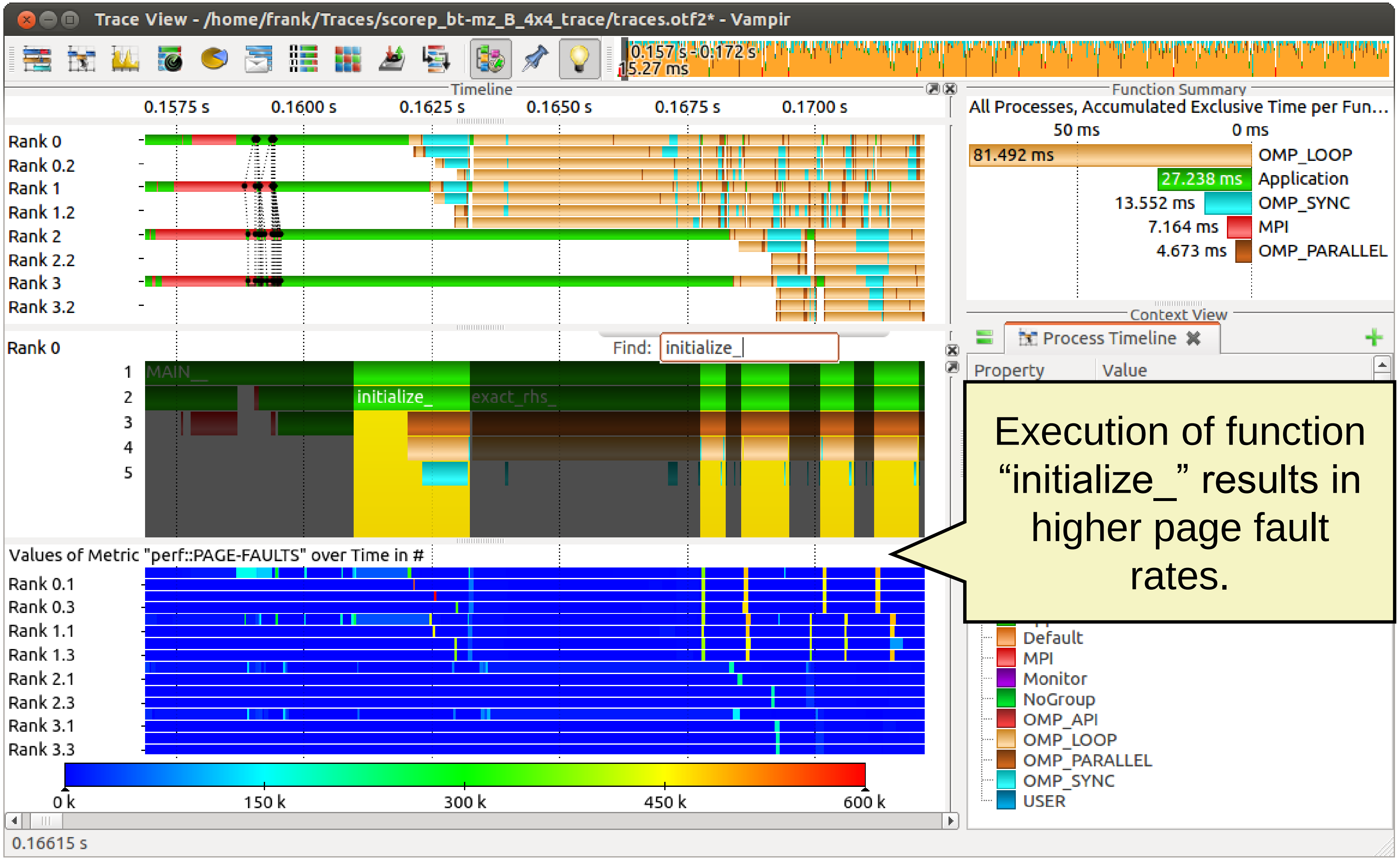
Zoom in: Initialization Phase



Property	Value
Display	Process Timeline
Type	Function
Function	initialize_ 🟩
Function Group	Application 🟩
Interval Begin	0.161012 s
Interval End	0.163247 s
Duration	2.234525 ms

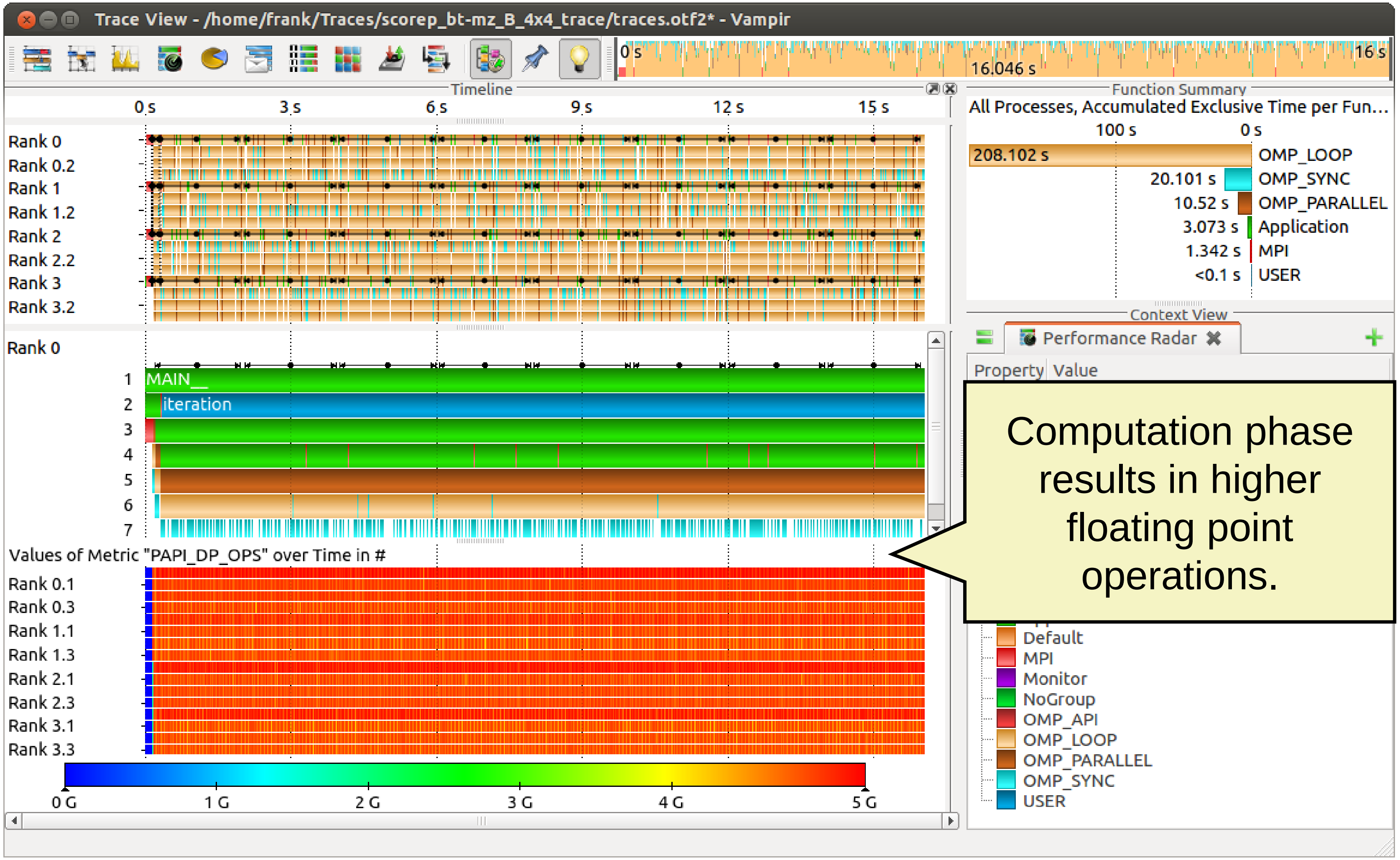
Context View:
Detailed information about function “initialize_”.

Feature: Find Function


Trace View - /home/frank/Traces/scorep_bt-mz_B_4x4_trace/traces.otf2* - Vampir

Execution of function “initialize_” results in higher page fault rates.

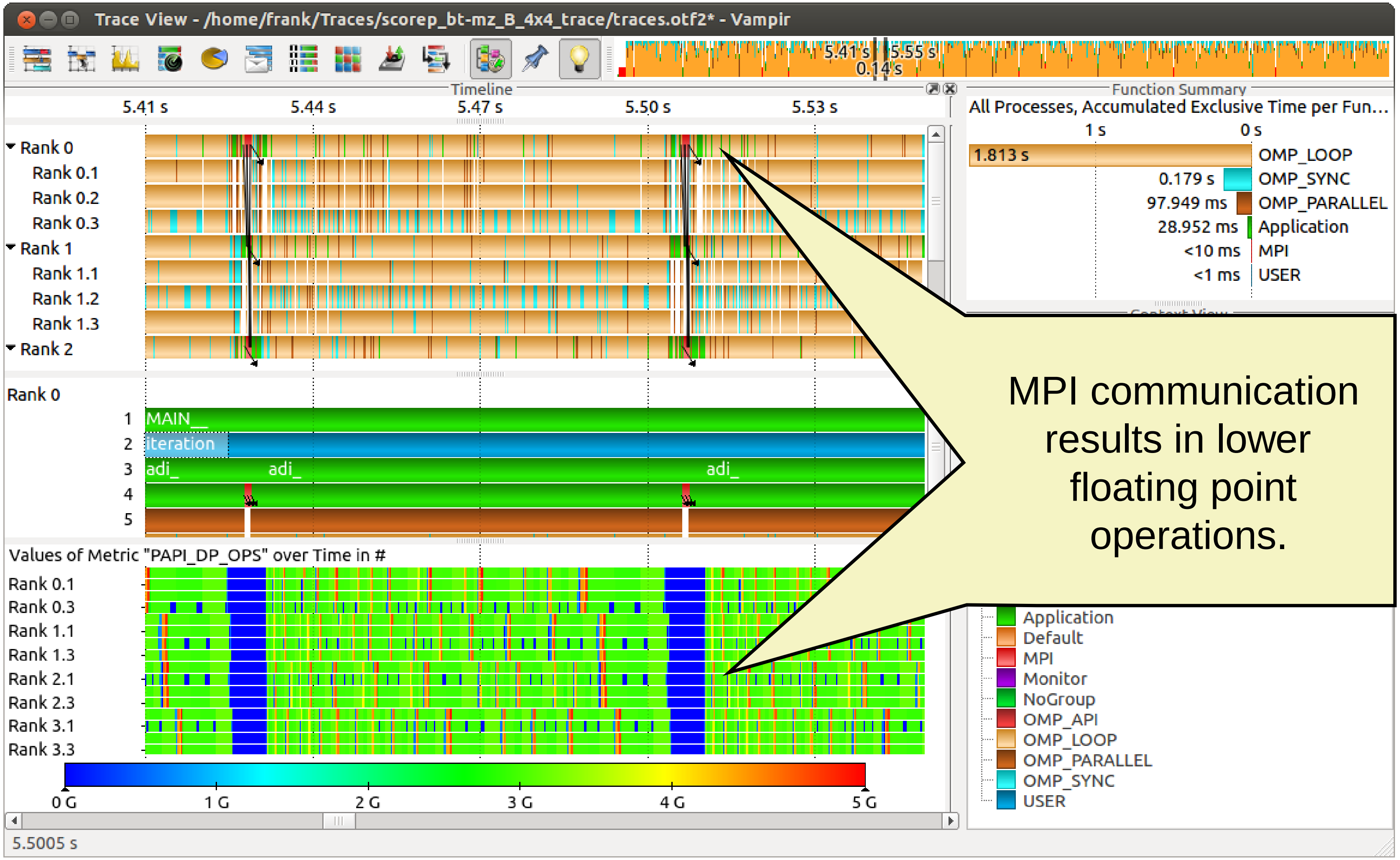
Computation Phase



Computation phase results in higher floating point operations.

Vampir: Visualization of the NPB-MZ-MPI / BT trace

Zoom in: Computation Phase



Trace View - /home/frank/Traces/scorep_bt-mz_B_4x4_trace/traces.otf2* - Vampir

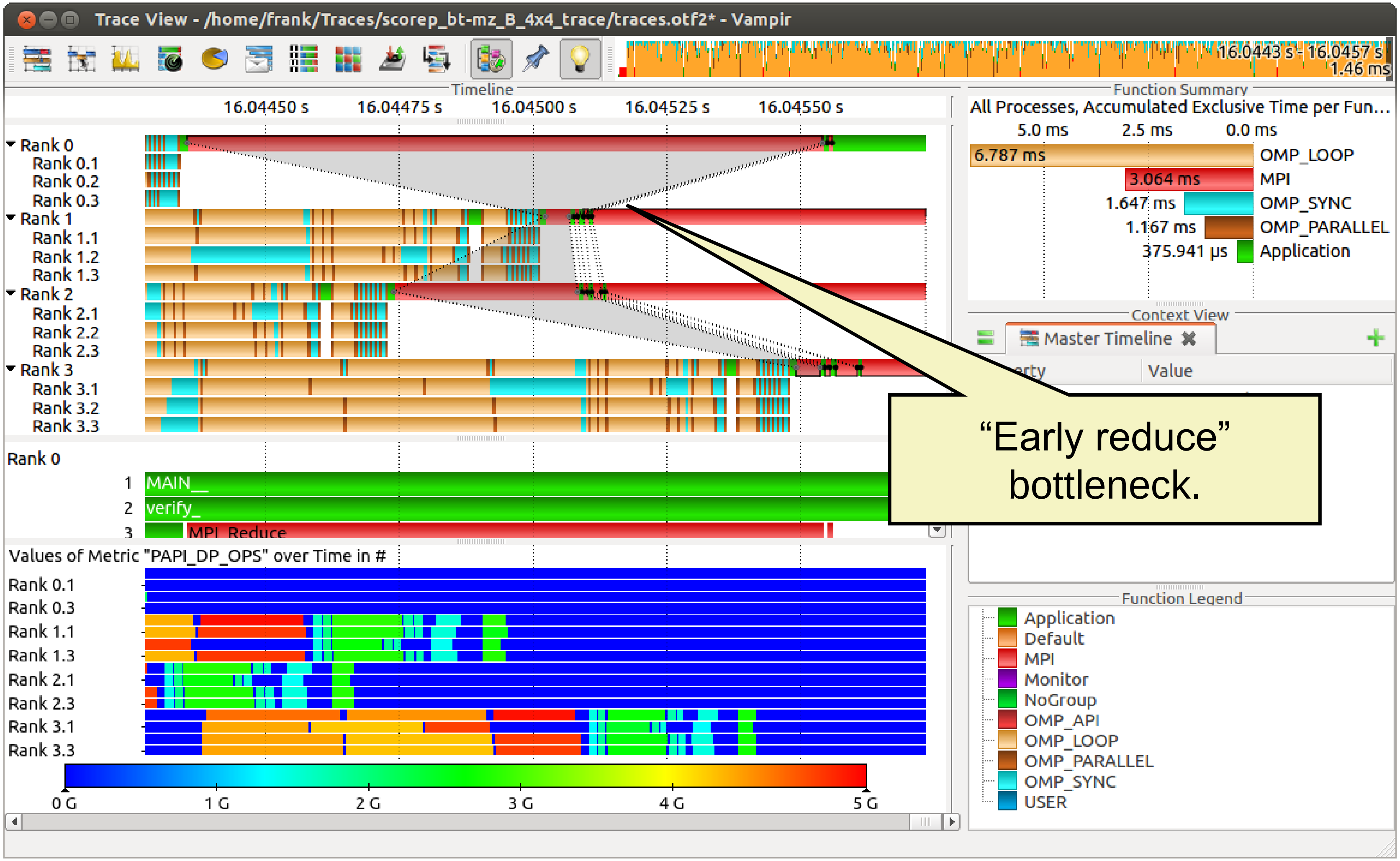
All Processes, Accumulated Exclusive Time per Fun...

Time	Function
1.813 s	OMP_LOOP
0.179 s	OMP_SYNC
97.949 ms	OMP_PARALLEL
28.952 ms	Application
<10 ms	MPI
<1 ms	USER

MPI communication results in lower floating point operations.

Vampir: Visualization of the NPB-MZ-MPI / BT trace

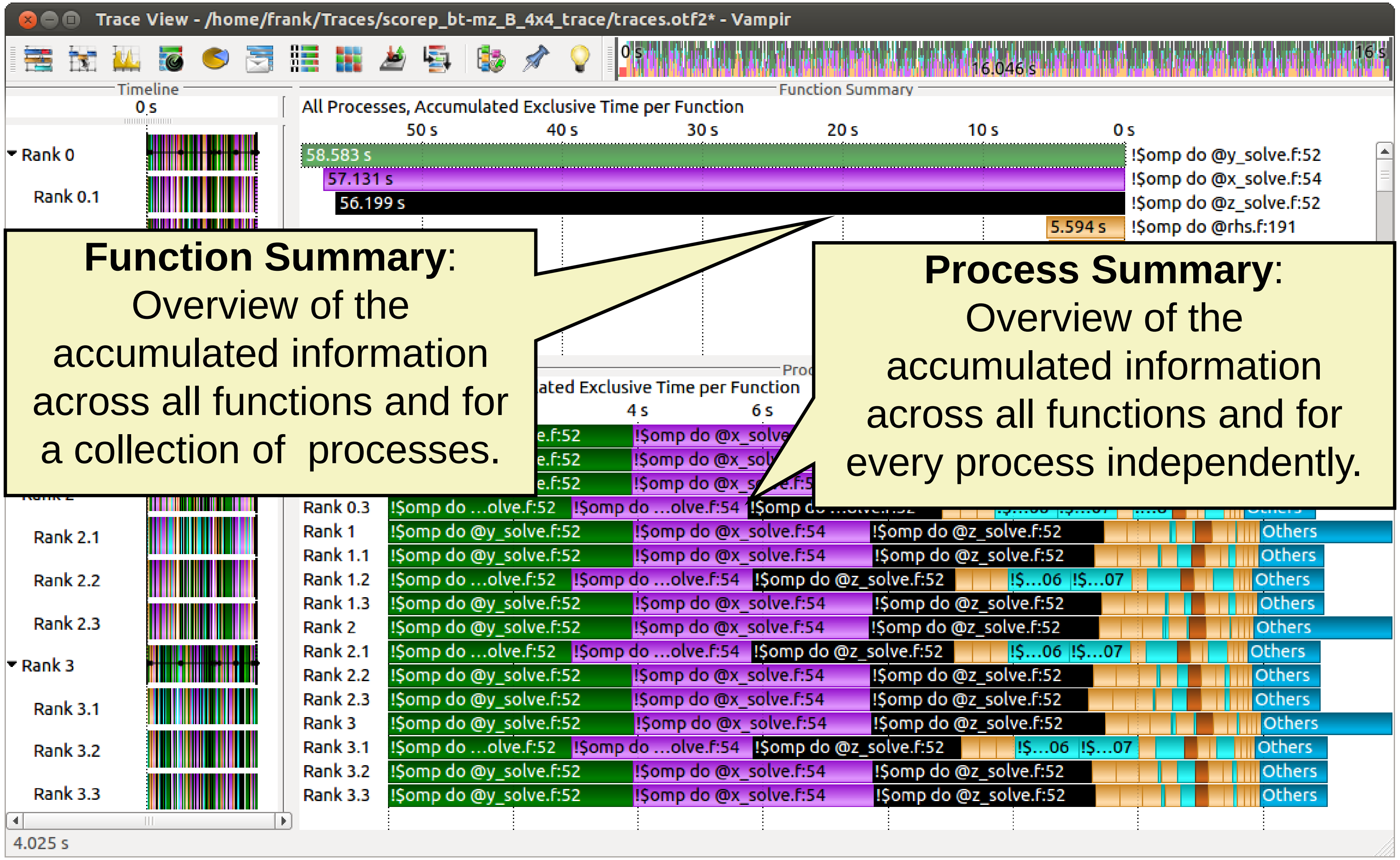
Zoom in: Finalization Phase



"Early reduce" bottleneck.

Vampir: Visualization of the NPB-MZ-MPI / BT trace

Process Summary



Trace View - /home/frank/Traces/scorep_bt-mz_B_4x4_trace/traces.otf2* - Vampir

All Processes, Accumulated Exclusive Time per Function

Time	Function
58.583 s	!$omp do @y_solve.f:52
57.131 s	!$omp do @x_solve.f:54
56.199 s	!$omp do @z_solve.f:52
5.594 s	!$omp do @rhs.f:191

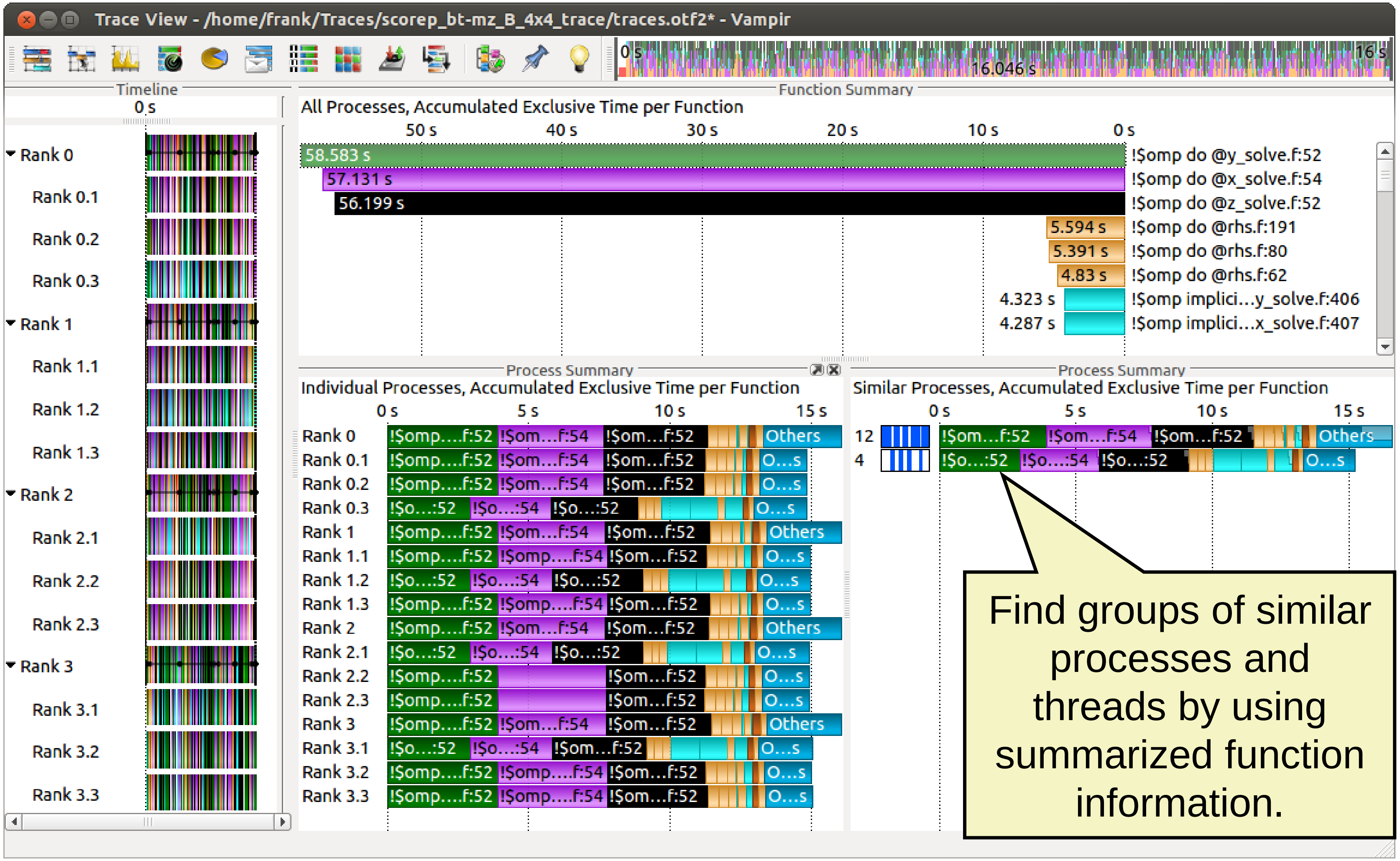
ated Exclusive Time per Function

	e.f:52	!$omp do @x_solve				
	e.f:52	!$omp do @x_sol				
	e.f:52	!$omp do @x_solve.f:5				
Rank 0.3	!$omp do ...olve.f:52	!$omp do ...olve.f:54	!$omp d			
Rank 1	!$omp do @y_solve.f:52	!$omp do @x_solve.f:54	!$omp do @z_solve.f:52			Others
Rank 1.1	!$omp do @y_solve.f:52	!$omp do @x_solve.f:54	!$omp do @z_solve.f:52			Others
Rank 1.2	!$omp do ...olve.f:52	!$omp do ...olve.f:54	!$omp do @z_solve.f:52	!$...06	!$...07	Others
Rank 1.3	!$omp do @y_solve.f:52	!$omp do @x_solve.f:54	!$omp do @z_solve.f:52			Others
Rank 2	!$omp do @y_solve.f:52	!$omp do @x_solve.f:54	!$omp do @z_solve.f:52			Others
Rank 2.1	!$omp do ...olve.f:52	!$omp do ...olve.f:54	!$omp do @z_solve.f:52	!$...06	!$...07	Others
Rank 2.2	!$omp do @y_solve.f:52	!$omp do @x_solve.f:54	!$omp do @z_solve.f:52			Others
Rank 2.3	!$omp do @y_solve.f:52	!$omp do @x_solve.f:54	!$omp do @z_solve.f:52			Others
Rank 3	!$omp do @y_solve.f:52	!$omp do @x_solve.f:54	!$omp do @z_solve.f:52			Others
Rank 3.1	!$omp do ...olve.f:52	!$omp do ...olve.f:54	!$omp do @z_solve.f:52	!$...06	!$...07	Others
Rank 3.2	!$omp do @y_solve.f:52	!$omp do @x_solve.f:54	!$omp do @z_solve.f:52			Others
Rank 3.3	!$omp do @y_solve.f:52	!$omp do @x_solve.f:54	!$omp do @z_solve.f:52			Others

Function Summary: Overview of the accumulated information across all functions and for a collection of processes.

