Mitglied der Helmholtz-Gemeinschaft

SC12 Salt Lake City, Utah

JÜLICH FORSCHUNGSZENTRUM

Parallel performance measurement & analysis scaling lessons

2012-11-16 |

Brian J. N. Wylie
Jülich Supercomputing Centre

b.wylie@fz-juelich.de

Overview

Scaling from 2^10 to 2^20 (one thousand to one million)

KOJAK to Scalasca

10 key scaling lessons

Current/future challenges

Conclusions

JSC tools scalability challenge

2003: IBM SP2 p690+ 1312 cores (dual-core POWER4+ processors)

- almost exclusively programmed with MPI
  - *some pure OpenMP with up to 16 threads within SMP nodes*

2006: IBM BlueGene/L 16,384 cores (dual-core PowerPC 440)

2009: IBM BlueGene/P 294,912 cores (quad-core PowerPC 450)

2012: IBM BlueGene/Q 393,216 cores (16-core Power A2)

- hardware support for *1.5 million* threads (64-way SMP nodes)
- most applications combine MPI and OpenMP

Scalasca toolset developed from predecessor KOJAK toolset to support performance analysis of increasingly large-scale parallel applications

What needs to scale?

Techniques that had been established for O(1024) processes/threads needed re-evaluation, re-design & re-engineering each doubling of scale

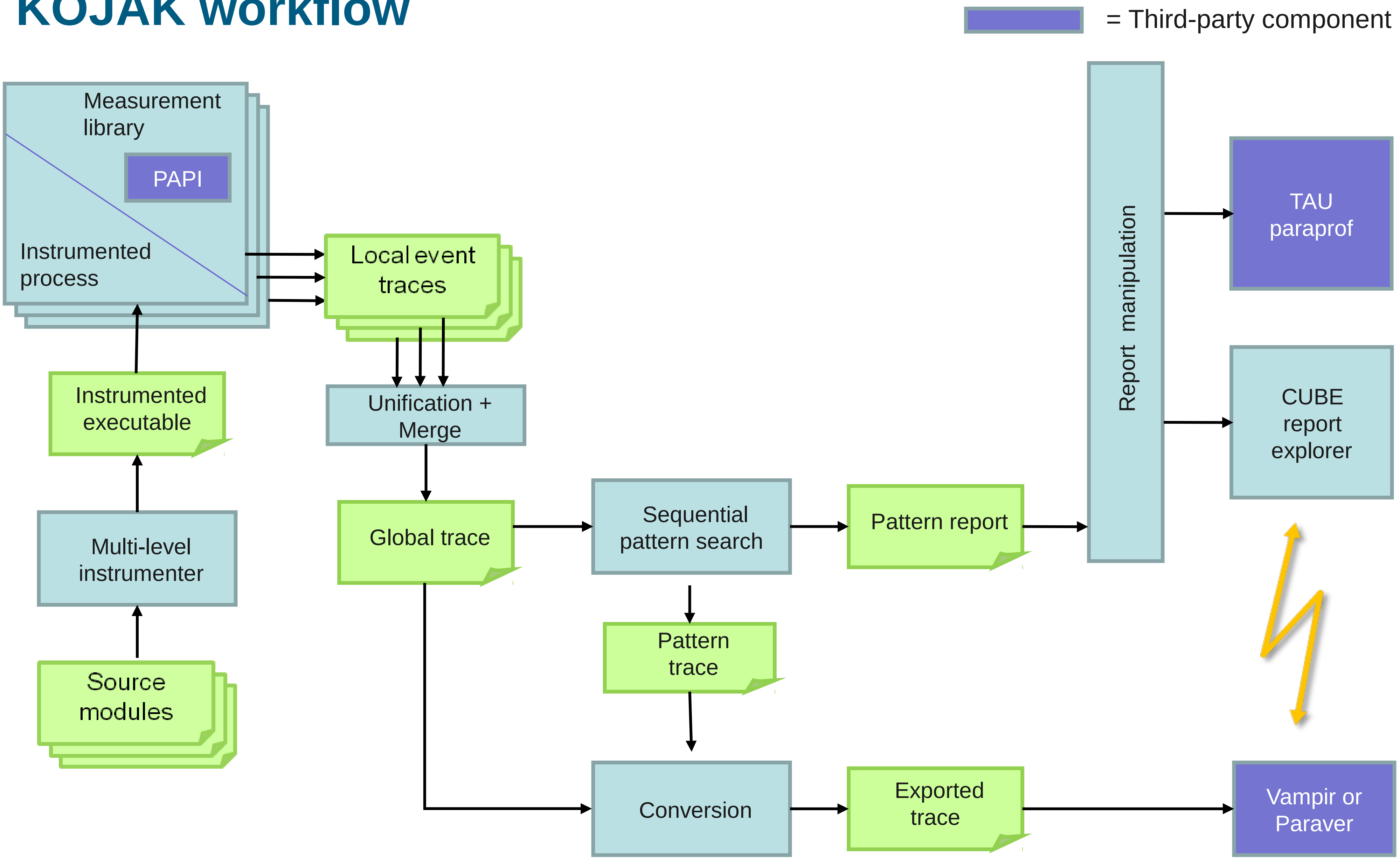
- Instrumentation of application
- Measurement collection
- Analysis of execution
- Examination of analysis results

