scalasca

Performance analysis & tuning case studies

Brian Wylie & Markus Geimer
Jülich Supercomputing Centre
scalasca@fz-juelich.de

August 2012

Additional Live-DVD example experiments

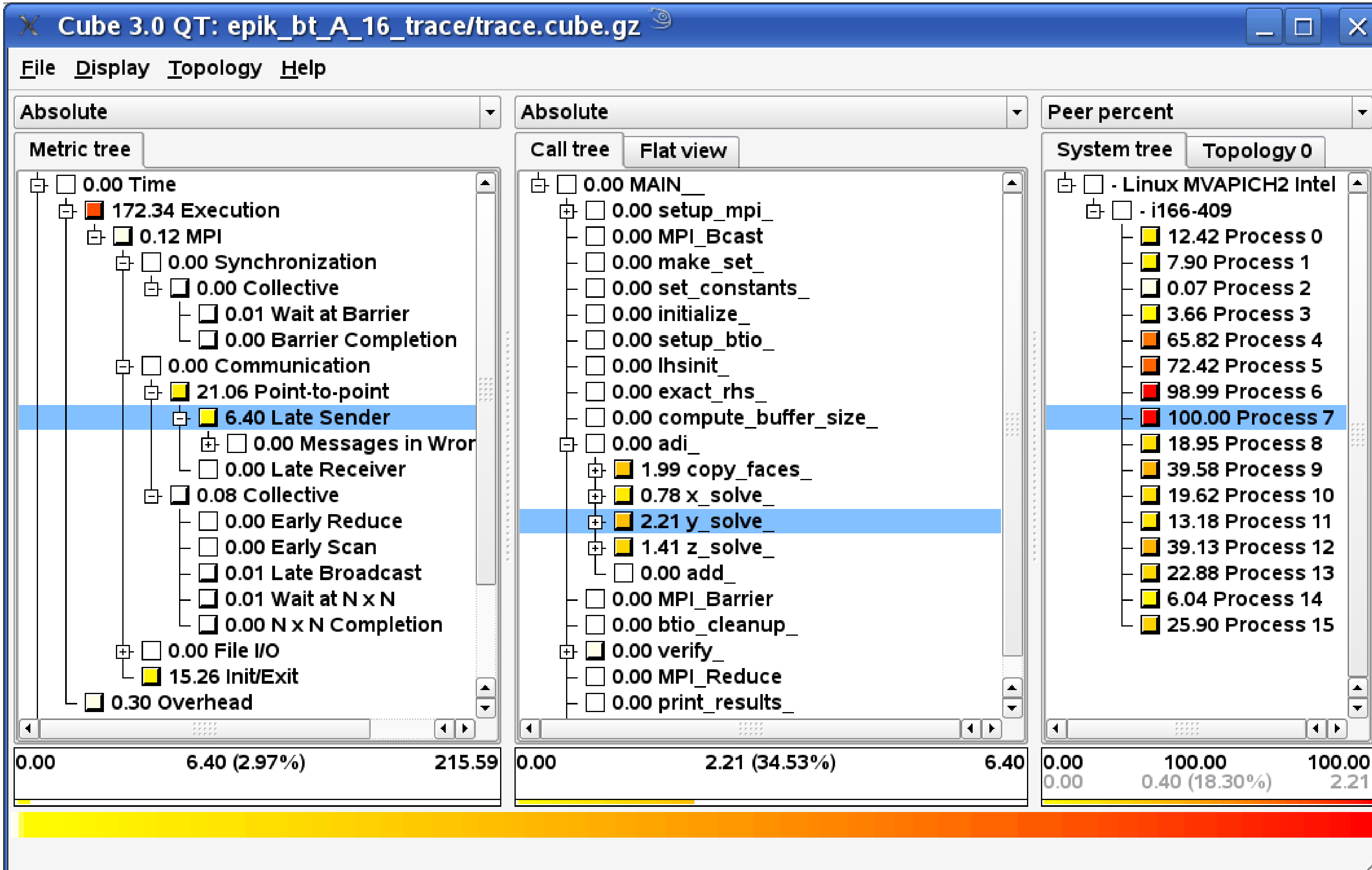
- Example experiment archives provided for examination:
  - jugene_sweep3d
    - 294,912 & 65,536 MPI processes on BG/P (trace)
  - jump_zeusmp2
    - 512 MPI processes on p690 cluster (summary & trace)
  - marenostrum_wrf-nmm
    - 1600 MPI processes on JS21 blade cluster, solver extract
    - summary analysis with 8 PowerPC hardware counters
    - trace analysis showing NxN completion problem on some blades
  - neptun_jacobi
    - 12 MPI processes, or 12 OpenMP threads, or 4x3 hybrid parallelizations implemented in C, C++ & Fortran on SGI Altix
  - ranger_smg2000
    - 12,288 MPI processes on Sun Constellation cluster, solve extract

Scalasca NPB-BT experiments

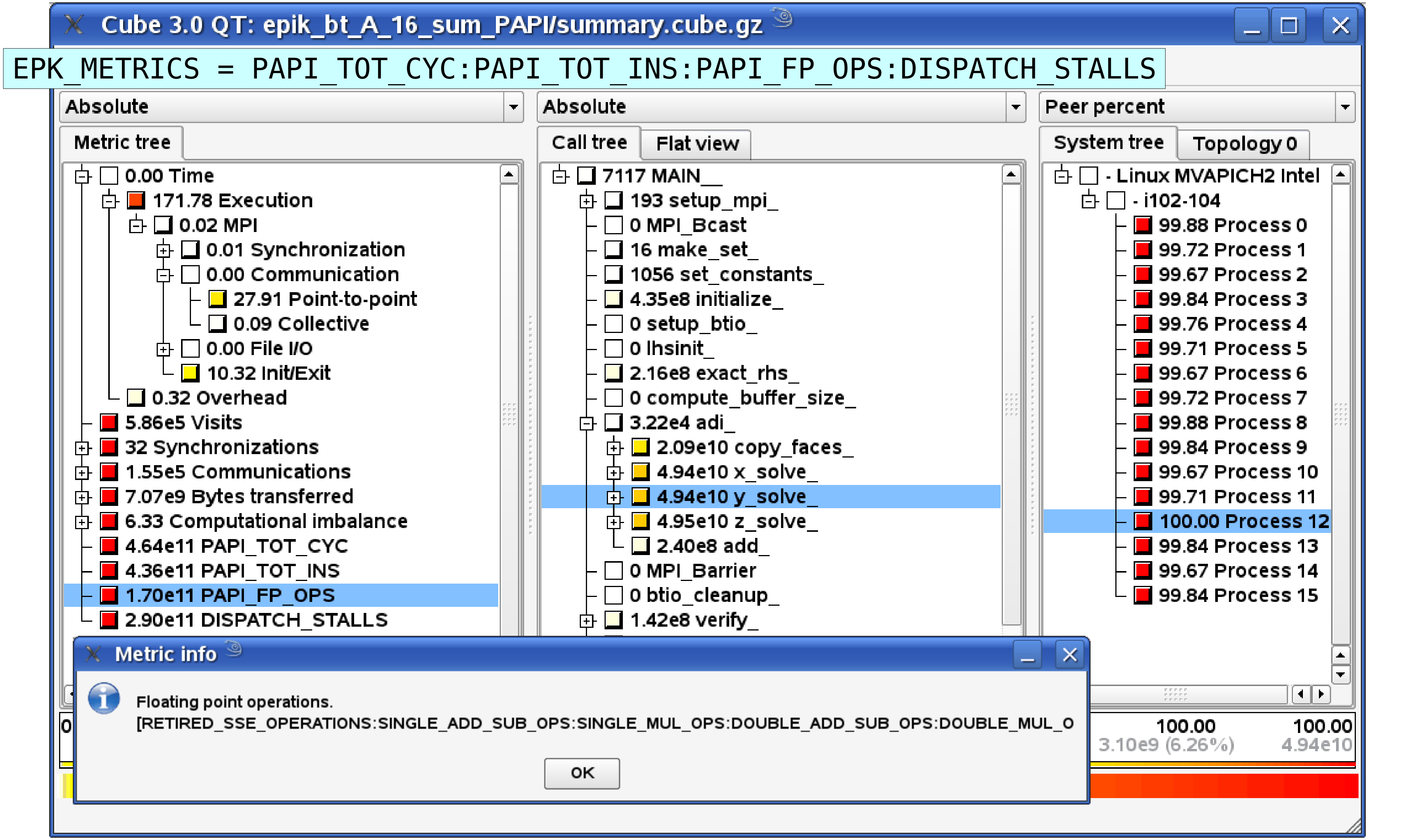
- Comparison of NPB-BT class A in various configurations run on a single dedicated 16-core cluster compute node
  - 16 MPI processes
    - optionally built using MPI File I/O (e.g., SUBTYPE=full)
    - optionally including PAPI counter metrics in measurement (e.g., EPK_METRICS=PAPI_FP_OPS:DISPATCH_STALLS)
  - 16 OpenMP threads
  - 4 MPI processes each with 4 OpenMP threads (MZ-MPI)
- NPB-BT-MZ class B on Cray XT5 (8-core compute nodes)
  - 32 MPI processes with OMP_NUM_THREADS=8
    - More threads created on some processes (and fewer on others) as application attempts to balance work distribution
- NPB-MPI-BT on BlueGene/P with 144k processes
  - 1536x1536x1536 gridpoints distributed on 384x384 processes

Concurrency & Computation: Practice & Experience 22(6):702-719 (2010)

16-process trace analysis

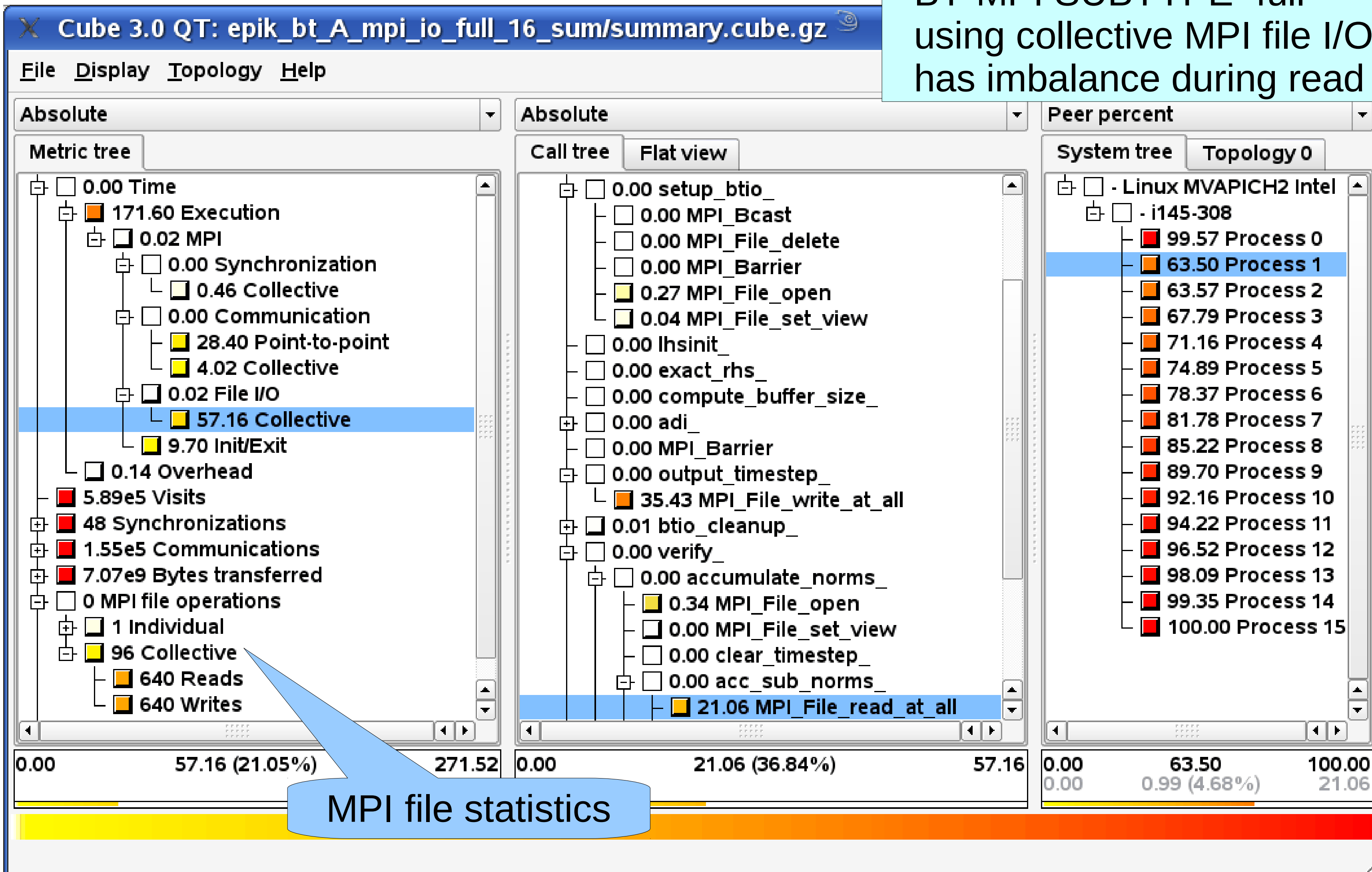


16-process summary analysis with HWC metrics



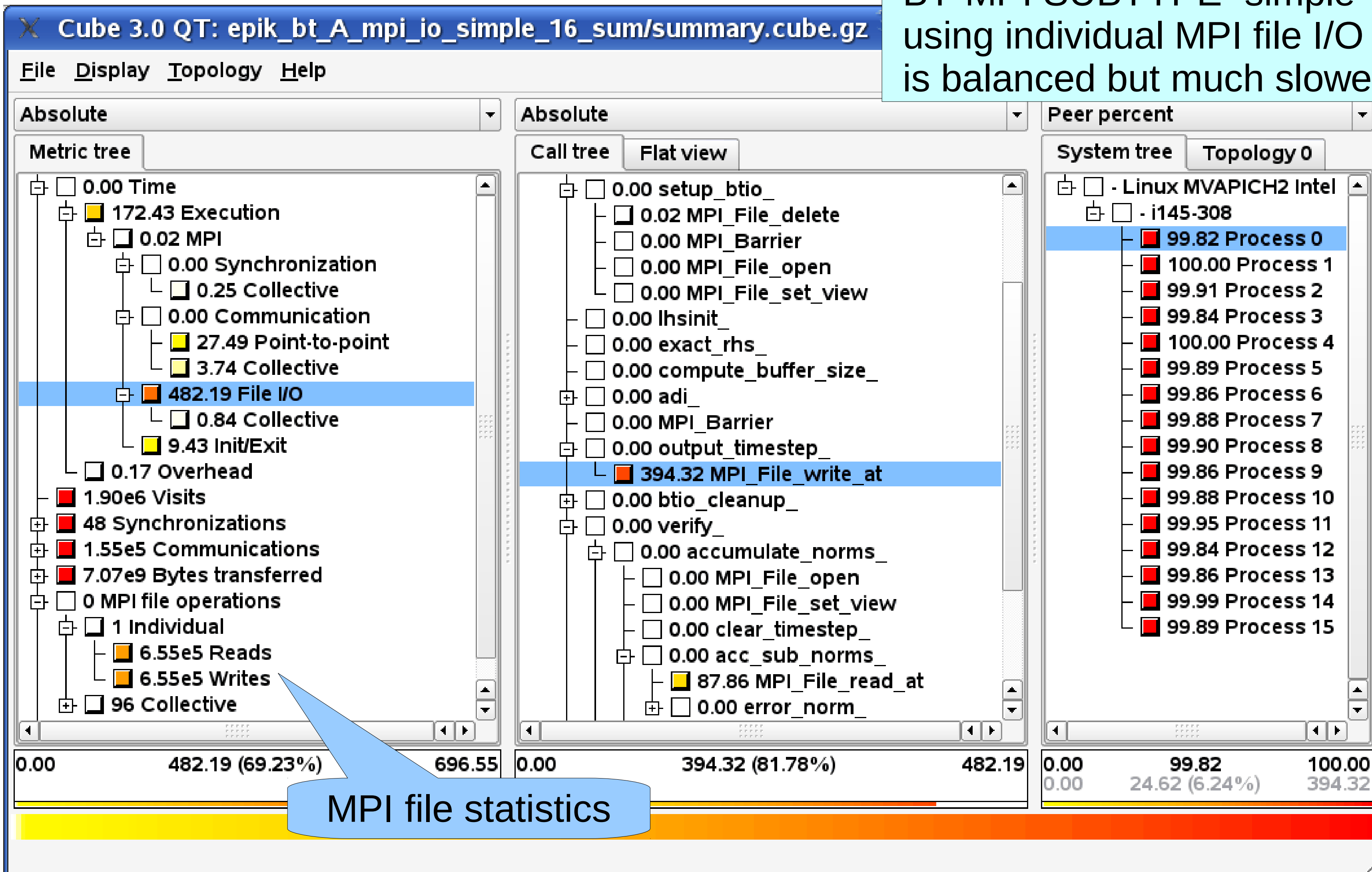
16-process summary analysis: MPI File I/O time

BT-MPI SUBTYPE=full using collective MPI file I/O has imbalance during read



MPI file statistics

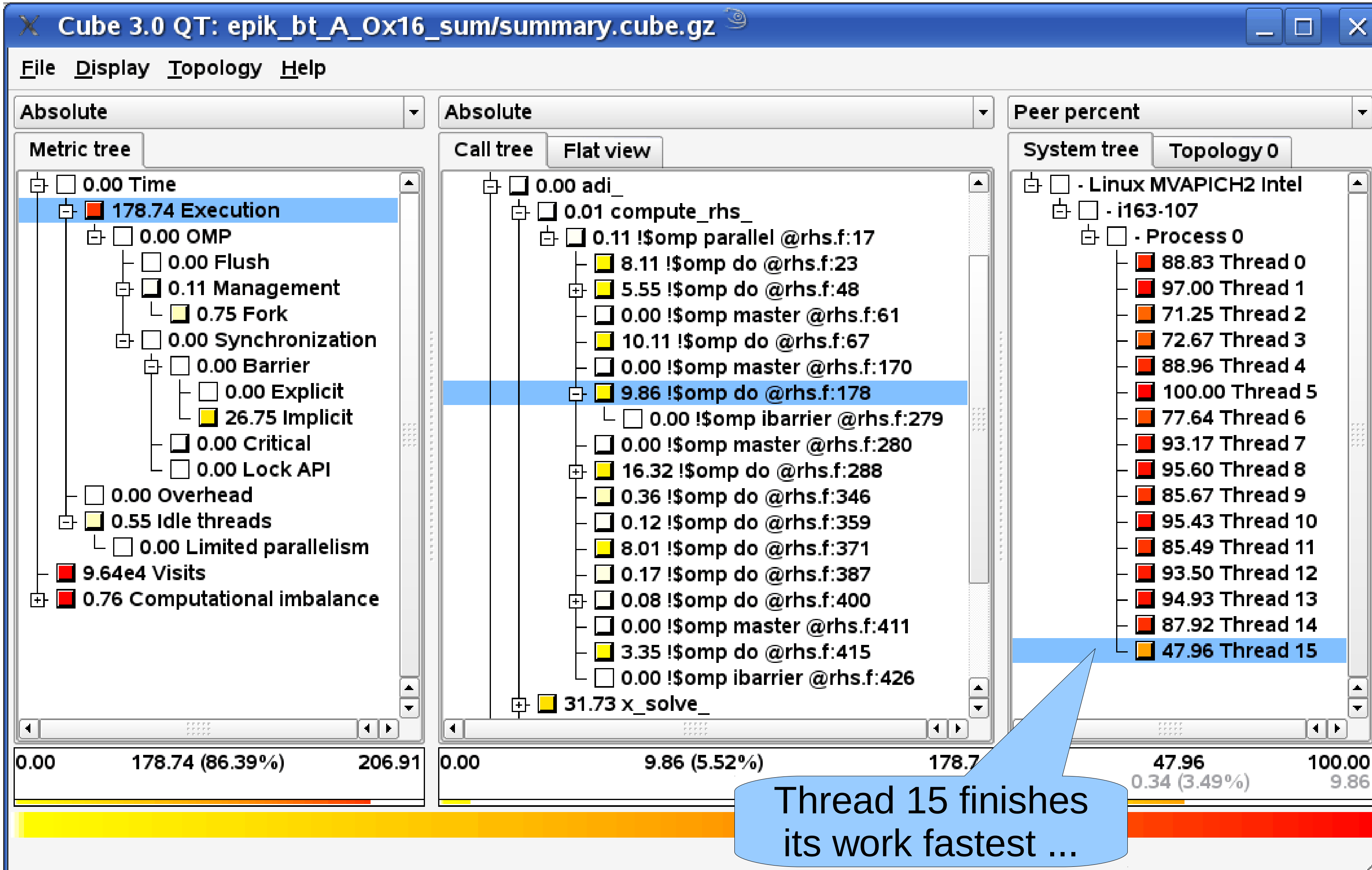
16-process summary analysis: MPI File I/O time



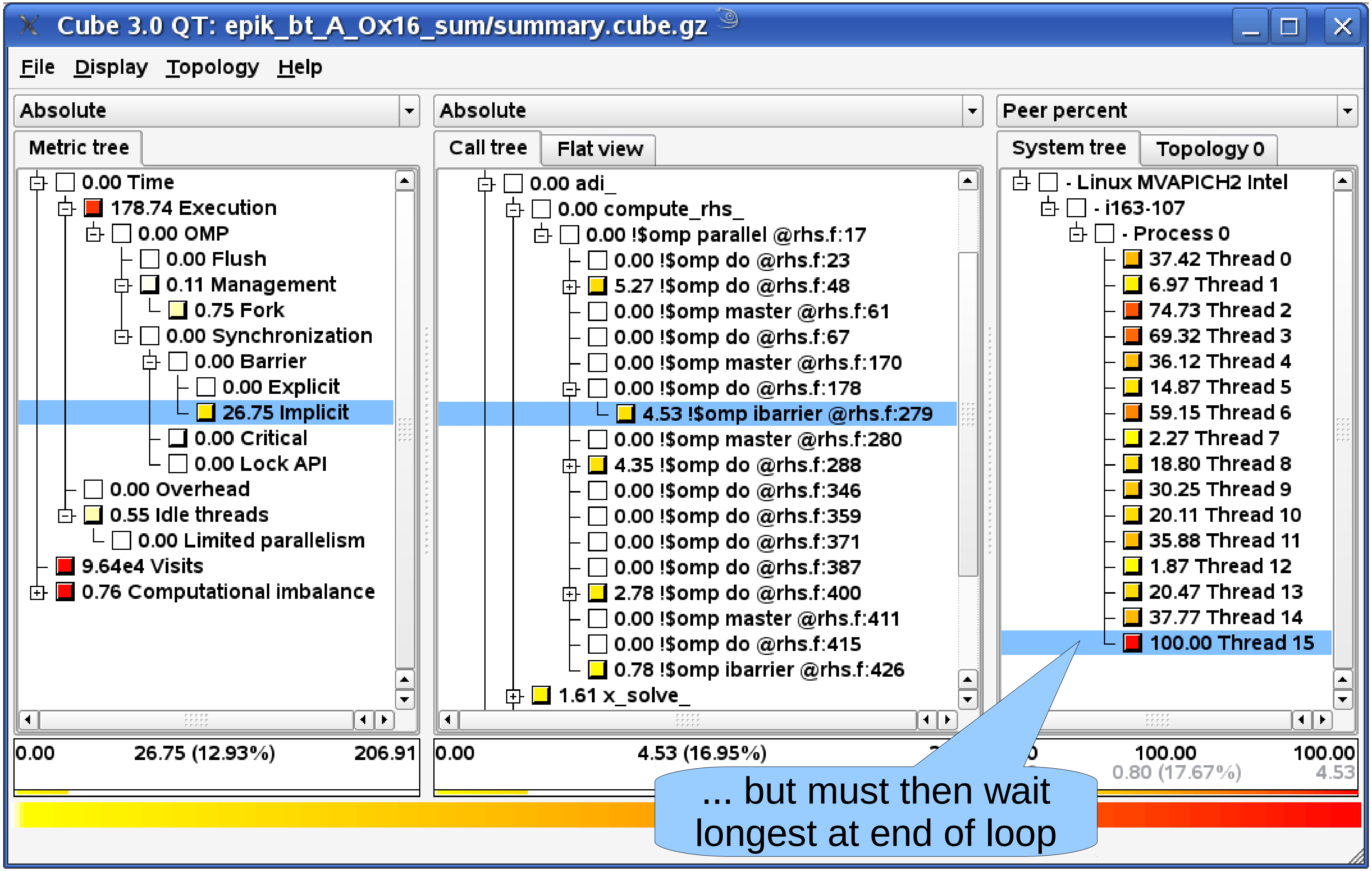
BT-MPI SUBTYPE=simple using individual MPI file I/O is balanced but much slower

MPI file statistics

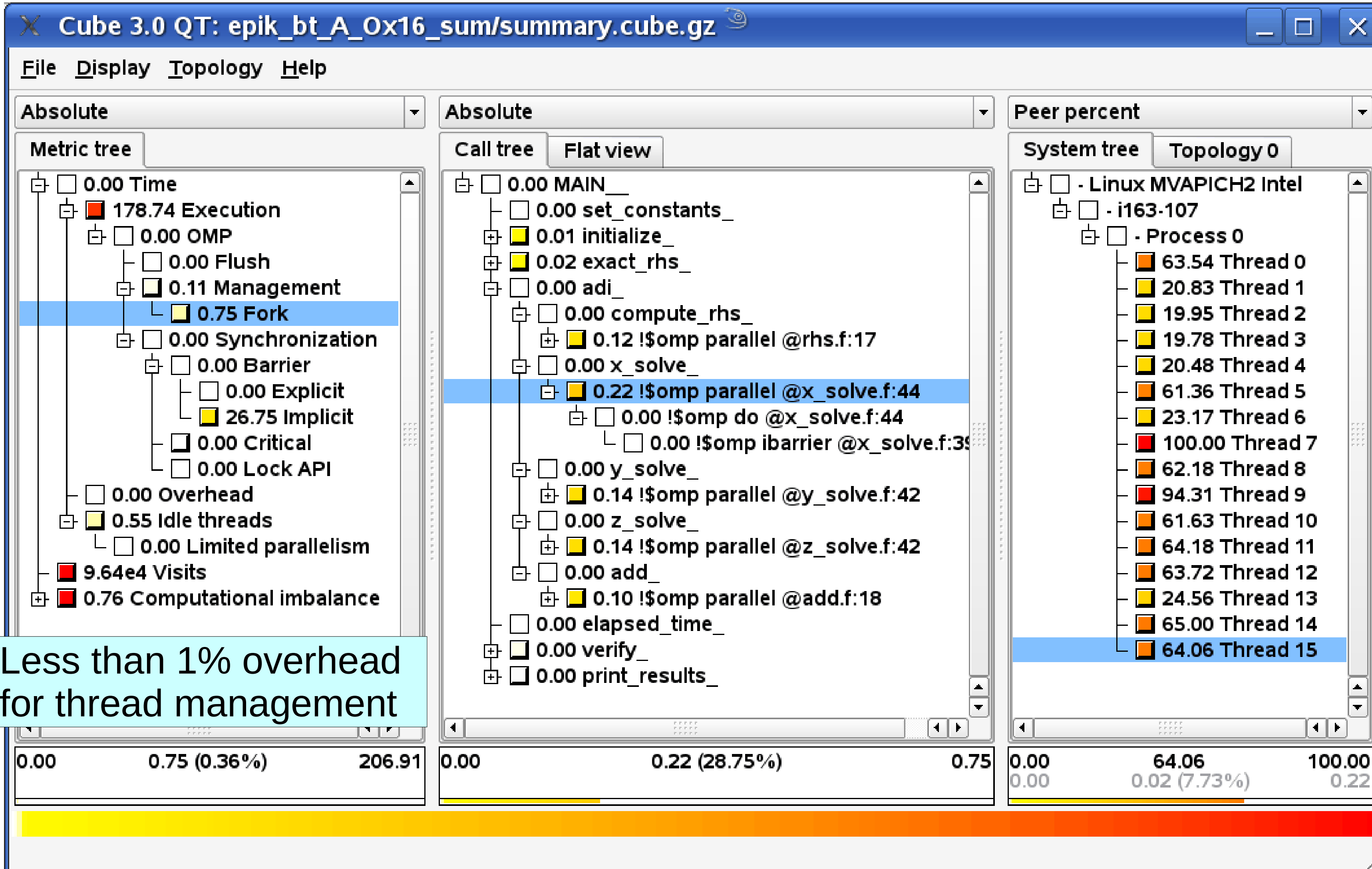
16-thread summary analysis: Execution time



