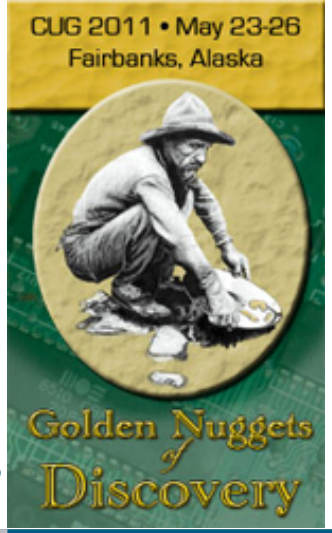


JÜLICH
FORSCHUNGSZENTRUM

Large-scale performance analysis of *PFLOTRAN* with Scalasca

2011-05-26 |

Brian J. N. Wylie & Markus Geimer
Jülich Supercomputing Centre

b.wylie@fz-juelich.de

Overview

Dagstuhl challenge

- *PFLOTRAN* at large-scale on Cray XT5 & IBM BG/P

Scalasca performance analysis toolset

- Overview of architecture and usage
- Instrumentation of *PFLOTRAN* & PETSc
- Summary & trace measurement experiments
- Analysis report & trace exploration

Conclusions

Dagstuhl challenge

Demonstrate performance measurement and analysis of *PFLOTRAN* using “2B” problem dataset and more than 10,000 MPI processes

Challenge issued for Dagstuhl seminar 10181 (3-7 May 2010) on “Program development for extreme-scale computing”

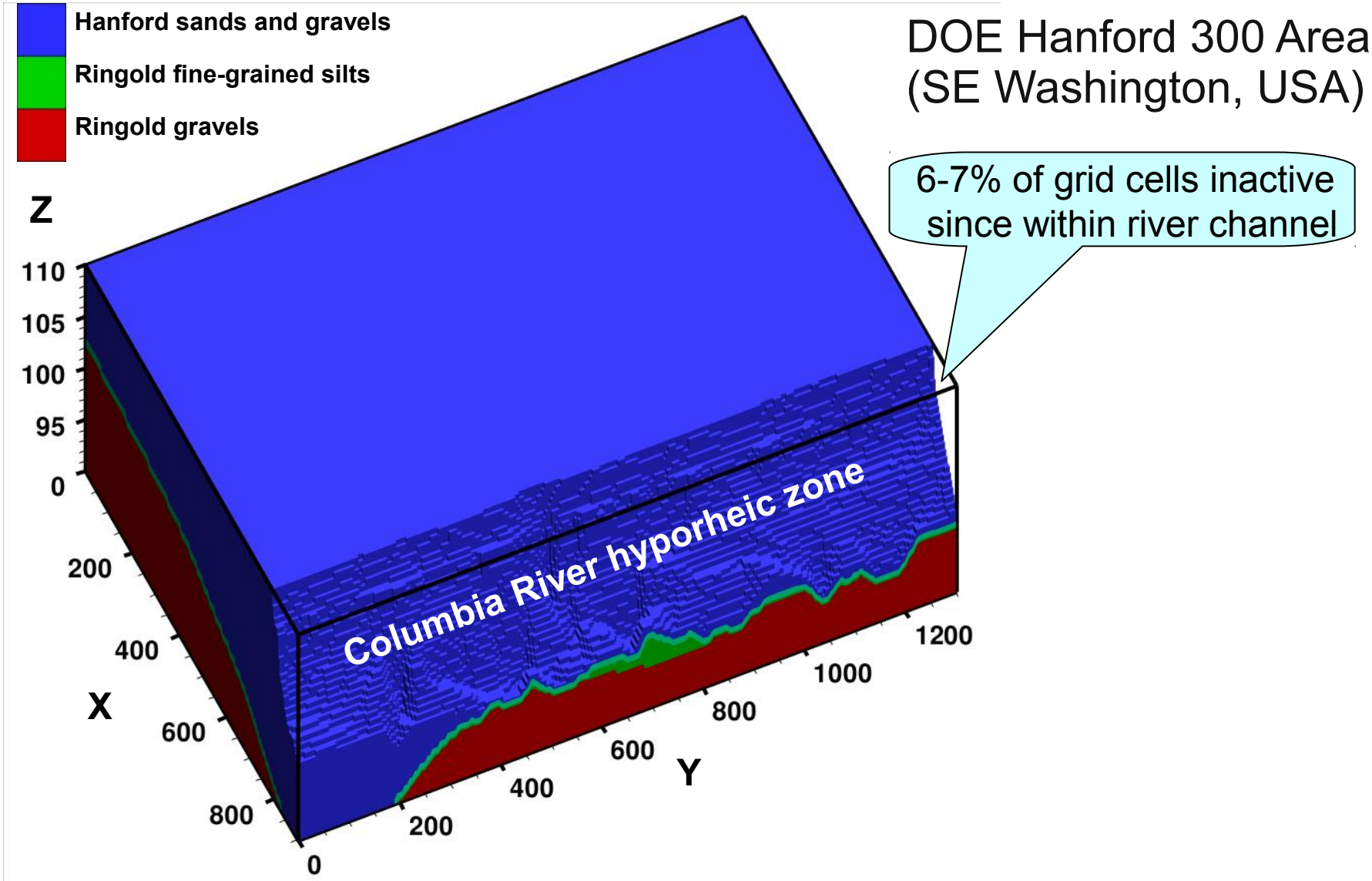
- two months notice with download/build/run instructions
- accounts/allocations provided by VI-HPS & INCITE/PEAC
  - *IBM BG/P (**jugene**.fz-juelich.de)*
  - *Cray XT5 (**jaguar**.nccs.ornl.gov)*
- numerous research groups and vendors presented their results and discussed issues encountered
  - *http://www.dagstuhl.de/10181*

PFLOTRAN

3D reservoir simulator actively developed by LANL/ORNL/PNNL

- approx. 80,000 lines of Fortran9X, combining solvers for
  - *PFLOW non-isothermal, multi-phase groundwater flow*
  - *PTRAN reactive, multi-component contaminant transport*
- employs PETSc, LAPACK, BLAS & HDF5 I/O libraries
  - *87 PFLOTRAN + 789 PETSc source files*
  - *parallel I/O tuning via PERI active liaison*
- "2B" input dataset run for 10 timesteps
  - *uses 3-dimensional (non-MPI) PETSc Cartesian grid*
  - *TRAN(sport) step scales much better than FLOW step*
  - *FLOW step generally faster, but crossover at larger scales*

PFLOTRAN "2B" test case



PFLOTRAN simulation of U(VI) migration

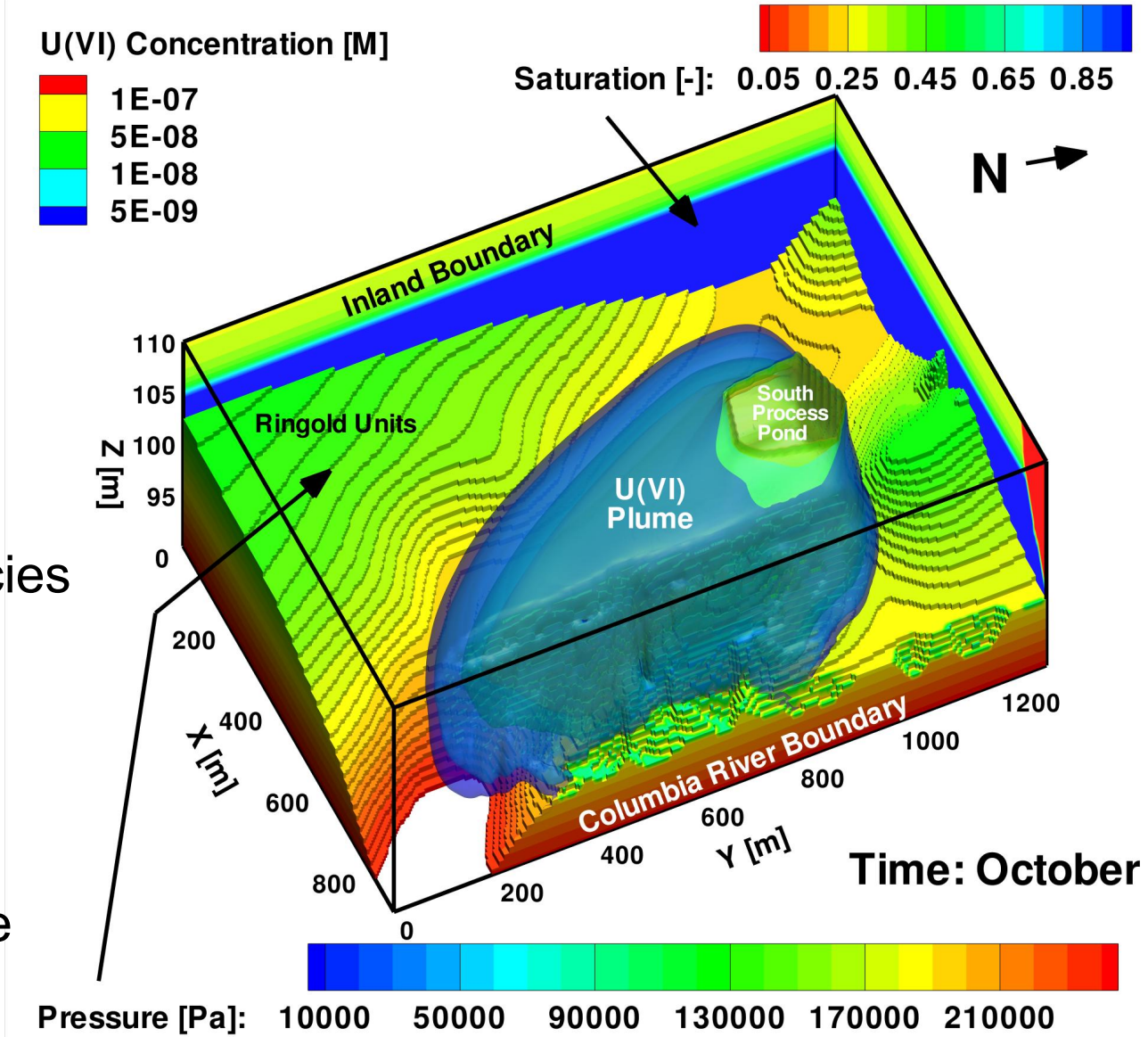
DOE Hanford 300 Area

Problem domain:

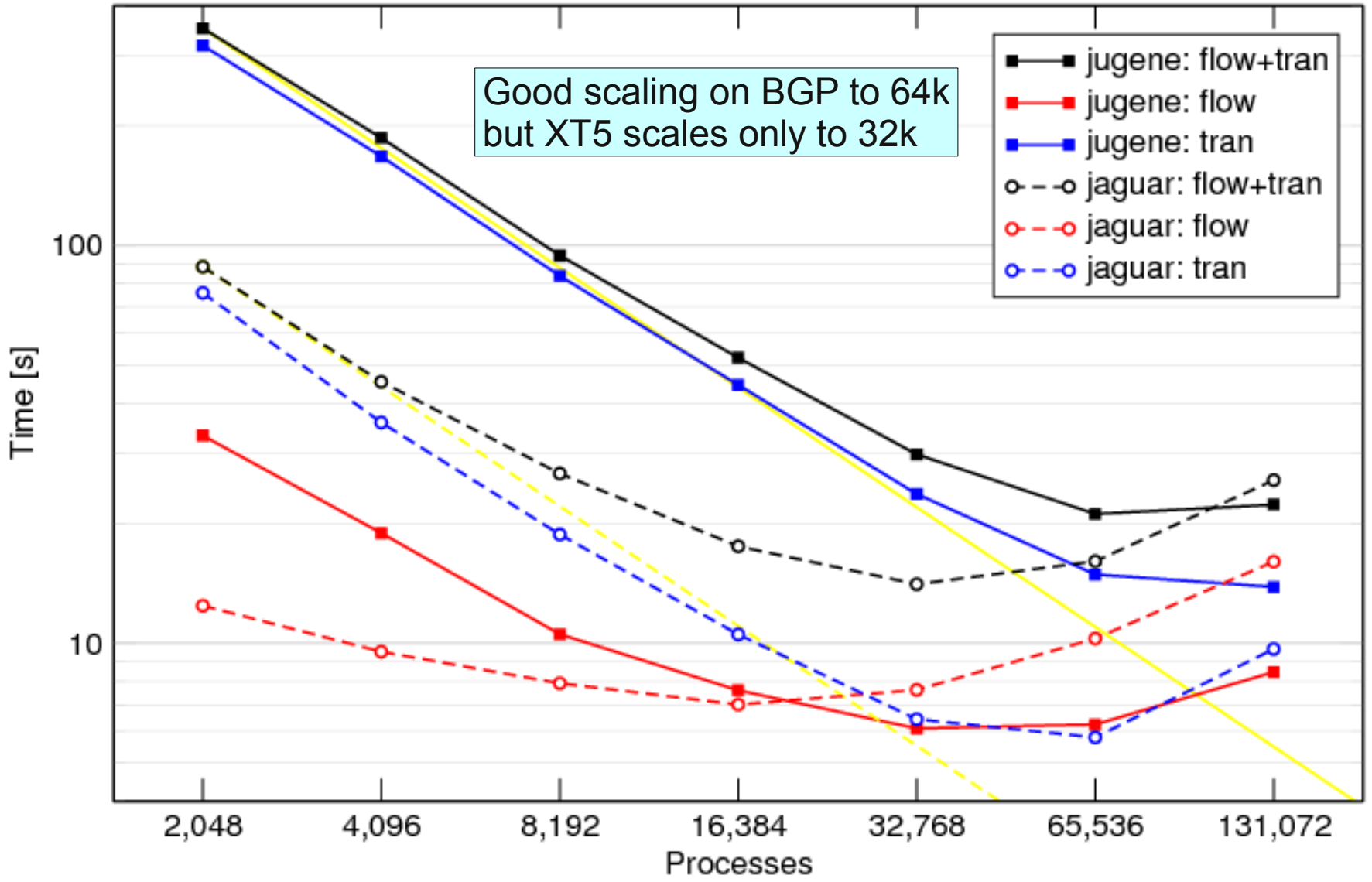
- 900x1300x20m
- $\Delta x/\Delta y$ = 5 m
- 1.87M grid cells
- 15 chemical species
- 28M DoF total

1-year simulation:

- Δt = 1 hour
- 5-10 hour runtime
- 4096 XT5 cores



PFLOTRAN "2B" strong scalability



Scalasca project

Overview

- Headed by Bernd Mohr (JSC) & Felix Wolf (GRS-Sim)
  - *Helmholtz Initiative & Networking Fund project started in 2006*
- 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 v1.3.3 released in March 2011
- Available for download from www.scalasca.org

Scalasca features

Open source, New BSD license

Portable

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

Supports typical HPC languages & parallel programming paradigms

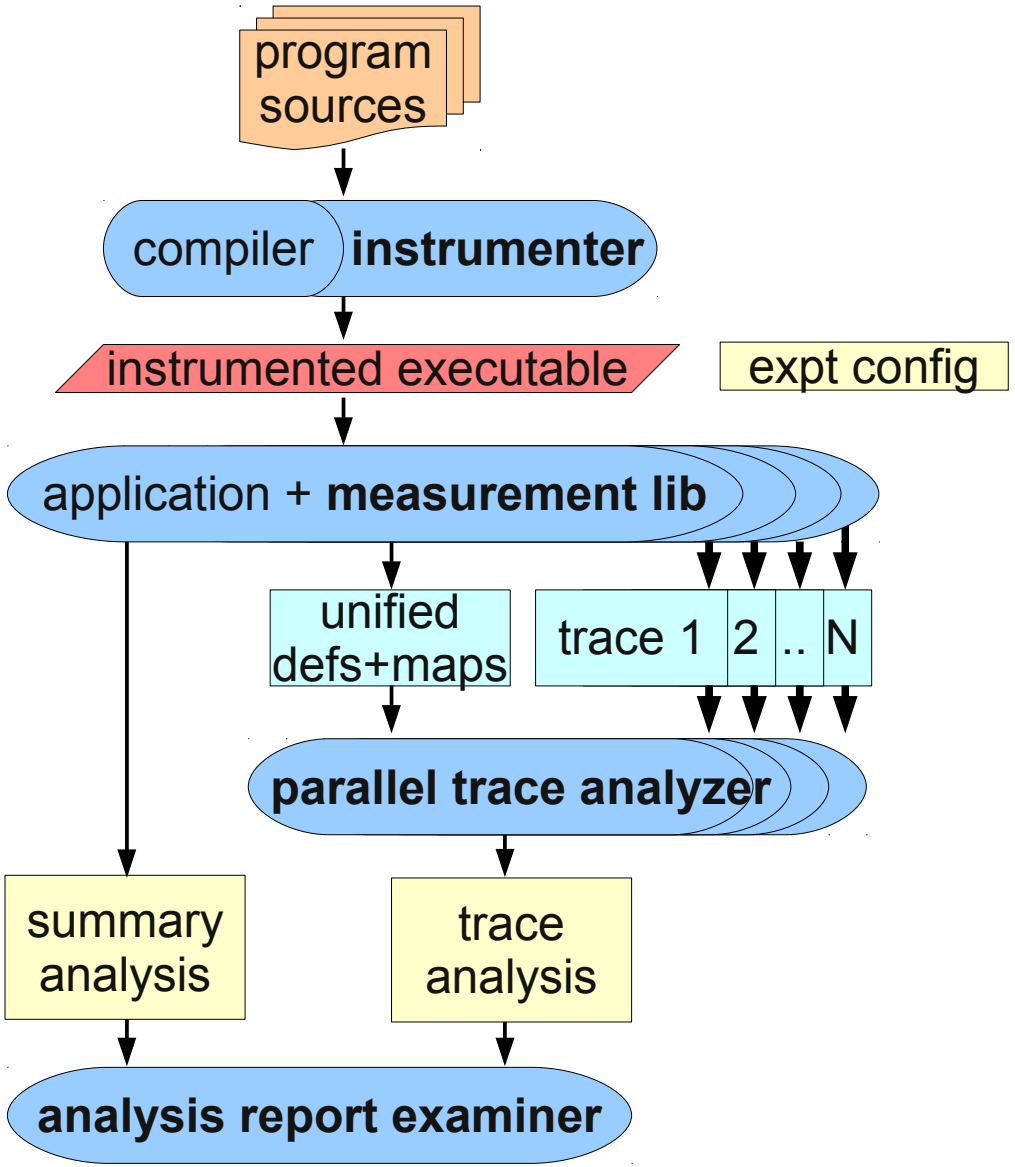
- Fortran, C, C++
- MPI, OpenMP & hybrid MPI+OpenMP

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 (commands)

1. Prepare application objects and executable for measurement:
   - **`scalasca -instrument`** `cc -O3 -c …`
   - **`scalasca -instrument`** `ftn -O3 -o pflotran.exe …`
     - *instrumented executable pflotran.exe produced*
2. Run application under control of measurement & analysis nexus:

   BATCH JOB
   - **`scalasca -analyze`** `aprun -N 12 -n 65536 pflotran.exe …`
     - *epik_pflotran_12p65536_sum* experiment produced
   - **`scalasca -analyze -t`** `aprun -N 12 -n 65536 pflotran.exe …`
     - *epik_pflotran_12p65536_trace* experiment produced
3. Interactively explore experiment analysis report:
   - **`scalasca -examine`** `epik_pflotran_12p65536_trace`
     - *epik_pflotran_12p65536_trace/trace.cube.gz* presented

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 measurement 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

Scalasca usage with *PFLOTRAN*

Automatic instrumentation

- both *PFLOTRAN* application (Fortran) & PETSc library (C)
- USR routines instrumented by IBM XL & PGI compilers
- MPI routine interposition with instrumented library (PMPI)

Initial summary measurements used to define filter files specifying all purely computational routines

