JÜLICH
FORSCHUNGSZENTRUM

CUBE: A success story -- But does it have a future?

Vi-HPS 10th Anniversary Workshop | Bernd Mohr

Context:
18 Years of Automatic Trace Analysis

JÜLICH FORSCHUNGSZENTRUM

- 1999 – 2004

- Working Group
  - EU ESPRIT + IST
- http://webarchiv.fz-juelich.de/apart/

- Sequential analyzer
  - EXPERT
- http://webarchiv.fz-juelich.de/jsc/kojak.html

- 2006 – now

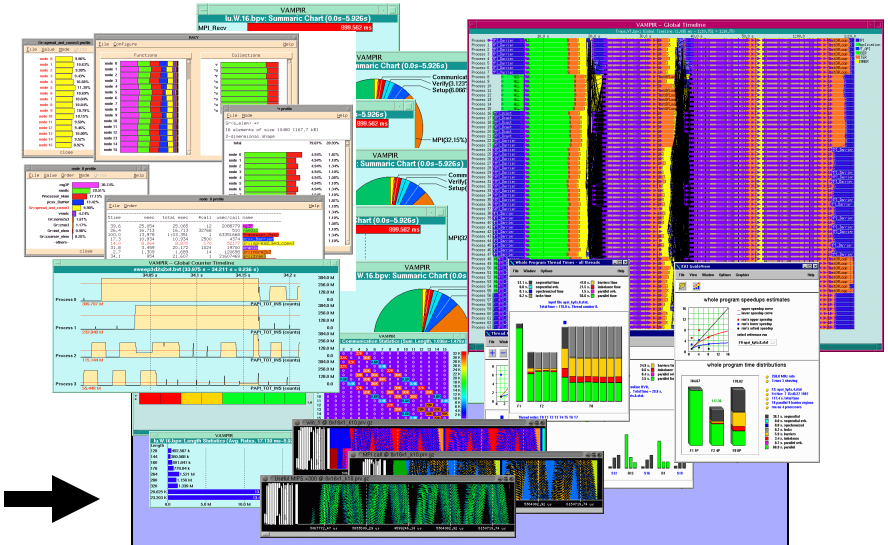
- Virtual Institute
  - Helmholtz
- http://www.vi-hps.org/

- Parallel analyzer
  - SCOUT
- http://www.scalasca.org

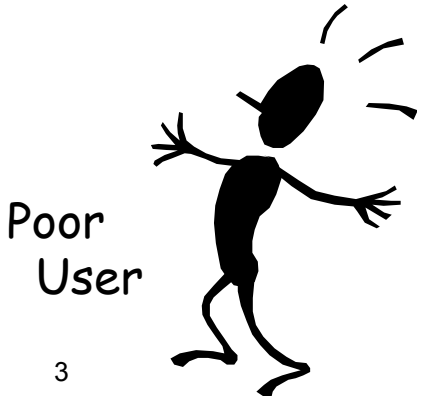
Traditional Performance Tools

Huge amount of Measurement data

Little Simple analysis

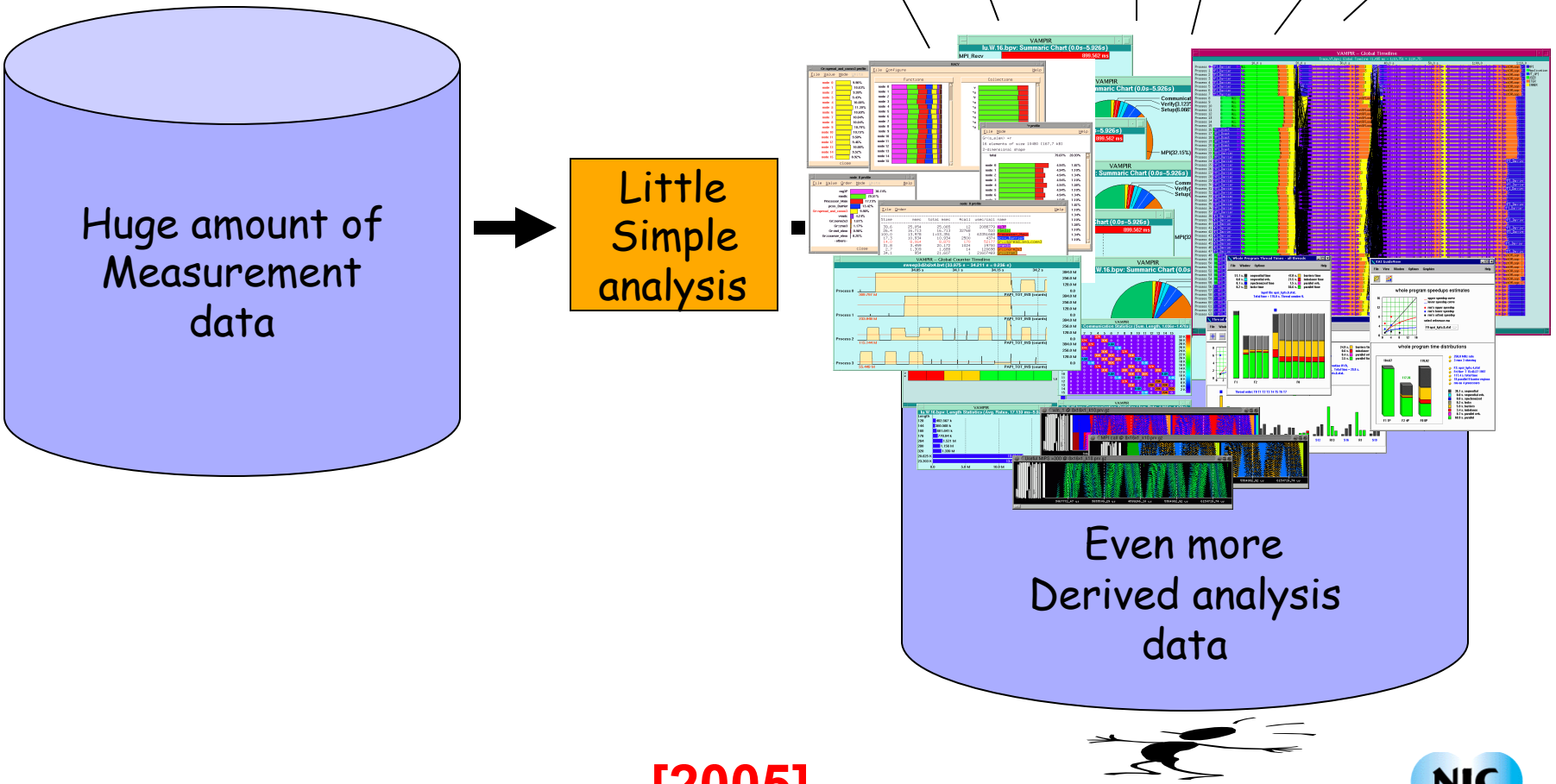


Even more Derived analysis data



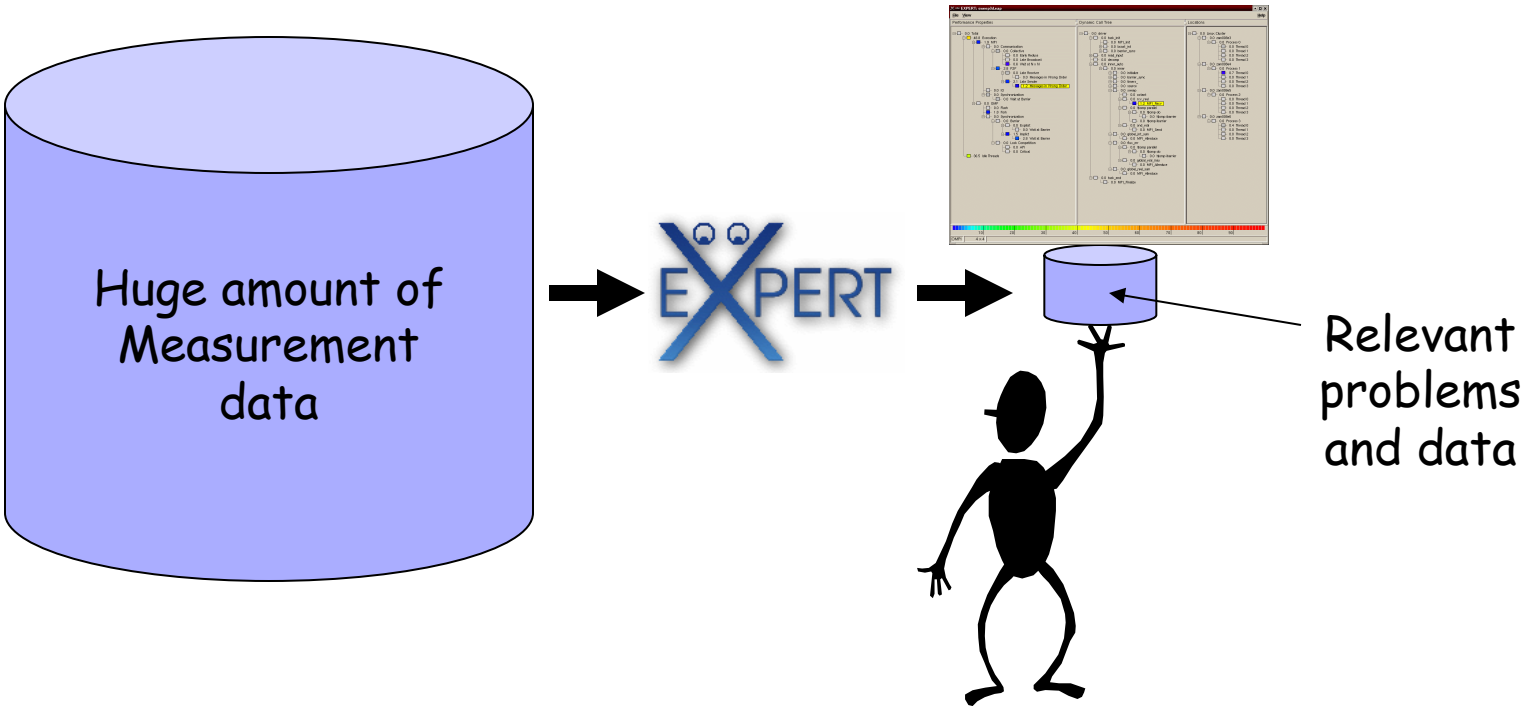
[2005]

Traditional Performance Tools



[2005]