Scalability of the entire process governed by the least scalable part

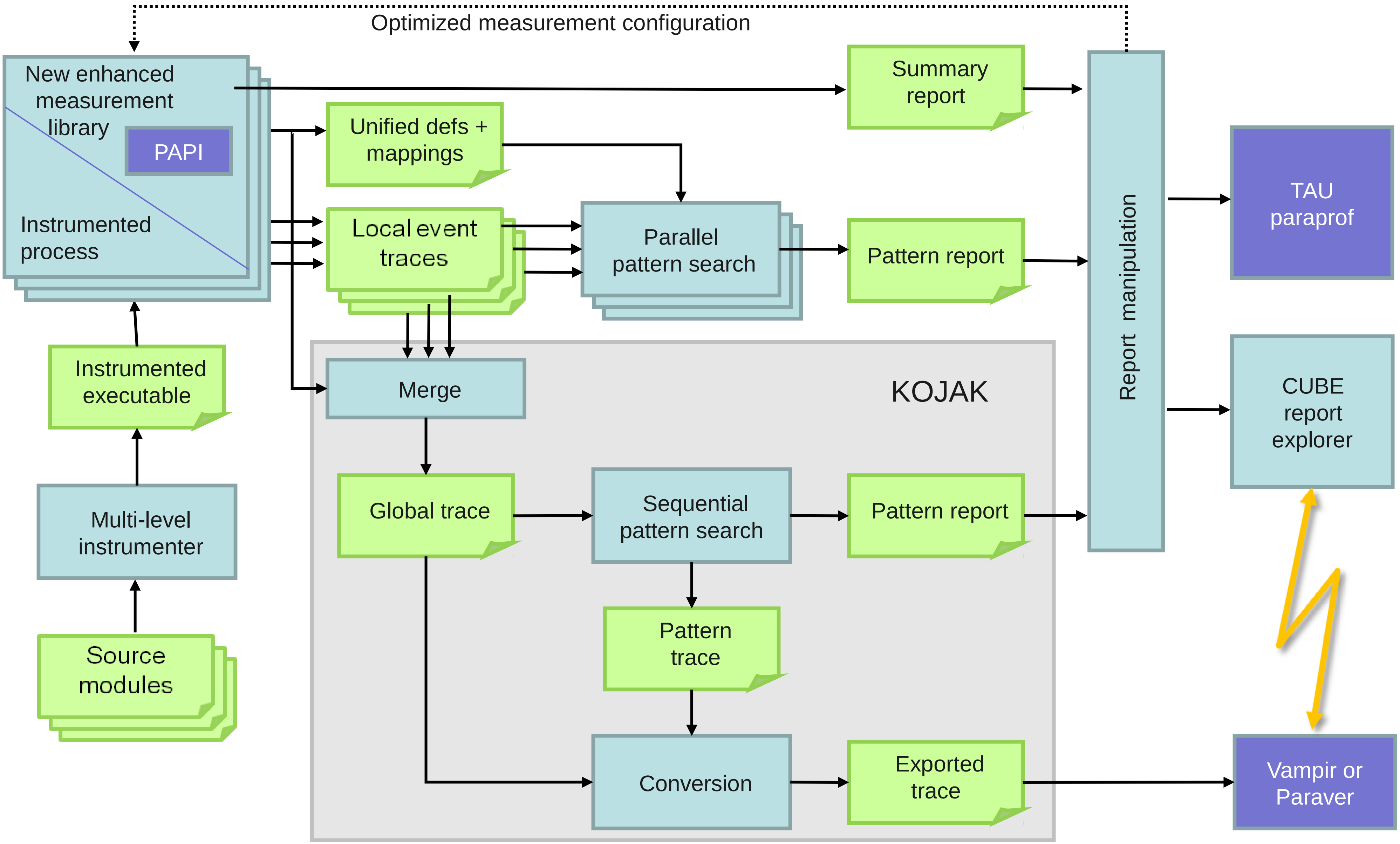
- not every application affected by each issue (to the same extent)

Applications themselves faced the same scalability challenges and needed similar re-engineering

KOJAK workflow



Scalasca workflow



10 key lessons

11. Collect and analyse measurements in memory
12. Analyse event traces in parallel
13. Avoid re-writing/merging event trace files
14. Avoid creating too many files
15. Manage MPI communicators
16. Unify metadata hierarchically
17. Summarize measurements during collection
18. Present analysis results associated to application/machine topologies
19. Provide statistical summaries
20. Load analysis results on-demand/incrementally

Collect and analyse measurements in memory

Storage required for measurement collection and analysis

- memory buffers for traced events of each thread
- full buffers flushed (asynchronously) to trace files on disk

However

- flushing disturbs measurement
  - *communication partners must wait for flush to complete*
- trace files too large to fit in memory may not be analysable
  - *analysis may require memory several times trace size on disk*

Therefore, specify trace buffer sizes and measurement intervals (with associated instrumentation/filtering) to avoid intermediate buffer flushes

Analyse event traces in parallel

Memory and time for serial trace analysis

- grow with number of processes/threads in measured application

However

- processors and memory available for execution analysis are identical to that for the subject parallel application execution itself
- event records contain the necessary attributes for a parallel replay

Therefore

- re-use allocated machine partition after measurement complete
- use pt2pt/collective operations to communicate partner data
  - *communication/synchronization replay time similar to original*
- [EuroPVM/MPI'06, PARA'06]

Avoid re-writing files

Merging events from separate trace files for each process and thread

- allowed traces to be written independently
- produced a single file and event stream for convenient analysis

However

- the single file becomes extremely large and unmanagable
- only a limited number of files can be opened simultaneously
- write/read/re-write becomes increasingly burdensome
  - *especially slow when using a single filesystem*
- parallel analysis ends up splitting stream again

Therefore write files in a form convenient for (parallel) reading

- [EuroPVM/MPI'06]

Avoid creating too many files

Separate trace files for each process and thread

- allowed traces to be written independently
- and read independently during parallel analysis