Summary & trace experiments collected using filters

- parallel trace analysis initiated automatically on same partition

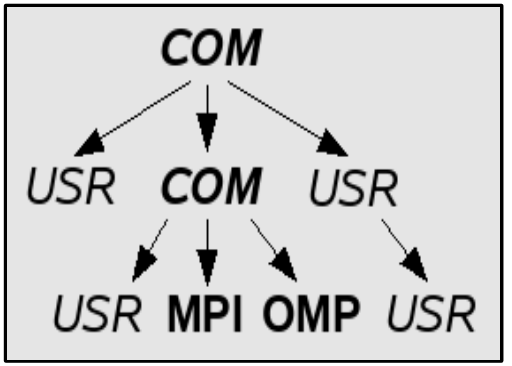
Post-processing of analysis reports

- cut to extract timestep loop; incorporation of application topology

Analysis report examination in GUI

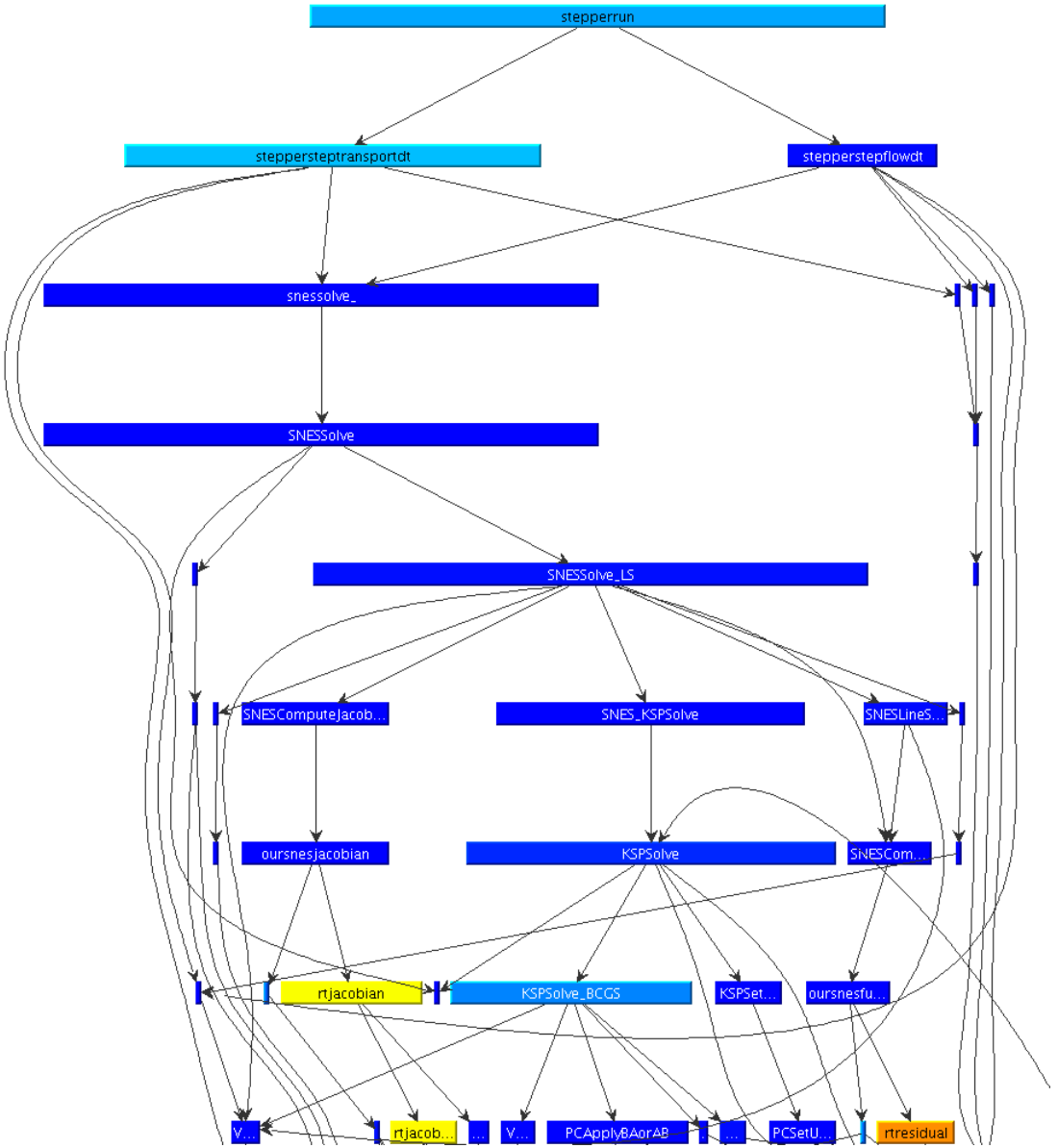
PFLOTRAN structural analysis

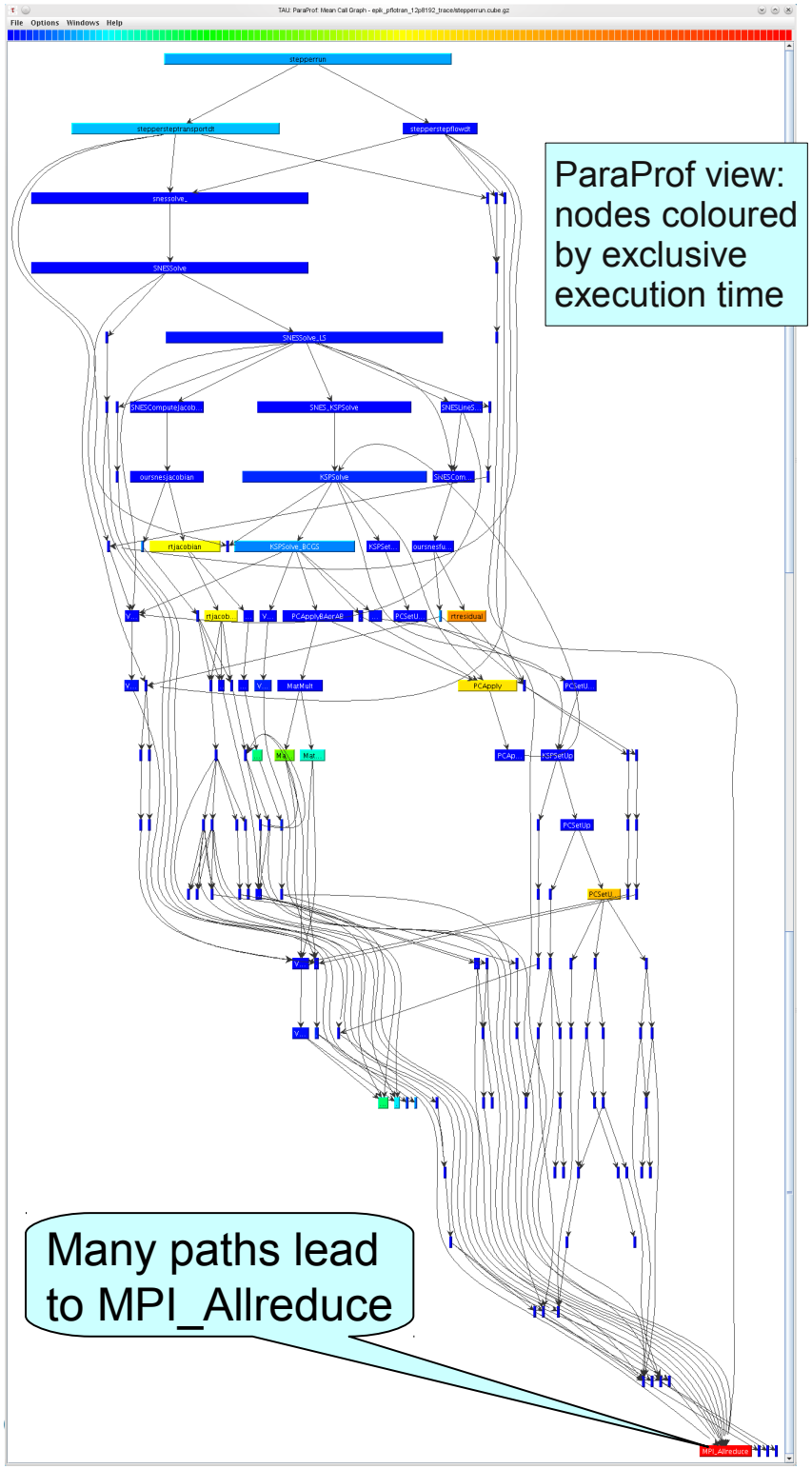
Determined by scoring summary experiment using fully-instrumented application executable



- 29 MPI library routines used
- 1100+ PFLOTRAN & PETSc routines
  - *most not on a callpath to MPI, purely local calculation (USR)*
  - *~250 on callpaths to MPI, mixed calculation & comm. (COM)*
- Using measurement filter listing all USR routines
  - *maximum callpath depth 22 frames*
  - *~1750 unique callpaths (399 in FLOW, 375 in TRAN)*
  - *633 MPI callpaths (121 in FLOW, 114 in TRAN)*
- FLOW callpaths very similar to TRAN callpaths