16-thread summary analysis: Implicit barrier time

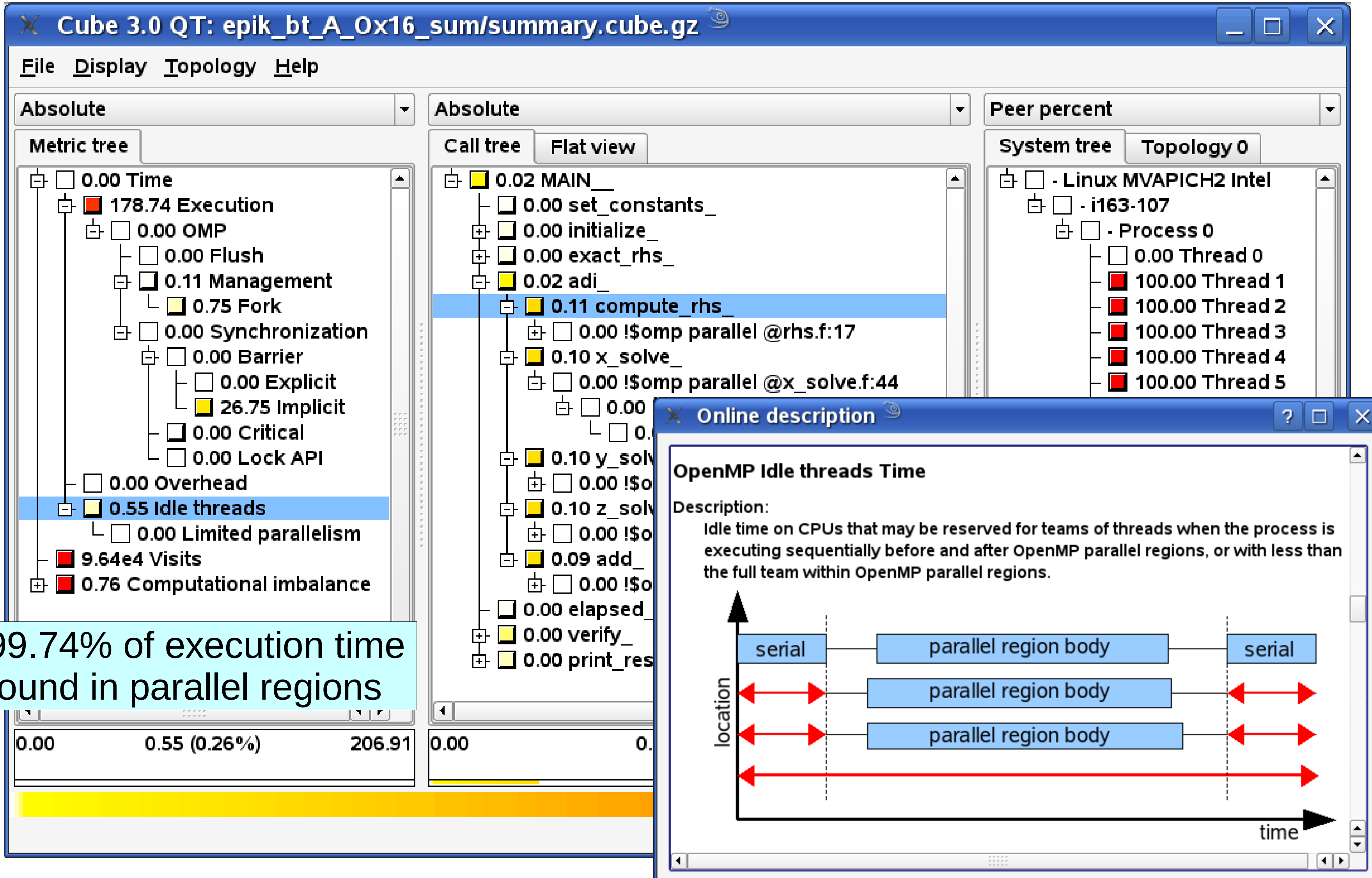


16-thread summary analysis: Thread fork time



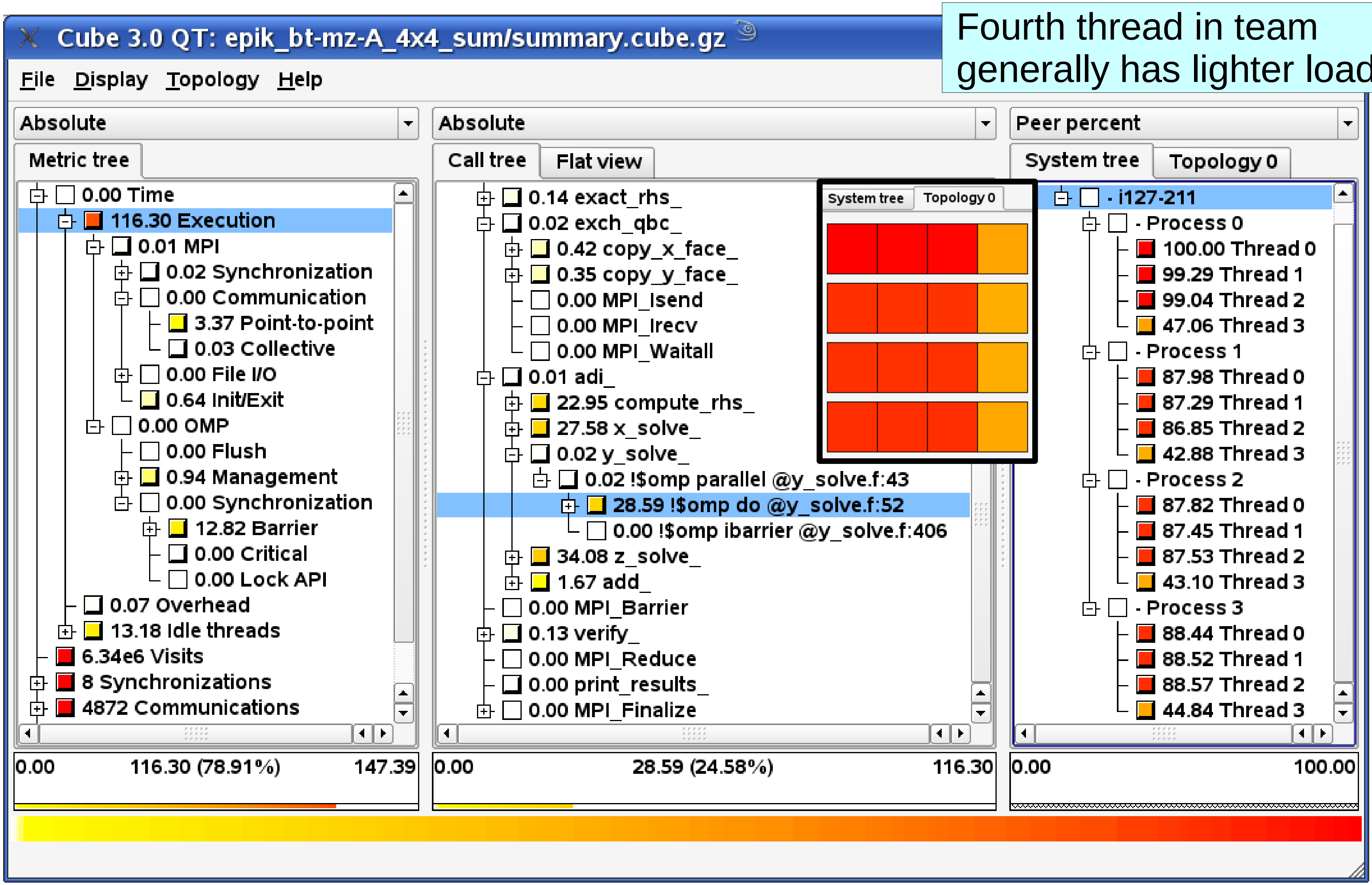
Less than 1% overhead for thread management

16-thread summary analysis: Idle threads time



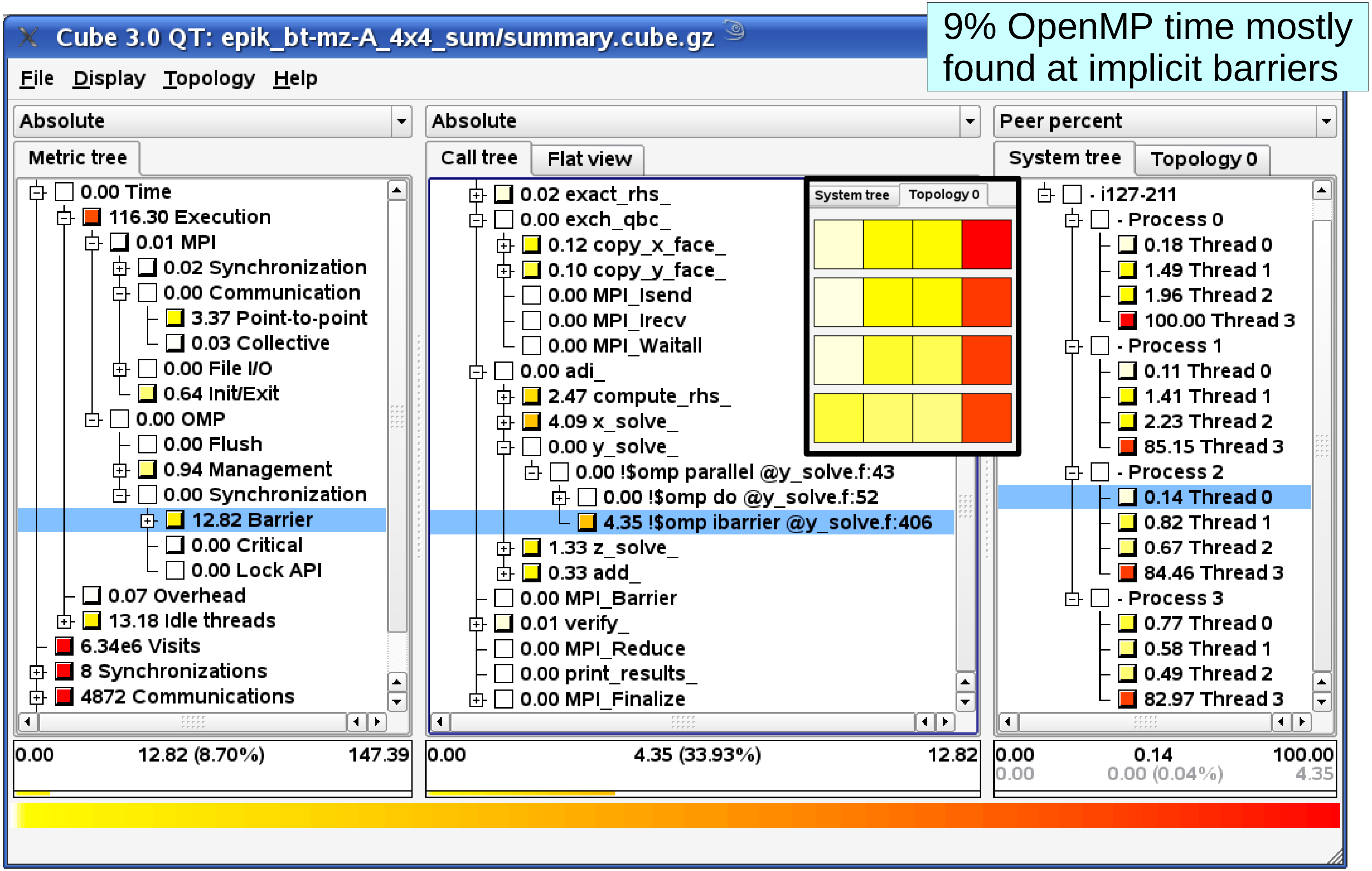
99.74% of execution time found in parallel regions

4x4 summary analysis: Execution time



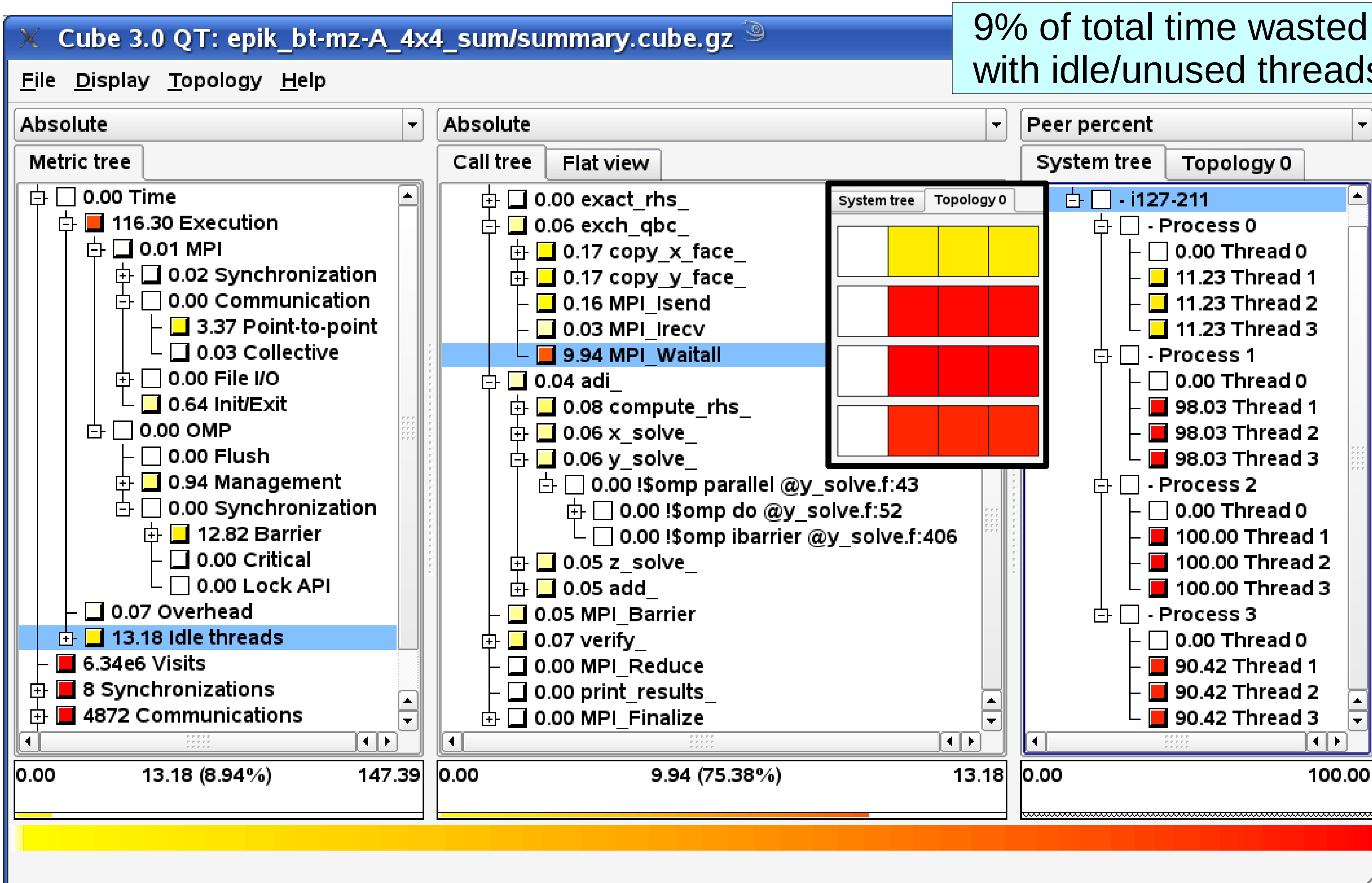
Fourth thread in team generally has lighter load

