



Scalable performance analysis of large-scale parallel applications on Cray XT systems with Scalasca

2010-05-25 | **Brian J. N. Wylie**, David Böhme, Wolfgang Frings, Markus Geimer, Bernd Mohr, Zoltán Szebenyi
Jülich Supercomputing Centre

Daniel Becker, Marc-André Hermanns, Felix Wolf
German Research School for Simulation Sciences

b.wylie@fz-juelich.de

Overview

Scalasca performance analysis toolset

- Overview of architecture and usage
- Developments for Cray XT systems

Scalasca measurement & analysis case studies

- Sweep3D
- PFLOTRAN
- PEPC
- GemsFDTD
- NPB-MZ-MPI BT

Conclusions

Scalasca project

Overview

- Helmholtz Initiative & Networking Fund project started in 2006
 - *Headed by Prof. Felix Wolf (RWTH Aachen, GRS & JSC)*
- Follow-up to pioneering KOJAK project (started 1998)
 - *Automatic pattern-based trace analysis*

Objective

- Development of a **scalable performance analysis** toolset
- Specifically targeting **large-scale parallel applications**

Status

- Scalasca v1.3 released in March 2010
- Available for download from www.scalasca.org

Scalasca features

Open source, New BSD license

Portable

- IBM BlueGene/L, BlueGene/P, SP & blade clusters, Cray XT4/5, NEC SX, SGI Altix, SiCortex, Linux cluster® (SPARC, x86-64), ...

Supports typical HPC languages & parallel programming paradigms

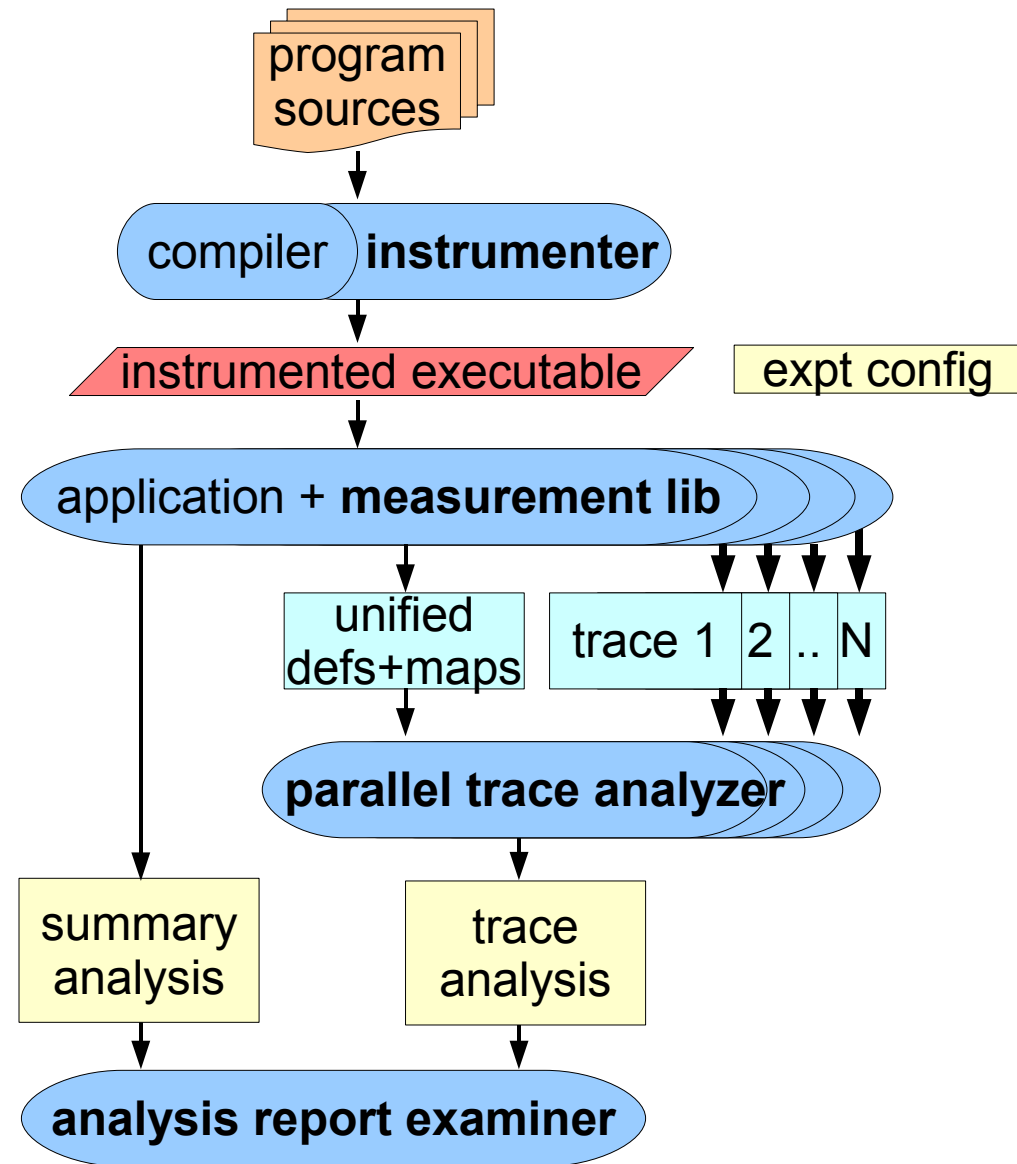
- Fortran, C, C++
- MPI, OpenMP & hybrid MPI/OpenMP

Integrated instrumentation, measurement & analysis toolset

- Customizable automatic/manual instrumentation
- Runtime summarization (*aka* profiling)
- Automatic event trace analysis

Scalasca components

- Automatic program instrumenter creates instrumented executable
- Unified measurement library supports both
 - *runtime summarization*
 - *trace file generation*
- Parallel, replay-based event trace analyzer invoked automatically on set of traces
- Common analysis report explorer & examination/processing tools



Scalasca usage

1. Prepare application objects and executable for measurement:

- `scalasca -instrument ftn -03 -c ...`
- `scalasca -instrument ftn -03 -o sweep3d.exe ...`
 - *instrumented executable sweep3d.exe produced*

2. Run application under control of measurement & analysis nexus:

- `scalasca -analyze aprun -n 196608 sweep3d.exe`
 - *epik_sweep3d_196608_sum* experiment produced
- `scalasca -analyze -t aprun -n 196608 sweep3d.exe`
 - *epik_sweep3d_196608_trace* experiment produced

BATCH JOB

3. Interactively explore measurement analysis report:

- `scalasca -examine epik_sweep3d_196608_trace`
 - *epik_sweep3d_196608_trace/trace.cube.gz* presented

Scalasca functionality in latest releases

Generic improvements

- Hybrid OpenMP/MPI measurement & analysis
- Selective routine instrumentation with PDToolkit instrumenter
- Runtime filtering of routines instrumented by PGI compilers
- Clock synchronization and event timestamp correction
- SIONlib mapping of task-local files into physical files
- Separate CUBE3 GUI package for shared/distributed installations

Enhancements for Cray XT

- Support for multiple programming environments (PrgEnvs)
 - *PGI, GNU, Intel, Pathscale, CCE (partial)*
- Capture of mappings of processes to processors in machine

Measurement & analysis methodology

1. Run uninstrumented/optimized version (as reference for validation)
 - determine memory available for measurement
 2. Run automatically-instrumented version collecting runtime summary
 - determine functions with excessive overheads
 - *examine distortion and trace buffer capacity requirement*
 - if necessary, prepare filter file and repeat measurement
 3. Reconfigure measurement to collect and automatically analyze traces
- (optional) Customize analysis report and/or instrumentation, e.g.,
- extract key code sections into specific analysis reports
 - annotate key code sections with EPIK instrumentation API macros

Sweep3D

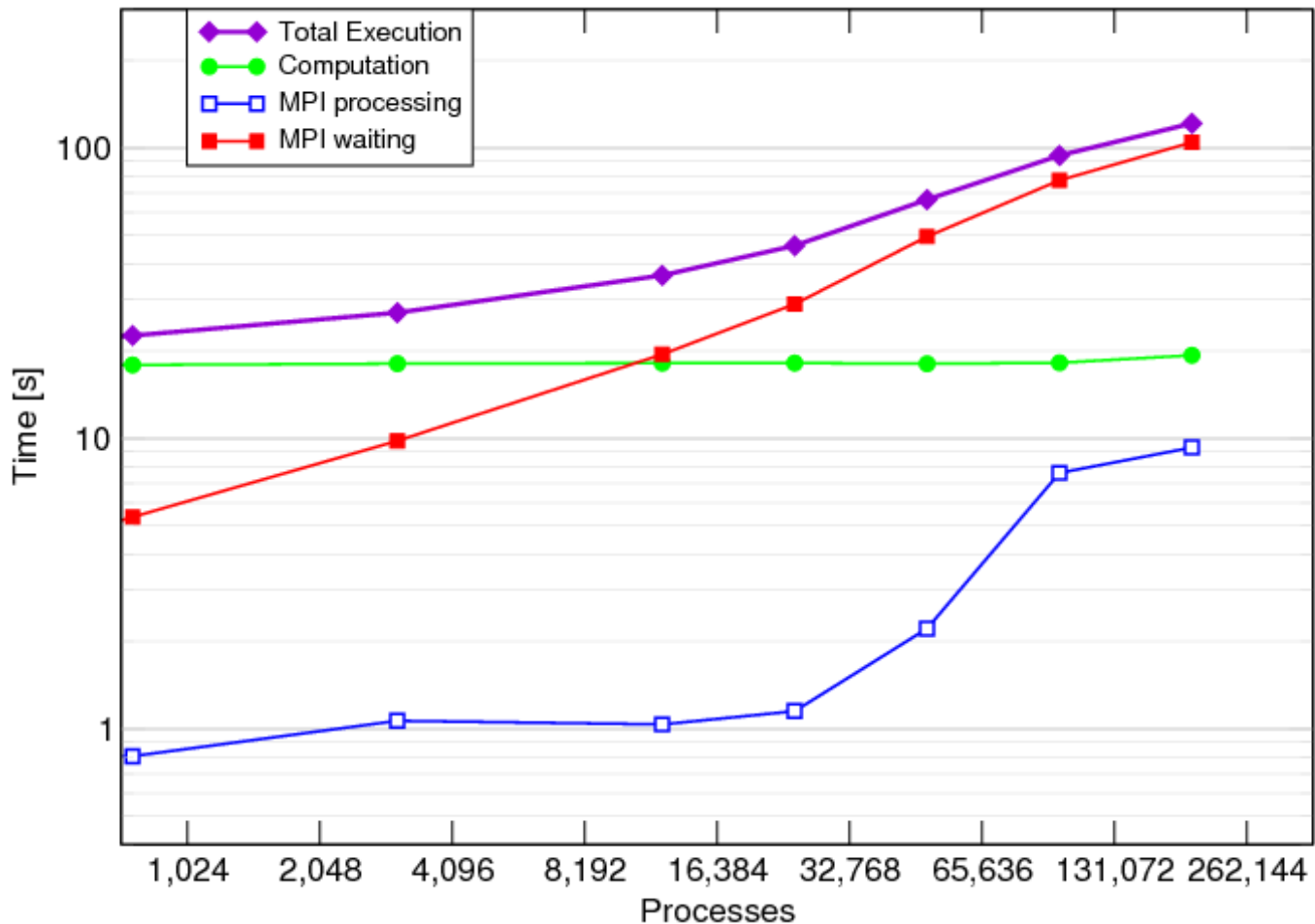
Ubiquitous ASCI benchmark code from Los Alamos National Laboratory

- 3-dimensional neutron transport simulation
- direct order solve uses diagonal wavefront sweeps over grid cells combined with pipelining of blocks of k -planes and octants
- execution performance extensively modeled & analyzed

MPI parallel version using 2D domain decomposition

- ~2,000 lines of code (12 source modules), mostly Fortran77
- very portable, and highly scalable
- tunable via input deck, e.g., number of k -planes in blocks (MK)
- benchmark configuration does 12 iterations
 - *flux correction 'fixups' applied after 7th iteration*