Process Summary: Overview of the accumulated information across all functions and for every process independently.

Process Summary



Trace View - /home/frank/Traces/scorep_bt-mz_B_4x4_trace/traces.otf2* - Vampir

All Processes, Accumulated Exclusive Time per Function

Time	Function
58.583 s	!$omp do @y_solve.f:52
57.131 s	!$omp do @x_solve.f:54
56.199 s	!$omp do @z_solve.f:52
5.594 s	!$omp do @rhs.f:191
5.391 s	!$omp do @rhs.f:80
4.83 s	!$omp do @rhs.f:62
4.323 s	!$omp implici...y_solve.f:406
4.287 s	!$omp implici...x_solve.f:407

Individual Processes, Accumulated Exclusive Time per Function

```
Rank 0    !$omp....f:52  !$om...f:54  !$om...f:52  Others
Rank 0.1  !$omp....f:52  !$om...f:54  !$om...f:52  O...s
Rank 0.2  !$omp....f:52  !$om...f:54  !$om...f:52  O...s
Rank 0.3  !$o...:52  !$o...:54  !$o...:52  O...s
Rank 1    !$omp....f:52  !$om...f:54  !$om...f:52  Others
Rank 1.1  !$omp....f:52  !$omp....f:54  !$om...f:52  O...s
Rank 1.2  !$o...:52  !$o...:54  !$o...:52  O...s
Rank 1.3  !$omp....f:52  !$omp....f:54  !$om...f:52  O...s
Rank 2    !$omp....f:52  !$om...f:54  !$om...f:52  Others
Rank 2.1  !$o...:52  !$o...:54  !$o...:52  O...s
Rank 2.2  !$omp....f:52  !$om...f:52  O...s
Rank 2.3  !$omp....f:52  !$om...f:52  O...s
Rank 3    !$omp....f:52  !$om...f:54  !$om...f:52  Others
Rank 3.1  !$o...:52  !$o...:54  !$om...f:52  O...s
Rank 3.2  !$omp....f:52  !$omp....f:54  !$om...f:52  O...s
Rank 3.3  !$omp....f:52  !$omp....f:54  !$om...f:52  O...s
```

Similar Processes, Accumulated Exclusive Time per Function

```
12  !$om...f:52  !$om...f:54  !$om...f:52  Others
4   !$o...:52  !$o...:54  !$o...:52  O...s
```

Find groups of similar processes and threads by using summarized function information.

Vampir is available at http://www.vampir.eu,

Get support via vampirsupport@zih.tu-dresden.de

Hardware performance/soft counter measurements hands-on

VI-HPS Team

Advanced Measurement Configuration: Metrics

- If Score-P has been built with performance metric support it is capable of recording performance counter information
- Requested counters will be recorded with every enter/exit event
- Supported metric sources
  - PAPI
  - Resource usage statistics
  - Custom written metric plug-ins

Note: Additional memory is needed to store metric values.
Therefore, you may have to adjust SCOREP_TOTAL_MEMORY,
for example as reported using “scorep-score -c”

Advanced Measurement Configuration: Metrics

- Recording hardware counters via PAPI

```
% export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_FP_INS
% OMP_NUM_THREADS=4 mpiexec -n 4 ./bt-mz_W.4

 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark

 [... More application output ...]
```

- Also possible to record them only per rank

```
% export SCOREP_METRIC_PAPI_PER_PROCESS=PAPI_L3_DCM
% OMP_NUM_THREADS=4 mpiexec -n 4 ./bt-mz_W.4

 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark

 [... More application output ...]
```

Advanced Measurement Configuration: Metrics

- Available PAPI metrics
  - Preset events: common set of events deemed relevant and useful for application performance tuning
    - Abstraction from specific hardware performance counters, mapping onto available events done by PAPI internally

```
% papi_avail
```

  - Native events: set of all events that are available on the CPU (**platform dependent**)

```
% papi_native_avail
```

Note:
Due to hardware restrictions
- number of concurrently measured events is limited
- there may be unsupported combinations of concurrent events
- Use `papi_event_chooser` tool to test event combinations

Advanced Measurement Configuration: Metrics

- Recording operating system resource usage

```
% export SCOREP_METRIC_RUSAGE=ru_stime
% OMP_NUM_THREADS=4 mpiexec -n 4 ./bt-mz_W.4

 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark

 [... More application output ...]
```

- Also possible to record them only per rank

```
% export SCOREP_METRIC_RUSAGE_PER_PROCESS=ru_maxrss
% OMP_NUM_THREADS=4 mpiexec -n 4 ./bt-mz_W.4

 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark

 [... More application output ...]
```

Advanced Measurement Configuration: Metrics

- Available resource usage metrics

```
% man getrusage
 [... Output ...]

struct rusage {
    struct timeval ru_utime; /* user CPU time used */
    struct timeval ru_stime; /* system CPU time used */
    long   ru_maxrss;        /* maximum resident set size */
    long   ru_ixrss;         /* integral shared memory size */
    long   ru_idrss;         /* integral unshared data size */
    long   ru_isrss;         /* integral unshared stack size */
    long   ru_minflt;        /* page reclaims (soft page faults) */
    long   ru_majflt;        /* page faults (hard page faults) */
    long   ru_nswap;         /* swaps */
    long   ru_inblock;       /* block input operations */
    long   ru_oublock;       /* block output operations */
    long   ru_msgsnd;        /* IPC messages sent */
    long   ru_msgrcv;        /* IPC messages received */
    long   ru_nsignals;      /* signals received */
    long   ru_nvcsw;         /* voluntary context switches */
    long   ru_nivcsw;        /* involuntary context switches */
};

 [... More output ...]
```

Note:

(1) Not all fields are maintained on each platform.

(2) Check scope of metrics (per process vs. per thread)

Score-P Hands-On

CUDA: Jacobi example

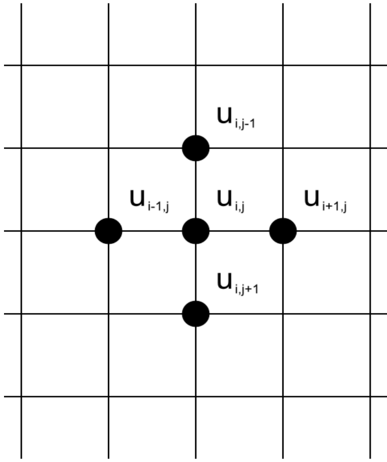
Jacobi Solver

- Jacobi Example
  - Iterative solver for system of equations

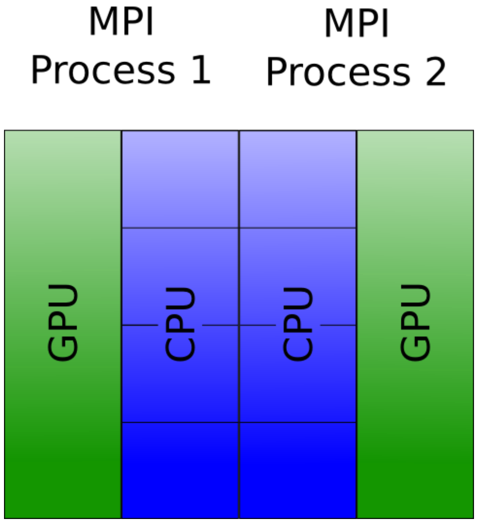
$$U_{old} = U$$

$$u_{i,j} = bu_{old,i,j} + a_x(u_{old,i-1,j} + u_{old,i+1,j}) + a_y(u_{old,i,j-1} + u_{old,i,j+1}) - rHs/b$$

  - Code uses OpenMP, CUDA and MPI for parallelization



- Domain decomposition
  - Halo exchange at boundaries:
    - Via MPI between processes
    - Via CUDA between hosts and accelerators



Jacobi Without Instrumentation

```
# Compile host code
%        mpicc -O3 -fopenmp -DUSE_MPI –I<path_to_cuda_header>
          -c jacobi_cuda.c -o jacobi_mpi+cuda.o

# Compile CUDA kernel
%        nvcc -O3 -c jacobi_cuda_kernel.cu
          -o jacobi_cuda_kernel.o

# Link executable
%        mpicc -fopenmp -lm –L<path_tocuda_libs> -lcudart
          jacobi_mpi+cuda.o jacobi_cuda_kernel.o -o ./jacobi_mpi+cuda
```

Instrumentation with Score-P

```
# Compile host code
% scorep mpicc -O3 -fopenmp -DUSE_MPI –I<path_to_cuda_header>
         -c jacobi_cuda.c -o jacobi_mpi+cuda.o

# Compile CUDA kernel
% scorep nvcc -O3 -c jacobi_cuda_kernel.cu
         -o jacobi_cuda_kernel.o

# Link executable
% scorep mpicc -fopenmp -lm –L<path_tocuda_libs> -lcudart
         jacobi_mpi+cuda.o jacobi_cuda_kernel.o -o ./jacobi_mpi+cuda
```

CUDA Advanced Measurement Configuration

- Enable recording of CUDA events with the CUPTI interface via environment variable **`SCOREP_CUDA_ENABLE`**

- Provide a list of recording types, e.g.

```
% export SCOREP_CUDA_ENABLE=runtime,driver,gpu,kernel,idle
```

- Start with using the default configuration

```
% export SCOREP_CUDA_ENABLE=yes
```

- Adjust CUPTI buffer size (in bytes) as needed

```
% export SCOREP_CUDA_BUFFER=100000
```

SCOREP_CUDA_ENABLE: Recording Types

Recording type	Remark
yes/DEFAULT/1	"runtime, kernel, memcpy"
no	Disable CUDA measurement (same as unset SCOREP_CUDA_ENABLE)
runtime	CUDA runtime API
driver	CUDA driver API
kernel	CUDA kernels
kernel_counter	Fixed CUDA kernel metrics
idle	GPU compute idle time
pure_idle	GPU idle time (memory copies are not idle)
memcpy	CUDA memory copies
sync	Record implicit and explicit CUDA synchronization
gpumemusage	Record CUDA memory (de)allocations as a counter

Measurement (Profiling)

```
% export OMP_NUM_THREADS=6
% export SCOREP_CUDA_ENABLE=yes
% export SCOREP_CUDA_BUFFER=500000
% export SCOREP_EXPERIMENT_DIRECTORY=jacobi_cuda_profile

% mpirun -n 2 ./jacobi_mpi+cuda 4096 4096 0.15

Jacobi relaxation Calculation: 4096 x 4096 mesh with
 2 processes and 6 threads + one Tesla T10 Processor for each process.
 307 of 2049 local rows are calculated on the CPU to balance the load
 between the CPU and the GPU.
    0, 0.113429
  … … … … … …
   900, 0.000101
 total: 12.83581
```

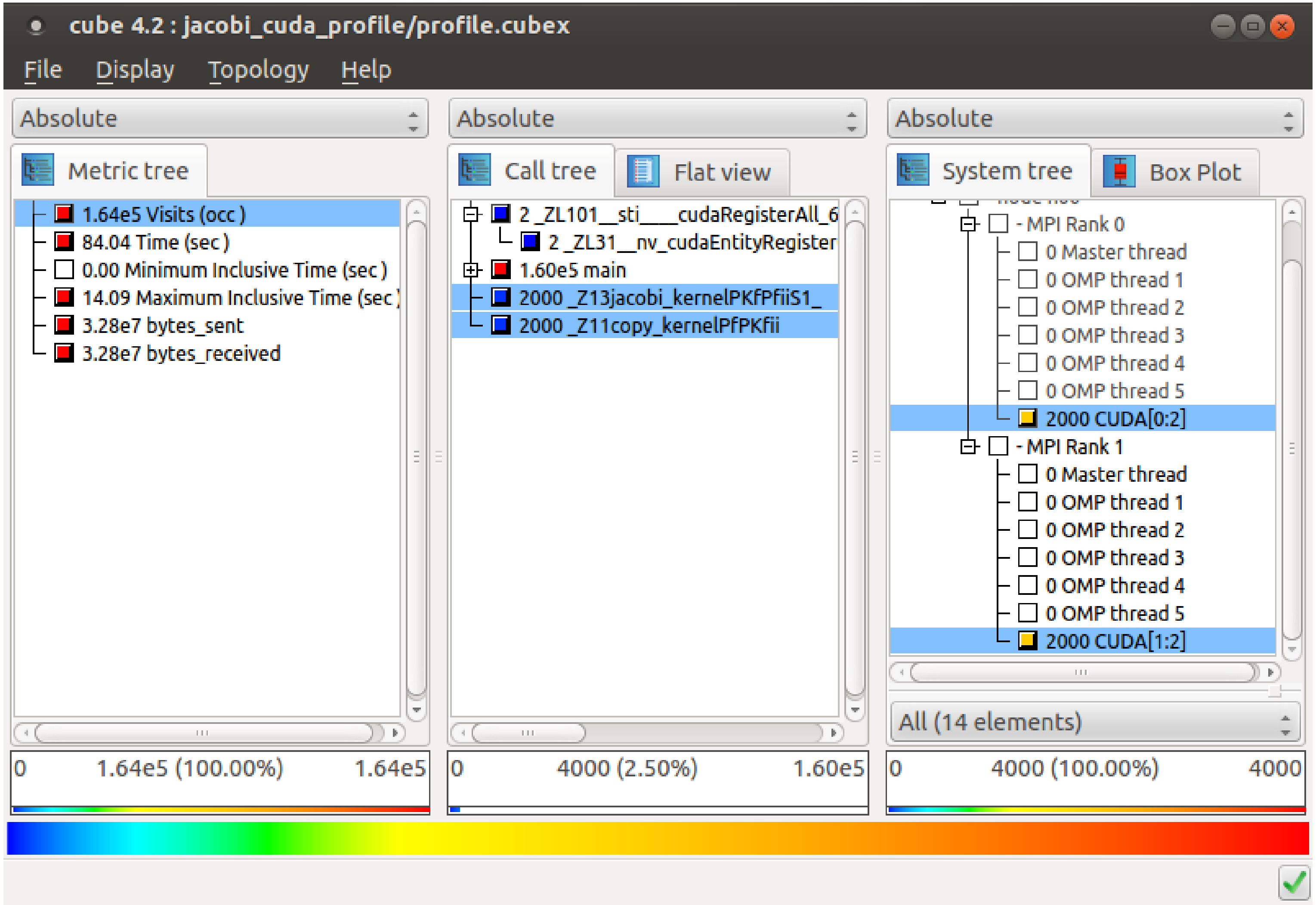
Problem size (x dimension)

Problem size (y dimension)

Load balancing factor (in this example 15% of the computations are calculated on the CPU)

CUBE4 Analysis

```
% cube jacobi_cuda_profile/profile.cubex
```



```
├ ■ 1.64e5 Visits (occ)
├ ■ 84.04 Time (sec)
├ □ 0.00 Minimum Inclusive Time (sec)
├ ■ 14.09 Maximum Inclusive Time (sec)
├ ■ 3.28e7 bytes_sent
└ ■ 3.28e7 bytes_received
```

```
⊟ ■ 2 _ZL101__sti____cudaRegisterAll_6
  └ ■ 2 _ZL31__nv_cudaEntityRegister
⊞ ■ 1.60e5 main
├ ■ 2000 _Z13jacobi_kernelPKfPfiiS1_
└ ■ 2000 _Z11copy_kernelPfPKfii
```

```
⊟ □ - MPI Rank 0
    ├ □ 0 Master thread
    ├ □ 0 OMP thread 1
    ├ □ 0 OMP thread 2
    ├ □ 0 OMP thread 3
    ├ □ 0 OMP thread 4
    ├ □ 0 OMP thread 5
    └ ■ 2000 CUDA[0:2]
⊟ □ - MPI Rank 1
    ├ □ 0 Master thread
    ├ □ 0 OMP thread 1
    ├ □ 0 OMP thread 2
    ├ □ 0 OMP thread 3
    ├ □ 0 OMP thread 4
    ├ □ 0 OMP thread 5
    └ ■ 2000 CUDA[1:2]
```

All (14 elements)

```
0    1.64e5 (100.00%)    1.64e5
0    4000 (2.50%)        1.60e5
0    4000 (100.00%)      4000
```

Scoring

- Do we need to filter? (Overhead and memory footprint)

```
% scorep-score jacobi_cuda_profile/profile.cubex
Estimated aggregate size of event trace (total_tbc):       3.875.472 bytes
Estimated requirements for largest trace buffer (max_tbc): 1.937.936 bytes
(hint: When tracing set SCOREP_TOTAL_MEMORY > max_tbc to avoid
       intermediate flushes or reduce requirements using file listing
       names of USR regions to be filtered.)

flt type        max_tbc          time      % region
     ALL        1937936         24.97  100.0 ALL
     OMP        1154110         18.78   75.2 OMP
     USR         667480          5.95   23.8 USR
     MPI         116192          0.14    0.5 MPI
     COM            154          0.10    0.4 COM
```

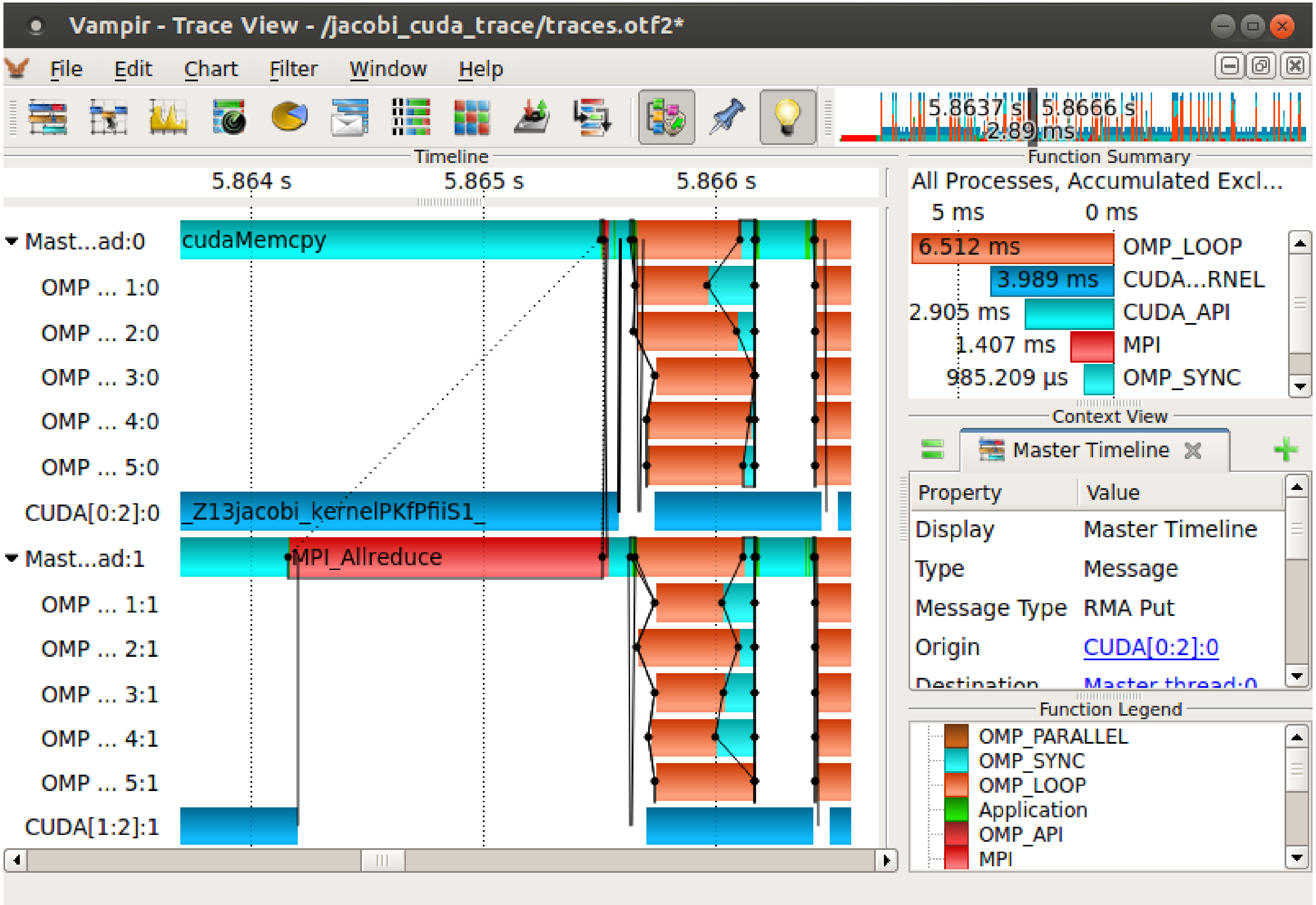
☞ Very small example => no filtering

Measurement (Tracing)

```
% export OMP_NUM_THREADS=6
% export SCOREP_CUDA_ENABLE=yes
% export SCOREP_CUDA_BUFFER=500000
% export SCOREP_EXPERIMENT_DIRECTORY=jacobi_cuda_trace
% export SCOREP_ENABLE_PROFILING=false
% export SCOREP_ENABLE_TRACING=true

% mpirun -n 2 ./jacobi_mpi+cuda 4096 4096 0.15

Jacobi relaxation Calculation: 4096 x 4096 mesh with
 2 processes and 6 threads + one Tesla T10 Processor for each process.
 307 of 2049 local rows are calculated on the CPU to balance the load
 between the CPU and the GPU.
    0, 0.113429
  … … … … … …
  900, 0.000101
 total: 12.875220 s
```

Vampir Analysis

```
% vampir jacobi_cuda_trace/traces.otf2
```



All Processes, Accumulated Excl...