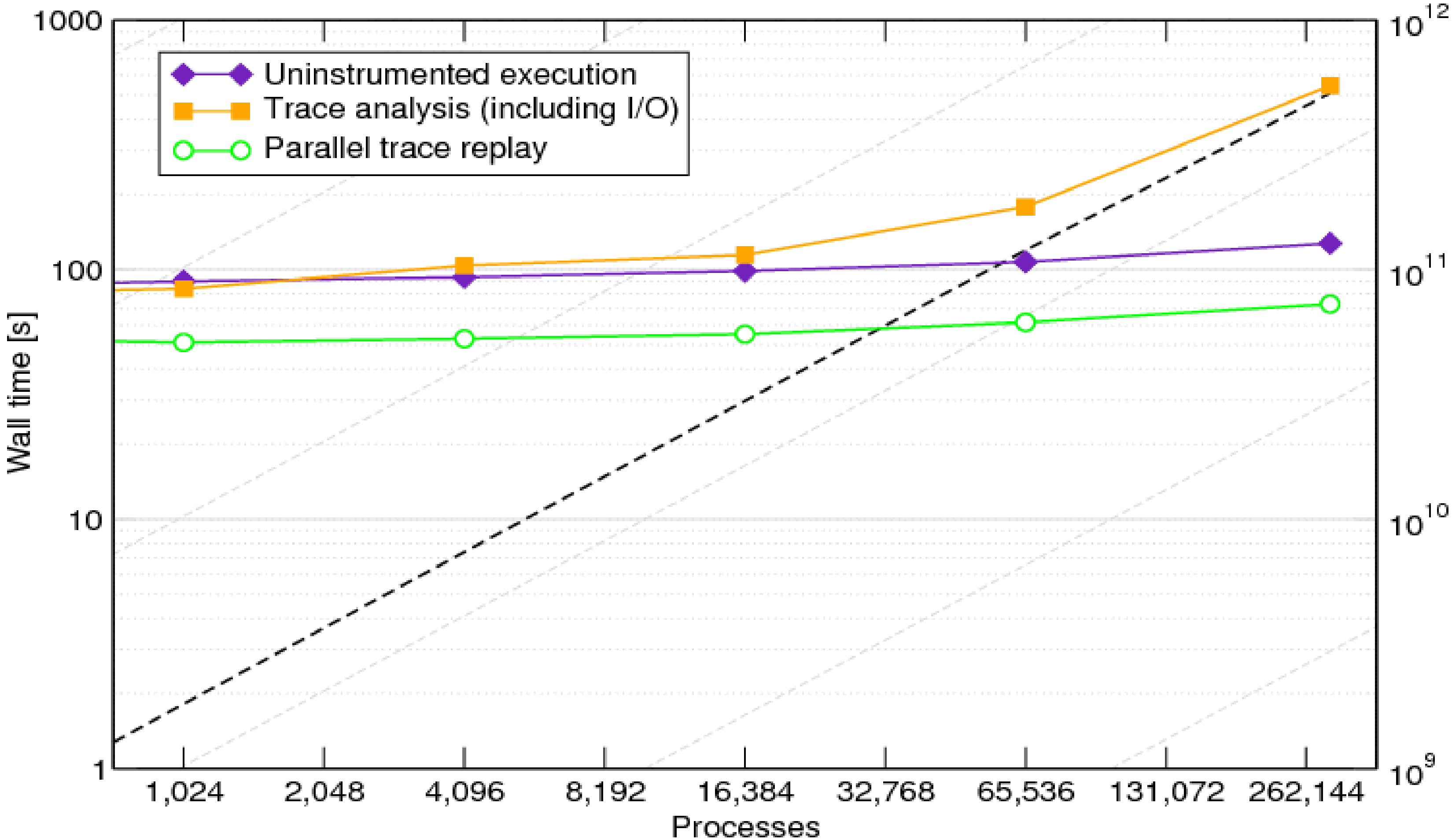
However

- creating the files burdens the filesystem
  - *locking required to ensure directory metadata consistency*
- simultaneous creation typically slower than serialized
- listing/archiving/deleting directories becomes painful

Therefore write filesystem blocks offset in a few multifiles

- [SIONlib, SC'09]

Trace analysis scaling (Sweep3D on BG/P)



- Total trace size (---) increases to 7.6TB for 510G events
- Parallel analysis replay time scales with application execution time

Manage MPI communicators

MPI communicators organise process communication & synchronization

- describe process group membership and ranking for MPI events
  - *MPI_COMM_SELF & MPI_COMM_WORLD are special*
- required for event replay

However

- array representation grows with total number of processes
- cost of translation of local to global rank increases too
  - *MPI_Group_translate_ranks also varies with rank to translate*

Therefore define communicator creation relationship (with special handling of MPI_COMM_SELF) and record events with local ranks (translated when required by analysis)

- [EuroMPI'11]

Unify metadata hierarchically

Merging of individual process definitions and generation of mappings

- allowed event data for traces to be written independently
- provides a consistent unified view of the set

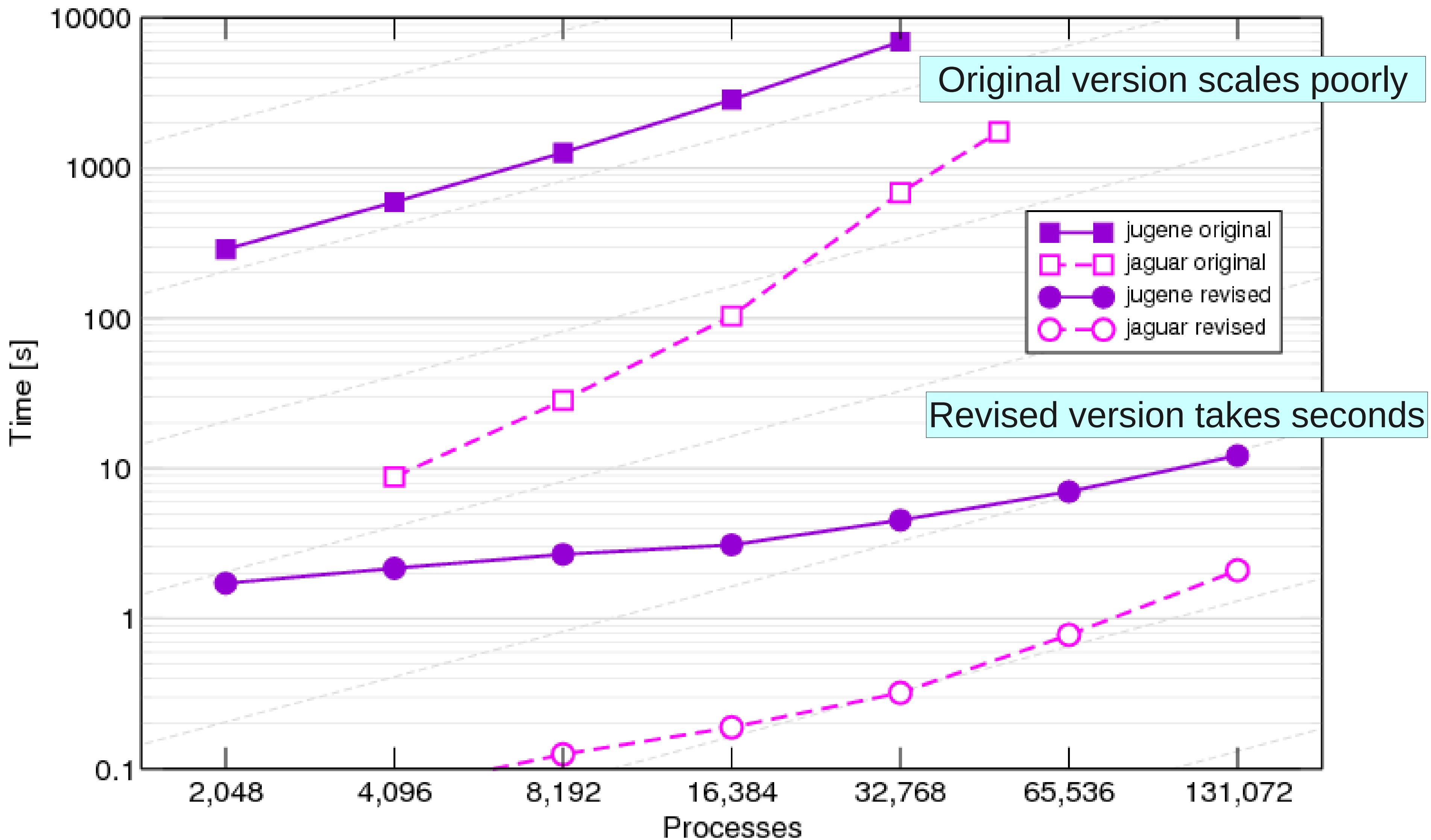
However

- time increases linearly with number of processes if serialized
- or a reduction/multicast infrastructure needs to be overlaid