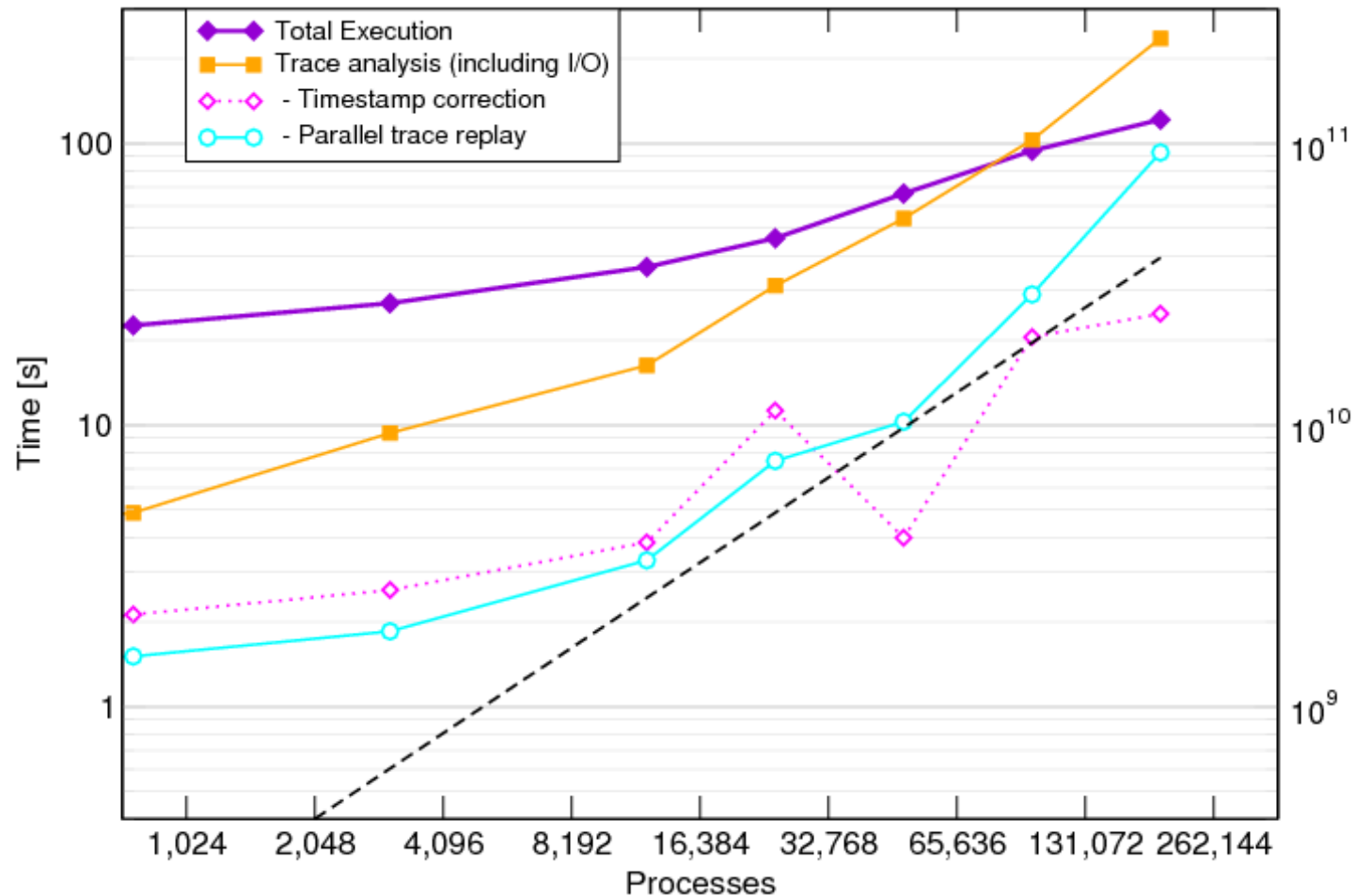
Sweep3D application execution time scaling



- Default input deck settings
- *Weak scaling* due to fixed problem size per process
- Reasonable scalability(?)
- Constant time for *computation*
- Rapid growth in *MPI* time, which is almost all *waiting* time

[Scalasca 1.3 summary & trace experiments on Jaguar XT5:
less than 3% measurement dilation compared to uninstrumented]

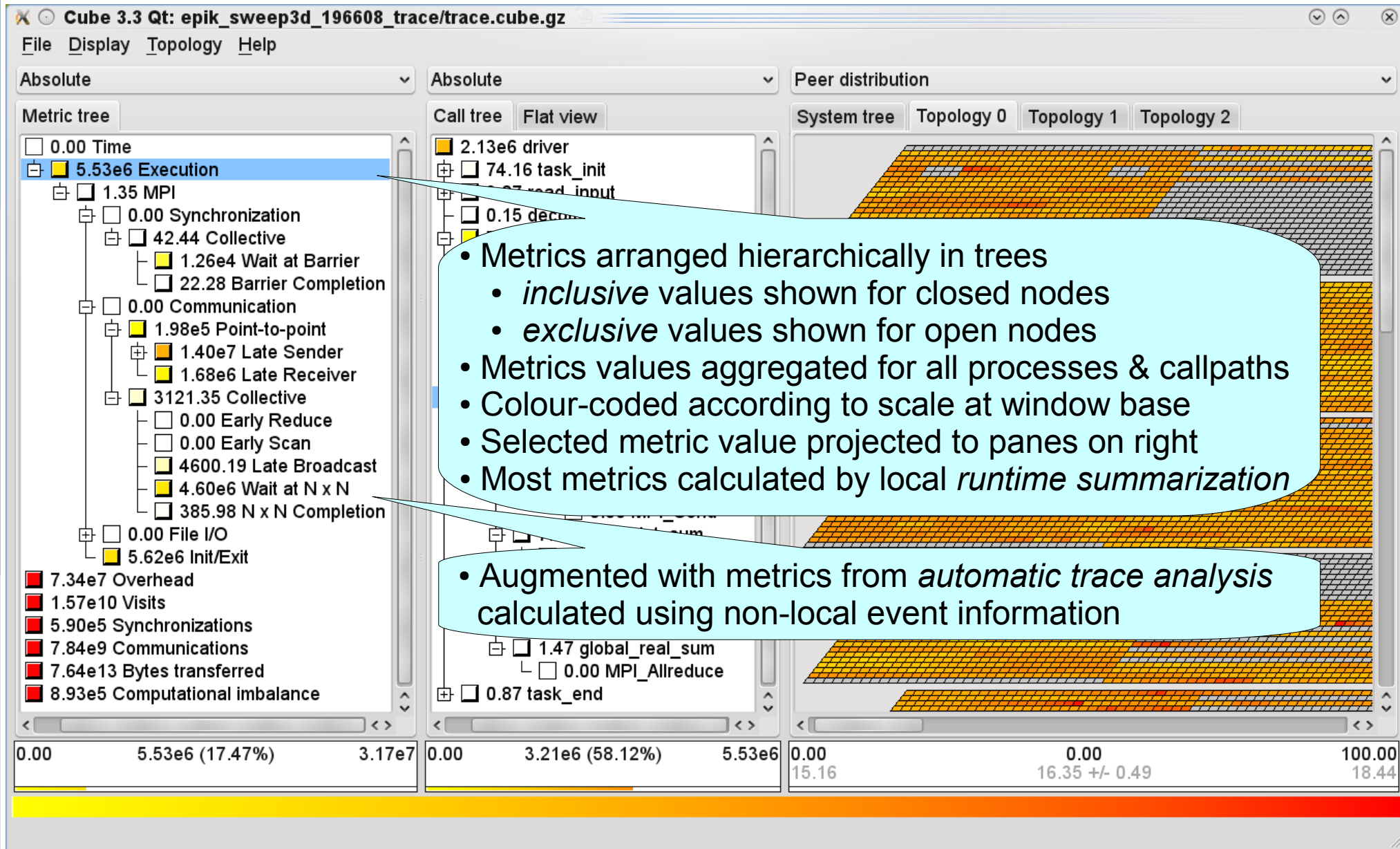
Scalasca Sweep3D trace analysis time scaling



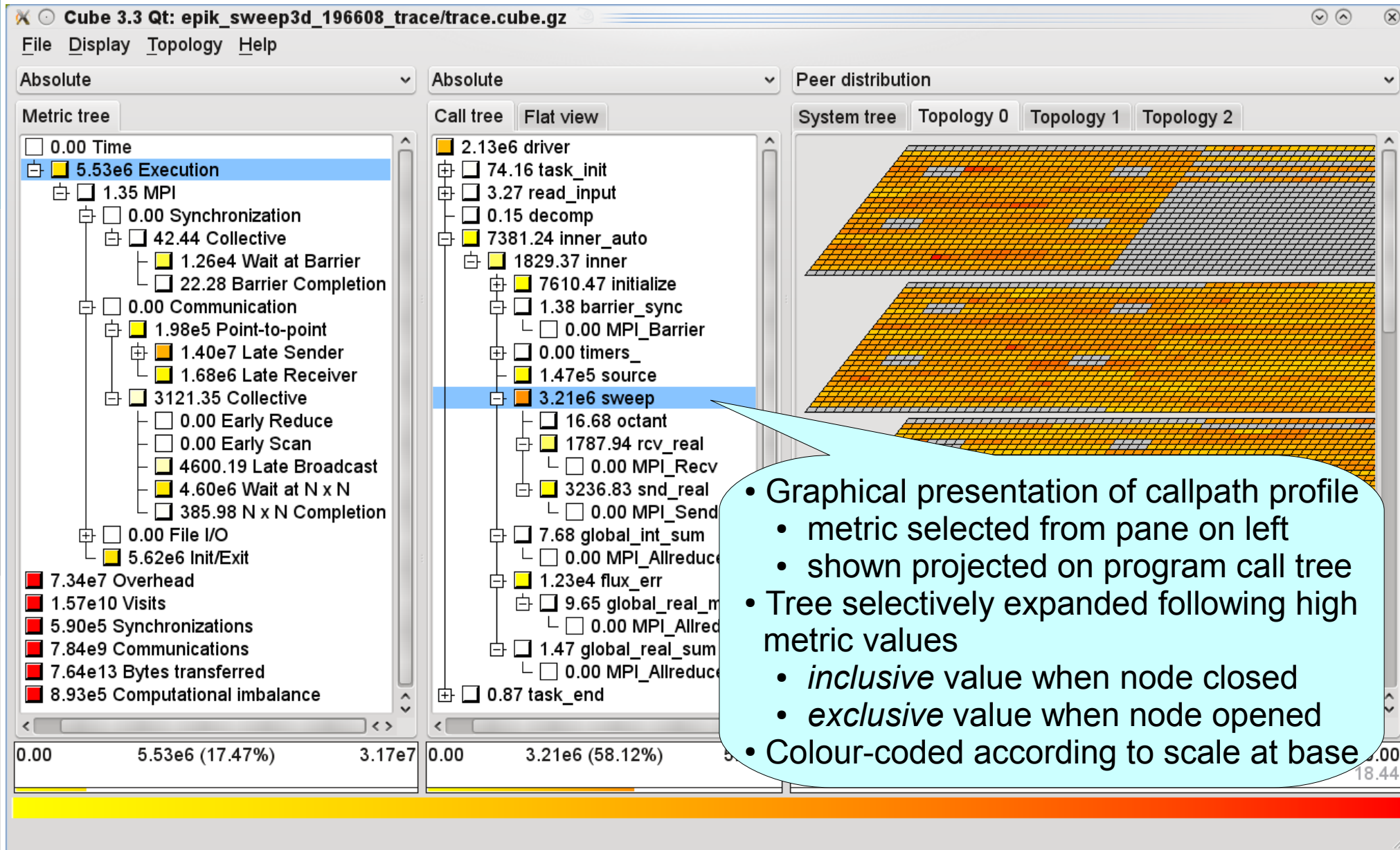
- 2.75MB of trace event records per process
- Total trace size (---) increases linearly to 526GB
- Trace analysis comparable with application execution time
- Timestamp correction time varies according to violations

[Scalasca v1.3 on Jaguar XT5]