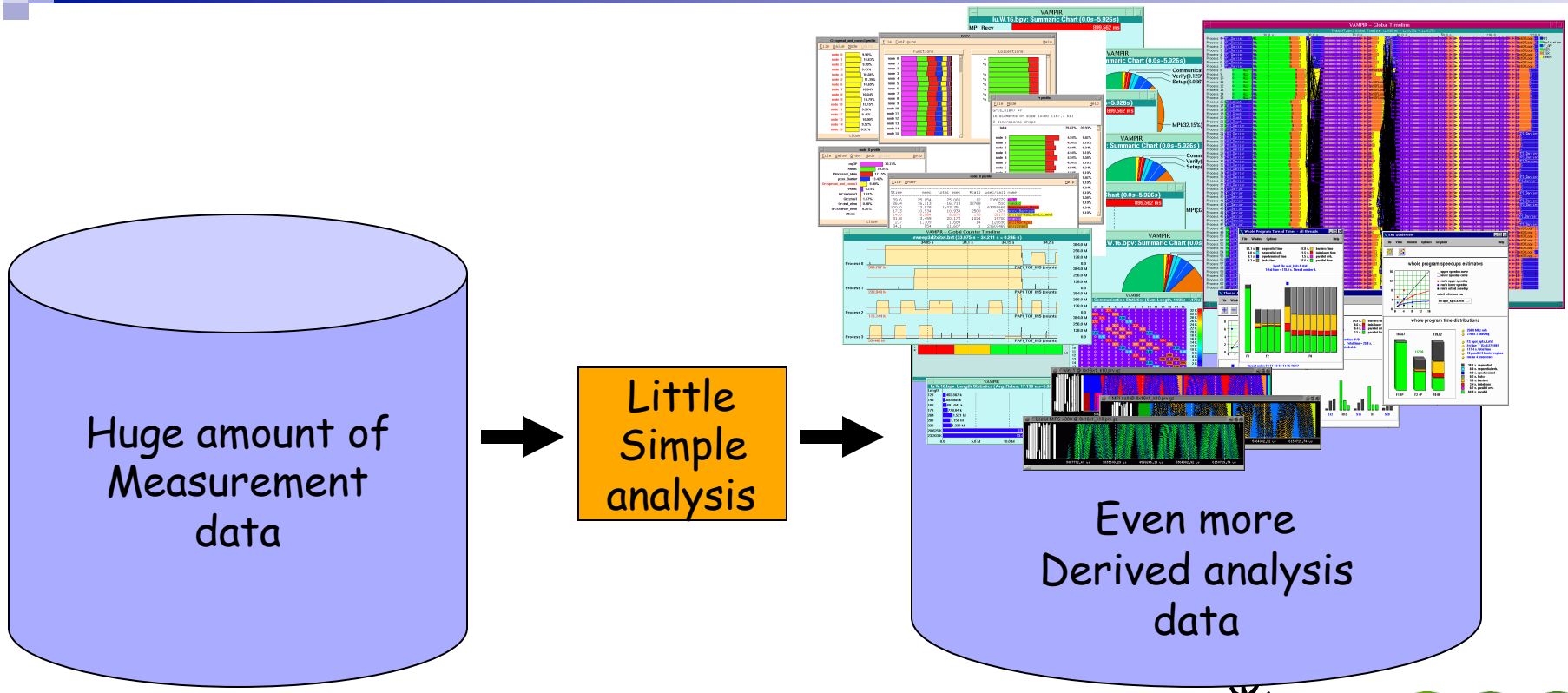
Solution Part 1: Automatic Tool



- For standard cases (90% ?!)
- For "normal" users
- Starting point for experts

[2005]

Solution Part 2: Expert Tools + Expert



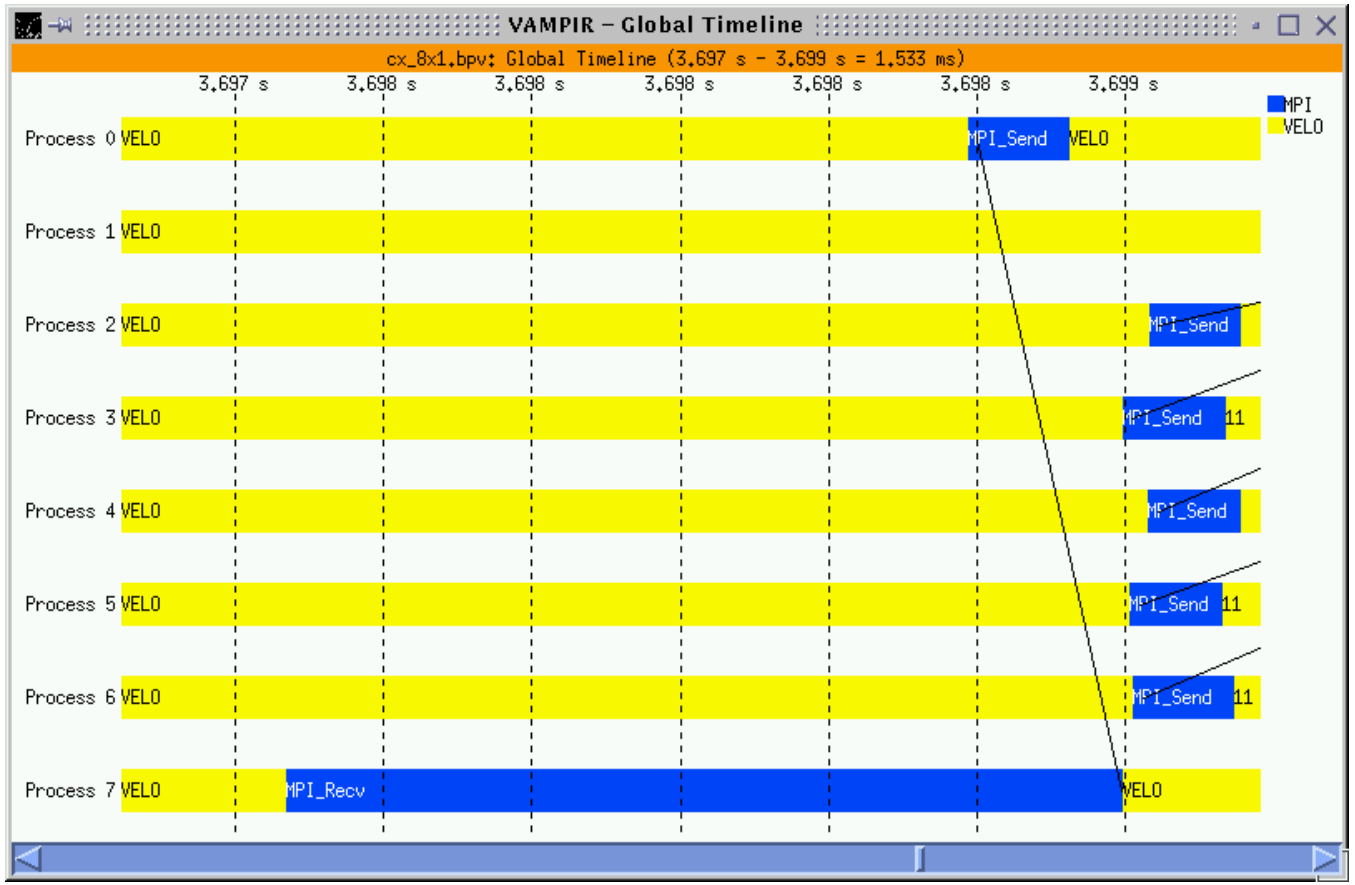
- For non-standard / tricky cases (10%)

⇨ More productivity for performance analysis process!

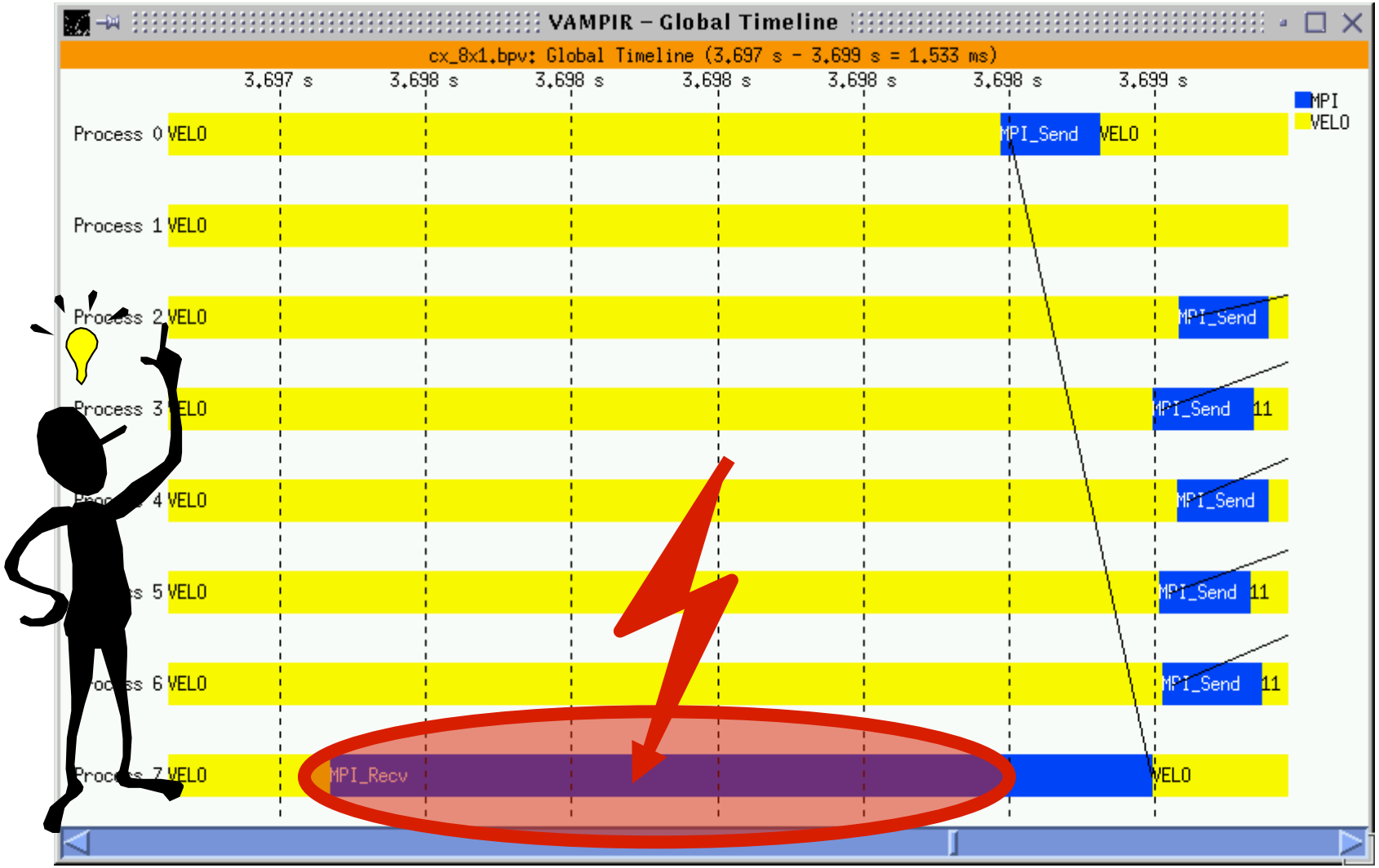
[2005]

Example: Late Sender (blocked receiver) [2001]