PFLOTRAN stepperrun callgraph

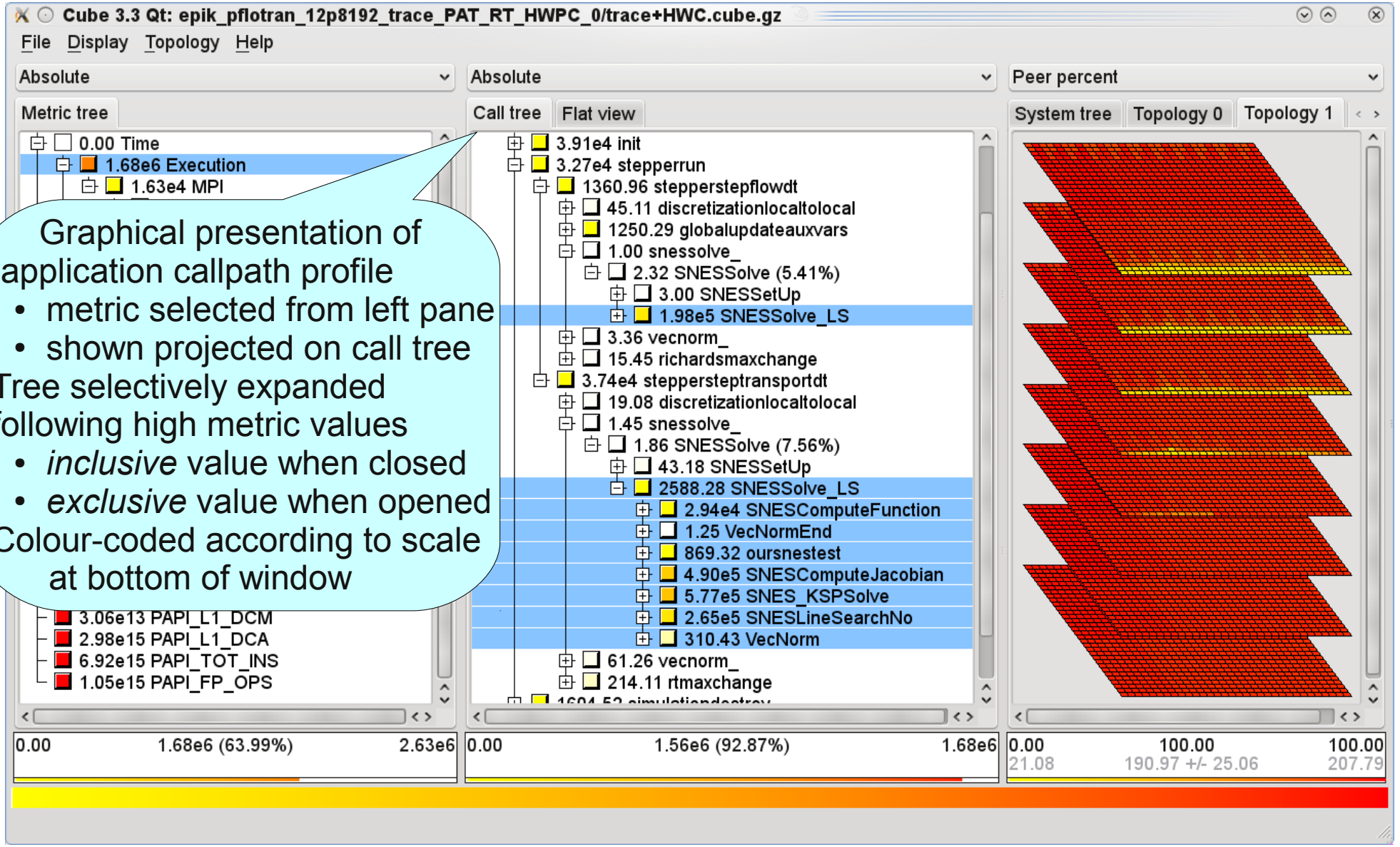




ParaProf view: nodes coloured by exclusive execution time

Many paths lead to MPI_Allreduce

Scalasca analysis report: program call tree



Graphical presentation of application callpath profile

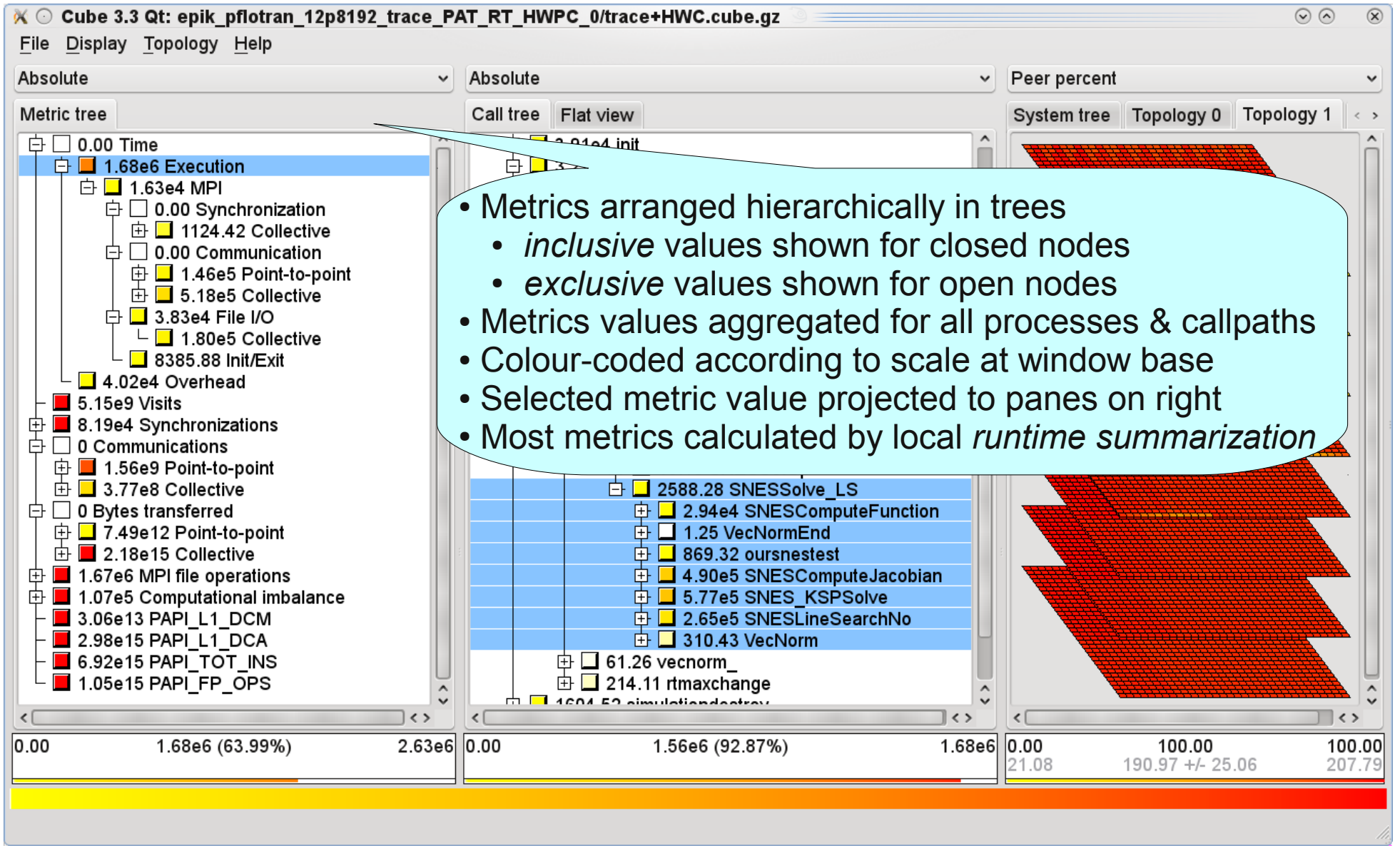
- metric selected from left pane
- shown projected on call tree

Tree selectively expanded following high metric values

- *inclusive* value when closed
- *exclusive* value when opened

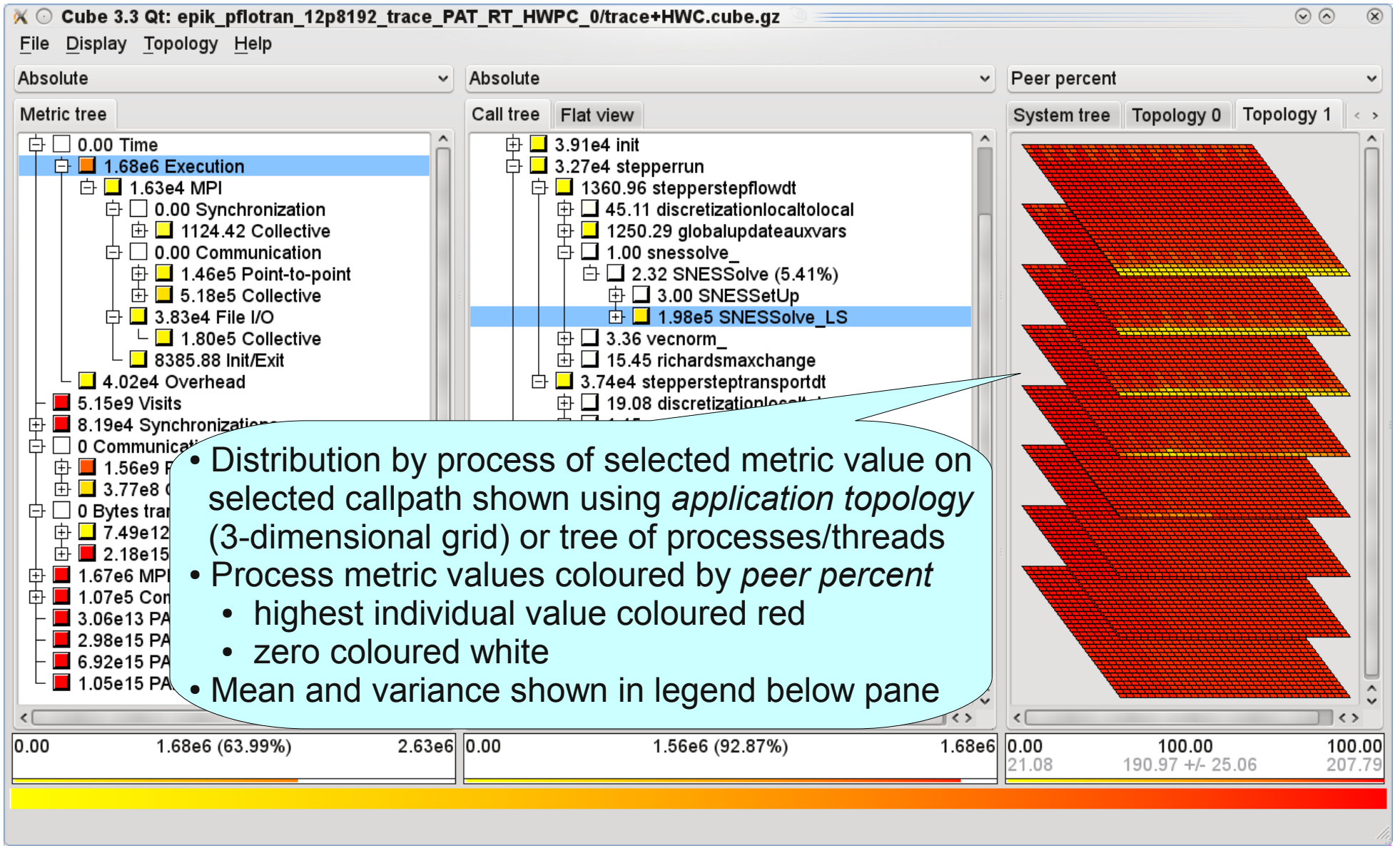
Colour-coded according to scale at bottom of window