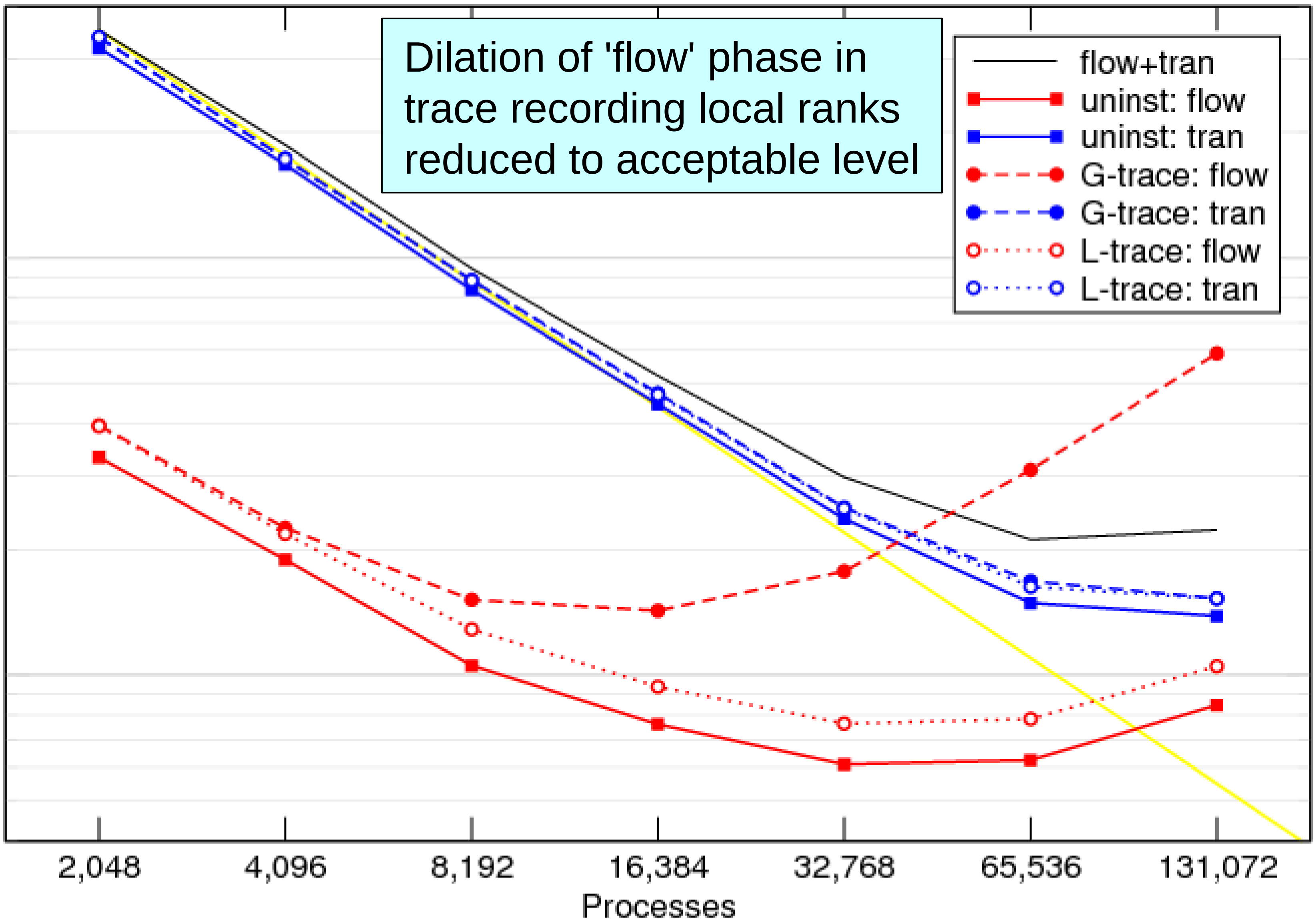
Therefore employ a hierarchical unification scheme during finalization

- [PARA'10, EuroMPI'11]

Improved unification of identifiers (PFLOTRAN)



Reduction of trace measurement dilation (PFLOTRAN)



Summarize measurements during collection

Event trace size grows with duration and level of detail, per thread

- not always practical or productive to record every detail
- overhead for frequent short events particularly counter-productive
  - *may distort timing measurements of interest*

Therefore

- start with per-thread runtime summarization of events
  - *ideal for hardware counter measurements*
- produce aggregated execution profiles to identify events and execution intervals with(out) sufficient value for tracing
  - *filter and pause measurement*
  - *determine buffer/disk storage requirements*
- [PARA'06]

Present analysis results associated with topology

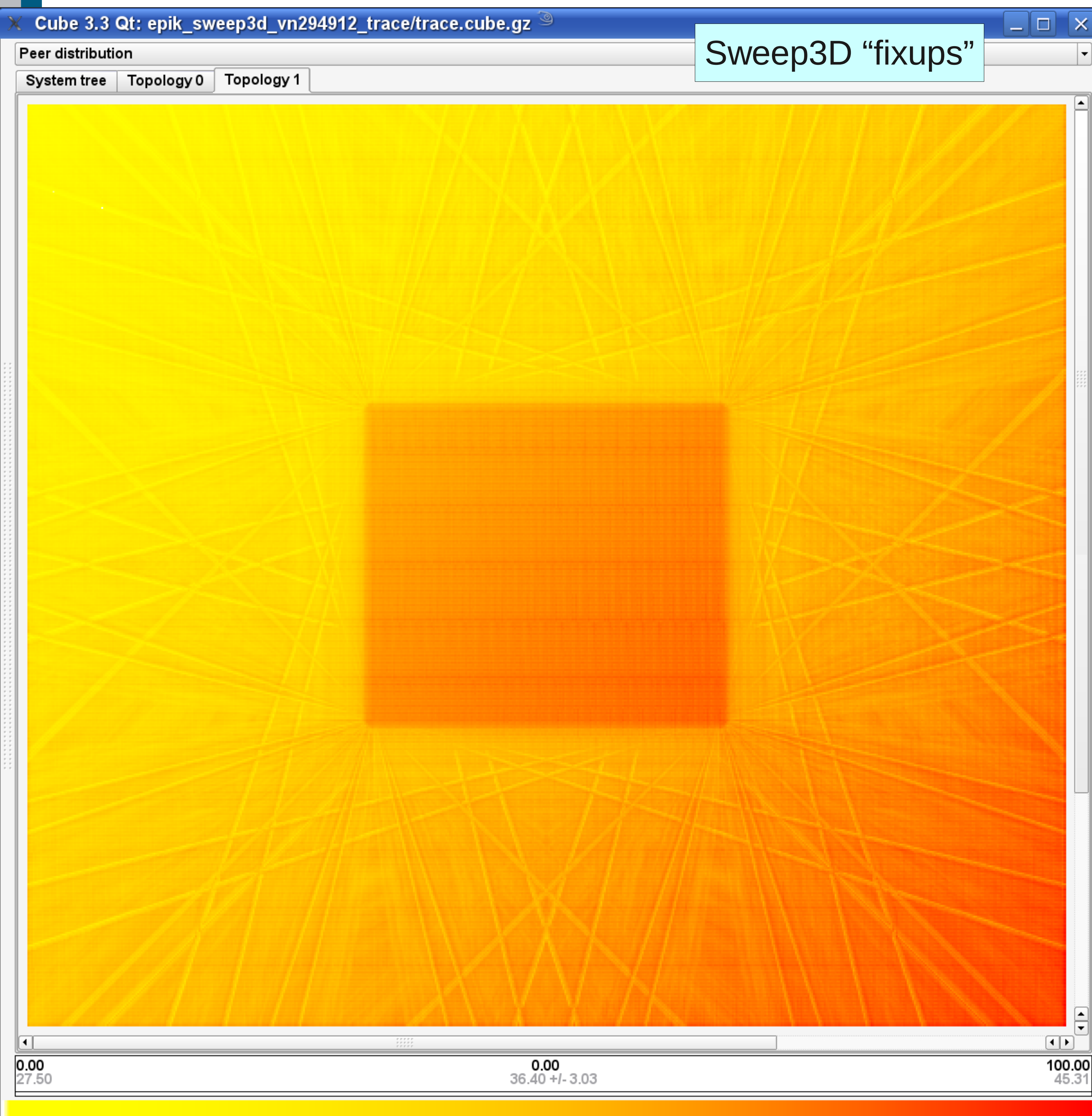
Process and thread ranks are only one aspect of application execution