5 ms	0 ms
6.512 ms	OMP_LOOP
3.989 ms	CUDA...RNEL
2.905 ms	CUDA_API
1.407 ms	MPI
985.209 µs	OMP_SYNC

Property	Value
Display	Master Timeline
Type	Message
Message Type	RMA Put
Origin	CUDA[0:2]:0
Destination	Master thread:0

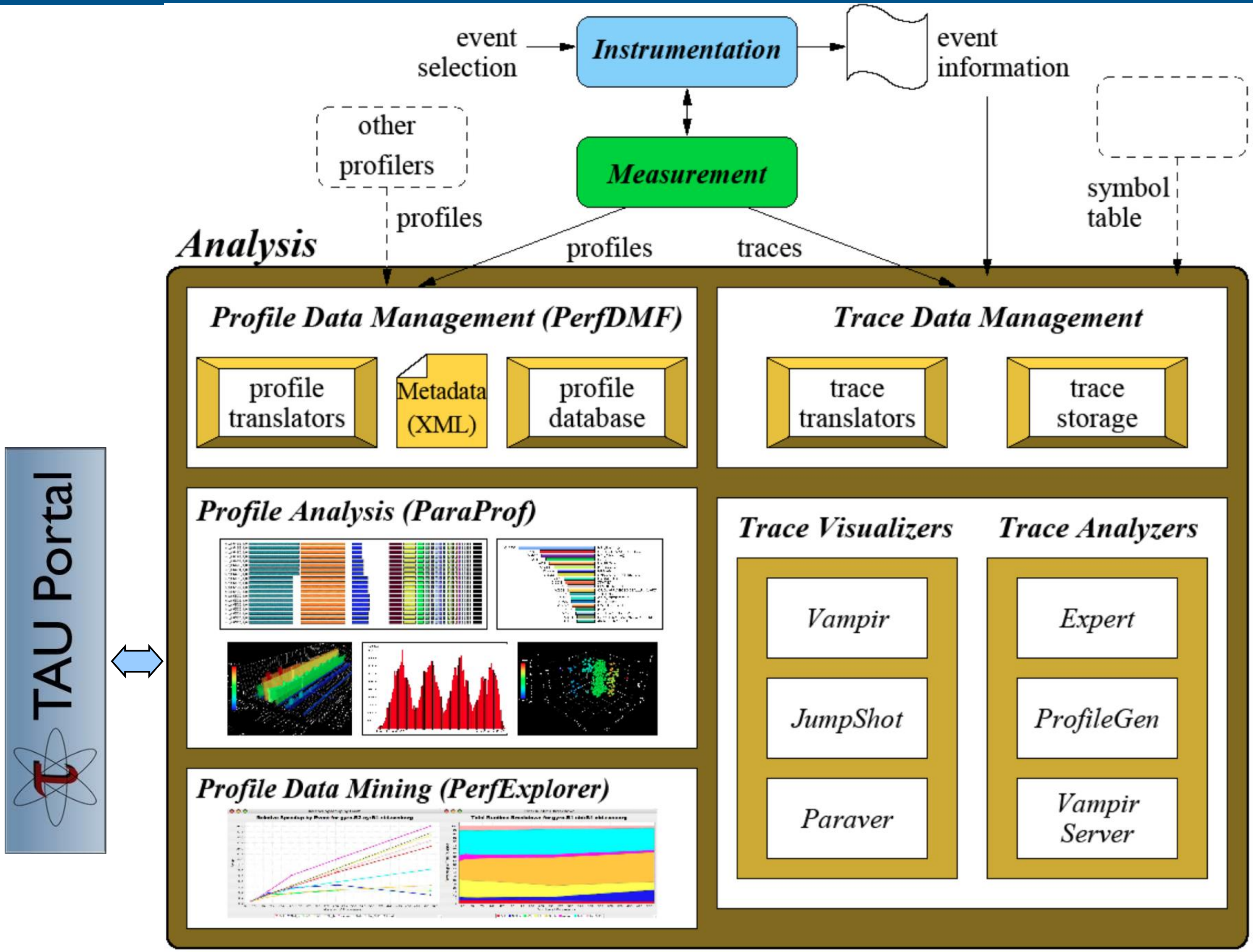
- 🟫 OMP_PARALLEL
- 🟦 OMP_SYNC
- 🟧 OMP_LOOP
- 🟩 Application
- 🟥 OMP_API
- 🟥 MPI

Performance Data Management with TAU PerfExplorer

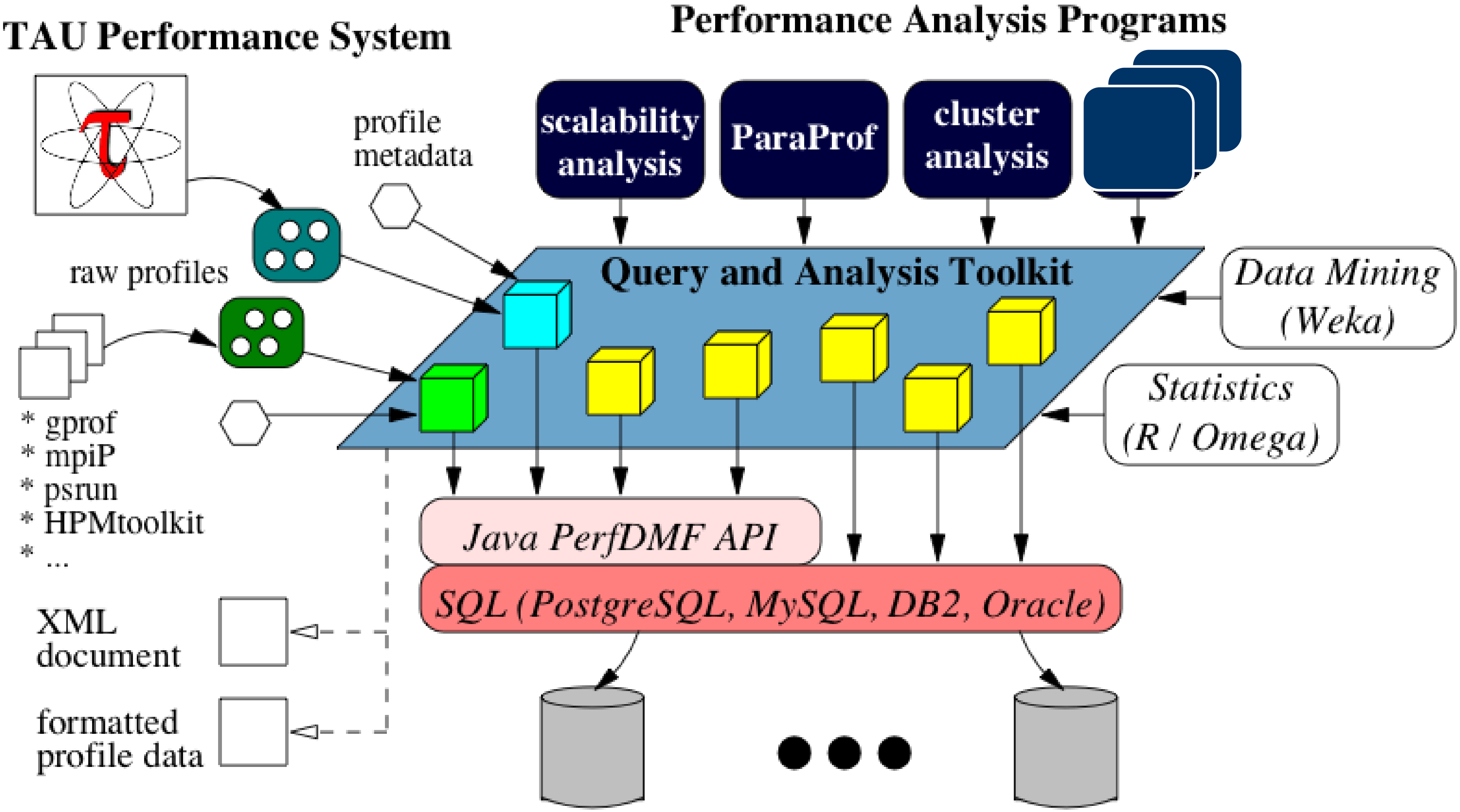
Sameer Shende
Performance Research Lab, University of Oregon

http://TAU.uoregon.edu

TAU Analysis



TAUdb: Performance Data Management Framework



Using TAUdb

- Configure TAUdb (Done by each user)

  ```
  % taudb_configure --create-default
  ```

  - Choose derby, PostgreSQL, MySQL, Oracle or DB2
  - Hostname
  - Username
  - Password
  - Say yes to downloading required drivers (we are not allowed to distribute these)
  - Stores parameters in your ~/.ParaProf/taudb.cfg file
- Configure PerfExplorer (Done by each user)

  ```
  % perfexplorer_configure
  ```
- Execute PerfExplorer

  ```
  % perfexplorer
  ```

Using PerfExplorer

```
% wget http://tau.uoregon.edu/data.tgz  (Contains CUBE profiles from Score-P)
% taudb_configure --create-default
(Chooses derby, blank user/passwd, yes to save passwd, defaults)
% perfexplorer_configure
(Yes to load schema, defaults)
% paraprof
(load each trial: DB -> Add Trial -> Type (Paraprof Packed Profile) -> OK) OR use
    taudb_loadtrial –a “app” –x “experiment” –n “name” file.ppk
Then,
% tar zxf $TAU/data.tgz; cd data/tau;
% taudb_loadtrial –a BT_MZ –x “Class_B” bt-mz_B.*.ppk
% perfexplorer
(Select experiment, Menu: Charts -> Speedup)
```

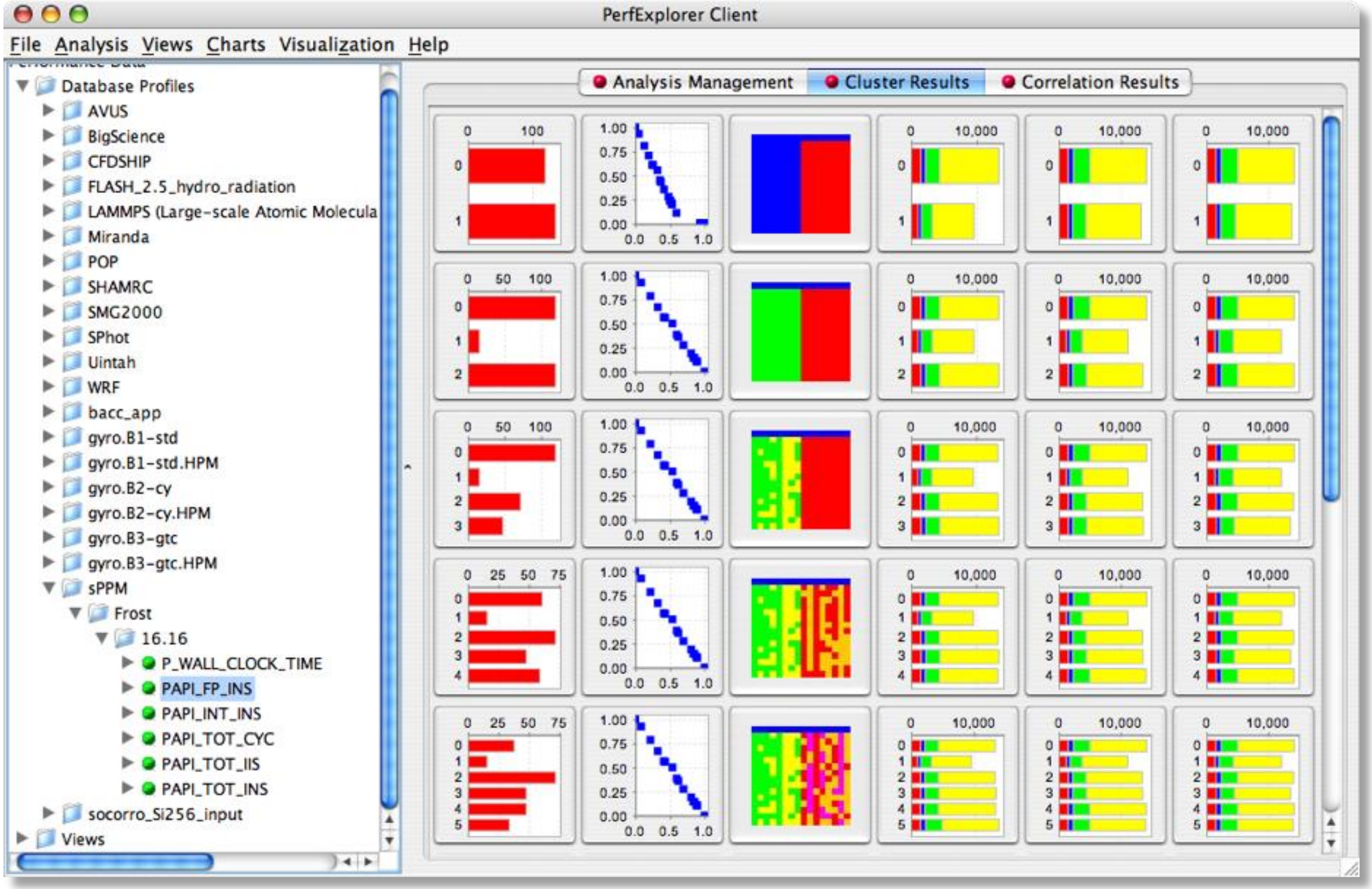
Performance Data Mining (PerfExplorer)

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

PerfExplorer: Using Cluster Analysis

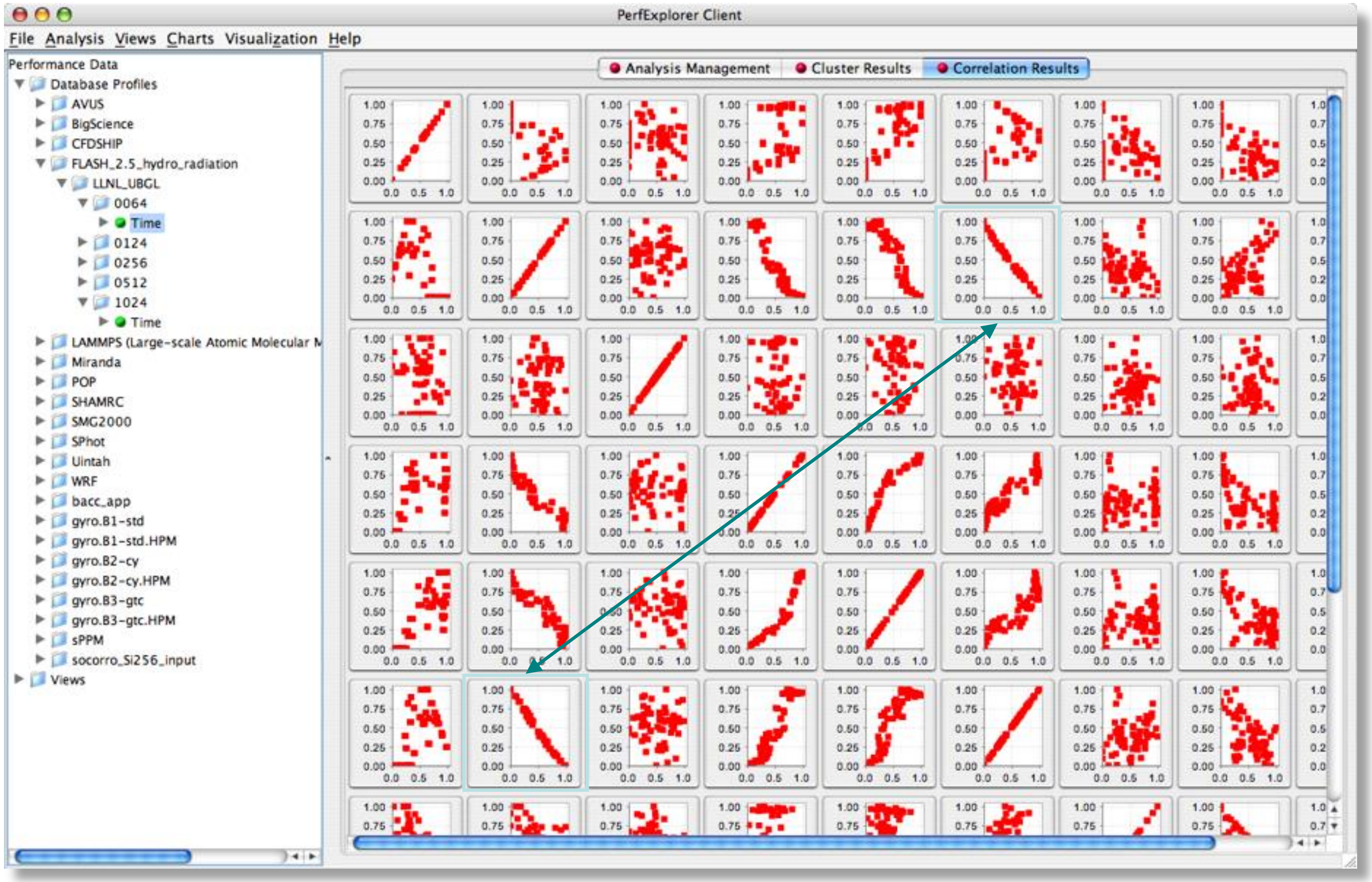
- Performance data represented as vectors - each dimension is the cumulative time for an event
- *k*-means: *k* random centers are selected and instances are grouped with the "closest" (Euclidean) center
- New centers are calculated and the process repeated until stabilization or max iterations
- Dimension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

PerfExplorer - Cluster Analysis (sPPM)



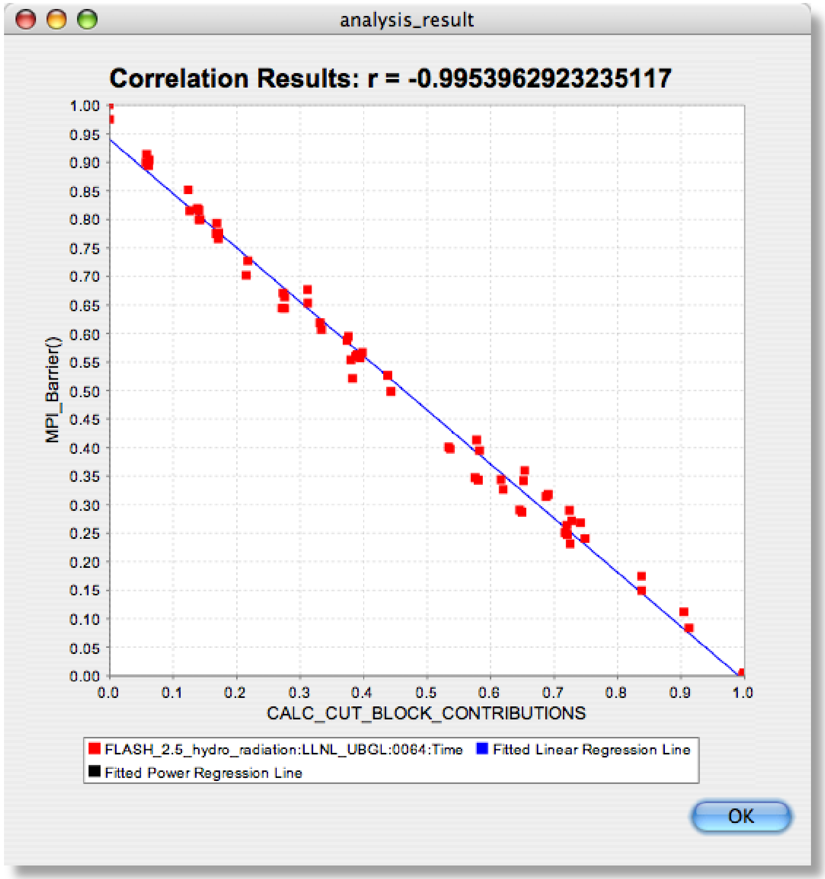
PerfExplorer - Correlation Analysis (Flash)

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



PerfExplorer - Correlation Analysis (Flash)

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



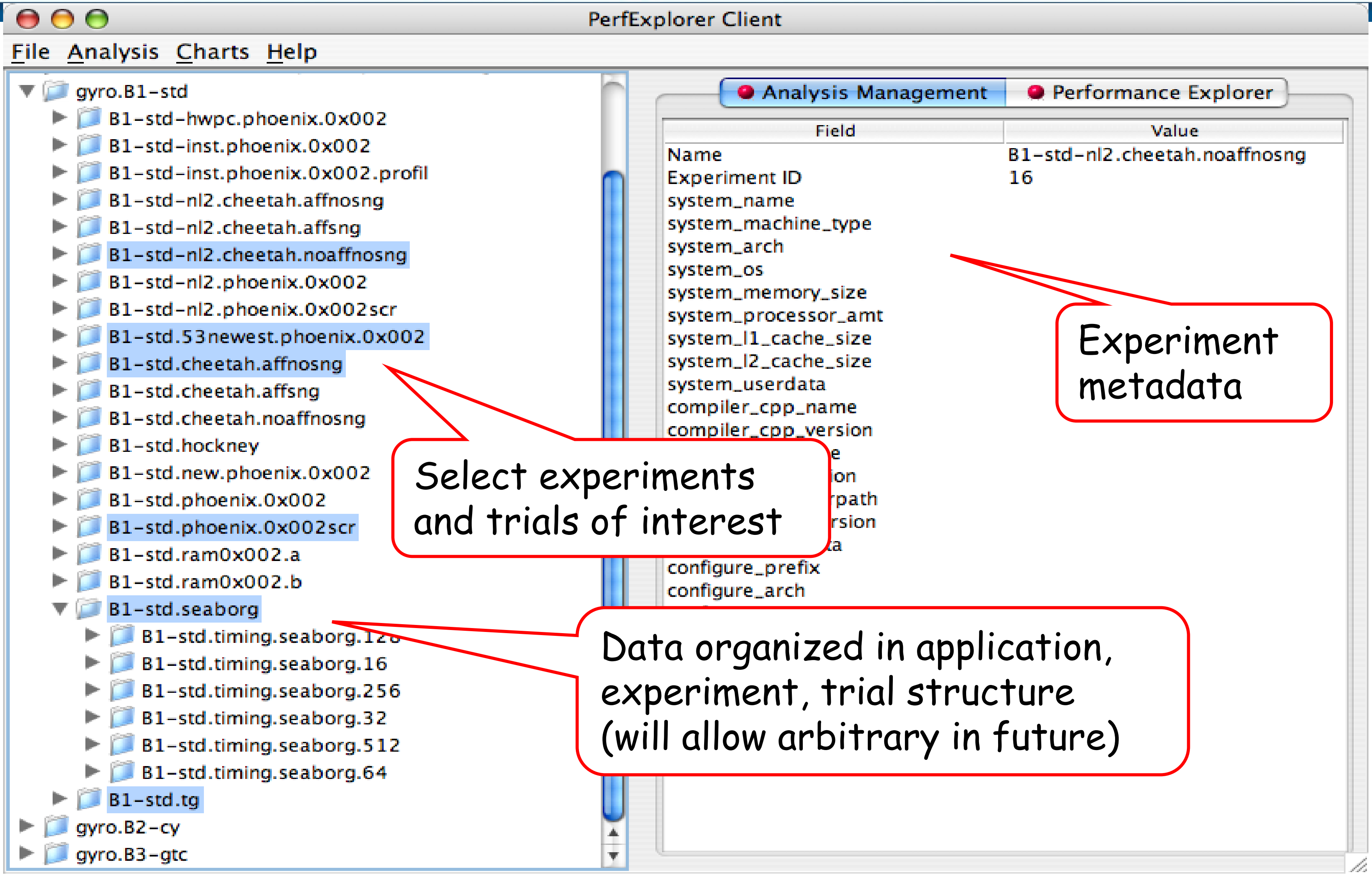
analysis_result

Correlation Results: r = -0.9953962923235117

PerfExplorer - Comparative Analysis

- Relative speedup, efficiency
  - total runtime, by event, one event, by phase
- Breakdown of total runtime
- Group fraction of total runtime
- Correlating events to total runtime
- Timesteps per second

PerfExplorer - Interface



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

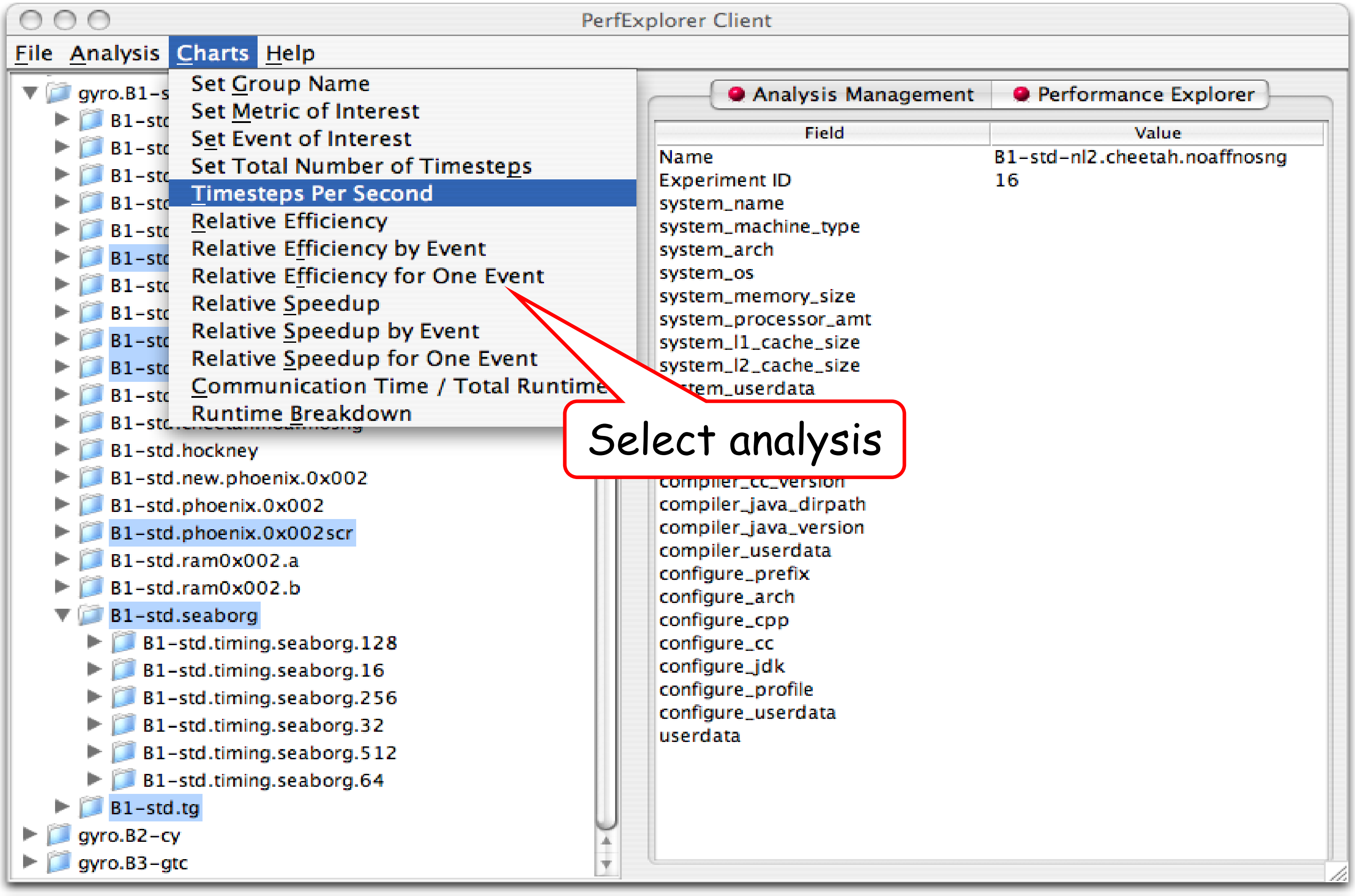
Field	Value
Name	B1-std-nl2.cheetah.noaffnosng
Experiment ID	16
system_name	
system_machine_type	
system_arch	
system_os	
system_memory_size	
system_processor_amt	
system_l1_cache_size	
system_l2_cache_size	
system_userdata	
compiler_cpp_name	
compiler_cpp_version	
e	
ion	
rpath	
rsion	
a	
configure_prefix	
configure_arch	