Scalasca analysis report: performance metric tree



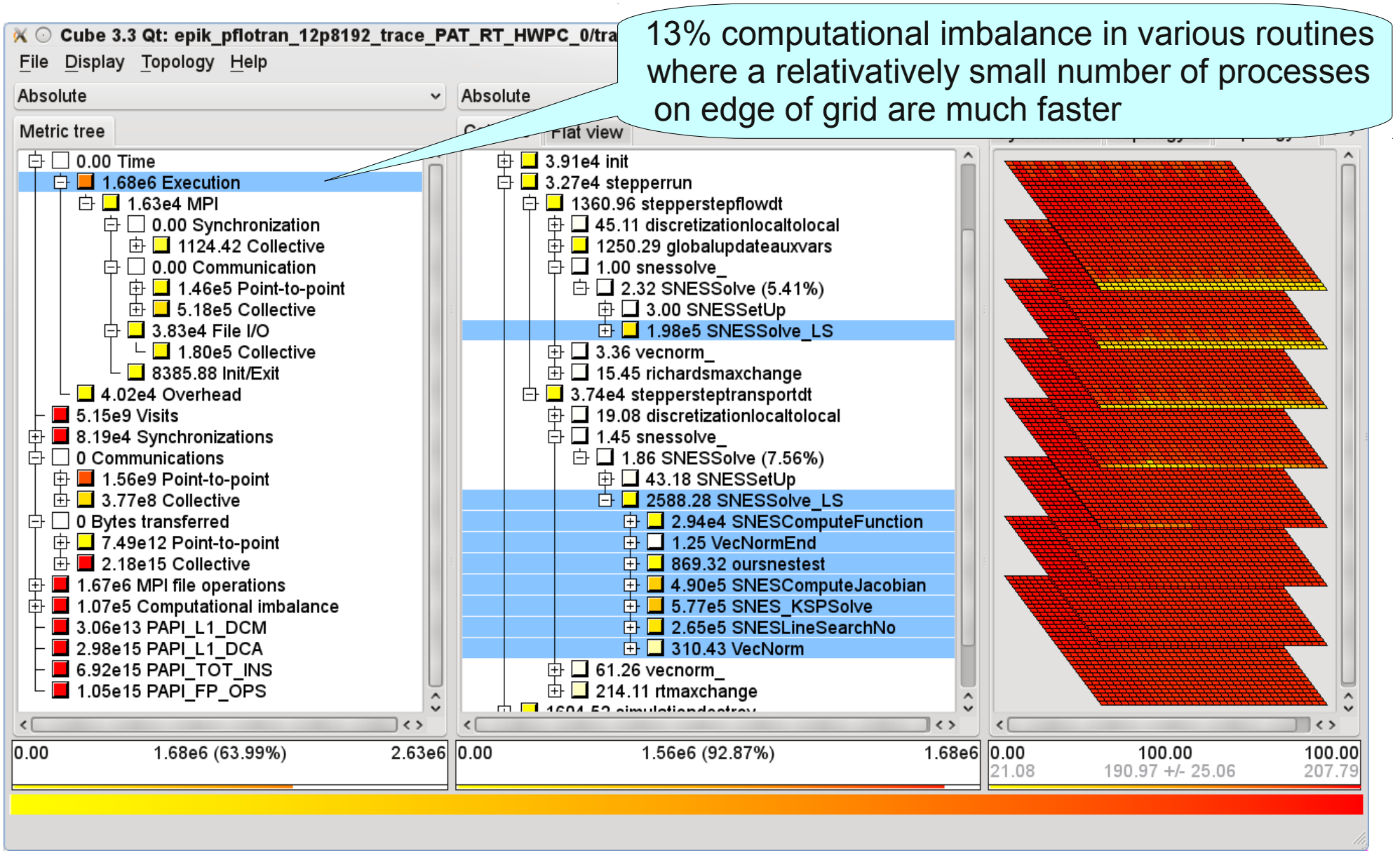
- Metrics arranged hierarchically in trees
  - *inclusive* values shown for closed nodes
  - *exclusive* values shown for open nodes
- Metrics values aggregated for all processes & callpaths
- Colour-coded according to scale at window base
- Selected metric value projected to panes on right
- Most metrics calculated by local *runtime summarization*

Scalasca analysis report: system tree/topology

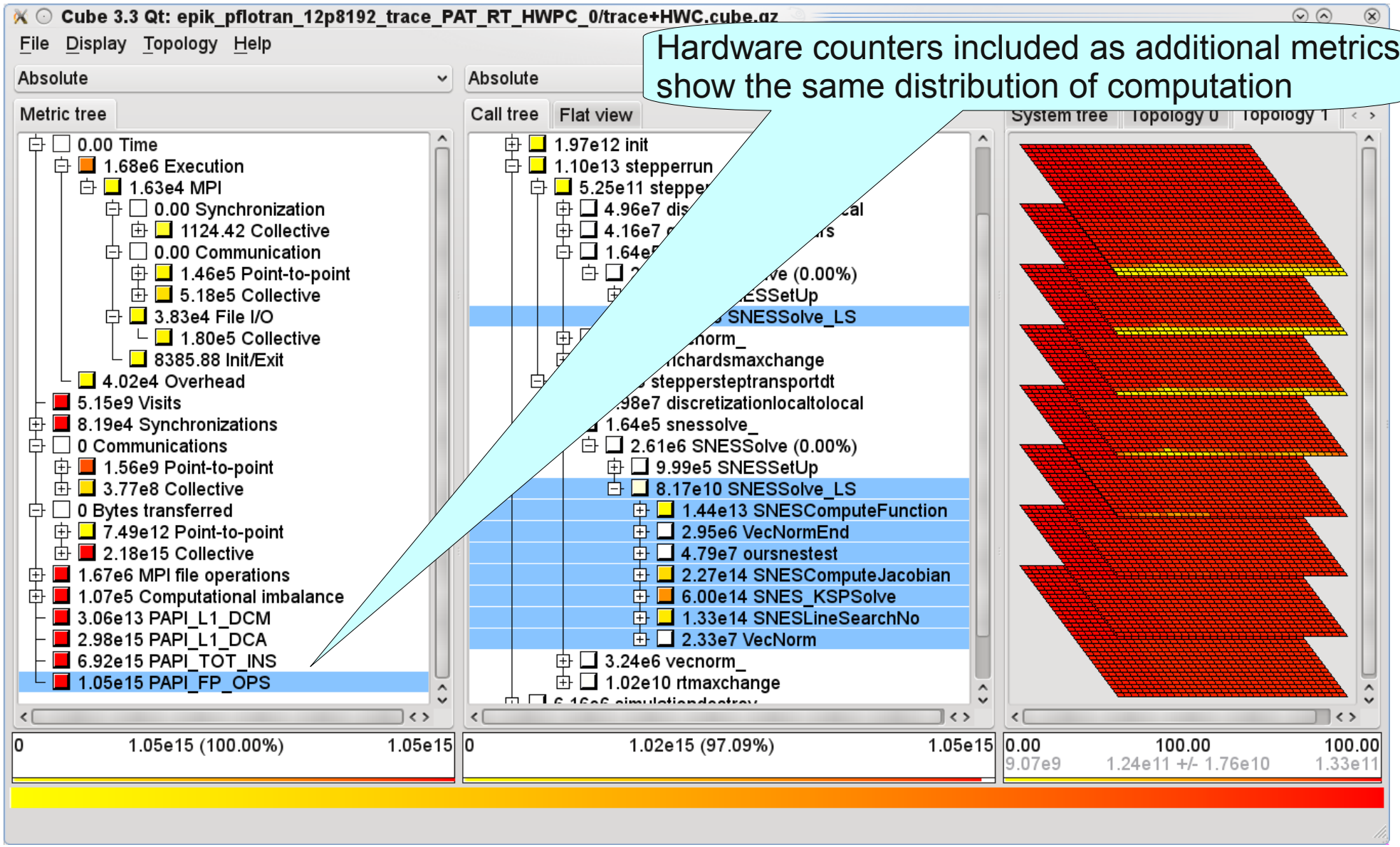


- Distribution by process of selected metric value on selected callpath shown using *application topology* (3-dimensional grid) or tree of processes/threads
- Process metric values coloured by *peer percent*
  - highest individual value coloured red
  - zero coloured white
- Mean and variance shown in legend below pane

