37 YEARS OF PERFORMANCE TOOLS DEVELOPMENT: SUCCESS STORIES AND FAILURES

12 FEB 2024 I BERND MOHR

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

THE EARLY YEARS

1987 TO 1992 / FRIEDRICH ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

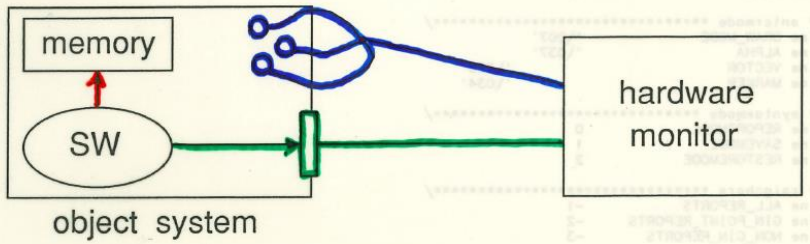
CONTEXT

- Der SFB 182
  - „Multiprozessor- und Netzwerkkonfigurationen“
  - Four 3-year phases (1987 – 1998)

- Work Package C1
  - „Messung, Modellierung und Bewertung von Multiprozessoren und Rechnernetzen“

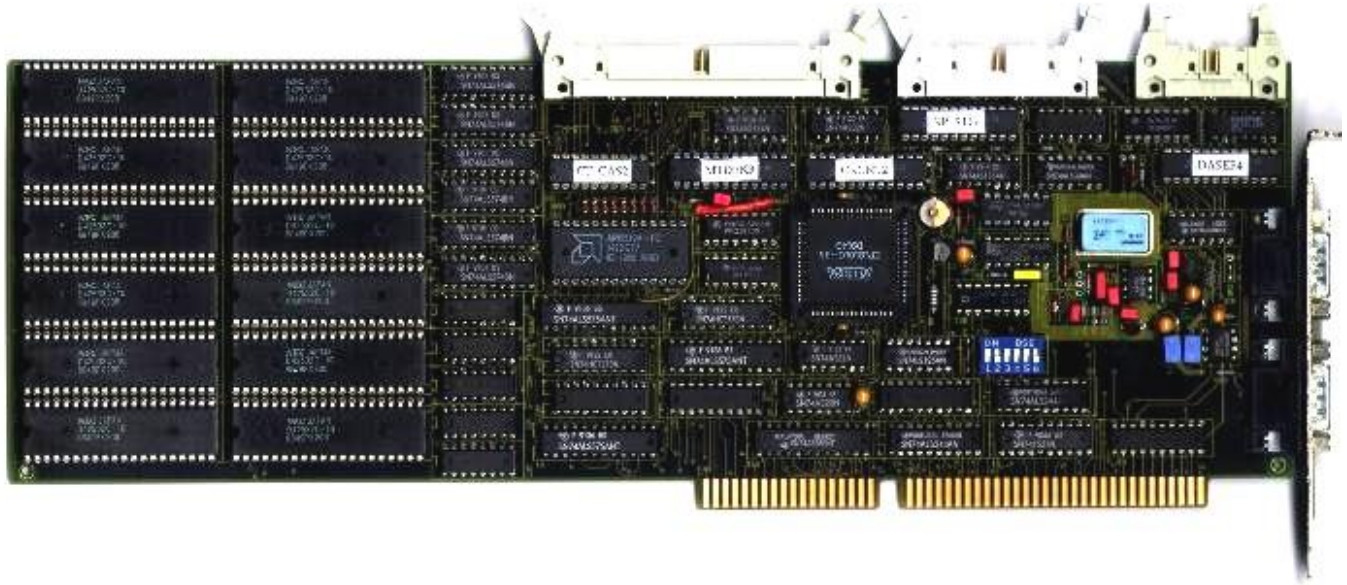
- Parallel system development @ FAU IMMD
  - EGPA (Erlangen General Purpose Array)
  - DIRMU (DIstributed Reconfigurable Multiprocesser kit)
  - MEMSY (Modular Expandable Multiprocessor System)

- monitoring techniques:
  - software
  - hardware
  - hybrid

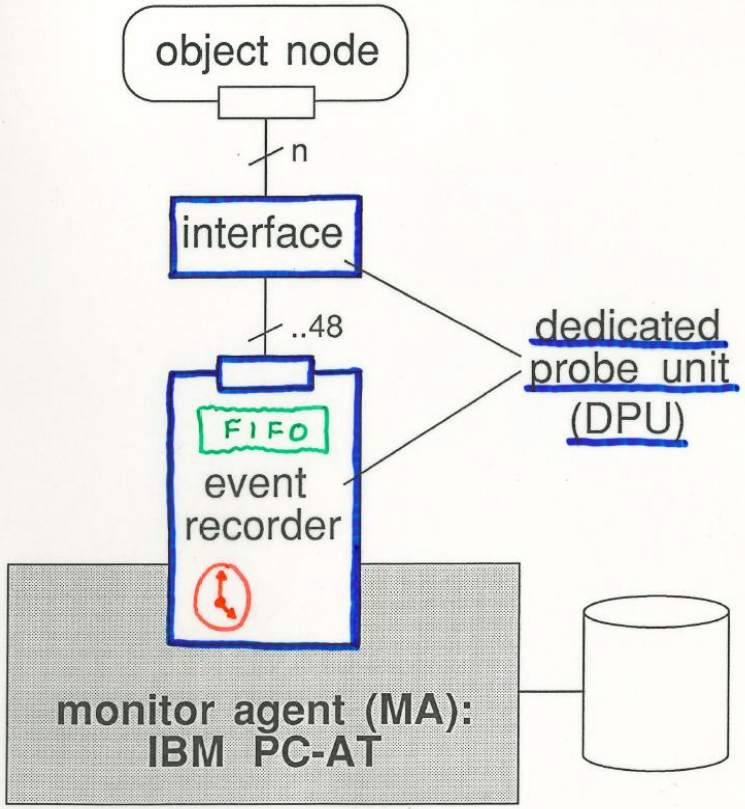


- **hybrid monitoring**:
  - **event definition** by inserting monitor instructions into the software (instrumentation)
  - **event recording** and **timestamping** with hardware

Event Recorder Board



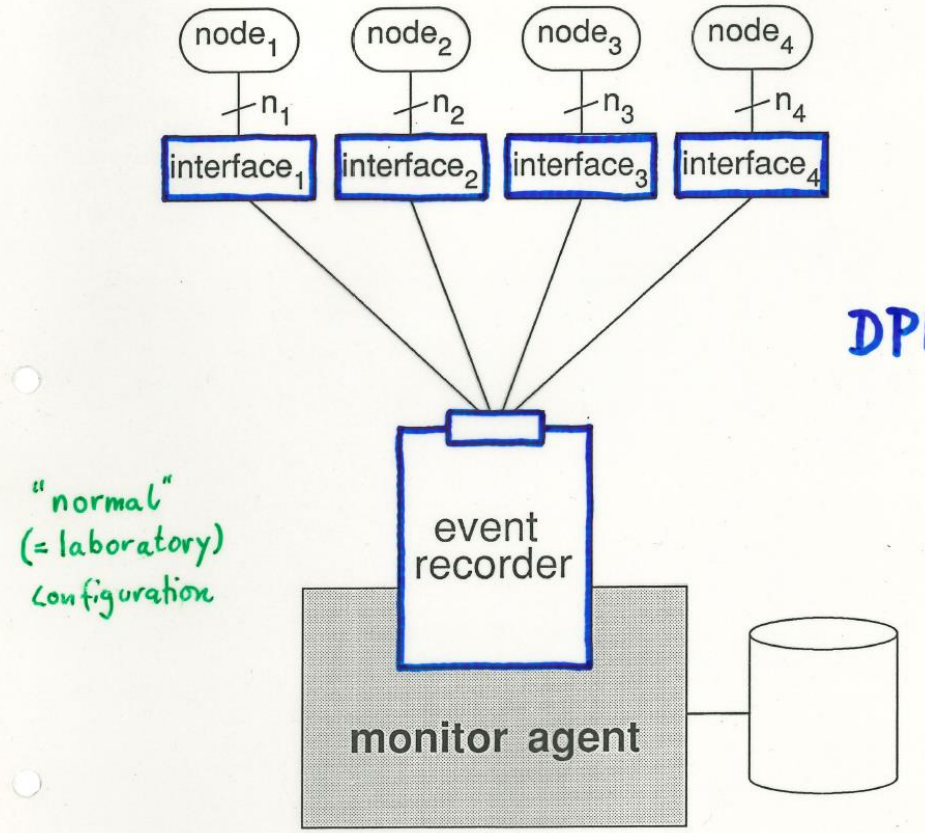
Monitoring (4)



- **event recorder**:
  - IBM PC-AT compatible host interface
  - high-resolution clock [100 ns]
  - 10,000,000 events/s peak rate
  - 32K x 96 bit event buffer (FIFO)
  - 10,000 events/s mean rate (per MA)

40 timestamp
8 flags
48 event data

Monitoring (5)

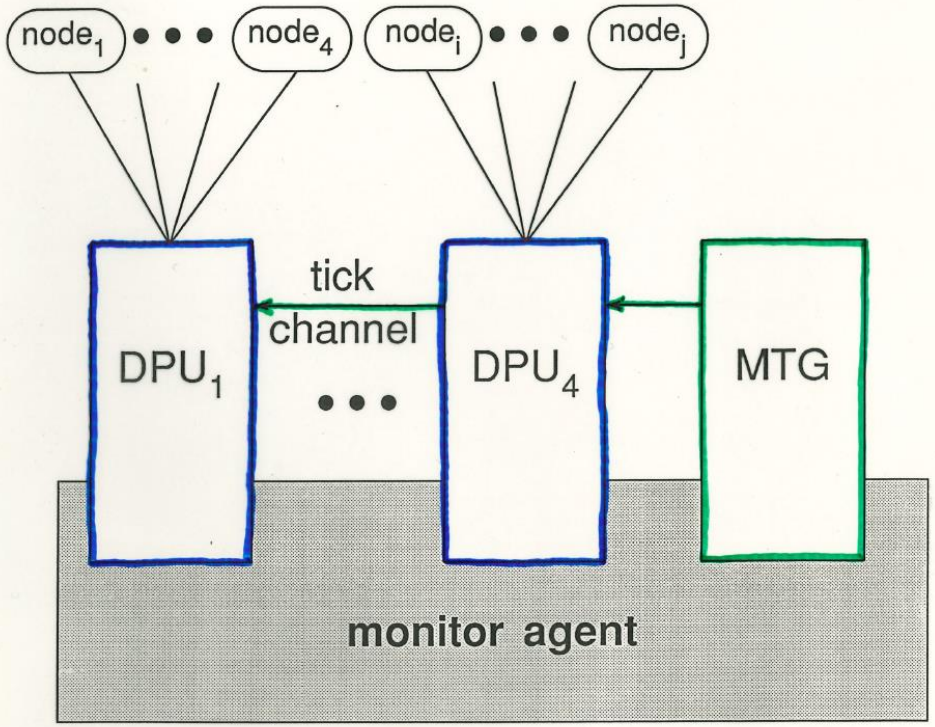


DPU

"normal" (= laboratory) configuration

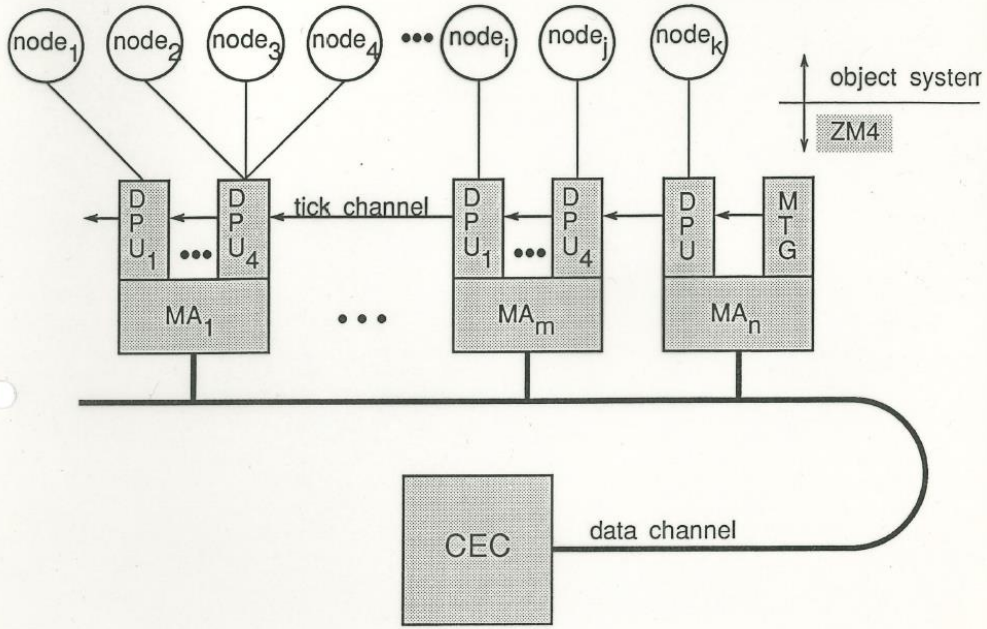
- **event recorder**:
  - 4 independent event streams

    $n_1+n_2+n_3+n_4 \leq 48$ bits



- **Measure Tick Generator:**
  - global time synchronization
  - global start / termination
  - fault-tolerant clock — sync events, sync ok detection

⇒ verification of clock data possible



- **Central Evaluation Computer**
  - central control
  - gathers traces via data channel
  - central (off-line) evaluation

distributed monitor system ZM4:
- adaptable to arbitrary computer systems
- global time with high-resolution [100 ns]

Applications

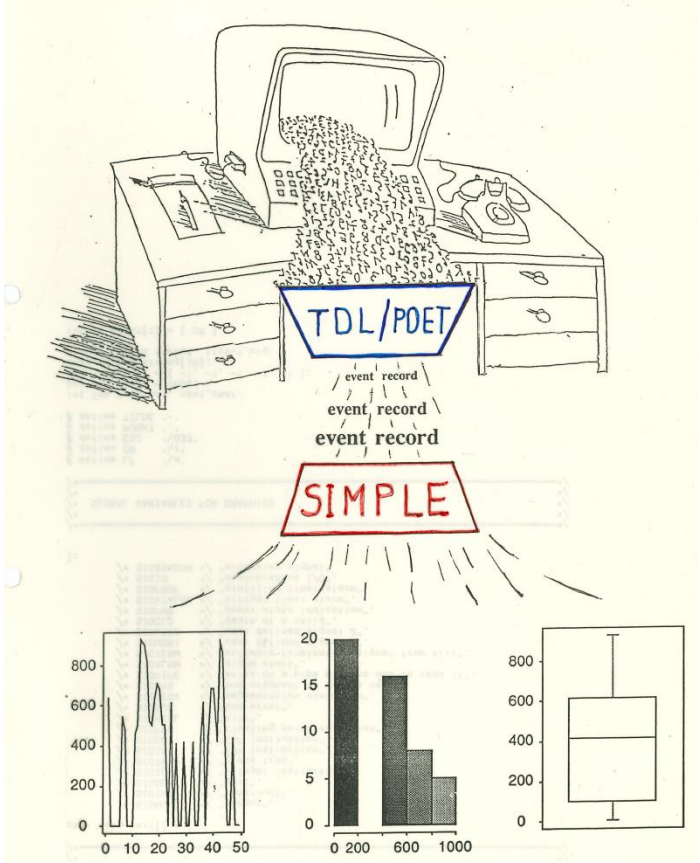
object system / operating system	monitor / interface	application	
DIRMU / DIRMOS	logic analyzer ZM4 / parallel port	numerical application simulation program	
Transputer	ZM4 / link adapter ZM4 / bus adapter	communication system TRACOS	IBM Zurich Research Lab
SUPRENUM / PEACE	ZM4 / 7 segment display	ray tracing	
IBM-PC / OS/2, MSDOS	ZM4 / Centronics	protocol software B-ISDN, FDDI	IBM ENC Heidelberg
IBM-PC / XENIX	ZM4 / Centronics	protocol software	Siemens Erlangen
SUN4 / SunOS	ZM4 / VME bus	X-Windows	
SIEMENS robot control	ZM4 / SMP bus	robot control software	Siemens Munich
CCC3280 / XELOS	software monitor	multiprocessor UNIX	
IBM-PC network	ZM4 / Centronics	Electrical Load Supervision Control System	Fudan Univ. Shanghai