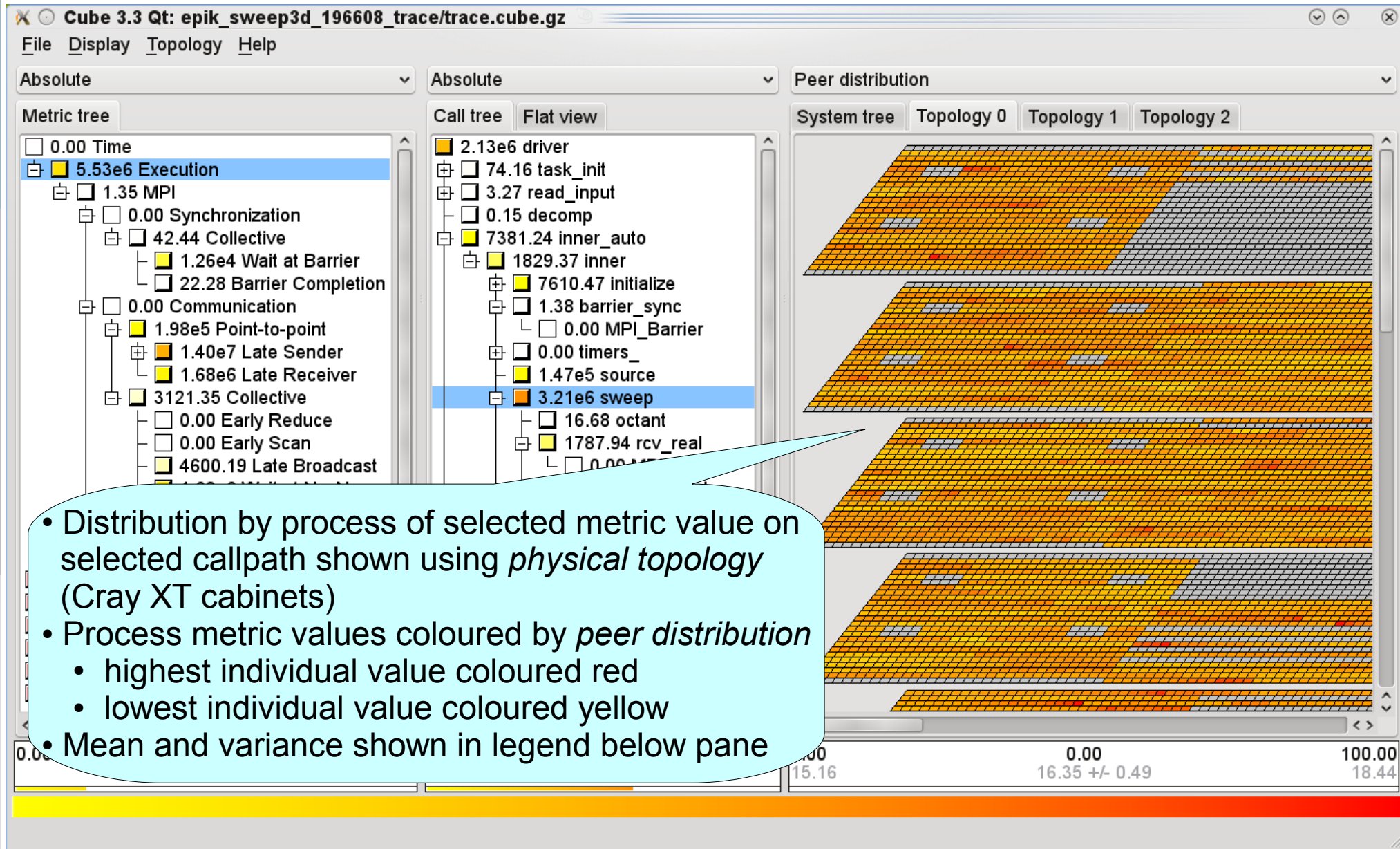
Scalasca analysis report: performance metric tree



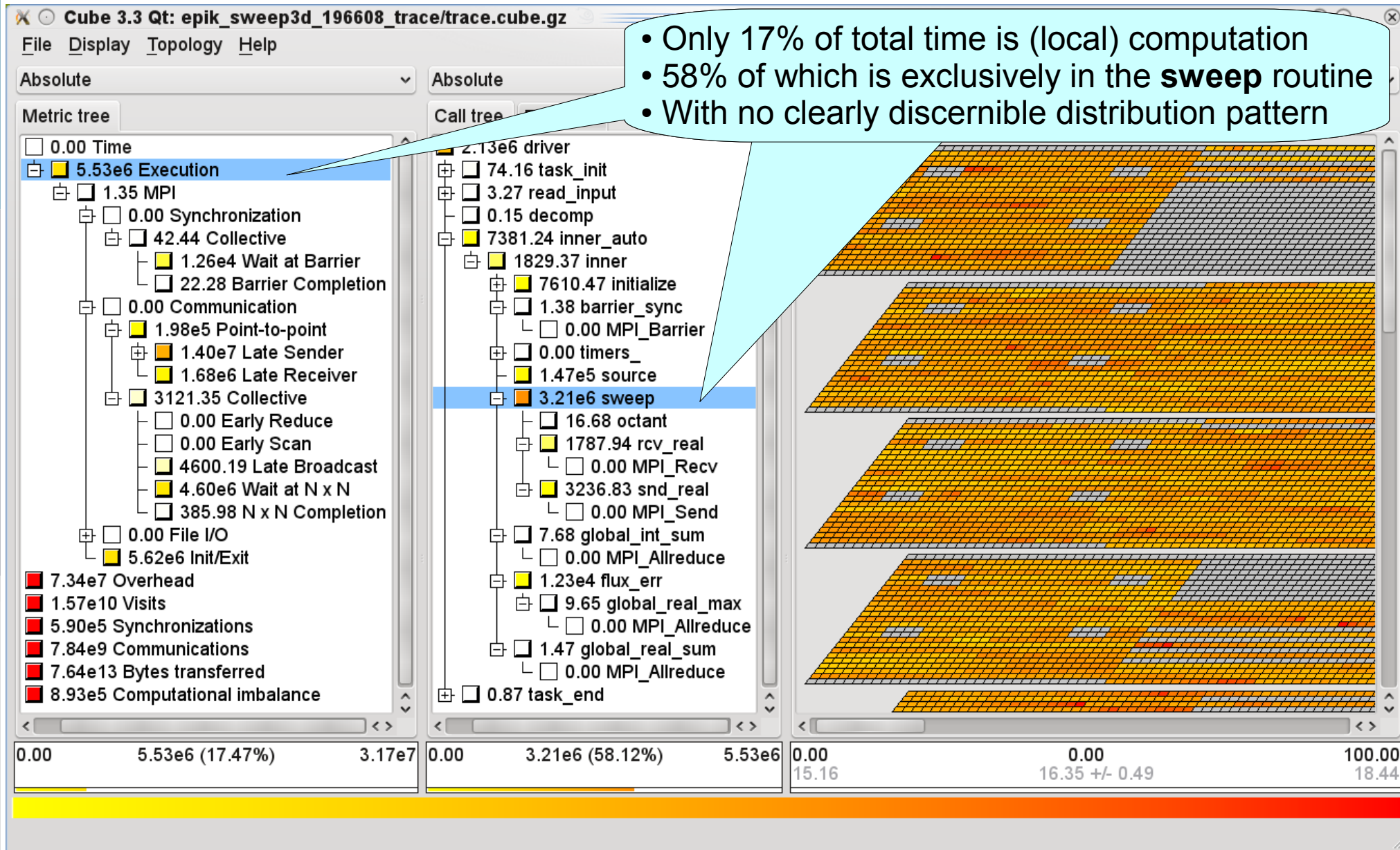
Scalasca analysis report: program call tree



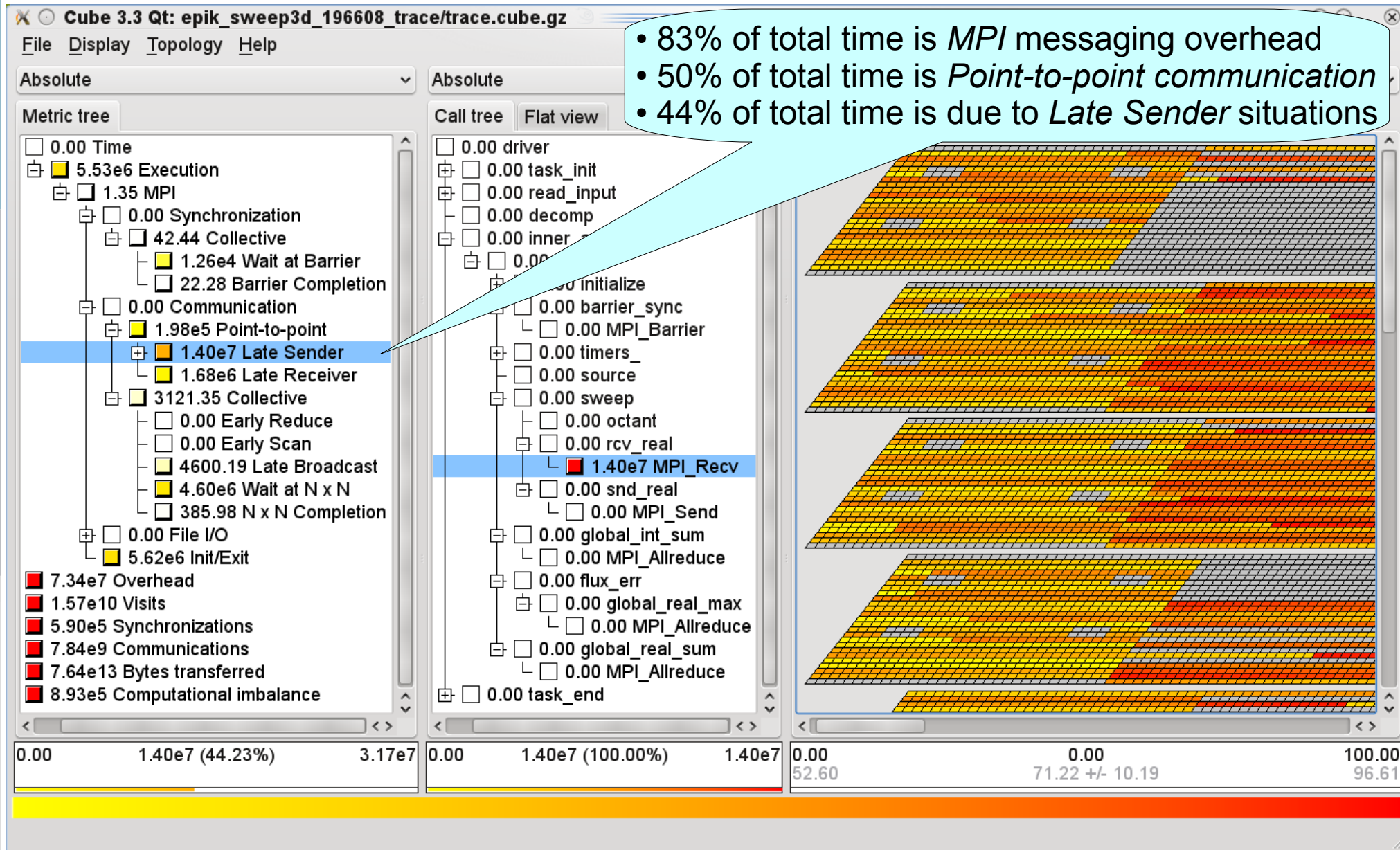
Scalasca analysis report: system tree/topology



Scalasca analysis report: sweep computation time



Scalasca analysis report: sweep *Late Sender* time

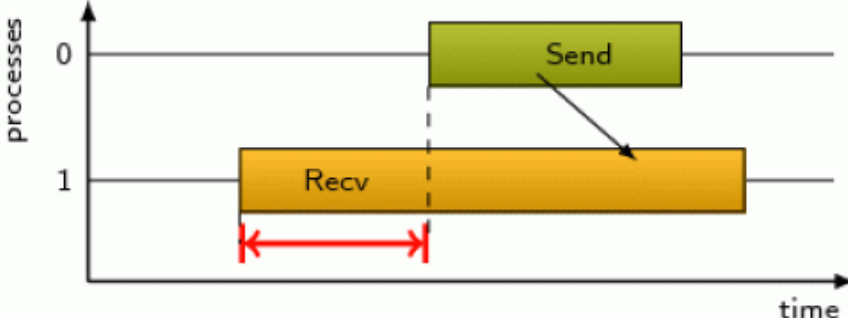


Scalasca description for *Late Sender* metric

Online 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.