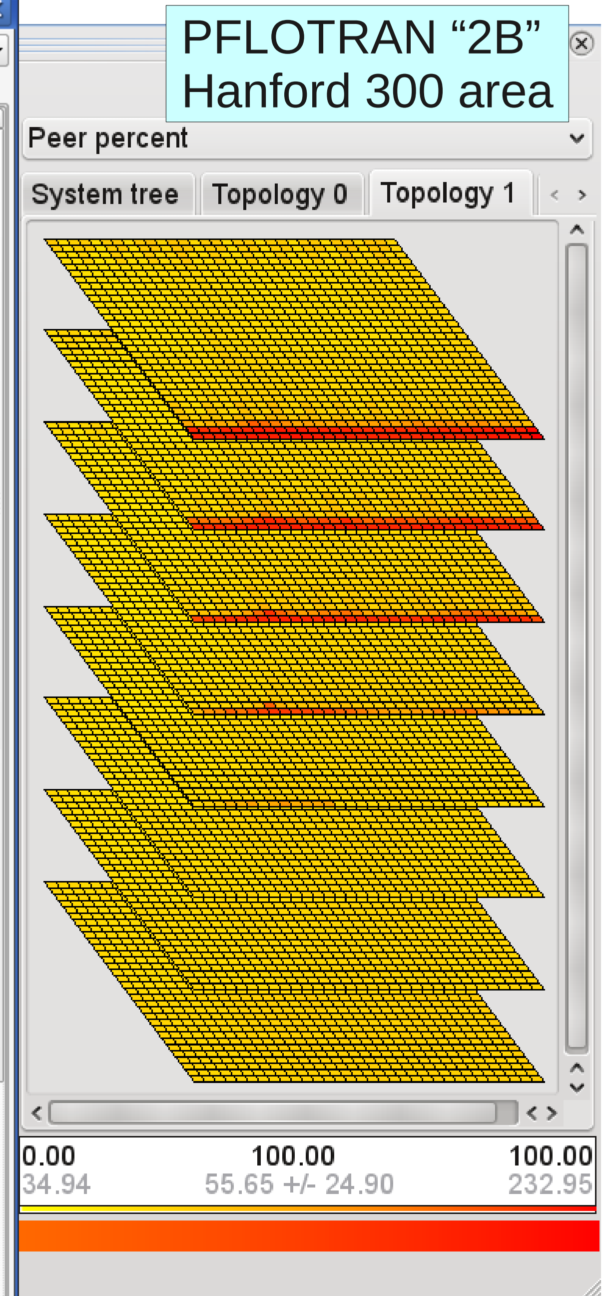
- presentation is natural but not particularly scalable
- complemented with application and machine topologies
  - *often make execution performance metrics more accessible*

Therefore

- record topologies as an integral part of measurements
- allow additional topologies (and mappings) to be manually defined
- allow topologies to be interactively adjusted
  - *slicing and folding of high-dimensional topologies*

Example: Sweep3D, PFLOTRAN, COSMO, WRF, ...





Application topologies

Provide statistical summaries

Presentation of metric values for all processes/threads individually

- provides a good overview to identify distribution and imbalance
- allows localization of extreme values