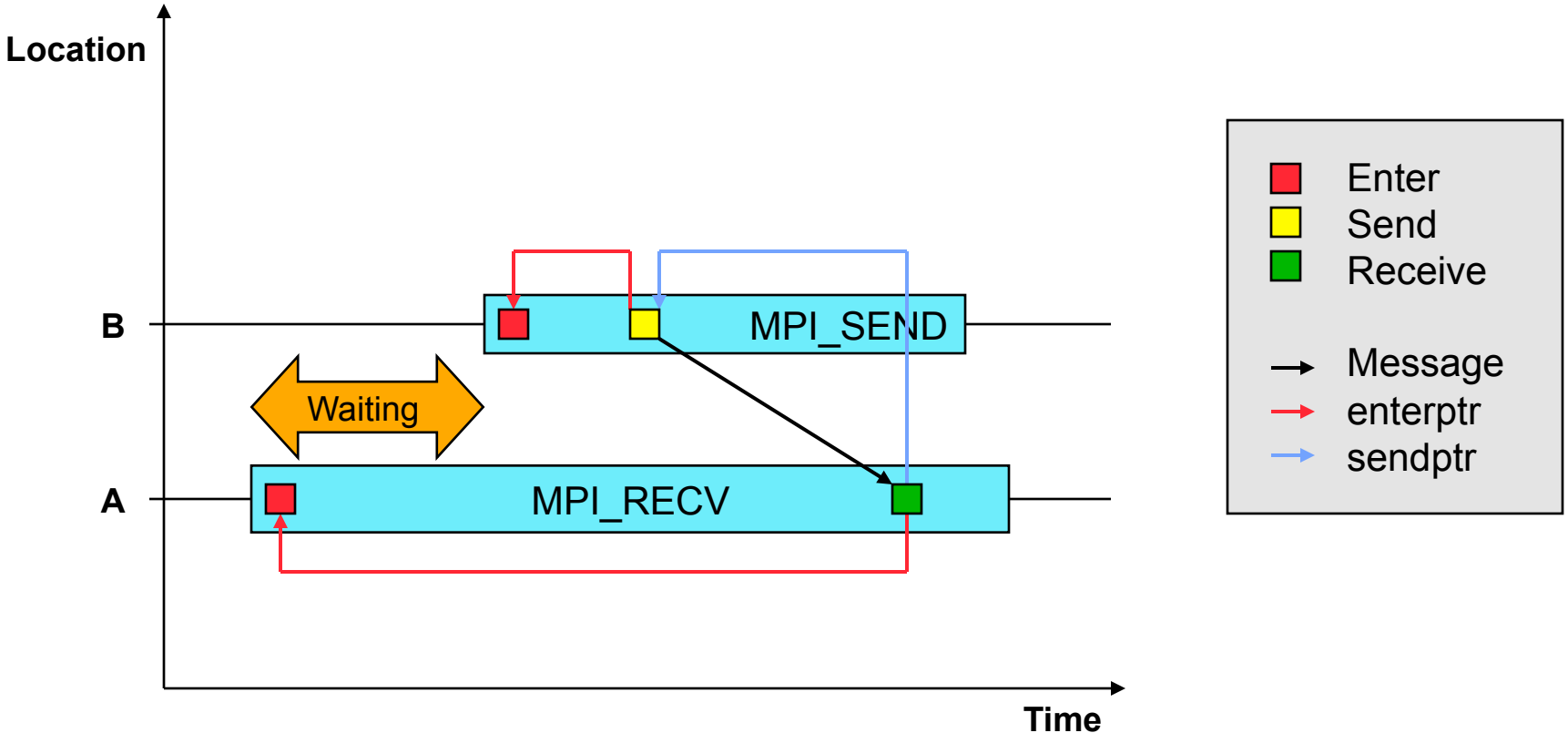
Czochralski crystal growth



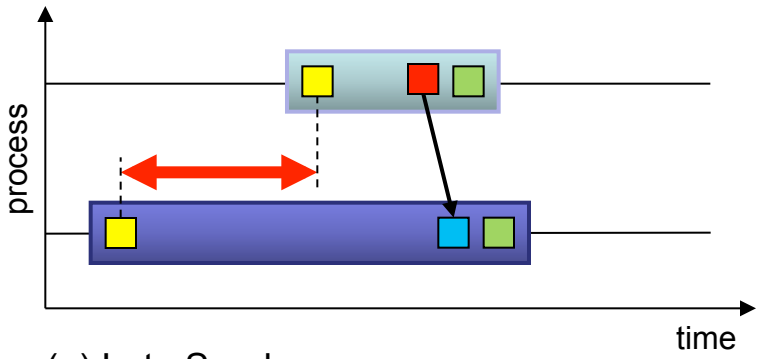
Example Automatic Analysis: Late Sender



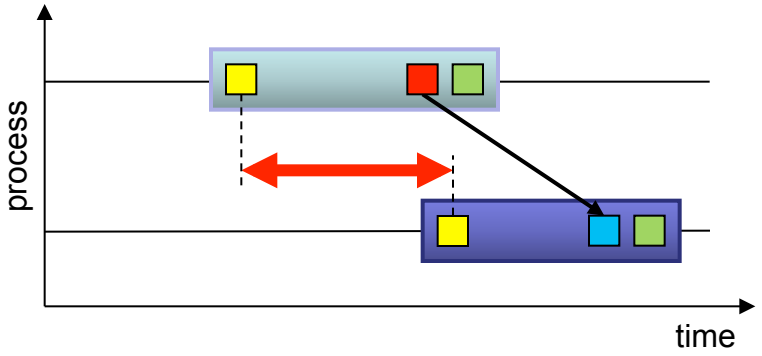
Example: Late Sender (blocked receiver) [2001]



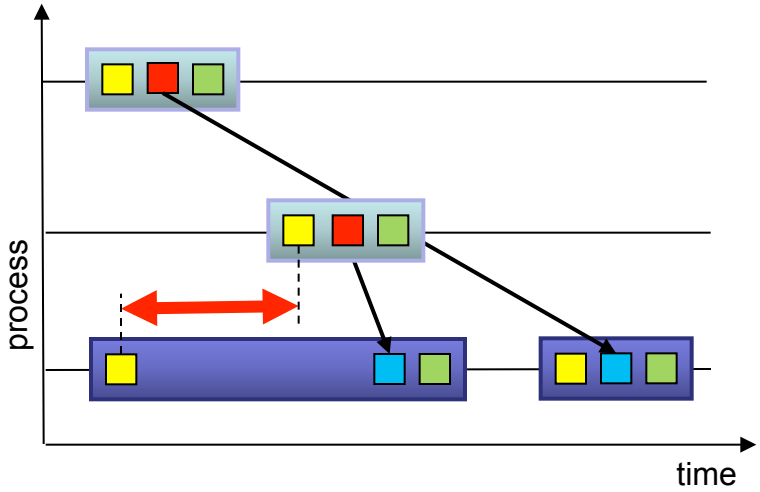
Scalasca: Example MPI Patterns



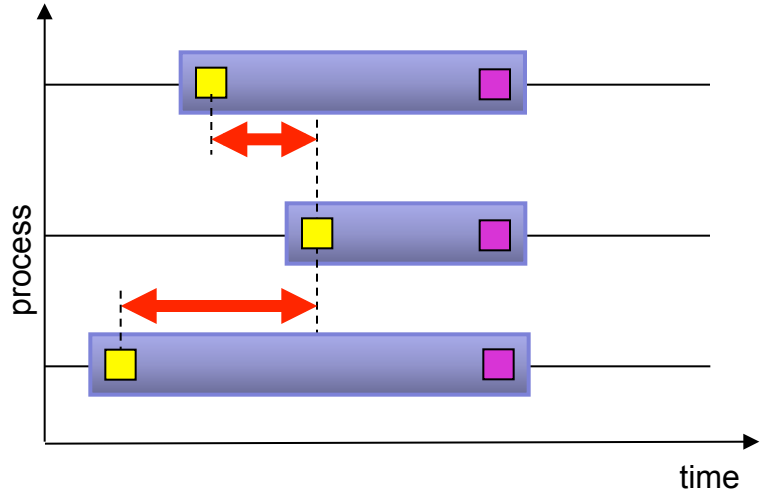
(a) Late Sender



(b) Late Receiver



(c) Late Sender / Wrong Order

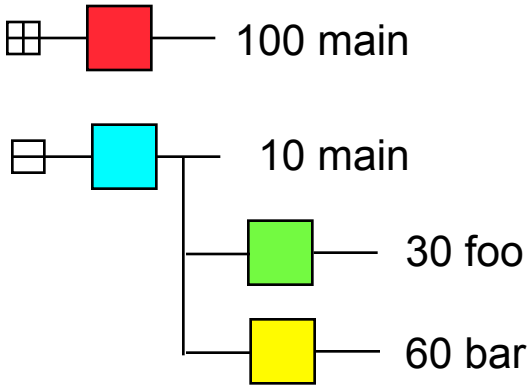


(d) Wait at N x N

ENTER EXIT SEND RECV COLLEXIT

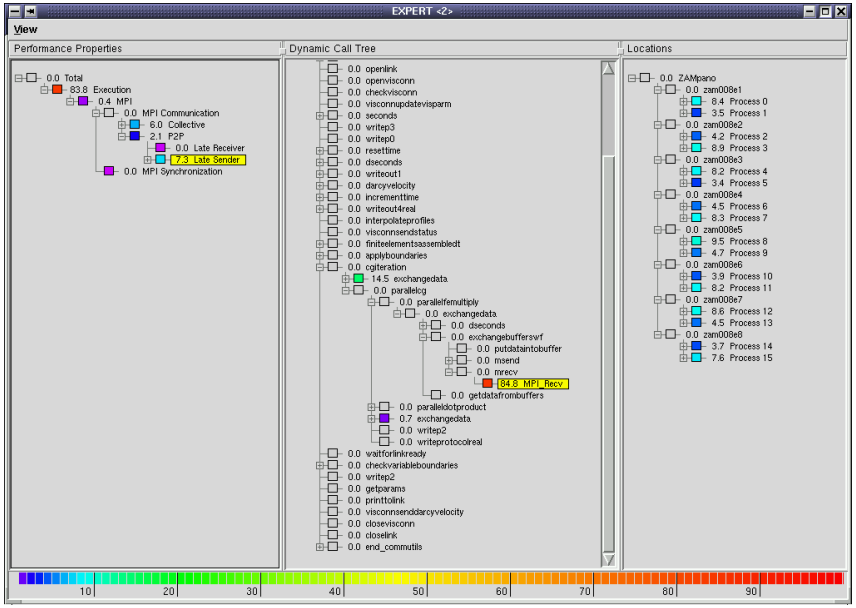
Presentation of Performance Behavior [2001]