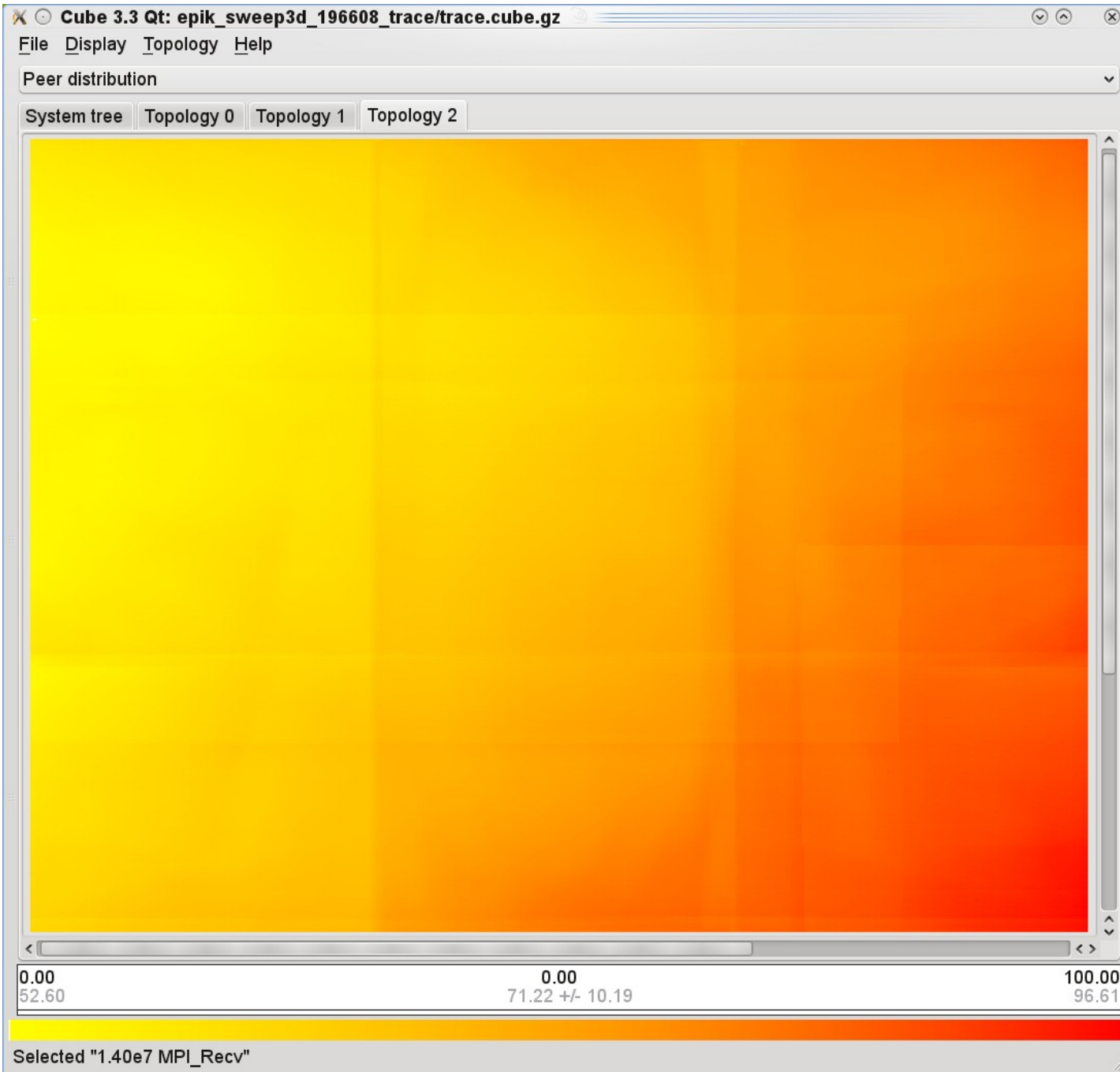


Unit:
 Seconds

Diagnosis:
 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.

- Analysis report explorer GUI provides hyperlinked online descriptions of metrics
- Diagnosis hints suggest how to refine diagnosis of performance problems and possible remediation

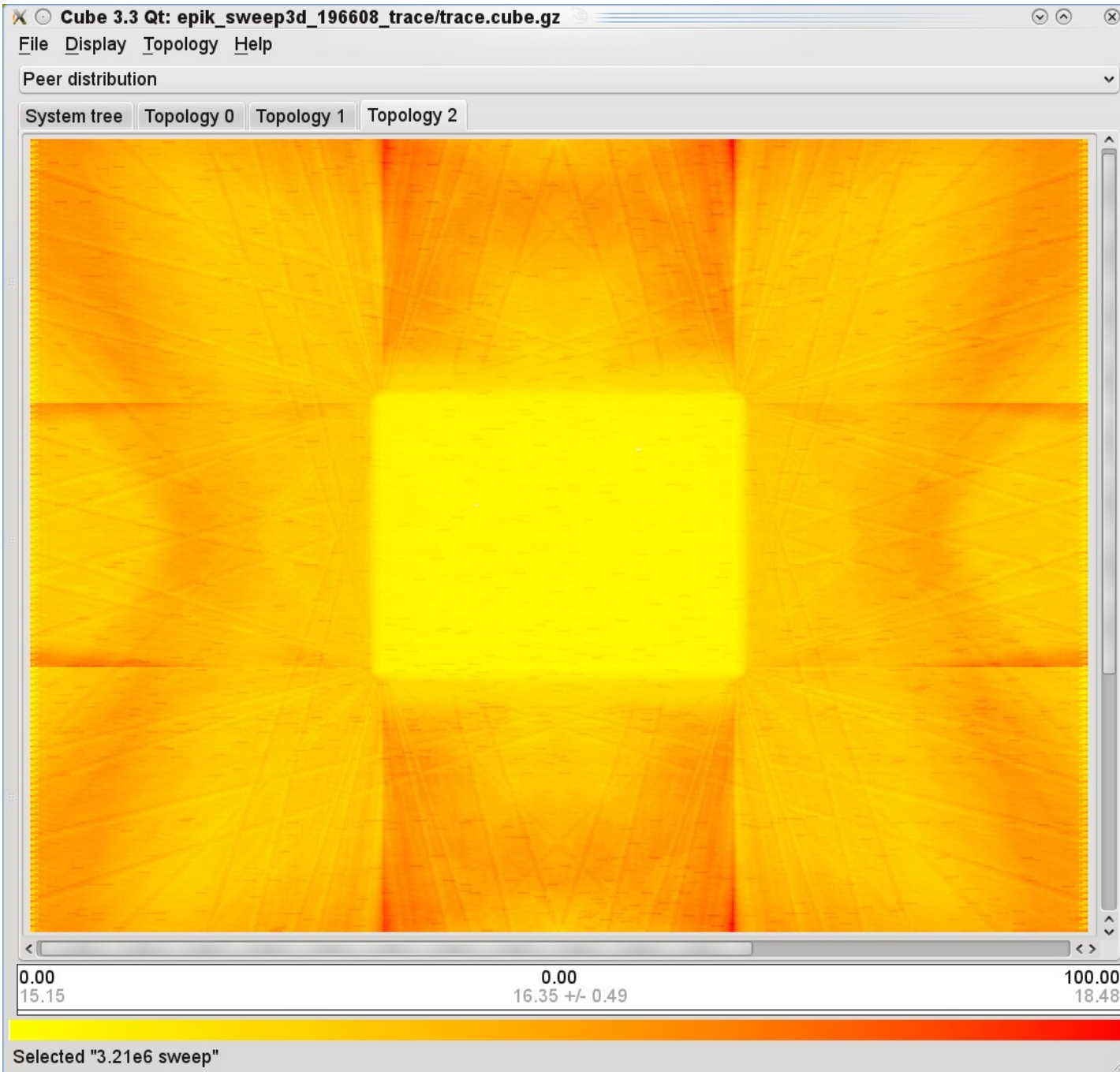
Sweep3D grid topology view



- Application's 2D topology of 512x384 procs
- Reveals clear pattern in the distribution of *Late Sender* metric values
- Only partially explicable by wavefront sweeps from each corner

Sweep3D grid topology view

- Application's 2D topology of 512x384 procs
- Computational imbalance in **sweep** routine flux corrections
- Superimposed with wavefront communication results in the distribution of *Late Sender* time



PFLOTRAN

3D reservoir simulator developed by LANL/ORNL

- Fortran code employing PETSc toolkit & HDF5 I/O
- built with PrgEnv-pgi compilers and typical optimizations
- run with 2b dataset for 10 timesteps on Jaguar Cray XT5
 - *most of run time in initialization phase*
 - *FLOW+TRAN(sport) stepper scaled to ~16k*

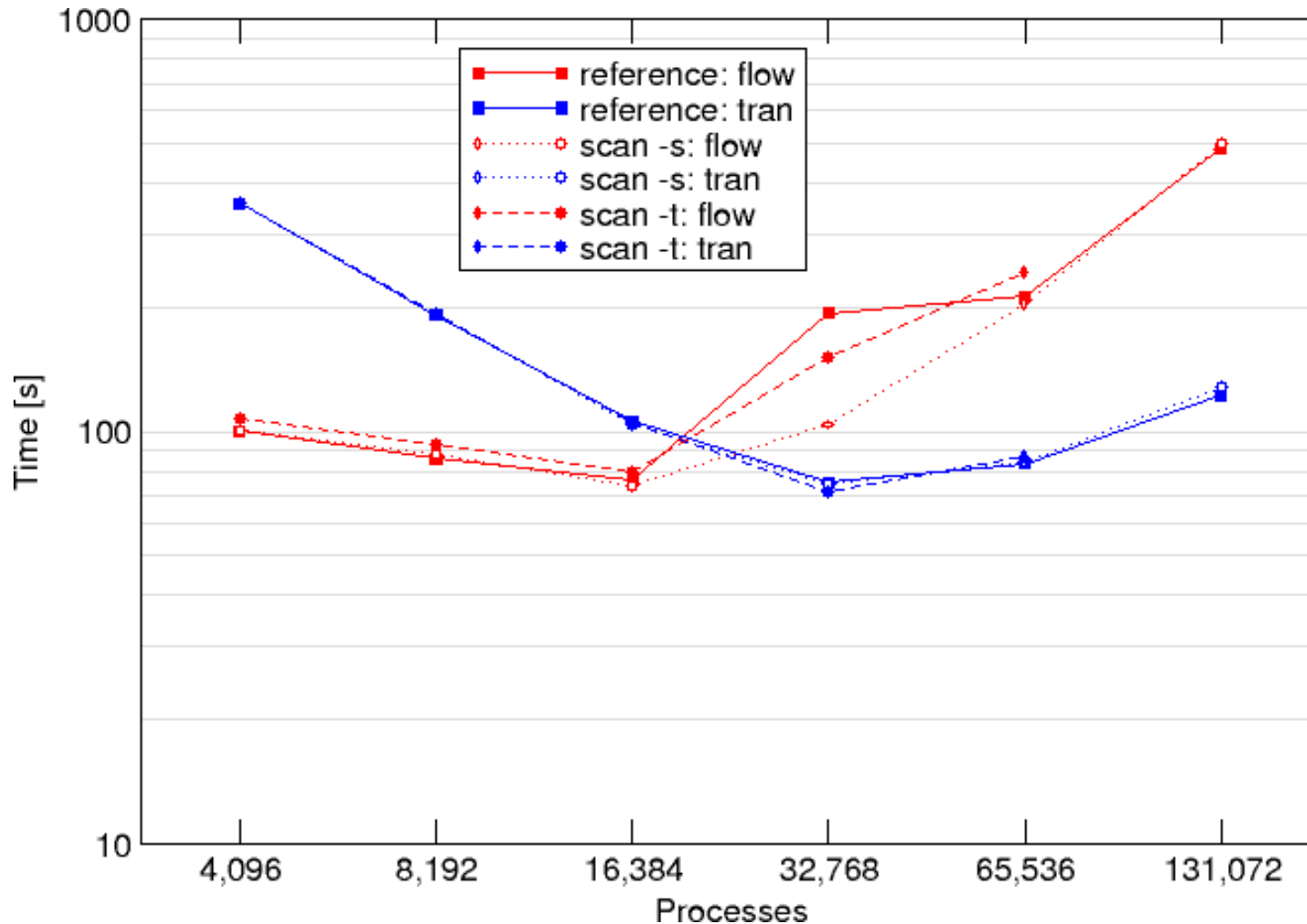
Automatically instrumented with Scalasca (using PGI compilers)