Motivation (4)

- **problems** with monitoring non-sequential systems:
  - distributed system ↔ central monitor
  - global time (virtual, real)
  - non-reproducible behavior
  - "Probe Effect"
- other problem: **many different objects**
  - structure / configurations
  - operating systems
  - programming models
  - applications
- but: almost the **same** tasks

➠ requirement:
object independent
Trace Monitoring and Analysis System

S ource related and
I ntegrated
M ultiprocessor and -computer
P erformance evaluation,
mode **L** ing, and visualization
E nvironment

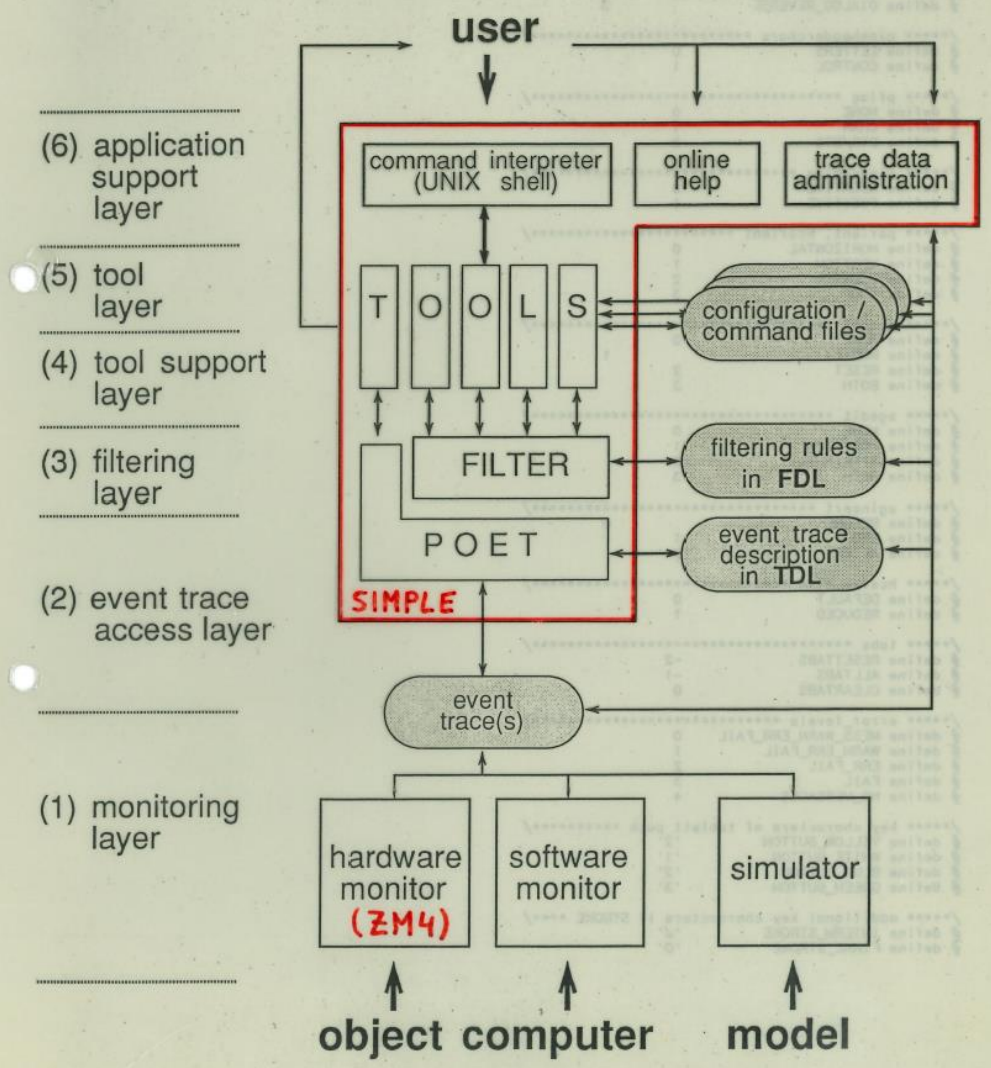


FAU

Implementation (2)

IMMD VII B

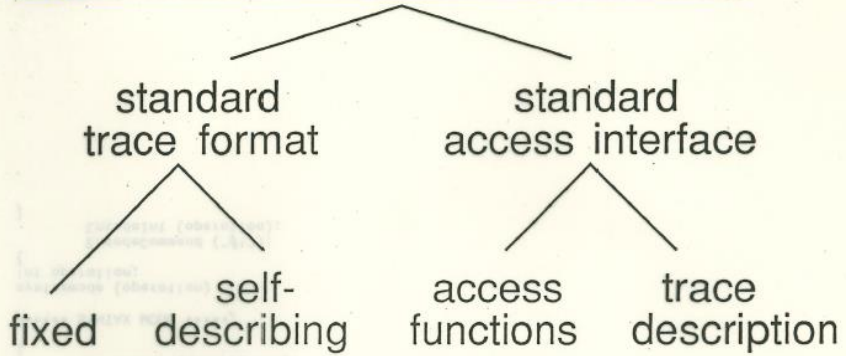
❑ **general structure** of SIMPLE:



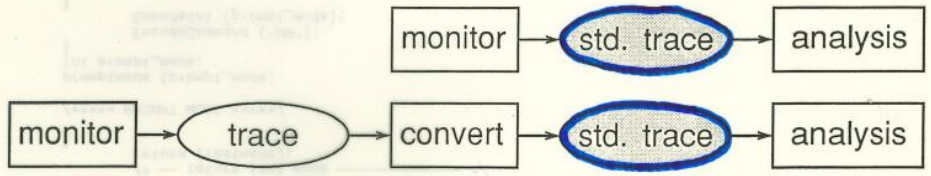
Bernd Mohr Z–6

Event Trace Access (1)

- **approaches to object independence**:



- fixed standard trace format:
  - PICL [ORNL 89], STIF [Apple 90]

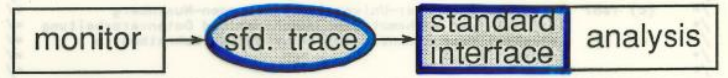


  - instrumentation workshop [LANL 90]

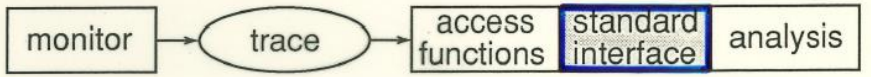
Event Trace Access (2)

- self-describing trace format:
  TRACEVIEW, Pablo [CSRD 91], ASN.1 [ISO 86]

Basic Encoding Rules

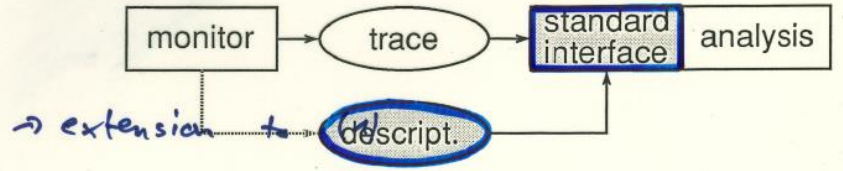


- standard access interface:
  - trace dependent access functions



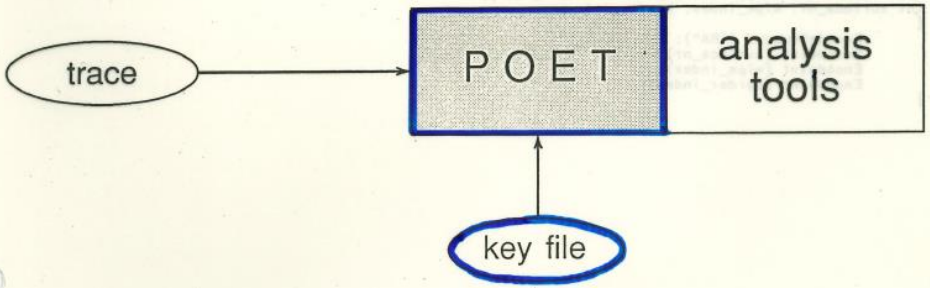
Intel → NX2

  - trace description: ASN.1 [ISO 86]
    DPM [Berkeley 86], TDL/POET [IMMD 87]



→ extension to

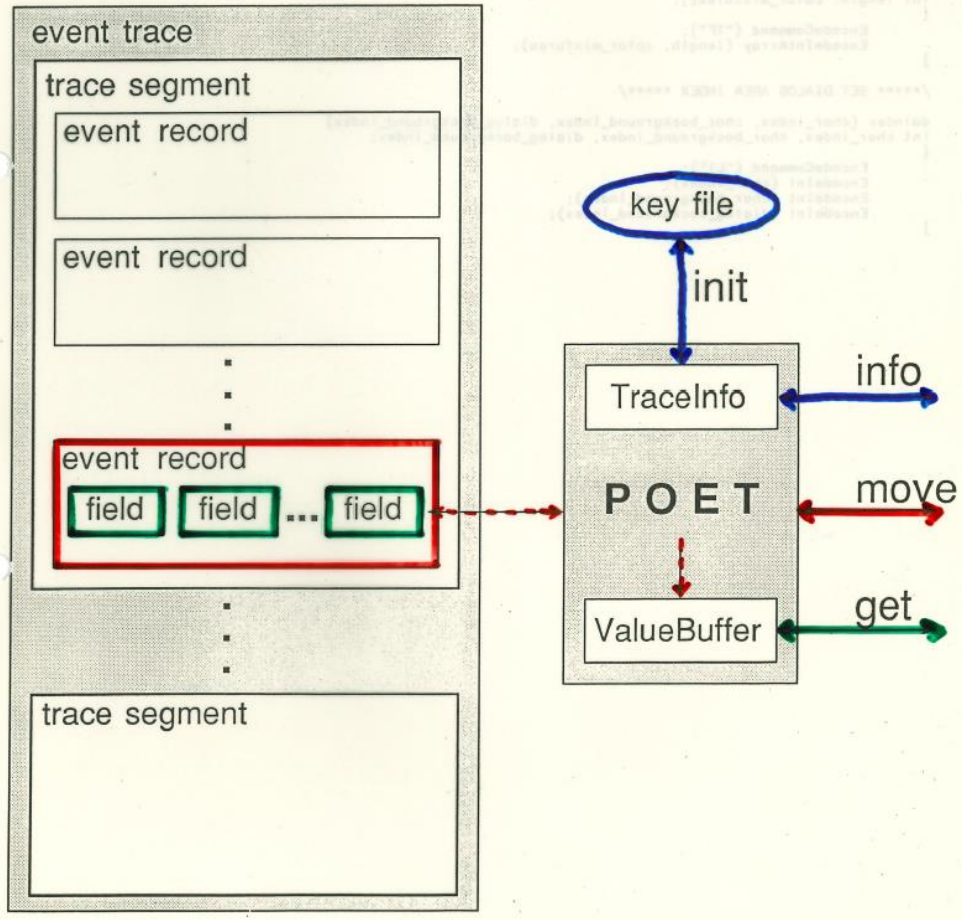
Event Trace Access (3)



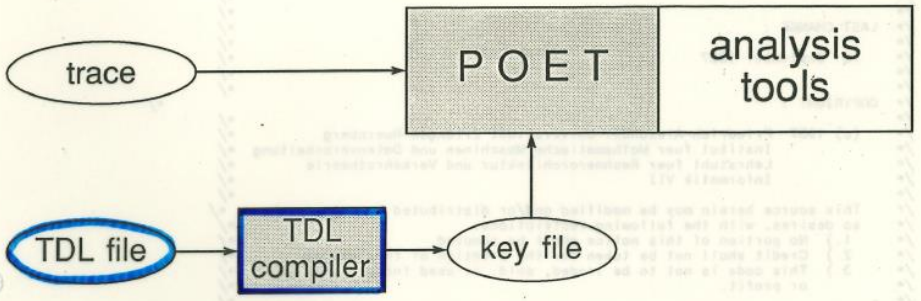
- Problem Oriented Event Trace interface
- **advantages:**
  - independent of event trace formats
  - standardized trace access interface
  - reusable function library in C
  - problem-oriented access

Event Trace Access (4)

- event trace handling with **POET**:
  - **generic abstract data type**

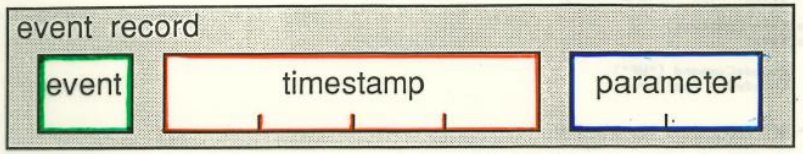


Event Trace Access (5)



- event **T**race **D**escription **L**anguage
- **differences** to other systems:
  - problem-oriented description
  - description of representation possible
  - ASCII + binary trace formats
- **advantages**:
  - trace documentation
  - syntax and consistency checks
  - fast initialization of POET

Event Trace Access (6)




```
TRACE DESCRIPTION:
   TRACE IS UNSEGMENTED;
DATA REPRESENTATION:
   4-BYTE ORDER IS 3-2-1-0;
EVENT RECORD:
   TOKEN:    NAME IS EVENT;
             LENGTH IS 1 BYTE;
             VALUES ARE [ #04, #0E, #0A, #16, #1E, ... ];
             INTERPRETATION
                #04 = 'IterationStart',
                #0E = 'SynchronizationStart',
                #16 = 'InterpolationStart',
                ...
   TIME:     NAME IS ACQUISITION;
             FORMAT IS ( UNSIGNED * 4, 100 ns );
             MODE IS POINT;
   CASE EVENT IN
   'IterationStep':
      DATA:  NAME IS IterationNumber;
             FORMAT IS INTEGER * 2;
   ...
   END
```

- **example**: FDL description

```
INIT <global variables>;

START FILTERMODULE :
    NAME IS <name>;
    ACTIONS
        IF <condition>

        DO
            PASS RECORD;
            SET <variable> = <expr>;
            SWITCH TO <name>;
        END

FILTERMODULE :
    <name>
    ACTIONS
        PASS RECORD;
        IF <condition>
            INCREMENT <variable>;
        IF <condition>
            DECREMENT <variable>;
        IF <condition> EXIT;
```


FAU Event Trace Analysis (5) IMMD VII B

- **example**: FDL description

```
INIT COUNTER no_of_proc;
START FILTERMODULE :
    NAME IS search_begin;
    ACTIONS
        IF EVENT=='PrgStart' AND
            PROCESSOR=='Master'
        DO
            PASS RECORD;
            SET no_of_proc = 1;
            SWITCH TO program_only;
        END
FILTERMODULE :
    NAME IS program_only;
    ACTIONS
        PASS RECORD;
        IF EVENT == 'PrgStart'
            INCREMENT no_of_proc;
        IF EVENT == 'PrgEnd'
            DECREMENT no_of_proc;
        IF no_of_proc == 0 EXIT;
```

trace validation:

- test whether measurement OK
- confirm expected behavior
- find unexpected behavior

- tool **CHECKTRACE**
  - for standard tests

- tool **VARUS**
  - **VA**lidation **RU**les checking **S**ystem
  - allows user specified and problem-oriented assertions:

```
ASSERT
  NUMBER (EVENT=='IterationStart') ==
  NUMBER (EVENT=='IterationEnd')
ELSE "Iteration counting error" ;
ASSERT
  IterationNumber INCREASING BY 1
ELSE "IterationNumber sequence error" ;
```


- example: **trace statistics**

- command language ⇔ tool features

  FREQUENCY <field name>
  DISTANCE <expr>

- identifiers, values ⇔ trace data

  FREQUENCY EVENT
  DISTANCE EVENT == 'receive' AND
  NODE == 'server'

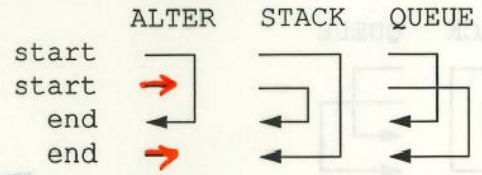
- for application-dependent features:
  - predefined **standard names**
    - EVENT
    - ACQUISITION
    - NODE
    - PROCESS
  - predefined **standard event types**
    - send / receive
    - ...