Experiment metadata

Select experiments and trials of interest

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

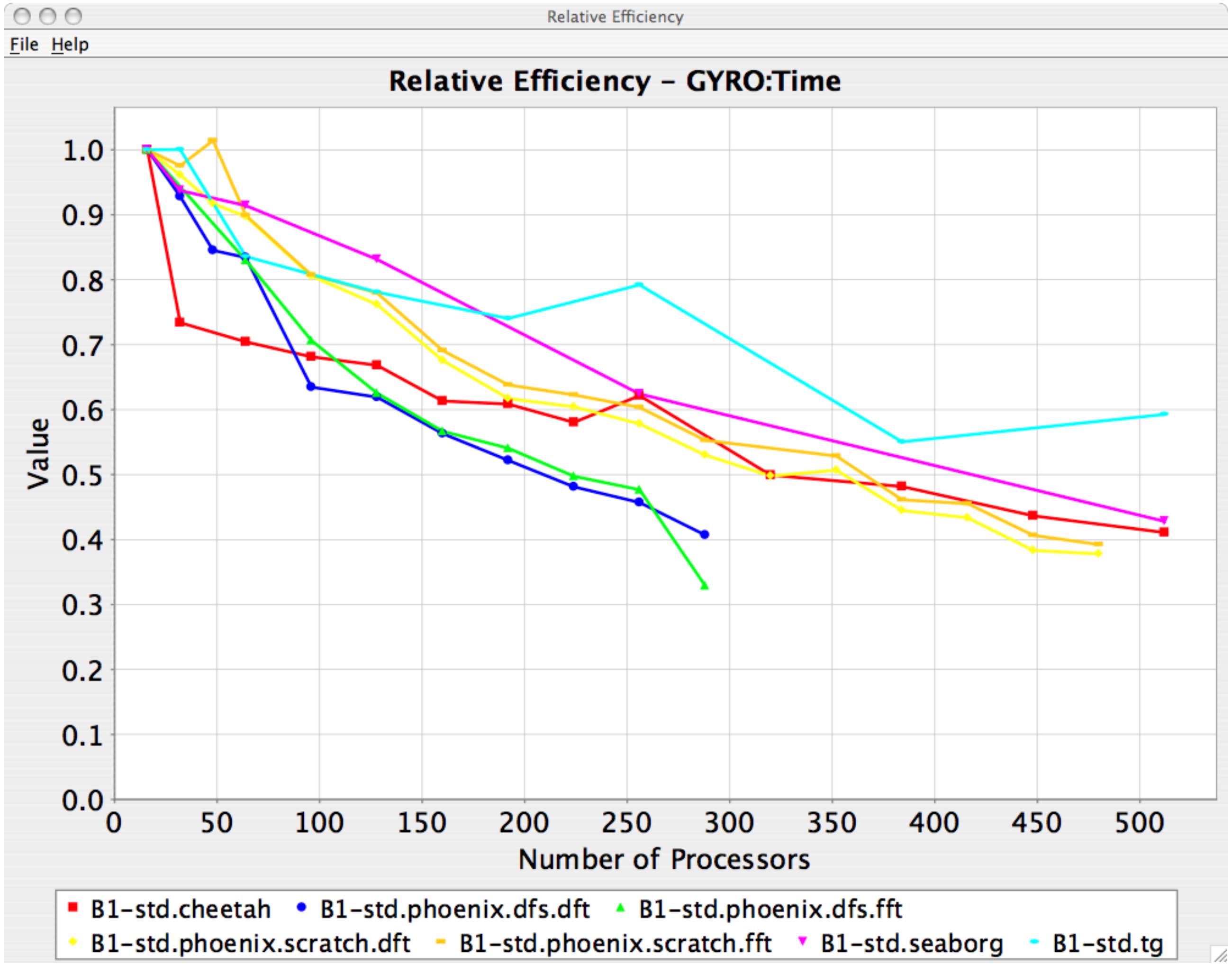
PerfExplorer - Interface



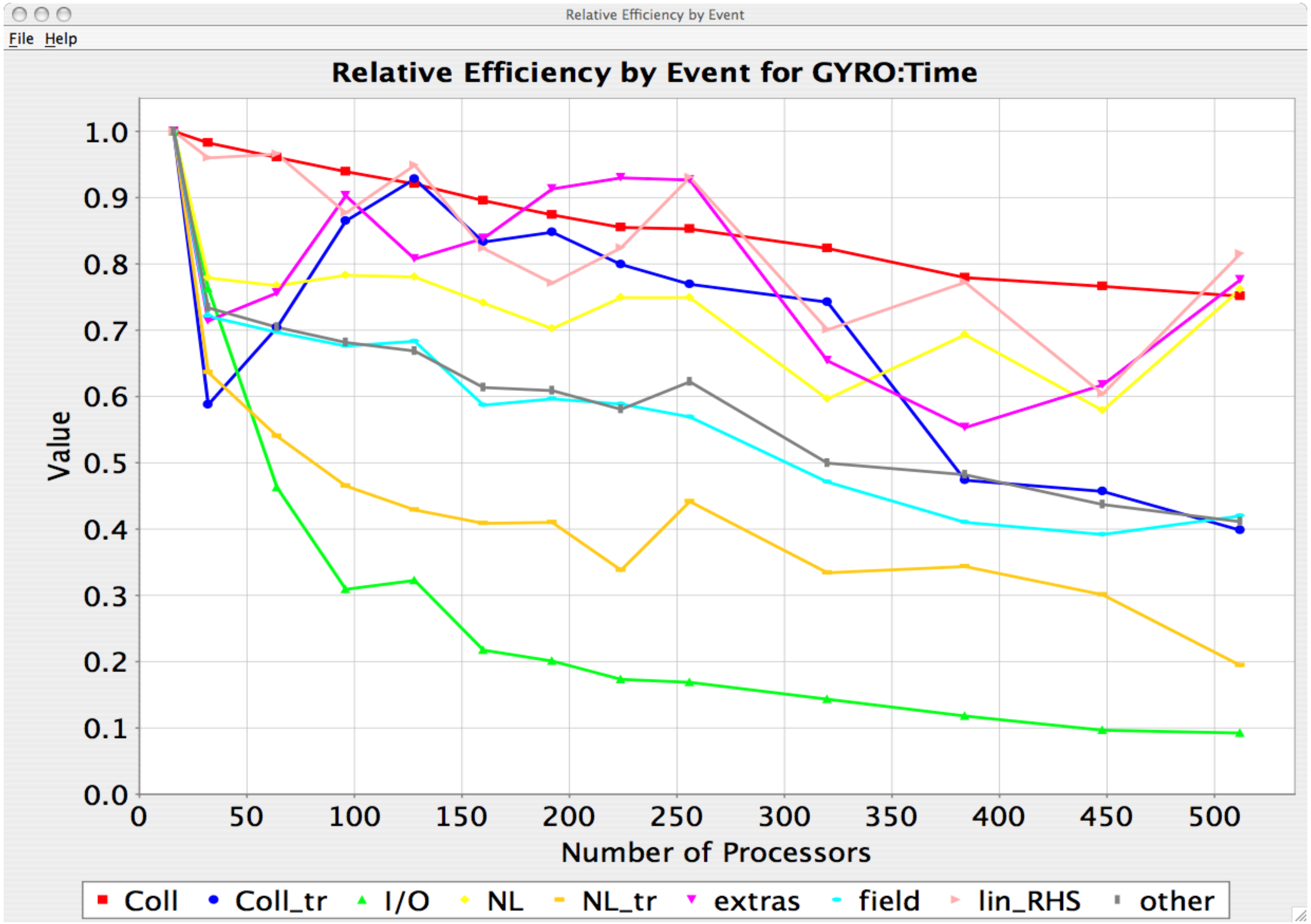
Field	Value
Name	B1-std-nl2.cheetah.noaffnosng
Experiment ID	16
system_name	
system_machine_type	
system_arch	
system_os	
system_memory_size	
system_processor_amt	
system_l1_cache_size	
system_l2_cache_size	
tem_userdata	
compiler_cc_version	
compiler_java_dirpath	
compiler_java_version	
compiler_userdata	
configure_prefix	
configure_arch	
configure_cpp	
configure_cc	
configure_jdk	
configure_profile	
configure_userdata	
userdata	

Select analysis

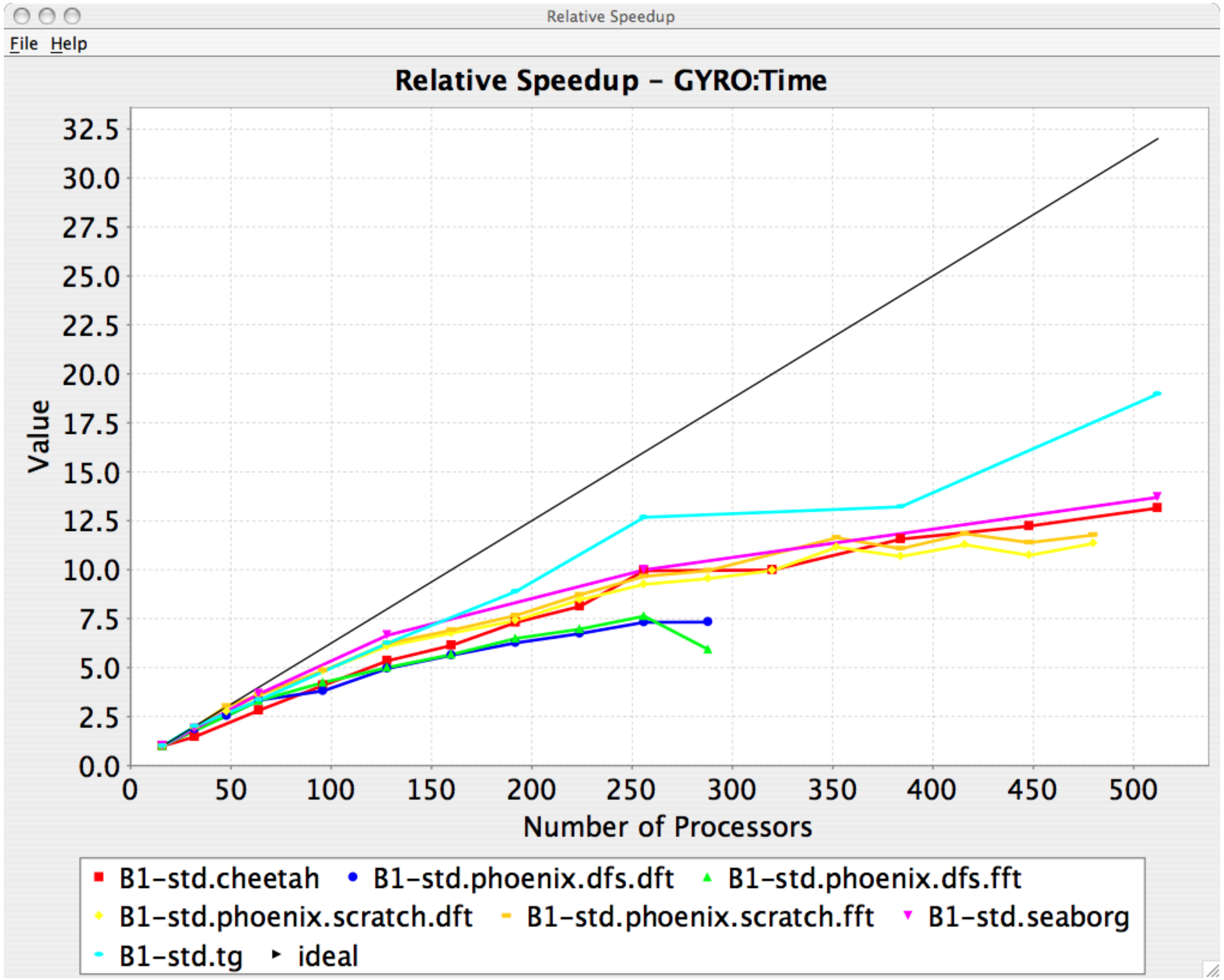
PerfExplorer - Relative Efficiency Plots



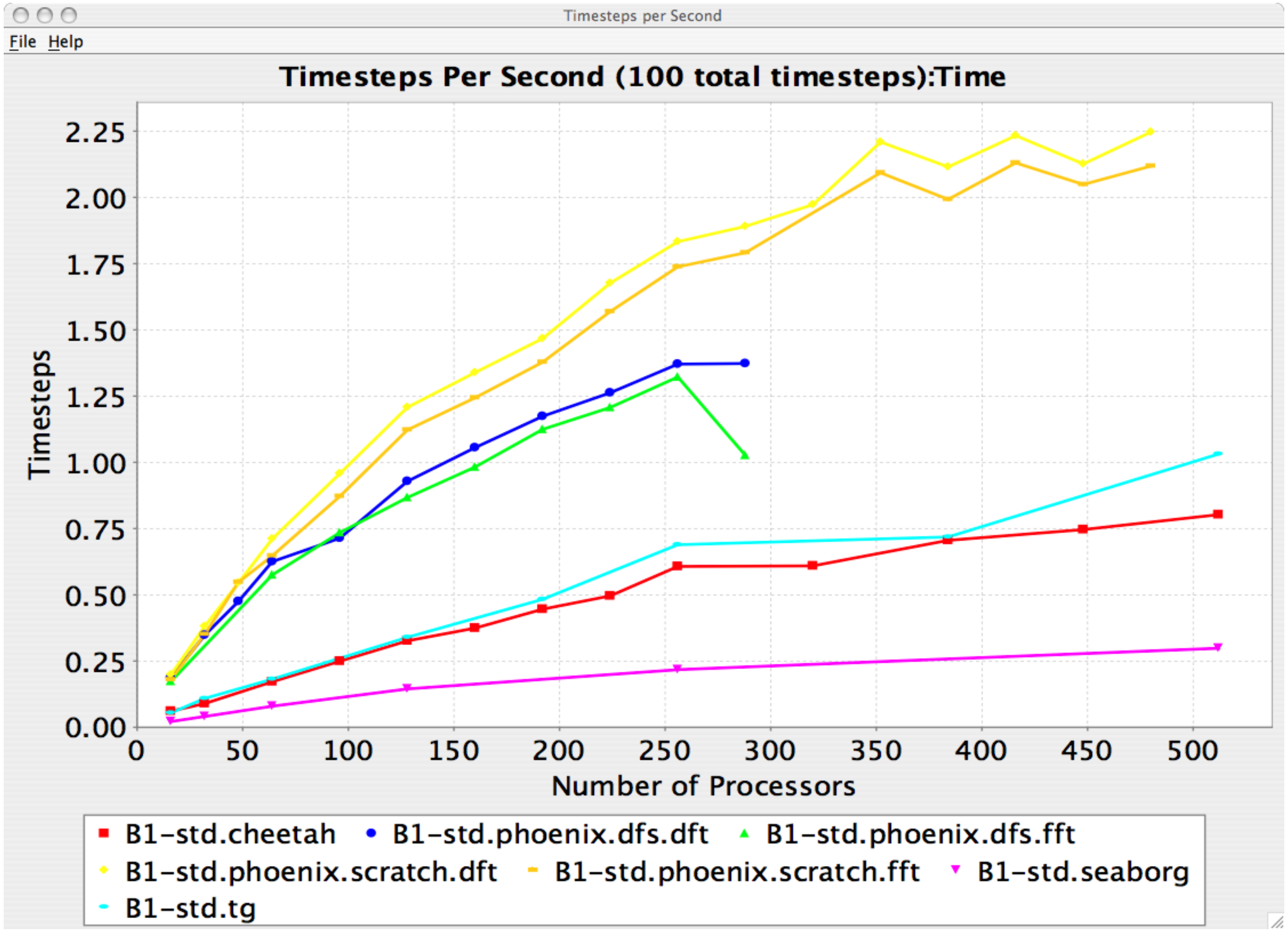
PerfExplorer - Relative Efficiency by Routine