4x4 summary analysis: OpenMP time



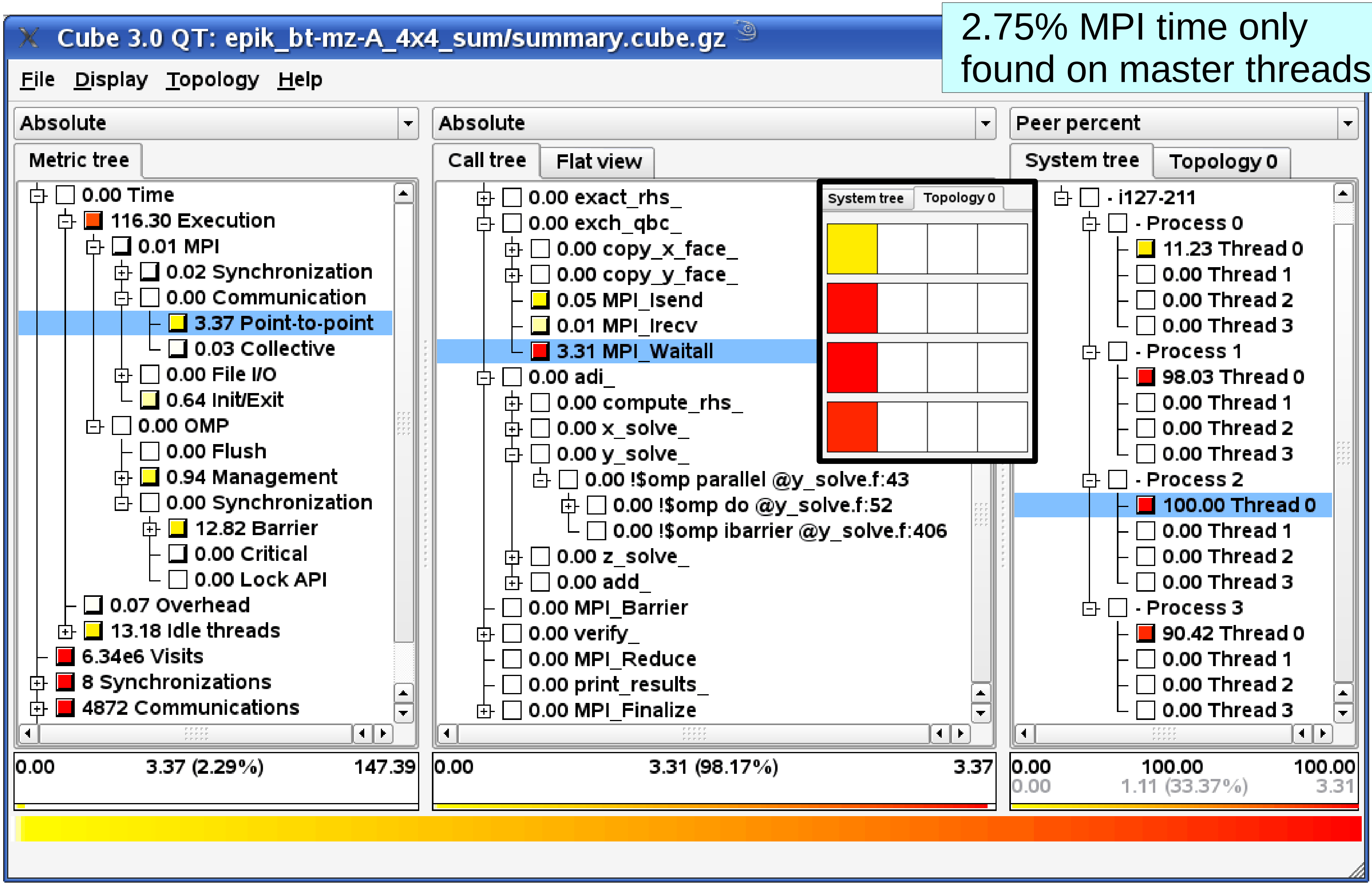
4x4 summary analysis: Idle threads time

9% of total time wasted with idle/unused threads

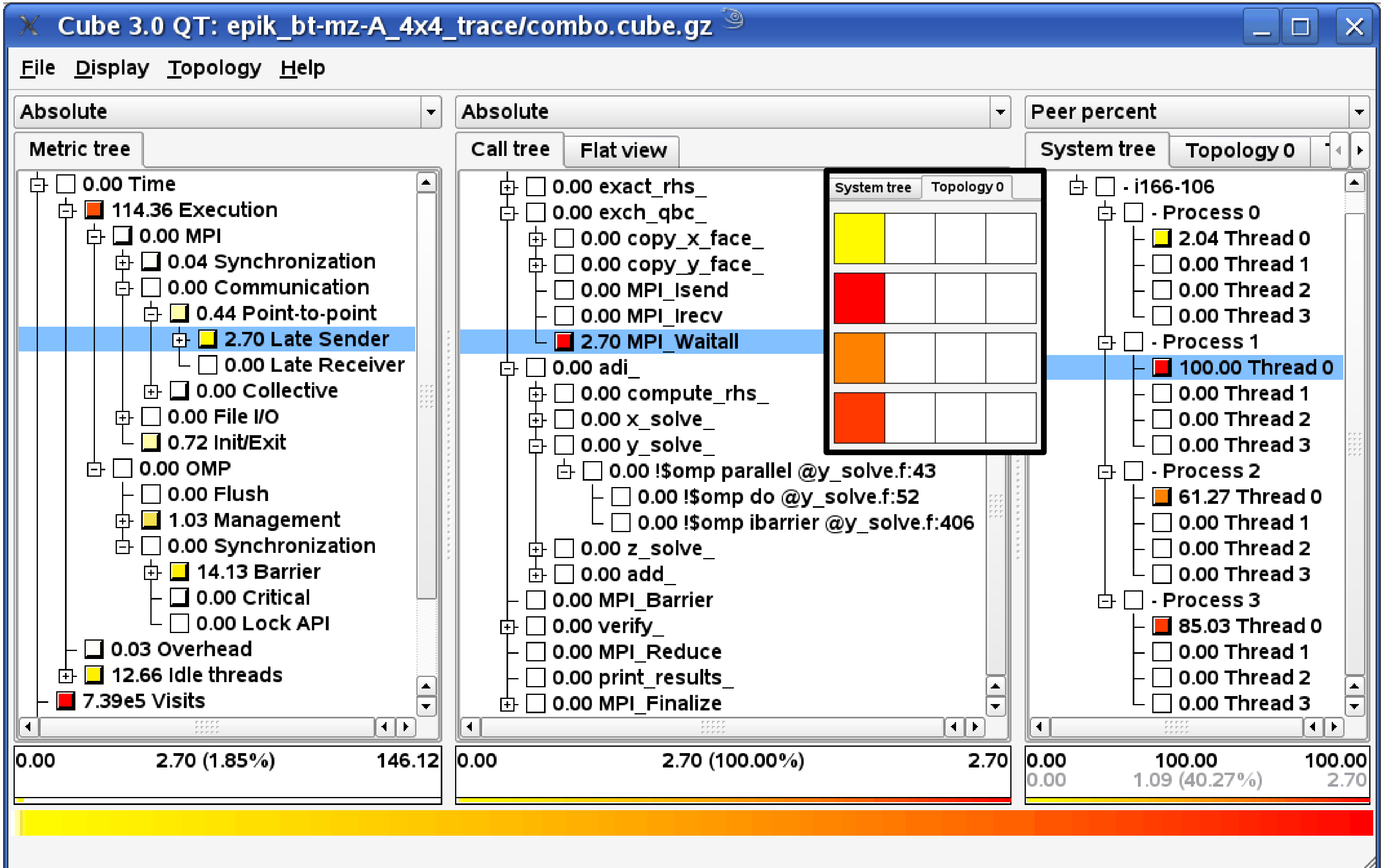


4x4 summary analysis: MPI time

2.75% MPI time only found on master threads

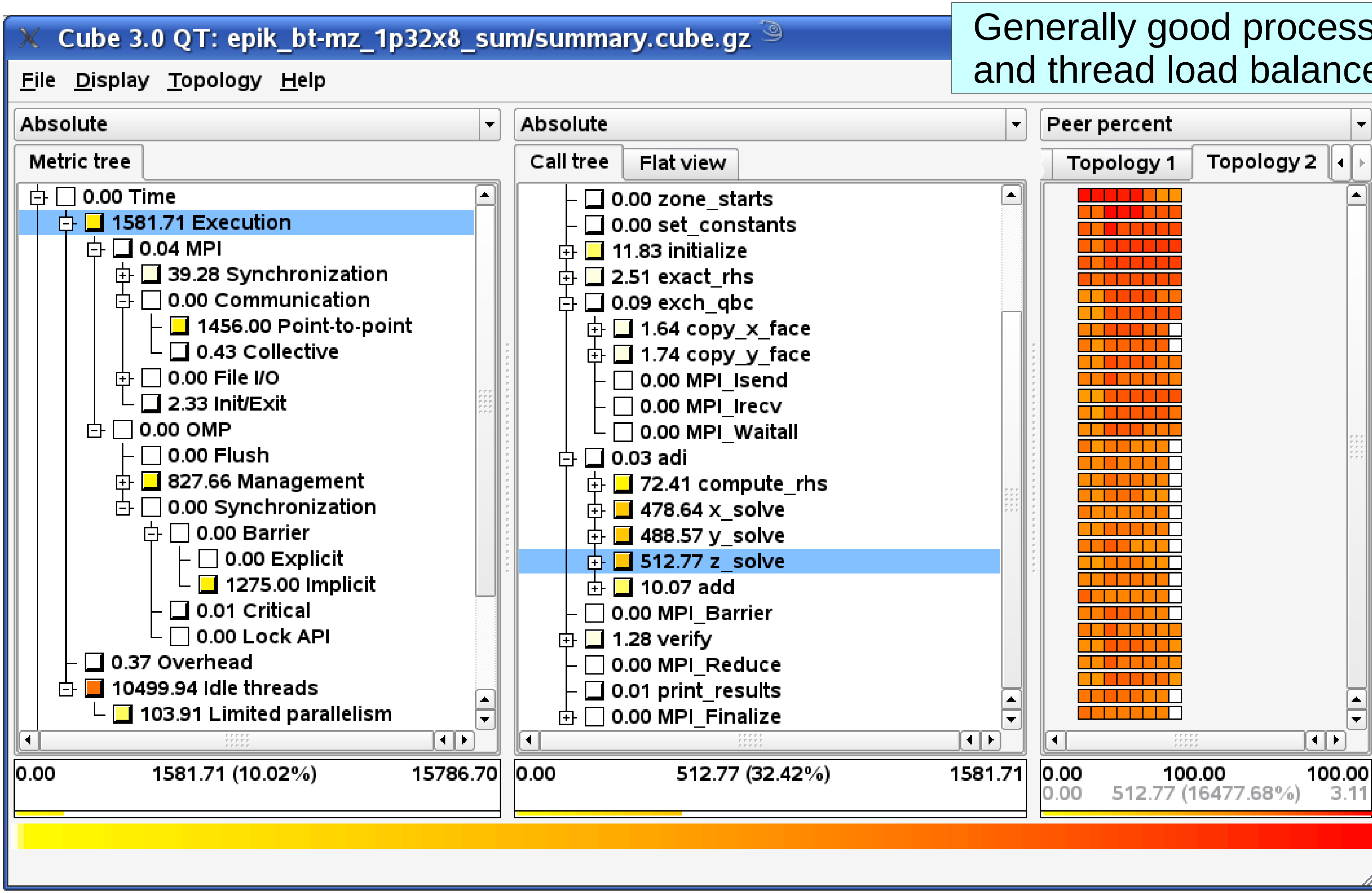


4x4 combined summary & trace analysis

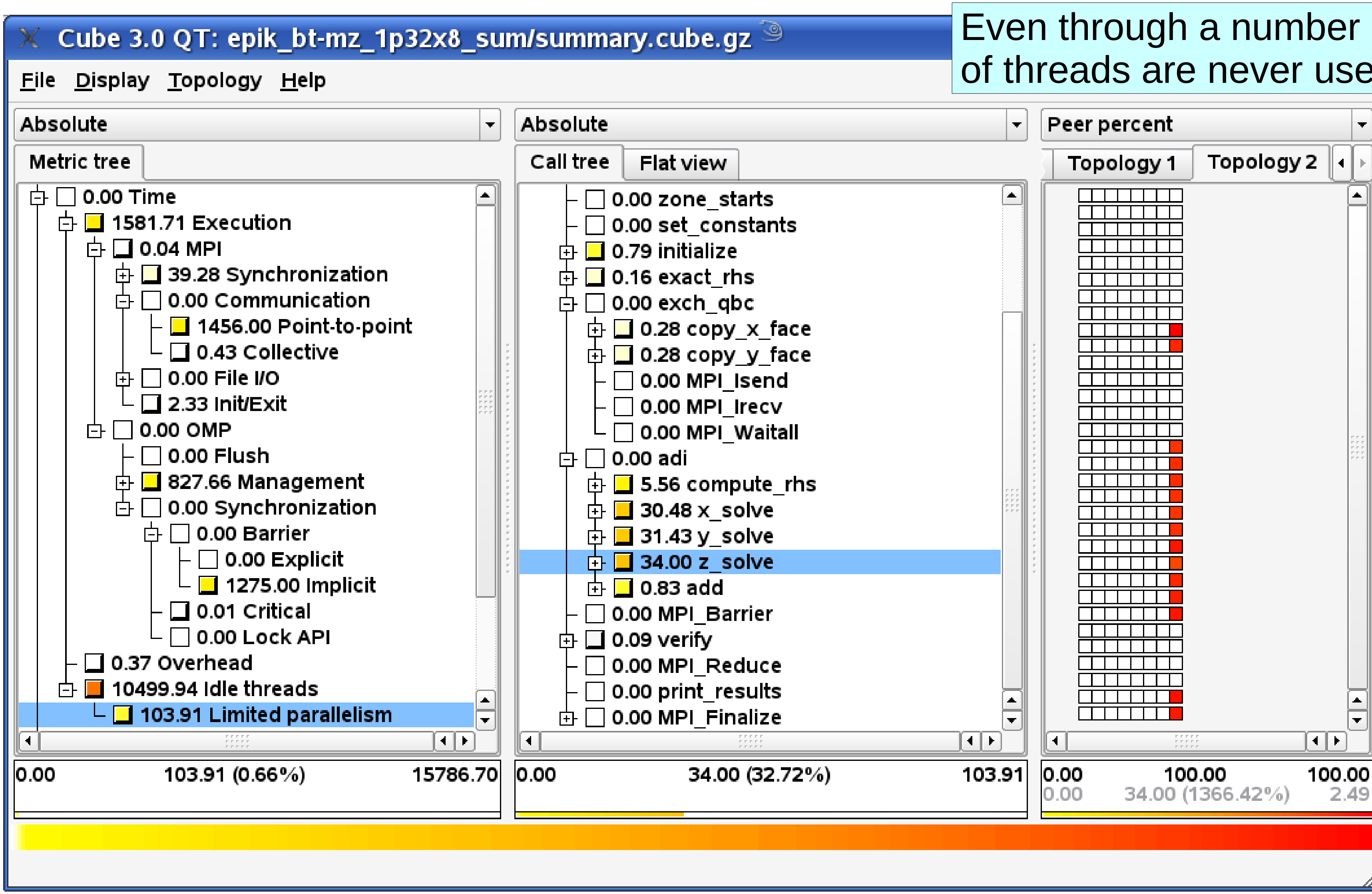


32x8 summary analysis: Excl. execution time

Generally good process and thread load balance

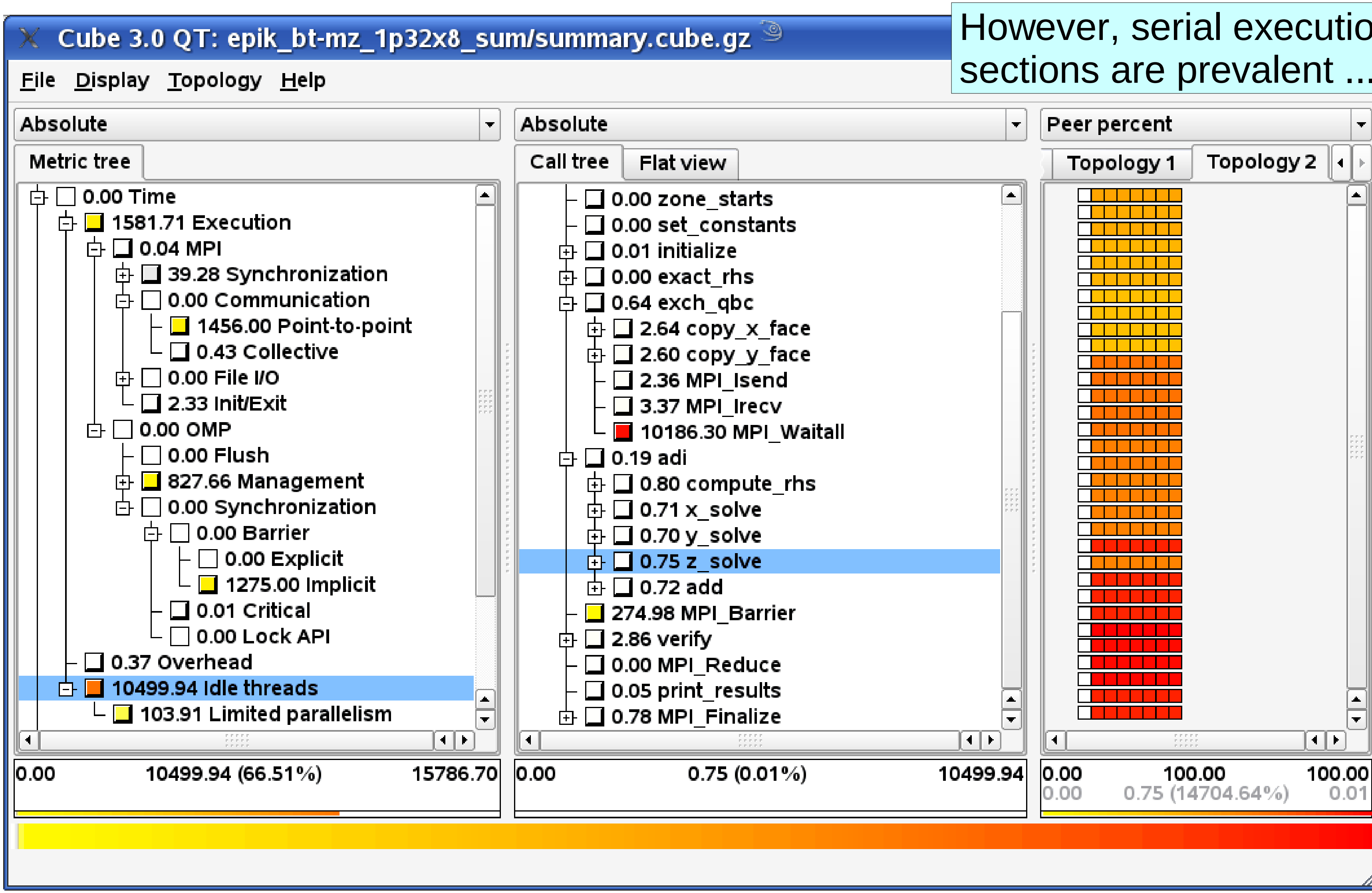


32x8 summary analysis: Limited parallelism



Even through a number of threads are never used

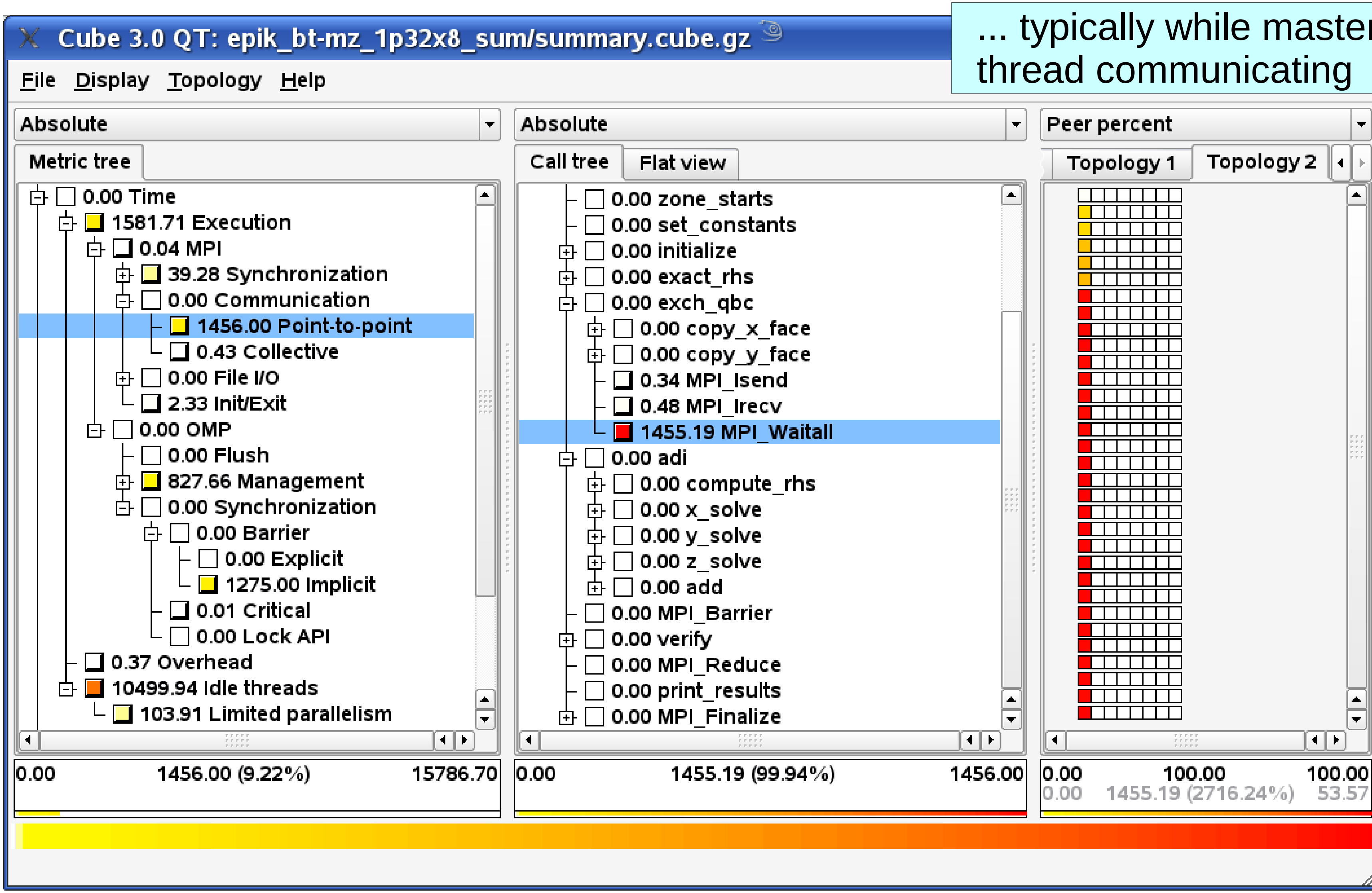
32x8 summary analysis: Idle threads time



However, serial execution sections are prevalent ...

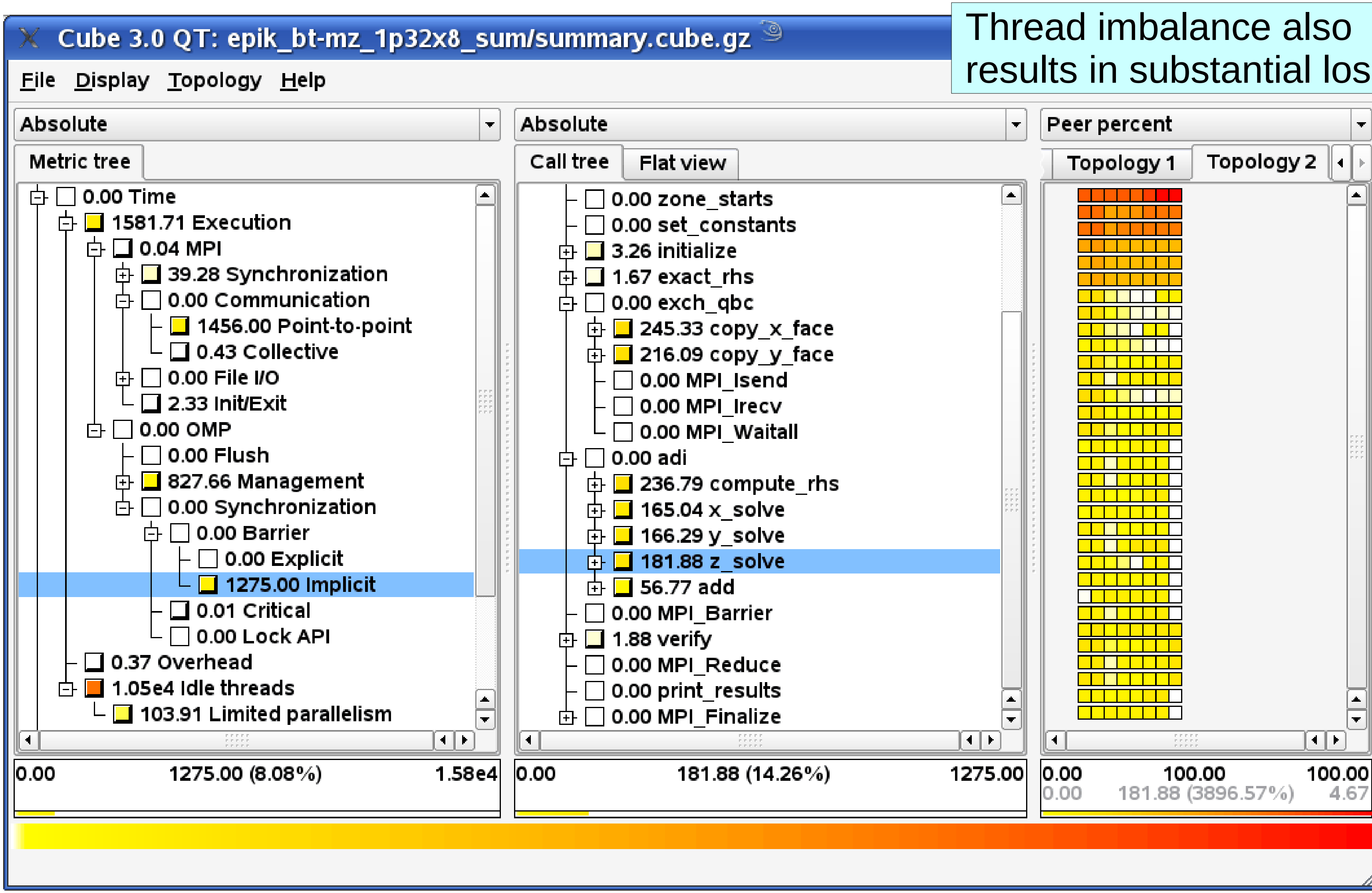
32x8 summary analysis: MPI communication time

... typically while master thread communicating



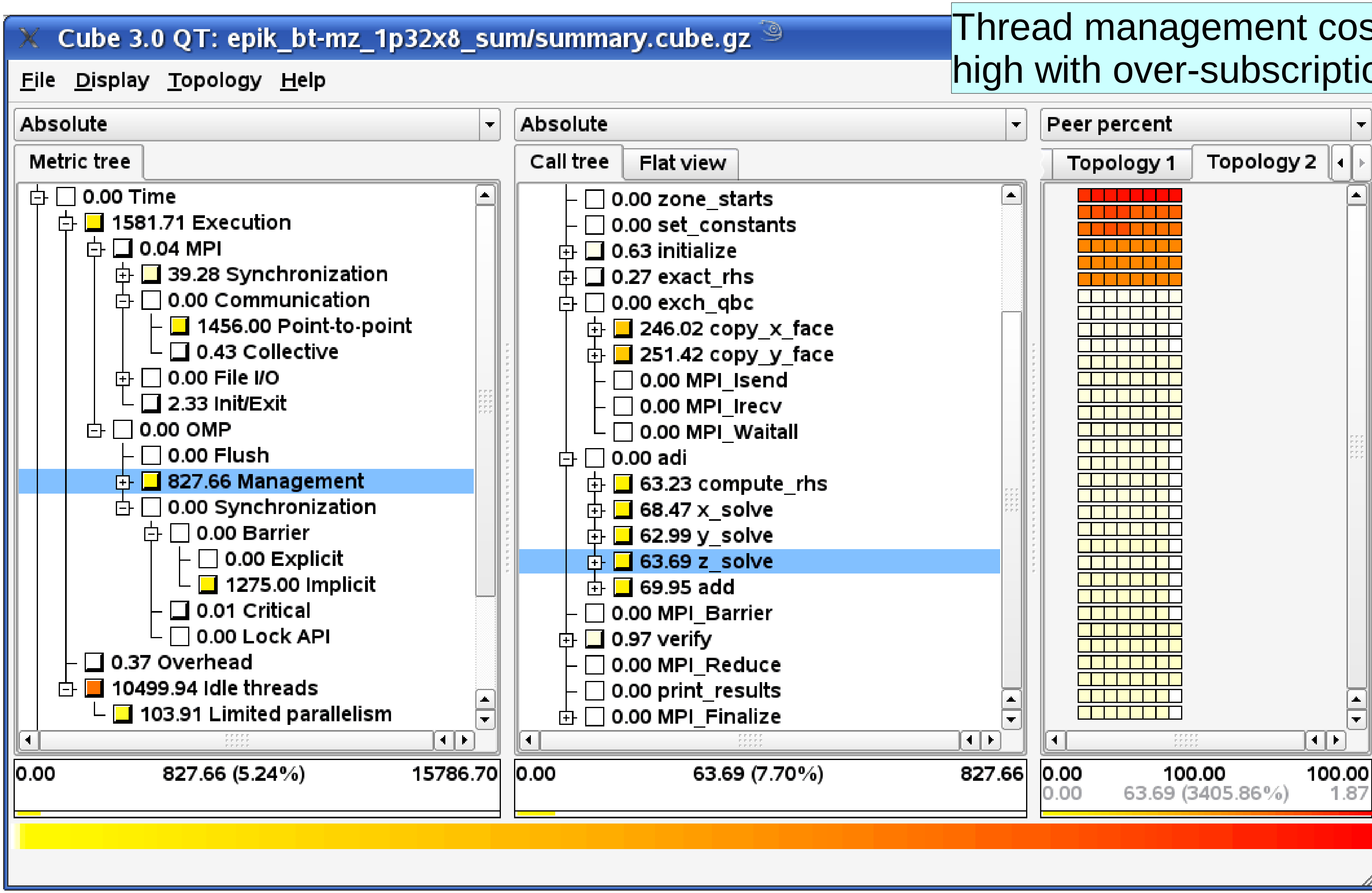
32x8 summary analysis: Implicit barrier time

Thread imbalance also results in substantial loss



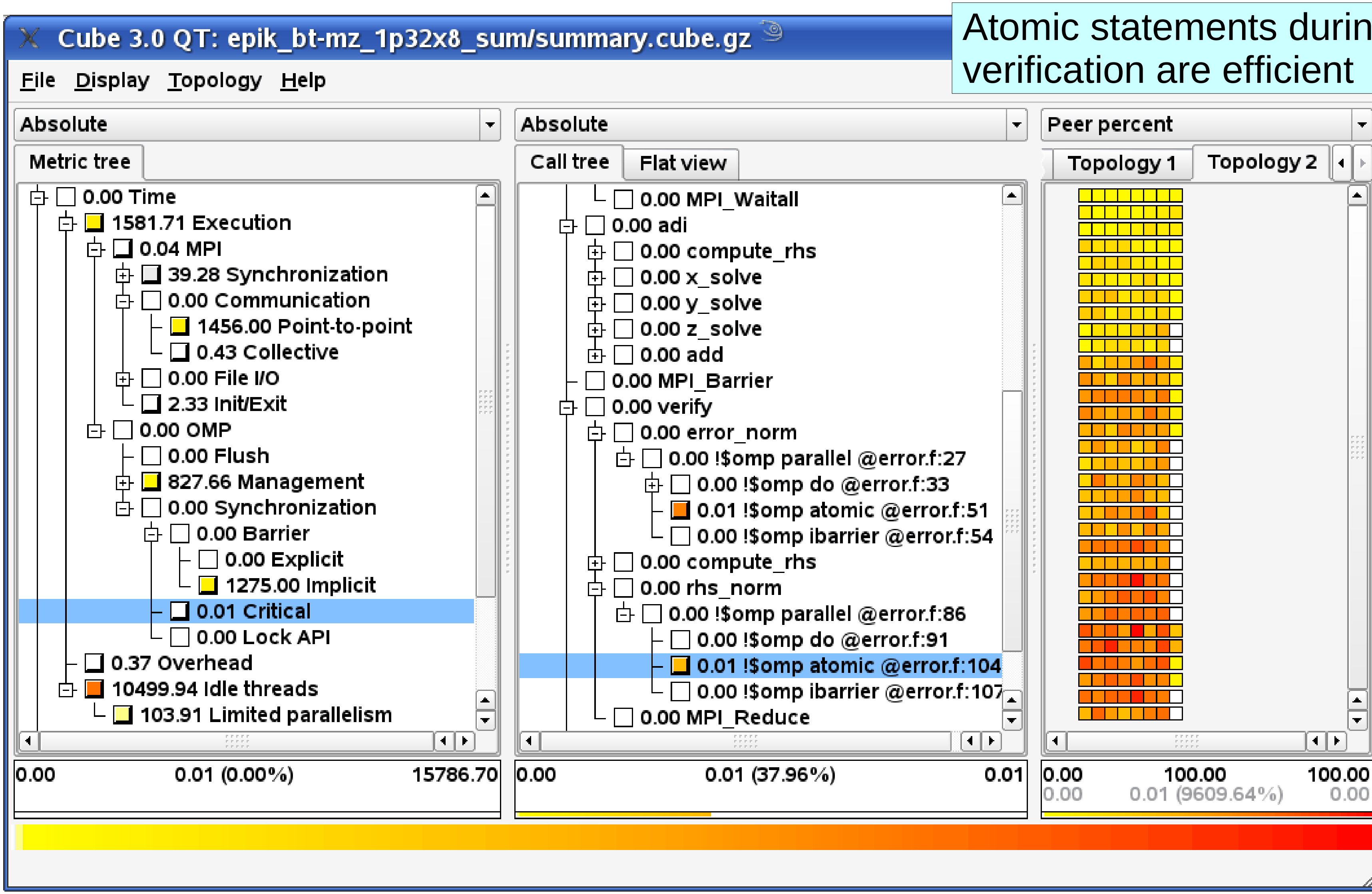
32x8 summary analysis: Thread management

Thread management cost high with over-subscription



32x8 summary analysis: Critical section time

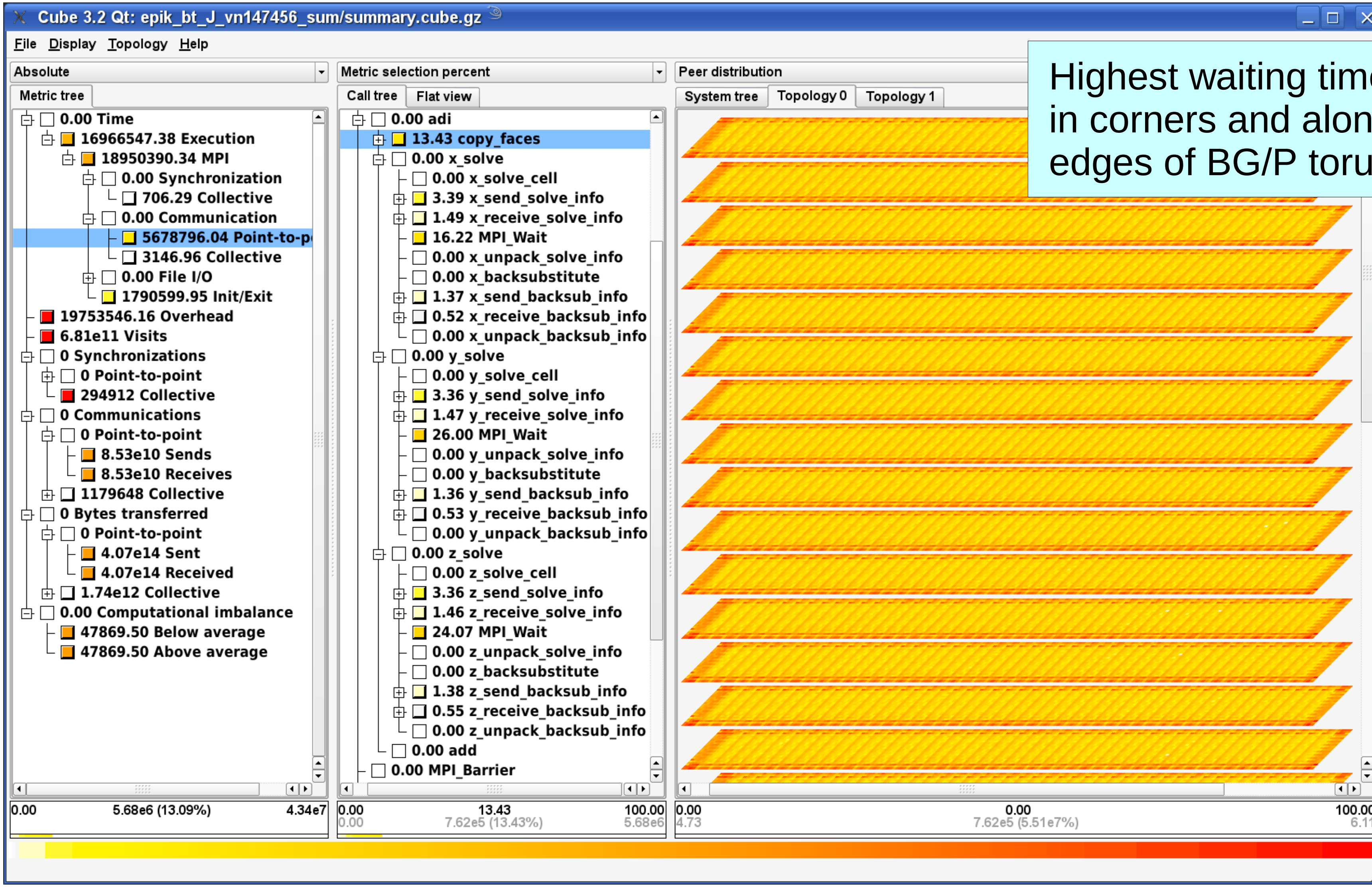
Atomic statements during verification are efficient



NPB-MPI-BT on BlueGene/P case study

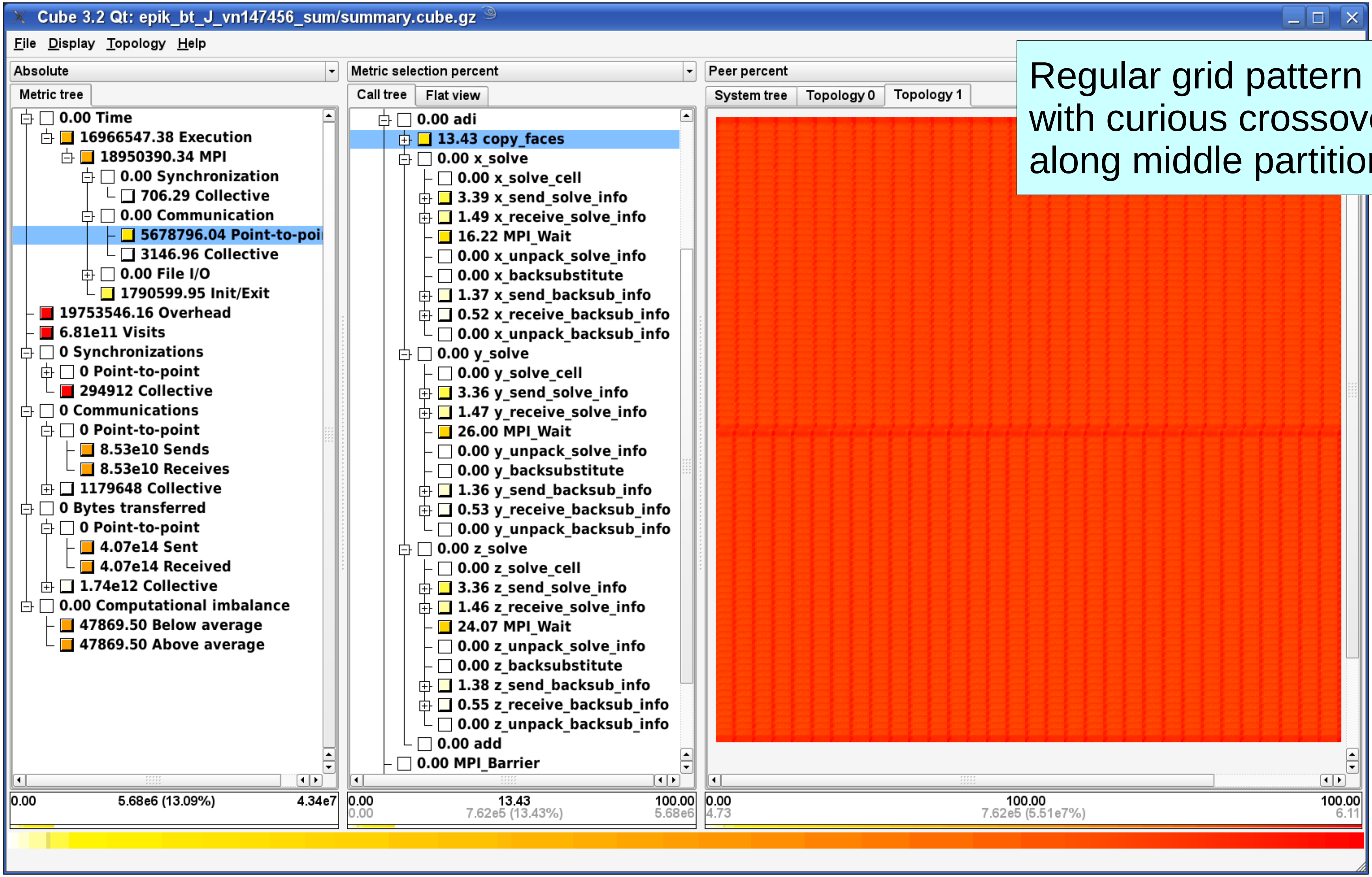
- 3D solution of unsteady, compressible Navier-Stokes eqs
  - NASA NAS parallel benchmark suite Block-Tridiagonal solver
  - series of ADI solve steps in X, Y & Z dimensions
  - ~9,500 lines (20 source modules), mostly Fortran77
- Run on IBM BlueGene/P in VN mode with 144k processes
  - Good scaling when problem size matched to architecture
    - 1536x1536x1536 gridpoints mapped onto 384x384 processes
  - Measurement collection took 53 minutes
  - 38% dilation for summarization measurement compared to uninstrumented execution (using 10 function filter)
  - MPI trace size would be 18.6TB
  - 25% of time in ADI is point-to-point communication time
    - 13% copy_faces, 23% x_solve, 33% y_solve, 31% z_solve
  - 128s for a single MPI_Comm_split during setup!