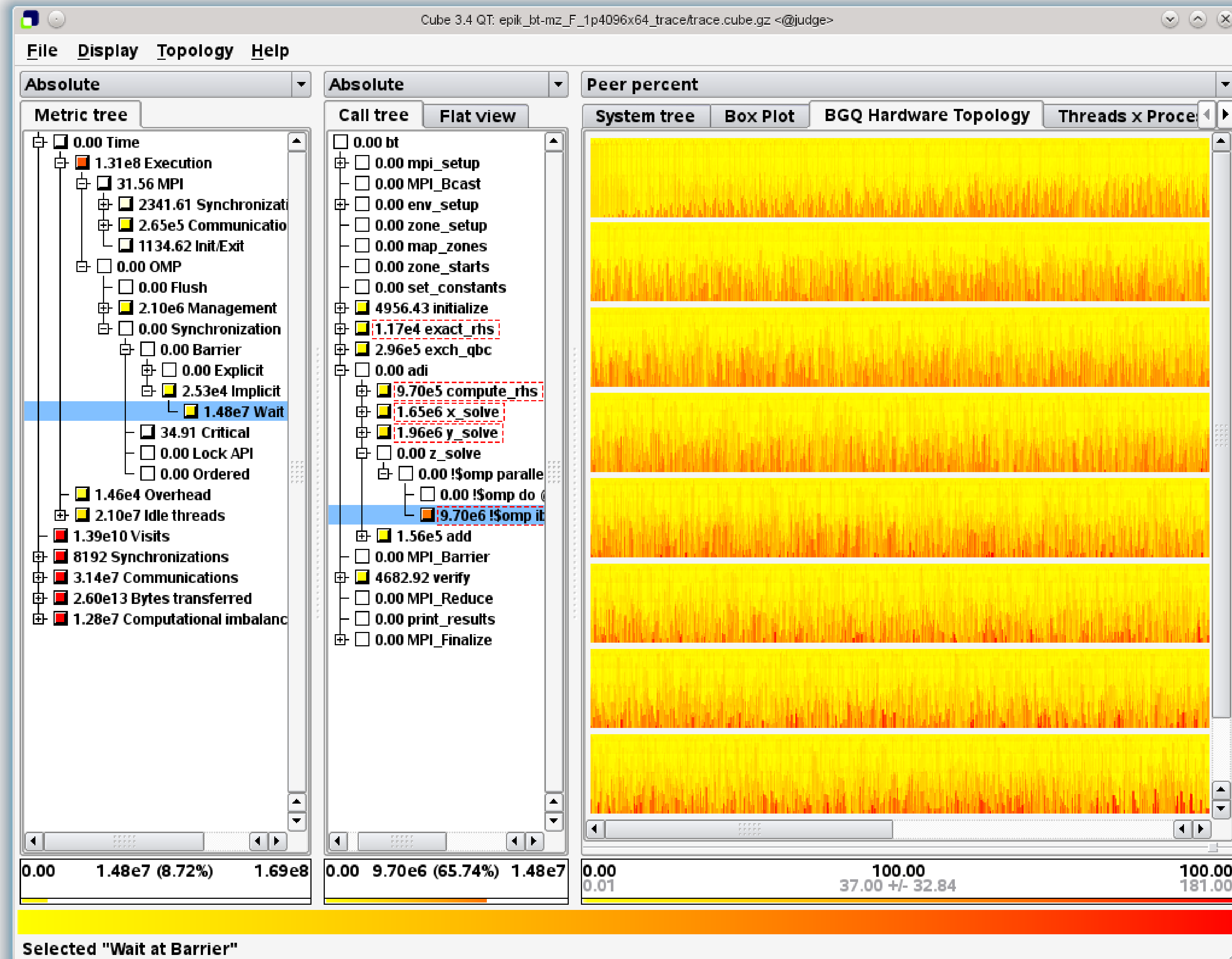
However

- requires display resolution which is not always available
  - *may have less than a pixel for each process/thread*
  - *topological presentation may obscure some values*
- not straightforward to quantify/compare

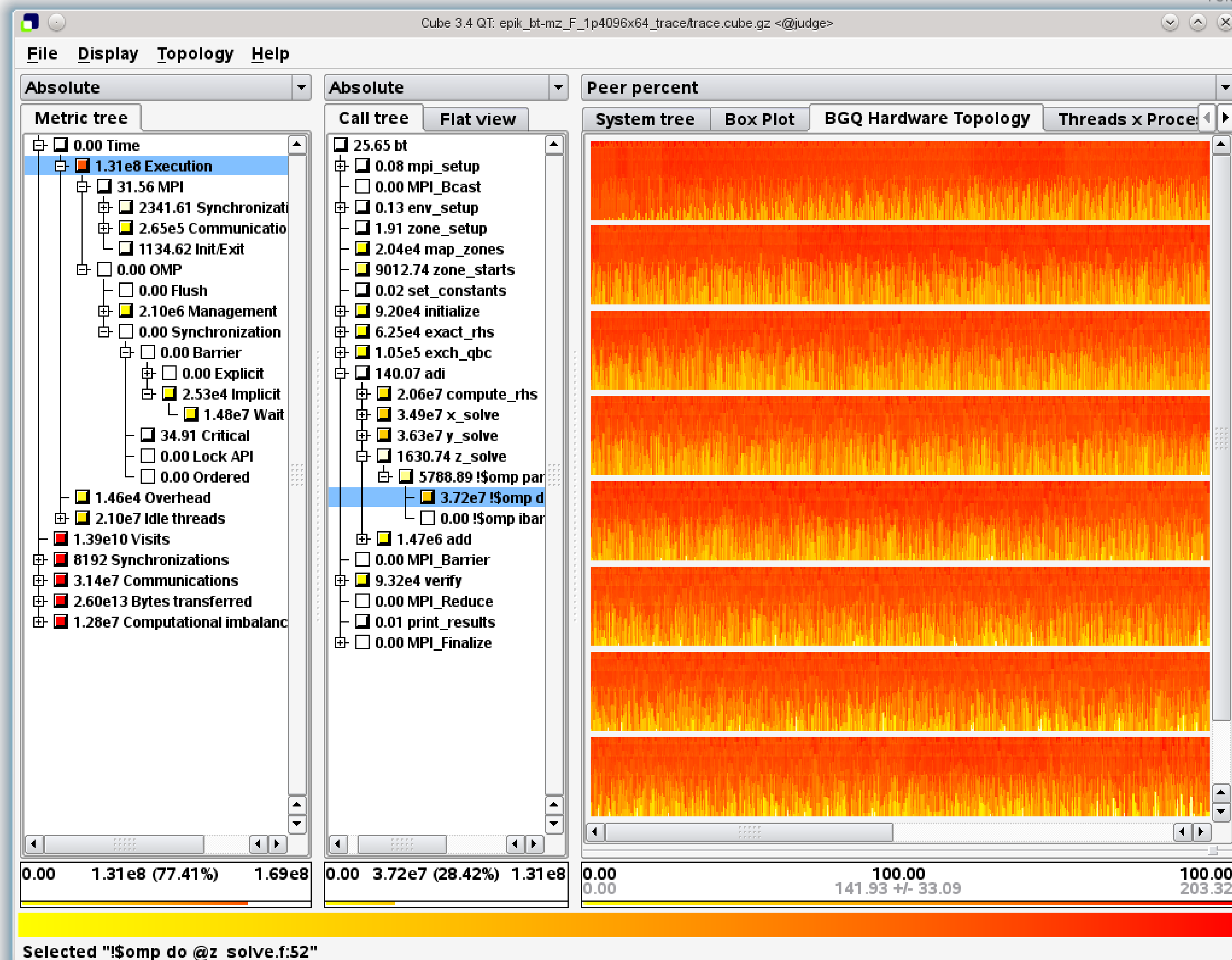
Therefore, include simple distribution statistics (min/mean/max, quartiles)