PerfExplorer - Relative Speedup

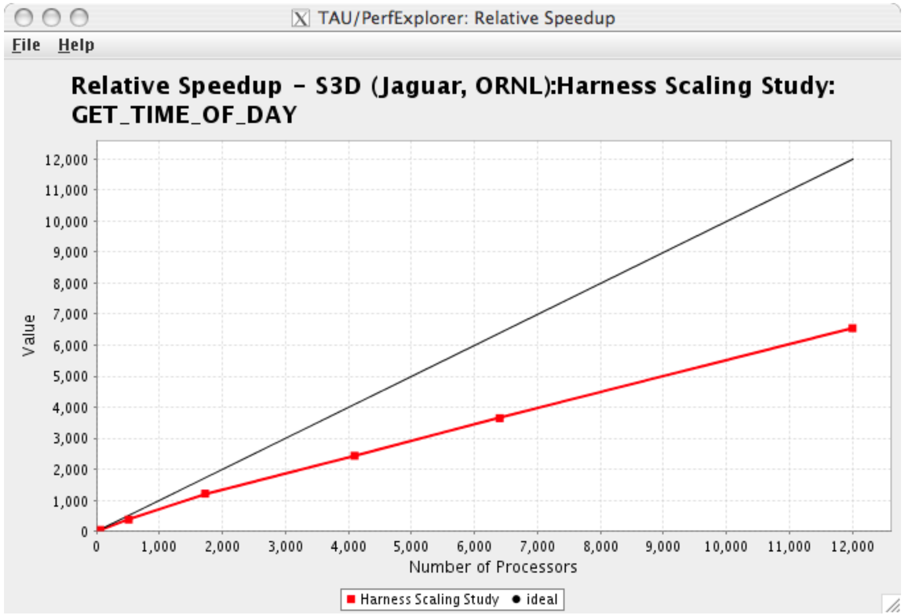


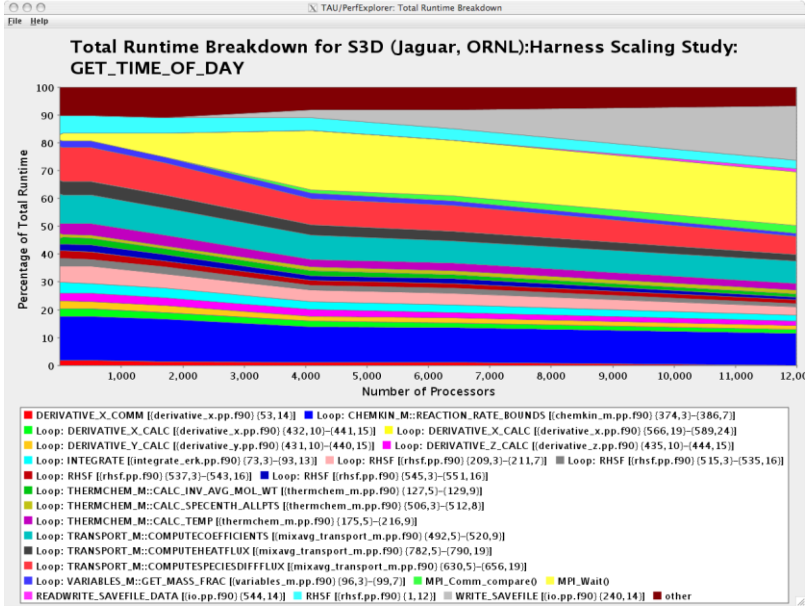
PerfExplorer - Timesteps Per Second



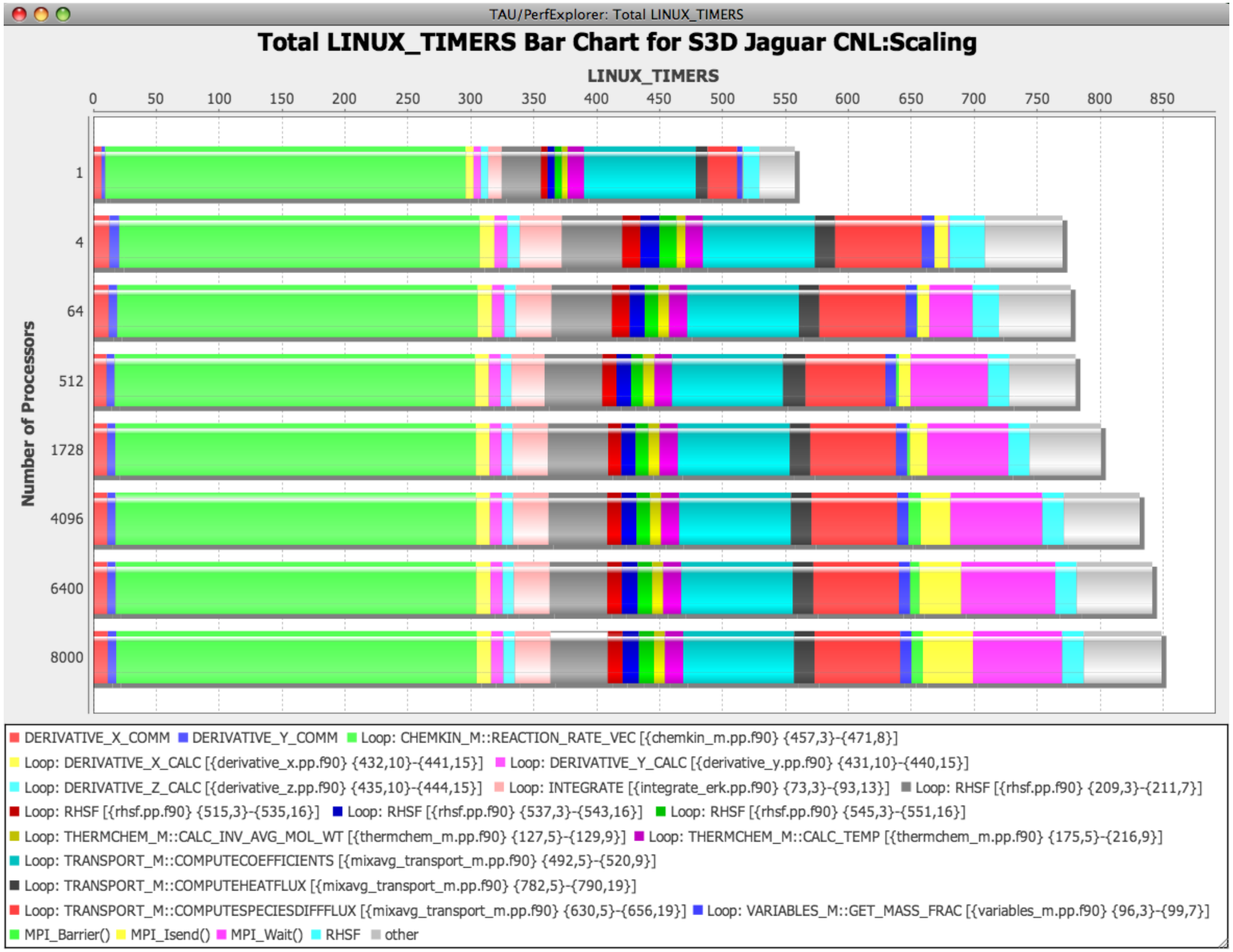
Evaluate Scalability

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

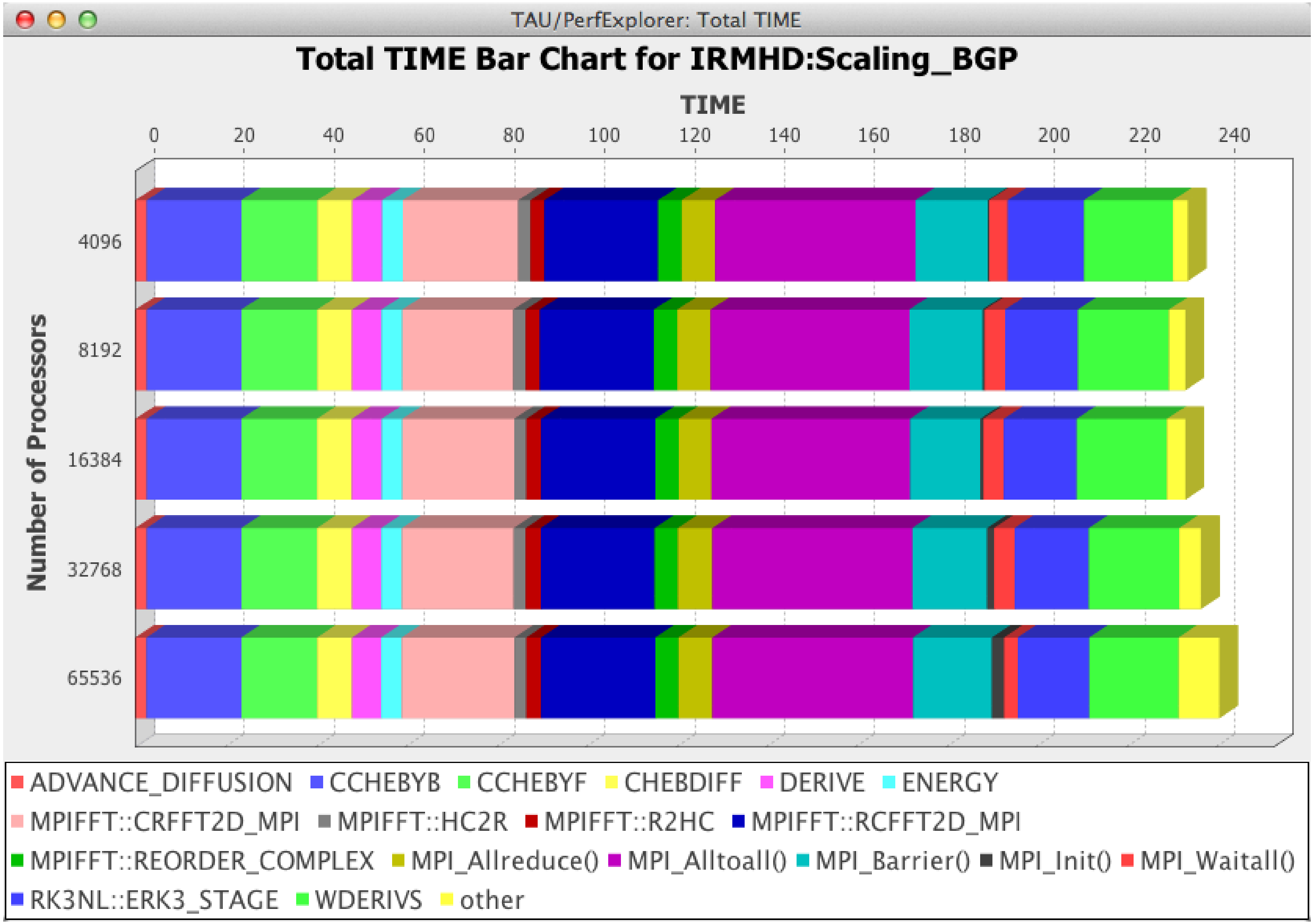




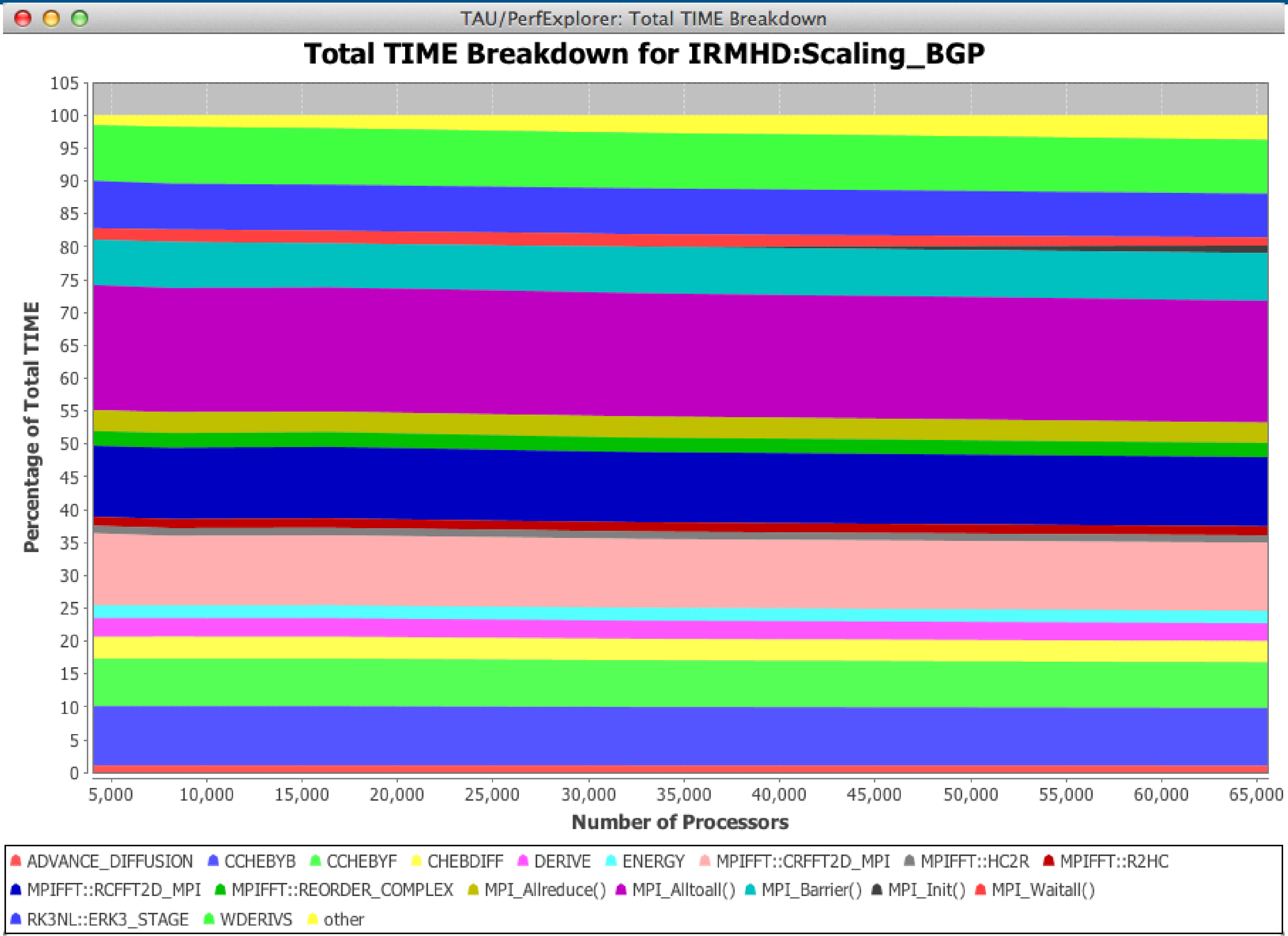
Evaluate Scalability



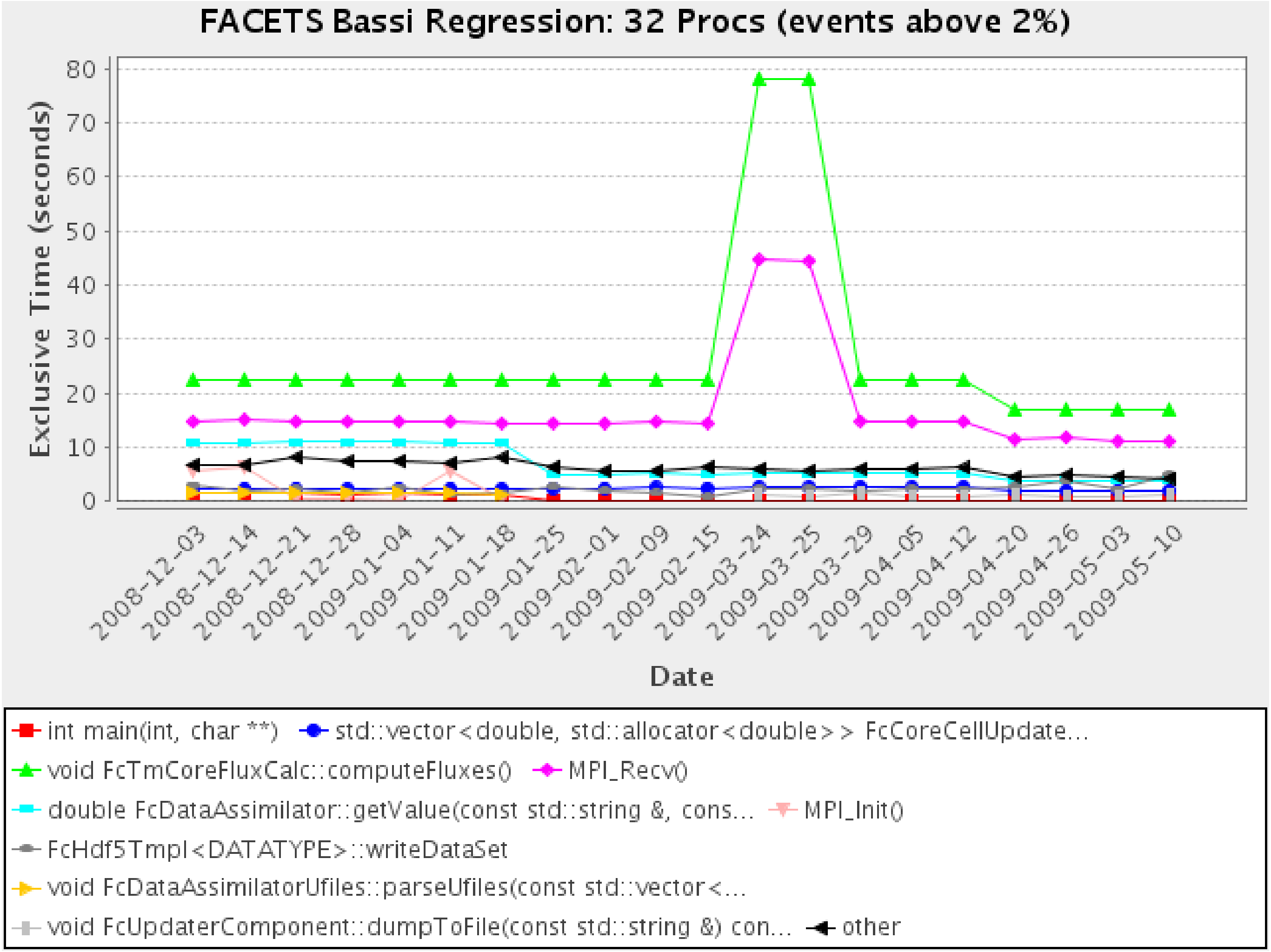
PerfExplorer



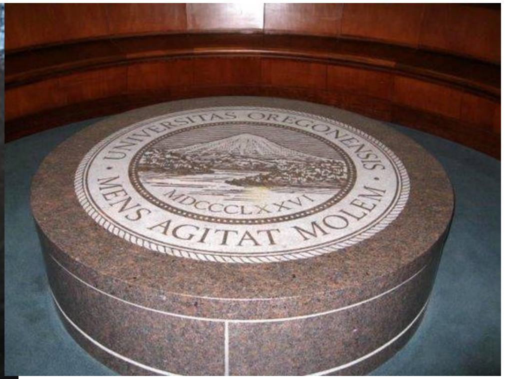
PerfExplorer



Performance Regression Testing



PRL, University of Oregon, Eugene



www.uoregon.edu

Support Acknowledgments

- U.S. Department of Energy (DOE)
  - Office of Science
  - PNNL, LBL, ORNL
  - ASC/NNSA, Tri-labs (LLNL,LANL, SNL)
- U.S. Department of Defense (DoD)
  - HPC Modernization Office (HPCMO)
- NSF Software Development for Cyberinfrastructure (SDCI)
- Juelich Supercomputing Center, NIC
- Argonne National Laboratory
- T.U. Dresden
- ParaTools, Inc.

Download TAU from U. Oregon

http://tau.uoregon.edu

http://www.hpclinux.com [LiveDVD, OVA]

Free download, open source, BSD license

Typical performance bottlenecks and how they can be identified

Bert Wesarg

ZIH, Technische Universität Dresden

Outline

- **Case I:**
  - **Load imbalances in OpenMP codes**

- **Case II:**
  - **Communication and computation overlapping in MPI codes**

- **Note:** We won’t do the complete performance engineering cycle here.

Case I: Sparse Matrix Vector Multiplication

$$\begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{n1} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

- A sparse matrix is a matrix populated primarily with zeros
- Only non-zero elements of a_{ij} are saved efficiently in memory
- Algorithm

```
foreach row r in A
  y[r.x] = 0
  foreach non-zero element e in row
    y[r.x] += e.value * x[e.y]
```

Case I: Sparse Matrix Vector Multiplication

- Naive OpenMP Algorithm

```
#pragma omp parallel for
foreach row r in A
  y[r.x] = 0
  foreach non-zero element e in row
    y[r.x] += e.value * x[e.y]
```

- Distributes the rows of A evenly across the threads in the parallel region
- The distribution of the non-zero elements may influence the load balance in the parallel application

Case I: Load imbalances in OpenMP codes

- Measuring the static OpenMP application

```
% cd ~/Bottlenecks/smxv
% make PREP=scorep
scorep   gcc -fopenmp -DLITTLE_ENDIAN \
    -DFUNCTION_INC='"y_Ax-omp.inc.c"' -DFUNCTION=y_Ax_omp \
    -o smxv-omp smxv.c -lm
scorep   gcc -fopenmp -DLITTLE_ENDIAN \
    -DFUNCTION_INC='"y_Ax-omp-dynamic.inc.c"‘ \
    -DFUNCTION=y_Ax_omp_dynamic -o smxv-omp-dynamic smxv.c -lm
% OMP_NUM_THREADS=8 scan -t ./smxv-omp yax_large.bin
```

Case I: Load imbalances in OpenMP codes: Profile

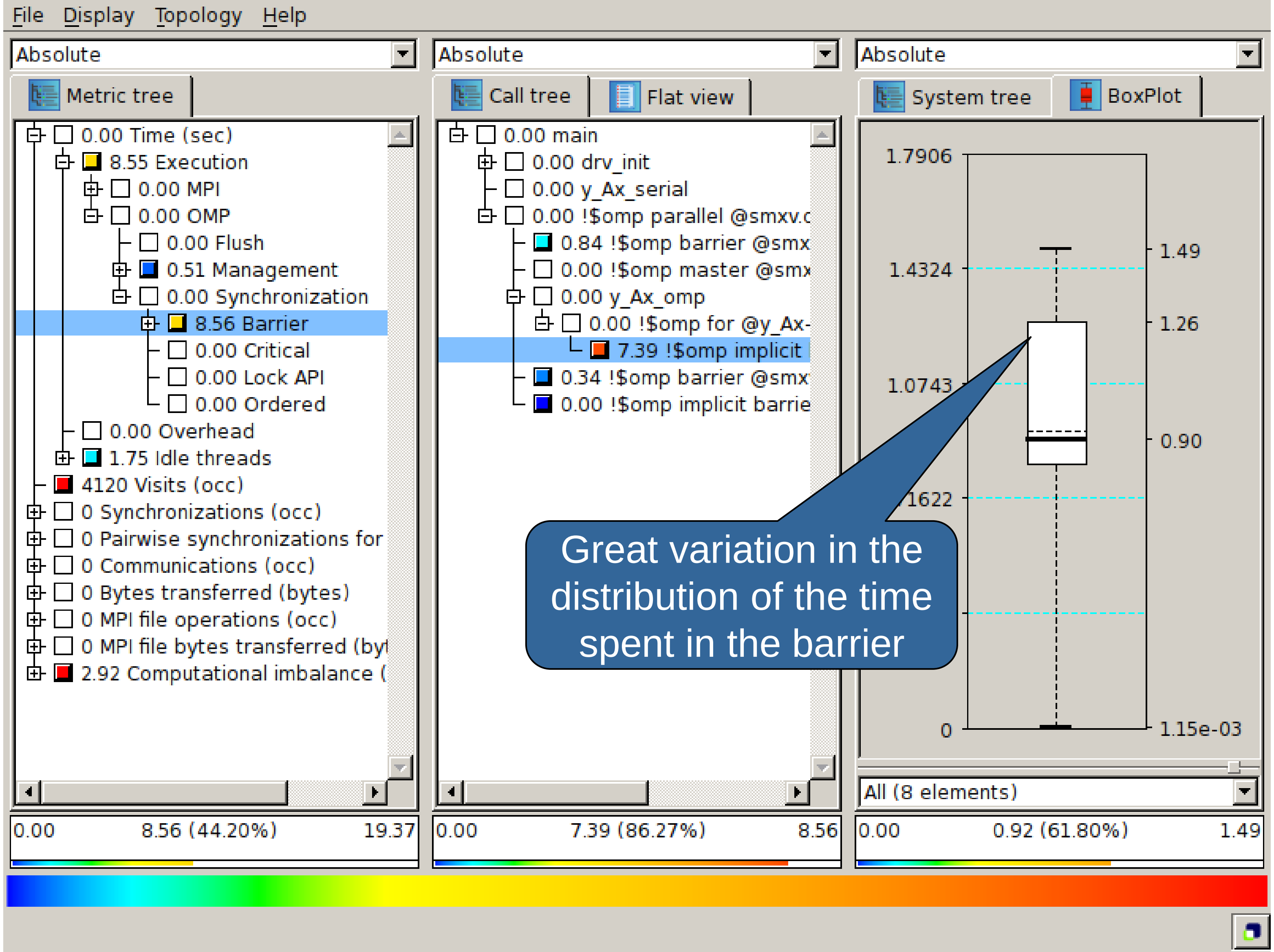
- Two metrics which indicate load imbalances:
  - Time spent in OpenMP barriers
  - Computational imbalance

- Open prepared measurement on the LiveDVD with Cube

```
% cube ~/Bottlenecks/smxv/scorep_smxv-omp_large/trace.cubex
```

[CUBE GUI showing trace analysis report]

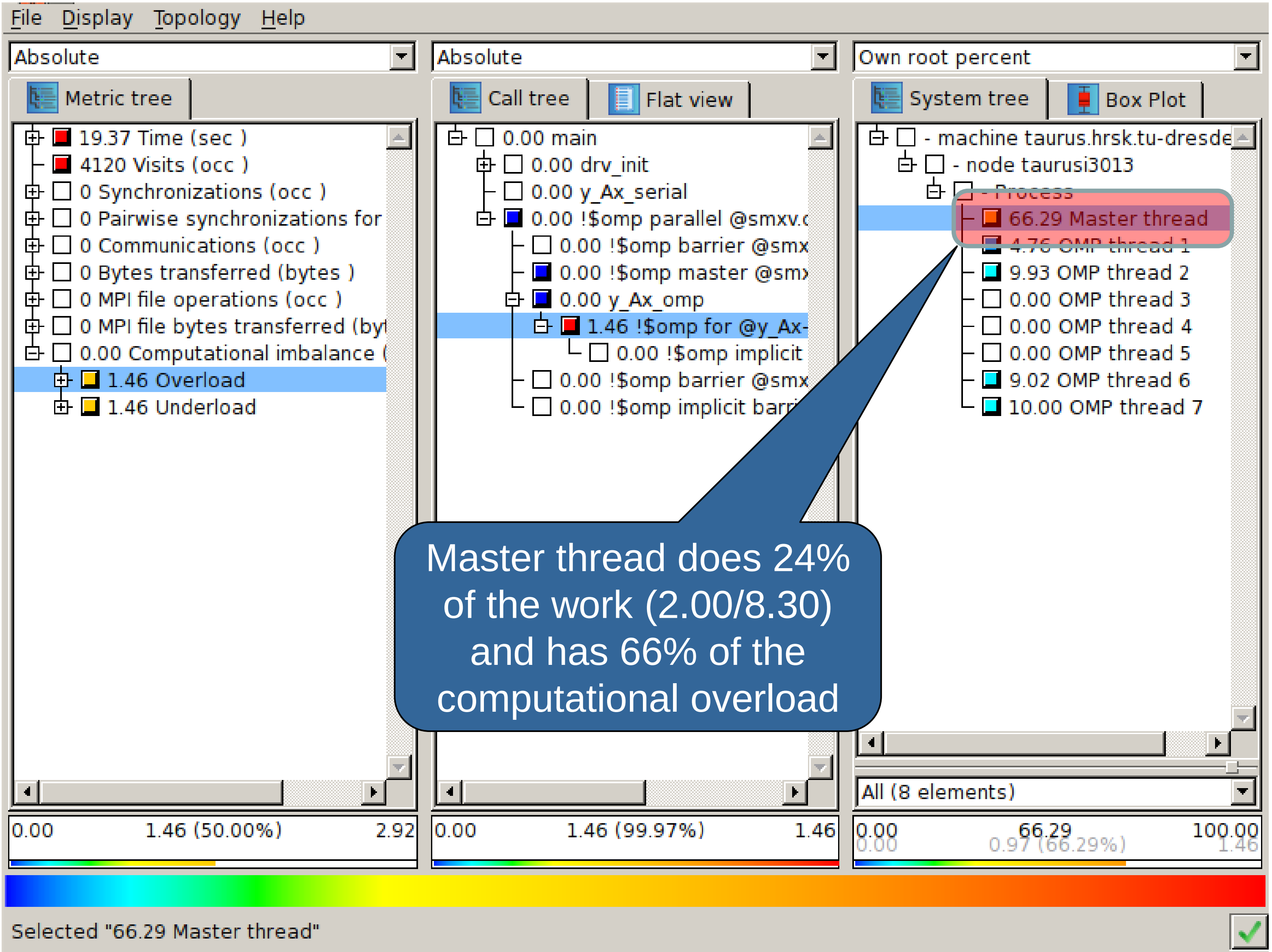
Case I: Time spent in OpenMP barriers



```
Absolute                              Absolute                              Absolute
Metric tree                           Call tree | Flat view                 System tree | BoxPlot

□ 0.00 Time (sec)                     □ 0.00 main
  □ 8.55 Execution                      □ 0.00 drv_init
    □ 0.00 MPI                          □ 0.00 y_Ax_serial
    □ 0.00 OMP                          □ 0.00 !$omp parallel @smxv.c
      □ 0.00 Flush                        □ 0.84 !$omp barrier @smx
      □ 0.51 Management                   □ 0.00 !$omp master @smx
      □ 0.00 Synchronization              □ 0.00 y_Ax_omp
        □ 8.56 Barrier                      □ 0.00 !$omp for @y_Ax-
        □ 0.00 Critical                       □ 7.39 !$omp implicit
        □ 0.00 Lock API                   □ 0.34 !$omp barrier @smx
        □ 0.00 Ordered                    □ 0.00 !$omp implicit barrie
    □ 0.00 Overhead
    □ 1.75 Idle threads
  □ 4120 Visits (occ)
□ 0 Synchronizations (occ)
□ 0 Pairwise synchronizations for
□ 0 Communications (occ)
□ 0 Bytes transferred (bytes)
□ 0 MPI file operations (occ)
□ 0 MPI file bytes transferred (byt
□ 2.92 Computational imbalance (

                                                                            All (8 elements)
0.00      8.56 (44.20%)      19.37    0.00      7.39 (86.27%)      8.56    0.00      0.92 (61.80%)      1.49
```

Great variation in the distribution of the time spent in the barrier

Case I: Computational imbalance



File Display Topology Help

Absolute | Absolute | Own root percent

Metric tree | Call tree | Flat view | System tree | Box Plot

Metric tree:

- 19.37 Time (sec)
- 4120 Visits (occ)
- 0 Synchronizations (occ)
- 0 Pairwise synchronizations for
- 0 Communications (occ)
- 0 Bytes transferred (bytes)
- 0 MPI file operations (occ)
- 0 MPI file bytes transferred (byt
- 0.00 Computational imbalance (
  - 1.46 Overload
  - 1.46 Underload

Call tree:

- 0.00 main
  - 0.00 drv_init
  - 0.00 y_Ax_serial
  - 0.00 !$omp parallel @smxv.c
    - 0.00 !$omp barrier @smx
    - 0.00 !$omp master @smx
    - 0.00 y_Ax_omp
      - 1.46 !$omp for @y_Ax-
        - 0.00 !$omp implicit
    - 0.00 !$omp barrier @smx
    - 0.00 !$omp implicit barr

System tree:

- - machine taurus.hrsk.tu-dresde
  - - node taurusi3013
    - - Process
      - 66.29 Master thread
      - 4.76 OMP thread 1
      - 9.93 OMP thread 2
      - 0.00 OMP thread 3
      - 0.00 OMP thread 4
      - 0.00 OMP thread 5
      - 9.02 OMP thread 6
      - 10.00 OMP thread 7

All (8 elements)

0.00 1.46 (50.00%) 2.92 | 0.00 1.46 (99.97%) 1.46 | 0.00 66.29 100.00
0.00 0.97 (66.29%) 1.46

Selected "66.29 Master thread"

Master thread does 24% of the work (2.00/8.30) and has 66% of the computational overload

Case I: Sparse Matrix Vector Multiplication

- Improved OpenMP Algorithm

```
#pragma omp parallel for schedule(dynamic,1000)
foreach row r in A
  y[r.x] = 0
  foreach non-zero element e in row
    y[r.x] += e.value * x[e.y]
```

- Distributes the rows of A *dynamically* across the threads in the parallel region