Trcstat Description Files

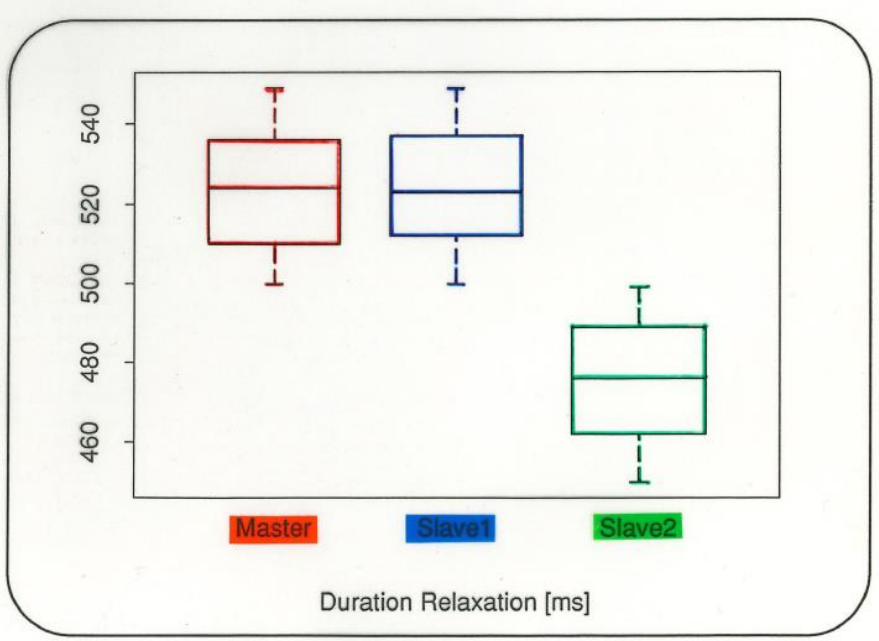
- general:
  - **FREQUENCY** <fieldname>
    [**NAME** = <identifier>]
  - **DISTANCE** <situation>
    [**NAME** = <identifier>]
  - **DURATION** <situation> <situation>
    [**NAME** = <identifier>]
    MODE = **ALTER** | **STACK** | **QUEUE**

  <situation> := <event> [@ <fieldname>]

- duration modes:




```
#01H  DURATION  'RelaxB'  'RelaxE'  PROCESSOR

#01S   461 [ ms ] on Slave2
#01S   523 [ ms ] on Master
#01S   527 [ ms ] on Slave1

          ...

#01S   470 [ ms ] on Slave2
#01S   502 [ ms ] on Master
#01S   520 [ ms ] on Slave1

#01T          Master  Slave1  Slave2
#01T     no:     231     231     231     693
#01T    min:     500     500     450     450  [ ms ]
#01T    max:     549     549     499     549  [ ms ]
#01T   mean:     524     524     475     508  [ ms ]
#01T    var:     207     215     216     748  [ ms ]
```



Event Trace Analysis (8)

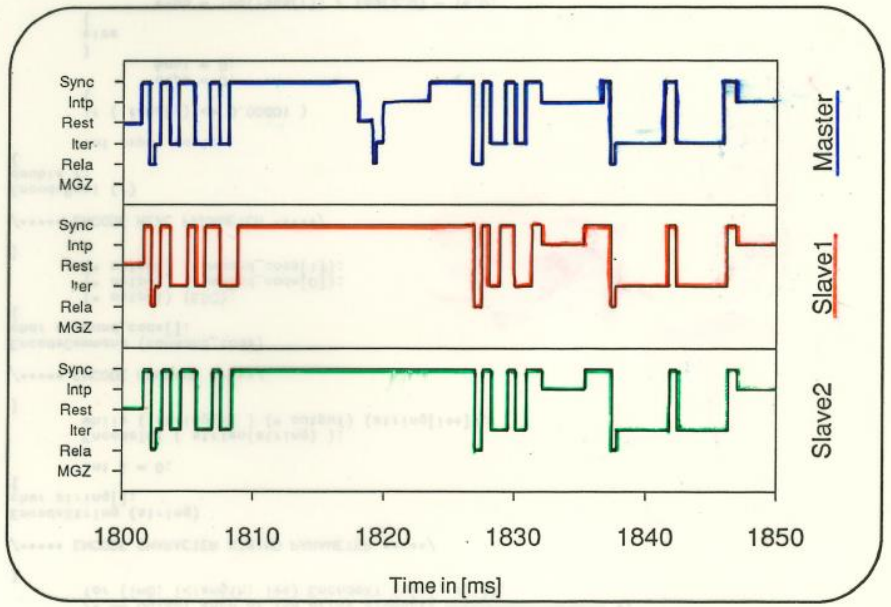
- **activity**:
  - interval in the dynamic behavior
  - defined by sequences of events
- inspired by **EDL** [Bates]
- tool **FACT**
  - **F**ind **ACT**tivities
  - allows user-specified and problem-oriented activity definitions as **regular event expressions**

```
ACTIVITY Iteration IS
    IterationStart  → IterationStep *
                    → Sync ?
                    → IterationEnd
END
```

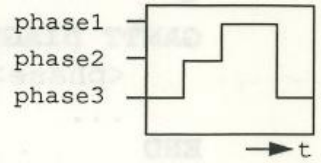
Event Trace Analysis (3)

- **Gantt diagram**:
  - program activities versus time
  - functional dynamic behavior
  - dependencies of activities
  - duration of activities

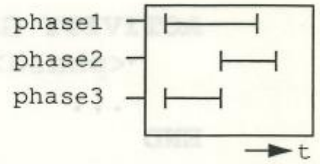
- tool **GANTT**



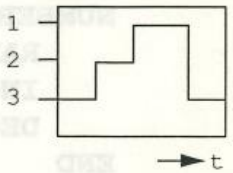

```
• GANTT DIAGRAM [<name>] WITH
      <phase> : <evlist> ;
      ...
  END
```




```
• ACTIVITY GANTT DIAGRAM [<name>] WITH
      <phase> : <evlist> TO <evtlist> ;
      ...
  END
```




```
• NUMBER GANTT DIAGRAM [<name>] WITH
      RANGE IS <num> TO <num>
      INCREASING BY <evlist> ;
      DECREASING BY <evlist> ;
  END

• FOR <tokenname> <interpretlist> DO
      <ganttdescriptions>
  END
```

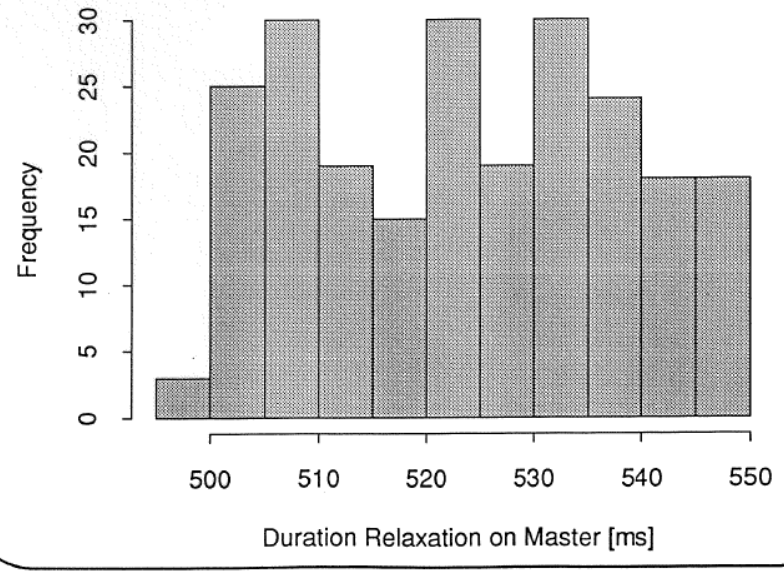



```
RESOLUTION IS  ms
LOWER BOUND IS  1800
UPPER BOUND IS  1850

FOR ALL PROCESSOR DO
  GANTT DIAGRAM WITH
    Sync  : SynchronizationStart;
    Intp  : InterpolationStart;
    Rest  : RestrictionStart;
    Iter  : IterationStart;
    Rela  : RelaxationStart;
    MGZ   : MGStart;
  END
END
```

example: trace analysis

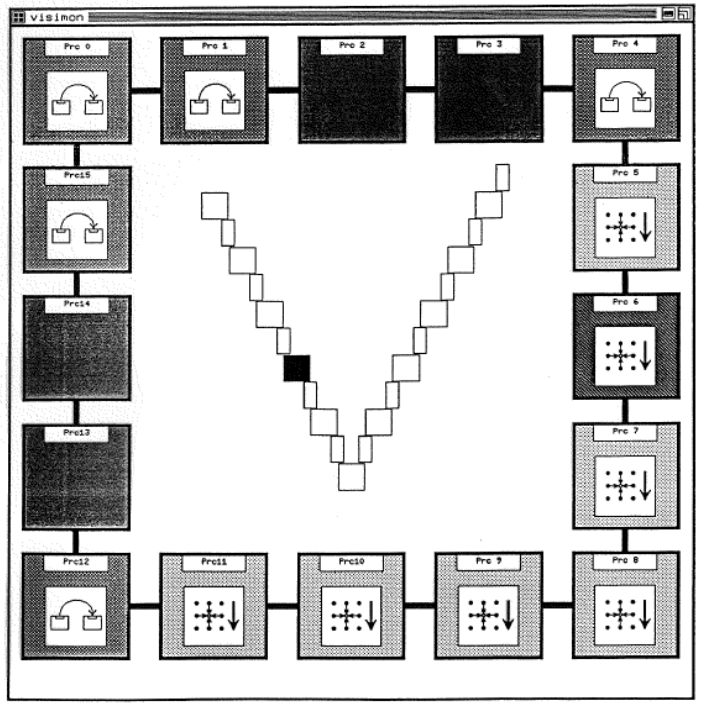
- data analysis and graphics package **S** (AT&T)
  - high-level programming environment
  - interactive query language
  - S-POET interface

```
> initkey ("mgz.key")
> readtrace ("mgz.trc")
> end <– ACQUISITION [ EVENT==’RelaxE’ &
                       PROCESSOR==’Master’ ]
> beg <– ACQUISITION [ EVENT==’RelaxB’ &
                       PROCESSOR==’Master’ ]
> hist ( beg – end )
```



example: animation

- animation tool **VISIMON**
  - based on X-Windows
  - user specified animation description
  - program and data animation
  - "slow-motion" and "event-by-event" mode



ASSESSMENT

[~1990 !!!]

- **What worked**
  - Hybrid instrumentation
  - Sophisticated and highly advanced hardware monitor (**ZM4**)
  - Fully flexible trace analysis framework (**SIMPLE**)
    - Generated by different sources (SW tracing, HW monitoring, LANalyzer, log files, simulation outputs, ….)
    - Of diverse applications (MP Unix, network stack, simulations, robot control software)
- **What didn’t**
  - ZM4 too expensive for really large configurations
  - SIMPLE unusable by non-expert (highly complex programming required)
  - IFF issue: what’s wrong? Investigated target? Description?

THE POSTDOC YEARS

1993 TO 1995 / UNIVERSITY OF OREGON, EUGENE

CONTEXT

- ARPA funded project “pC++”

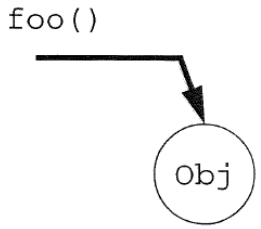
  - **Programming Environments, Compiler Technology and Runtime Systems for Object-Oriented Parallel Processing**
    - Dennis Gannon, Indiana University
    - Postdocs: Pete Beckman, Francois Bodin
    - **pC++** compiler and runtime system, **Sage++** toolkit

  - **Languages, Libraries and Performance Evaluation Tools for Scalable Parallel Systems**
    - A. Malony, J. Cuny, University of Oregon
    - Postdoc: Bernd Mohr
    - **TAU** program analysis tools

pC++ – The Programming Language Ideas

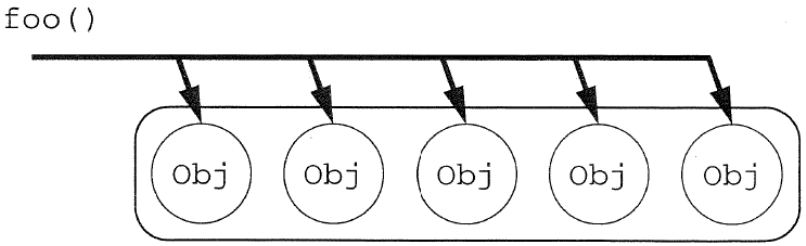
- **regular C++**: programmers apply operators and functions to objects as "messages"

```
class Obj {
    int x;
    void foo();
};

Obj myObj;

myObj.foo();
```



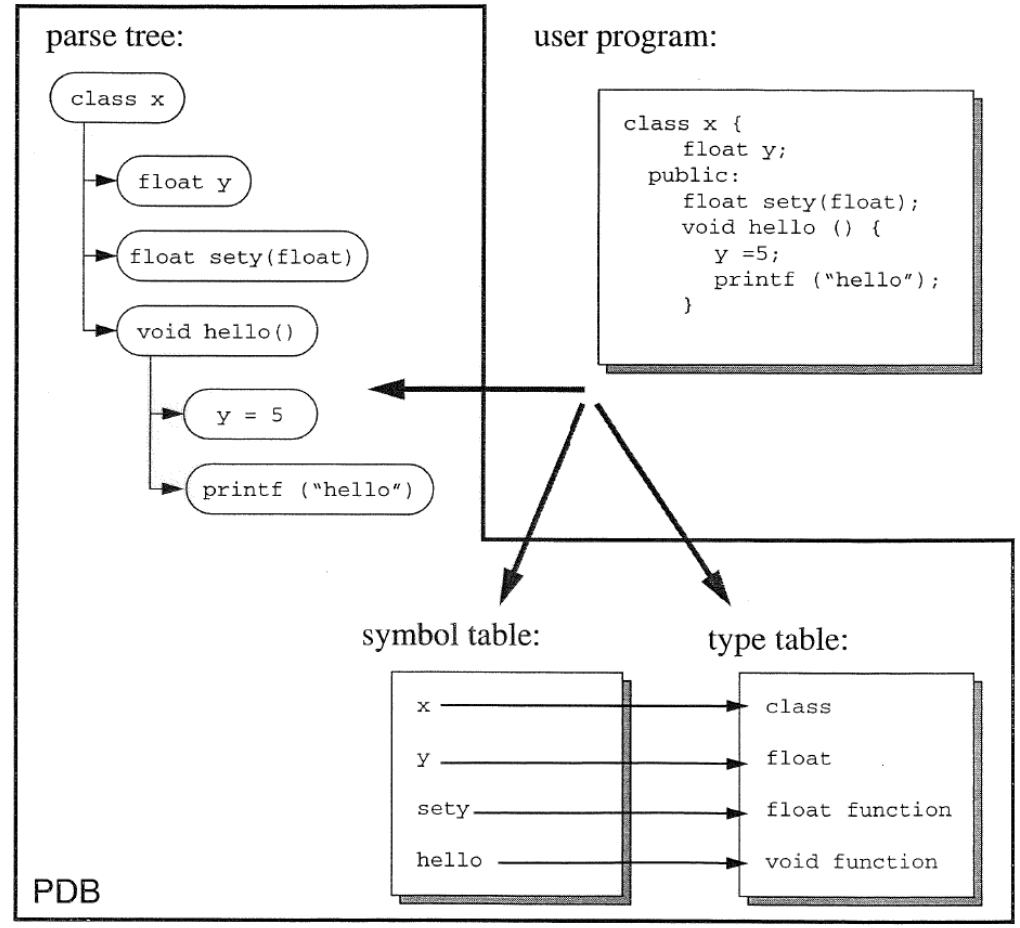
- **pC++**: this concept is extended so that an operator or function can be applied to a large set, grid, array (:= **collection**) of objects (:= **elements**) in parallel

```
Collection Vector { ... };

Vector<Obj> paraObj(AlignObj, DistrObj);

paraObj.foo();
```