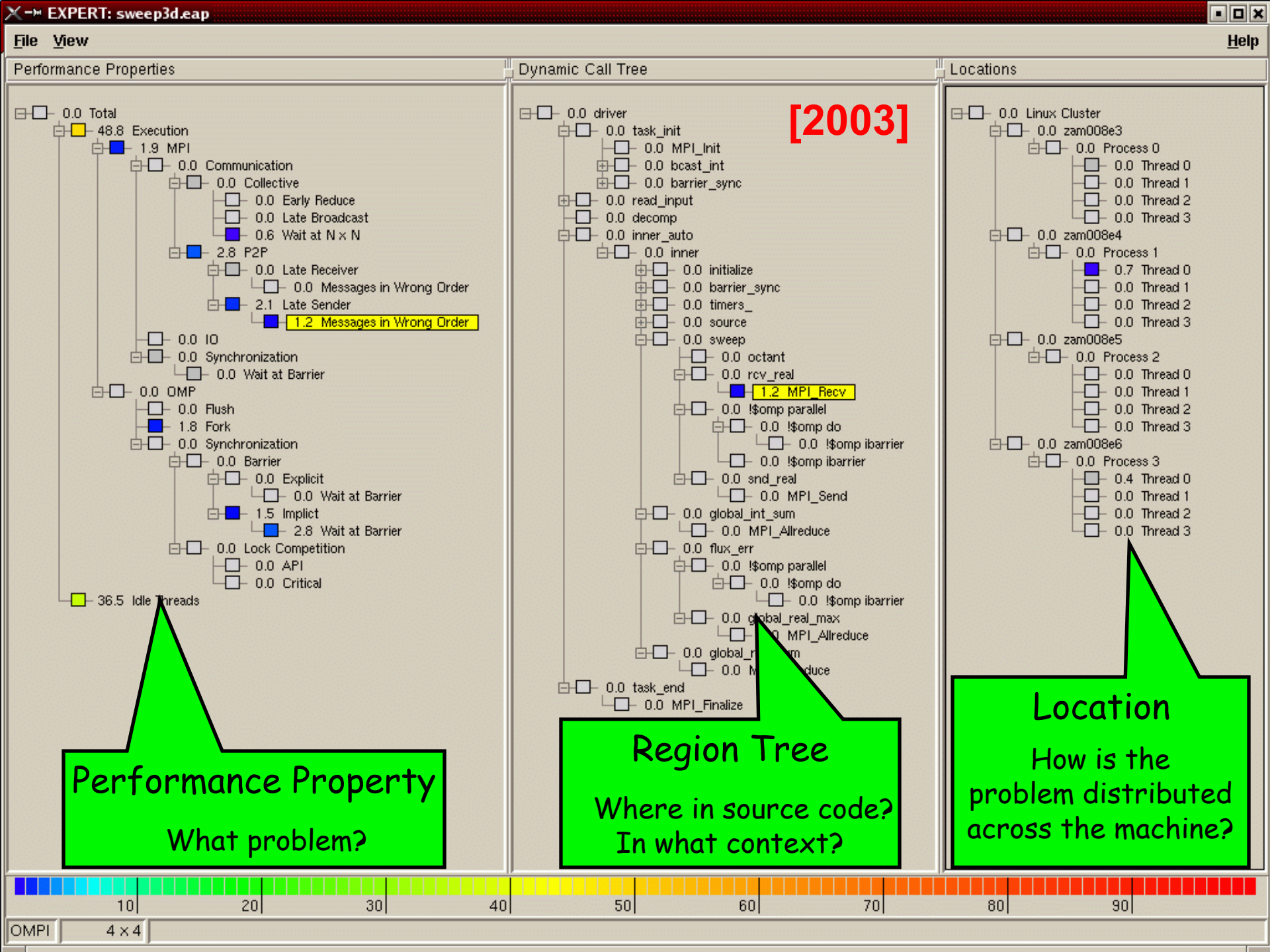
- Performance behavior
  - 3 dimensional matrix
  - Hierarchical dimensions
- Weighted tree
  - Tree browser
  - Each node has weight
    - Percentage of CPU allocation time
    - E.g. time spent in subtree of call tree
  - Displayed weight depends on state of node
    - Collapsed (including weight of descendants)
    - Expanded (without weight of descendants)
  - Displayed using
    - Color
      - Allows to easily identify hot spots (bottlenecks)
    - Numerical value
      - Detailed comparison



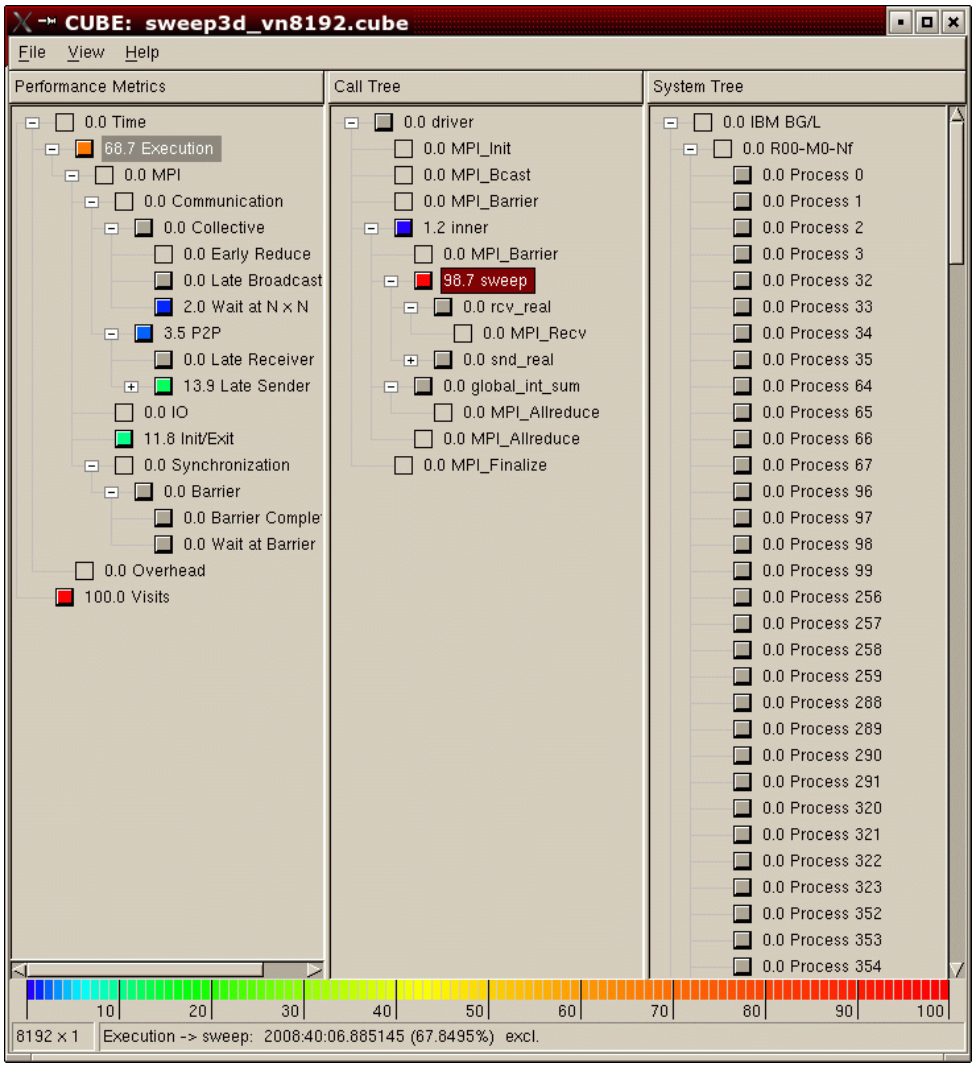
Presentation of Performance Behavior (2) [2001]

- Three views
  - Performance property
  - Call tree
  - Locations
- Interconnected
  - View refers to selection in left neighbor
- Two modes
  - Absolute: percent of total CPU allocation time
  - Relative: percent of selection in left neighbor
- Collapsing/expanding of nodes
  - Analysis on all hierarchy levels