NPB-MPI-BT on jugene@144k summary analysis



Highest waiting times in corners and along edges of BG/P torus

NPB-MPI-BT on jugene@144k summary analysis



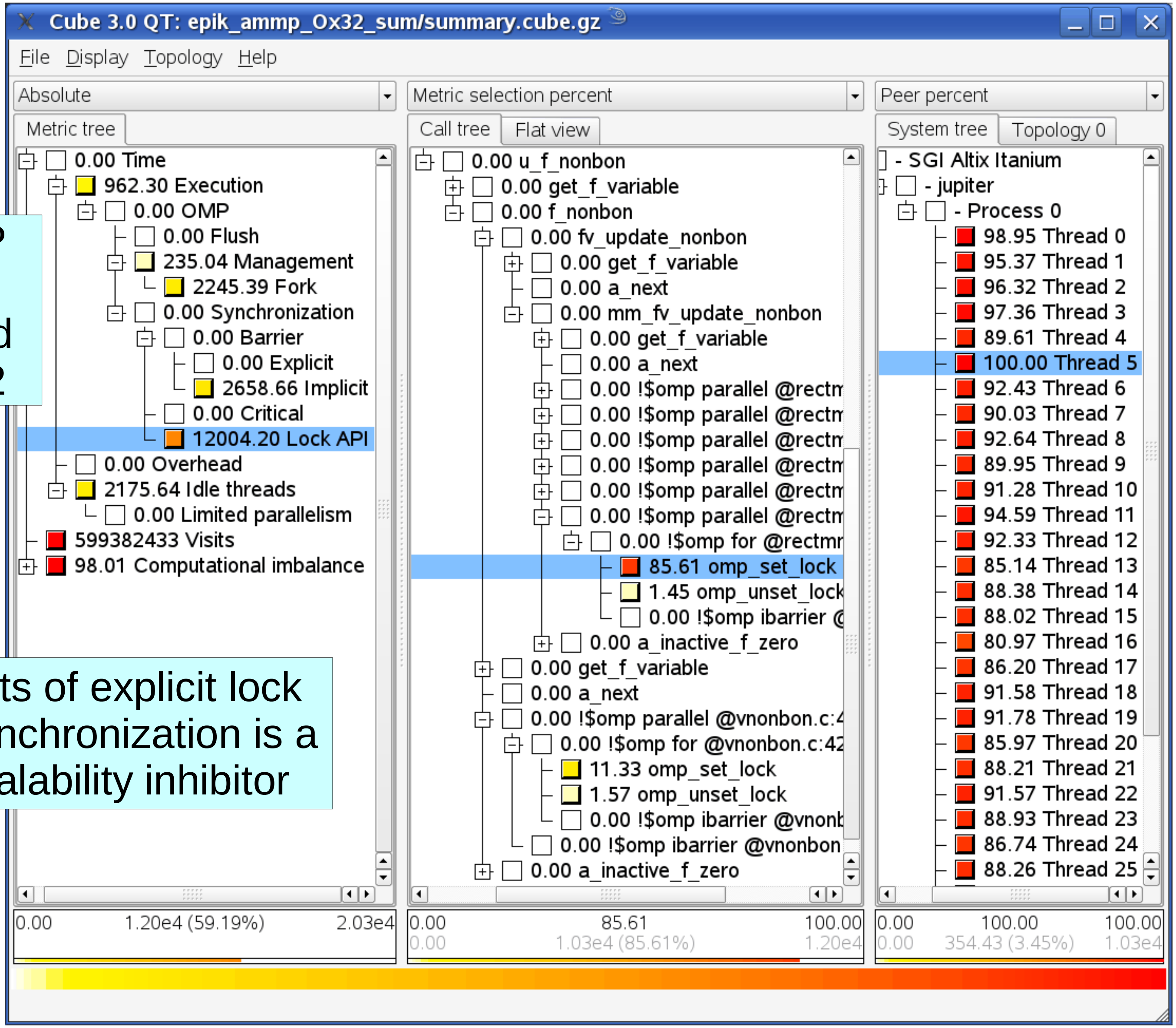
Regular grid pattern with curious crossover along middle partition

AMMP on Altix case study

- Molecular mechanics simulation
  - original version developed by Robert W. Harrison
- SPEC OMP benchmark parallel version
  - ~14,000 lines (in 28 source modules): 100% C
- Run with 32 threads on SGI Altix 4700 at TUD-ZIH
  - Built with Intel compilers
  - 333 simulation timesteps for 9,582 atoms
- Scalasca summary measurement
  - Minimal measurement dilation
  - 60% of total time lost in synchronization with lock API
  - 12% thread management overhead

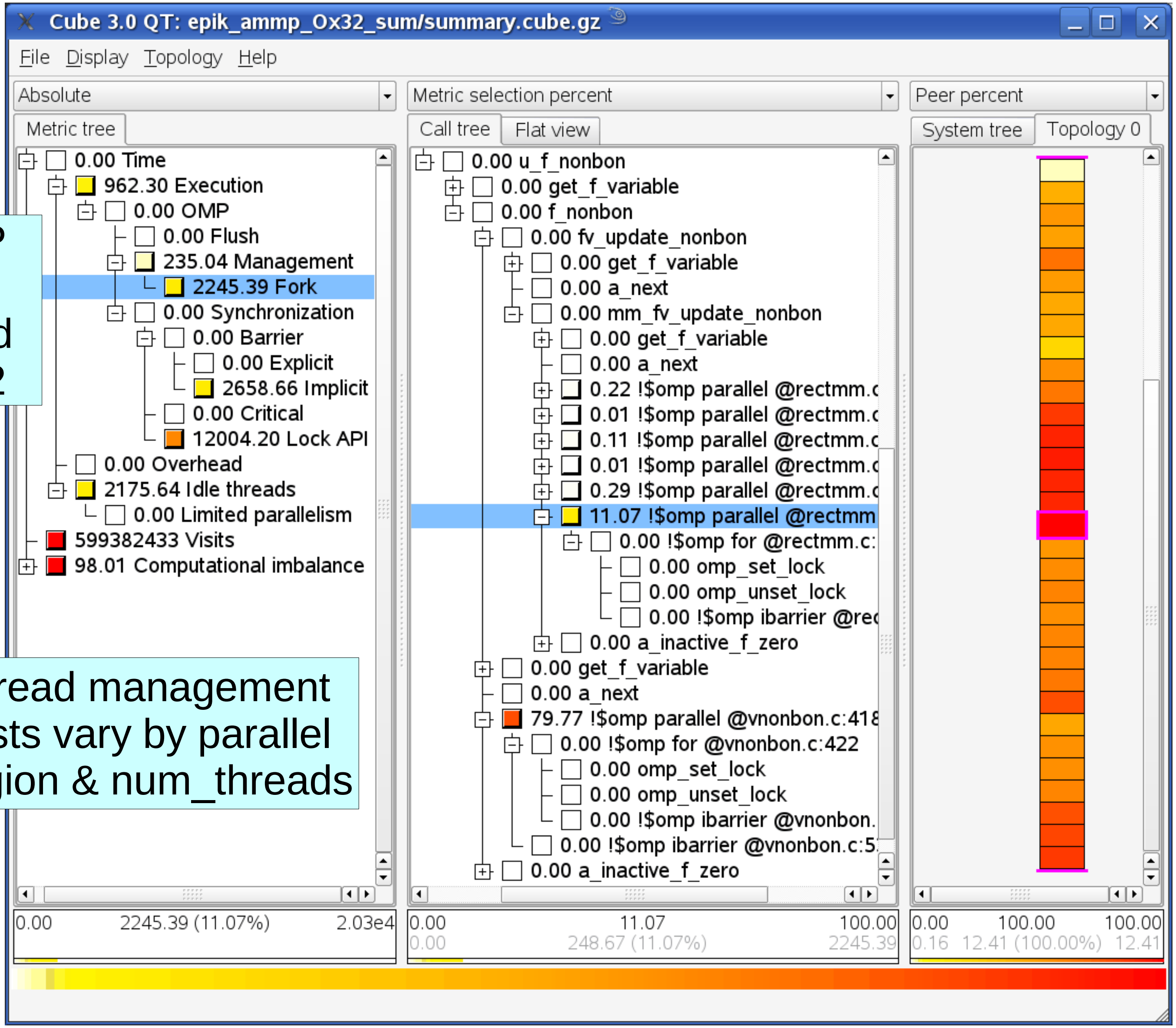
ammp on jupiter@32 OpenMP lock analysis

OpenMP metrics reworked with v1.2



Lots of explicit lock synchronization is a scalability inhibitor

ammp on jupiter@32 OpenMP fork analysis



OpenMP metrics reworked with v1.2

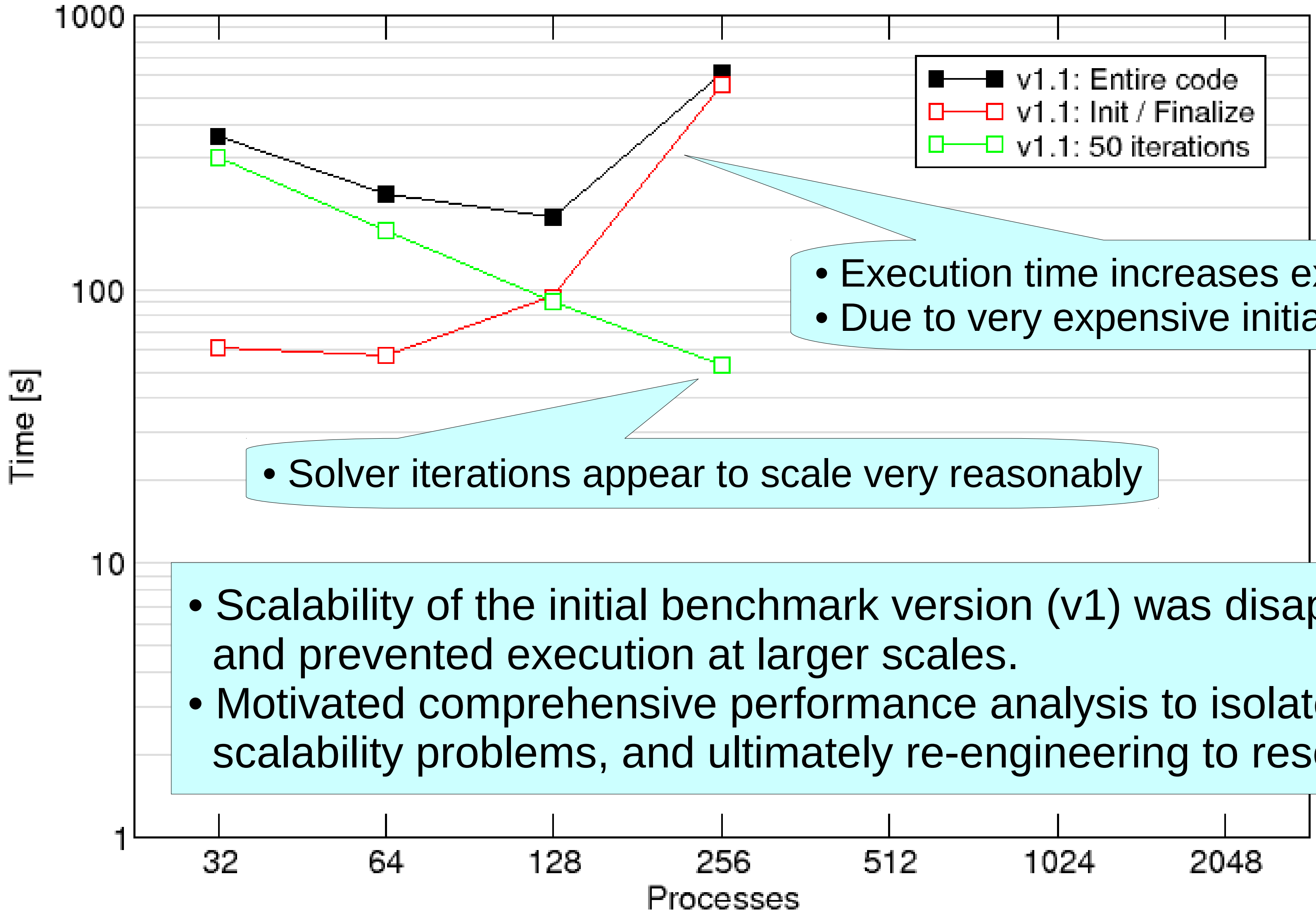
Thread management costs vary by parallel region & num_threads

GemsFDTD case study

- Computational electromagnetics solver
  - originates from KTH General ElectroMagnetics Solvers project
  - finite-difference time-domain method for Maxwell equations
- MPI parallel versions in SPEC MPI2007 benchmark suite
  - original ***v1.1*** (113.GemsFDTD) “medium” size
  - revised ***v2.0*** (145.lGemsFDTD) “large” size
  - built with PGI 9.0.4 Fortran90 compiler (21k lines of code)
    - typical benchmark optimization: `-fastsse -O3 -Mipa=fast,inline`
- Both run on 'hector' Cray XT4 at EPCC
  - using “ltrain” dataset from v2.0 benchmark (50 timesteps)
  - default Scalasca instrumentation for measurements
    - 9 of 90 application user-level source routines specified in filter determined by scoring initial summary experiment

Proc. 10th Int'l Workshop PARA (Reykjavík, Iceland, 2010)

GemsFDTD v1 scalability on Cray XT4



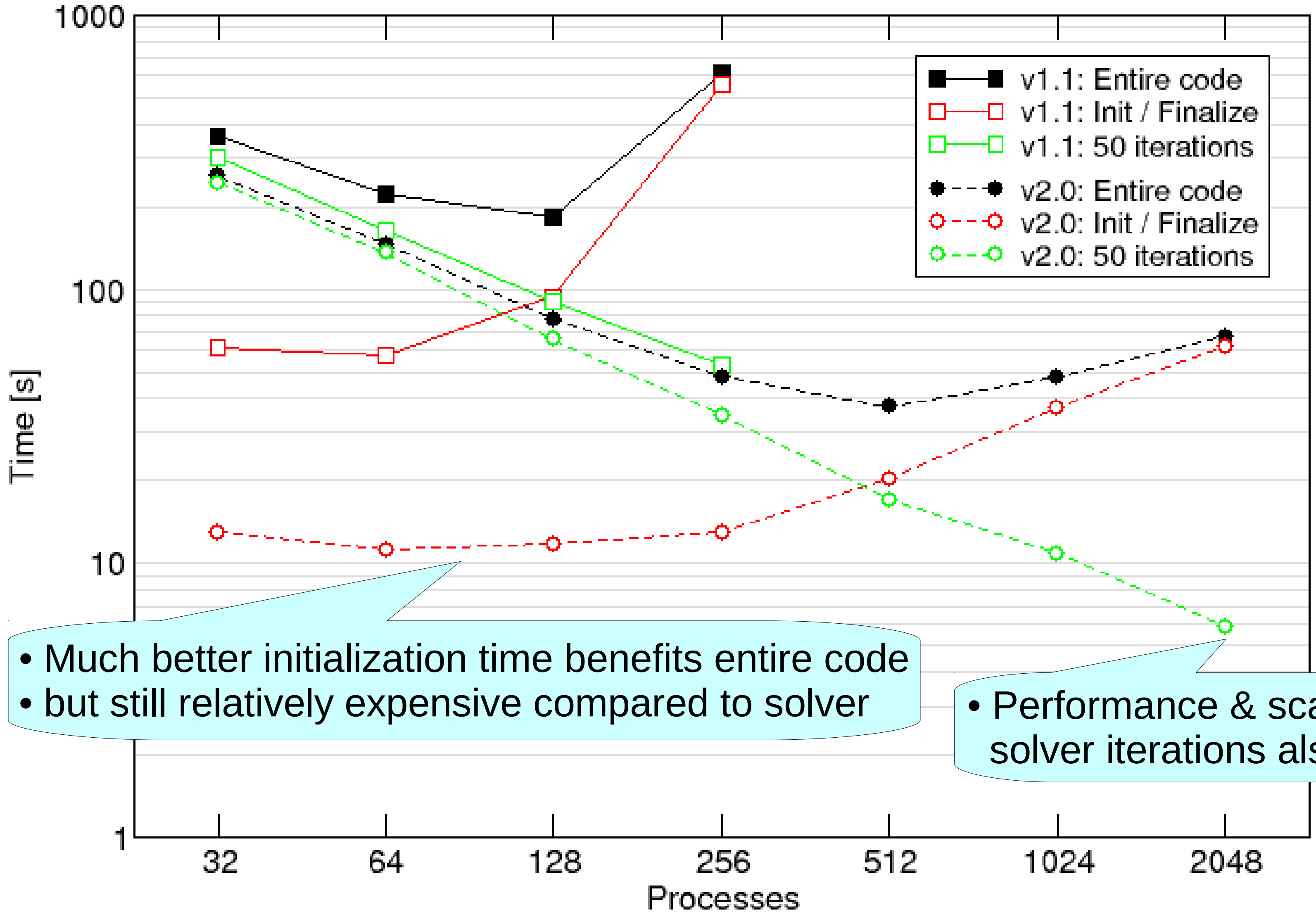
['ltrain' runs on CrayXT4 HECToR]

- Execution time increases exponentially
- Due to very expensive initialization

- Solver iterations appear to scale very reasonably

- Scalability of the initial benchmark version (v1) was disappointing and prevented execution at larger scales.
- Motivated comprehensive performance analysis to isolate scalability problems, and ultimately re-engineering to resolve them.

GemsFDTD v1 & v2 scalability on Cray XT4

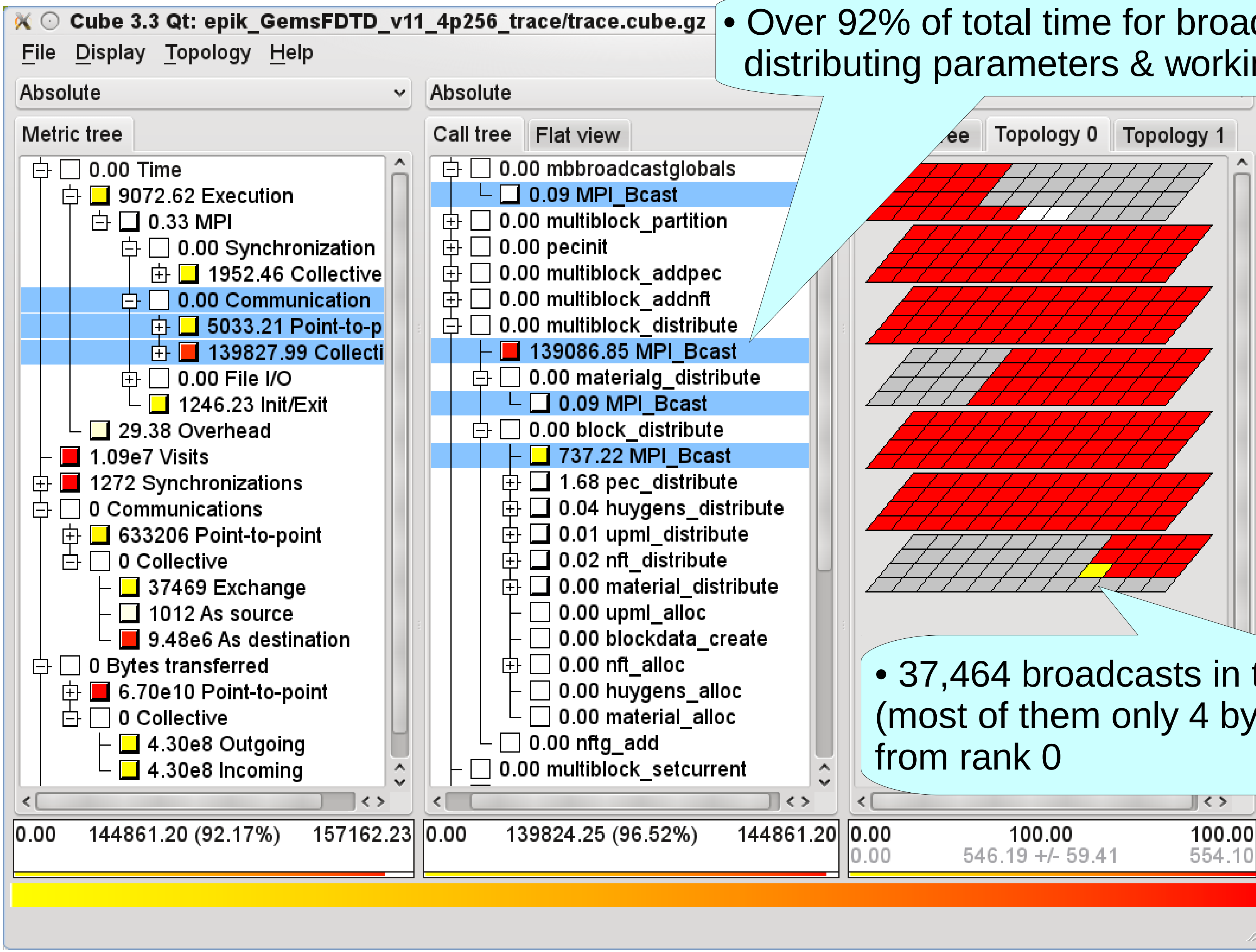


['ltrain' runs on CrayXT4 HECToR]

- Much better initialization time benefits entire code
- but still relatively expensive compared to solver

- Performance & scalability of solver iterations also improved

Time for initialization broadcasts (*v1.1*)

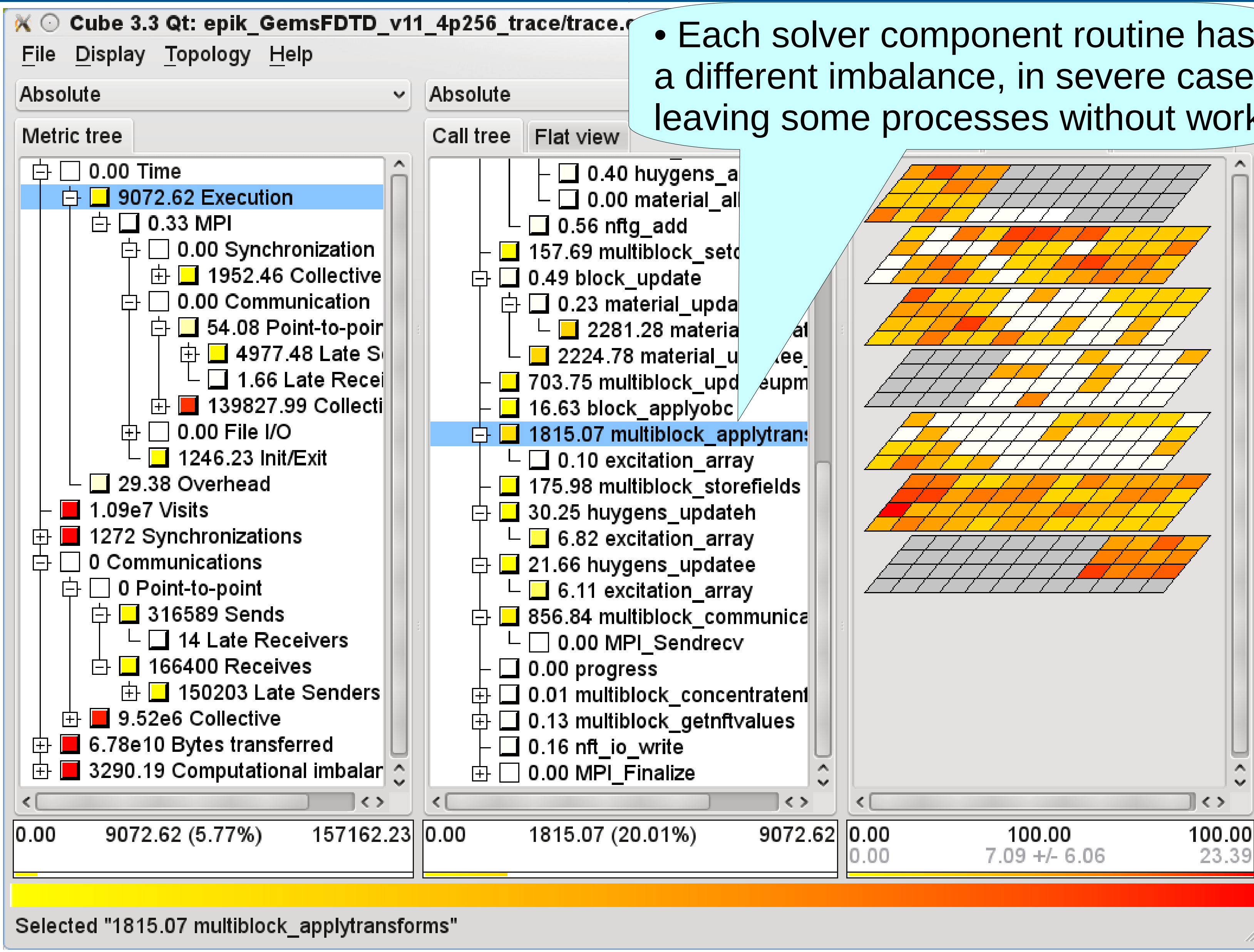


- Over 92% of total time for broadcasts distributing parameters & working set

- 37,464 broadcasts in total (most of them only 4 byes) from rank 0

Computation time in solver transforms (*v1.1*)

- Each solver component routine has a different imbalance, in severe cases leaving some processes without work



GemsFDTD case study

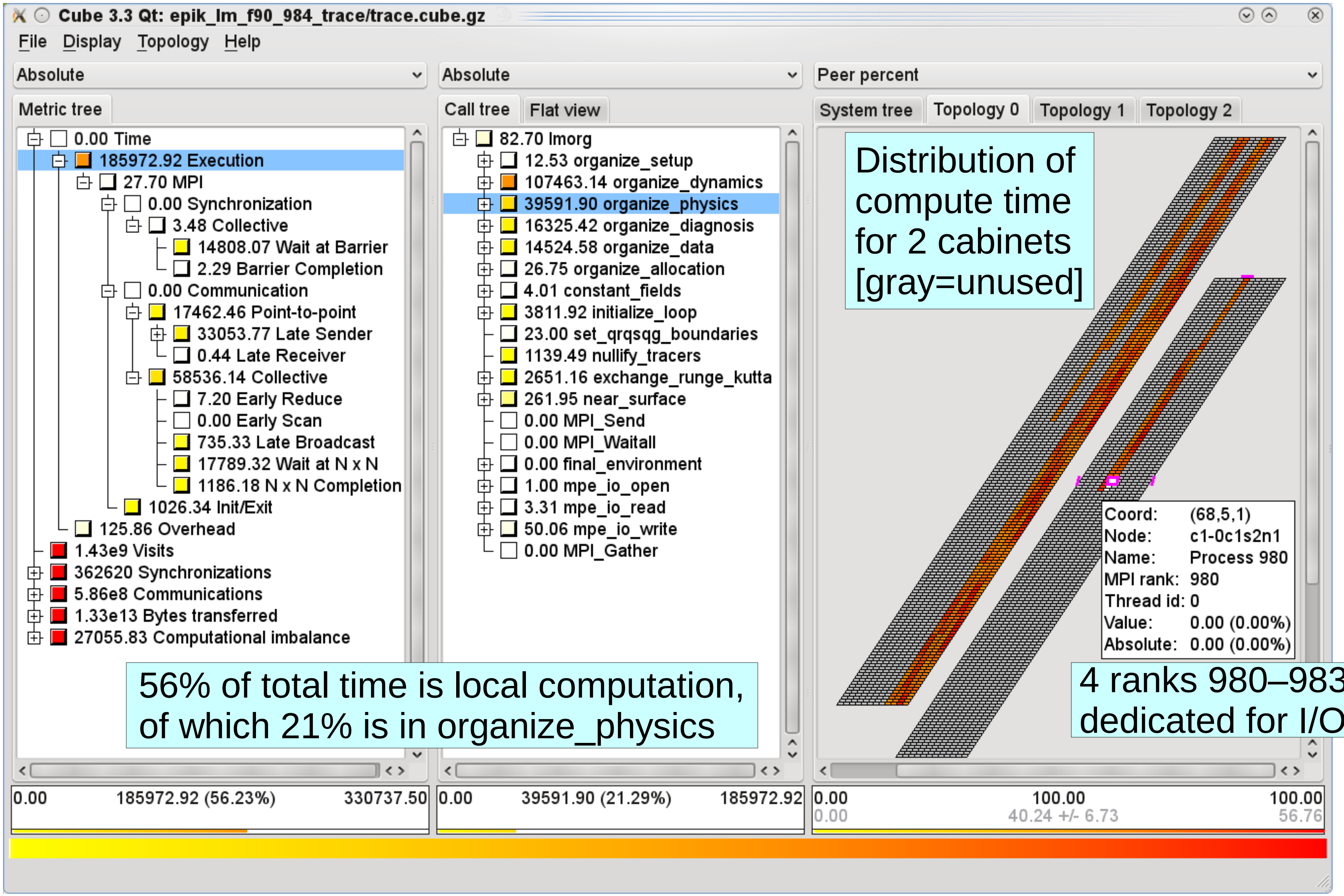
- Initialization originally dominated by numerous broadcasts and expensive serial multiblock partition by rank 0
  - Re-engineered implementation of scalable partition routine, aggregation of multiple data values into larger messages, and postpones allocations until all block information in broadcast
    - Initialization time reduced to less than 2% of total time
- Solver iterations using blocking communication manifests as *Late Sender* waiting originating from imbalance in local computation time (due to different computations)
  - Re-engineered implementation uses non-blocking comms and re-uses communication pattern used to exchange blocks (as well as 2 of 256 processes unintentionally idled throughout)
    - computation & communication time both improved more than 25%
- Scalabilty improved from 128 processes to more than 1024