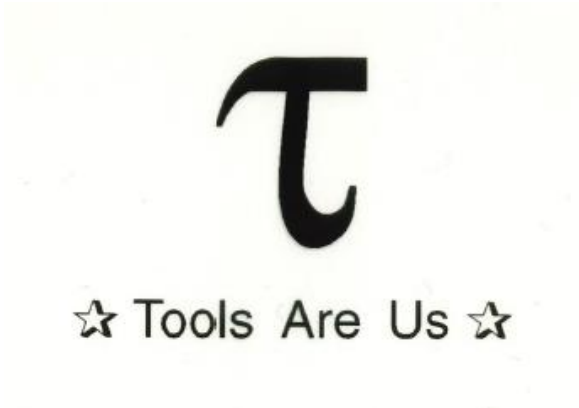


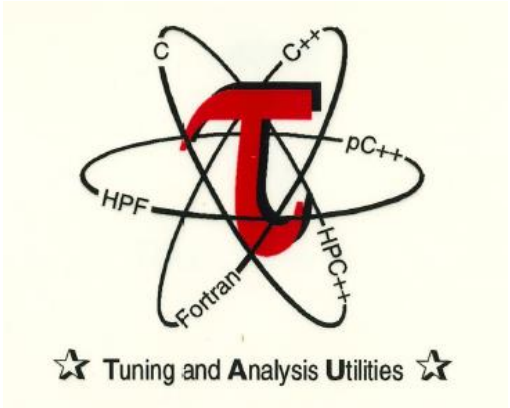
Sage++

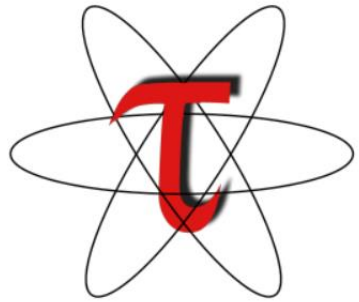
- C++ class library for building program analysis and transformation tools
- contains parsers for Fortran 77 / 90, ANSI C, and C++
- organized as a class hierarchy for accessing and modifying the parse tree, symbol table, and type table



TAU LOGO EVALUATION



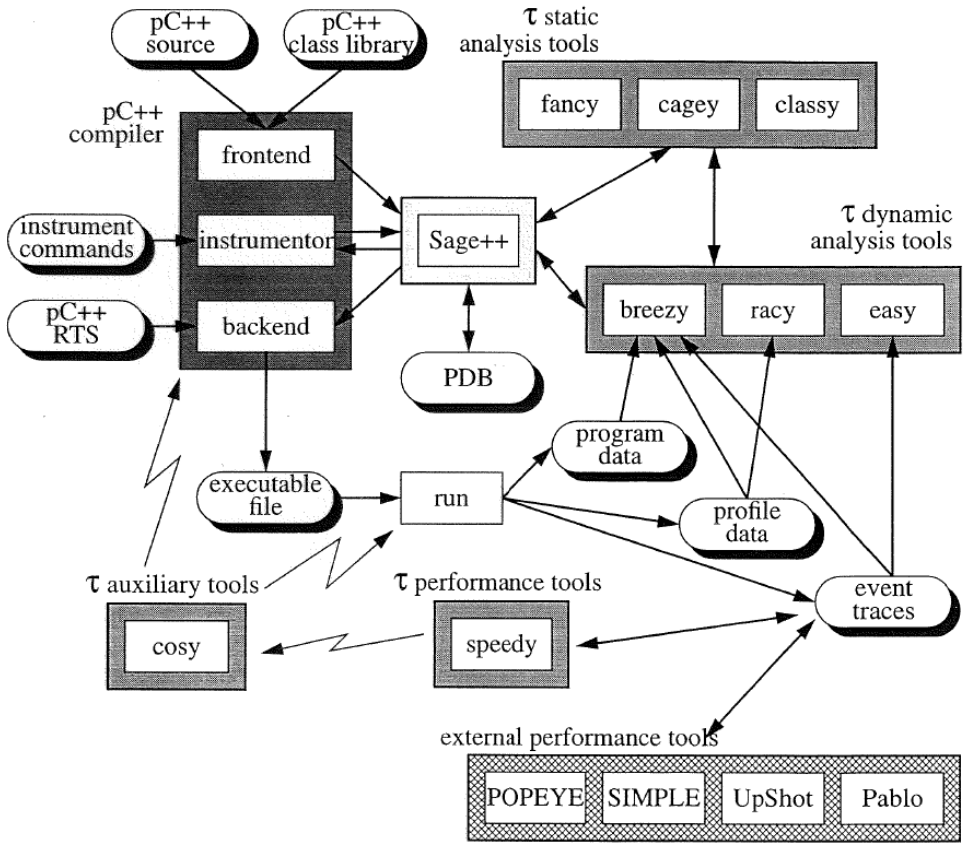




TAU Performance System ®

https://www.cs.uoregon.edu/research/paraducks/

The pC++ Programming Environment



☆ Tools Are Us ☆

- Currently available τ tools:
  - **cosy** (***CO***mpile manager ***S***tatus displa***Y***)
  - **fancy** (***F***ile ***AN***d ***C***lass displa***Y***)
  - **cagey** (***CA***ll ***G***raph ***E***xtended displa***Y***)
  - **classy** (***CLASS*** hierarch***Y*** browser)
  - **racy** (***R***outine and data ***AC***cess profile displa***Y***)
  - **speedy** (***S***peedup and ***P***arallel ***E***xecution ***E***xtrapolation ***D***displa***Y***)
  - **breezy** (***BR***eakpoint ***E***xecutive ***E***nvironment for visuali***Z***ation and data displa***Y***)
- Prototypes:
  - **easy** (***E***vent ***A***nd ***S***tate displa***Y***)
  - **dandy** (***D***istributed ***A***rray ***N***avigator ***D***displa***Y***)
  - **crafty** (***C***ont***R***ol flow ***A***nd ***F***unc***T***ion displa***Y***)
  - **geeky** (***GE***eky ***E***diting and symbol loo***K***up displa***Y***)
  - **POPEYE, DAQV** (data and performance visualization)
- τ can work with a local or remote pC++ language system
- τ originally designed for C++/pC++ programs

τ

☆ Tools Are Us ☆

- Providing a user (program-level) view
  - τ graphical interface objects represent pC++ language level objects: collections, classes, methods, functions
- Support for high-level, parallel programming languages
  - τ designed and implemented in concert with pC++ system
  - τ translates low-level performance data ⇔ language level
- Integration with compilers and runtime systems
  - τ integrated with pC++ runtime system
  - τ uses Sage++ for access to ***P***rogram ***D***ata ***B***ase (PDB)
- Portability, extensibility, and retargetability
  - τ implemented using C/C++ and Tcl/Tk for portability
  - τ implemented as **hypertools** for extensibility
  - τ uses Sage++ for retargetability
- Usability
  - τ tool objects represent **hyperlinks**
  - τ supports **global features**
  - τ has on-line hypertext help

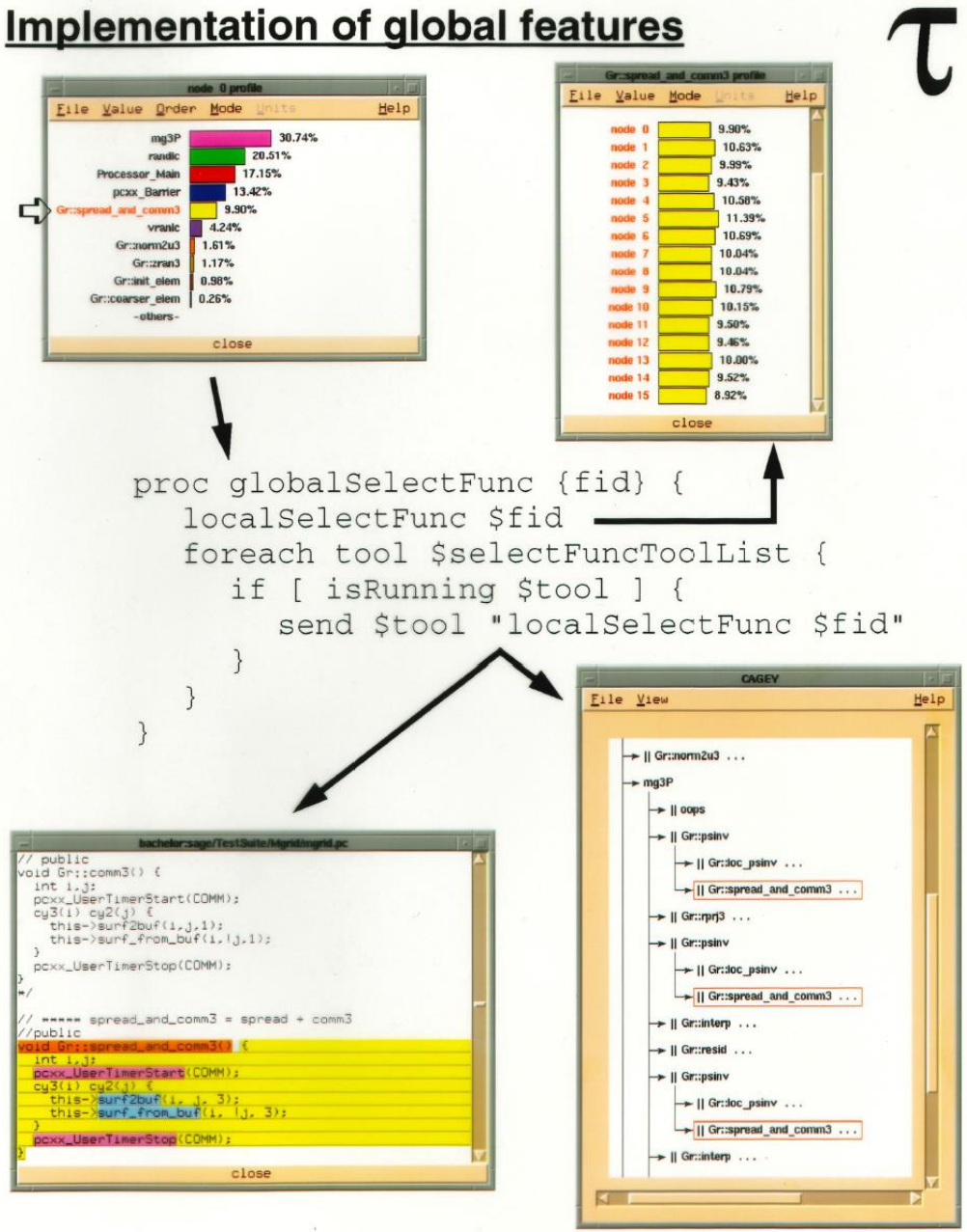
τ

☆ Tools Are Us ☆

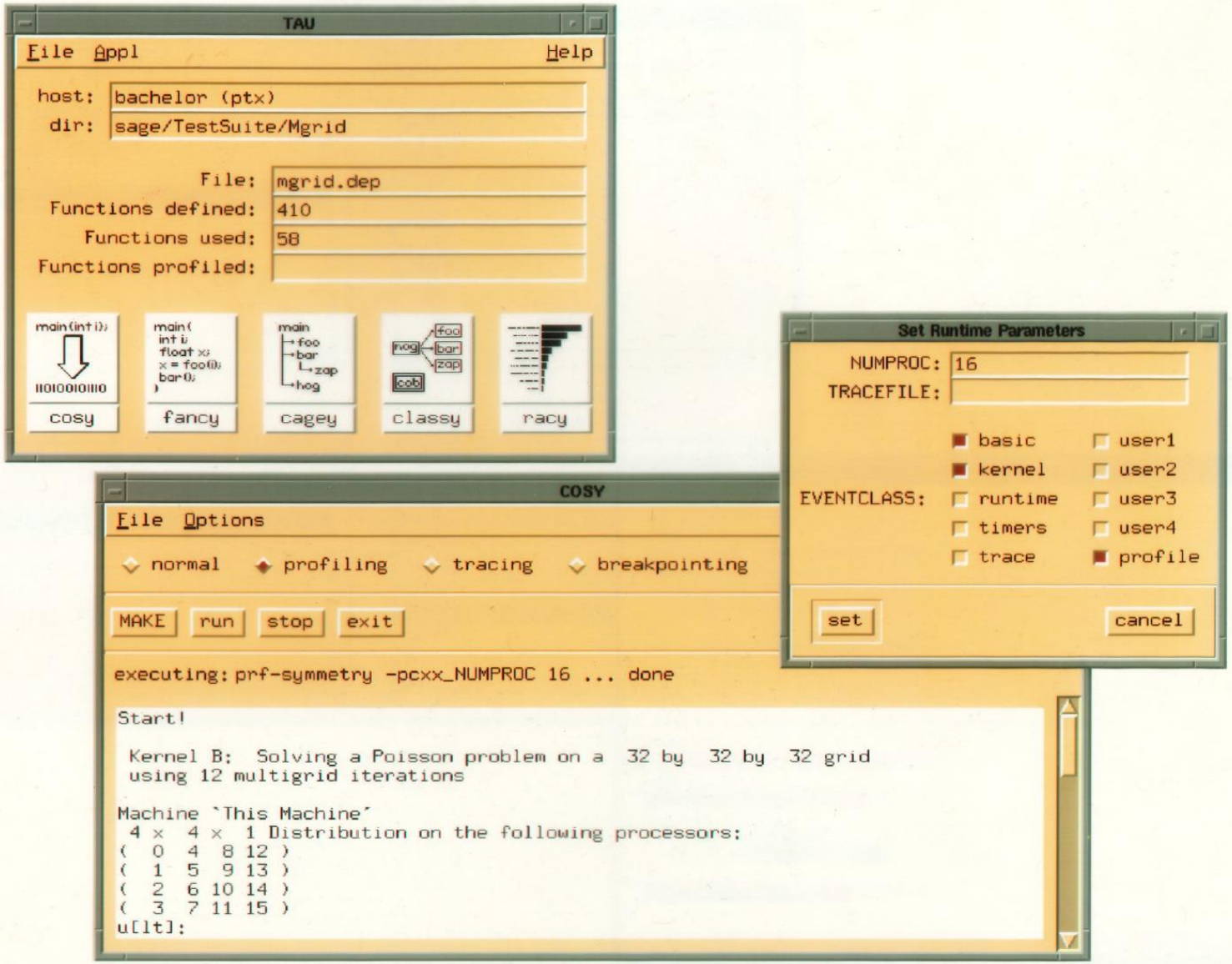
- **Hypertools**
  - τ tools are distinct tools, but they act in concert as if they were a single, monolithic application
  - implemented using **hyperlinks** and **global features**
- **Hyperlinks**
  - τ graphical interface objects act like in hypermedia documents
  - selecting an object of interest brings up a more detailed or related view of the object

    e.g., selecting a class in the class hierarchy graph displays a table of class members
- **global features**
  - **synchronized hyperlinks**: execution of a global feature in one tool automatically updates views in the other tools
  - currently implemented:
    - `select-function`
    - `load-depfile`
    - `select-class`
    - `select-callsite`