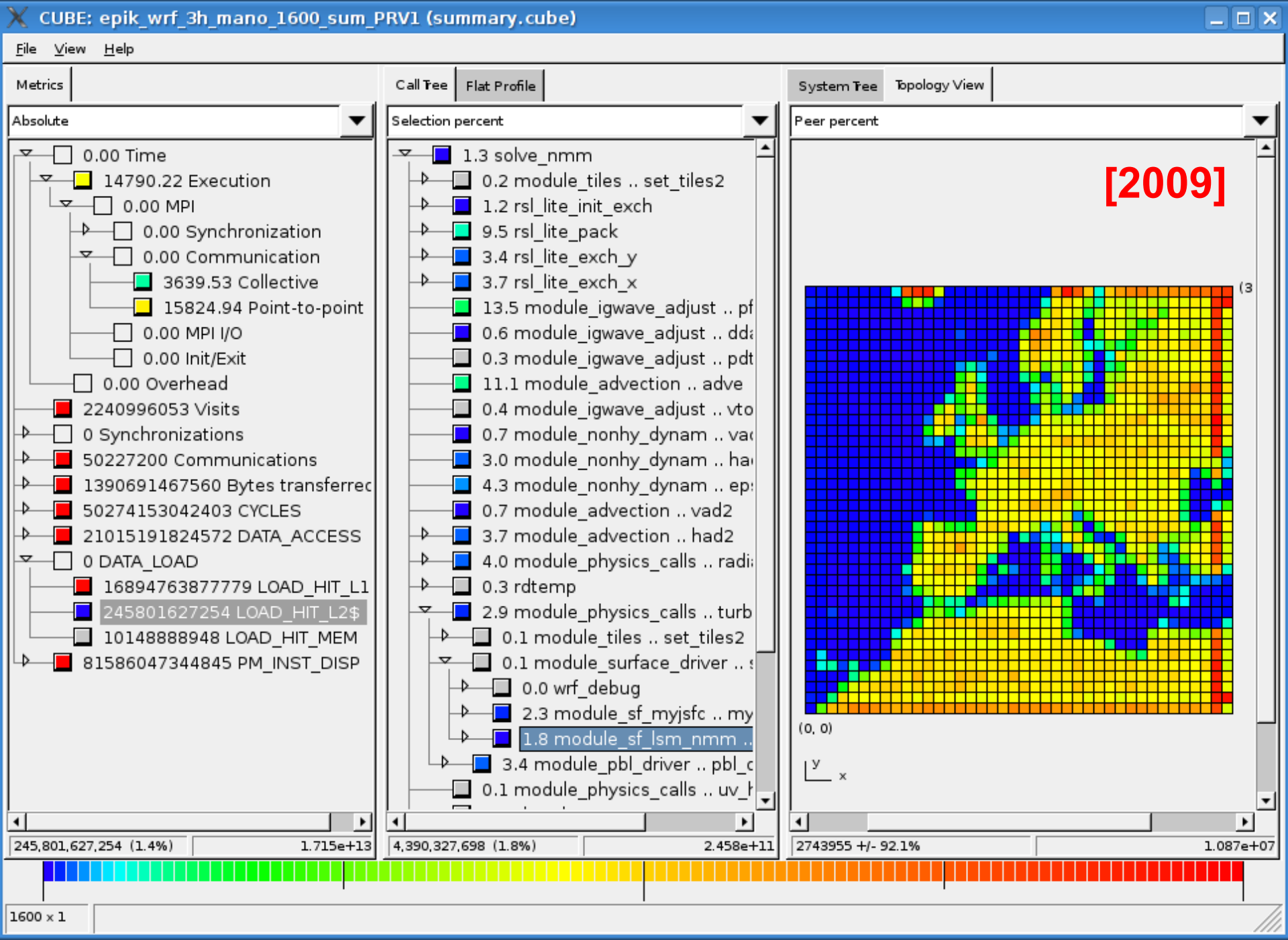

EXPERT: sweep3d.eap
File View Help
Performance Properties
Dynamic Call Tree
Locations
[2003]
0.0 Total
48.8 Execution
1.9 MPI
0.0 Communication
0.0 Collective
0.0 Early Reduce
0.0 Late Broadcast
0.6 Wait at N x N
2.8 P2P
0.0 Late Receiver
0.0 Messages in Wrong Order
2.1 Late Sender
1.2 Messages in Wrong Order
0.0 IO
0.0 Synchronization
0.0 Wait at Barrier
0.0 OMP
0.0 Flush
1.8 Fork
0.0 Synchronization
0.0 Barrier
0.0 Explicit
0.0 Wait at Barrier
1.5 Implict
2.8 Wait at Barrier
0.0 Lock Competition
0.0 API
0.0 Critical
36.5 Idle Threads
0.0 driver
0.0 task_init
0.0 MPI_Init
0.0 bcast_int
0.0 barrier_sync
0.0 read_input
0.0 decomp
0.0 inner_auto
0.0 inner
0.0 initialize
0.0 barrier_sync
0.0 timers_
0.0 source
0.0 sweep
0.0 octant
0.0 rcv_real
1.2 MPI_Recv
0.0 !$omp parallel
0.0 !$omp do
0.0 !$omp ibarrier
0.0 !$omp ibarrier
0.0 snd_real
0.0 MPI_Send
0.0 global_int_sum
0.0 MPI_Allreduce
0.0 flux_err
0.0 !$omp parallel
0.0 !$omp do
0.0 !$omp ibarrier
0.0 global_real_max
MPI_Allreduce
0.0 task_end
0.0 MPI_Finalize
0.0 Linux Cluster
0.0 zam008e3
0.0 Process 0
0.0 Thread 0
0.0 Thread 1
0.0 Thread 2
0.0 Thread 3
0.0 zam008e4
0.0 Process 1
0.7 Thread 0
0.0 Thread 1
0.0 Thread 2
0.0 Thread 3
0.0 zam008e5
0.0 Process 2
0.0 Thread 0
0.0 Thread 1
0.0 Thread 2
0.0 Thread 3
0.0 zam008e6
0.0 Process 3
0.4 Thread 0
0.0 Thread 1
0.0 Thread 2
0.0 Thread 3
Performance Property
What problem?
Region Tree
Where in source code?
In what context?
Location
How is the problem distributed across the machine?
10
20
30
40
50
60
70
80
90
OMPI
4 x 4

Example: sweep3D on 8192 BG/L PEs



- New topology display
- Shows distribution of pattern over HW topology
- Scales to larger systems

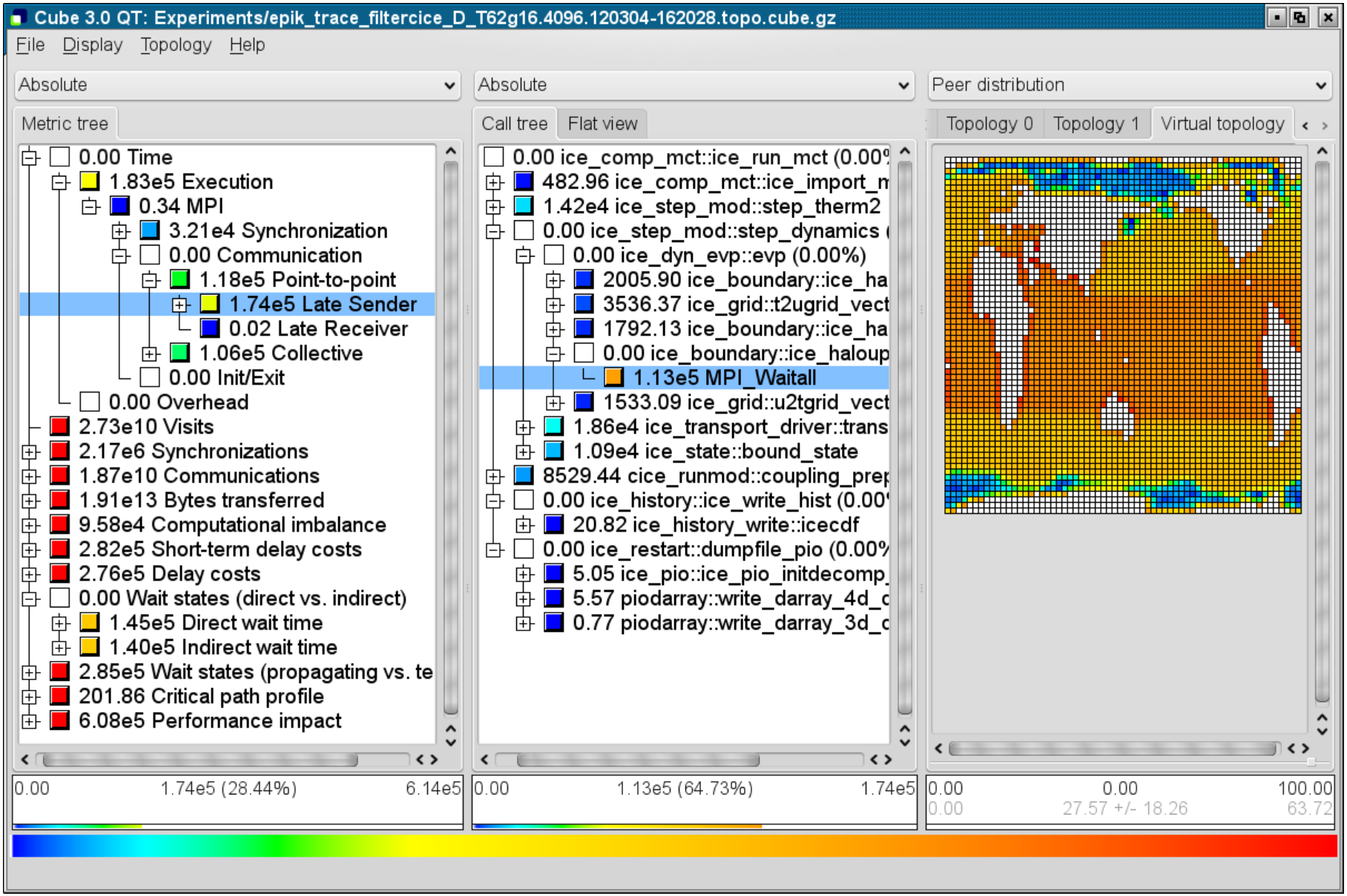
[2007]



WRF-NMM weather prediction code on MareNostrum @ 1600 CPUs

Scalasca Example: CESM Sea Ice Module Late Sender Analysis + Application Topology

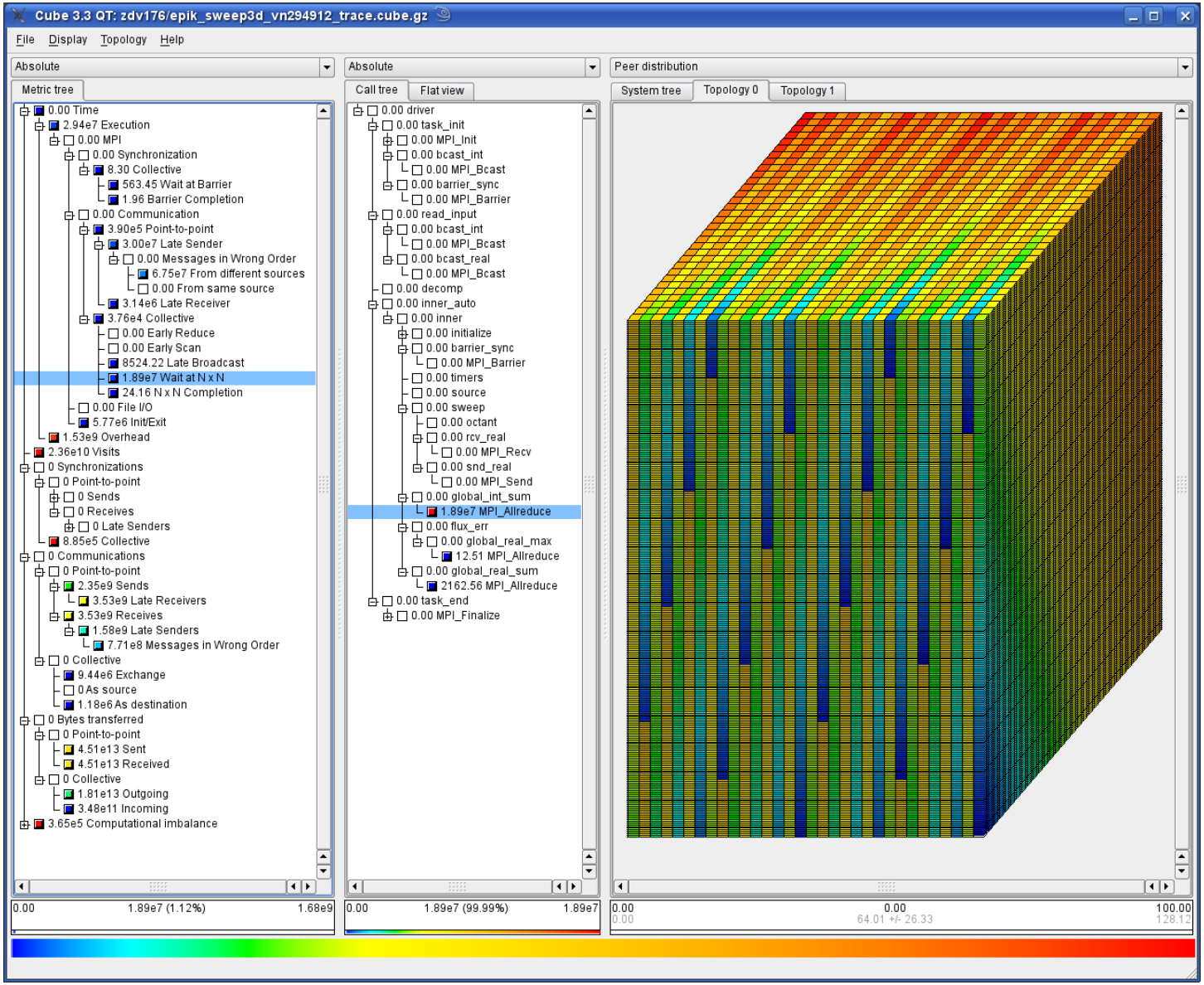
JÜLICH FORSCHUNGSZENTRUM



Trace analysis sweep3D@294,912

- 10 min sweep3D runtime
- 11 sec replay
- 4 min trace data write/read (576 files)
- 7.6 TB buffered trace data
- 510 billion events