COSMO-7/XE6 case study

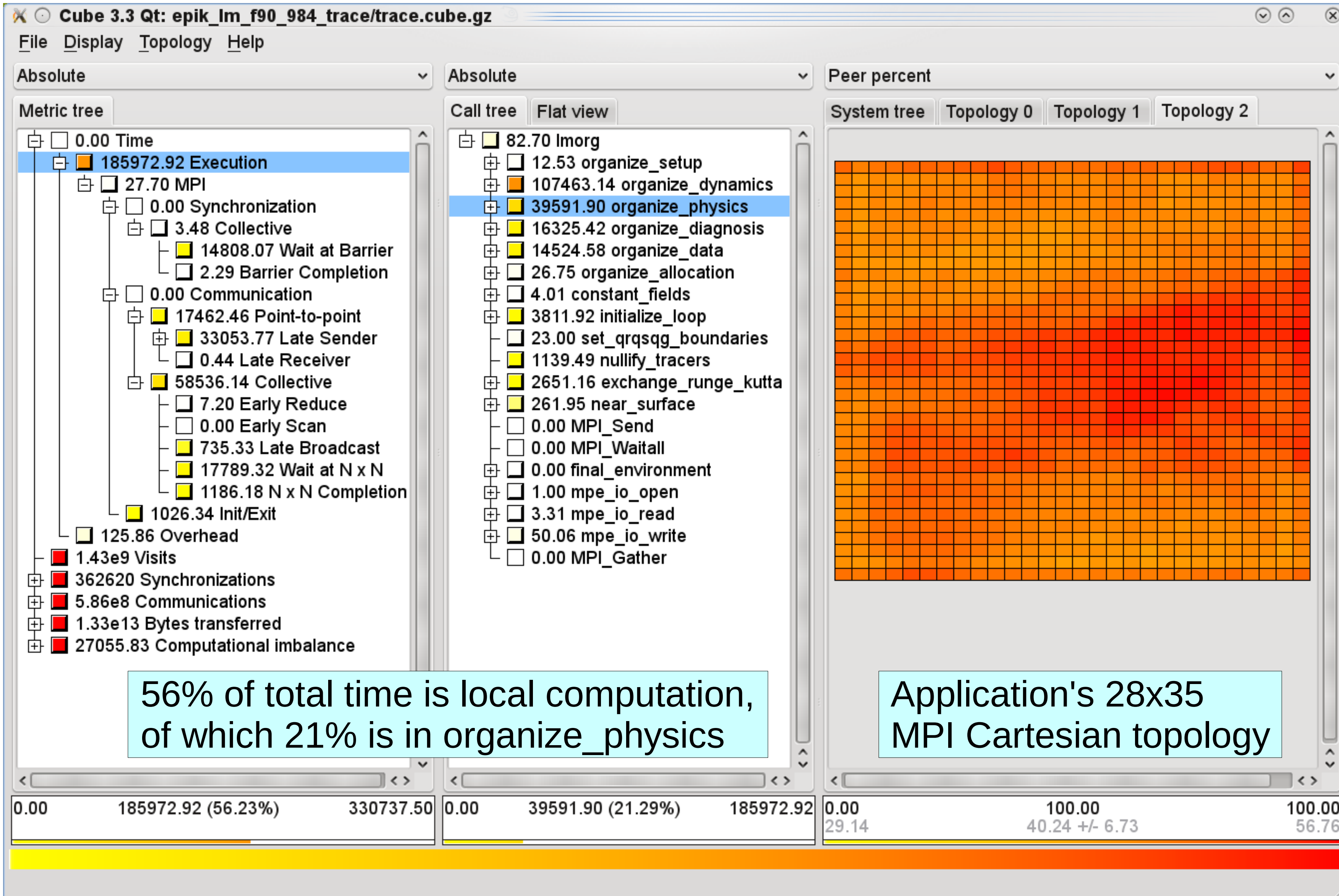
- Regional climate and weather model
  - developed by Consortium for Small-scale Modeling (COSMO)
    - DWD, MeteoSwiss and others
  - non-hydrostatic limited-area atmospheric model (6.6km grid)
- MPI parallel version 4.12 (Jan-2011)
  - built with PGI 10.9 Fortran90 compiler (222k lines of code)
- MeteoSwiss operational 24-hour forecast of 06-Dec-2010
  - Western Europe 393x338x60 resolution, 1440 timesteps
- Run with 984 processes on 'palu' Cray XE6 at CSCS
  - 28x35 compute grid + 4 dedicated I/O processes
  - used 41 Opteron compute nodes each with 24 cores
  - Scalasca trace measurement with 19 of 178 routines filtered
  - 44GB trace written in 23s and analyzed in 82s

Courtesy of Oliver Fuhrer (MeteoSwiss) & CSCS

COSMO/XE6 physics computation time



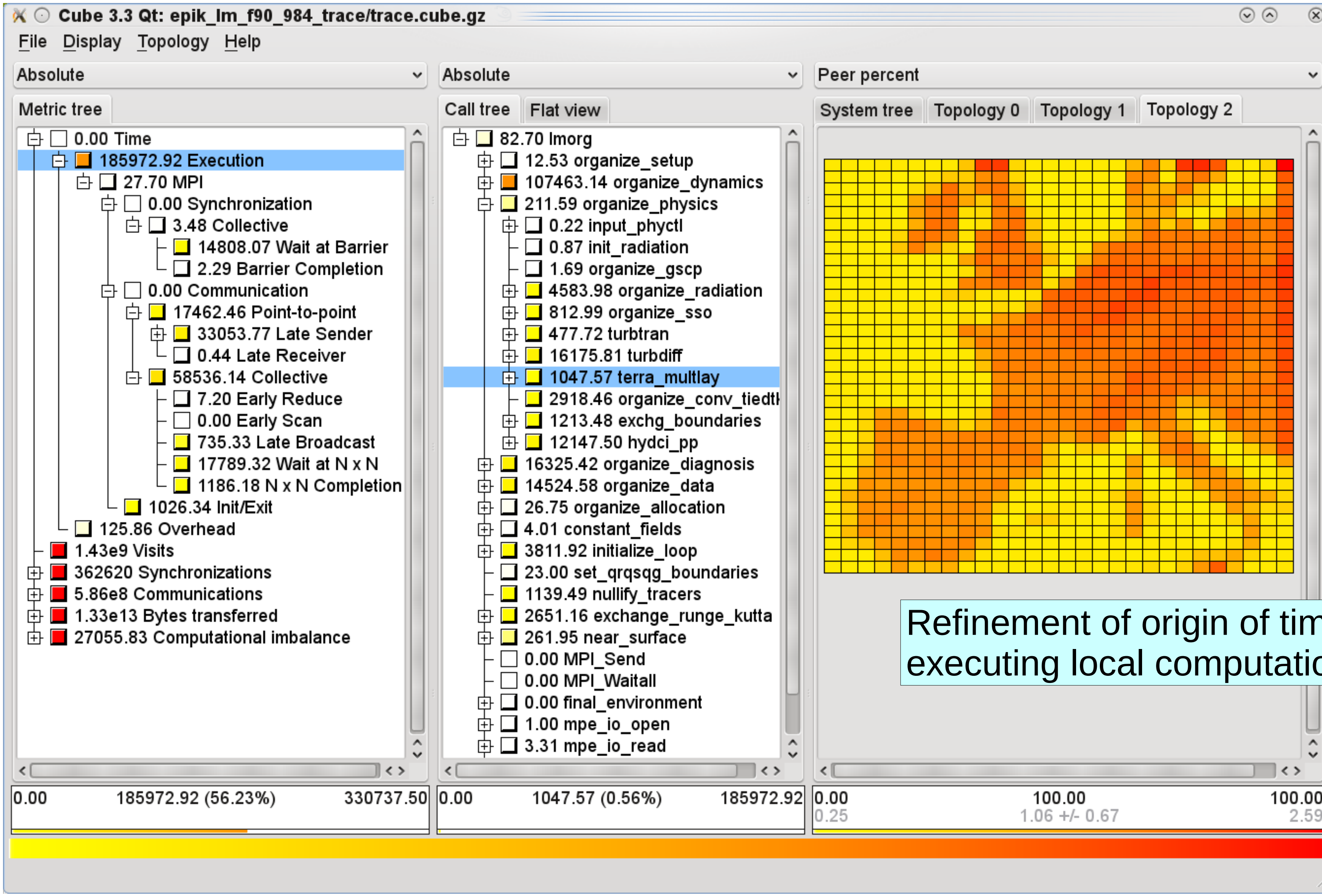
COSMO/XE6 physics computation time



56% of total time is local computation, of which 21% is in organize_physics

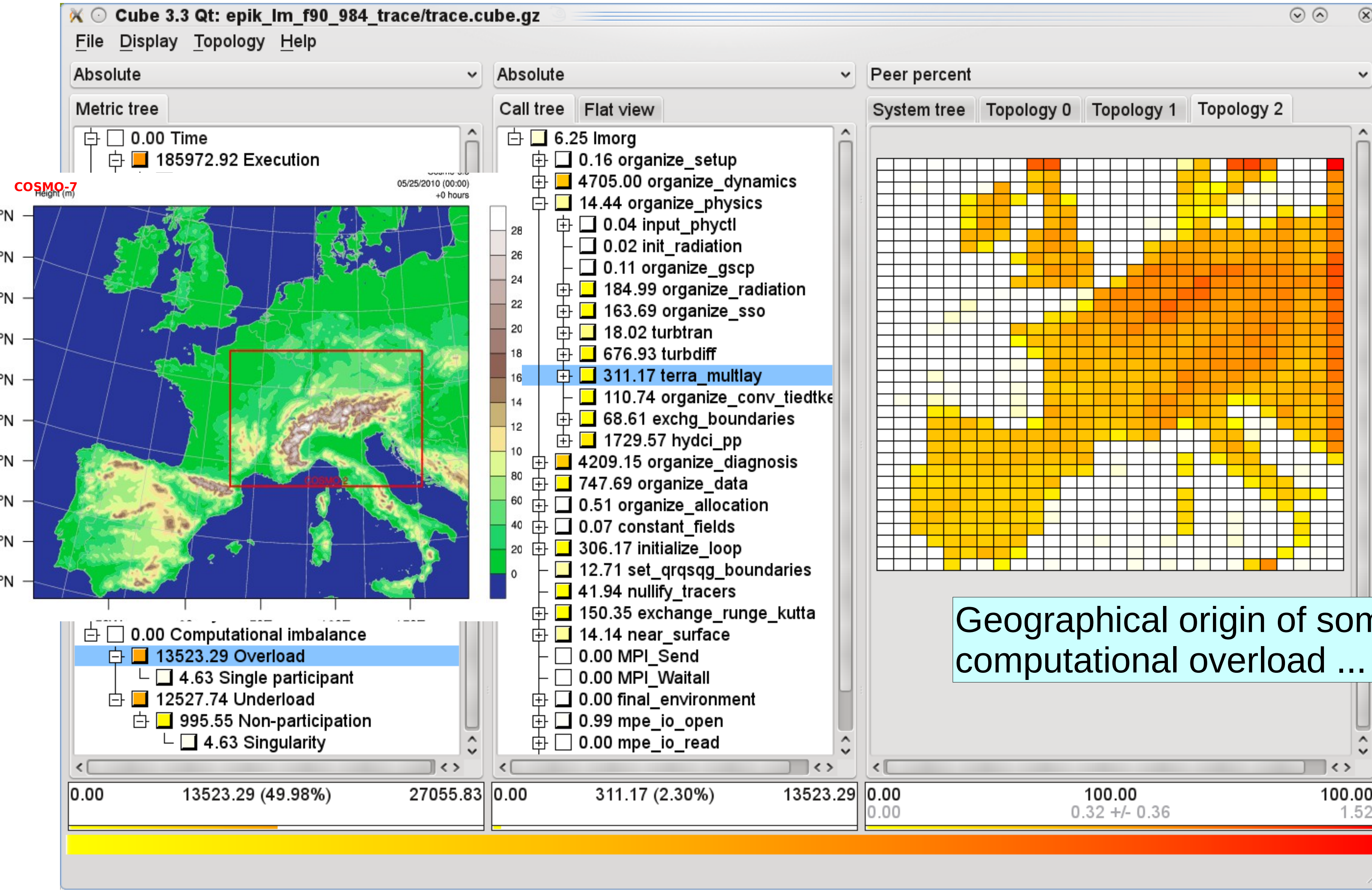
Application's 28x35 MPI Cartesian topology

COSMO/XE6 physics computation imbalance



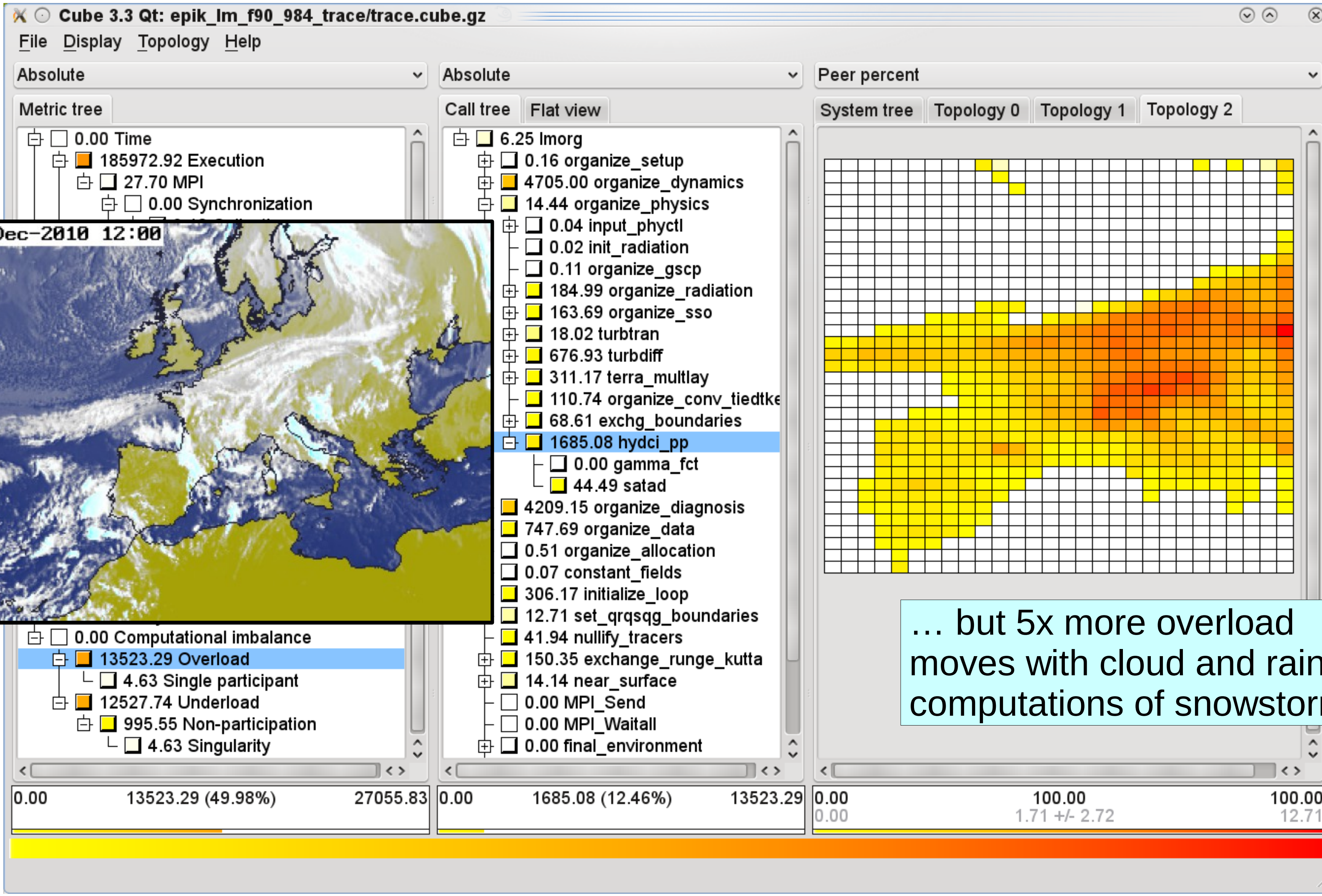
Refinement of origin of time executing local computation

COSMO/XE6 computational overload (geo)



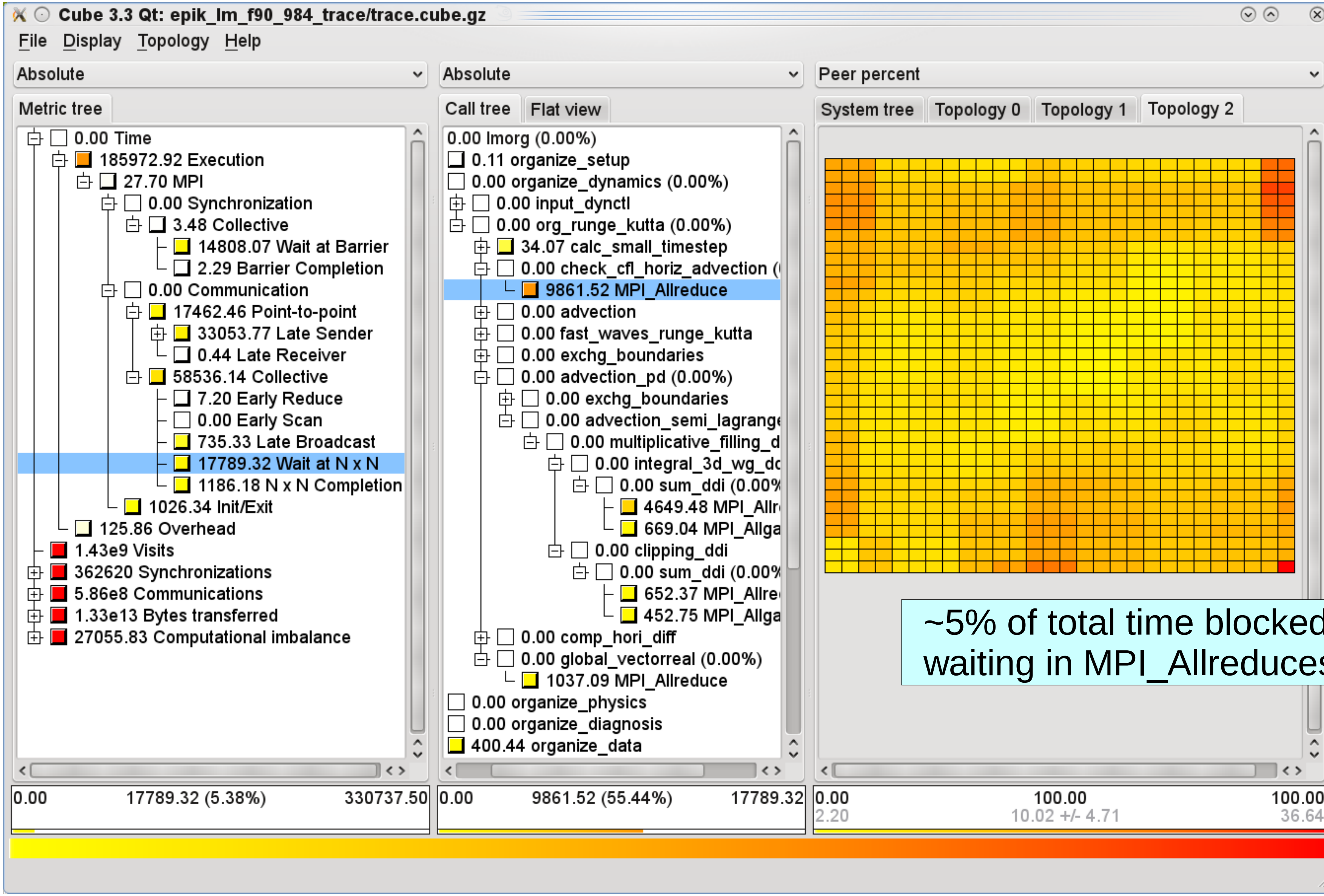
Geographical origin of some computational overload ...

COSMO/XE6 computational overload (hydro)



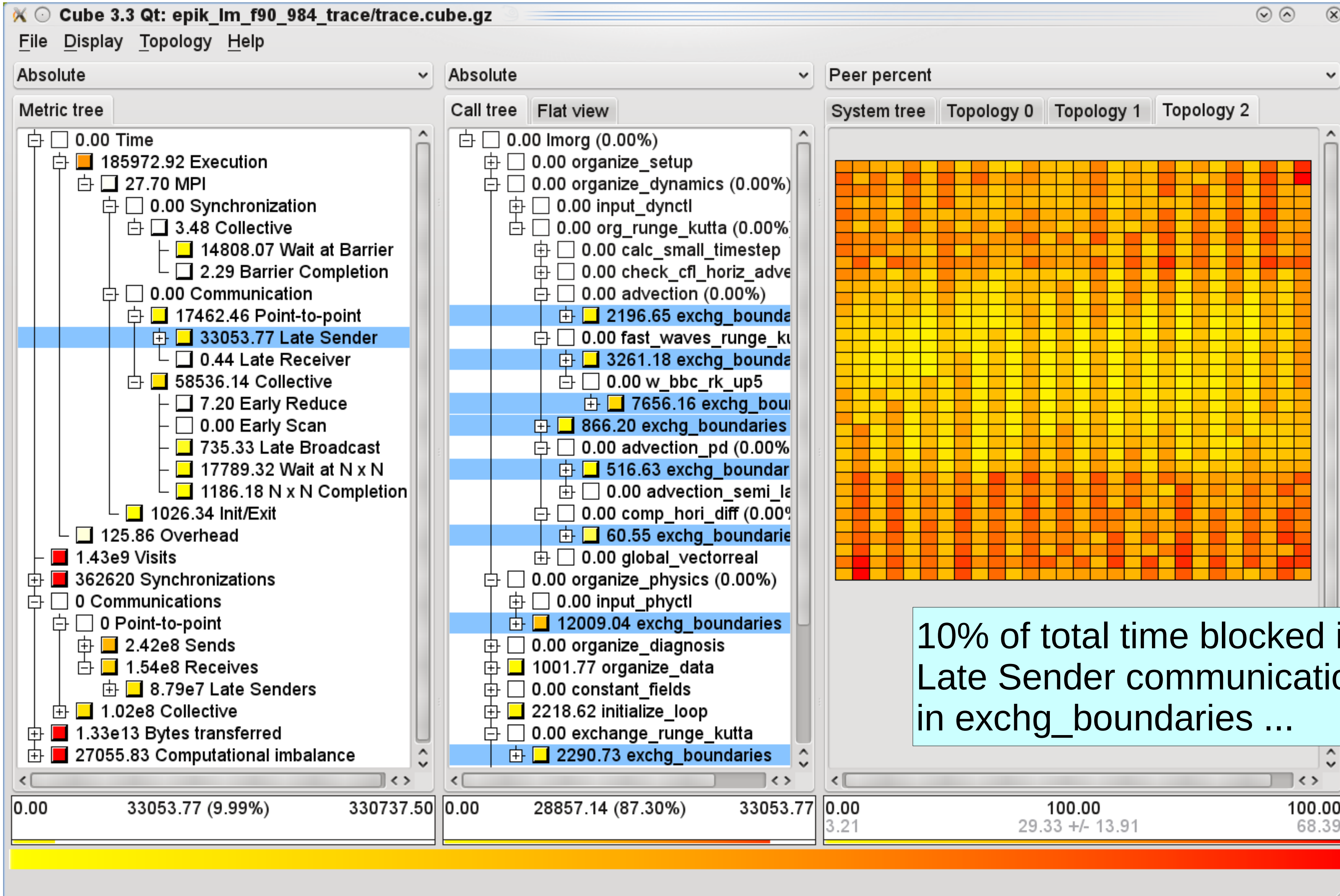
… but 5x more overload moves with cloud and rain computations of snowstorm

COSMO/XE6 collective wait at N x N time



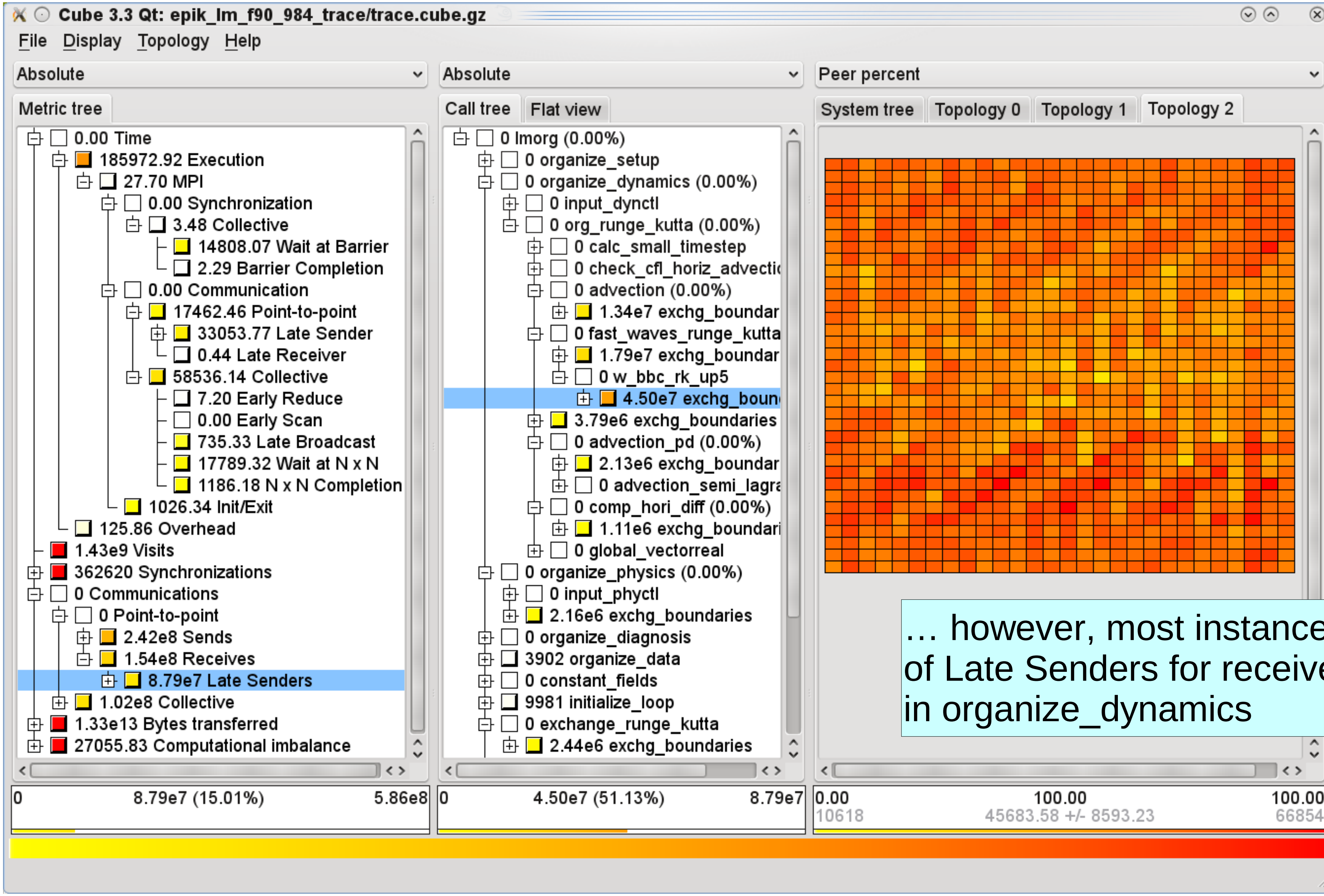
~5% of total time blocked waiting in MPI_Allreduces

COSMO/XE6 late sender waiting time



10% of total time blocked in Late Sender communication in exchg_boundaries ...