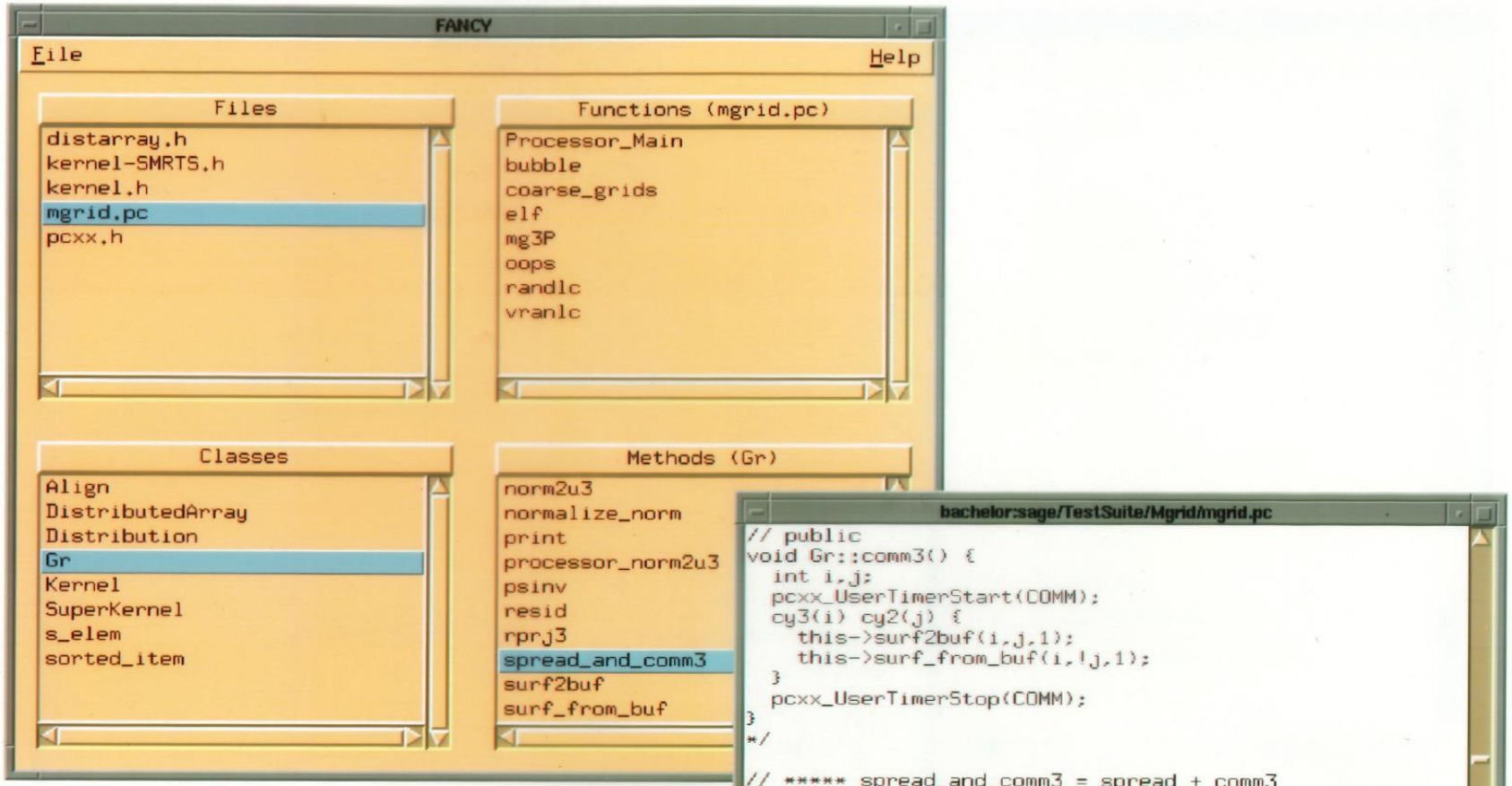
Implementation of global features



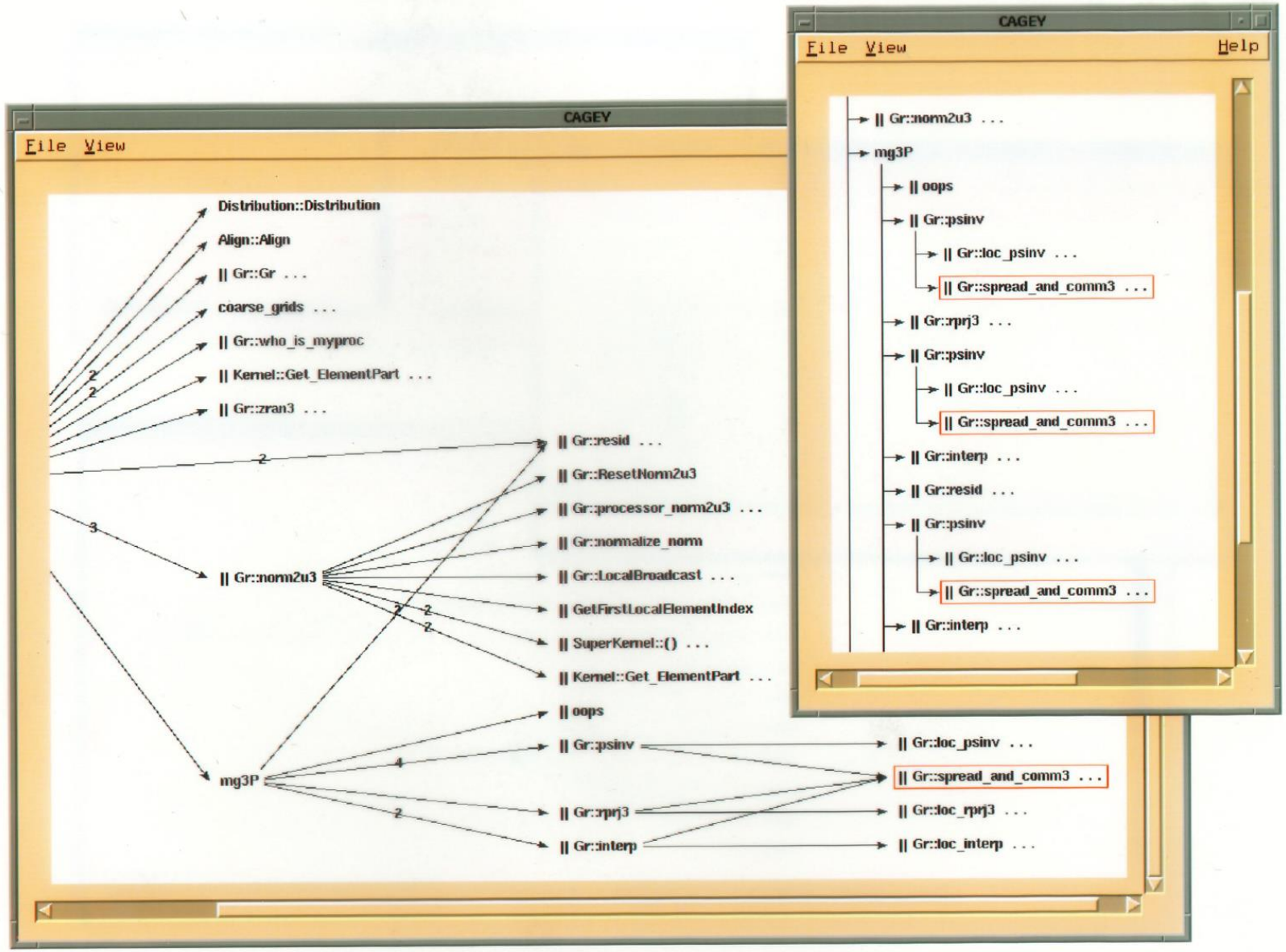
TAU / COSY
Main Control Panel / Compile and Execute Manager



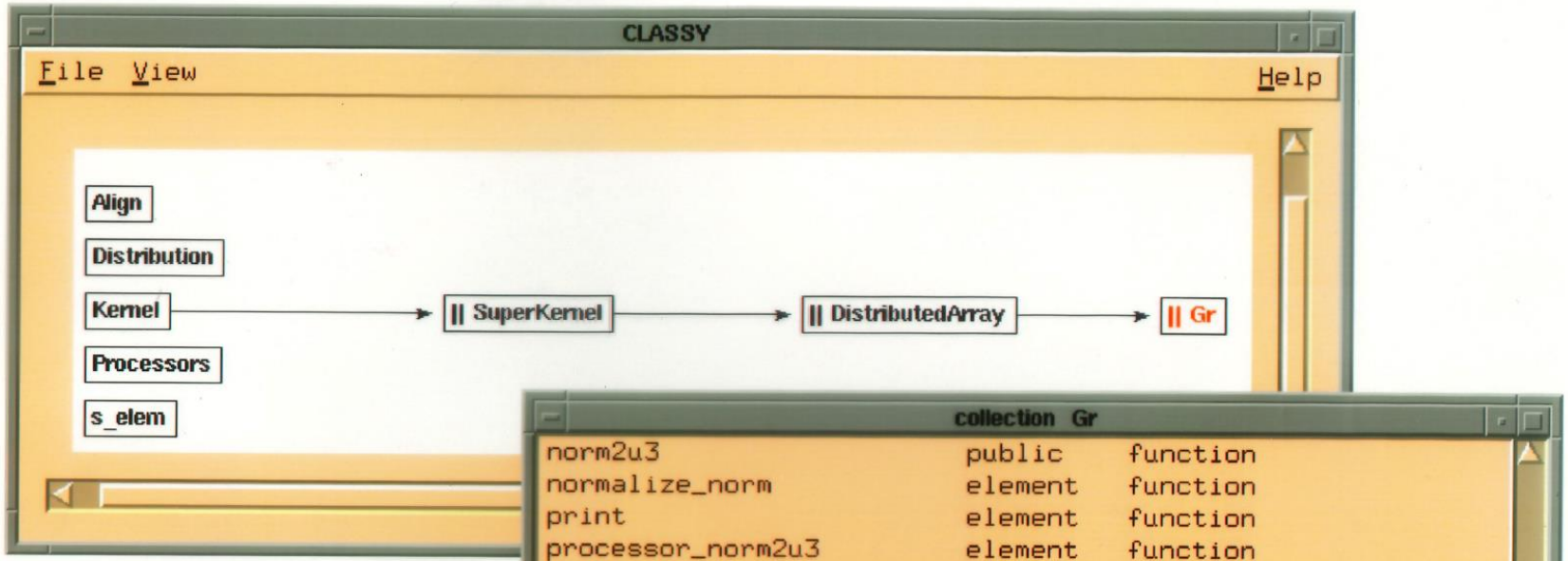
FANCY
Source Code Browser

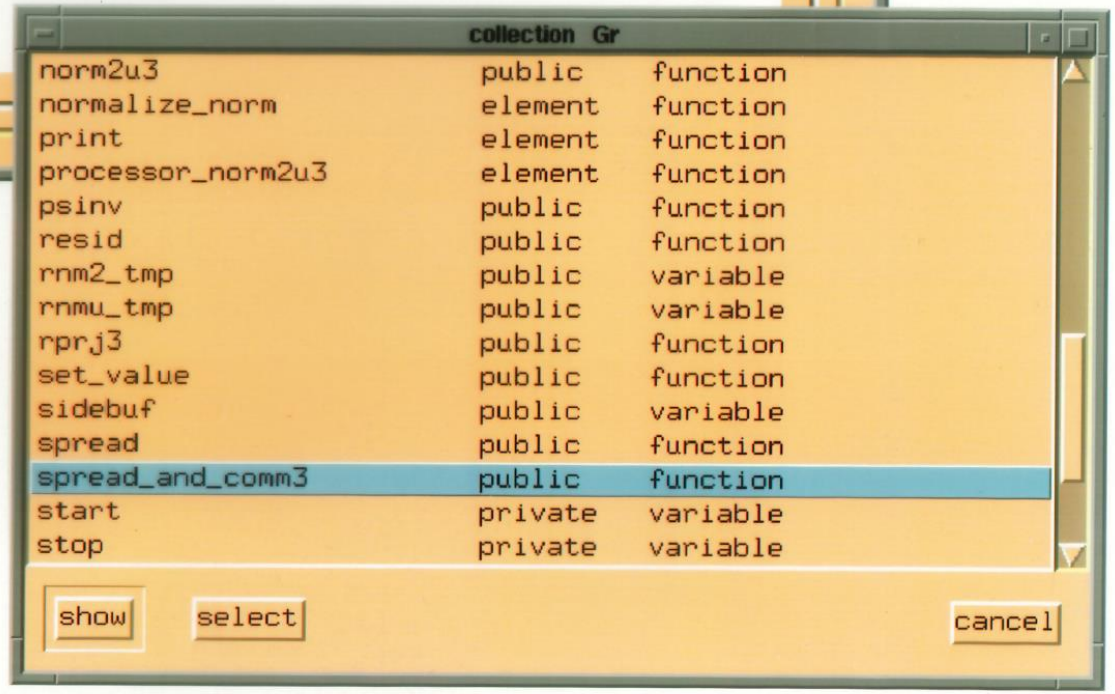


CAGEY
Static Callgraph Browser



CLASSY
Class Hierarchy Browser





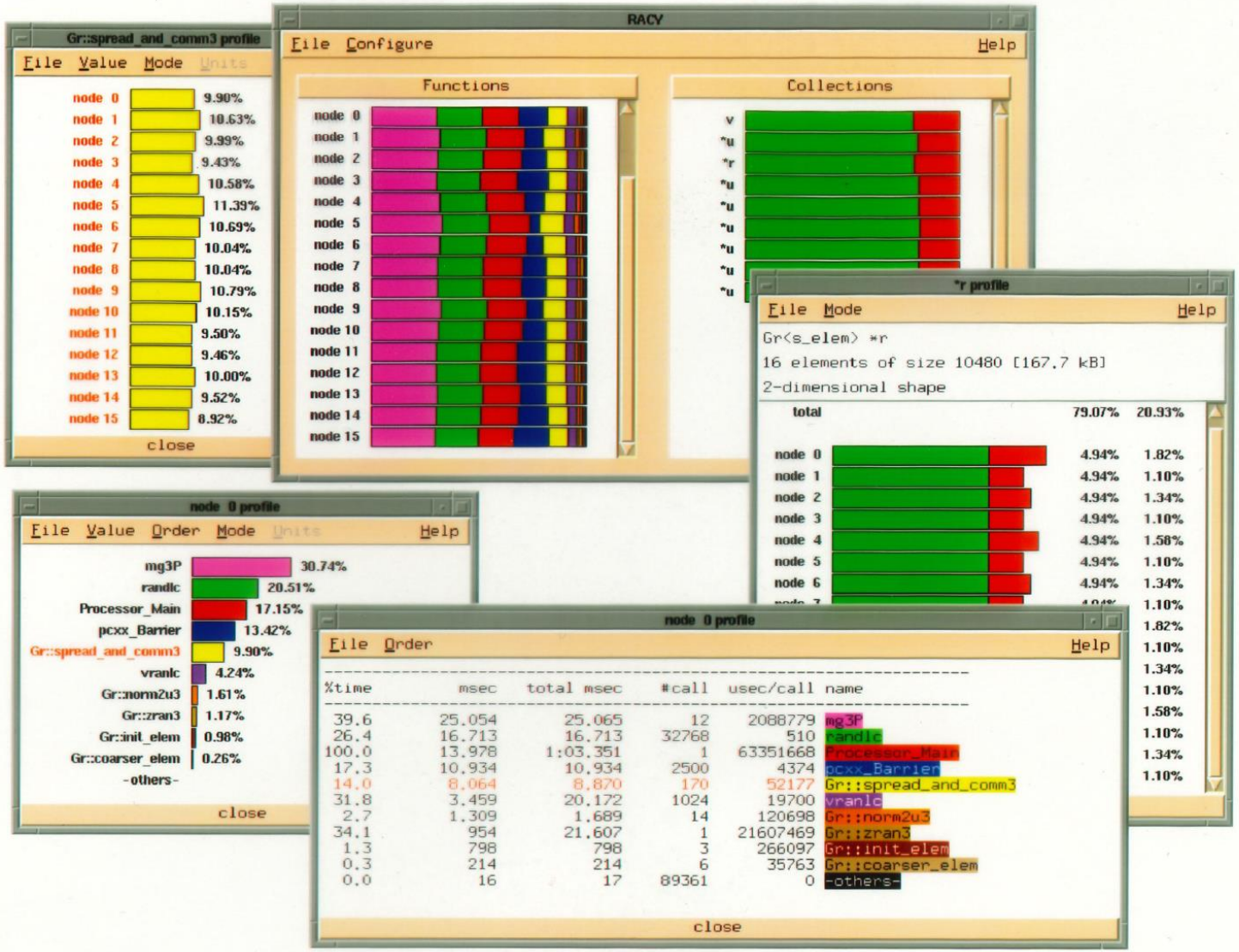
PPROF / RACY
Parallel Program Profiler

- uses **instr** tool to dynamically instrument
  - entry and exit of
    - functions, class member functions, constructors, and destructors
    - in the user application and the pC++ kernel
- uses “profile-instrumented” runtime system
- data gathered for each profiled function:
  - time spent in function including and excluding its children
  - number of calls
- data gathered for each pC++ collection:
  - number of local and remote accesses per node
- profile data is stored in one file per node in a **machine-independent** format
- **pprof**: prints ASCII profile report (like UNIX’s prof)
- **racy**: graphical profile data browser