Initial (small-scale) summary measurement used to define filter files specifying all purely computational user-level source routines

Scalasca summary & trace experiments collected using filters

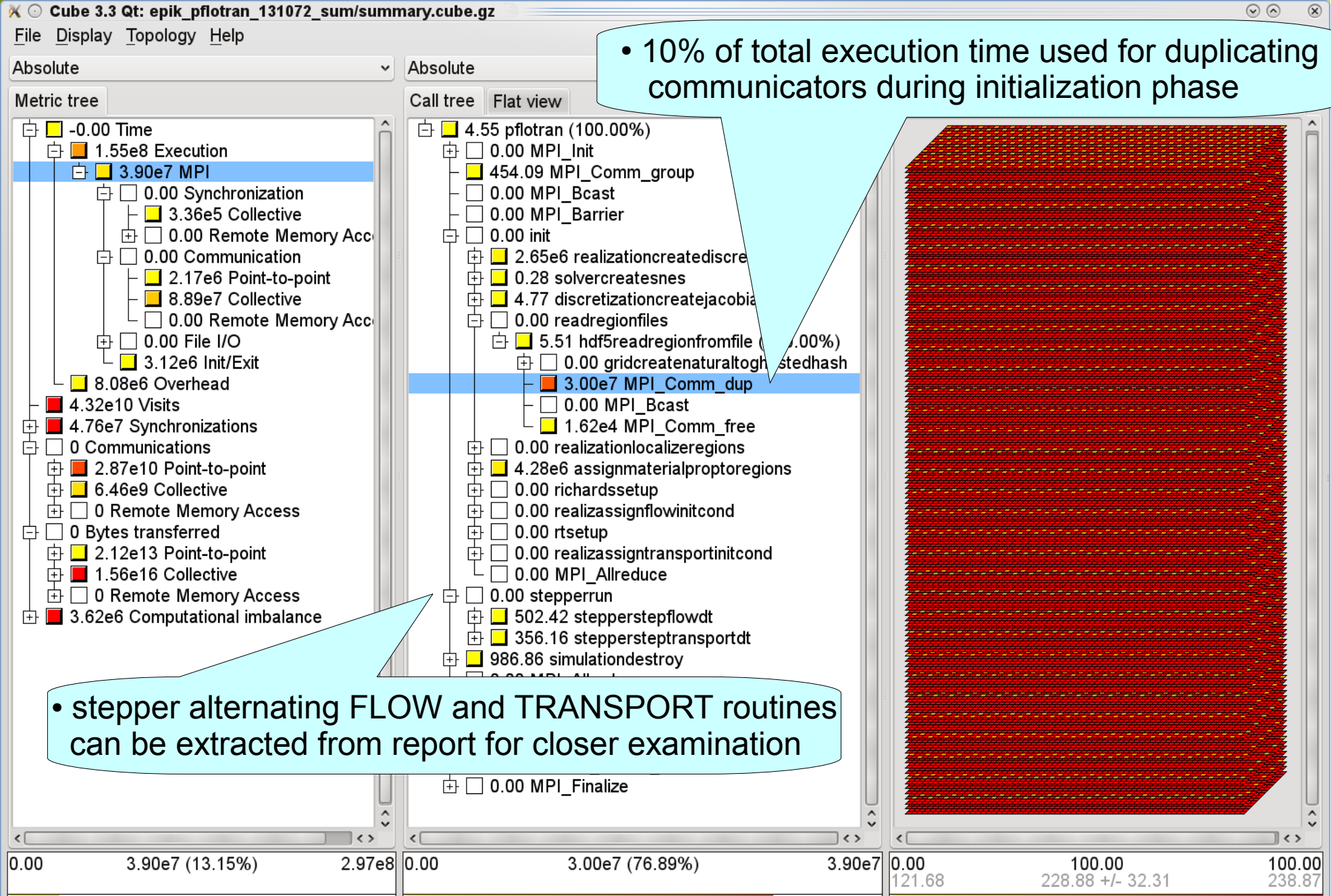
- dilation <3% compared to fully-optimized reference execution

PFLOTRAN scalability measurements

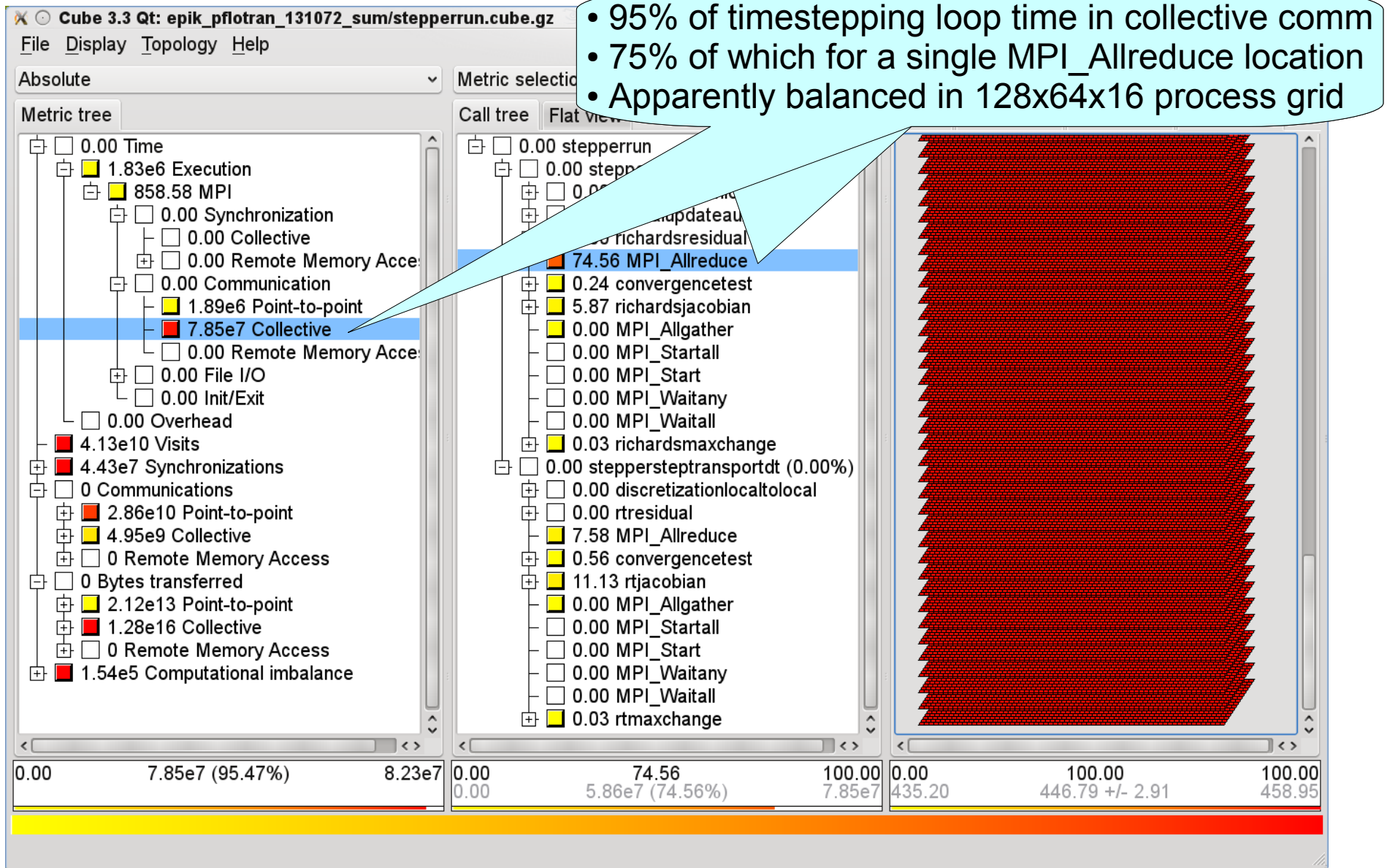


- 2b dof dataset, 10 timesteps
- *Strong scaling* due to fixed problem size in each run
- Transport scales to 32k, Flow to 16k
- Scalasca summary measurement experiments taken to 128k

[Scalasca 1.3 summary & trace experiments on Jaguar XT5:
less than 3% measurement dilation compared to uninstrumented]

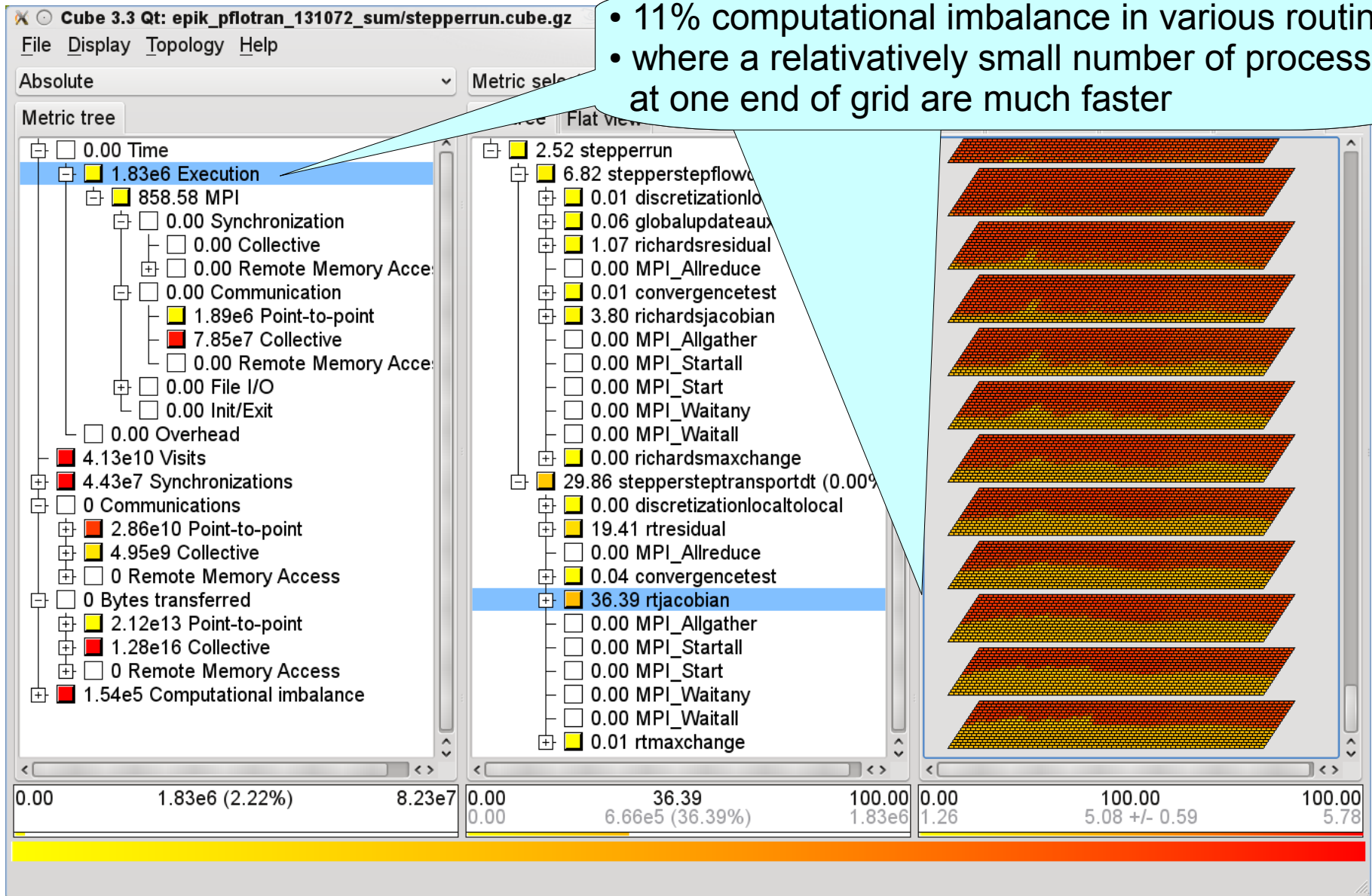


PFLOTRAN stepper bottleneck



PFLOTRAN computation time imbalance

- 11% computational imbalance in various routines
- where a relatively small number of processes at one end of grid are much faster