COSMO/XE6 late sender communications



… however, most instances of Late Senders for receives in organize_dynamics

COSMO-7/XE6 performance review

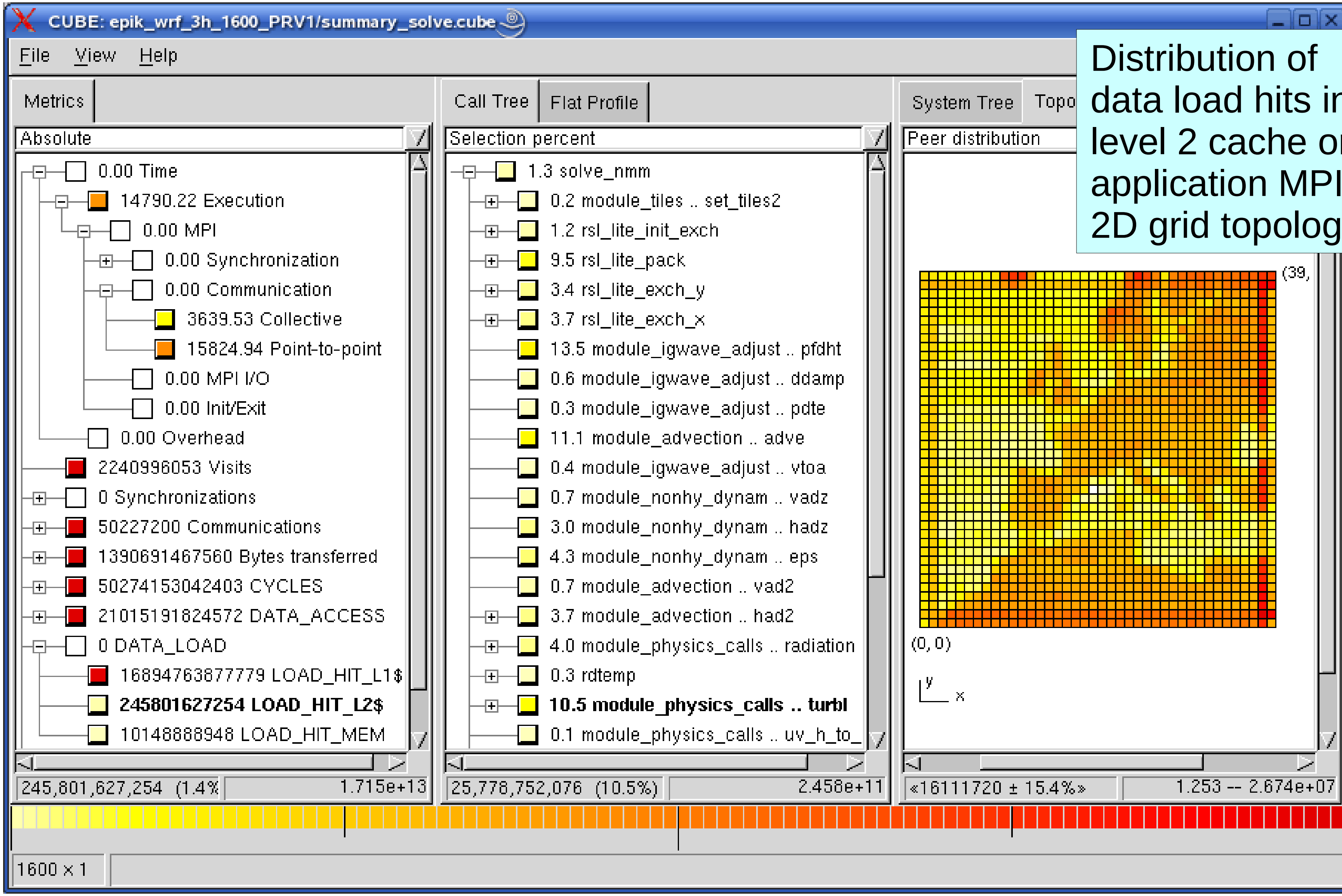
- 56% of total time in local computation
  - 32% in dynamics which is quite well balanced (11% std.dev)
  - 12% in physics is rather less well balanced (17% std.dev)
  - much of the imbalance is inherently physical/geographical
- 44% of total time in MPI
  - 5% collective synchronization (92% output_data)
  - 24% collective communication
    - 14% for MPI_Gather operations in output_data
    - 5% “Wait at NxN” mostly in dynamics check_cfl_horiz_advection
  - 15% point-to-point communication (91% exchg_boundaries)
    - 10% “Late Sender” time (44% dynamics, 36% physics)
    - 36% of receives are for “Late Senders” (95% in dynamics)
- Communication associated with file I/O was a major factor
  - the 4 dedicated I/O processes idle 95% of the time

WRF/MareNostrum case study

- Numerical weather prediction
  - public domain code developed by US NOAA
  - flexible, state-of-the-art atmospheric simulation
  - Non-hydrostatic Mesoscale Model (NMM)
- MPI parallel version 2.1.2 (Jan-2006)
  - >315,000 lines (in 480 source modules): 75% Fortran, 25% C
- Eur-12km dataset configuration
  - 3-hour forecast (360 timesteps) with checkpointing disabled
- Run with 1600 processes on MareNostrum
  - IBM BladeCenter cluster at BSC
- Scalasca summary and trace measurements
  - 15% measurement dilation with 8 hardware counters
  - 23GB trace analysis in 5 mins

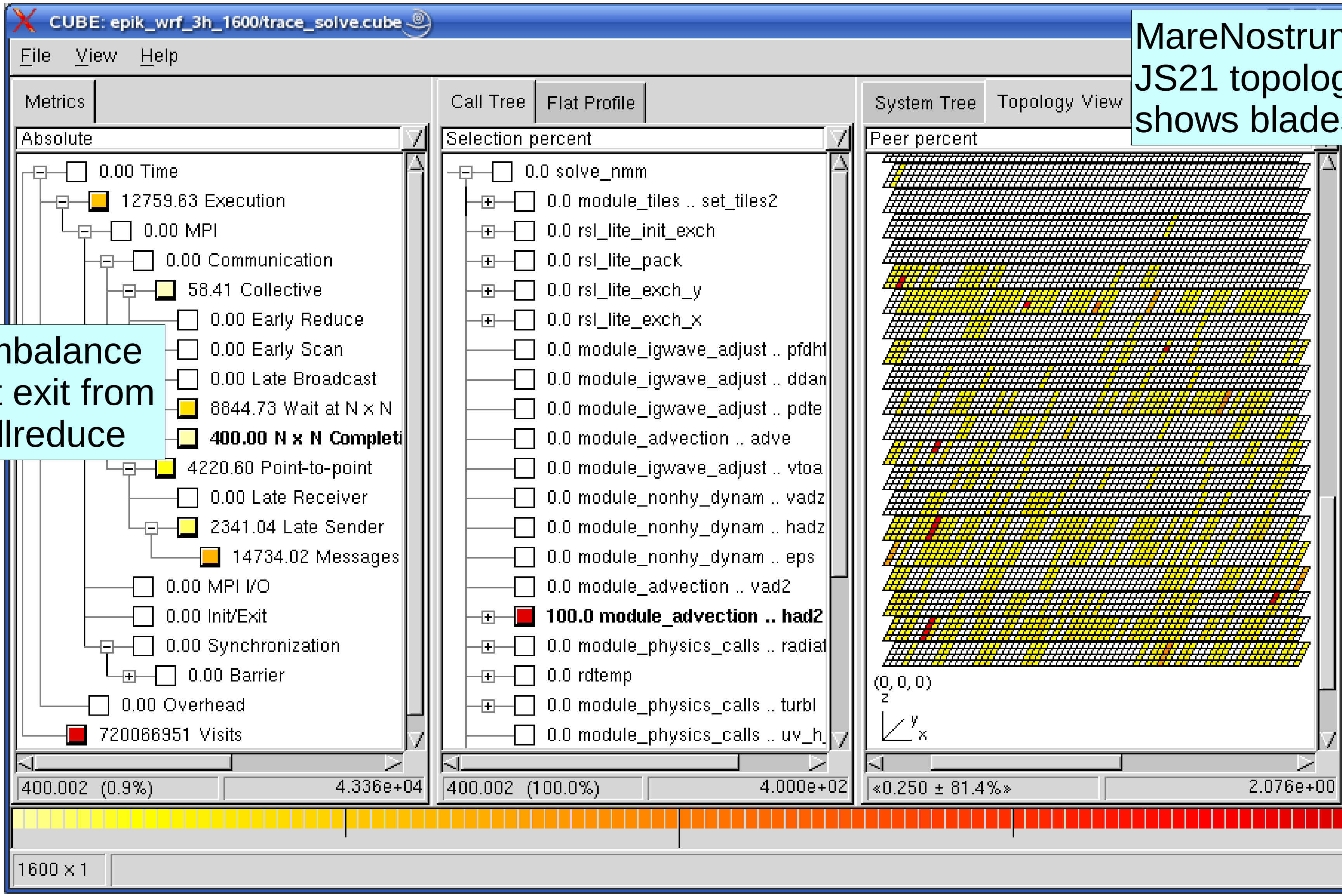
Journal of Scientific Programming 16(2-3):167-181 (2008)

WRF on MareNostrum@1600 with HWC metrics



Distribution of data load hits in level 2 cache on application MPI 2D grid topology

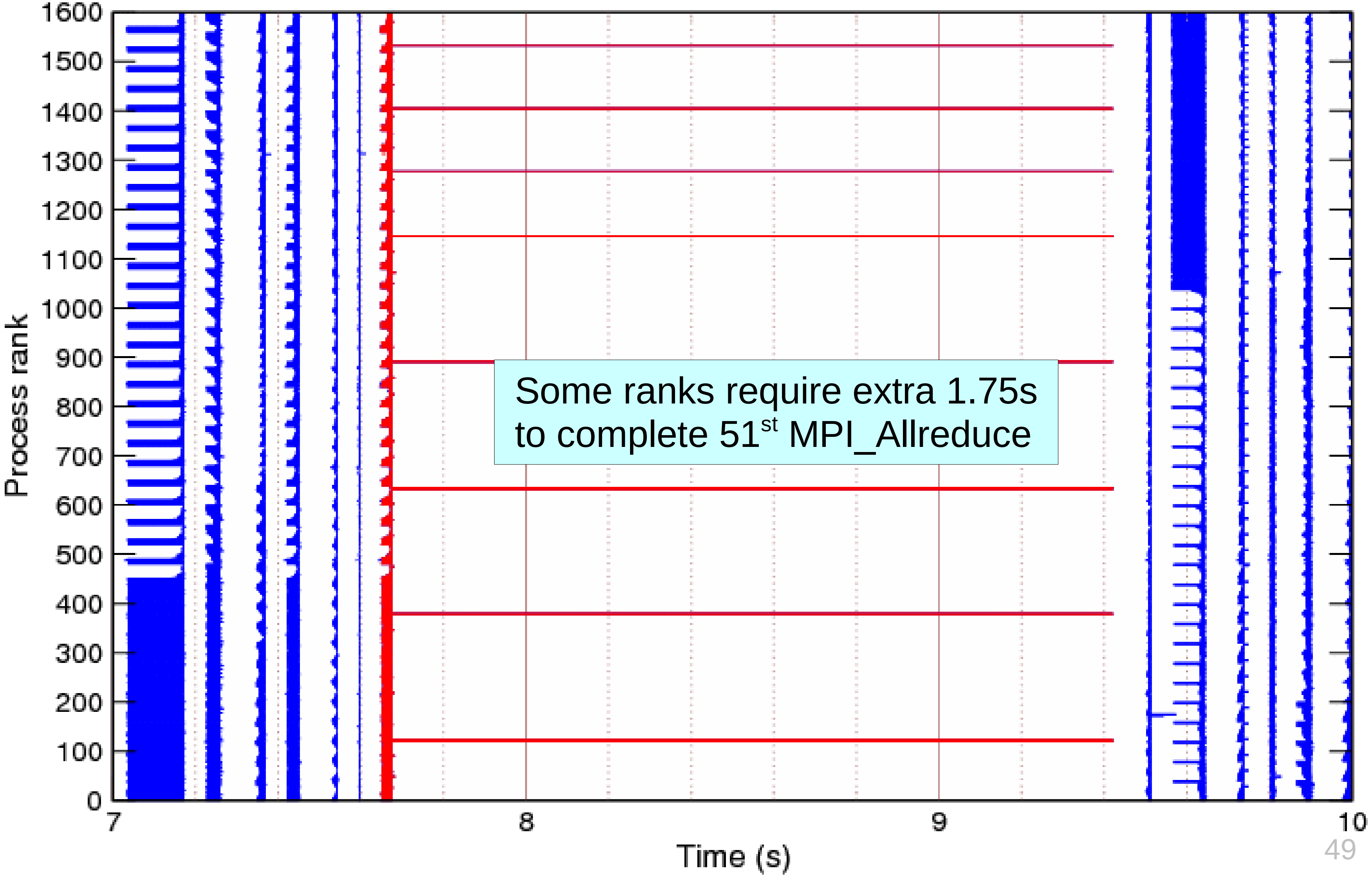
WRF on MareNostrum@1600 trace analysis



MareNostrum JS21 topology shows blades

Imbalance at exit from Allreduce

WRF on MareNostrum@1600 time-line extract



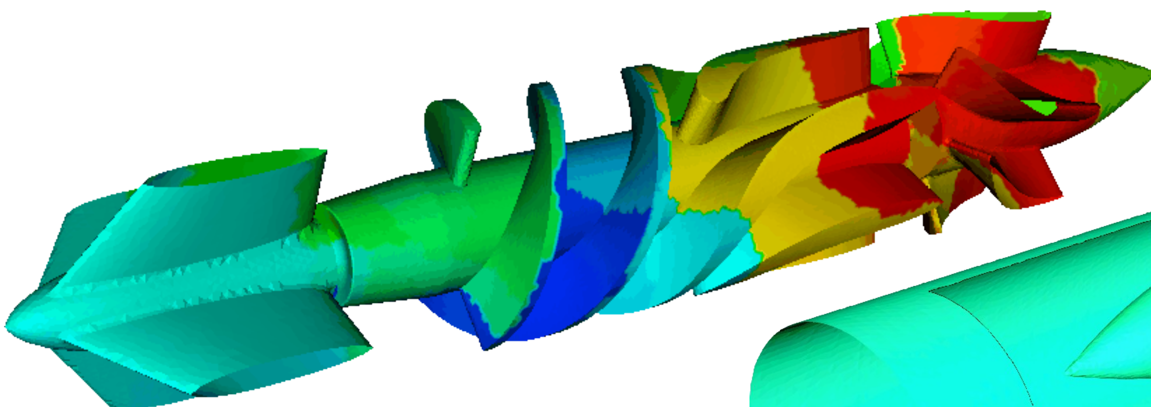
WRF/MareNostrum experience

- Limited system I/O requires careful management
  - Selective instrumentation and measurement filtering
- PowerPC hardware counter metrics included in summary
  - Memory/cache data access hierarchy constructed
- Automated trace analysis quantified impact of imbalanced exit from MPI_Allreduce in “NxN completion time” metric
  - Intermittent but serious MPI library/system problem, that restricts application scalability
  - Only a few processes directly impacted, however, communication partners also quickly blocked
- Presentation using logical and physical topologies
  - MPI Cartesian topology provides application insight
  - Hardware topology helps localize system problems

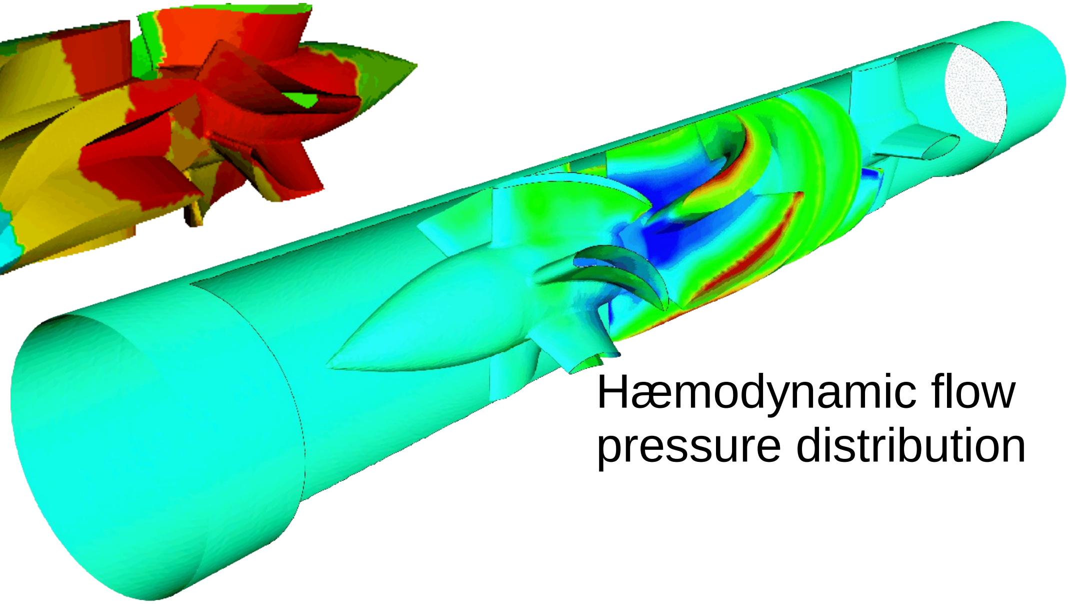
Journal of Scientific Programming 16(2-3):167-181 (2008)

XNS on BlueGene/L case study

- CFD simulation of unsteady flows
  - developed by RWTH CATS group of Marek Behr
  - exploits finite-element techniques, unstructured 3D meshes, iterative solution strategies
- MPI parallel version (Dec-2006)
  - >40,000 lines of Fortran & C
  - DeBakey blood-pump dataset (3,714,611 elements)



Partitioned finite-element mesh

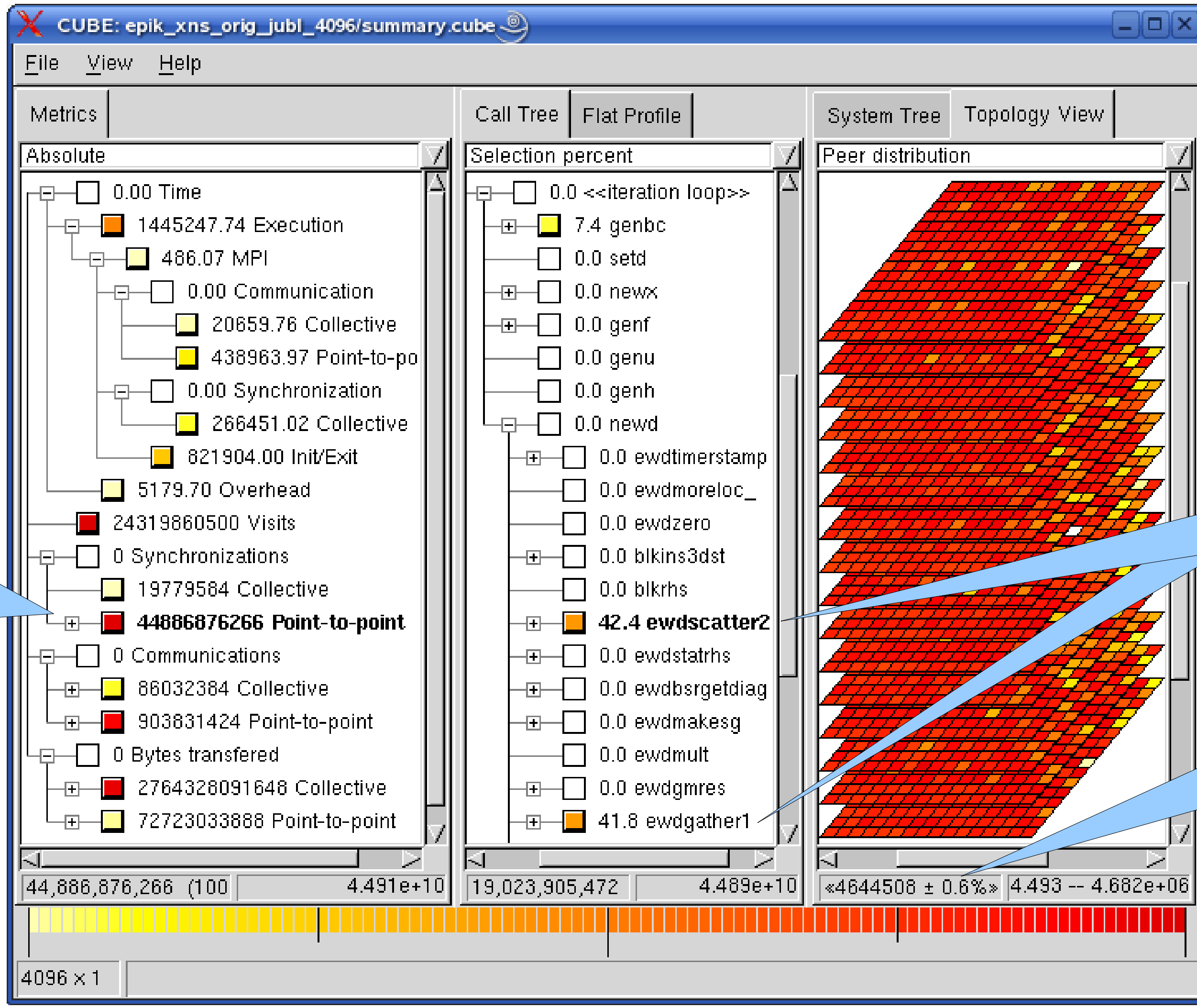


Hæmodynamic flow pressure distribution

XNS-DeBakey on jubl@4096 summary analysis

Point-to-point msgs w/o data

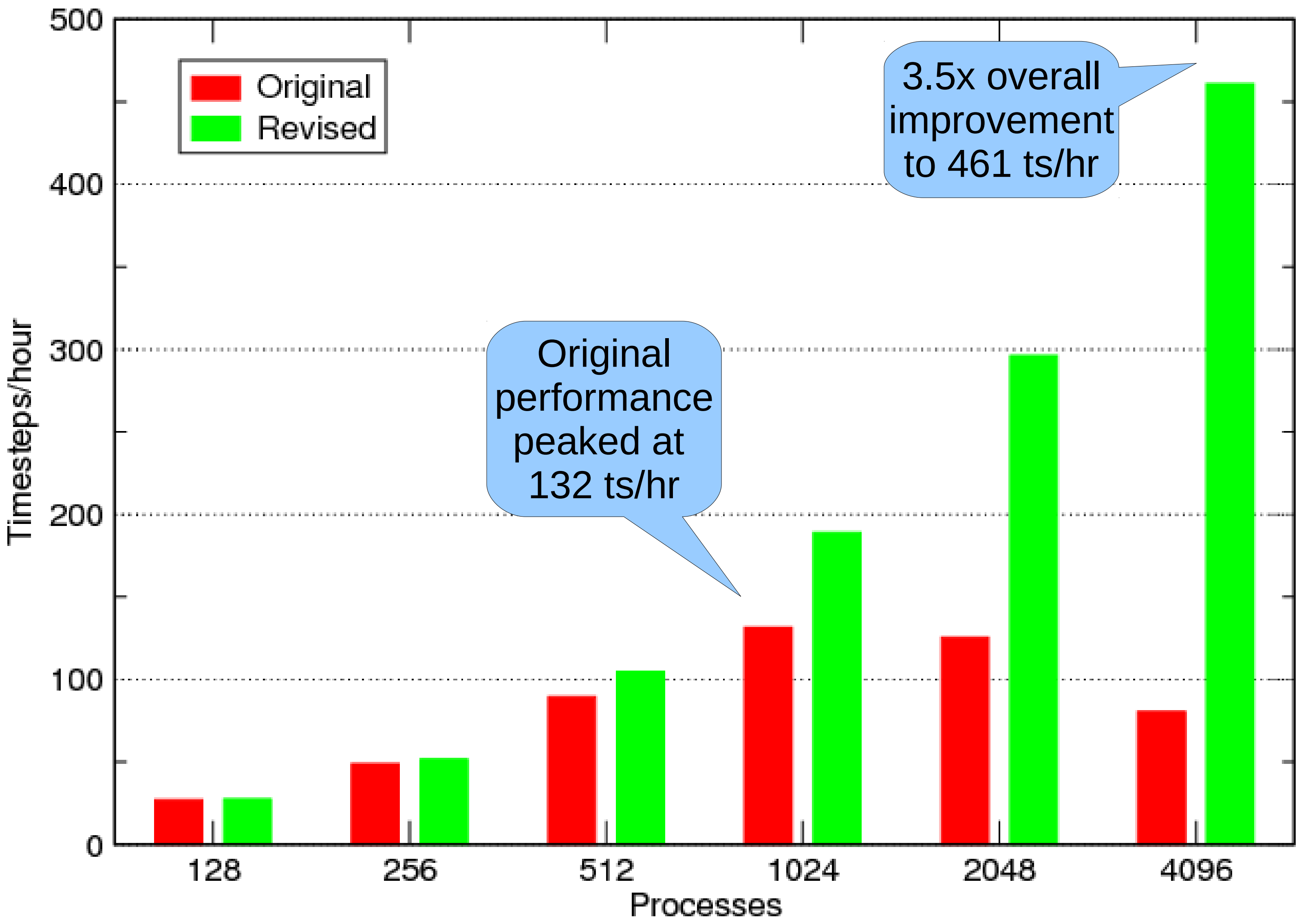
Masses of P2P synch operations



Primarily in scatter & gather

Processes all equally responsible

XNS-DeBakey scalability on BlueGene/L



XNS on BlueGene/L experience

- Globally synchronized high-resolution clock facilitates efficient measurement & analysis
- Restricted compute node memory limits trace buffer size and analyzable trace size
- Summarization identified bottleneck due to unintended P2P synchronizations (messages with zero-sized payload)
- 4x solver speedup after replacing MPI_Sendrecv operations with size-dependant separate MPI_Send and MPI_Recv
- Significant communication imbalance remains due to mesh partitioning and mapping onto processors
- MPI_Scan implementation found to contain implicit barrier
  - responsible for 6% of total time with 4096 processes
  - decimated when substituted with simultaneous binomial tree