Example pprof Output

```
NODE 0:
---------------------------------------------------------------
%time   msec total msec  #call usec/call name
---------------------------------------------------------------
100.0 15.761     28.869      1  28869157 Processor_Main
 14.5  4.166      4.172      1   4172459 initw
 12.2  3.512      3.512 560567         6 Vector::get
  6.9      9      2.004      2   1002201 DistVector::DistVector
  6.6  1.919      1.919     44     43628 Barrier
  5.9  1.694      1.694     32     52958 Vector::allocData
  5.5    132      1.578      1   1578414 poisson_solve
  2.9      1        835     32     26117 ParSinTransform
  2.9    638        833     32     26055 Vector::SinTransform

   local    remote collection
accesses accesses num   name
   16619      496   0   F
    9789        5   1   U

// similar output from other 31 processors deleted

FUNCTION SUMMARY (mean):
---------------------------------------------------------------
%time   msec total msec  #call usec/call name
---------------------------------------------------------------
100.0 15.312     28.301      1  28301704 Processor_Main
 14.4  4.058      4.064      1   4064819 initw
 12.4  3.519      3.519 563042         6 Vector::get
  7.9  2.246      2.246     44     51065 Barrier
  7.1      9      2.004      2   1002249 DistVector::DistVector
  5.6    116      1.578      1   1578486 poisson_solve
  4.8  1.348      1.348     32     42138 Vector::allocData
  3.0      1        845     32     26418 ParSinTransform
  3.0    640        843     32     26356 Vector::SinTransform

COLLECTION SUMMARY:
---------------------------------------------------------------
DistVector<Vector> F, collection #0
      513 elements of size 4216, 1-dimensional
      532996 local / 15841 remote accesses
DistVector<Vector> U, collection #1
      513 elements of size 4216, 1-dimensional
      282720 local /   423 remote accesses
```

RACY
Parallel Program Profile Visualizer



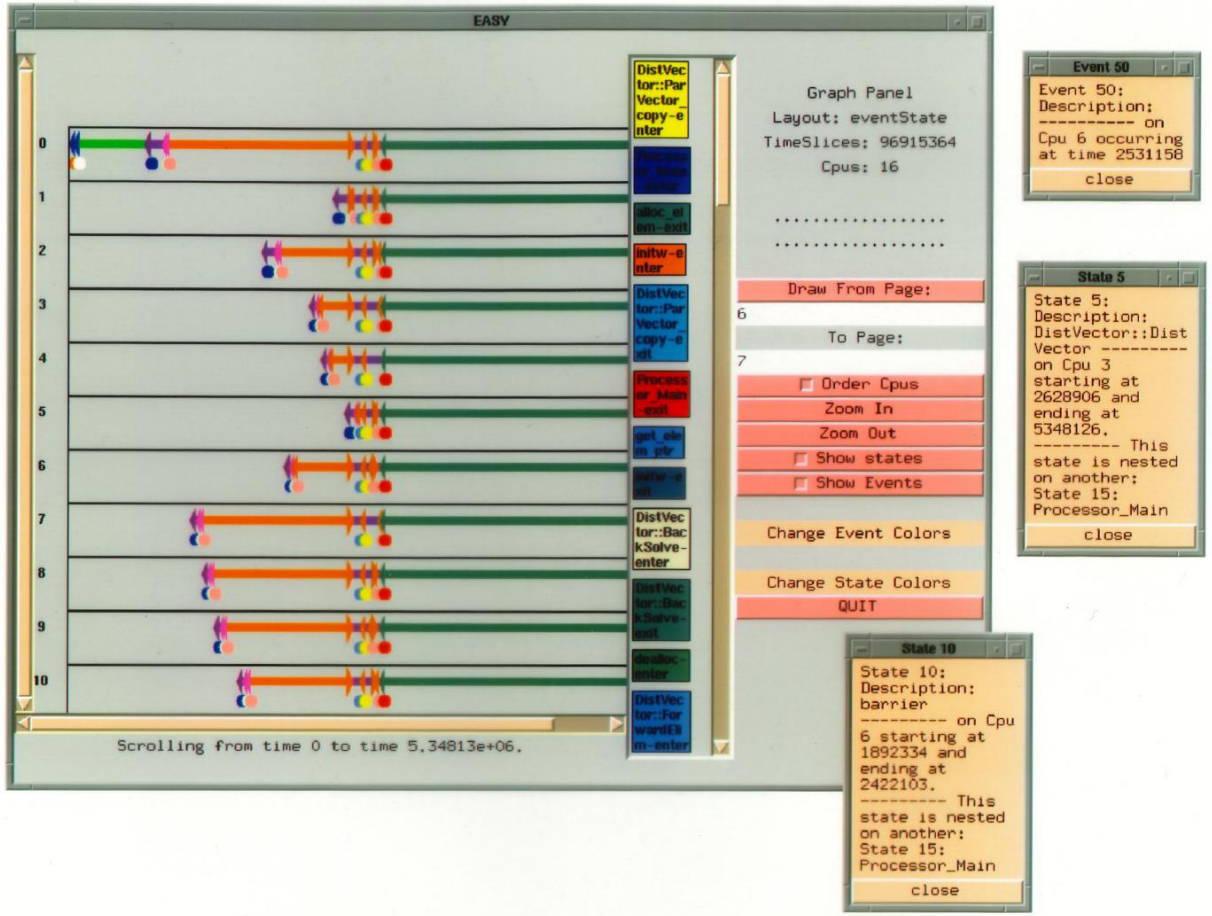
Event Tracing Support Tools

- **pcxx_merge**:
  - merges node traces to global system trace according to timestamps
  - establishes global timescale, if target system doesn't have global clock
- **pcxx_convert**: trace format converter
  - generic ASCII
  - alog
  - SDDF

Event Trace Tools

- **easy**
  - τ event trace browser for alog format
- **SIMPLE** (University of Erlangen, Germany)
  - trace format independent event trace analysis environment
- **Upshot** (Argonne National Laboratory)
  - simple+portable X11 event trace browser for alog format
- **Pablo** (University of Illinois)
  - sophisticated event trace analysis environment based on SDDF format

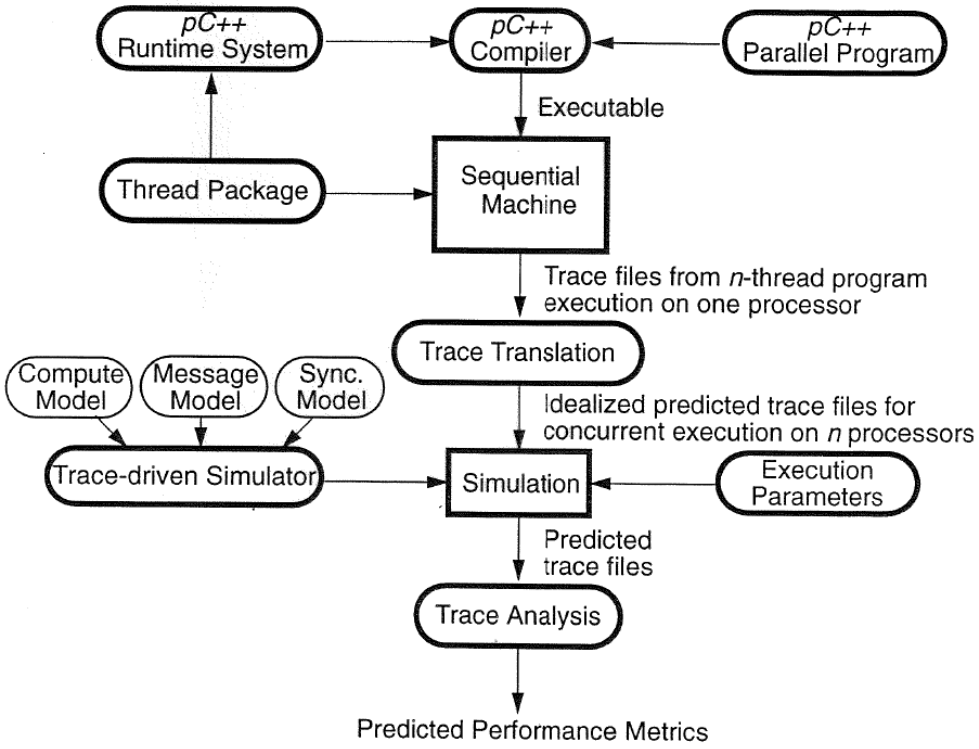
EASY
Event Trace and State Browser



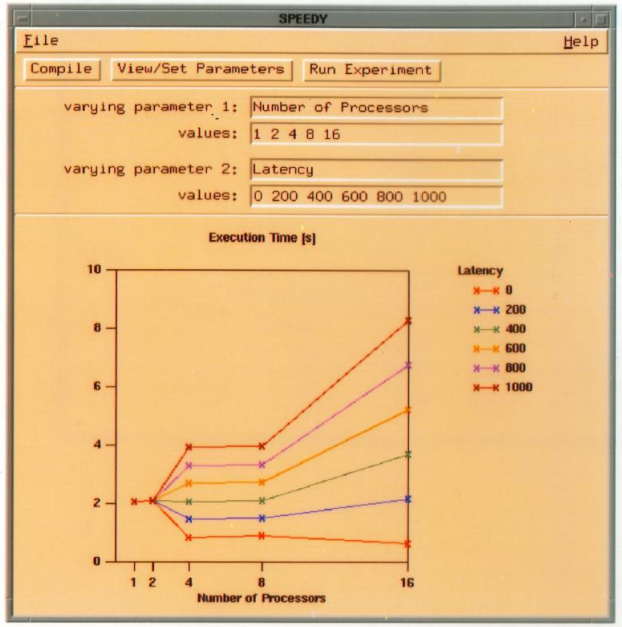
ExtraP
Performance Extrapolation and Analysis

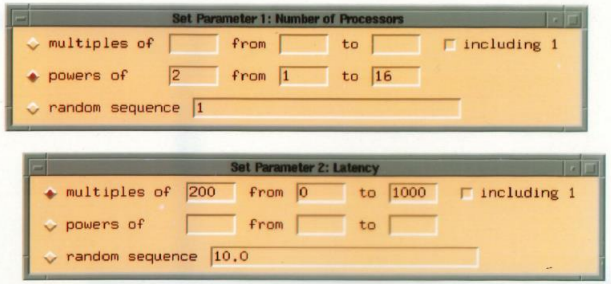
Felix ☺

- ExtraP
  - high-level event tracing of a *n*-thread pC++ program on a uniprocessor workstation
  - trace-driven simulation for prediction of performance on *n*-processor parallel machine

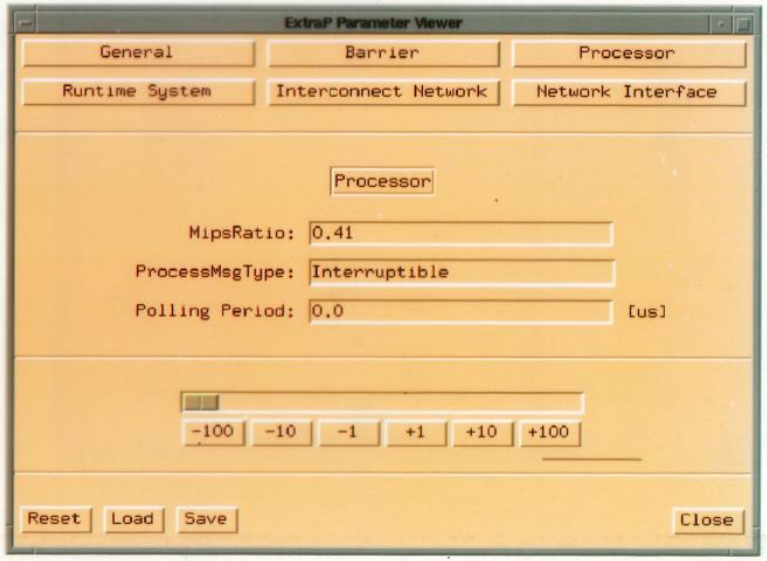


SPEEDY
Performance Extrapolation and Analysis





ExtraP



JÜLICH SUPERCOMPUTING CENTRE

pC++ – Supported Systems

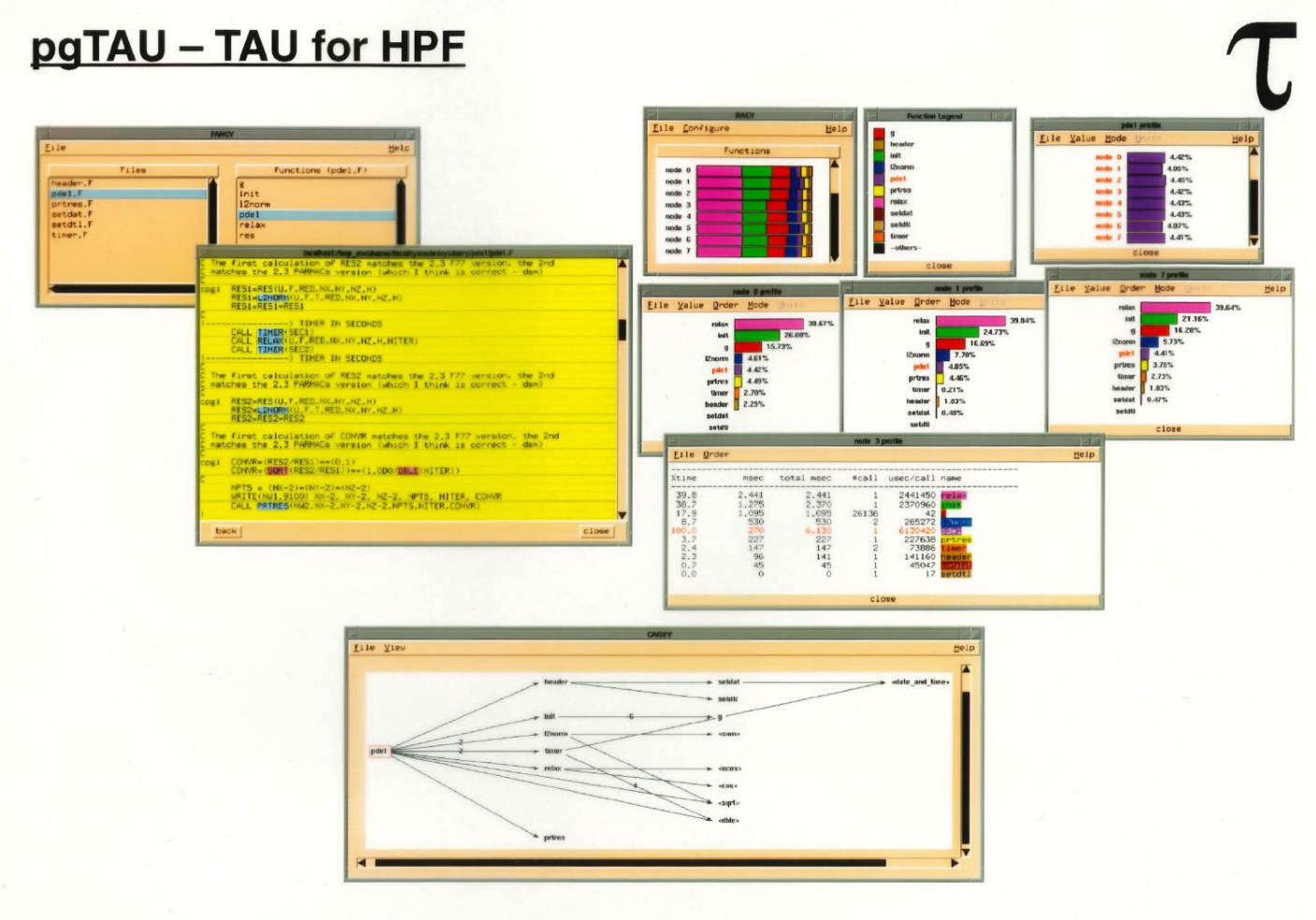
- Shared memory systems
  - Kendall Square KSR-1 / KSR-2
  - Sequent Symmetry (under Dynix + PTX)
  - SGI (Power) Challenge + Onyx
  - Convex SPP-1
  - (BBN TC2000)

- Distributed memory systems
  - TMC CM-5
  - Intel Paragon
  - IBM SP-1 / SP-2
  - Cray T3D / T3E
  - Meiko-CS2
  - Workstation Clusters with PVM + MPI (homogeneous)

- UNIX Workstations (SUN, HP, DEC, IBM, SGI, ...)
  - serialized
  - thread-based (Pthreads, LWP, AT&T tasks, Awesime)

➠ The same pC++ program will run without modification on all platforms

BEYOND PC++



- prototype port of
  - **fancy** (function browser)
  - **cagey** (callgraph browser)
  - **racy** (profile data browser)

  to HPF compiler system of The Portland Group, Inc.