Exclusive Execution time

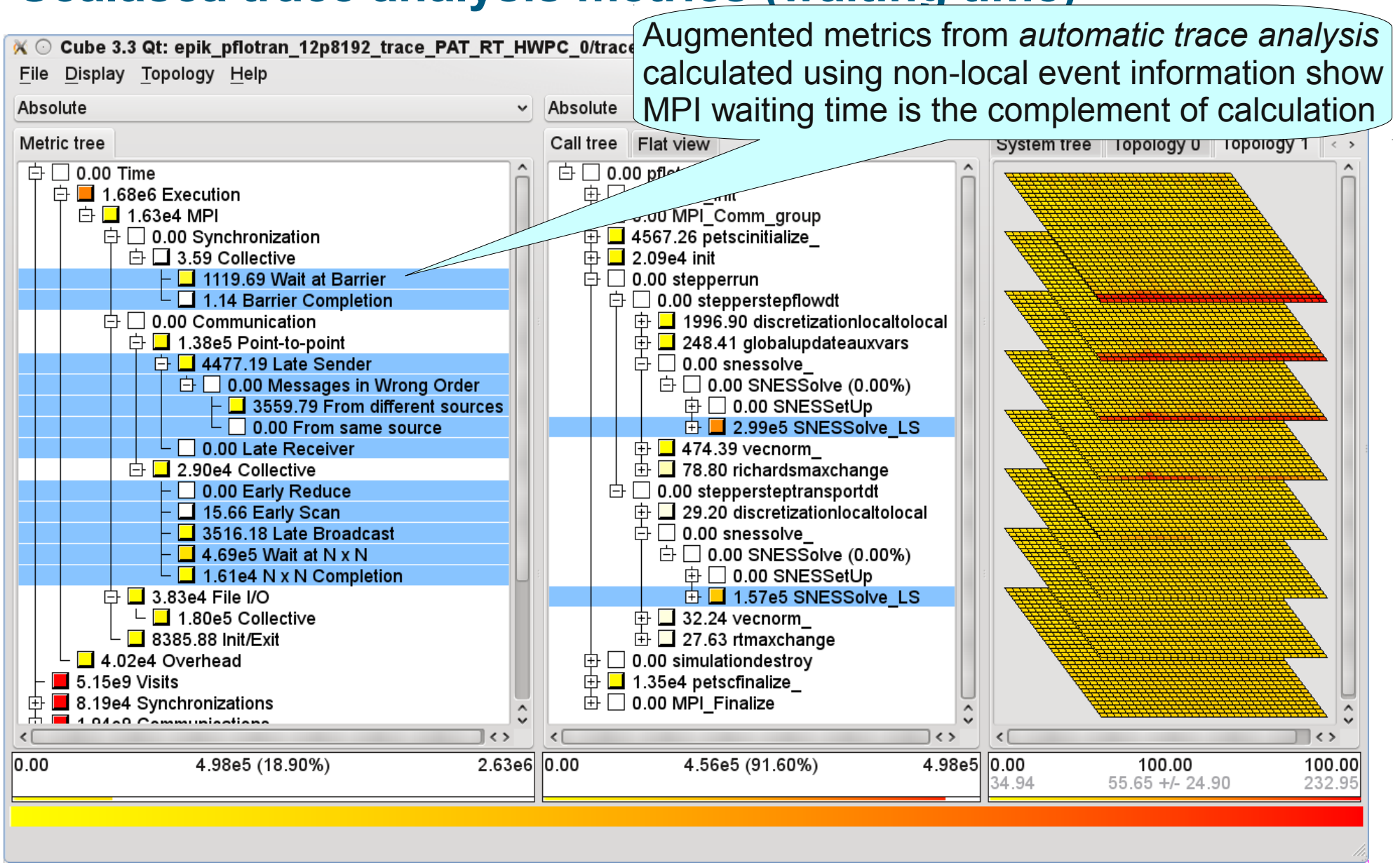


Floating-point operations

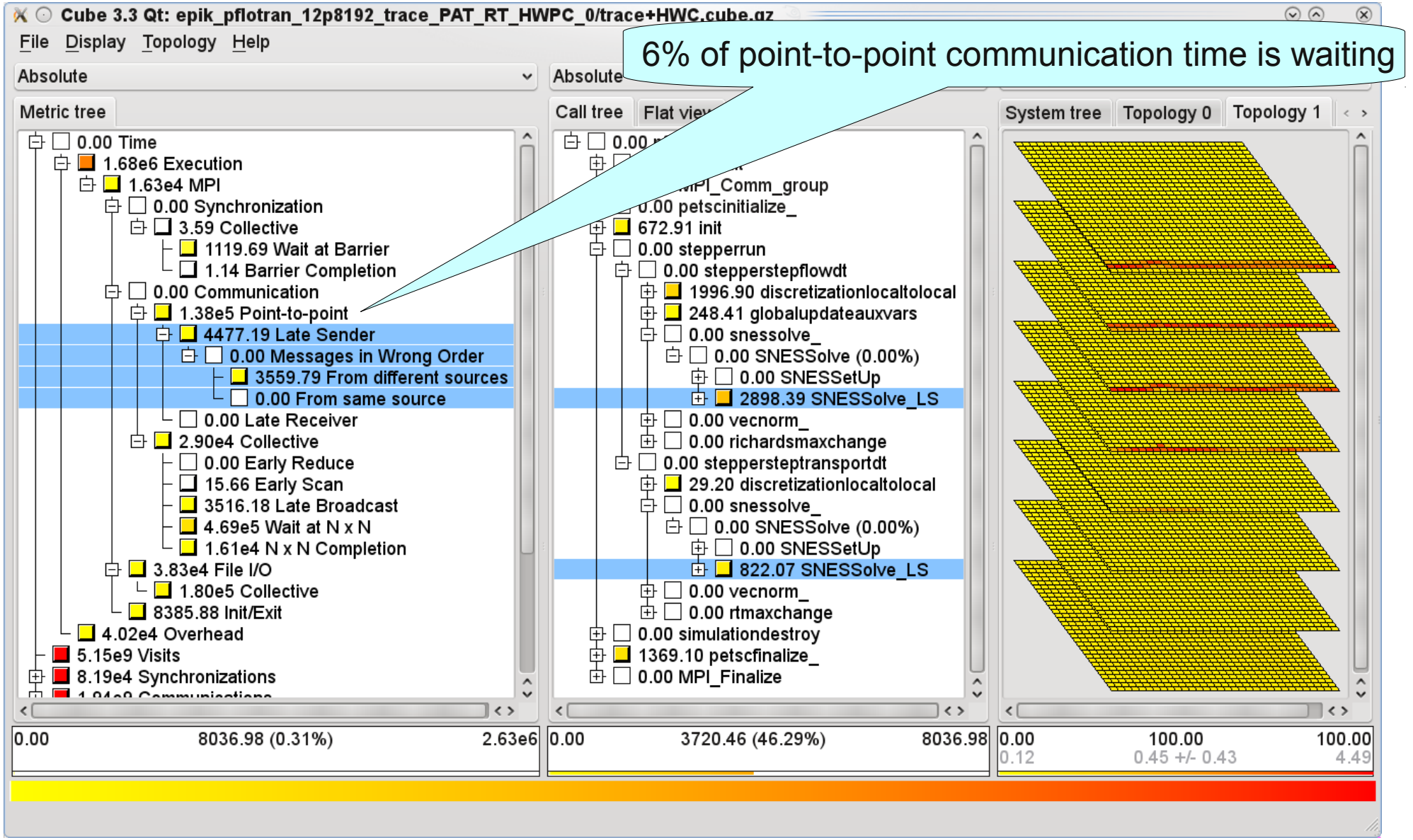


Hardware counters included as additional metrics show the same distribution of computation

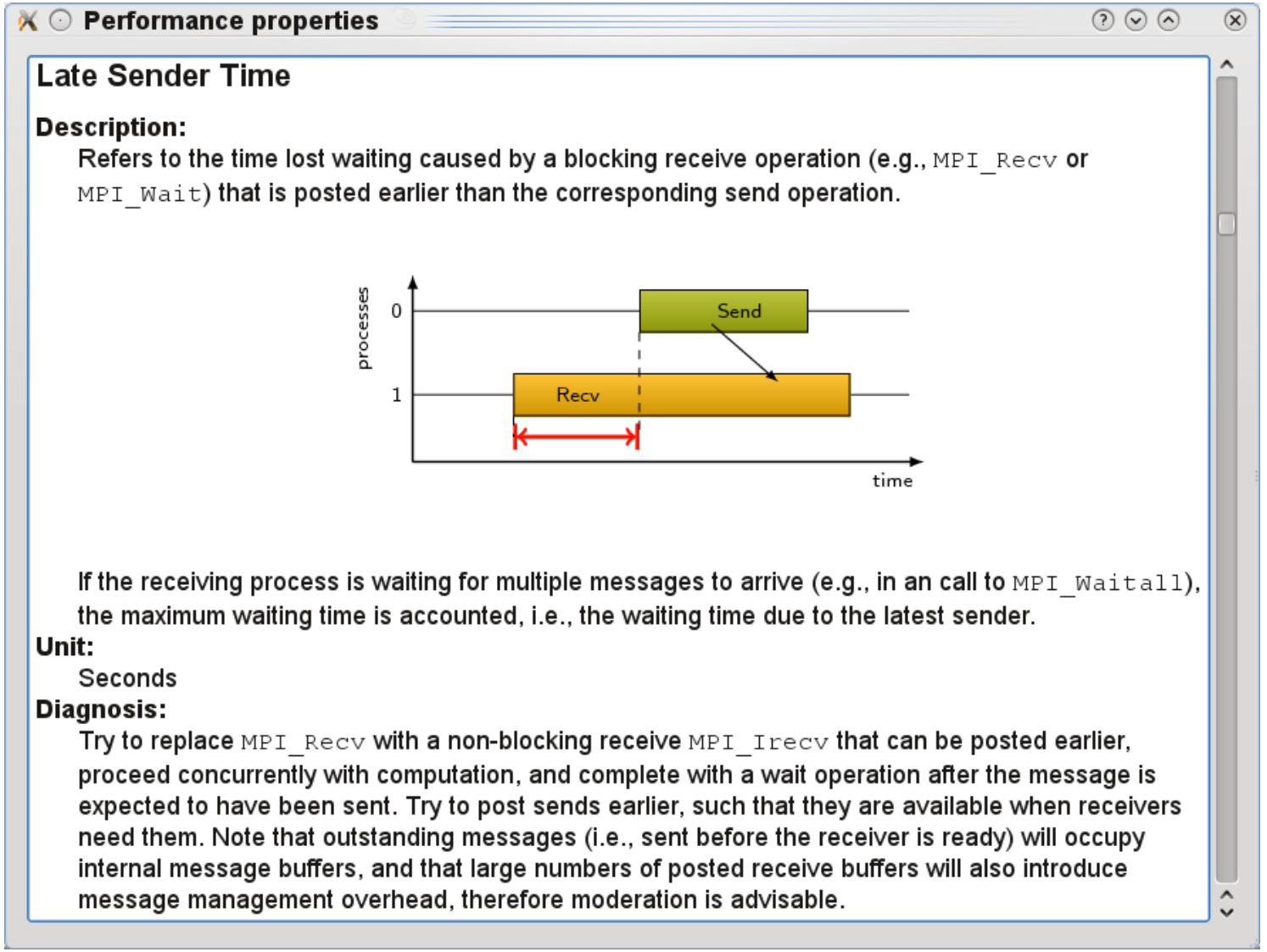
Scalasca trace analysis metrics (waiting time)