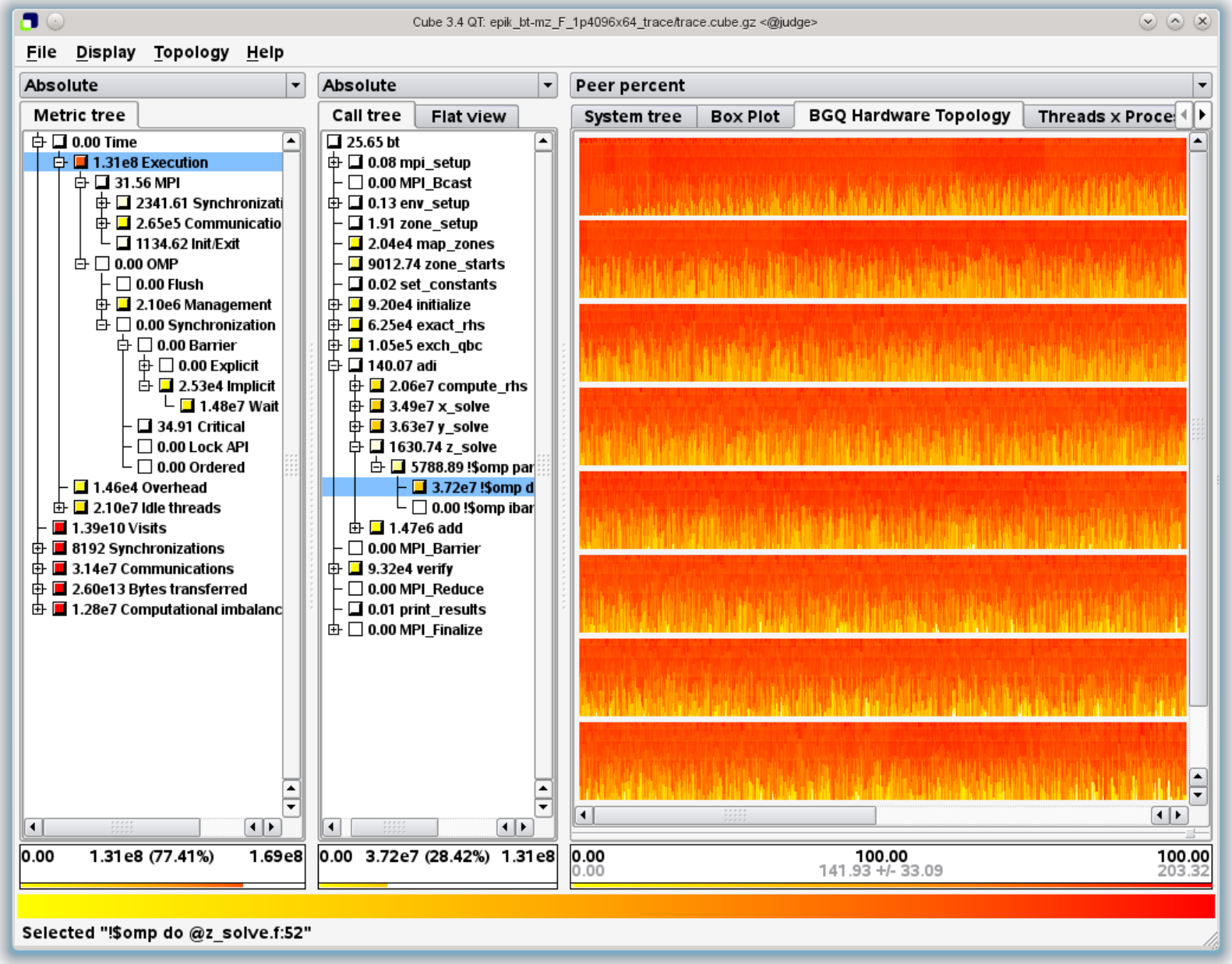
[2010]



Scalasca trace analysis bt-mz@1,048,704 BGQ

JÜLICH FORSCHUNGSZENTRUM

[2013]



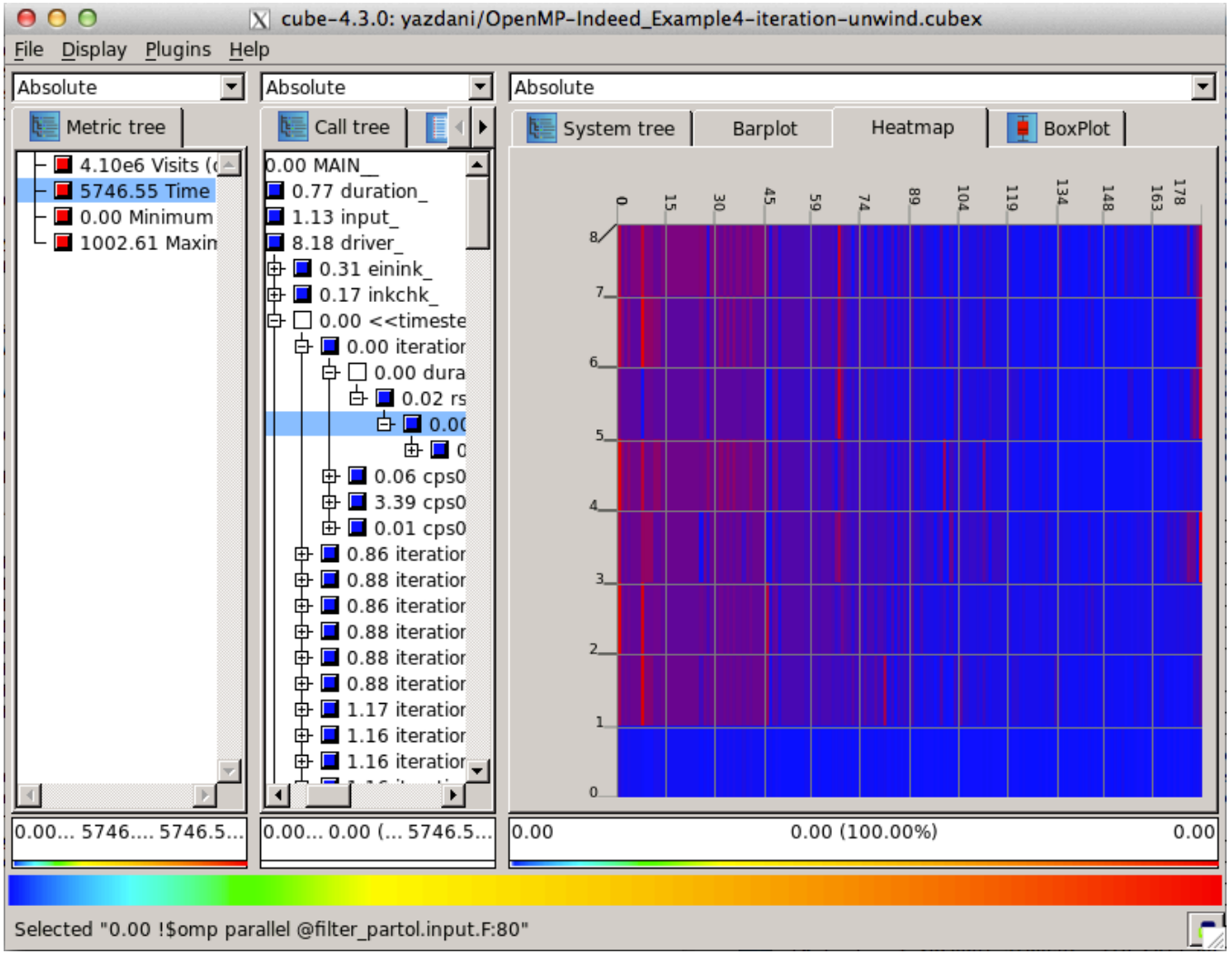
Cube Viz Plugins: Phase Heatmap [2015]

JÜLICH FORSCHUNGSZENTRUM

- **Phase profiling**

- Collects data for each instance of phases marked in program instead of aggregating it

- Shows data over "time" (phase instances) for each rank/ thread



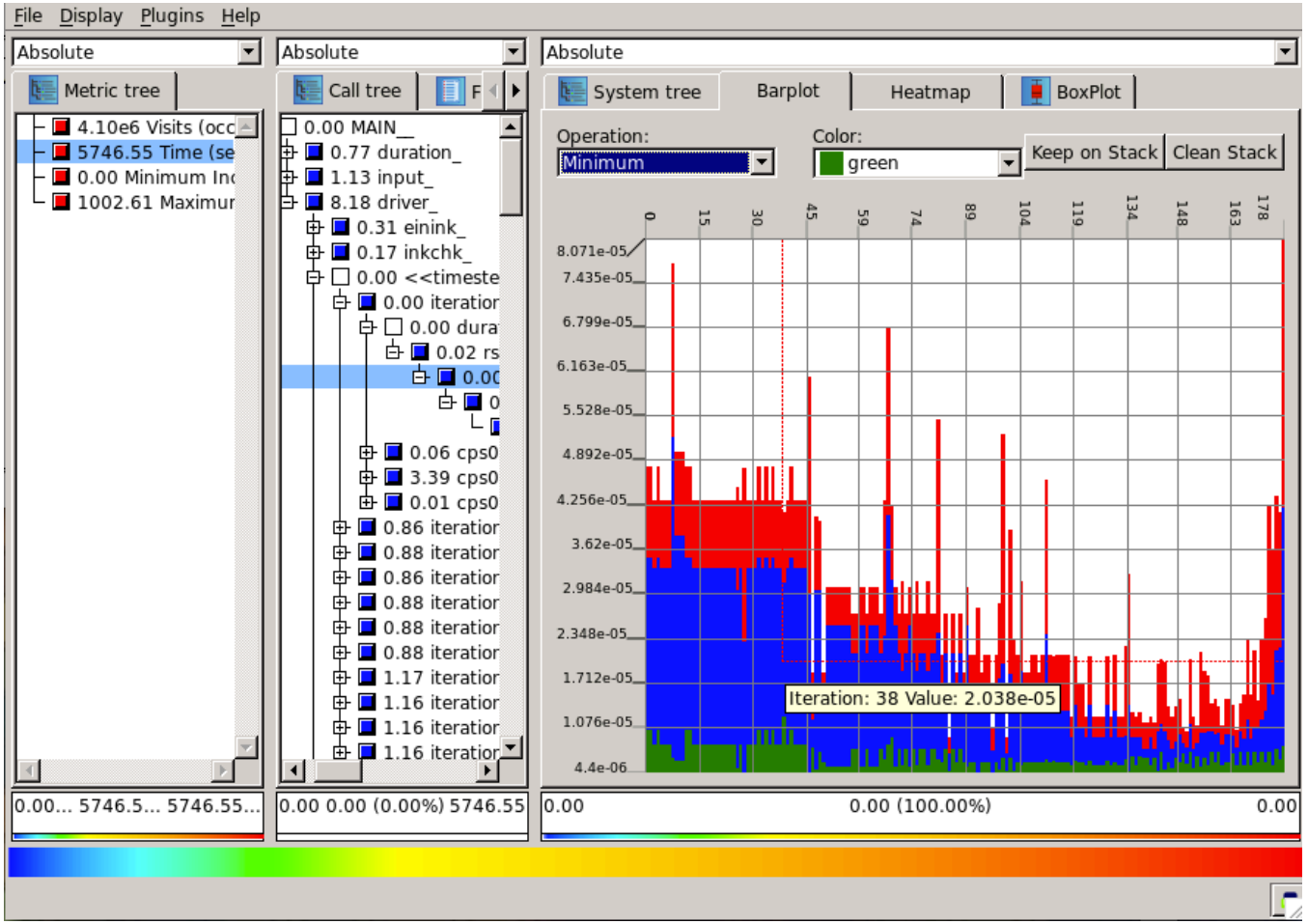
Cube Viz Plugins: Phase Barplot [2015]