PEPC-B on Cray XT4 & BG/P case study

Coulomb solver used for laser-plasma simulations

- Developed by Paul Gibbon (JSC)
- Tree-based particle storage with dynamic load-balancing
- PRACE benchmark configuration, including file I/O

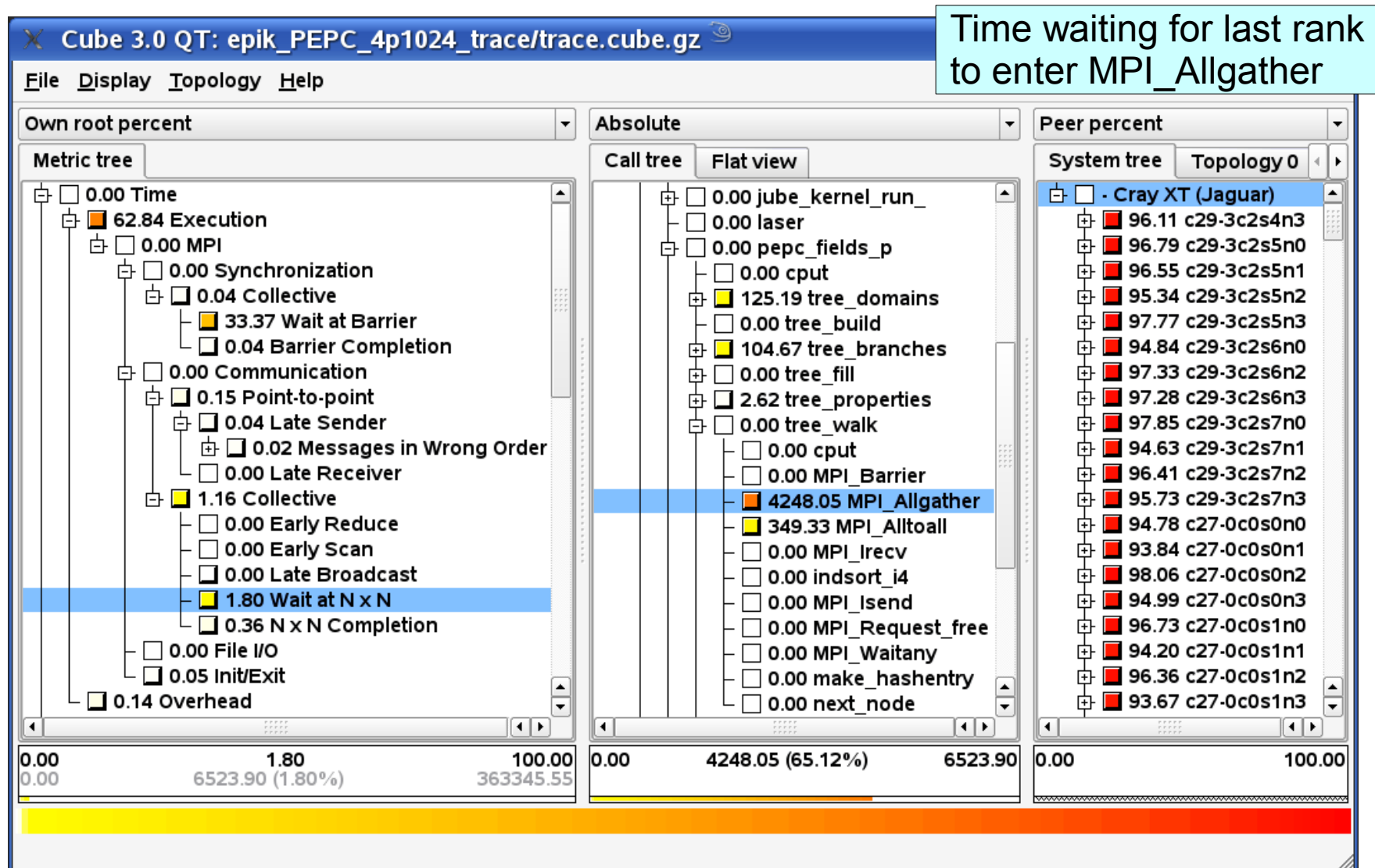
Run on *Jaguar* Cray XT4 in VN (4p) mode with 1024 processes

- 4 processes per quad-core Opteron node, 360 seconds
- PGI compilers and Cray MPI, CNL, SeaStar interconnect

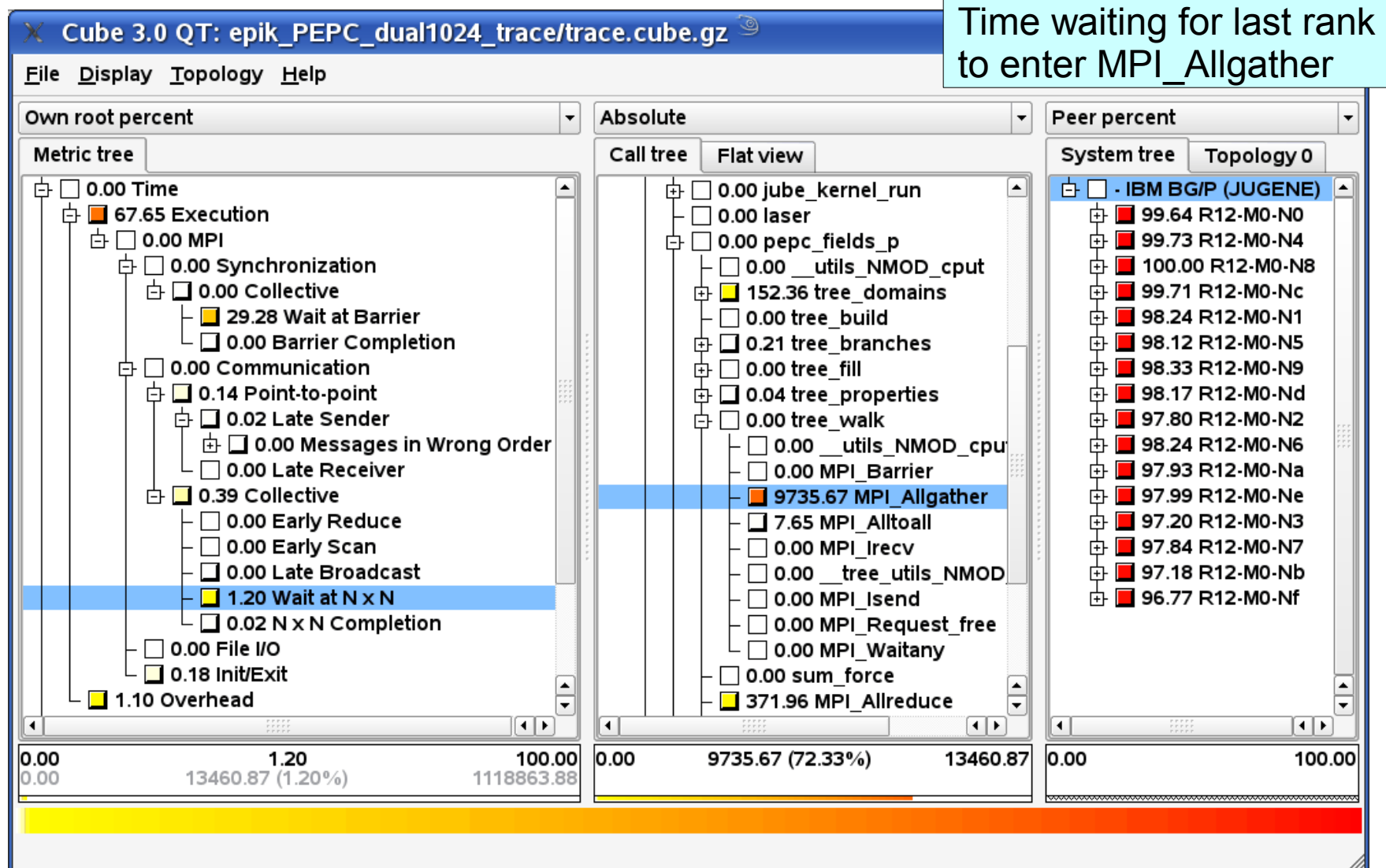
Run on *Jugene* BlueGene/P in dual mode with 1024 processes

- 2 processes per quad-core PowerPC node, 1100 seconds
- IBM XL compilers, MPI library and torus/tree interconnect

PEPC@1024 on Jaguar Cray XT4: Wait at NxN time



PEPC@1024 on Jugene IBM BG/P: Wait at NxN time



PEPC-B comparison of Cray XT4 & IBM BG/P

Scalasca measurements and analysis reports can be readily compared, despite very different processor and network performance

- different compilers affect function naming & in-lining
- both spend roughly two-thirds of time in computation
 - *tree_walk has expensive computation & communication*
- both waste 30% of time waiting to enter MPI_Barrier
 - *not localized to particular processes, since particles are regularly redistributed*
- Most of collective communication time is also time waiting for last ranks to enter MPI_Allgather & MPI_Alltoall
 - *imbalance for MPI_Allgather twice as severe on BlueGene/P, however, almost 50x less for MPI_Alltoall*
 - *collective completion times also notably longer on Cray XT*

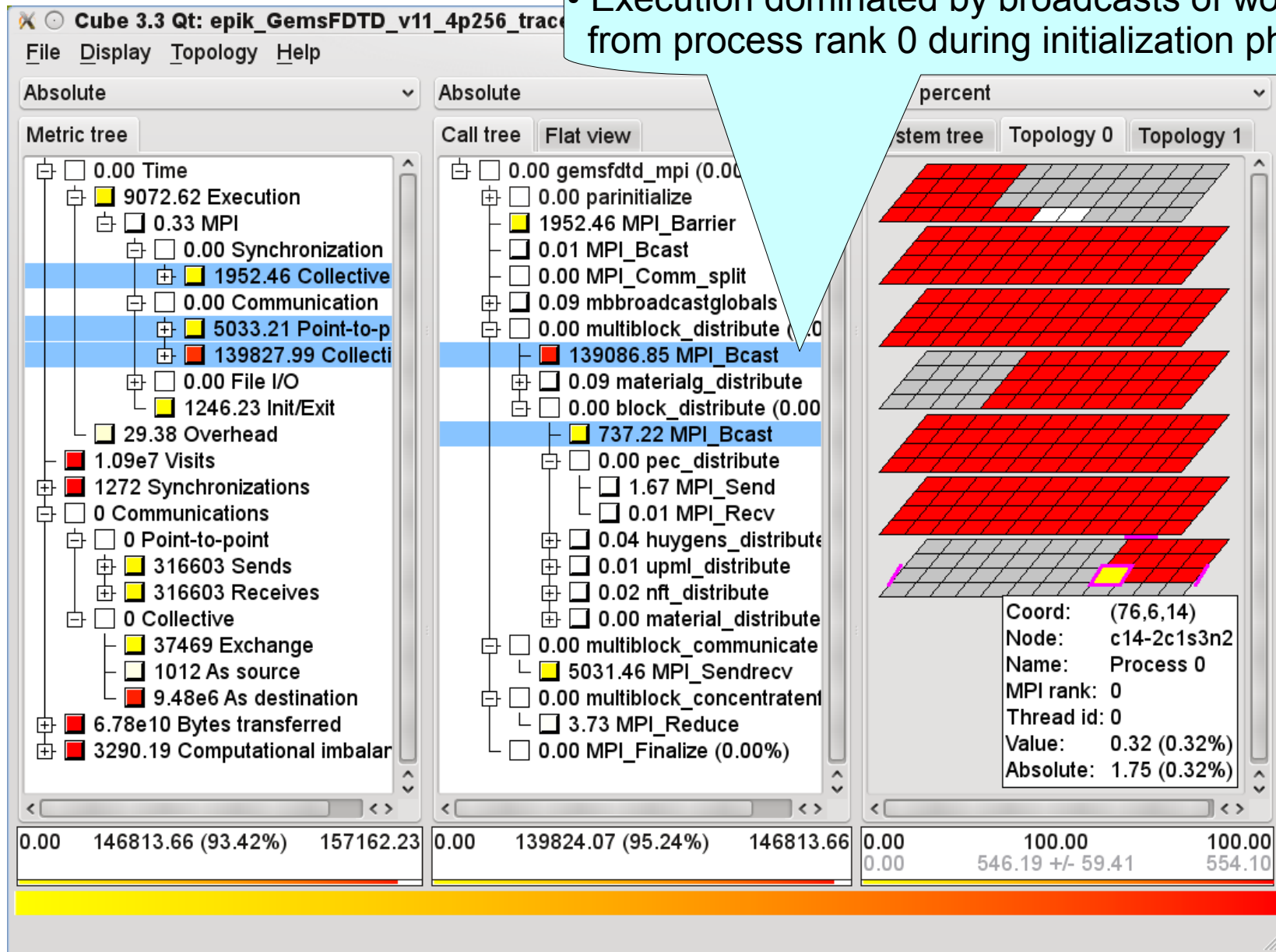
GemsFDTD

Computational electromagnetics solver from KTH GEMS project

- uses finite-difference time-domain method for Maxwell equations
 - *second-order accurate central-difference approximations of Faraday and Ampere laws on staggered Cartesian grid (Yee)*
- subset version included in SPEC MPI2007 benchmark suite
 - *computes radar cross-section of a perfectly conducting object*
 - *incident plane wave results in split of computational domain into scattered-field surrounded by total-field*
- scalability inhibitor discovered in original v1.1 initialization scheme
- performance re-engineered using Scalasca analyses for v2.0
 - *both absolute performance & scalability substantially improved*