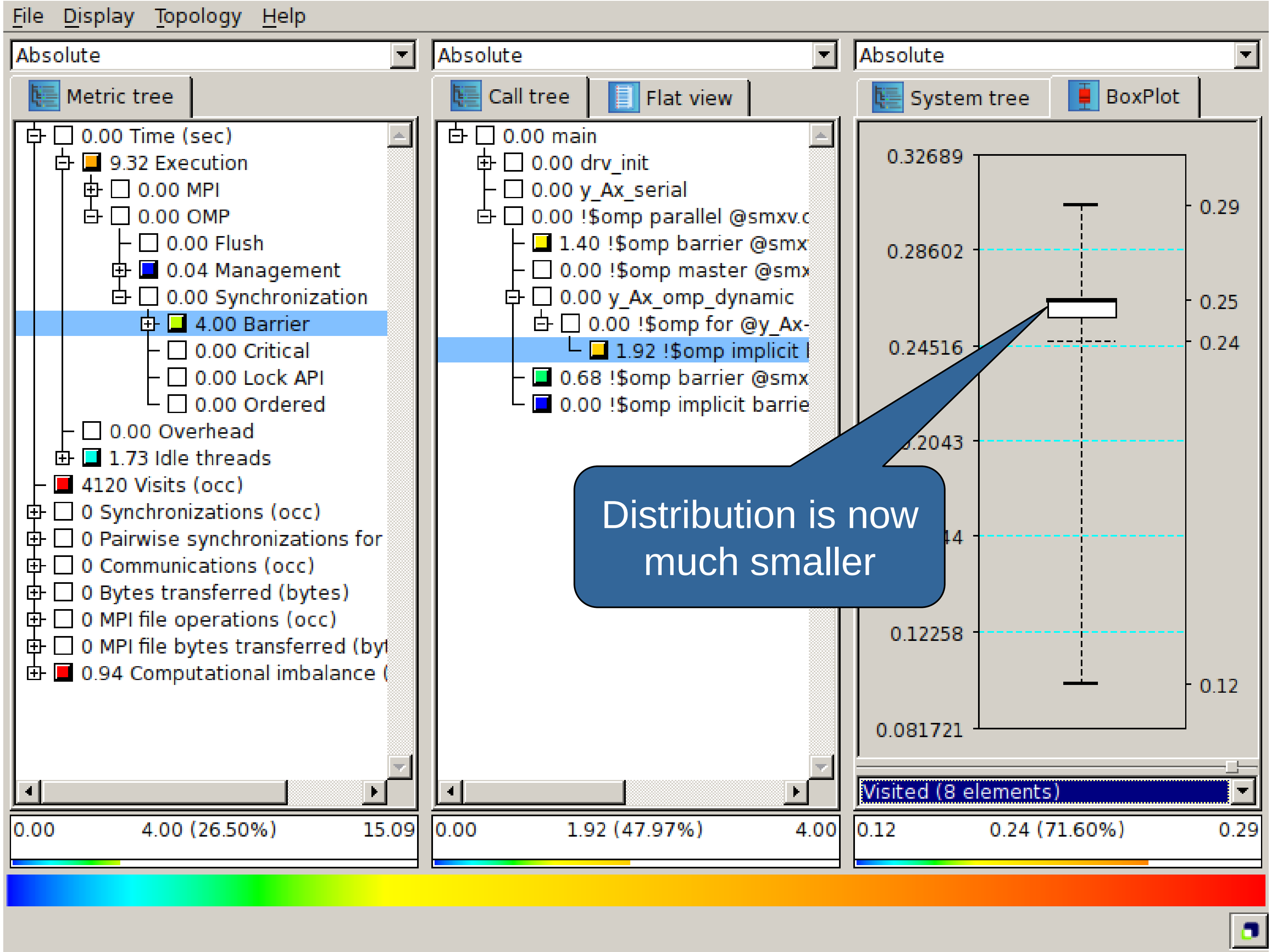
Case I: Profile Analysis

- Two metrics which indicate load imbalances
  - Time spent in OpenMP barriers
  - Computational imbalance

- Open prepared measurement on the LiveDVD with Cube

```
% cube ~/Bottlenecks/smxv/scorep_smxv-omp-dynamic_large/trace.cubex

            [CUBE GUI showing trace analysis report]
```

Case I: Time spent in OpenMP barriers



File Display Topology Help

Absolute | Absolute | Absolute

Metric tree

- ⊟ □ 0.00 Time (sec)
  - ⊟ ■ 9.32 Execution
    - ⊞ □ 0.00 MPI
    - ⊟ □ 0.00 OMP
      - □ 0.00 Flush
      - ⊞ ■ 0.04 Management
      - ⊟ □ 0.00 Synchronization
        - ⊞ ■ 4.00 Barrier
        - □ 0.00 Critical
        - □ 0.00 Lock API
        - □ 0.00 Ordered
    - □ 0.00 Overhead
    - ⊞ ■ 1.73 Idle threads
- ■ 4120 Visits (occ)
- ⊞ □ 0 Synchronizations (occ)
- ⊞ □ 0 Pairwise synchronizations for
- ⊞ □ 0 Communications (occ)
- ⊞ □ 0 Bytes transferred (bytes)
- ⊞ □ 0 MPI file operations (occ)
- ⊞ □ 0 MPI file bytes transferred (byt
- ⊞ ■ 0.94 Computational imbalance (

0.00 4.00 (26.50%) 15.09

Call tree | Flat view

- ⊟ □ 0.00 main
  - ⊞ □ 0.00 drv_init
  - □ 0.00 y_Ax_serial
  - ⊟ □ 0.00 !$omp parallel @smxv.c
    - ■ 1.40 !$omp barrier @smx
    - □ 0.00 !$omp master @smx
    - ⊟ □ 0.00 y_Ax_omp_dynamic
      - ⊟ □ 0.00 !$omp for @y_Ax-
        - ■ 1.92 !$omp implicit I
    - ■ 0.68 !$omp barrier @smx
    - ■ 0.00 !$omp implicit barrie

0.00 1.92 (47.97%) 4.00

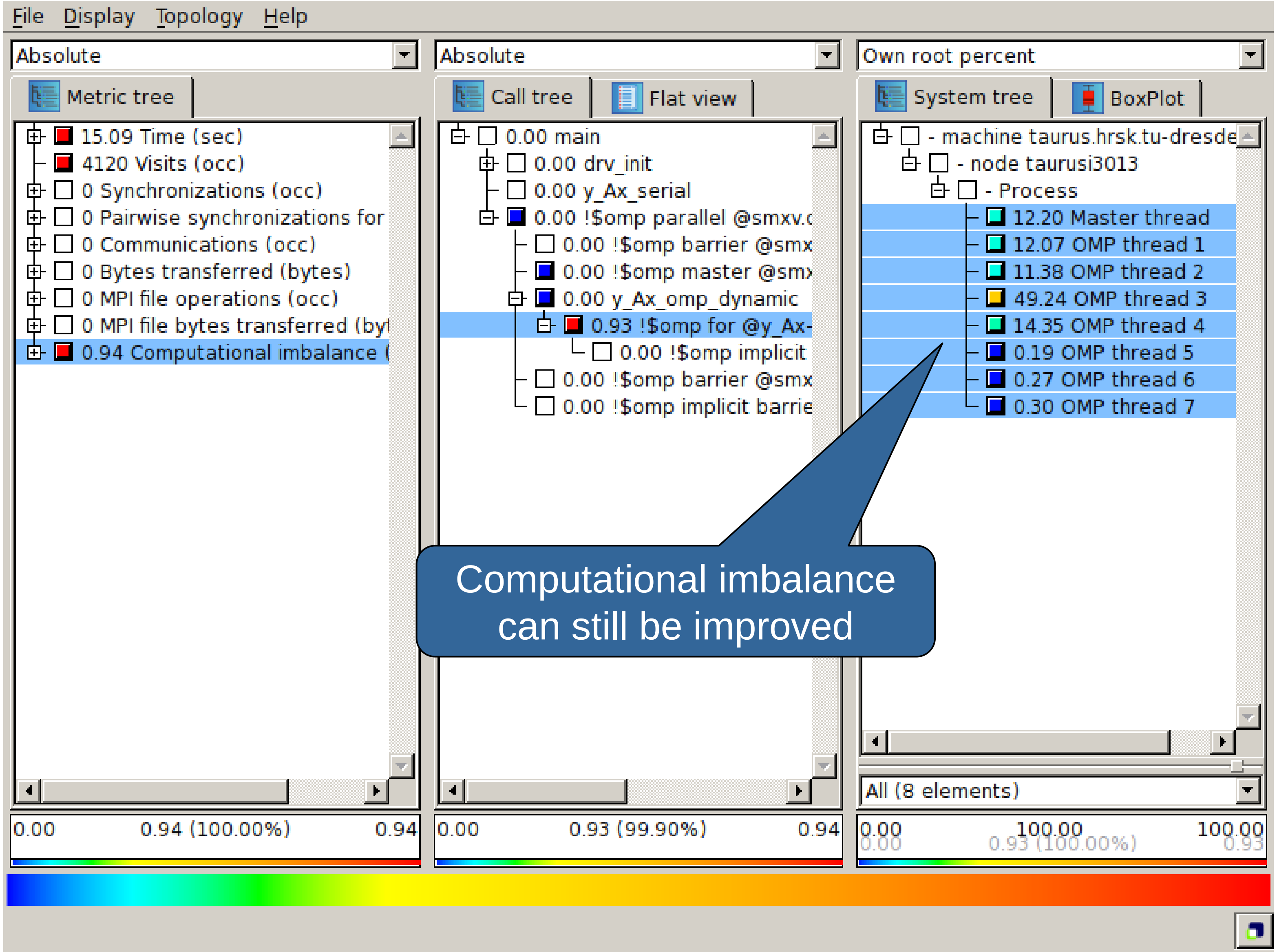
System tree | BoxPlot

Visited (8 elements)

0.12 0.24 (71.60%) 0.29

Distribution is now much smaller

Case I: Computational imbalance



```
15.09 Time (sec)
4120 Visits (occ)
0 Synchronizations (occ)
0 Pairwise synchronizations for
0 Communications (occ)
0 Bytes transferred (bytes)
0 MPI file operations (occ)
0 MPI file bytes transferred (by
0.94 Computational imbalance (
```

```
0.00 main
  0.00 drv_init
  0.00 y_Ax_serial
  0.00 !$omp parallel @smxv.c
    0.00 !$omp barrier @smx
    0.00 !$omp master @smx
    0.00 y_Ax_omp_dynamic
      0.93 !$omp for @y_Ax-
        0.00 !$omp implicit
    0.00 !$omp barrier @smx
    0.00 !$omp implicit barrie
```

```
- machine taurus.hrsk.tu-dresde
  - node taurusi3013
    - Process
      12.20 Master thread
      12.07 OMP thread 1
      11.38 OMP thread 2
      49.24 OMP thread 3
      14.35 OMP thread 4
      0.19 OMP thread 5
      0.27 OMP thread 6
      0.30 OMP thread 7
```

```
0.00        0.94 (100.00%)        0.94
0.00        0.93 (99.90%)         0.94
0.00        100.00                100.00
0.00        0.93 (100.00%)        0.93
```

Computational imbalance can still be improved

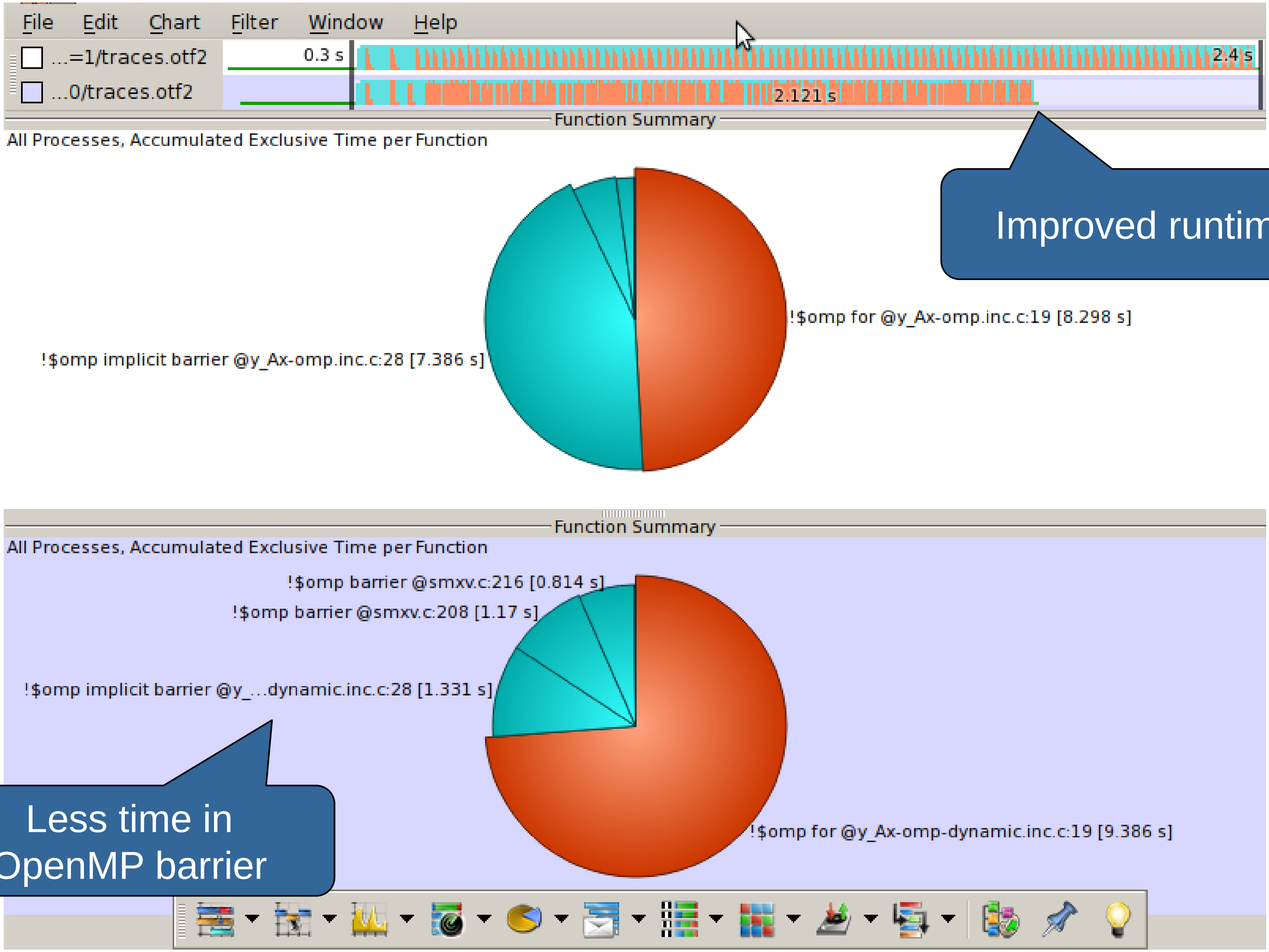
Case I: Trace Comparison

- Open prepared measurement on the LiveDVD with Vampir

```
% vampir ~/Bottlenecks/smxv/scorep_smxv-omp_large/traces.otf2 \
         ~/Bottlenecks/smxv/scorep_smxv-omp-dynamic_large/traces.otf2
```

[Vampir GUI showing trace]

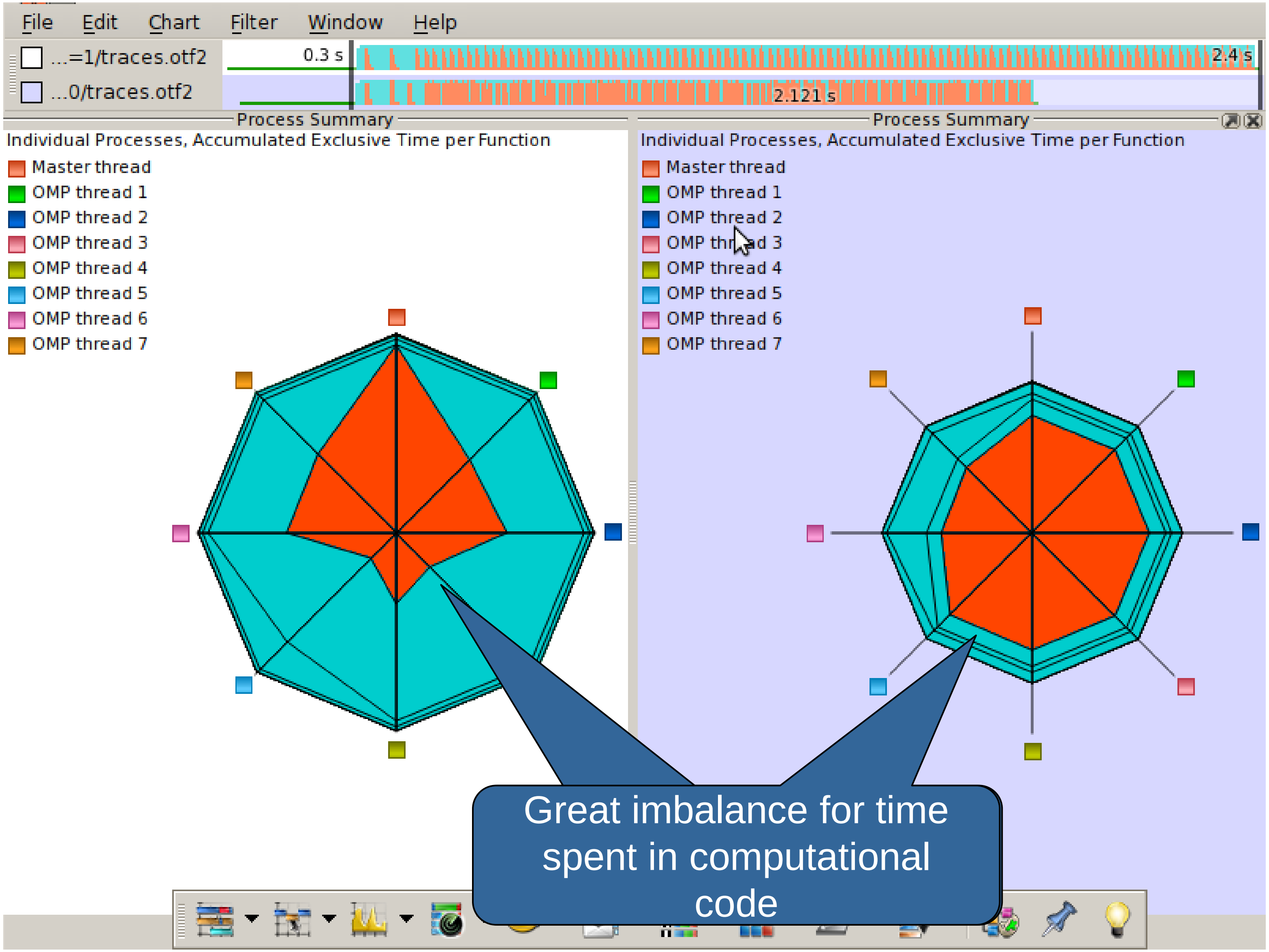
Case I: Time spent in OpenMP barriers


File
Edit
Chart
Filter
Window
Help
□ ...=1/traces.otf2
□ ...0/traces.otf2
0.3 s
2.4 s
2.121 s
Function Summary
All Processes, Accumulated Exclusive Time per Function
!$omp for @y_Ax-omp.inc.c:19 [8.298 s]
!$omp implicit barrier @y_Ax-omp.inc.c:28 [7.386 s]
Function Summary
All Processes, Accumulated Exclusive Time per Function
!$omp barrier @smxv.c:216 [0.814 s]
!$omp barrier @smxv.c:208 [1.17 s]
!$omp implicit barrier @y_...dynamic.inc.c:28 [1.331 s]
!$omp for @y_Ax-omp-dynamic.inc.c:19 [9.386 s]

Improved runtime

Less time in OpenMP barrier

Case I: Computational imbalance



Great imbalance for time spent in computational code

Outline

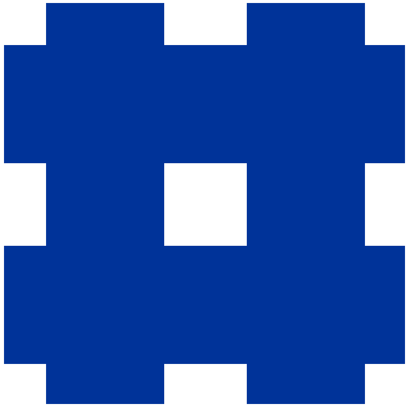
- Case I:
  - Load imbalances in OpenMP codes

- **Case II:**
  - **Communication and computation overlapping in MPI codes**

Case II: Heat Conduction Simulation

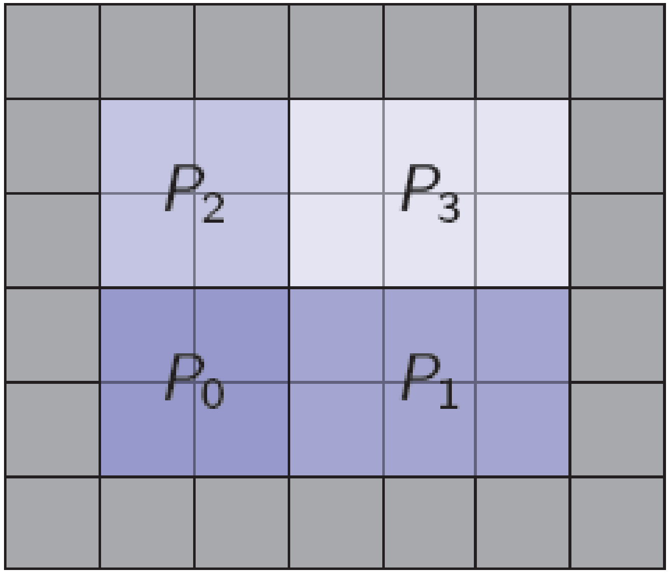
- Calculating the heat conduction at each time step
- Discretized formula for space dx, dy and time dt

$$\theta_{i,j}^{t+1} = \theta_{i,j}^{t} + \left(\frac{\theta_{i+1,j}^{t} - 2\theta_{i,j}^{t} + 2\theta_{i-1,j}^{t}}{dx^2} + \frac{\theta_{i,j+1}^{t} - 2\theta_{i,j}^{t} + 2\theta_{i,j-1}^{t}}{dy^2} \right) \cdot k \cdot dt$$



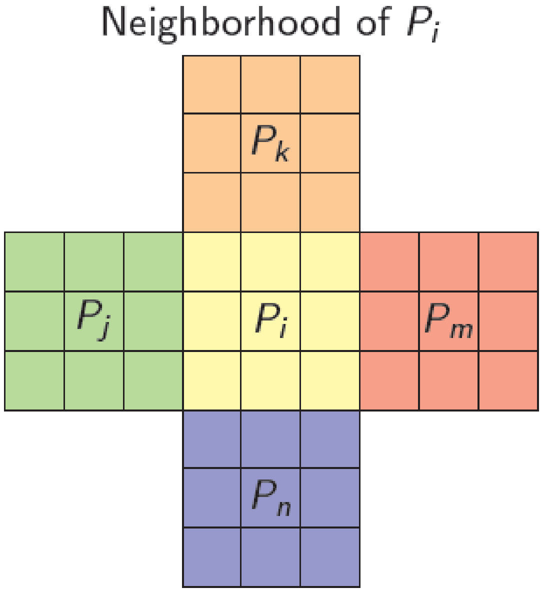
Case II: Heat Conduction Simulation

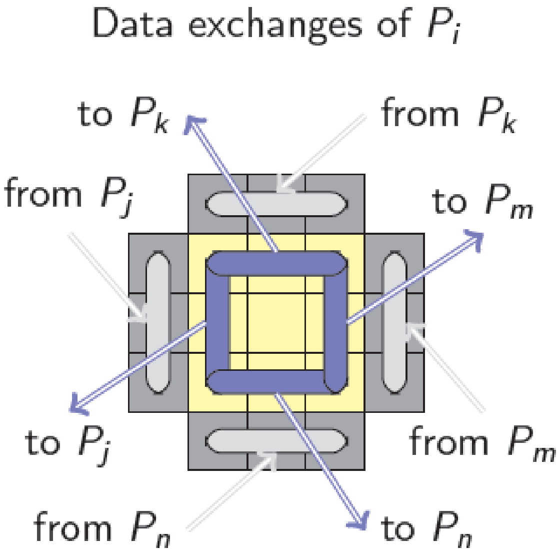
- Application uses MPI for boundary exchange
- Simulation grid is distributed across MPI ranks



Case II: Heat Conduction Simulation

- Ranks need to exchange boundaries before next iteration step





Case II: Profile Analysis

- MPI algorithm

```
foreach step in [1:nsteps]
  exchangeBoundaries
  computeHeatConduction
```