- changes needed for static browsers
  - **pgdep**
    - tool for generating HPF PDB (Program Data Base)
    - generated from intermediate f77 sources
  - **om**
    - new object manager which provides the TAU standard static browser interface to HPF PDB

- changes needed for profiling
  - HPF compiler already supports instrumentation
  - rewrite of the profiler runtime system functions to output **pprof** / **racy** compatible profile data files

ASSESSMENT

- **What worked**
  - Early fully featured parallel programming environment (pC++ and TAU)
    - Easy to use (build, run, analyze)
    - Global features (hyper tools)
    - Although build for / integrated into pC++, TAU was easy to retarget
  - **Undecided**
    - Moat innovative or worst configuration system (before GNU configure or Cmake)
  - **What didn’t**
    - C++ parsing is just to complicated to be implemented within an University project
    - Didn’t support traces very well

THE LATER YEARS

1996 TO NOW / JÜLICH SUPERCOMPUTING CENTRE

CONTEXT: 25 YEARS OF AUTOMATIC TRACE ANALYSIS

- 1999 – 2004

APART

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

- Sequential analyzer EXPERT
- http://www.fz-juelich.de/zam/kojak/*

- 2006 – now

VI-HPS

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

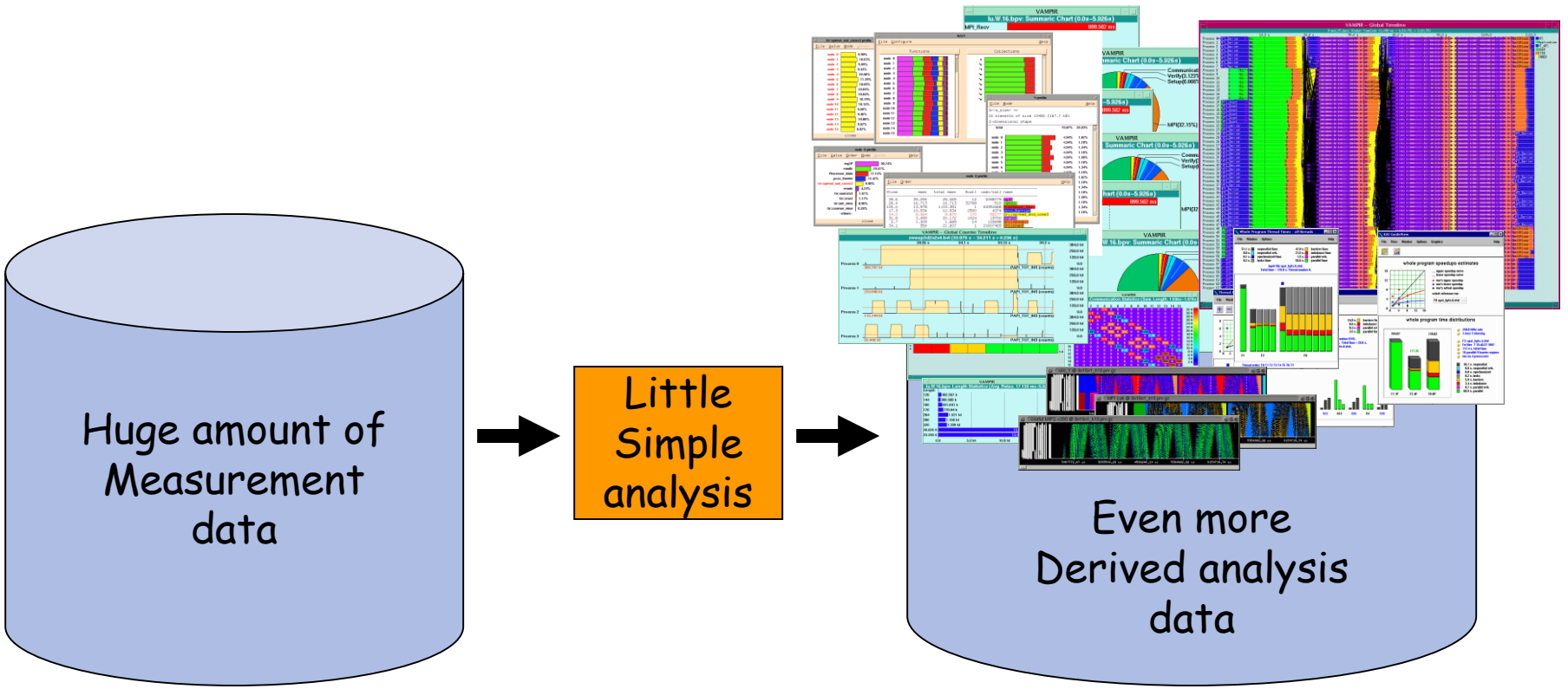
scalasca

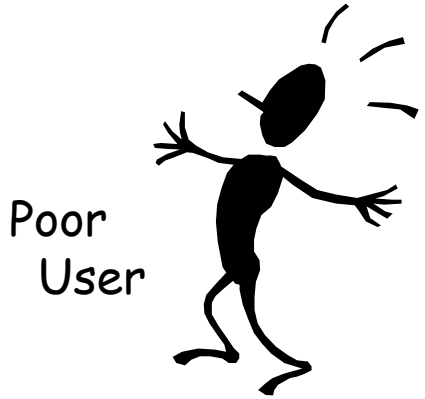
- Parallel analyzer SCOUT
- http://www.scalasca.org

*HINT: https://web.archive.org/

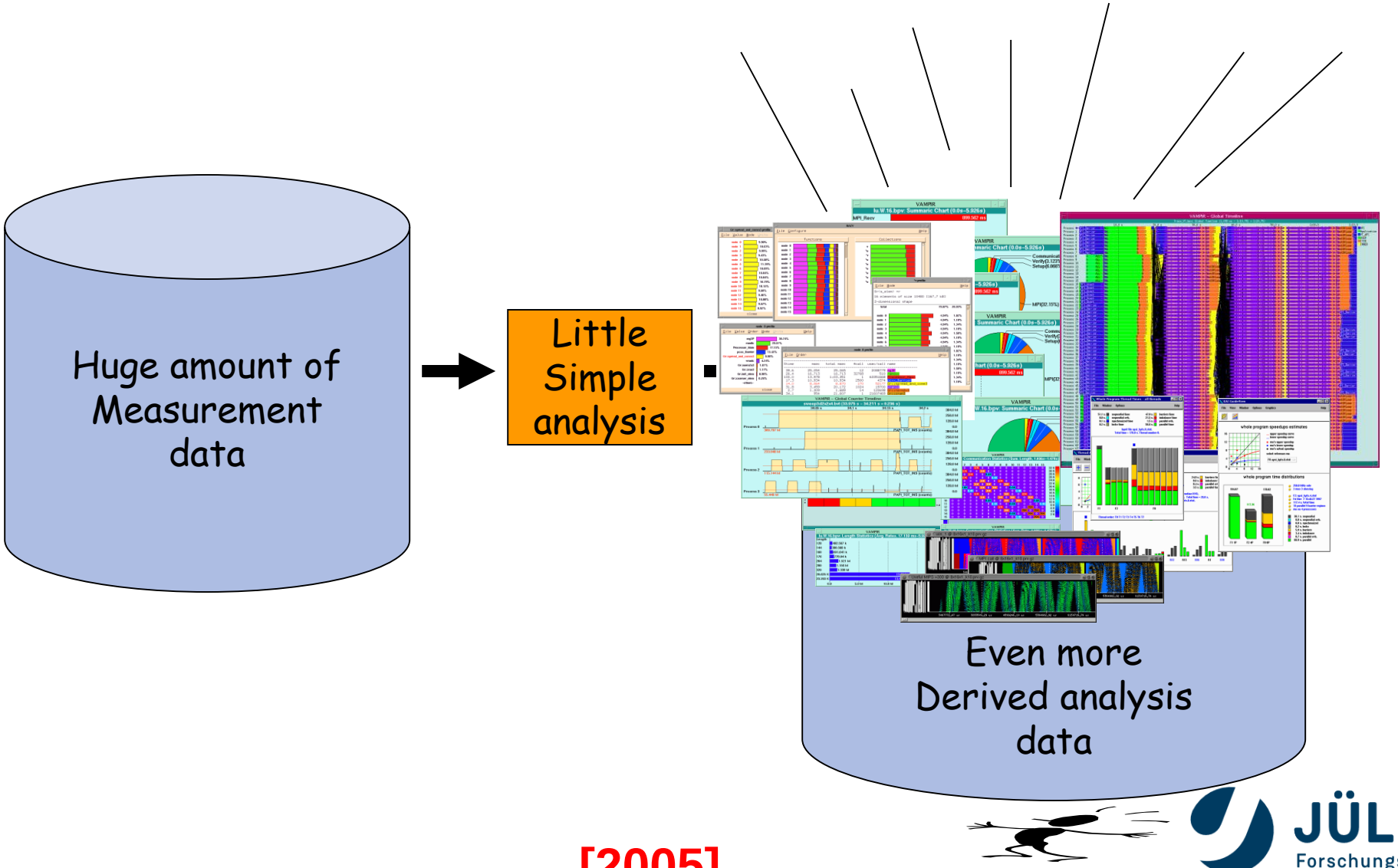
JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