JÜLICH FORSCHUNGSZENTRUM

- **Phase profiling**

- Collects data for each instance of phases marked in program instead of aggregating it

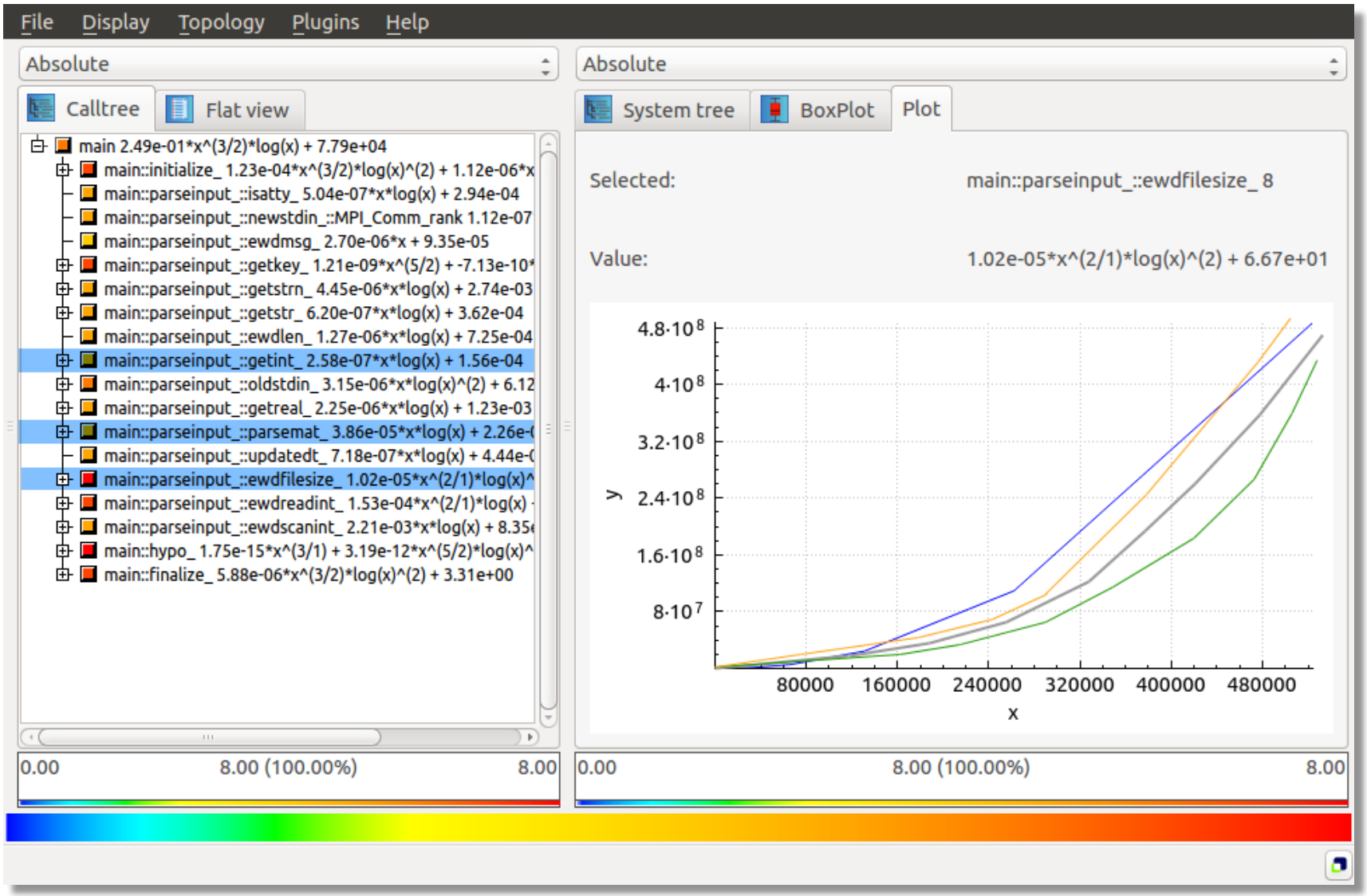
- Shows min/ max/avg metric value over “time” (phase instances)



Catwalk: Result Visualization

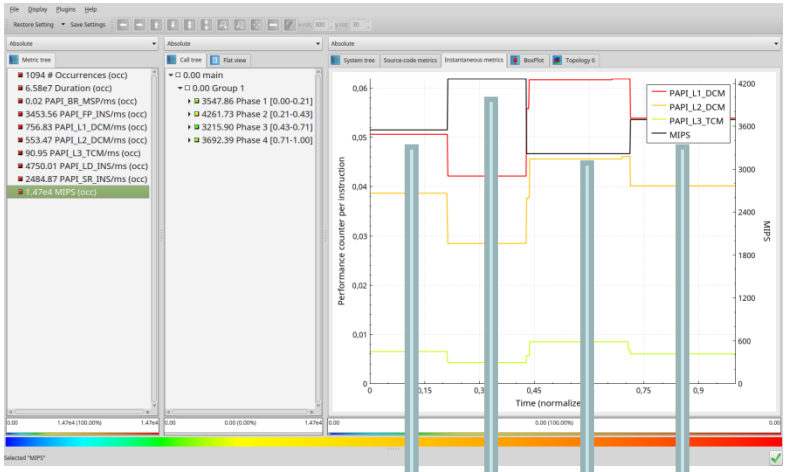
[2015]

JÜLICH FORSCHUNGSZENTRUM



T5.4 – Example: Stream Benchmark [2015]

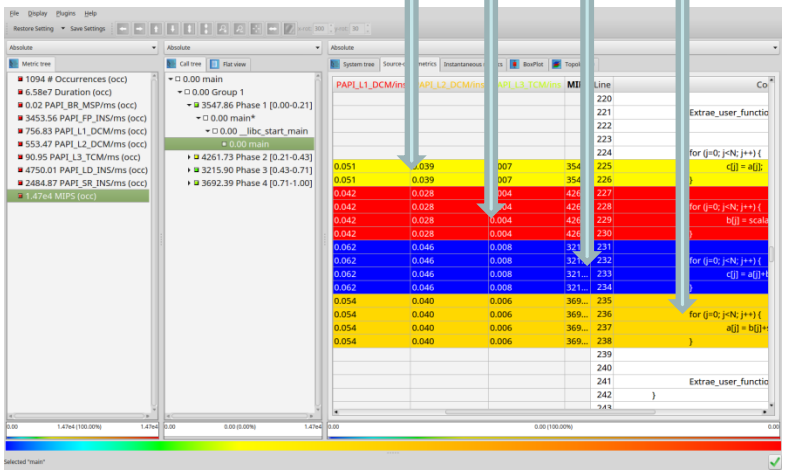
- Detailed performance progression (4 phases)



Stream code

```
for (i = 0; i < NITERS; i++)
{
     Extrae_event (BEGIN);
     for (j = 0; j < N; j++)
          c[j] = a[j];
     for (j = 0; j < N; j++)
          b[j] = s*c[j];
     for (j = 0; j < N; j++)
          c[j] = a[j]+b[j];
     for (j = 0; j < N; j++)
          a[j] = b[j]+s*c[j];
    Extrae_event (END);
}
```

- Source code reference (4 kernels within the instrumented region)



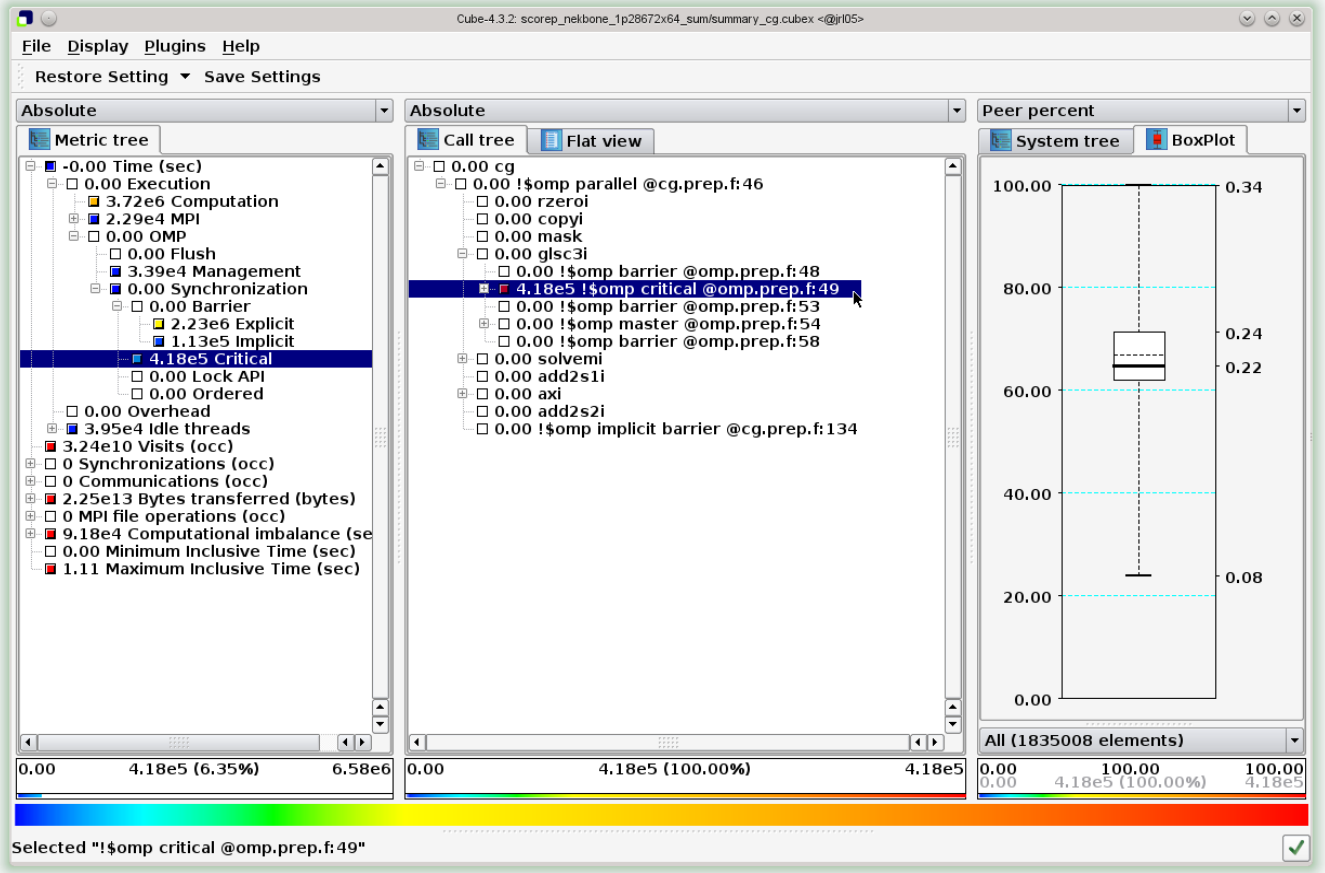
Scalasca: 1,835,008 Threads Test Case

JÜLICH FORSCHUNGSZENTRUM

[2016]