- Example: BT-MZ with 1M threads

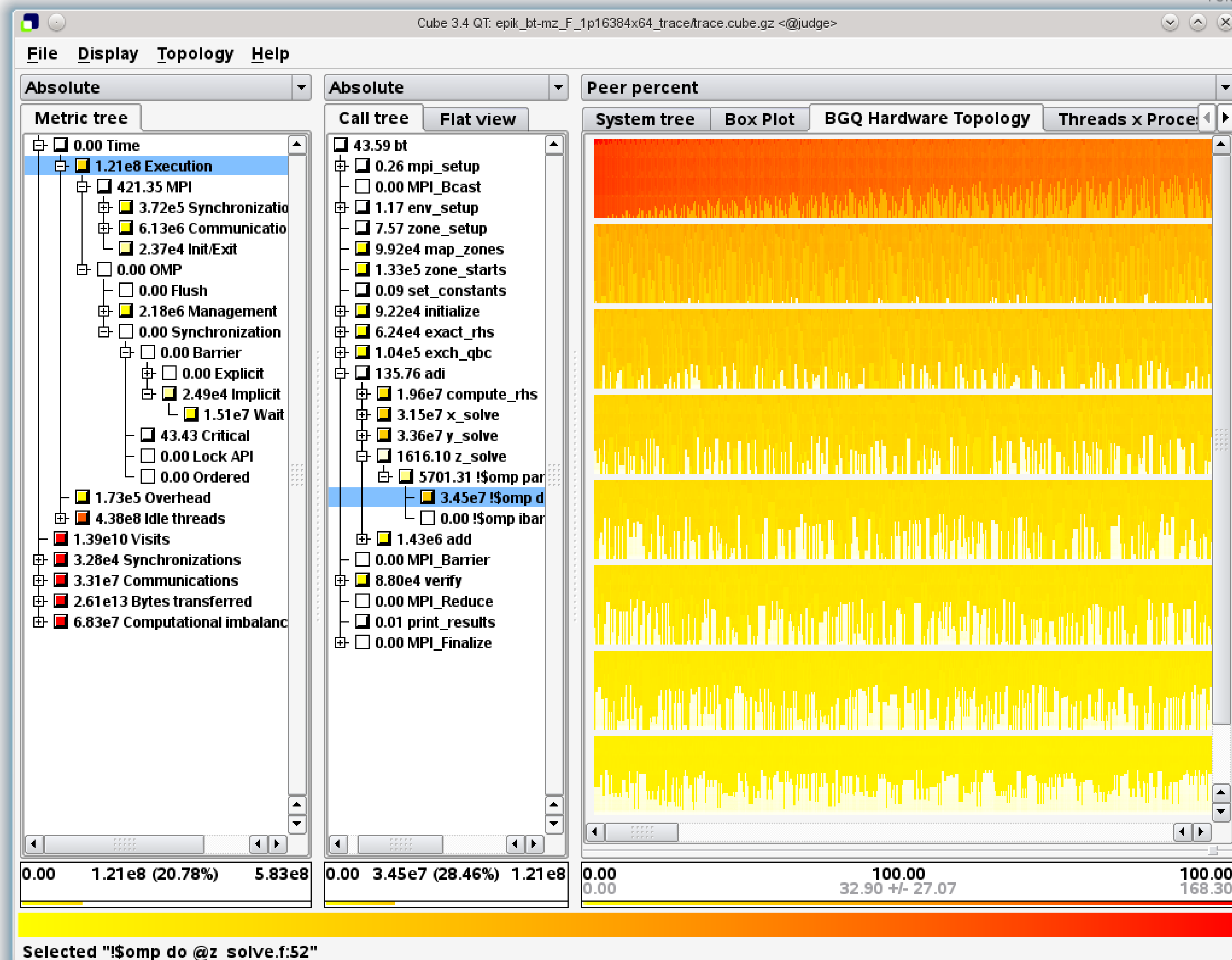
BT-MZ.F 4096x64 *z_solve* wait at implicit barrier