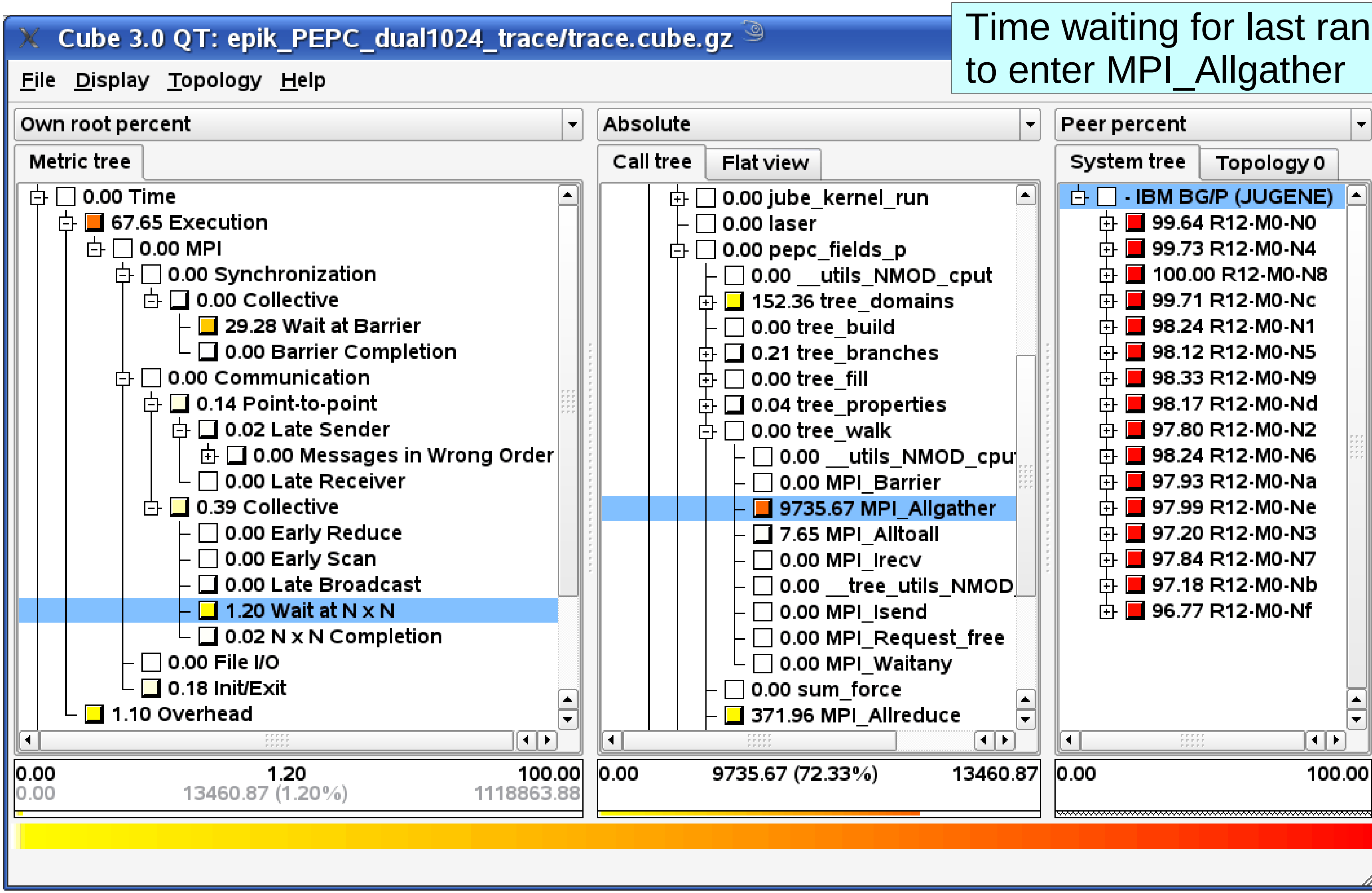
Proc. 14th EuroPVM/MPI, LNCS 4757 (2007)

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

- Coulomb solver used for laser-plasma simulations
  - Developed by Paul Gibbon (JSC)
  - Tree-based particle storage with dynamic load-balancing
- MPI version
  - PRACE benchmark configuration, including file I/O
- Run on 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
- Run on Cray XT in VN (4p) mode with 1024 processes
  - 4 processes per quad-core Opteron node, 360 seconds
  - PGI compilers and Cray MPI, CNL, SeaStar interconnect

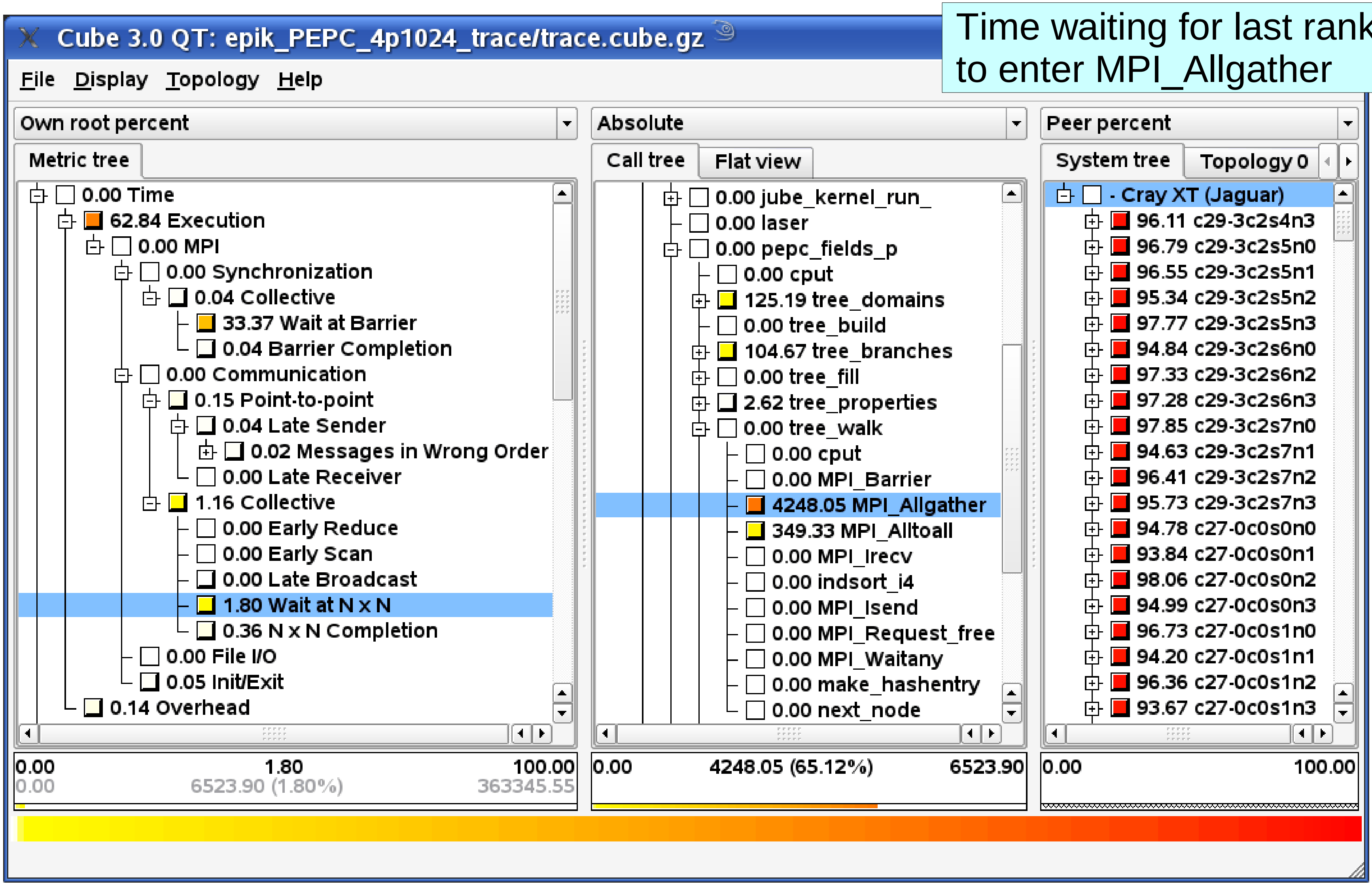
Proc. 52nd Cray User Group (Edinburgh, 2010)

PEPC@1024 on BlueGene/P: Wait at NxN time



Time waiting for last rank to enter MPI_Allgather

PEPC@1024 on Cray XT4: Wait at NxN time



Time waiting for last rank to enter MPI_Allgather

PEPC-B on BG/P & Cray XT experience

- Despite very different processor and network performance, measurements and analyses can be easily compared
  - 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

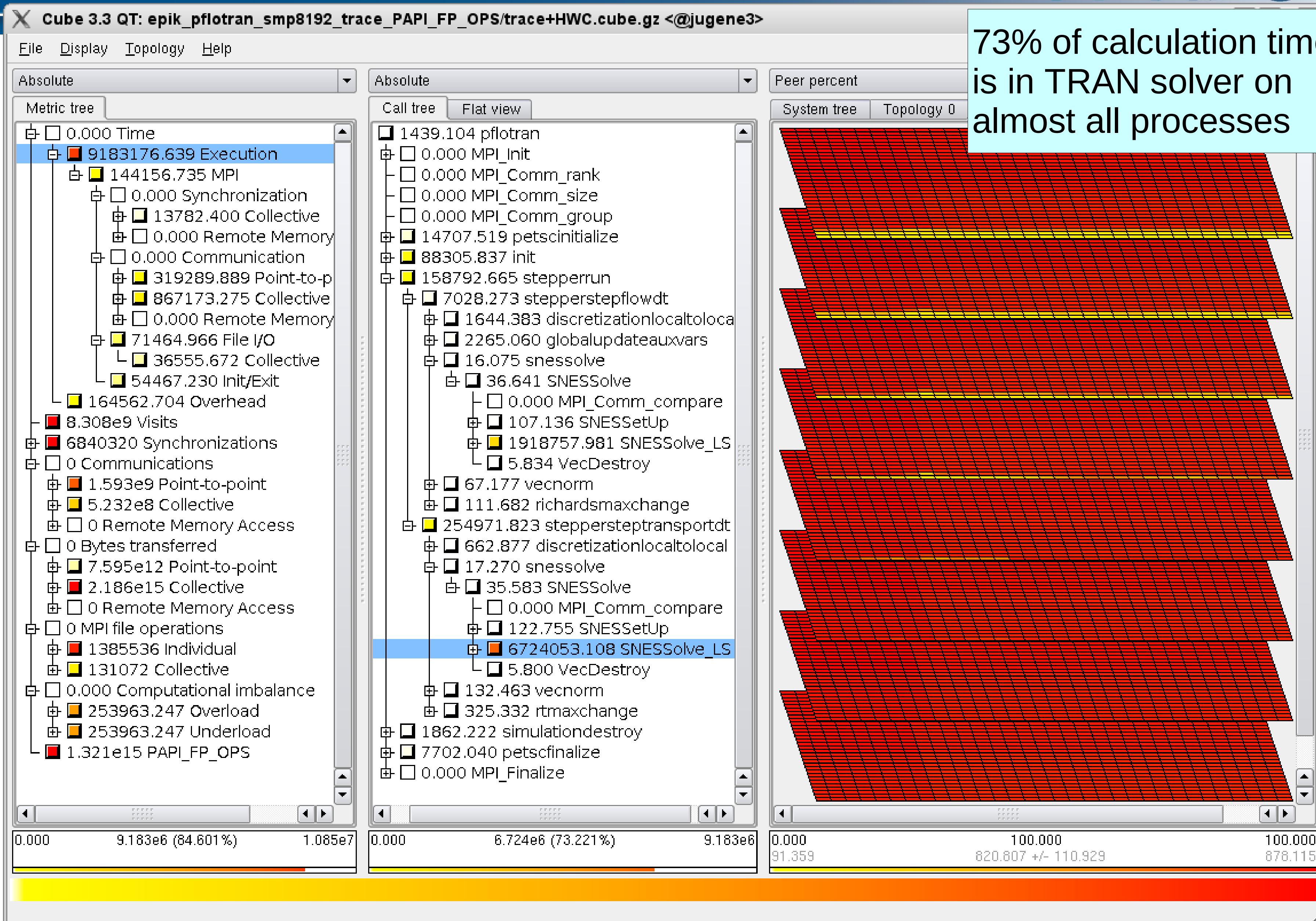
Proc. 52nd Cray User Group (Edinburgh, 2010)

PFLOTRAN on BlueGene/P case study

- 3D reservoir simulator combining alternating
  - PFLOW non-isothermal, multiphase groundwater flow
  - PTRAN reactive, multi-component contaminant transport
  - developed by LANL/ORNL/PNNL
- MPI with PETSc, LAPACK, BLAS & HDF5 I/O libraries
  - ~80,000 lines (97 source files) Fortran9X
  - PFLOTRAN & PETSc fully instrumented by IBM XL compilers
    - filter produced listing 856 USR routines (leaving 291 COM)
    - 1732 unique callpaths (399 in FLOW, 375 in TRAN)
    - 633 MPI callpaths (121 in FLOW, 114 in TRAN)
      - 29 distinct MPI routines recorded (excludes 15 misc. routines)
- Run on IBM BlueGene/P with '2B' input dataset (10 steps)
  - Scalasca summary & trace measurements (some with PAPI)
  - 22% dilation of FLOW, 10% dilation of TRAN [8k summary]

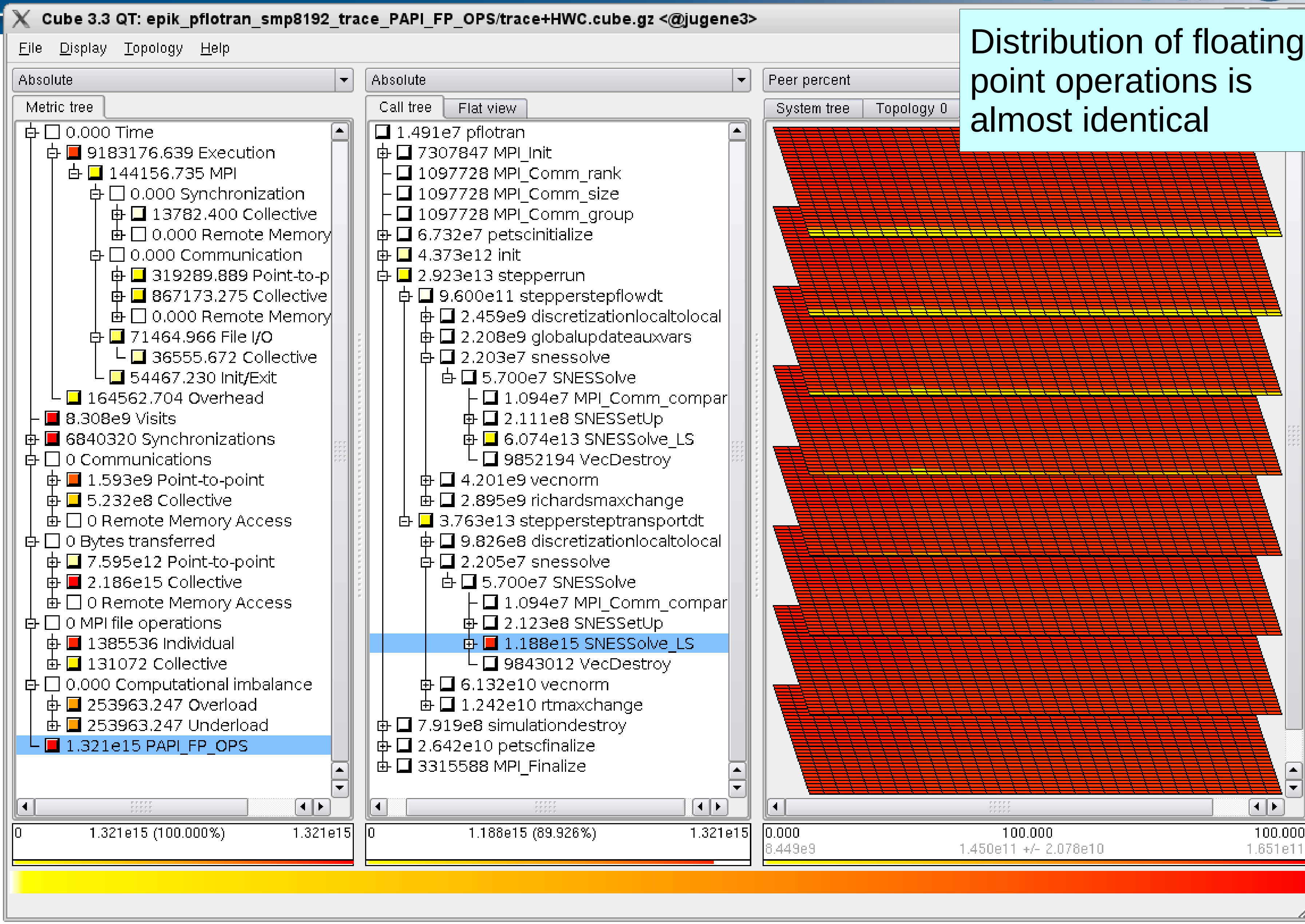
Proc. 53rd Cray User Group (Fairbanks, 2011)

PFLOTRAN jugene@smp8192 trace analysis



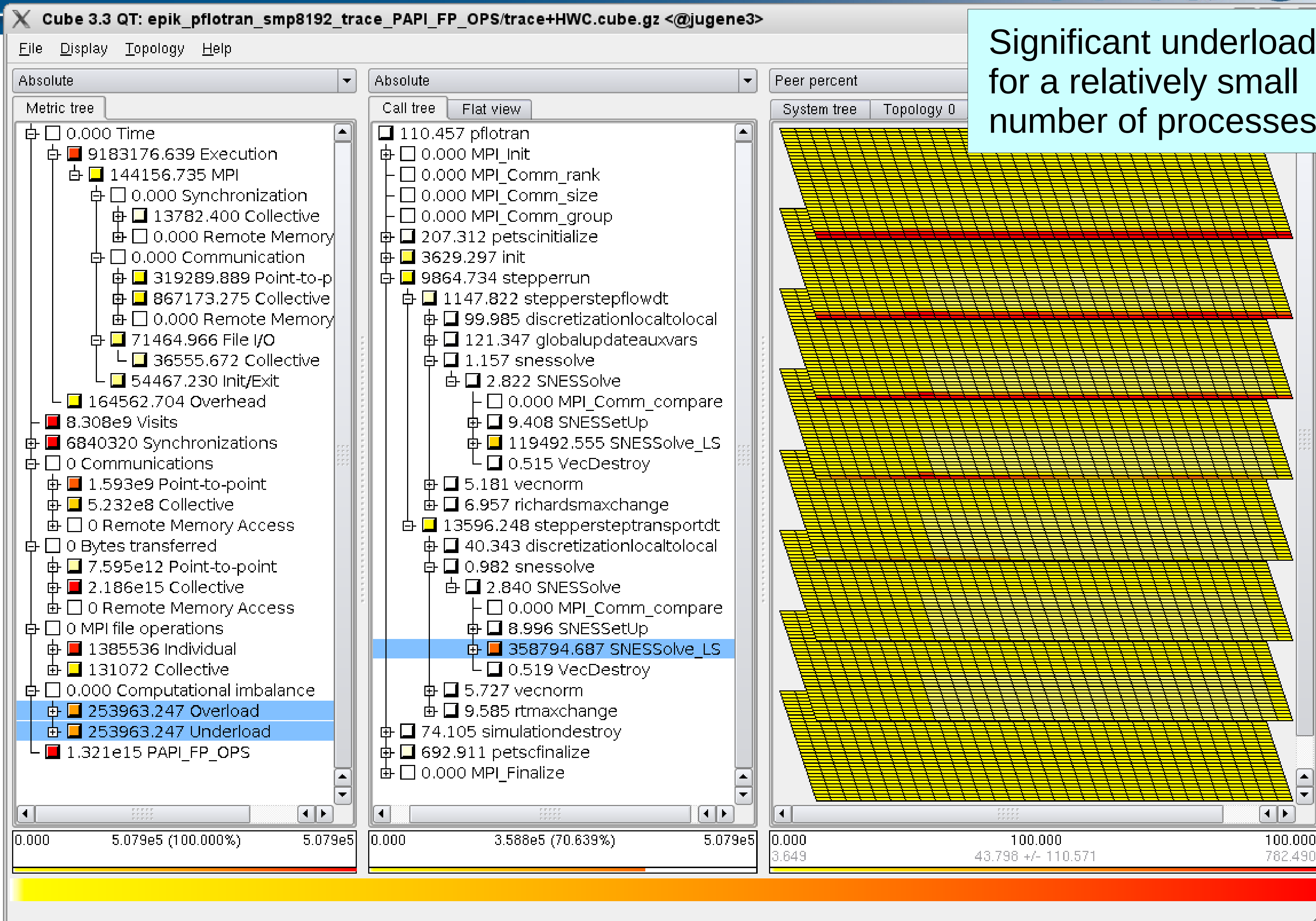
73% of calculation time is in TRAN solver on almost all processes

PFLOTRAN jugene@smp8192 trace analysis



Distribution of floating-point operations is almost identical

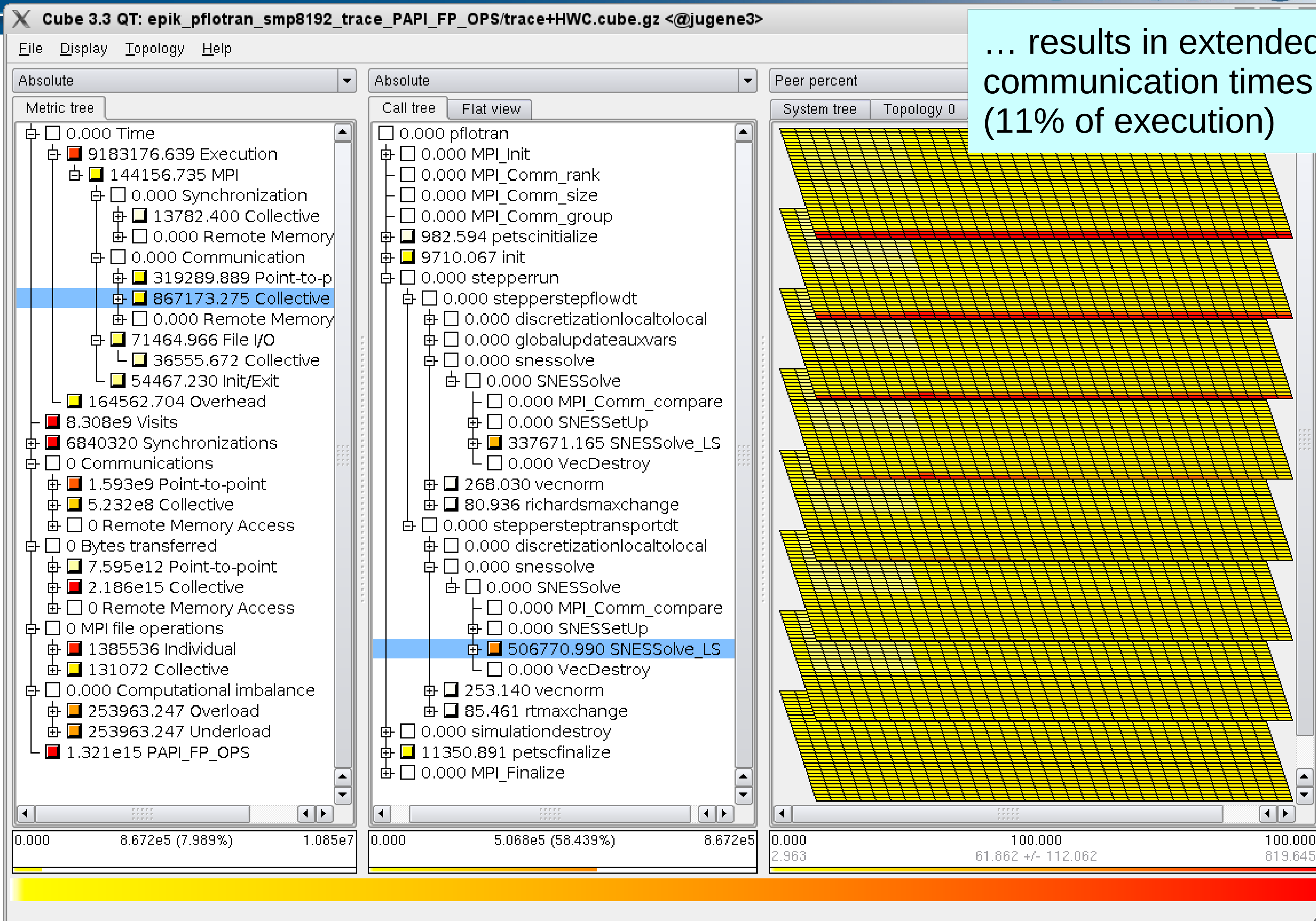
PFLOTRAN jugene@smp8192 trace analysis



Significant underload for a relatively small number of processes

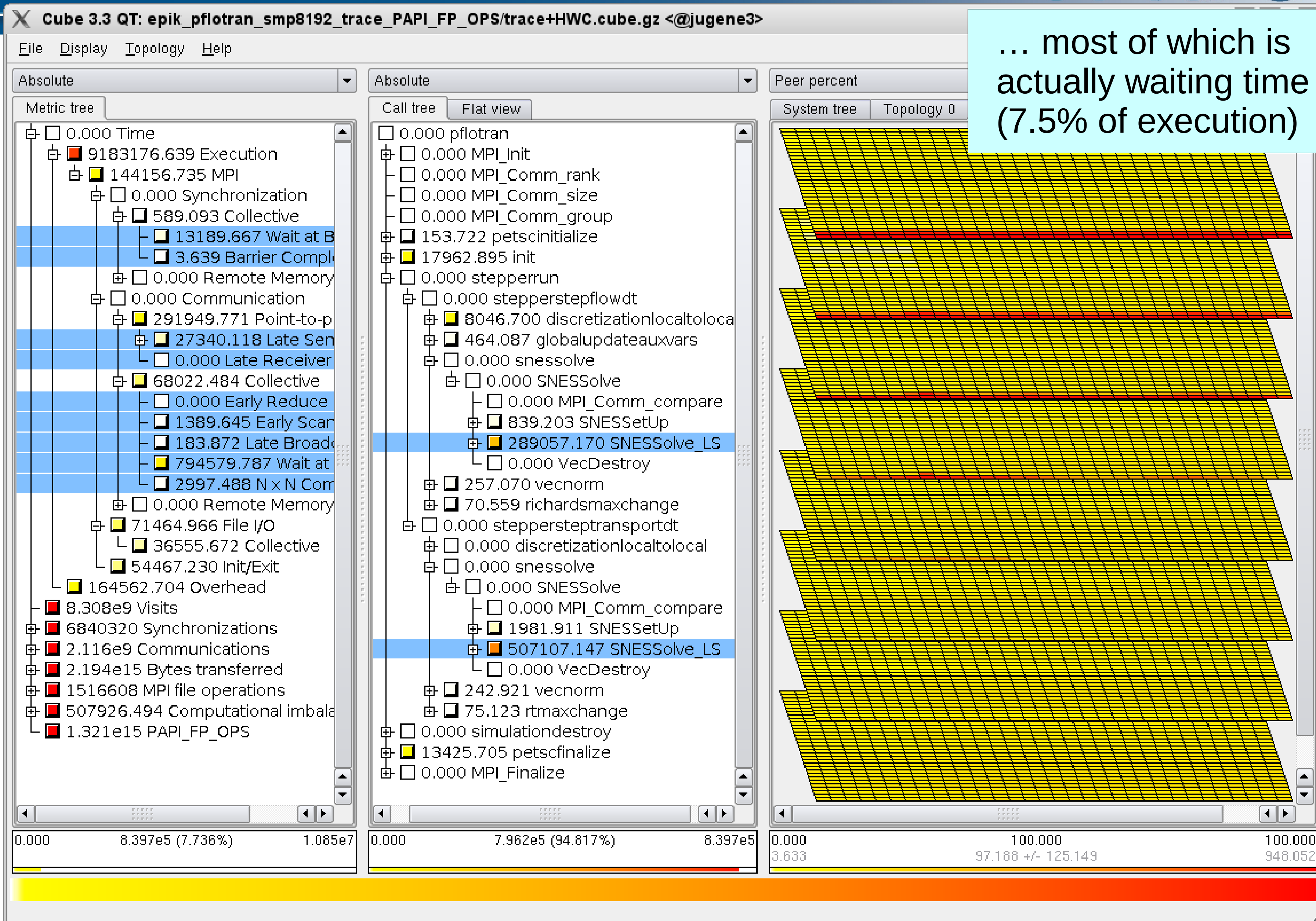
PFLOTRAN jugene@smp8192 trace analysis

VI-HPS



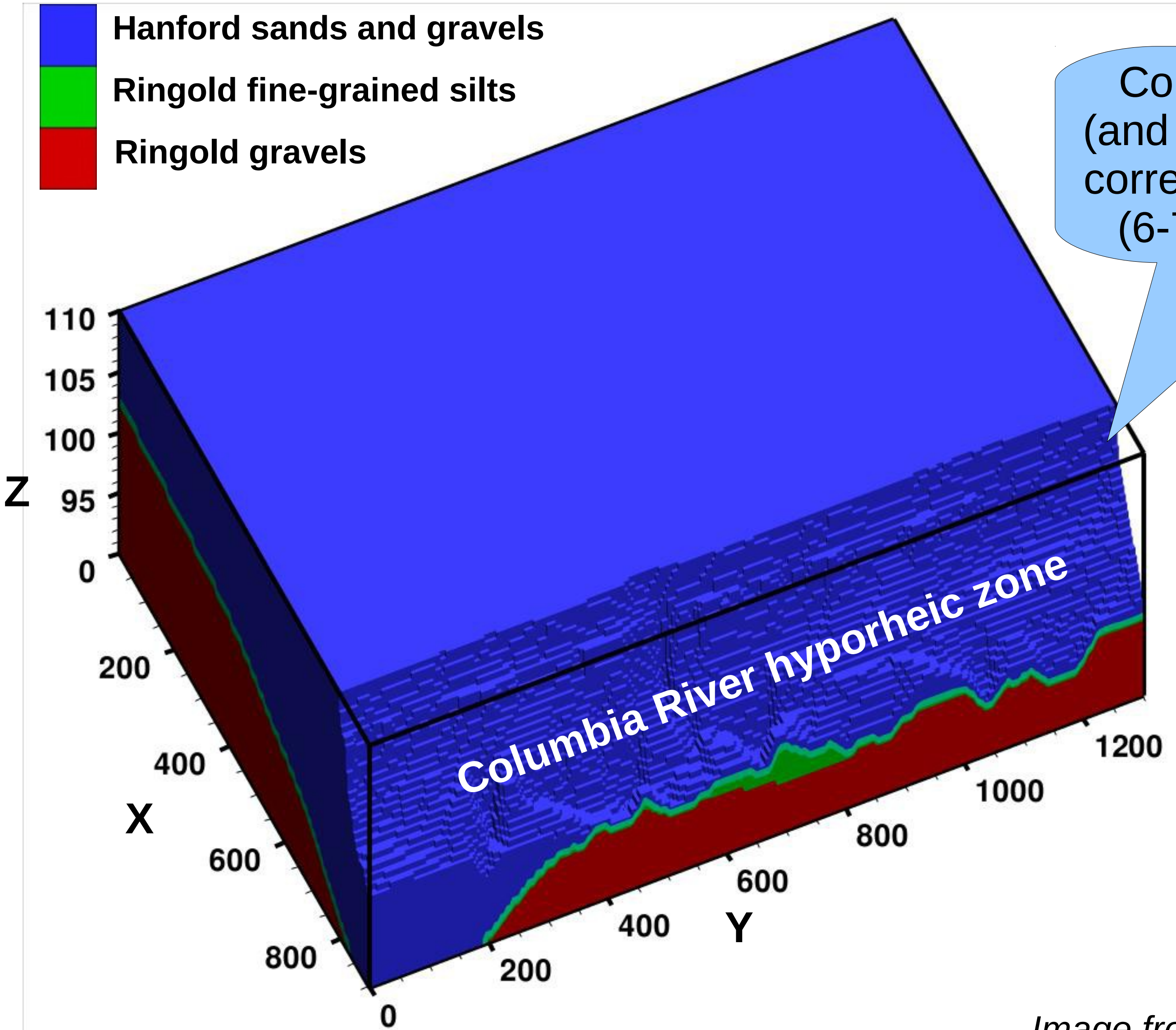
… results in extended communication times (11% of execution)