Late Sender time (for early receives)

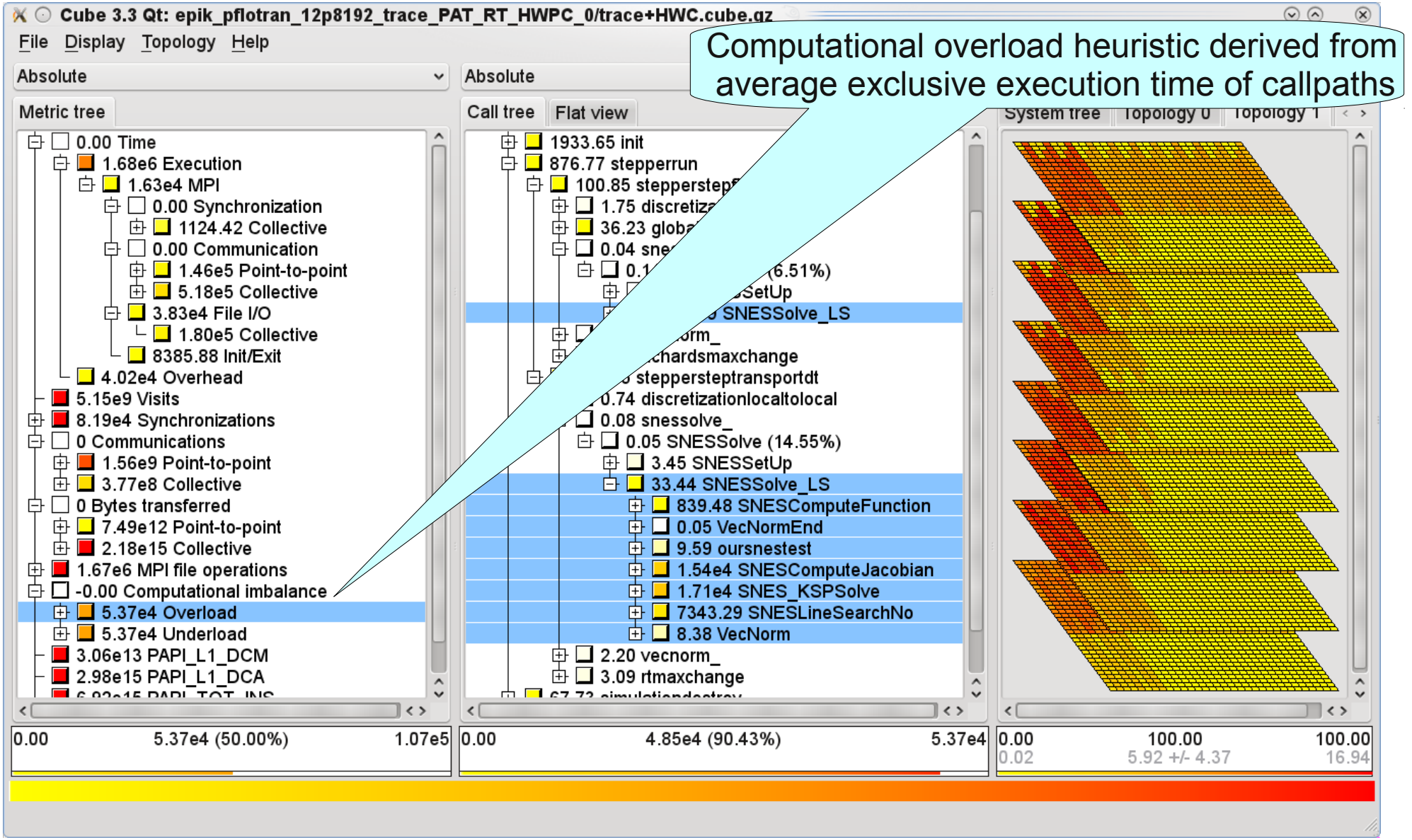


Scalasca description for *Late Sender* metric



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

Computational imbalance: overload



PFLOTRAN grid decomposition imbalance

850x1000x160 cells decomposed on 65536=64x64x16 process grid

- *x*-axis: 850/64=13 plus 18 extra cells
- *y*-axs: 1000/64=15 plus 40 extra cells
- *z*-axis: 160/16=10

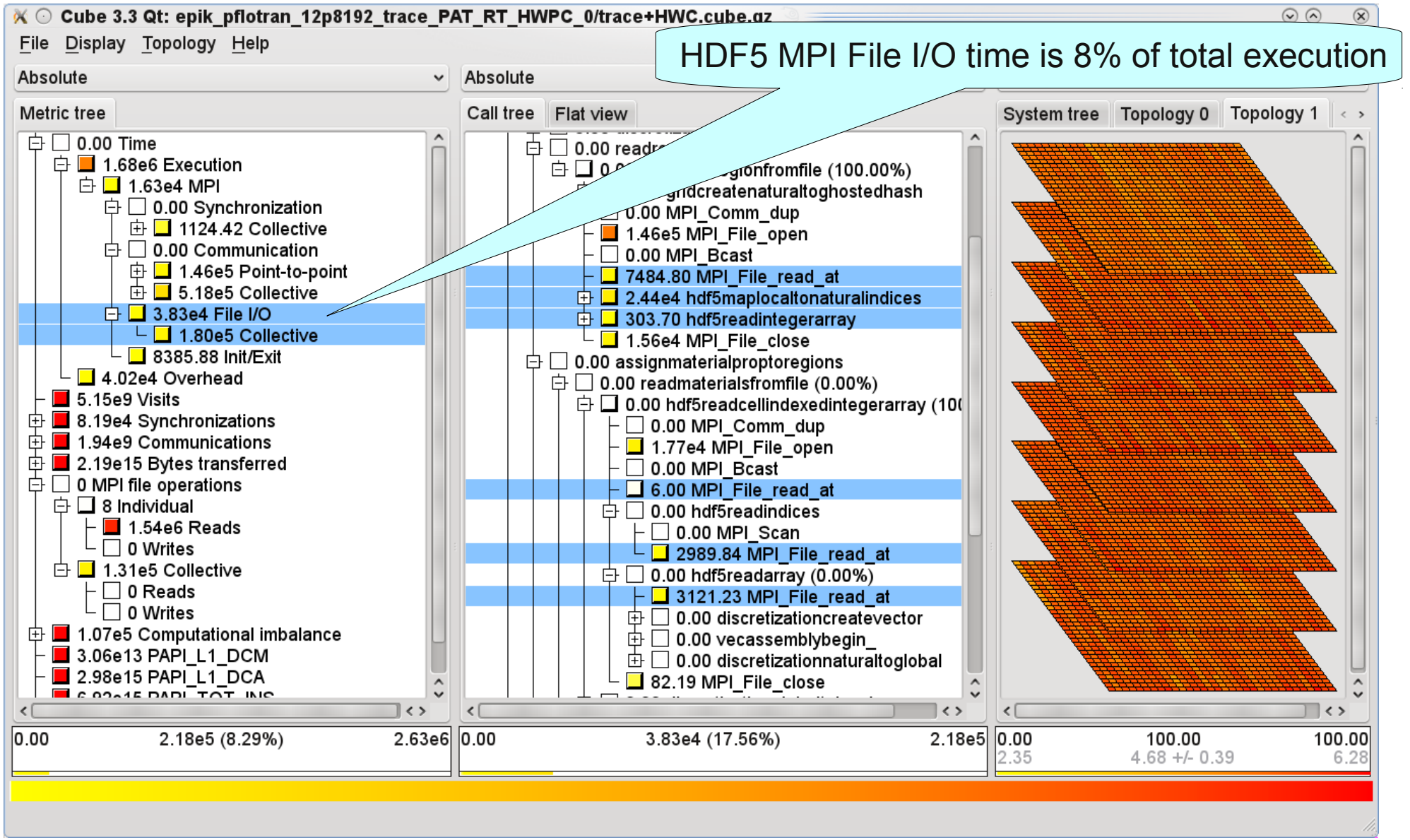
Perfect distribution would be 2075.2 cells per process

- but 20% of processes in each *z*-plane have 2240 cells
- 8% computation overload manifests as waiting times at the next communication/synchronization on the other processes

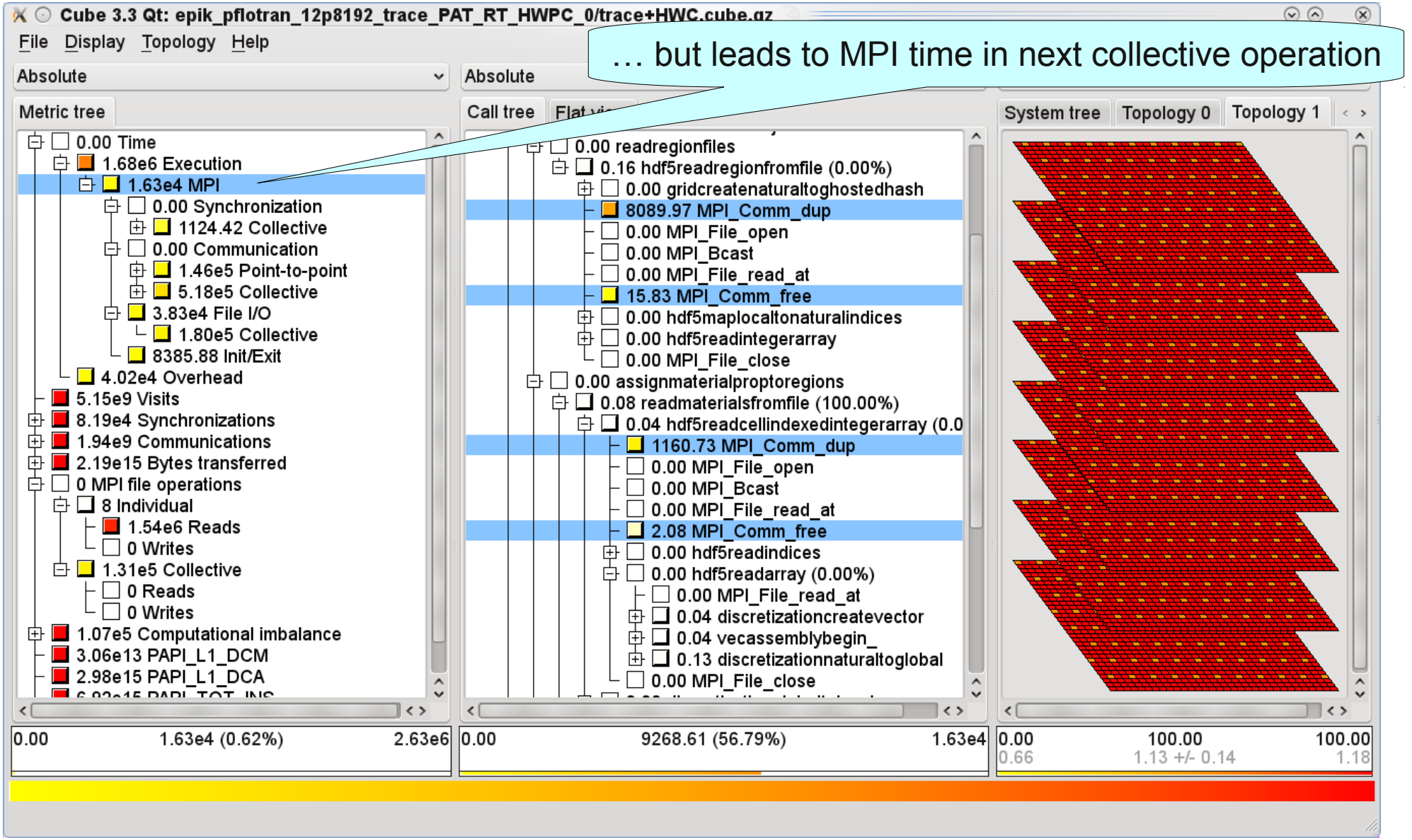
The problem-specific localized imbalance in the river channel is minor