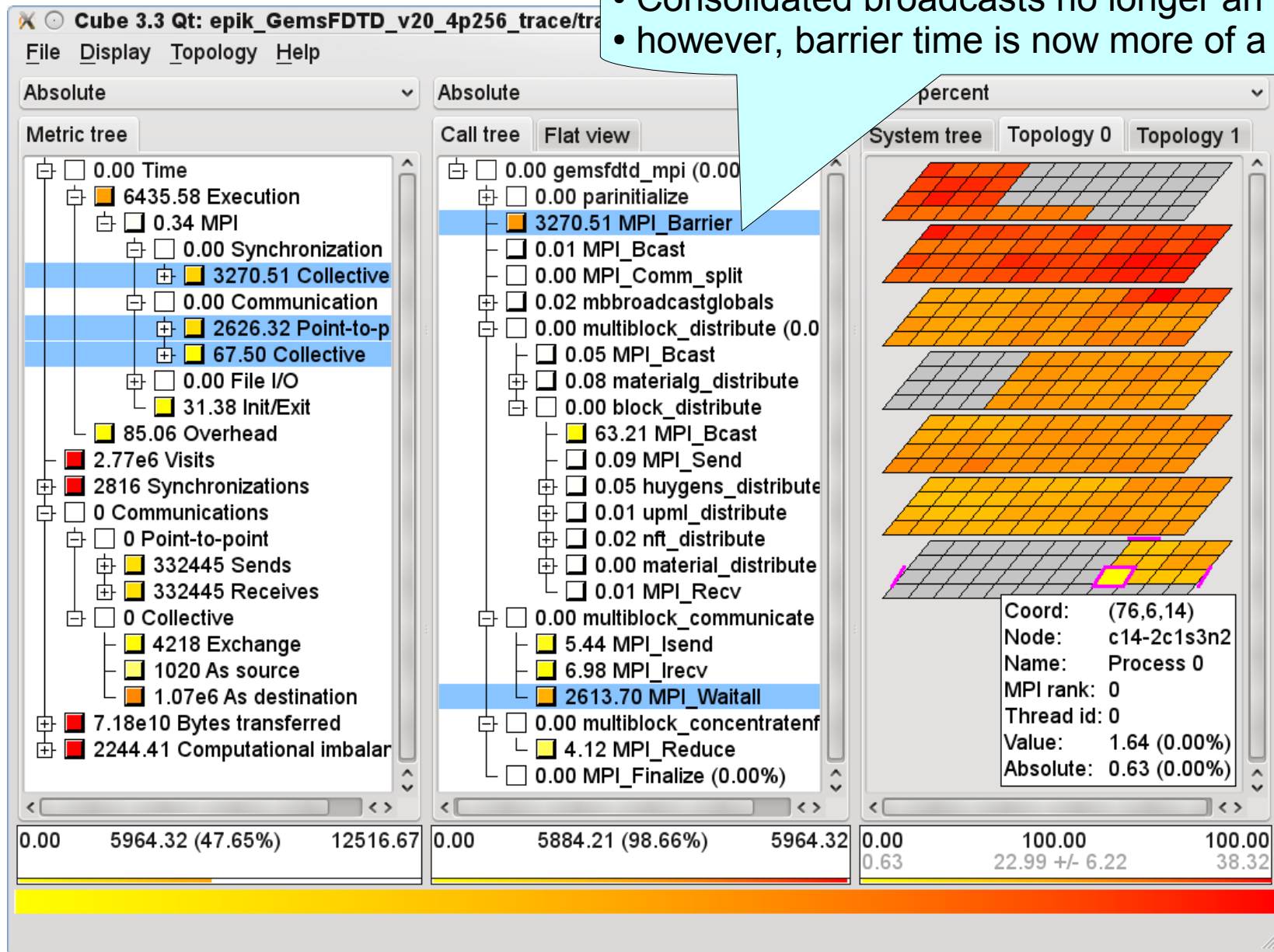
GemsFDTD v1.1

- Execution dominated by broadcasts of working set from process rank 0 during initialization phase



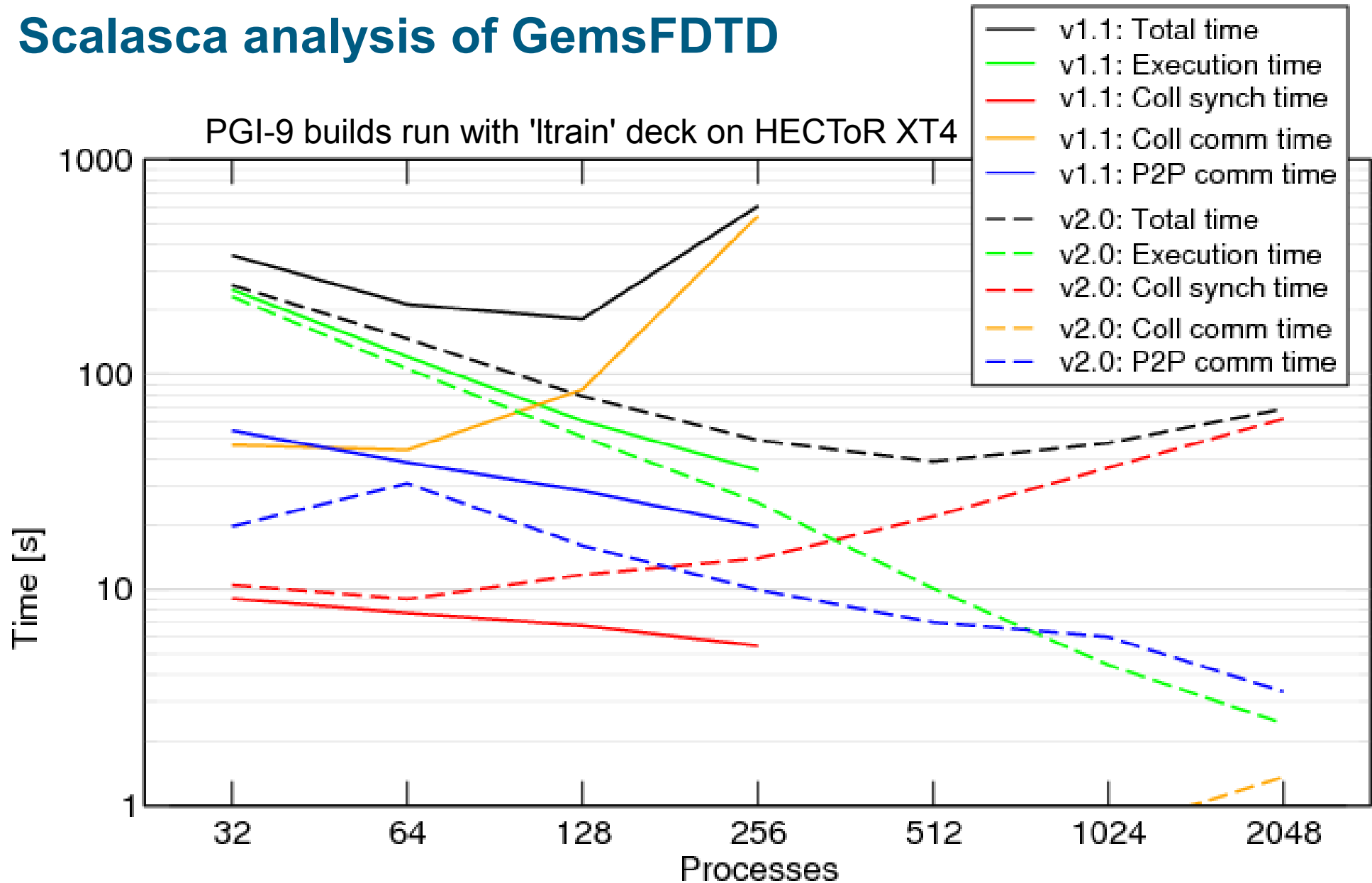
GemsFDTD v2.0

- Consolidated broadcasts no longer an issue
- however, barrier time is now more of a concern



Scalasca analysis of GemsFDTD

PGI-9 builds run with 'ltrain' deck on HECToR XT4



NPB3.3-MZ-MPI BT

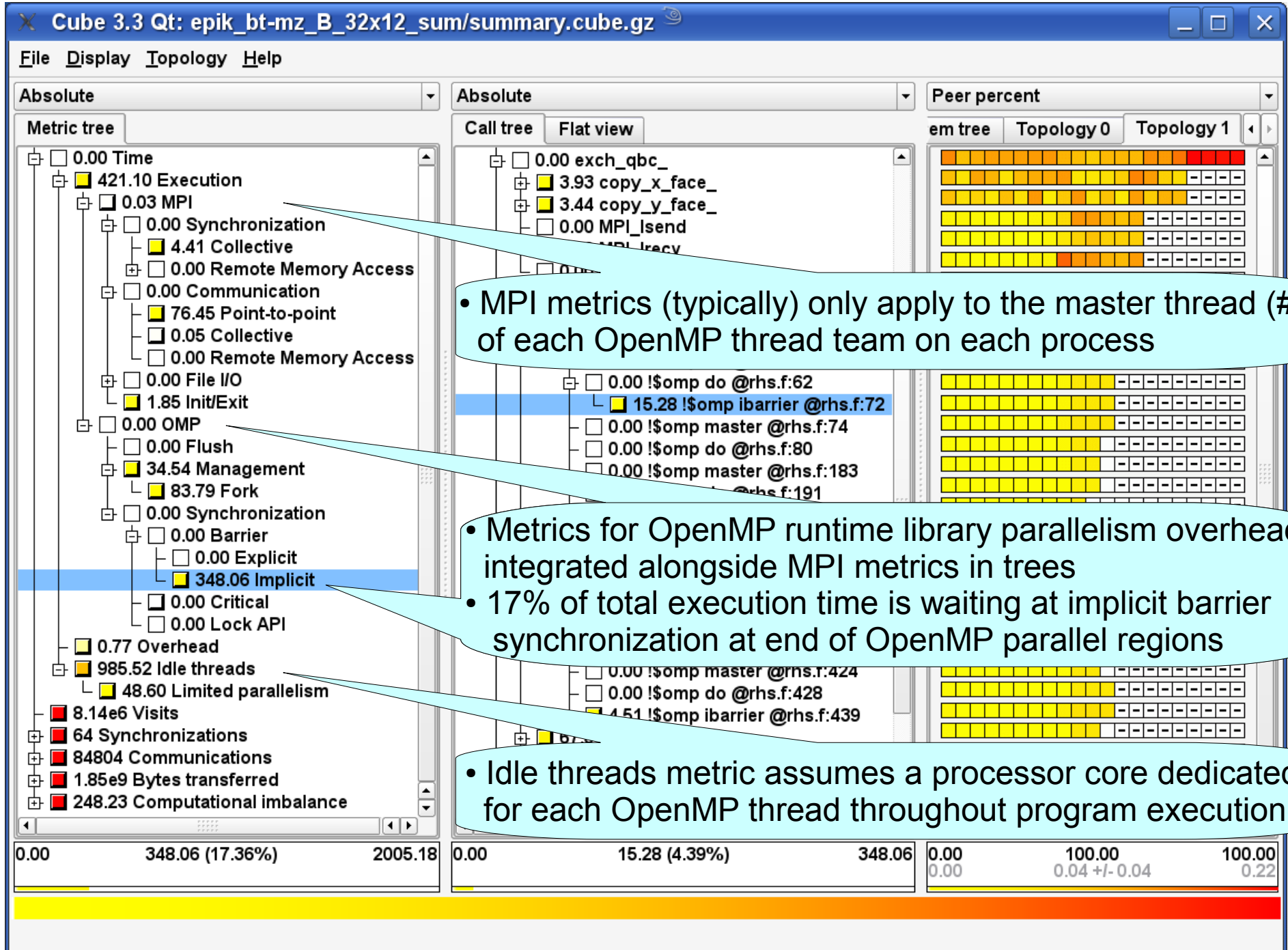
3D solution of unsteady, compressible Navier-Stokes equations