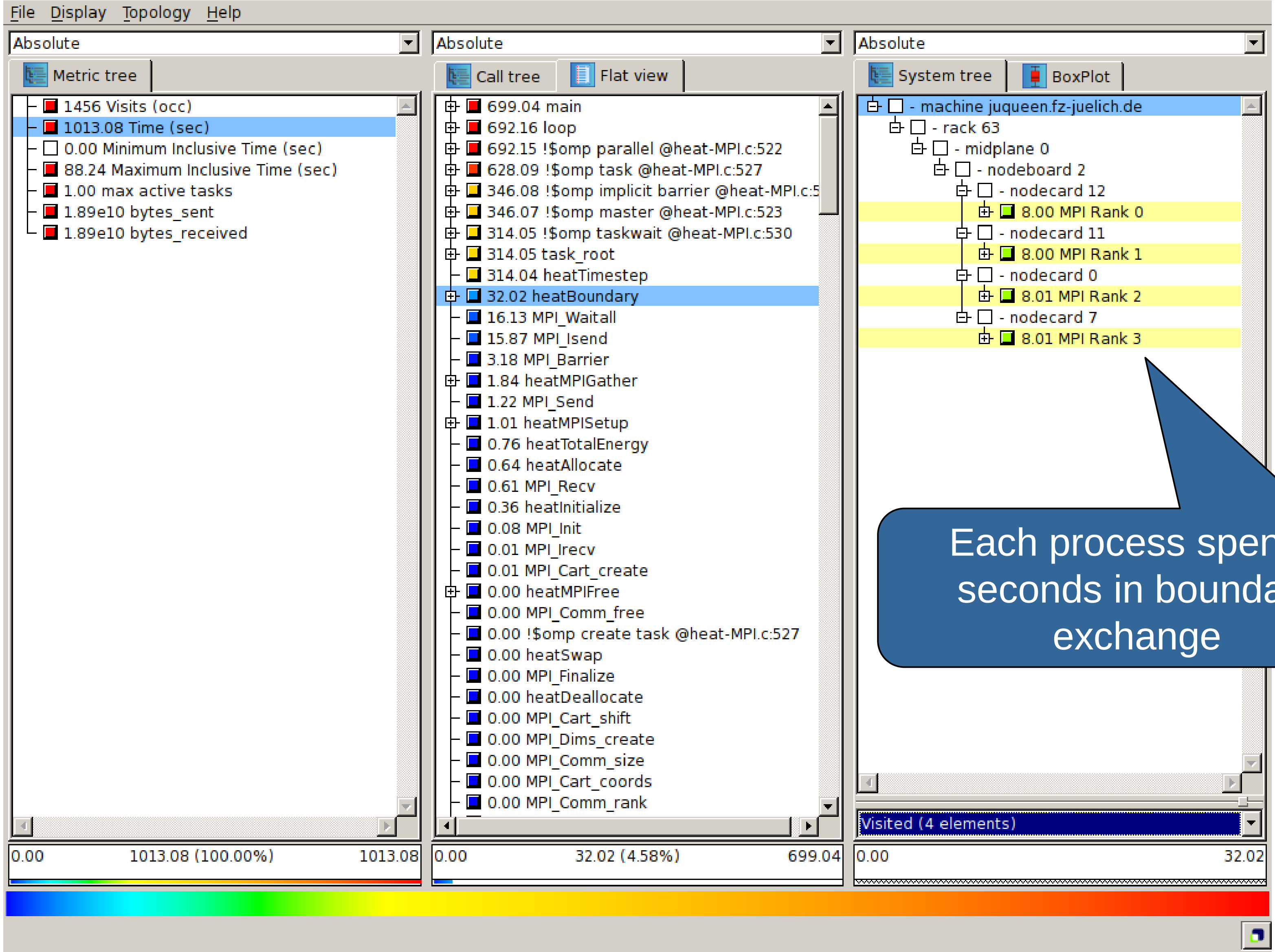
- Building and measuring the heat conduction application

```
% cd ~/Bottlenecks/heat
% make PREP=‘scorep --user’
  [... make output ...]
% scan mpirun –np 16 ./heat-MPI 3072 32
```

- Open prepared measurement on the LiveDVD with Cube

```
% cube ~/Bottlenecks/heat/scorep_heat-MPI_small/profile.cubex

               [CUBE GUI showing trace analysis report]
```

Case II: Time spent in Boundary Exchange



```
1456 Visits (occ)
1013.08 Time (sec)
0.00 Minimum Inclusive Time (sec)
88.24 Maximum Inclusive Time (sec)
1.00 max active tasks
1.89e10 bytes_sent
1.89e10 bytes_received
```

```
699.04 main
692.16 loop
692.15 !$omp parallel @heat-MPI.c:522
628.09 !$omp task @heat-MPI.c:527
346.08 !$omp implicit barrier @heat-MPI.c:5
346.07 !$omp master @heat-MPI.c:523
314.05 !$omp taskwait @heat-MPI.c:530
314.05 task_root
314.04 heatTimestep
32.02 heatBoundary
16.13 MPI_Waitall
15.87 MPI_Isend
3.18 MPI_Barrier
1.84 heatMPIGather
1.22 MPI_Send
1.01 heatMPISetup
0.76 heatTotalEnergy
0.64 heatAllocate
0.61 MPI_Recv
0.36 heatInitialize
0.08 MPI_Init
0.01 MPI_Irecv
0.01 MPI_Cart_create
0.00 heatMPIFree
0.00 MPI_Comm_free
0.00 !$omp create task @heat-MPI.c:527
0.00 heatSwap
0.00 MPI_Finalize
0.00 heatDeallocate
0.00 MPI_Cart_shift
0.00 MPI_Dims_create
0.00 MPI_Comm_size
0.00 MPI_Cart_coords
0.00 MPI_Comm_rank
```

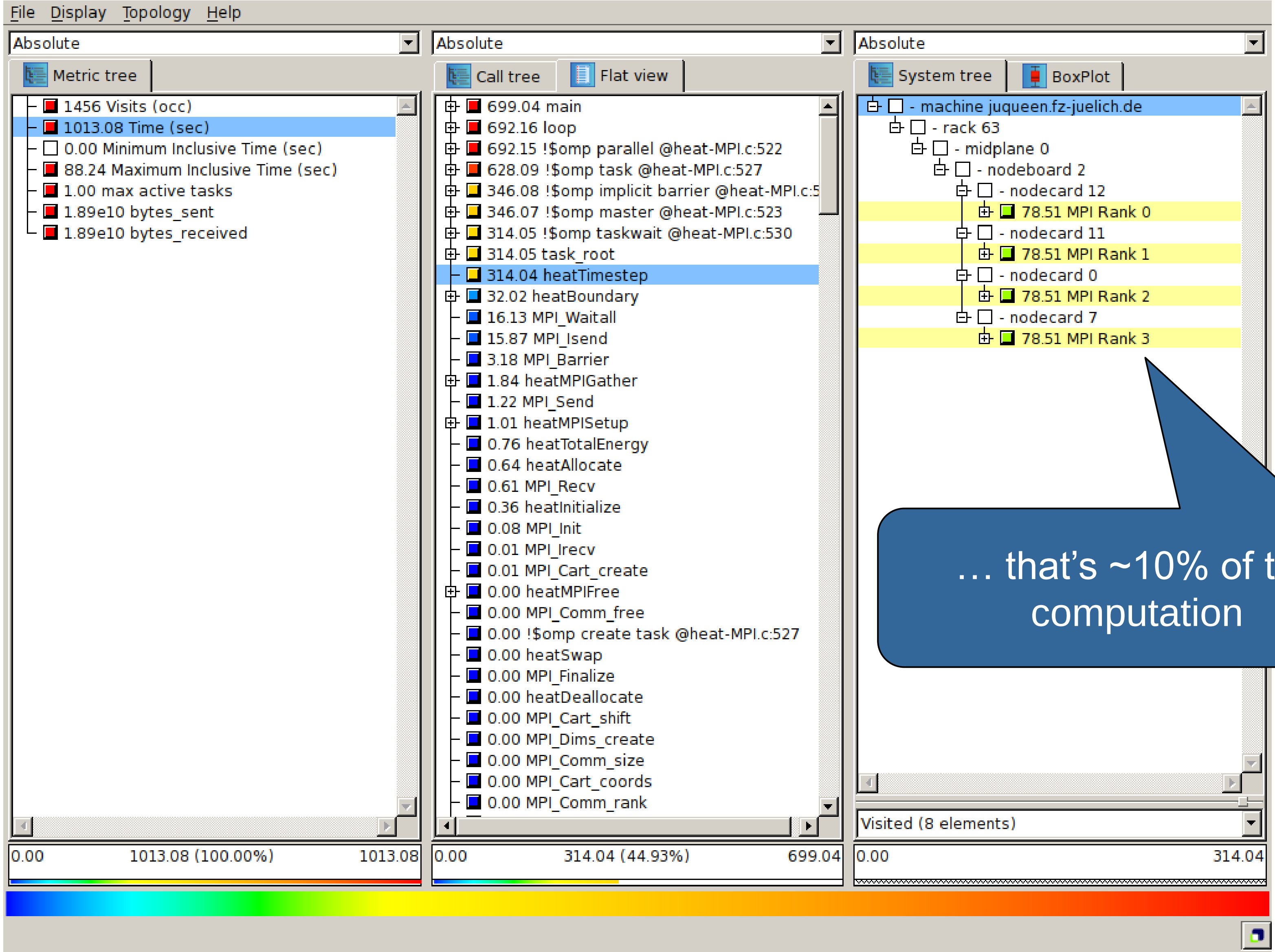
```
- machine juqueen.fz-juelich.de
  - rack 63
    - midplane 0
      - nodeboard 2
        - nodecard 12
          8.00 MPI Rank 0
        - nodecard 11
          8.00 MPI Rank 1
        - nodecard 0
          8.01 MPI Rank 2
        - nodecard 7
          8.01 MPI Rank 3
```

Visited (4 elements)

```
0.00    1013.08 (100.00%)    1013.08
0.00    32.02 (4.58%)    699.04
0.00    32.02
```

Each process spent 8 seconds in boundary exchange

Case II: Time spent in Boundary Exchange



```
1456 Visits (occ)
1013.08 Time (sec)
0.00 Minimum Inclusive Time (sec)
88.24 Maximum Inclusive Time (sec)
1.00 max active tasks
1.89e10 bytes_sent
1.89e10 bytes_received
```

```
699.04 main
692.16 loop
692.15 !$omp parallel @heat-MPI.c:522
628.09 !$omp task @heat-MPI.c:527
346.08 !$omp implicit barrier @heat-MPI.c:5
346.07 !$omp master @heat-MPI.c:523
314.05 !$omp taskwait @heat-MPI.c:530
314.05 task_root
314.04 heatTimestep
32.02 heatBoundary
16.13 MPI_Waitall
15.87 MPI_Isend
3.18 MPI_Barrier
1.84 heatMPIGather
1.22 MPI_Send
1.01 heatMPISetup
0.76 heatTotalEnergy
0.64 heatAllocate
0.61 MPI_Recv
0.36 heatInitialize
0.08 MPI_Init
0.01 MPI_Irecv
0.01 MPI_Cart_create
0.00 heatMPIFree
0.00 MPI_Comm_free
0.00 !$omp create task @heat-MPI.c:527
0.00 heatSwap
0.00 MPI_Finalize
0.00 heatDeallocate
0.00 MPI_Cart_shift
0.00 MPI_Dims_create
0.00 MPI_Comm_size
0.00 MPI_Cart_coords
0.00 MPI_Comm_rank
```

```
- machine juqueen.fz-juelich.de
  - rack 63
    - midplane 0
      - nodeboard 2
        - nodecard 12
          78.51 MPI Rank 0
        - nodecard 11
          78.51 MPI Rank 1
        - nodecard 0
          78.51 MPI Rank 2
        - nodecard 7
          78.51 MPI Rank 3
```

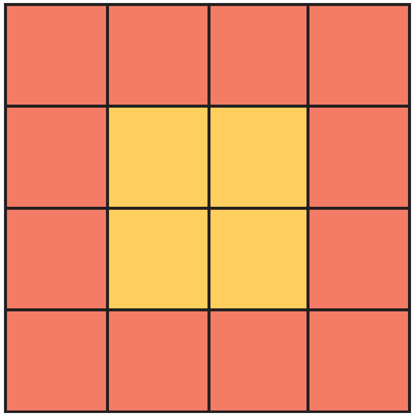
```
0.00        1013.08 (100.00%)        1013.08
0.00        314.04 (44.93%)          699.04
0.00                                 314.04
```

… that’s ~10% of the computation

Case II: Hide MPI communication with computation

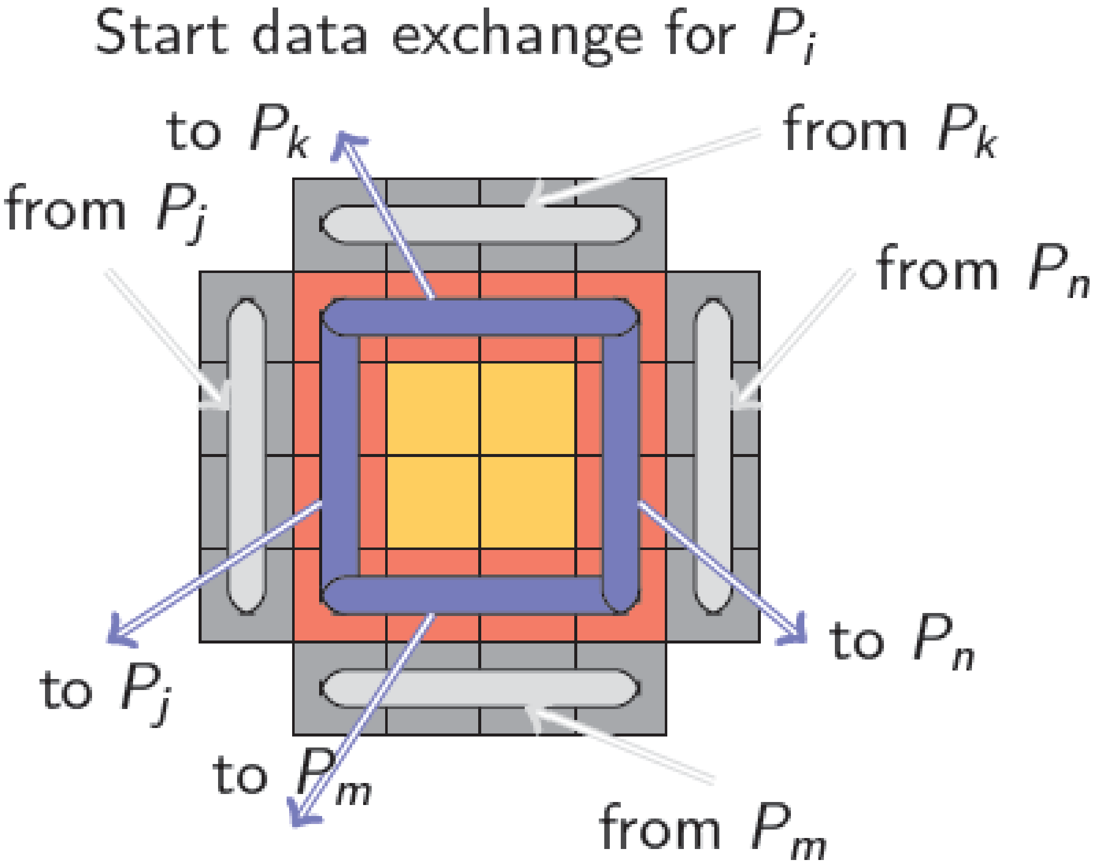
- Step 1: Compute heat in the area which is communicated to your neighbors

Compute heat conduction
in the boundaries of P_i



Case II: Hide MPI communication with computation

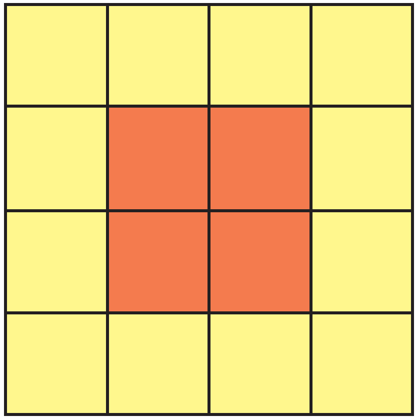
- Step 2: Start communicating boundaries with your neighbors



Case II: Hide MPI communication with computation

- Step 3: Compute heat in the interior area

Compute heat conduction
in the interior of P_i



Case II: Profile Analysis

- Improved MPI algorithm

```
foreach step in [1:nsteps]
  computeHeatConductionInBoundaries
  startBoundaryExchange
  computeHeatConductionInInterior
  waitForCompletionOfBoundaryExchange
```

- Note: As not all MPI implementations support overlapping, it is here done with the help of OpenMP tasks.
- Measuring the improved heat conduction application

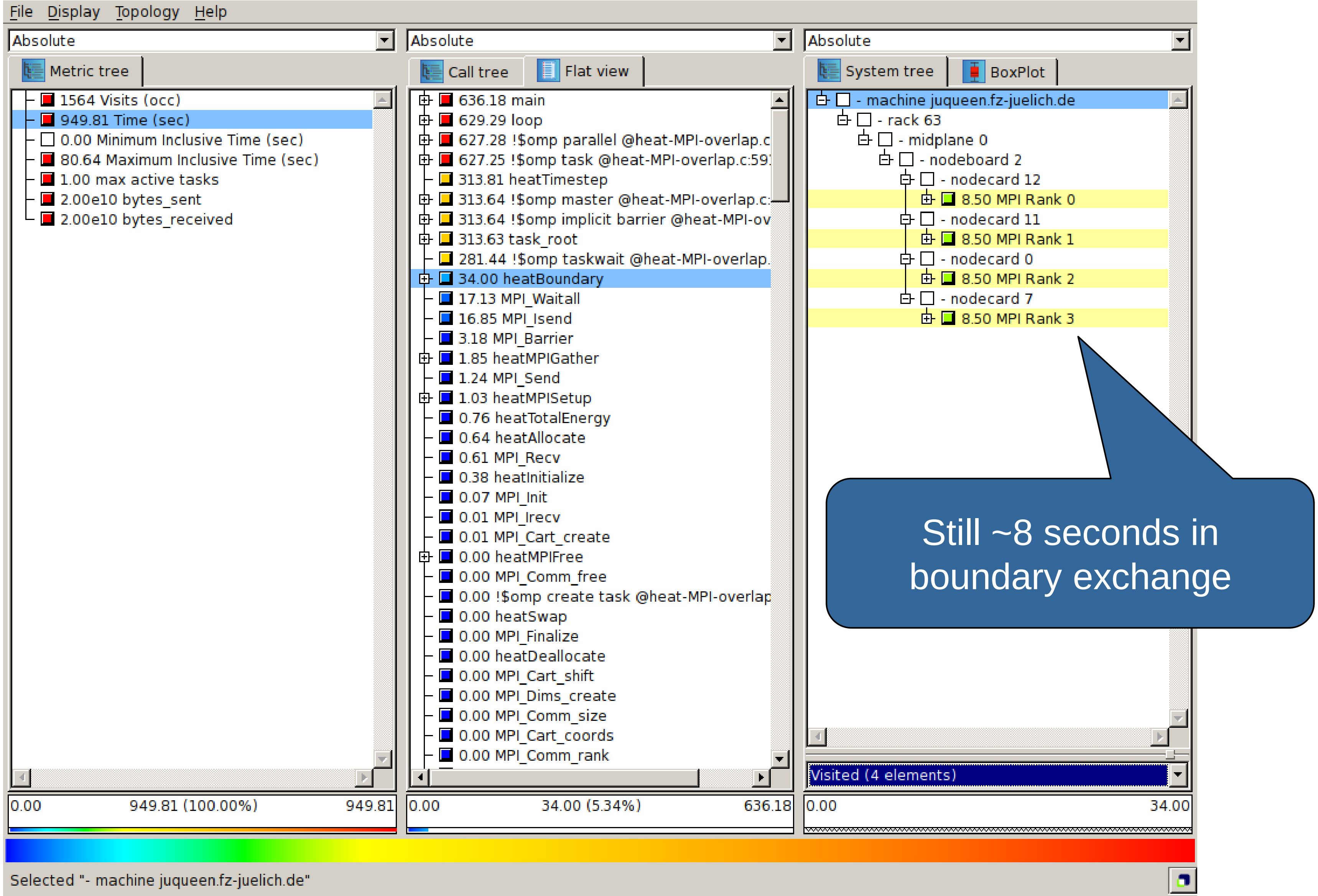
```
% scan mpirun -np 16 ./heat-MPI-overlap 3072 32
```

- Open prepared measurement on the LiveDVD with Cube

```
% cube ~/Bottlenecks/heat/scorep_heat-MPI-overlap_small/profile.cubex
```

[CUBE GUI showing trace analysis report]

Case II: Time spent in Boundary Exchange



```
1564 Visits (occ)
949.81 Time (sec)
0.00 Minimum Inclusive Time (sec)
80.64 Maximum Inclusive Time (sec)
1.00 max active tasks
2.00e10 bytes_sent
2.00e10 bytes_received
```

```
636.18 main
629.29 loop
627.28 !$omp parallel @heat-MPI-overlap.c
627.25 !$omp task @heat-MPI-overlap.c:59
313.81 heatTimestep
313.64 !$omp master @heat-MPI-overlap.c:
313.64 !$omp implicit barrier @heat-MPI-ov
313.63 task_root
281.44 !$omp taskwait @heat-MPI-overlap.
34.00 heatBoundary
17.13 MPI_Waitall
16.85 MPI_Isend
3.18 MPI_Barrier
1.85 heatMPIGather
1.24 MPI_Send
1.03 heatMPISetup
0.76 heatTotalEnergy
0.64 heatAllocate
0.61 MPI_Recv
0.38 heatInitialize
0.07 MPI_Init
0.01 MPI_Irecv
0.01 MPI_Cart_create
0.00 heatMPIFree
0.00 MPI_Comm_free
0.00 !$omp create task @heat-MPI-overlap
0.00 heatSwap
0.00 MPI_Finalize
0.00 heatDeallocate
0.00 MPI_Cart_shift
0.00 MPI_Dims_create
0.00 MPI_Comm_size
0.00 MPI_Cart_coords
0.00 MPI_Comm_rank
```

```
- machine juqueen.fz-juelich.de
  - rack 63
    - midplane 0
      - nodeboard 2
        - nodecard 12
          8.50 MPI Rank 0
        - nodecard 11
          8.50 MPI Rank 1
        - nodecard 0
          8.50 MPI Rank 2
        - nodecard 7
          8.50 MPI Rank 3
```

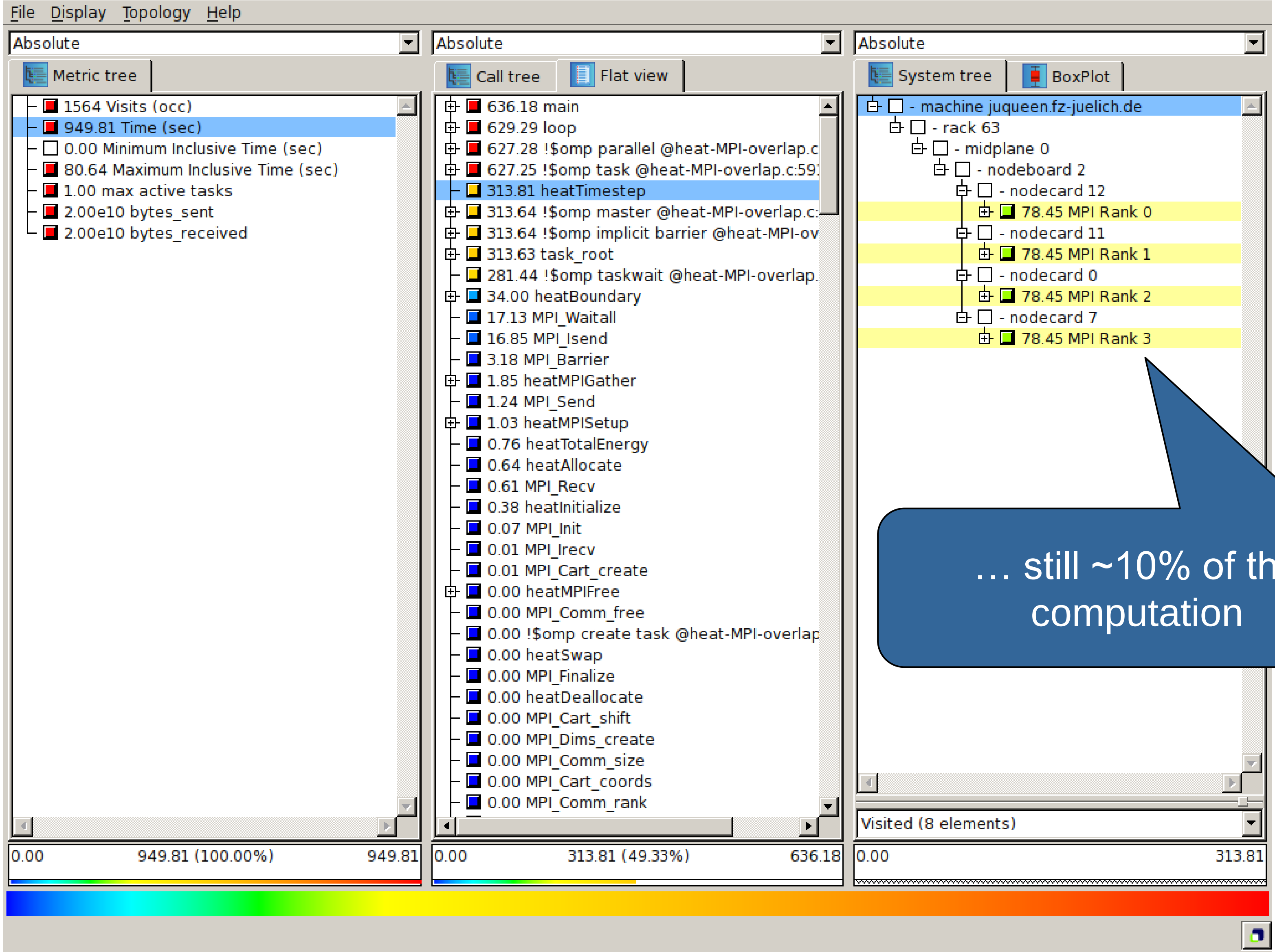
Visited (4 elements)

0.00	949.81 (100.00%)	949.81
0.00	34.00 (5.34%)	636.18
0.00		34.00

Selected "- machine juqueen.fz-juelich.de"