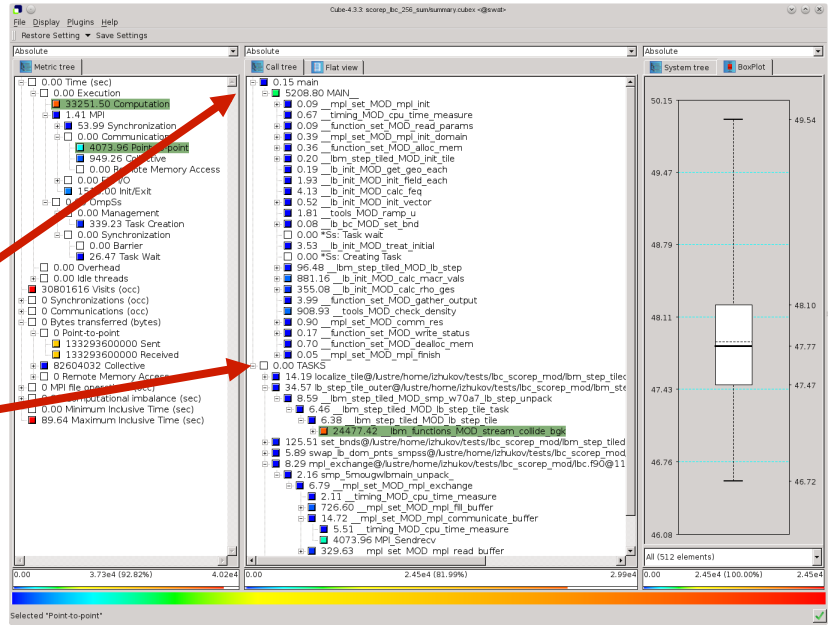
- Nekbone
- CORAL benchmark
- JuQueen experiment
- 28,672 x 64 = 1,835,008 threads

- Load imbalance at OpenMP critical section



Issue in the Future?

- Trend to asynchronous programming models
  - Kernels (CUDA, OpenCL)
  - Tasks (OpenMP, OmpSs)
  - Threads (Pthreads, …)
- CUBE representation of these items
  - Separate list of trees besides main
    - Non-scalable
    - Does not show dependencies
      - To creation / launch / spawn location
      - Between items (e.g. tasks)
- **Need new concept?**



YOU KNOW YOU MADE IT …

… IF LARGE COMPANIES "COPY" YOUR STUFF

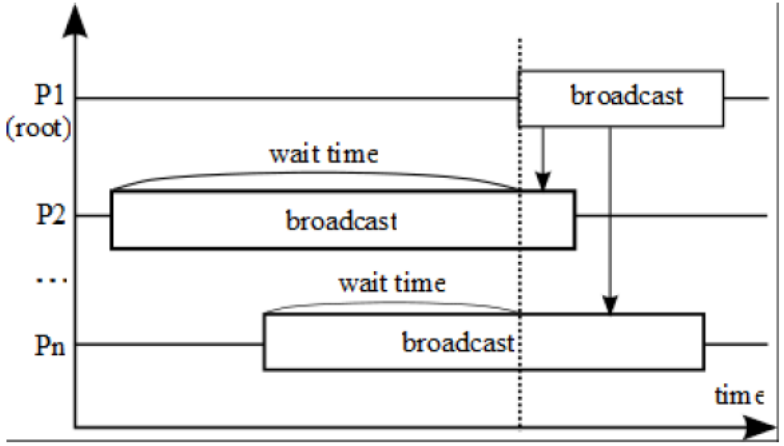
Introducing the Intel® Trace Analyzer and Collector Performance Assistant

Motivation: Improve method of performance analysis via the GUI

Solution:

- Define common/known performance problems
- Automate detection via the Intel® Trace Analyzer

Example: A “Late Broadcast” is not easy to identify with existing views



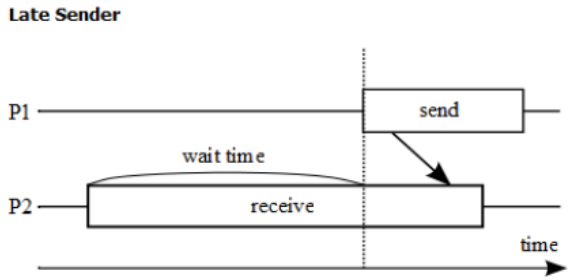
[2014]

Source: https://software.intel.com/en-us/videos/quickly-discover-performance-issues-with-the-intel-trace-analyzer-and-collector-90-beta

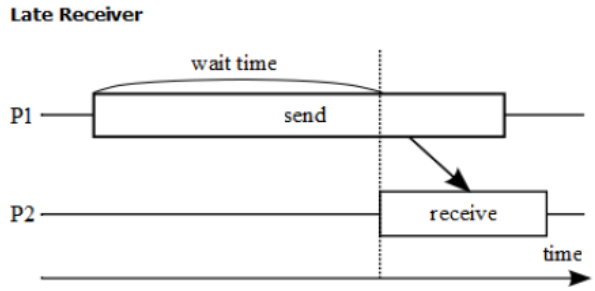
Which Performance Issues are automatically identified?

Point-to-point exchange problems:

- Late Sender

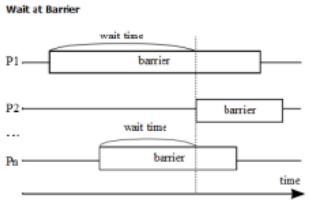


- Late Receiver

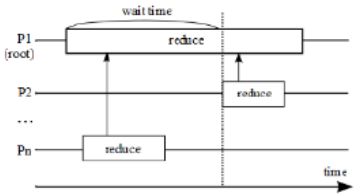


Problems with global collective operation performance:

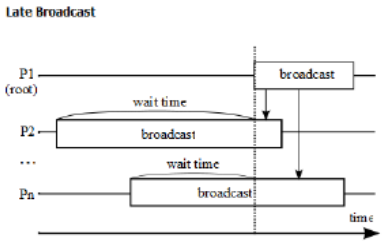
- Wait at Barrier



- Early Reduce

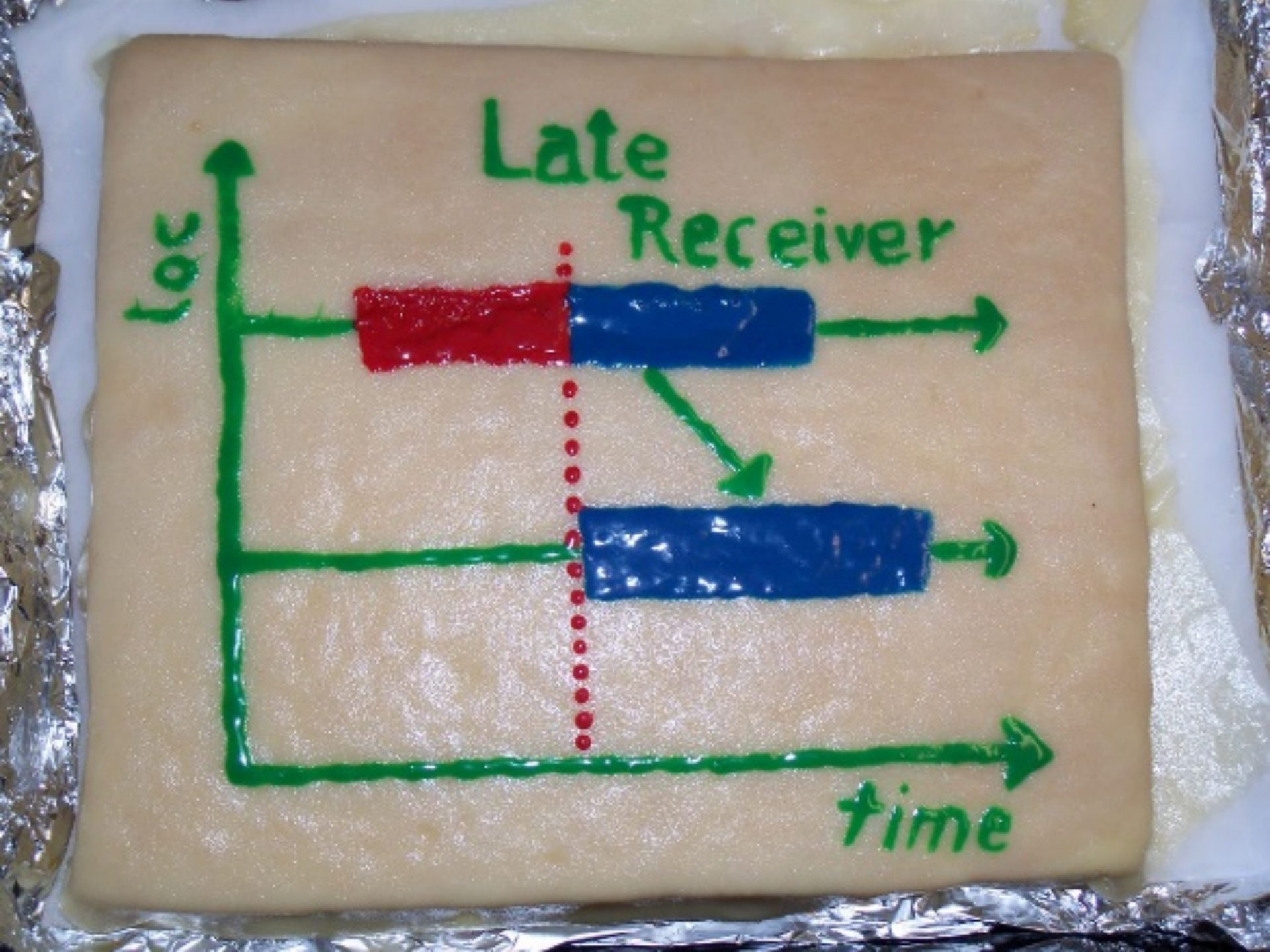


- Late Broadcast



[2014]

Source: https://software.intel.com/en-us/videos/quickly-discover-performance-issues-with-the-intel-trace-analyzer-and-collector-90-beta


Late
Receiver
loc
time

Questions?

scalasca

- http://www.scalasca.org
- scalasca@fz-juelich.de

Score-P
Scalable performance measurement infrastructure for parallel codes

- http://www.score-p.org
- support@score-p.org