PFLOTRAN jugene@smp8192 trace analysis



… most of which is actually waiting time (7.5% of execution)

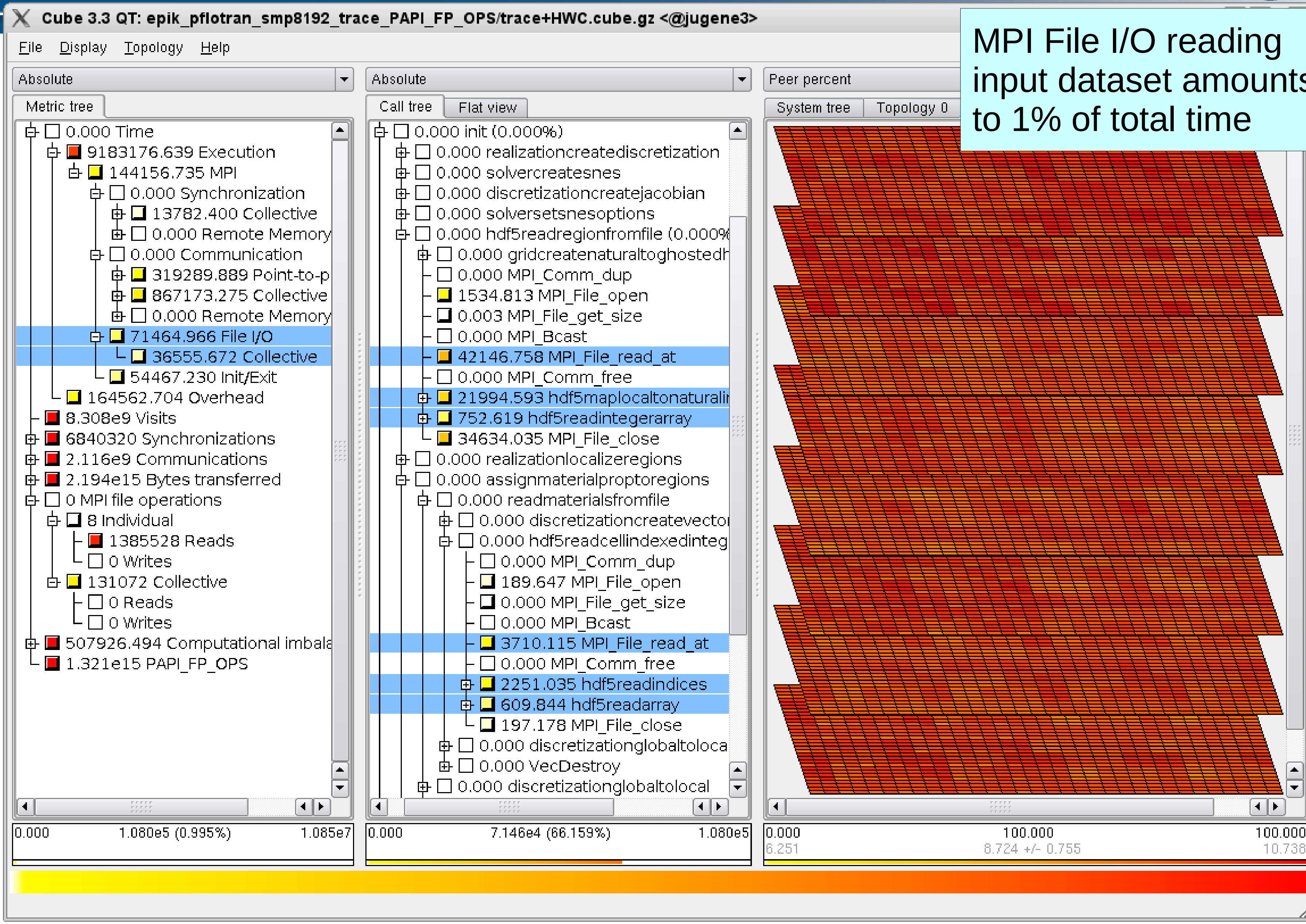
Geology of DOE Hanford 300 area (WA, USA)



Computational imbalance (and associated waiting time) correlate to inactive grid cells (6-7% of total) within river

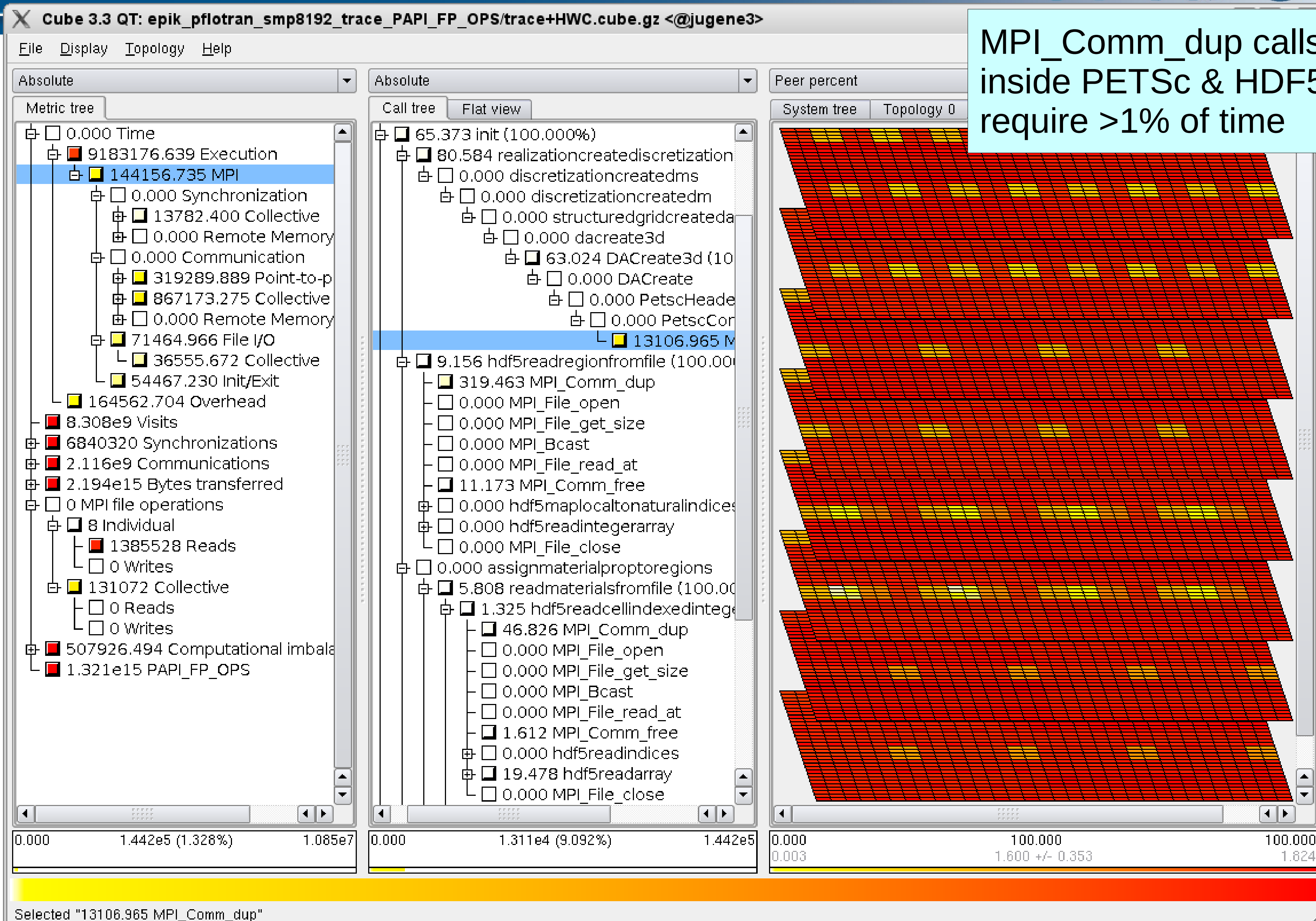
Image from Glenn Hammond (PNNL)

PFLOTRAN jugene@smp8192 trace analysis



MPI File I/O reading input dataset amounts to 1% of total time

PFLOTRAN jugene@smp8192 trace analysis



MPI_Comm_dup calls inside PETSc & HDF5 require >1% of time

PFLOTRAN on BlueGene/P experience

- Initialization phase dominates at larger scales
  - 10% of total execution time spent duplicating communicators with 128k processes on Cray XT5
  - otherwise collective MPI File I/O relatively efficient
  - typically amortized in long simulation runs
- Solver scaled well to 64k processes before degrading
  - similar computation/communication patterns in FLOW & TRAN
    - callpath profiles distinguish costs
    - MPI_Allreduce collective communication becomes a bottleneck
    - communication overhead explodes for smaller FLOW problem
      - TRAN problem is 15x larger due to 15 chemical species
  - inactive processes induce clear computational imbalance
    - and are associated with large amounts of MPI waiting time
    - however, they constitute a relatively small minority

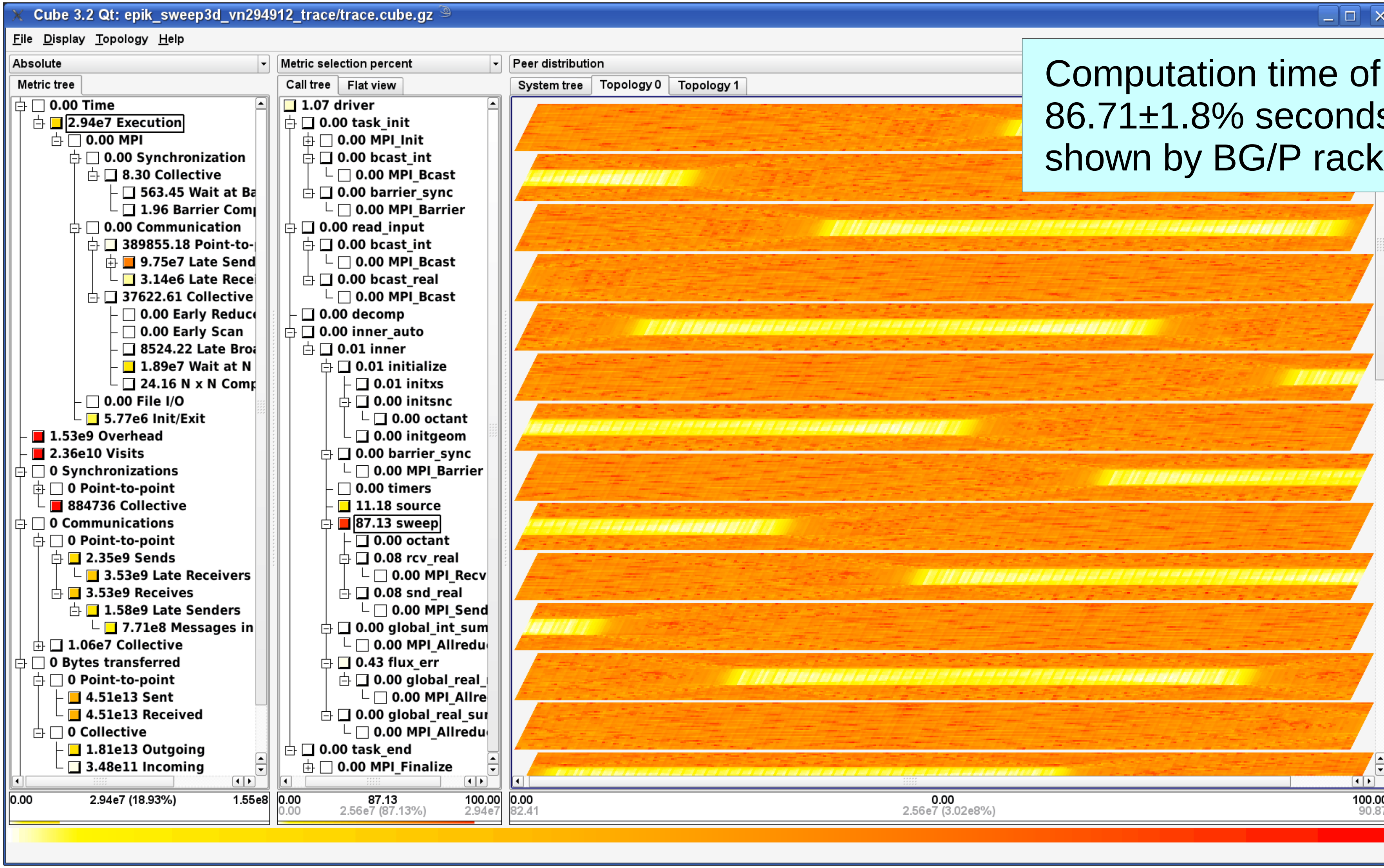
Proc. 53rd Cray User Group (Fairbanks, 2011)

Sweep3d on BlueGene/P case study

- 3D neutron transport simulation
  - ASC benchmark
  - direct order solve uses diagonal sweeps through grid cells
  - 'fixups' applied to correct unphysical (negative) fluxes
- MPI parallel version 2.2b using 2D domain decomposition
  - ~2,000 lines (12 source modules), mostly Fortran77
- Run on IBM BlueGene/P in VN mode with 288k processes
  - 7.6TB trace written in 17 minutes, analyzed in 10 minutes
    - of which 10 minutes for SIONlib open/create of 576 physical files
    - (compared to 86 minutes just to open/create a file per MPI rank)
  - Mapping of metrics onto application's 576x512 process grid reveals regular pattern of performance artifacts
    - computational imbalance originates from 'fixup' calculations
    - combined with diagonal wavefront sweeps amplifies waiting times

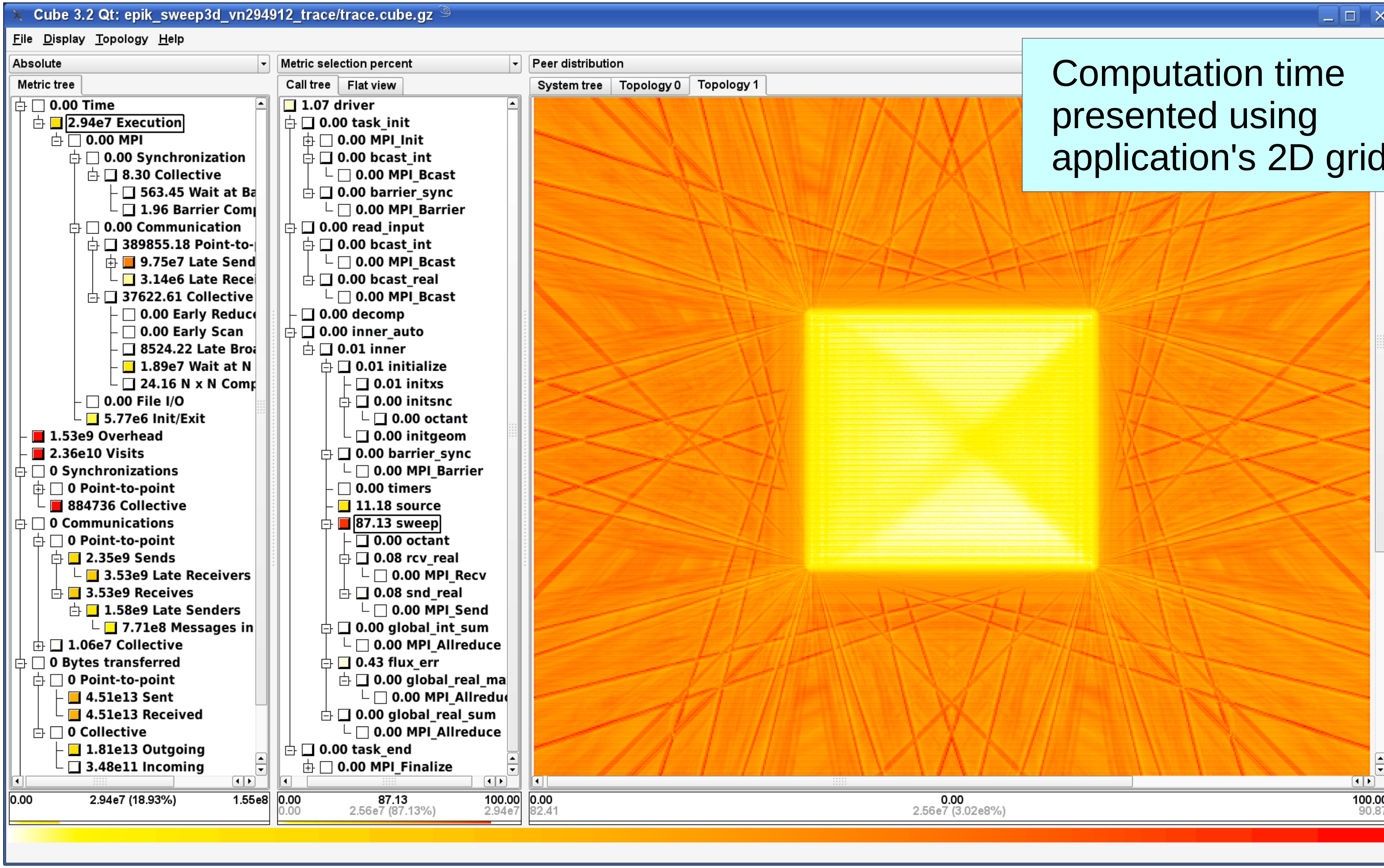
Proc. IPDPS Workshop on Large-Scale Parallel Processing (2010)

sweep3d on jugene@288k trace analysis



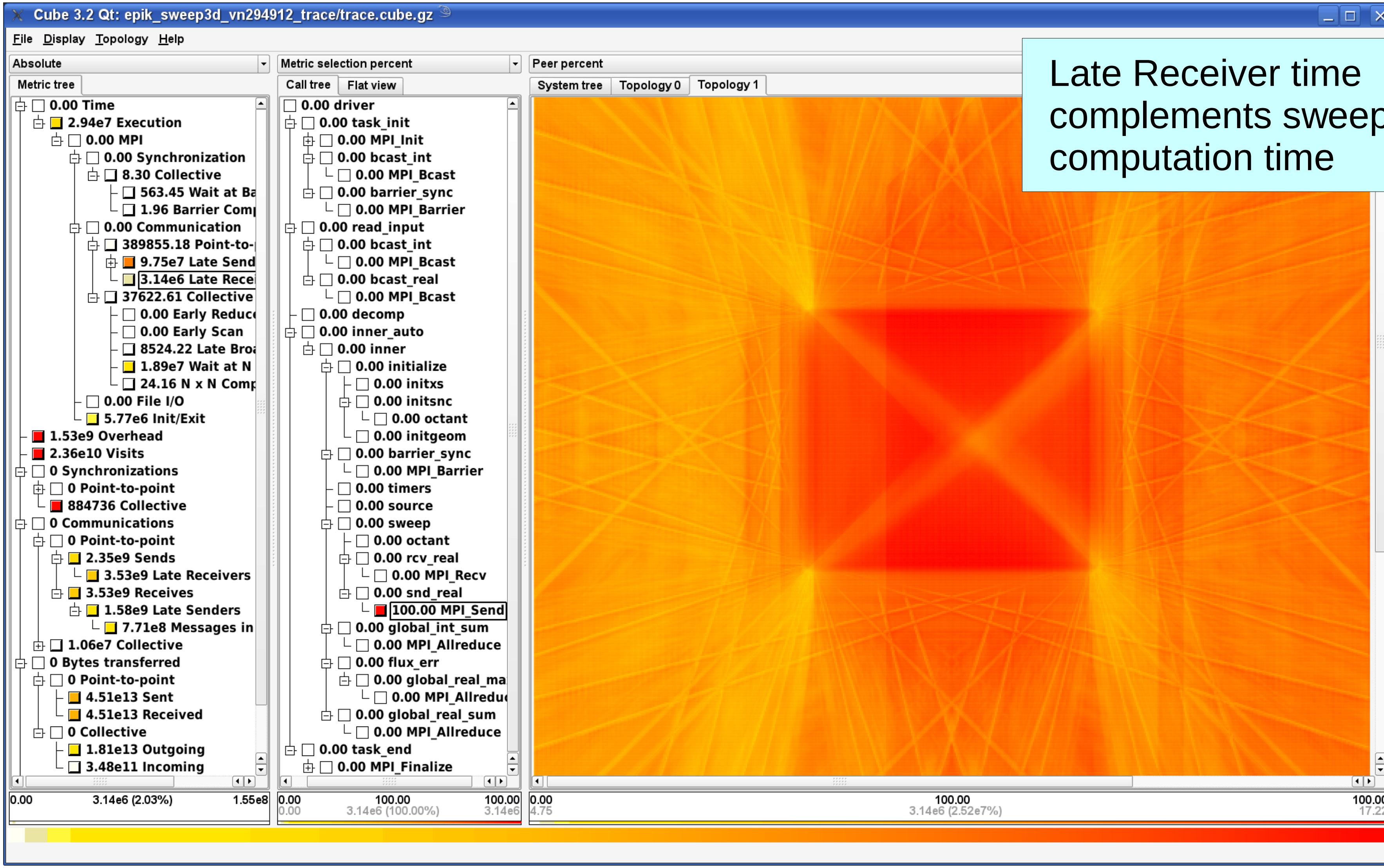
Computation time of 86.71±1.8% seconds shown by BG/P rack

sweep3d on jugene@288k trace analysis



Computation time presented using application's 2D grid

sweep3d on jugene@288k trace (wait) analysis



Late Receiver time complements sweep computation time

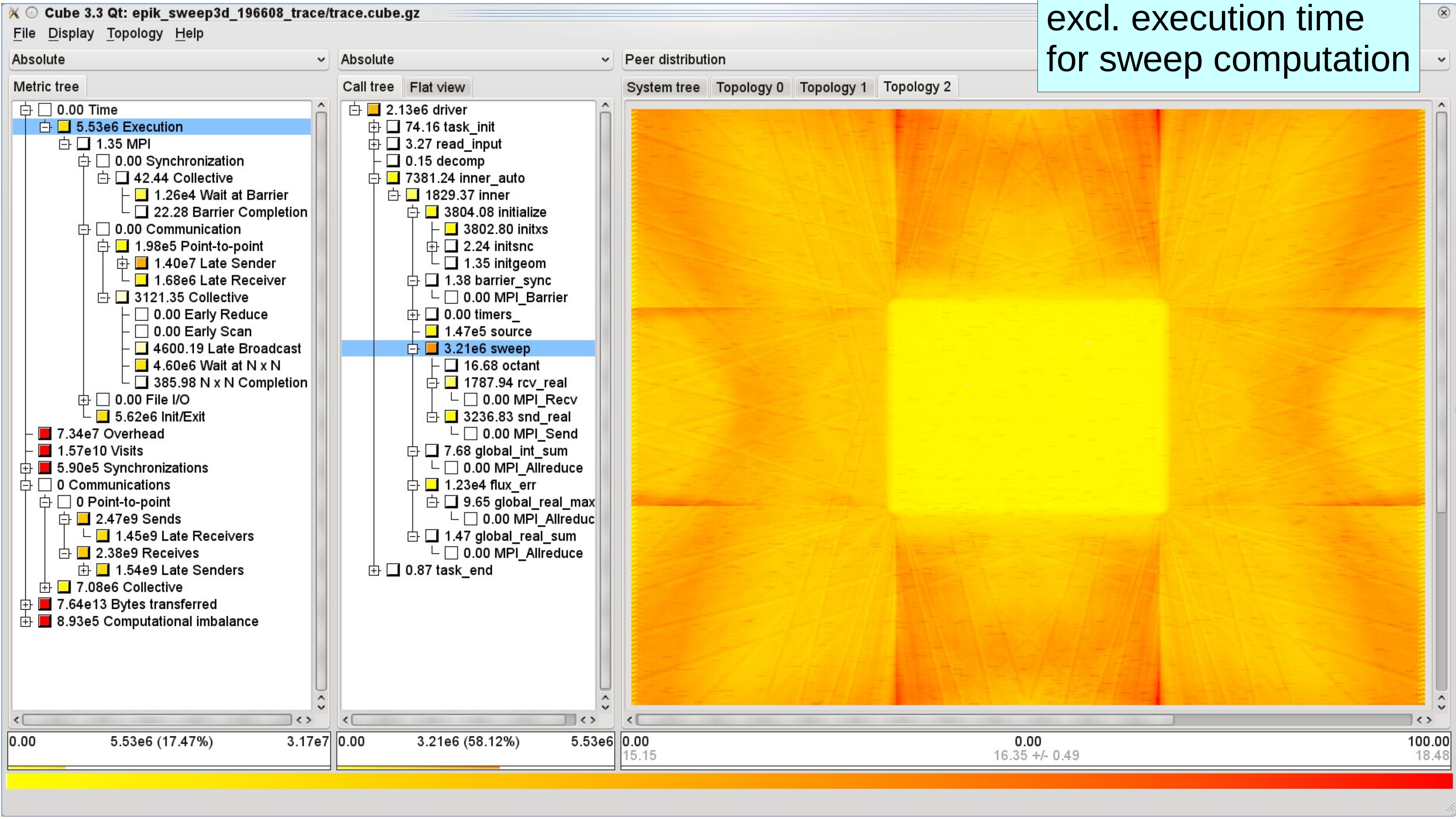
Sweep3d on Cray XT5 case study

- 3D neutron transport simulation
  - ASC benchmark
  - direct order solve uses diagonal sweeps through grid cells
  - 'fixups' applied to correct unphysical (negative) fluxes
- MPI parallel version 2.2b using 2D domain decomposition
  - ~2,000 lines (12 source modules), mostly Fortran77
- Run on Cray XT5 with 192k processes
  - 0.5TB trace written in 10 minutes, analyzed in 4 minutes
    - 6 minutes to open/create trace file for each rank
    - 25s for timestamp correction, 93s for parallel event replay
  - Mapping of metrics onto application's 512x384 process grid reveals regular pattern of performance artifacts
    - computational imbalance originates from 'fixup' calculations
    - combined with diagonal wavefront sweeps amplifies waiting times

Parallel Processing Letters 20(4):397-414 (2010)

sweep3d on jaguar@192k trace analysis

Regular imbalance in excl. execution time for sweep computation



Acknowledgements

- 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, CINECA, CMM, CSC, CSCS, DKRZ, EPCC, HLRN, HLRS, ICM, IMAG, JSC, KSL, KTH, LLNL, LRZ, NCAR, NCCS, NICS, RWTH, RZG, SARA, TACC, TGCC, ZIH
    - Access & usage supported by European Union, German and other national funding organizations
- The Scalasca users for their comprehensive problem reports and improvement requests
  - as well as sharing reports of their analysis & tuning successes
- The Scalasca development team

Scalable performance **a**nalysis of **la**rge-**sc**ale parallel **a**pplications

- toolset for scalable performance measurement & analysis of MPI, OpenMP & hybrid parallel applications
- supporting most popular HPC computer systems
- available under New BSD open-source license
- sources, documentation & publications:
  - http://www.scalasca.org
  - mailto: scalasca@fz-juelich.de