Still ~8 seconds in boundary exchange

Case II: Time spent in Boundary Exchange



Absolute | Absolute | Absolute

```
1564 Visits (occ)
949.81 Time (sec)
0.00 Minimum Inclusive Time (sec)
80.64 Maximum Inclusive Time (sec)
1.00 max active tasks
2.00e10 bytes_sent
2.00e10 bytes_received
```

```
636.18 main
629.29 loop
627.28 !$omp parallel @heat-MPI-overlap.c
627.25 !$omp task @heat-MPI-overlap.c:59
313.81 heatTimestep
313.64 !$omp master @heat-MPI-overlap.c:
313.64 !$omp implicit barrier @heat-MPI-ov
313.63 task_root
281.44 !$omp taskwait @heat-MPI-overlap.
34.00 heatBoundary
17.13 MPI_Waitall
16.85 MPI_Isend
3.18 MPI_Barrier
1.85 heatMPIGather
1.24 MPI_Send
1.03 heatMPISetup
0.76 heatTotalEnergy
0.64 heatAllocate
0.61 MPI_Recv
0.38 heatInitialize
0.07 MPI_Init
0.01 MPI_Irecv
0.01 MPI_Cart_create
0.00 heatMPIFree
0.00 MPI_Comm_free
0.00 !$omp create task @heat-MPI-overlap
0.00 heatSwap
0.00 MPI_Finalize
0.00 heatDeallocate
0.00 MPI_Cart_shift
0.00 MPI_Dims_create
0.00 MPI_Comm_size
0.00 MPI_Cart_coords
0.00 MPI_Comm_rank
```

```
- machine juqueen.fz-juelich.de
  - rack 63
    - midplane 0
      - nodeboard 2
        - nodecard 12
          78.45 MPI Rank 0
        - nodecard 11
          78.45 MPI Rank 1
        - nodecard 0
          78.45 MPI Rank 2
        - nodecard 7
          78.45 MPI Rank 3
```

Visited (8 elements)

```
0.00        949.81 (100.00%)        949.81
0.00        313.81 (49.33%)         636.18
0.00                                313.81
```

… still ~10% of the computation

Case II: Profile Comparison

- Calculate differences between profiles

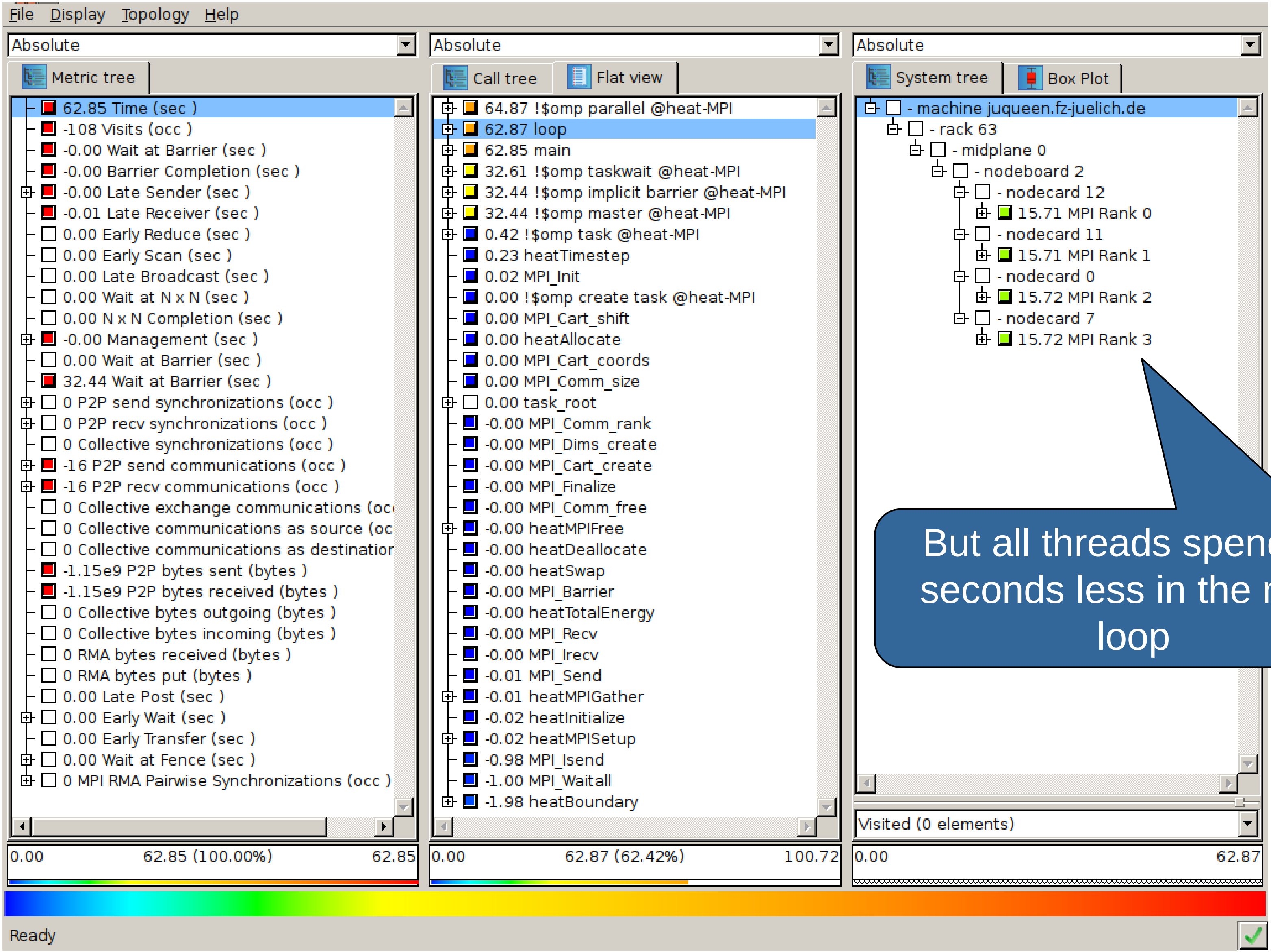
```
% cube_diff ~/Bottlenecks/heat/scorep_heat-MPI_small/profile.cubex \
            ~/Bottlenecks/heat/scorep_heat-MPI-overlap_small/profile.cubex
```

- Open prepared profile diff on the LiveDVD with Cube

```
% cube ~/Bottlenecks/heat/diff.cubex
```

[CUBE GUI showing trace analysis report]

Case II: Profile Comparison



File Display Topology Help

Absolute

Metric tree

- 62.85 Time (sec)
- -108 Visits (occ)
- -0.00 Wait at Barrier (sec)
- -0.00 Barrier Completion (sec)
- -0.00 Late Sender (sec)
- -0.01 Late Receiver (sec)
- 0.00 Early Reduce (sec)
- 0.00 Early Scan (sec)
- 0.00 Late Broadcast (sec)
- 0.00 Wait at N x N (sec)
- 0.00 N x N Completion (sec)
- -0.00 Management (sec)
- 0.00 Wait at Barrier (sec)
- 32.44 Wait at Barrier (sec)
- 0 P2P send synchronizations (occ)
- 0 P2P recv synchronizations (occ)
- 0 Collective synchronizations (occ)
- -16 P2P send communications (occ)
- -16 P2P recv communications (occ)
- 0 Collective exchange communications (oc
- 0 Collective communications as source (oc
- 0 Collective communications as destinatior
- -1.15e9 P2P bytes sent (bytes)
- -1.15e9 P2P bytes received (bytes)
- 0 Collective bytes outgoing (bytes)
- 0 Collective bytes incoming (bytes)
- 0 RMA bytes received (bytes)
- 0 RMA bytes put (bytes)
- 0.00 Late Post (sec)
- 0.00 Early Wait (sec)
- 0.00 Early Transfer (sec)
- 0.00 Wait at Fence (sec)
- 0 MPI RMA Pairwise Synchronizations (occ)

0.00 62.85 (100.00%) 62.85

Absolute

Call tree Flat view

- 64.87 !$omp parallel @heat-MPI
- 62.87 loop
- 62.85 main
- 32.61 !$omp taskwait @heat-MPI
- 32.44 !$omp implicit barrier @heat-MPI
- 32.44 !$omp master @heat-MPI
- 0.42 !$omp task @heat-MPI
- 0.23 heatTimestep
- 0.02 MPI_Init
- 0.00 !$omp create task @heat-MPI
- 0.00 MPI_Cart_shift
- 0.00 heatAllocate
- 0.00 MPI_Cart_coords
- 0.00 MPI_Comm_size
- 0.00 task_root
- -0.00 MPI_Comm_rank
- -0.00 MPI_Dims_create
- -0.00 MPI_Cart_create
- -0.00 MPI_Finalize
- -0.00 MPI_Comm_free
- -0.00 heatMPIFree
- -0.00 heatDeallocate
- -0.00 heatSwap
- -0.00 MPI_Barrier
- -0.00 heatTotalEnergy
- -0.00 MPI_Recv
- -0.00 MPI_Irecv
- -0.01 MPI_Send
- -0.01 heatMPIGather
- -0.02 heatInitialize
- -0.02 heatMPISetup
- -0.98 MPI_Isend
- -1.00 MPI_Waitall
- -1.98 heatBoundary

0.00 62.87 (62.42%) 100.72

Absolute

System tree Box Plot

- machine juqueen.fz-juelich.de
  - rack 63
    - midplane 0
      - nodeboard 2
        - nodecard 12
          - 15.71 MPI Rank 0
        - nodecard 11
          - 15.71 MPI Rank 1
        - nodecard 0
          - 15.72 MPI Rank 2
        - nodecard 7
          - 15.72 MPI Rank 3

Visited (0 elements)

0.00 62.87

Ready

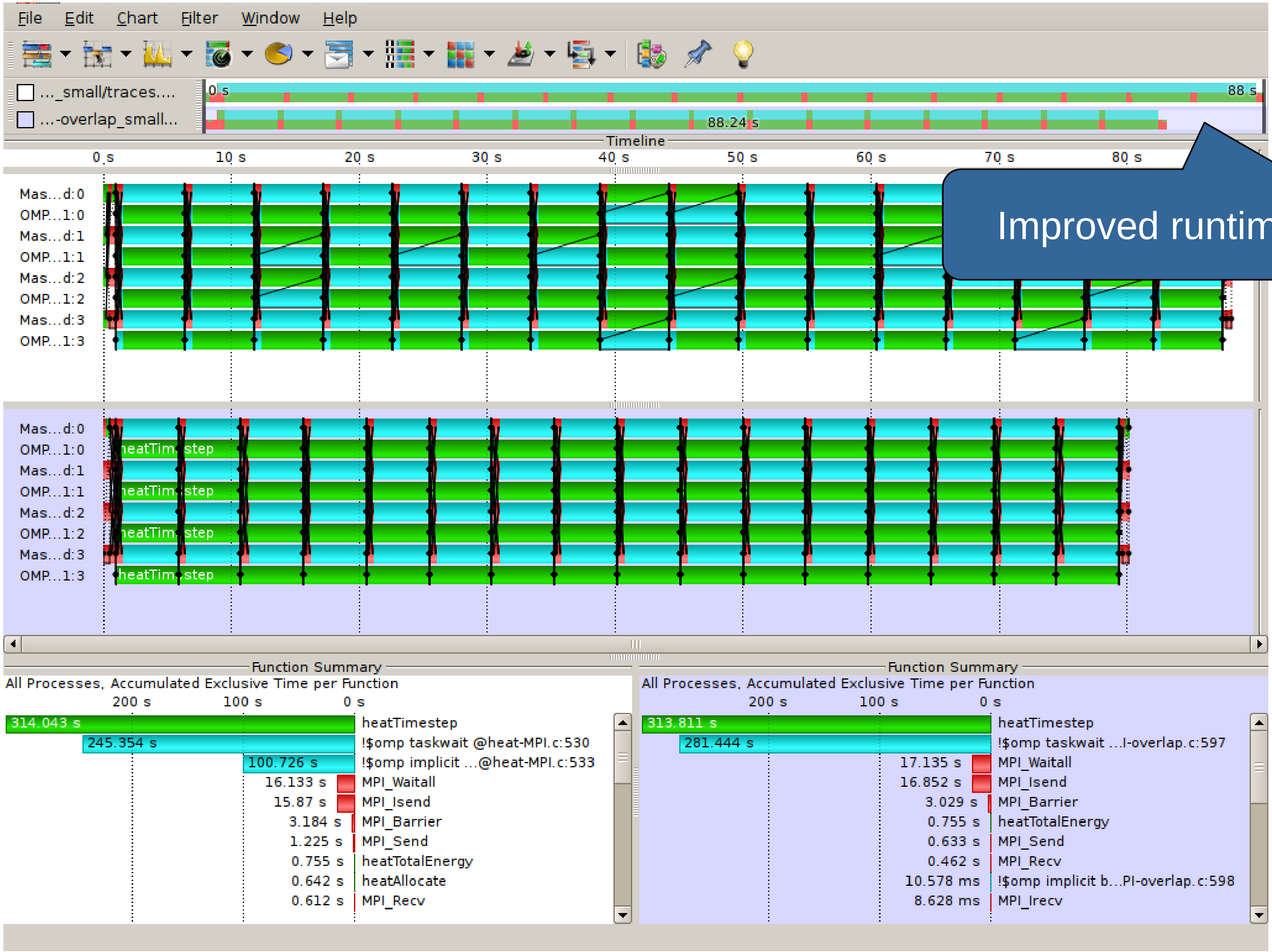
But all threads spend ~8 seconds less in the main loop

Case II: Trace Comparison

- Open prepared measurement on the LiveDVD with Vampir

```
% vampir ~/Bottlenecks/heat/scorep_heat-MPI_small/traces.otf2 \
         ~/Bottlenecks/heat/scorep_heat-MPI-overlap_small/traces.otf2

                       [Vampir GUI showing trace]
```

Case II: Trace Comparison



Improved runtime

Function Summary

All Processes, Accumulated Exclusive Time per Function

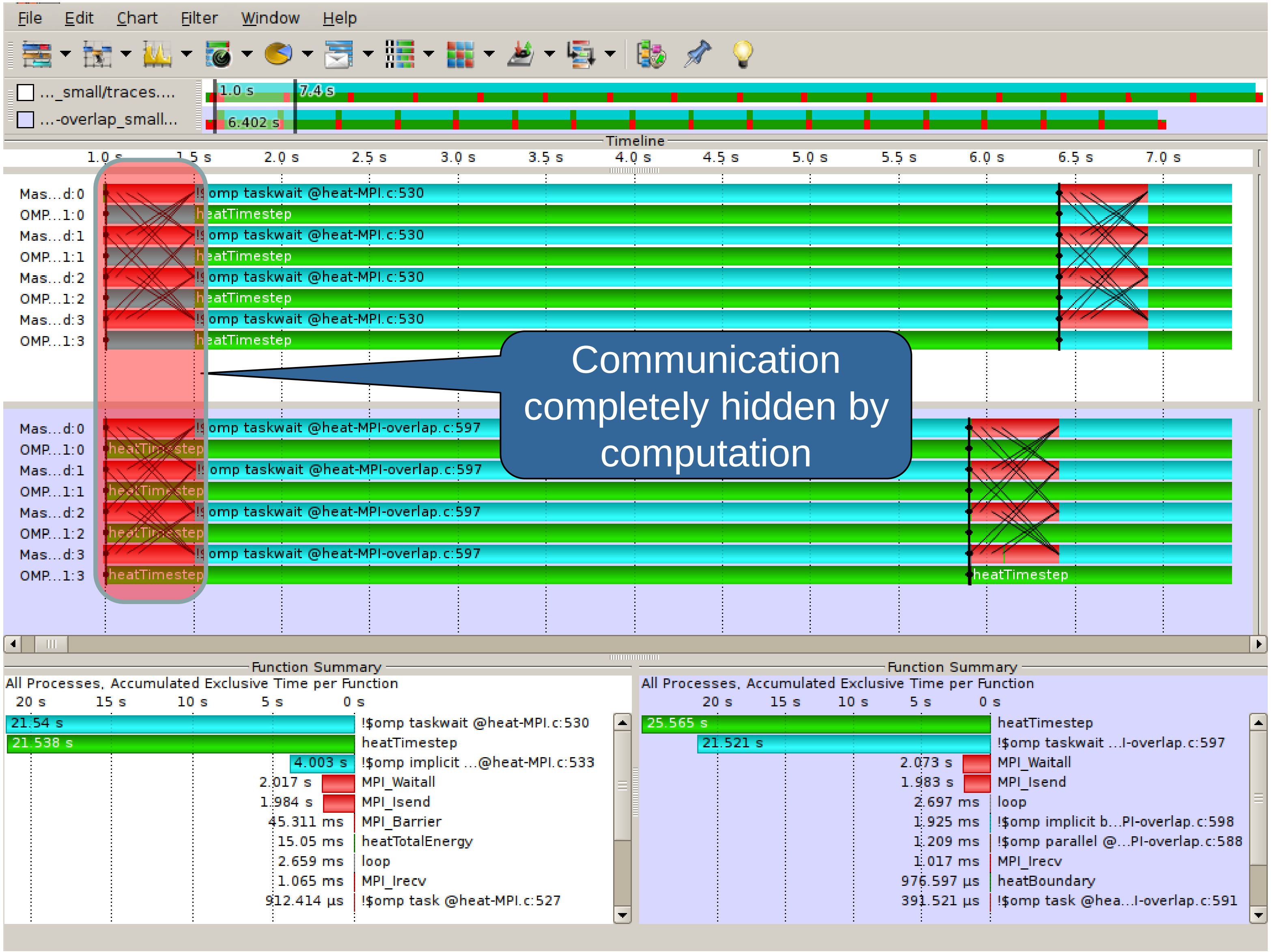
314.043 s	heatTimestep
245.354 s	!$omp taskwait @heat-MPI.c:530
100.726 s	!$omp implicit …@heat-MPI.c:533
16.133 s	MPI_Waitall
15.87 s	MPI_Isend
3.184 s	MPI_Barrier
1.225 s	MPI_Send
0.755 s	heatTotalEnergy
0.642 s	heatAllocate
0.612 s	MPI_Recv

Function Summary

All Processes, Accumulated Exclusive Time per Function

313.811 s	heatTimestep
281.444 s	!$omp taskwait …l-overlap.c:597
17.135 s	MPI_Waitall
16.852 s	MPI_Isend
3.029 s	MPI_Barrier
0.755 s	heatTotalEnergy
0.633 s	MPI_Send
0.462 s	MPI_Recv
10.578 ms	!$omp implicit b…PI-overlap.c:598
8.628 ms	MPI_Irecv

Case II: Trace Comparison



Communication completely hidden by computation

Function Summary

All Processes, Accumulated Exclusive Time per Function

20 s 15 s 10 s 5 s 0 s

21.54 s !$omp taskwait @heat-MPI.c:530
21.538 s heatTimestep
4.003 s !$omp implicit …@heat-MPI.c:533
2.017 s MPI_Waitall
1.984 s MPI_Isend
45.311 ms MPI_Barrier
15.05 ms heatTotalEnergy
2.659 ms loop
1.065 ms MPI_Irecv
912.414 µs !$omp task @heat-MPI.c:527

Function Summary

All Processes, Accumulated Exclusive Time per Function

20 s 15 s 10 s 5 s 0 s

25.565 s heatTimestep
21.521 s !$omp taskwait …I-overlap.c:597
2.073 s MPI_Waitall
1.983 s MPI_Isend
2.697 ms loop
1.925 ms !$omp implicit b…PI-overlap.c:598
1.209 ms !$omp parallel @…PI-overlap.c:588
1.017 ms MPI_Irecv
976.597 µs heatBoundary
391.521 µs !$omp task @hea…I-overlap.c:591

Acknowledgments

- Thanks to Dirk Schmidl, RWTH Aachen, for providing the sparse matrix vector multiplication code

Review

Brian Wylie
Jülich Supercomputing Centre

Summary

You’ve been introduced to a variety of tools

- with hints to apply and use the tools effectively

- Tools provide complementary capabilities
  - computational kernel & processor analyses
  - communication/synchronization analyses
  - load-balance, scheduling, scaling, …
- Tools are designed with various trade-offs
  - general-purpose versus specialized
  - platform-specific versus agnostic
  - simple/basic versus complex/powerful

Tool selection

- Which tools you use and when you use them likely to depend on situation
  - which are available on (or for) your computer system
  - which support your programming paradigms and languages
  - which you are familiar (comfortable) with using
  - which type of issue you suspect
  - which question you want to have answered
- Being aware of (potentially) available tools and their capabilities can help finding the most appropriate tools

Workflow (getting started)

- First ensure that the parallel application runs correctly
  - no-one will care how quickly you can get invalid answers or produce a directory full of corefiles
  - parallel debuggers help isolate known problems
  - correctness checking tools can help identify other issues
  - (that might not cause problems right now, but will eventually)
    - e.g., race conditions, invalid/non-compliant usage
- Generally valuable to start with an overview of execution performance
  - fraction of time spent in computation vs comm/synch vs I/O
  - which sections of the application/library code are most costly
- and how it changes with scale or different configurations
  - processes vs threads, mappings, bindings

Workflow (communication/synchronization)

- Communication/synchronization issues generally apply to every computer system (to different extents) and typically grow with the number of processes/threads
  - *Weak scaling*: fixed computation per thread, and perhaps fixed localities, but increasingly distributed
  - *Strong scaling*: constant total computation, increasingly divided amongst threads, while communication grows
  - Collective communication (particularly of type “all-to-all”) result in increasing data movement
  - Synchronizations of larger groups are increasingly costly
  - Load-balancing becomes increasingly challenging, and imbalances increasingly expensive
    - generally manifests as waiting time at following collective ops

Workflow (wasted waiting time)

- Waiting times are difficult to determine in basic profiles
  - Part of the time each process/thread spends in communication & synchronization operations may be wasted waiting time
  - Need to correlate event times between processes/threads
    - *Periscope* uses augmented messages to transfer timestamps and additional on-line analysis processes
    - Post-mortem event trace analysis avoids interference and provides a complete history
    - *Scalasca* automates trace analysis and ensures waiting times are completely quantified
    - *Vampir* allows interactive exploration and detailed examination of reasons for inefficiencies

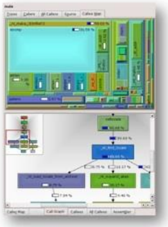
Workflow (core computation)

Effective computation within processors/cores is also vital

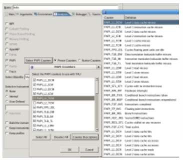
- Optimized libraries may already be available
- Optimizing compilers can also do a lot
  - provided the code is clearly written and not too complex
  - appropriate directives and other hints can also help
- Processor hardware counters can also provide insight
  - although hardware-specific interpretation required
- Tools available from processor and system vendors help navigate and interpret processor-specific performance issues

Technologies and their integration

KCACHEGRIND

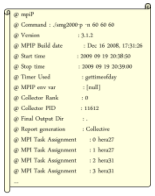


PAPI



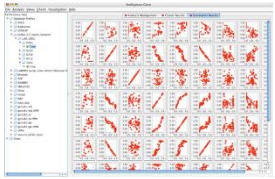
Hardware monitoring

LWM2 / MAP / MPIP / O|SS / MAQAO



```
@ mpiP
@ Command : ./smg2000-p -n 60 60 60
@ Version                  : 3.1.2
@ MPIP Build date          : Dec 16 2008, 17:31:26
@ Start time               : 2009 09 19 20:38:50
@ Stop time                : 2009 09 19 20:39:00
@ Timer Used               : gettimeofday
@ MPIP env var             : [null]
@ Collector Rank           : 0
@ Collector PID            : 11612
@ Final Output Dir         : .
@ Report generation        : Collective
@ MPI Task Assignment      : 0 hera27
@ MPI Task Assignment      : 1 hera27
@ MPI Task Assignment      : 2 hera31
@ MPI Task Assignment      : 3 hera31
...
```

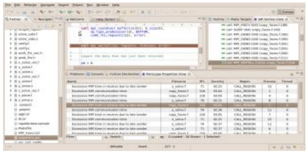
TAU



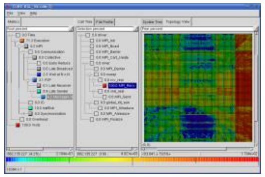
SCORE-P

Automatic profile & trace analysis

PERISCOPE

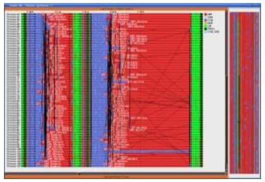


SCALASCA



Visual trace analysis

VAMPIR / PARAVER



Optimization

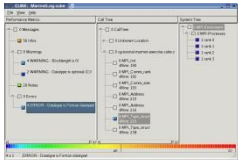
RUBIK / MAQAO

Execution

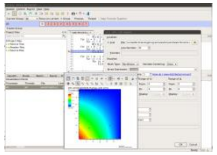
SYSMON / SPINDLE / SIONLIB / OPENMPI

Debugging, error & anomaly detection

MUST



DDT



STAT

Featured VI-HPS tools

- **Score-P**
  - community-developed instrumenter & measurement libraries for parallel profiling and event tracing
- **CUBE & ParaProf/PerfExplorer**
  - interactive parallel profile analyses
- **Scalasca**
  - automated event-trace analysis
- **Vampir**
  - interactive event-trace visualizations and analyses
- **TAU/PDT**
  - comprehensive performance system

Further information

- Website
  - Introductory information about the VI-HPS portfolio of tools for high-productivity parallel application development
    - VI-HPS Tools Guide
    - links to individual tools sites for details and download
  - Training material
    - tutorial slides
    - latest ISO image of VI-HPS Linux DVD with productivity tools
    - user guides and reference manuals for tools
  - News of upcoming events
    - tutorials and workshops
    - mailing-list sign-up for announcements

http://www.vi-hps.org