BT-MZ.F 4096x64 *z_solve* execution imbalance

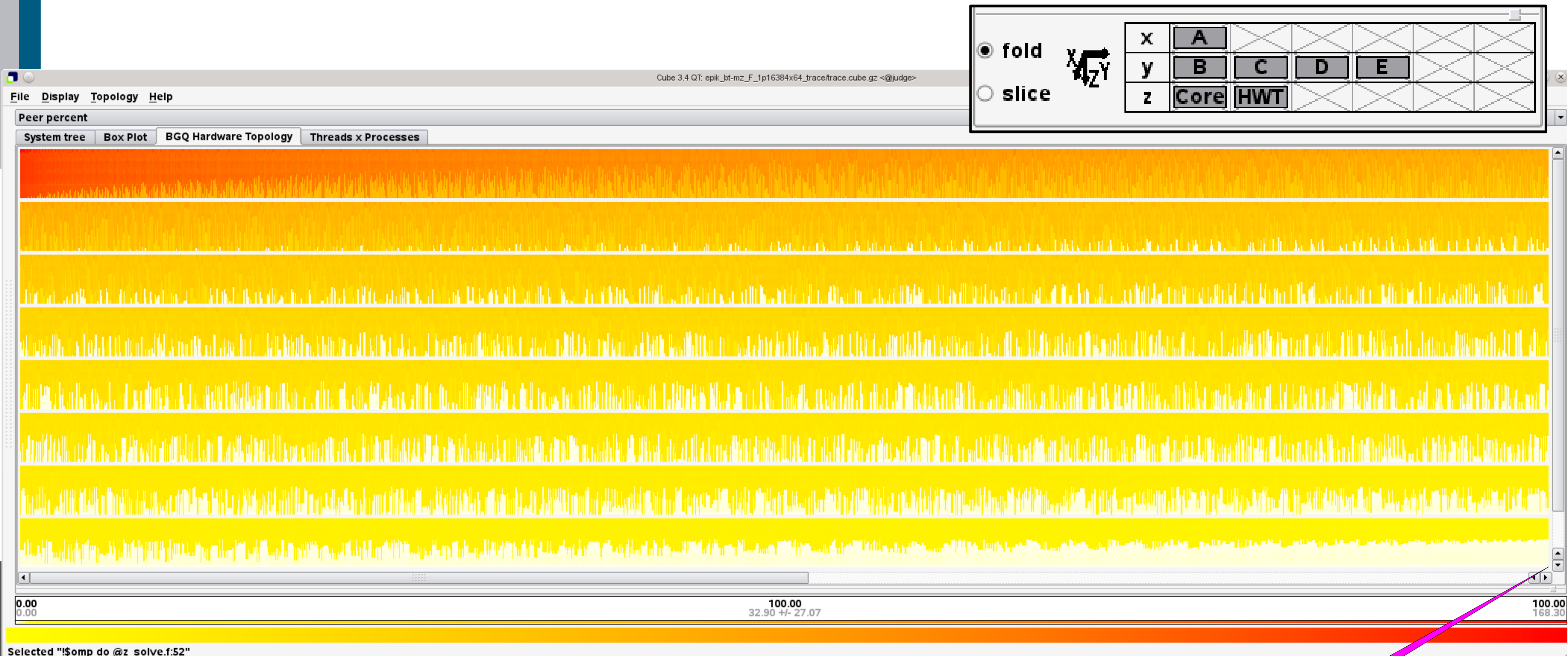


BT-MZ.F 16384x64 *z_solve* execution imbalance



BT-MZ.F 16384x64 *z_solve* execution imbalance

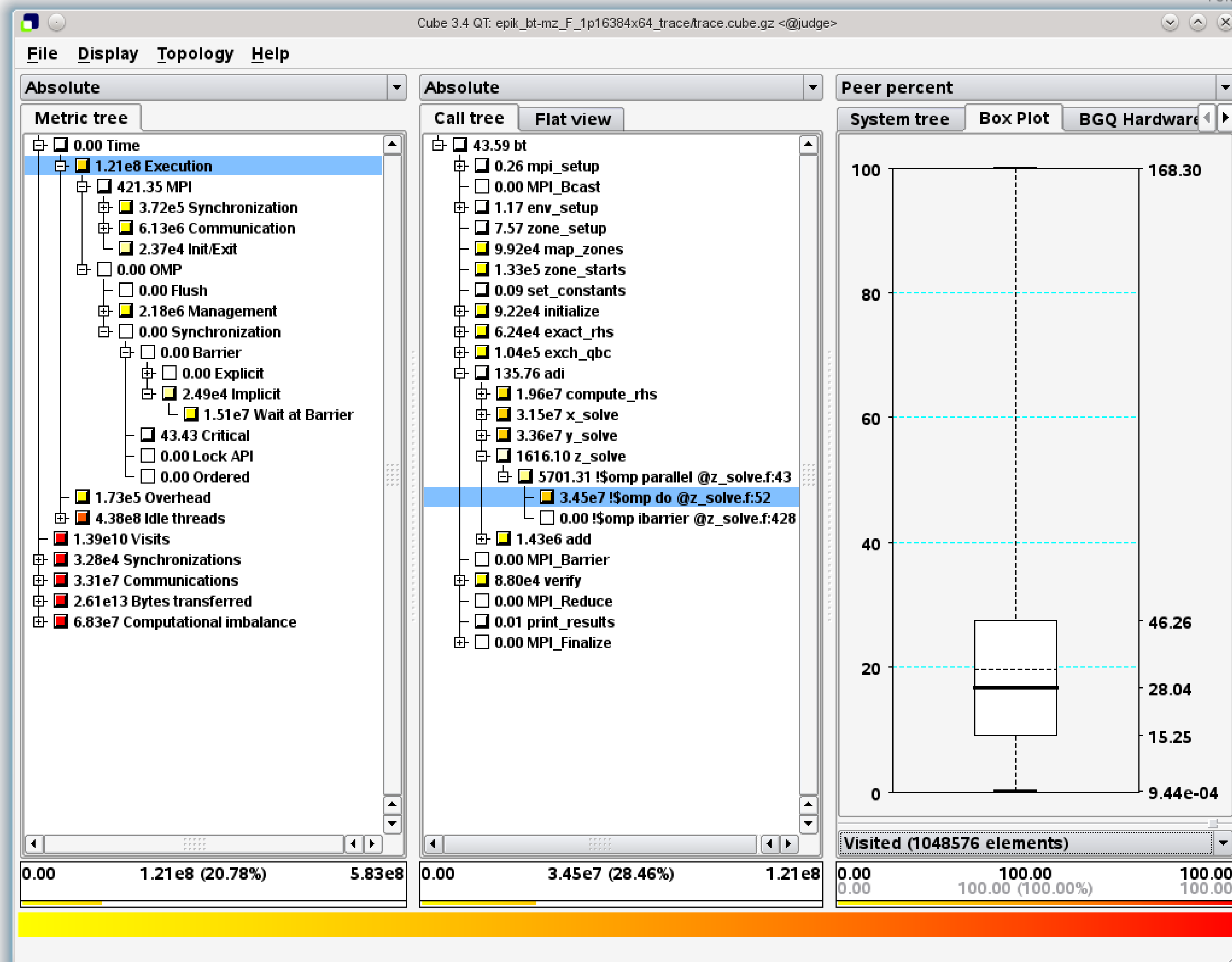
8x16x8x8x2 torus:



- 16384 MPI processes, 64 OpenMP threads per process
- 512 s benchmark execution (<3% measurement dilation)
- 2623 s extra for trace collection+analysis (1800 s collation)
- 312 GB trace data (256 intermediate SION files 8.0 TB)
- 2.8 GB scout.cube, 2.5 GB trace.cube.gz

Coord:	(A: 7,B: 15,C: 7,D: 7,E: 1,Core: 15,HWT: 3)
Node:	R33-M1-N0f-J00 <7,15,7,7,1>
Name:	Thread 63
MPI rank:	16383
Thread id:	63
Value:	0.00 (0.00%)

BT-MZ.F 16384x64 *z_solve* execution imbalance



Load analysis results on-demand/incrementally

Loading entire analysis reports into memory

- convenient for interactive exploration

However

- loading time and memory required grow with the size of the report
  - *proportional to numbers of metrics, callpaths, and threads*
  - *only a small subset can be shown at any time*
- inclusive metric values must be aggregated from exclusive ones

Therefore, store inclusive values in reports for incremental retrieval when required for presentation (or calculating exclusive metric values)

- [PARA'10]

Current/future challenges

Analysis report size & collation time (proportional to threads)

More processes and threads

More dynamic behaviour

- dynamically created processes and threads, tasks
- varying clock speed

More heterogeneous systems

- accelerators, combined programming models

More detailed measurements and analyses

- iterations, counters (at different levels)

More irregular behaviour (e.g., sampled events)

Conclusions

Complex large-scale applications 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 BlueGene, Cray, K & 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

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

JÜLICH FORSCHUNGSZENTRUM

German Research School for Simulation Sciences

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
- ready to run from VI-HPS HPC Linux Live DVD/ISO/OVA
- sources, documentation & publications:
  - *http://www.scalasca.org*
  - *mailto: scalasca@fz-juelich.de*

References

"Integrated runtime measurement summarization and selective event tracing for scalable parallel execution performance diagnosis," Wylie et al, Proc. PARA'06, LNCS 4699, pp.460-469.

"Scalable parallel trace-based performance analysis," Geimer et al, Proc. EuroPVM/MPI'06, LNCS 4192, pp.303-312.

"Performance measurement and analysis of large-scale parallel applications on leadership computing systems," Wylie et al, Scientific Programming 16(2-3):167-181, 2008.

"Scalable massively parallel I/O to task-local files," Frings et al, Proc. SC'09.

"Further improving the scalability of the Scalasca toolset," Geimer et al, Proc. PARA'10 Part II, LNCS 7134, pp.463-474

"Large-scale performance analysis of Sweep3D with the Scalasca toolset," Wylie et al, Parallel Processing Letters 20(4):397-414, 2010.

"Large-scale performance analysis of PFLOTRAN with Scalasca," Wylie et al, Proc. CUG'11.

"Scaling performance tools MPI communicator management," Geimer et al, Proc. EuroMPI'11, LNCS 6960, pp.178-187.

[Full Scalasca-related publication list available at www.scalasca.org]