- NASA NAS parallel benchmark suite Block-Tridiagonal solver
- series of ADI solve steps in X, Y & Z dimensions
- multizone version uses MPI and OpenMP hybrid parallelization

Class B on *Kraken* Cray XT5 (12-core compute nodes)

- run with 32 MPI processes and OMP_NUM_THREADS=12
 - *More threads created on some processes (and less on others) as application attempts to balance work distribution*
 - *Node over-subscription results in excessive waiting at barriers and additional thread management overheads*
- runs 3x faster with fixed teams of 12 threads per compute node

BT-MZ: Scalasca OpenMP & hybrid metrics

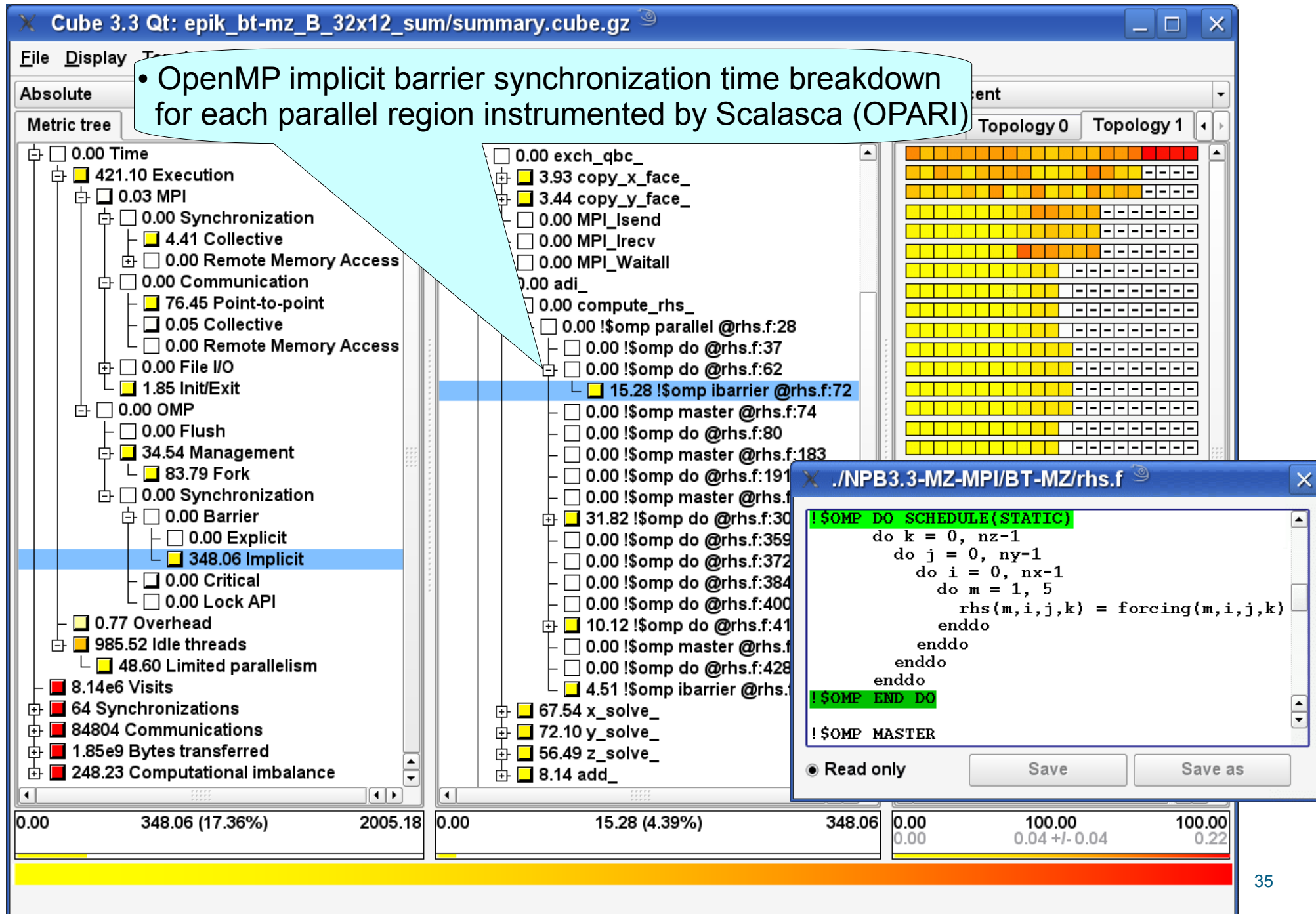


• MPI metrics (typically) only apply to the master thread (#0) of each OpenMP thread team on each process

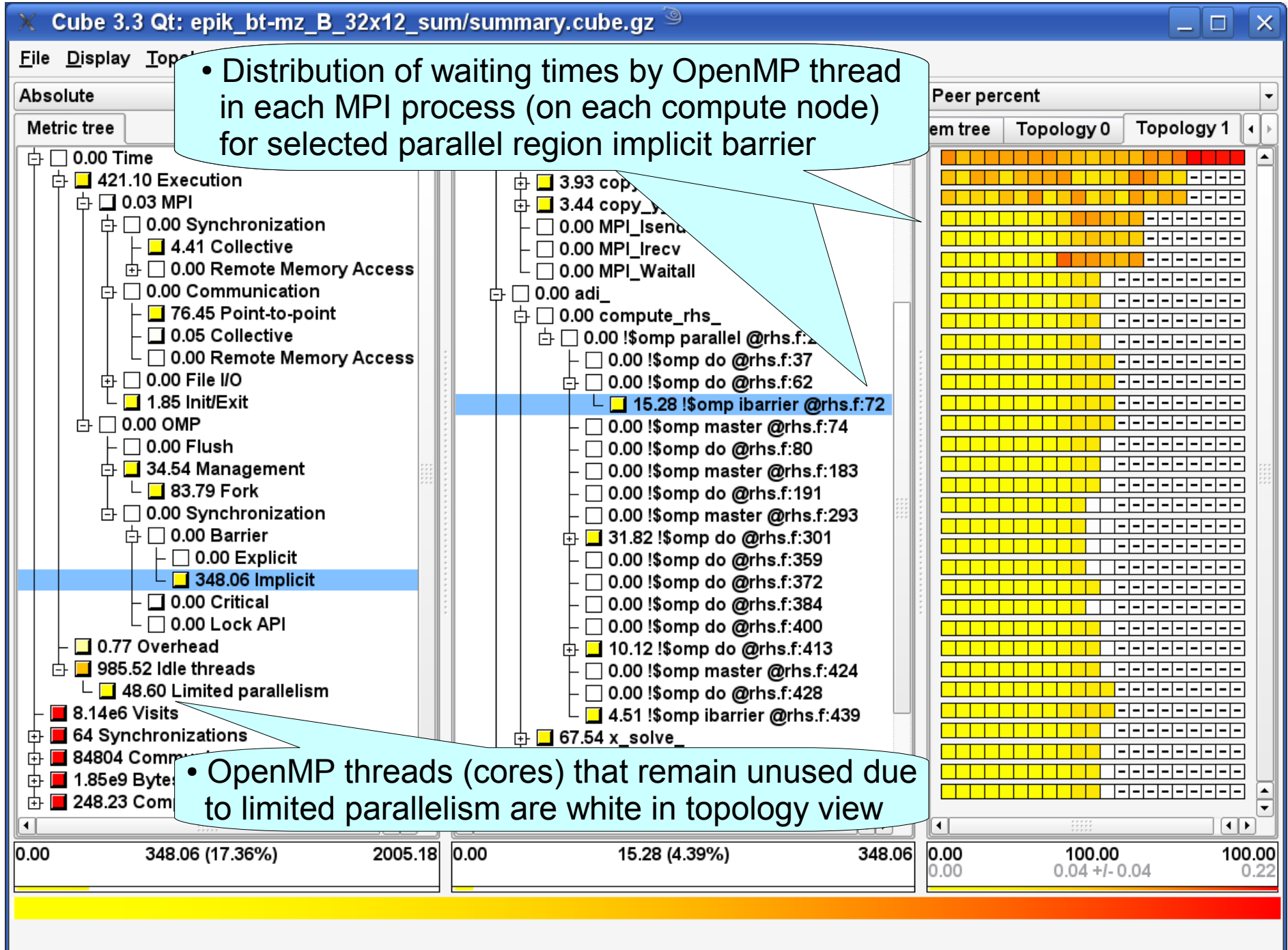
• Metrics for OpenMP runtime library parallelism overheads integrated alongside MPI metrics in trees
• 17% of total execution time is waiting at implicit barrier synchronization at end of OpenMP parallel regions

• Idle threads metric assumes a processor core dedicated for each OpenMP thread throughout program execution

BT-MZ: Scalasca OpenMP call tree



BT-MZ: Scalasca OpenMP thread metric values



Conclusions

Scalasca has been used to analyze & tune the execution performance of a variety of parallel applications

- MPI, OpenMP & hybrid OpenMP/MPI

Scalasca offers a range of instrumentation, measurement & analysis capabilities, with a simple GUI for interactive analysis report exploration

- complements the native Cray performance analysis toolset and third-party tools
- supports portability of toolset use and analysis report comparison

Scalasca has demonstrated performance measurement/analysis at unprecedented scale

- 192k processes on Cray XT
- 288k processes on IBM BG/P

Acknowledgments

The application and benchmark developers who generously provided their codes and/or measurement archives

The facilities who made their HPC resources available and associated support staff who helped us use them effectively

- ALCF, BSC, **CSC**, **CSCS**, **EPCC**, JSC, HLRN, HLRS, ICL, LRZ, NCAR, **NCCS**, **NICS**, RWTH, RZG, SARA, TACC, ZIH
- Access & usage supported by European Union, German and other national funding organizations

Cray personnel for answers to our detailed questions

Scalasca users who have provided valuable feedback and suggestions for improvements

Scalasca training

Scalasca-specific

- **CSCS**, Lugano, Switzerland: 16-17 June 2010

VI-HPS Tuning Workshops

- 3-day hands-on training with participants' own application codes, in context of Virtual Institute – High Productivity Supercomputing
- **SARA**, Amsterdam, The Netherlands: 26-28 May 2010

POINT/VI-HPS Tutorials

- full-day hands-on, in collaboration with US-based POINT project
- **Cluster2010**, Heraklion, Greece: 24 September 2010
- **SC2010**, New Orleans, USA: to be confirmed (November 2010)

Visit www.vi-hps.org/training for further details & announcements



Scalable performance analysis of large-scale parallel applications

- portable toolset for scalable performance measurement & analysis of MPI, OpenMP & hybrid OpenMP/MPI parallel applications
- supporting most popular HPC computer systems
- available under New BSD open-source license
- distributed on POINT/VI-HPS Parallel Productivity Tools Live-DVD
- sources, documentation & publications:
 - <http://www.scalasca.org>
 - [mailto: scalasca@fz-juelich.de](mailto:scalasca@fz-juelich.de)