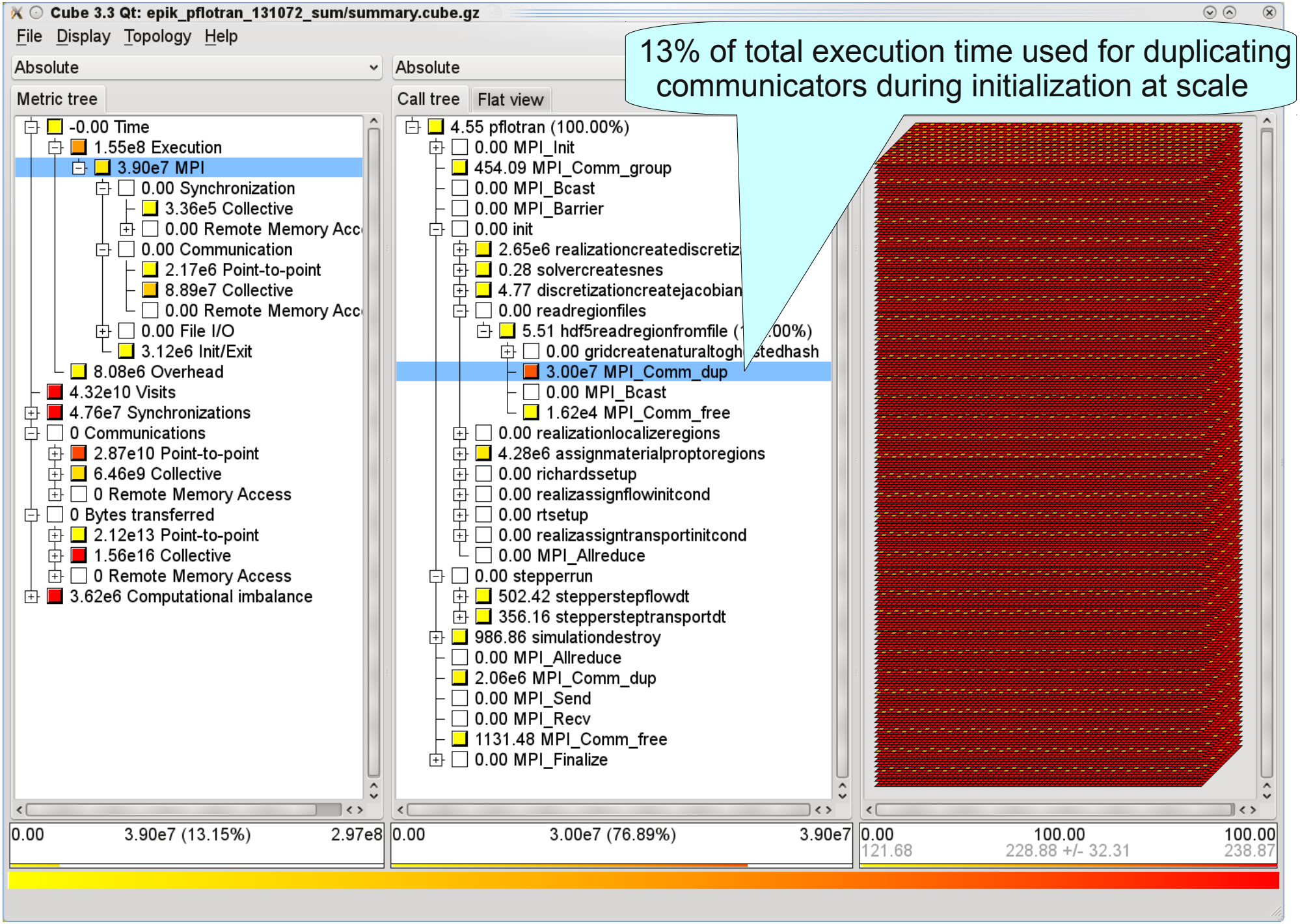
- reversing the assignments in the *x*-dimension won't help much since some of the *z*-planes have no inactive cells

MPI File I/O time



MPI Communicator duplication




Cube 3.3 Qt: epik_pflotran_131072_sum/summary.cube.gz
File Display Topology Help
13% of total execution time used for duplicating communicators during initialization at scale
Absolute
Absolute
Metric tree
Call tree
Flat view
-0.00 Time
1.55e8 Execution
3.90e7 MPI
0.00 Synchronization
3.36e5 Collective
0.00 Remote Memory Acc
0.00 Communication
2.17e6 Point-to-point
8.89e7 Collective
0.00 Remote Memory Acc
0.00 File I/O
3.12e6 Init/Exit
8.08e6 Overhead
4.32e10 Visits
4.76e7 Synchronizations
0 Communications
2.87e10 Point-to-point
6.46e9 Collective
0 Remote Memory Access
0 Bytes transferred
2.12e13 Point-to-point
1.56e16 Collective
0 Remote Memory Access
3.62e6 Computational imbalance
4.55 pflotran (100.00%)
0.00 MPI_Init
454.09 MPI_Comm_group
0.00 MPI_Bcast
0.00 MPI_Barrier
0.00 init
2.65e6 realizationcreatediscretiz
0.28 solvercreatesnes
4.77 discretizationcreatejacobian
0.00 readregionfiles
5.51 hdf5readregionfromfile (
0.00 gridcreatenaturaltogh stedhash
3.00e7 MPI_Comm_dup
0.00 MPI_Bcast
1.62e4 MPI_Comm_free
0.00 realizationlocalizeregions
4.28e6 assignmaterialproptoregions
0.00 richardssetup
0.00 realizassignflowinitcond
0.00 rtsetup
0.00 realizassigntransportinitcond
0.00 MPI_Allreduce
0.00 stepperrun
502.42 stepperstepflowdt
356.16 steppersteptransportdt
986.86 simulationdestroy
0.00 MPI_Allreduce
2.06e6 MPI_Comm_dup
0.00 MPI_Send
0.00 MPI_Recv
1131.48 MPI_Comm_free
0.00 MPI_Finalize
0.00 3.90e7 (13.15%) 2.97e8
0.00 3.00e7 (76.89%) 3.90e7
0.00 100.00 100.00
121.68 228.88 +/- 32.31 238.87

Conclusions

Very complex applications like *PFLOTRAN* provide significant challenges for performance analysis tools

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

- works across Cray, IBM BlueGene & many other HPC systems
- analysis reports and event traces can also be examined with complementary third-party tools such as TAU/ParaProf & Vampir
- convenient automatic instrumentation of applications and libraries must be moderated with selective measurement filtering
  - *analysis reports can still become awkwardly large*

Scalasca is continually improved in response to the evolving requirements of application developers and analysts

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, CINECA, DKRZ, EPCC, HLRN, HLRS, ICL, ICM, IMAG, JSC, KAUST, KTH, LRZ, NCAR, NCCS, NICS, NLHPC, RWTH, RZG, SARA, TACC, ZIH
- Access & usage supported by European Union, German and other national funding organizations

Scalasca users who have provided valuable feedback and suggestions for improvements

JÜLICH FORSCHUNGSZENTRUM

scalasca

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

- 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 VI-HPS Parallel Productivity Tools Live-DVD
- sources, documentation & publications:
  - *http://www.scalasca.org*
  - *mailto: scalasca@fz-juelich.de*

VI-HPS/Scalasca training

Tutorials

- full- or part-day hands-on training
- *PRACE Spring School*, Kraków, Poland: 18 May 2012
- *19th European MPI User's Meeting*, Vienna, Austria: 23 Sept 2012

VI-HPS Tuning Workshops

- Multi-day hands-on training with participants' own application codes, in context of Virtual Institute – High Productivity Supercomputing
- *9th VI-HPS Tuning Workshop*, St-Quentin, France: 23-27 Apr 2012
- *10th VI-HPS Tuning Workshop*, Garching, Germany: 16-19 Oct 2012

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

POINT/VI-HPS Linux Live-DVD/ISO

Bootable Linux installation on DVD or USB memory stick

Includes everything needed to try out parallel tools on an x86-architecture notebook computer

- VI-HPS tools: KCachegrind, Marmot, PAPI, Periscope, Scalasca, TAU, VT/Vampir*
- Also: Eclipse/PTP, TotalView*, etc.
  - ** time/capability-limited evaluation licenses provided*
- GCC (w/OpenMP), OpenMPI
- User Guides, exercises & examples

http://www.vi-hps.org/training/material/