TRADITIONAL PERFORMANCE TOOLS

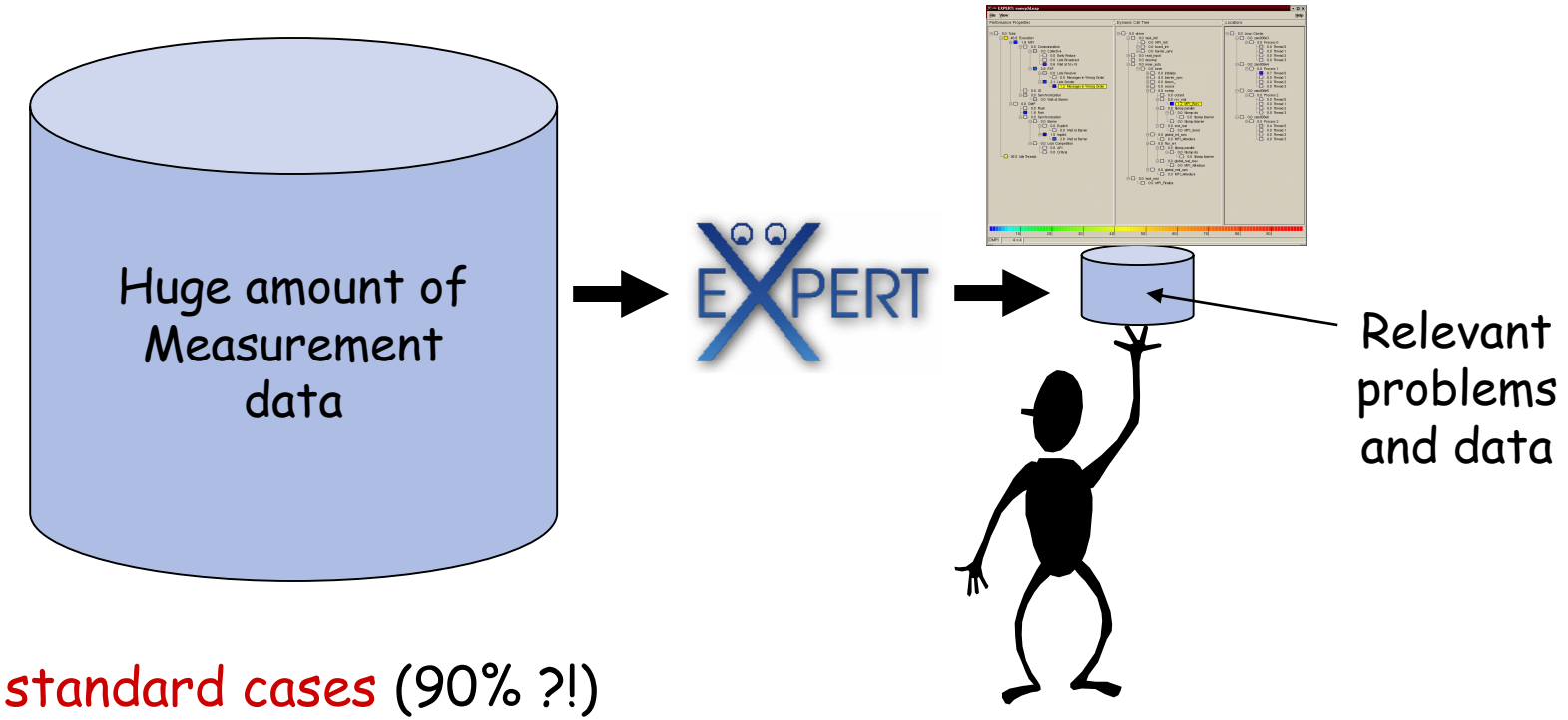




TRADITIONAL PERFORMANCE TOOLS

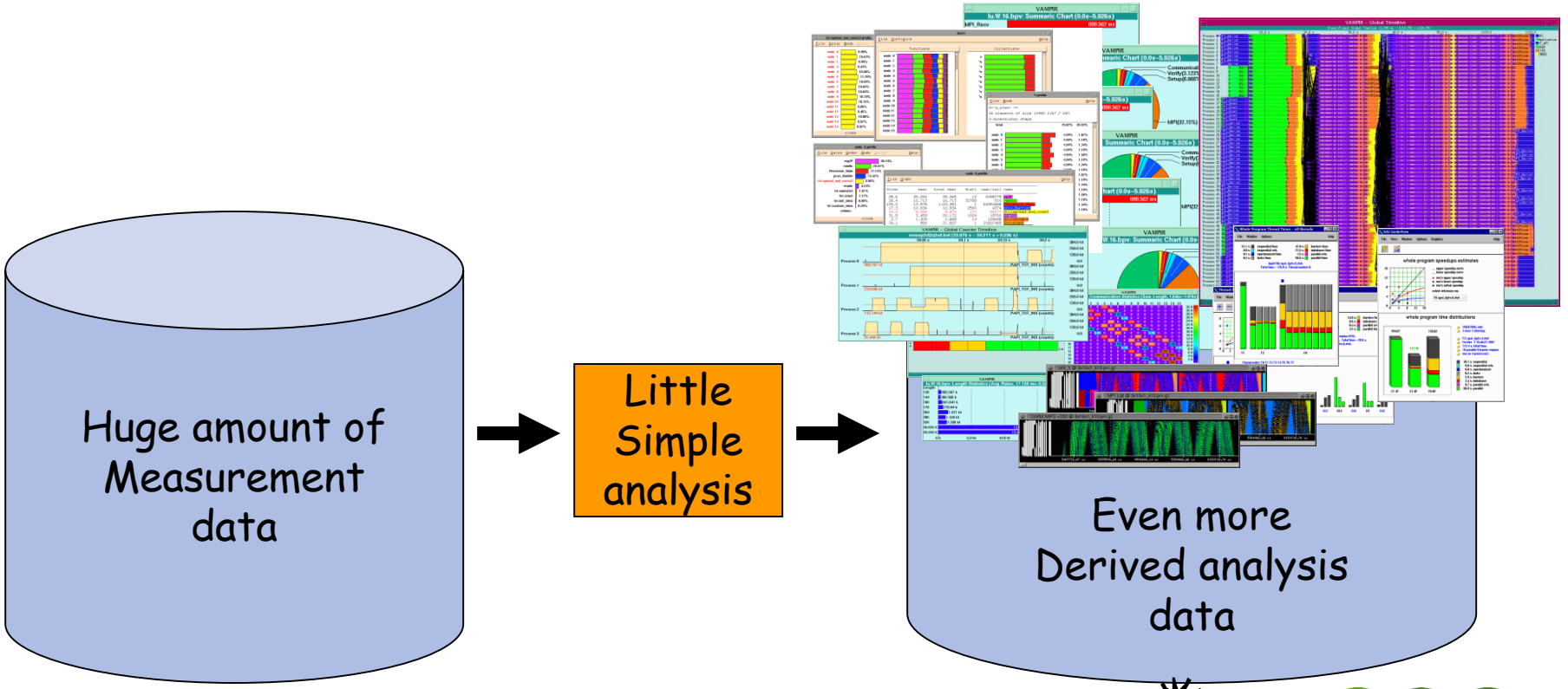


SOLUTION PART 1: AUTOMATIC TOOL



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

SOLUTION PART 2: EXPERT TOOLS + EXPERT

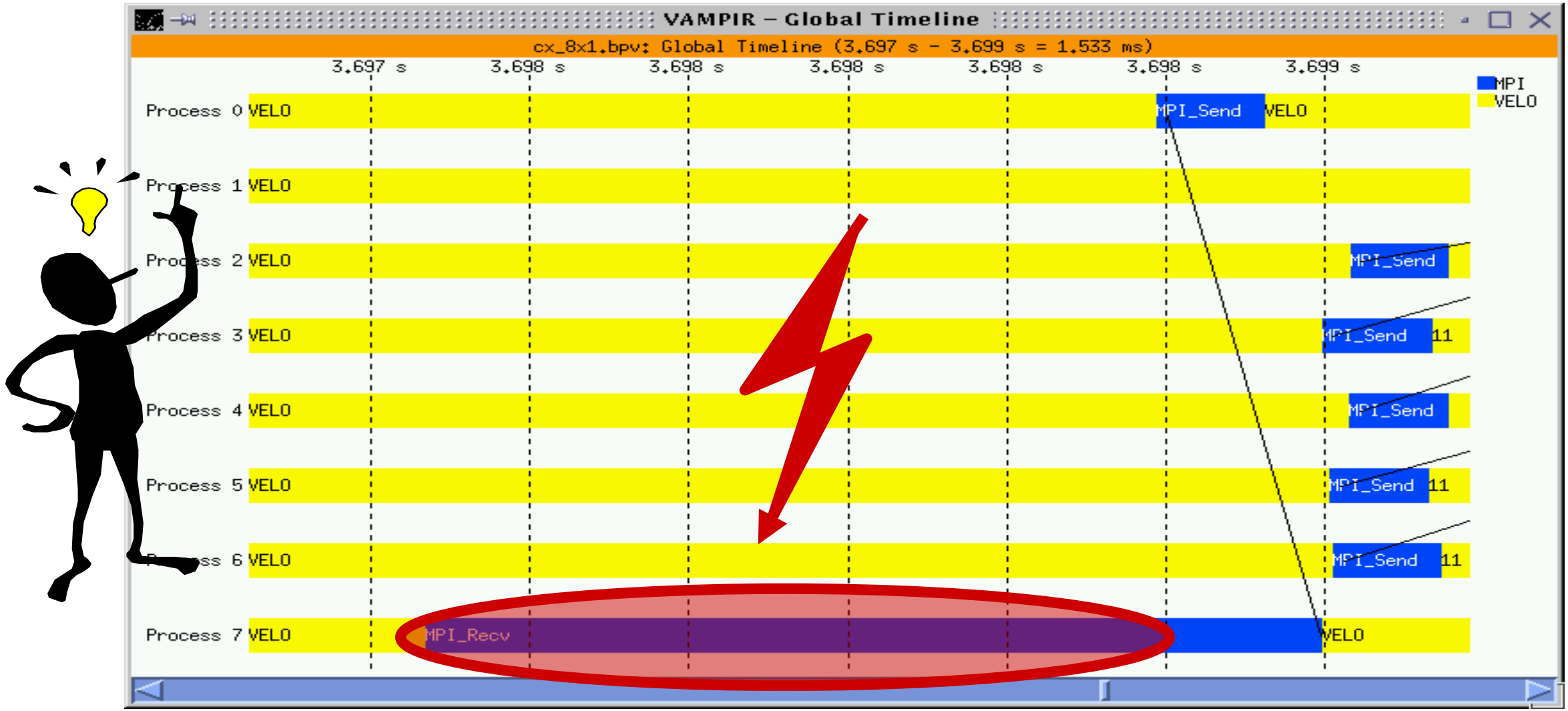


- For non-standard / tricky cases (10%)

⇨ More productivity for performance analysis process!

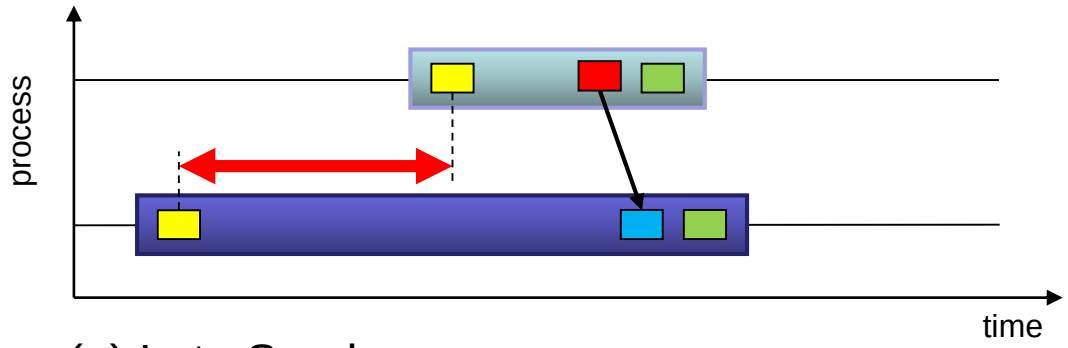
[2005]

EXAMPLE AUTOMATIC ANALYSIS: LATE SENDER

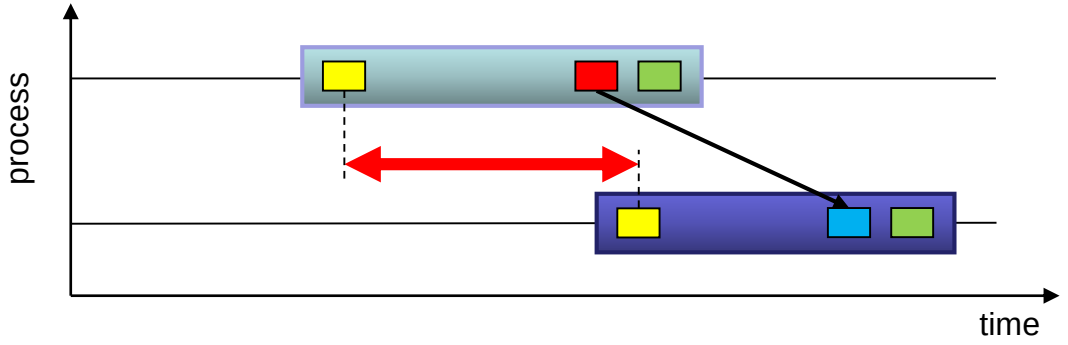


JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

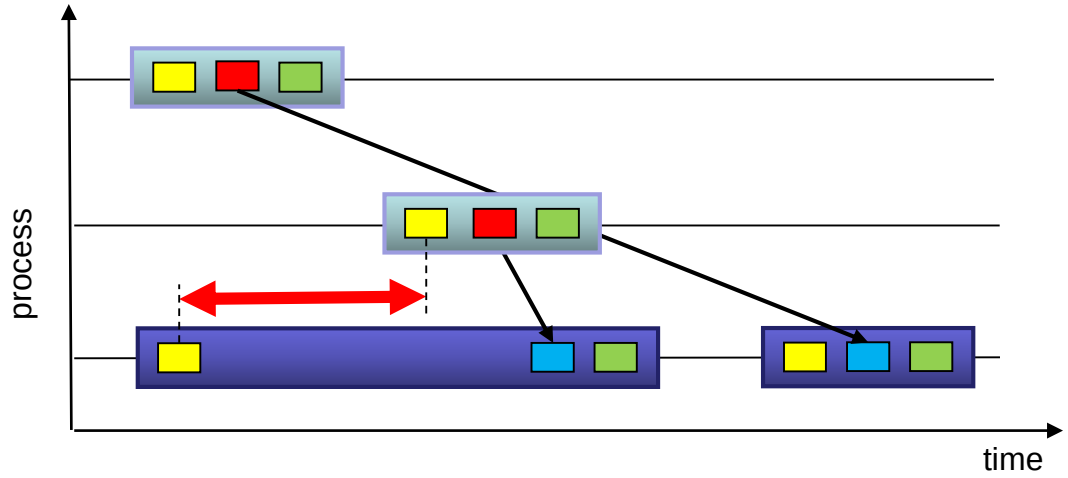
EXAMPLE MPI WAIT STATES



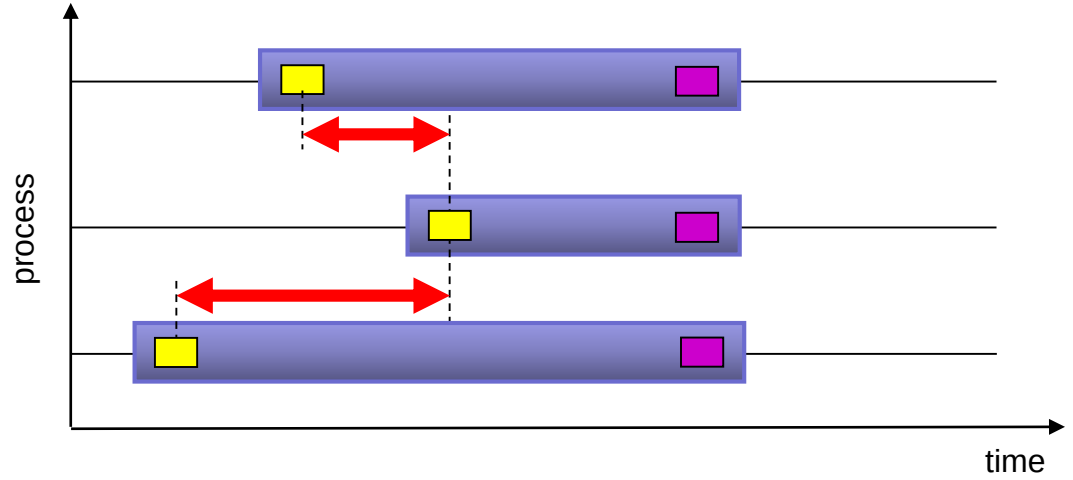
(a) Late Sender



(b) Late Receiver



(c) Late Sender / Wrong Order

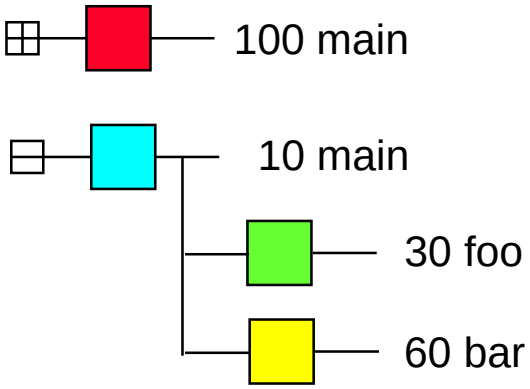


(d) Wait at N x N

ENTER EXIT SEND RECV COLLEXIT

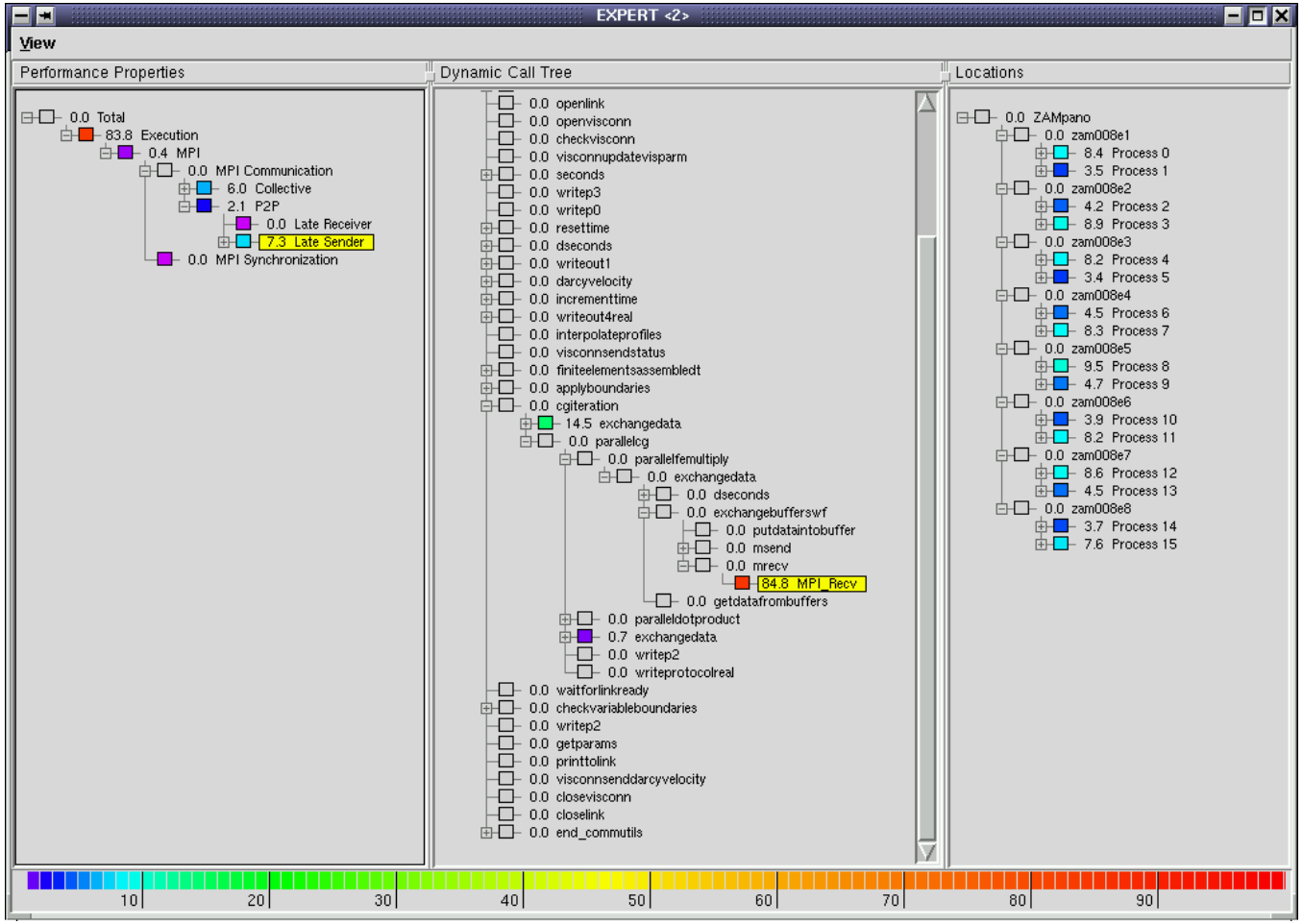
PRESENTATION OF PERFORMANCE BEHAVIOR [2001]

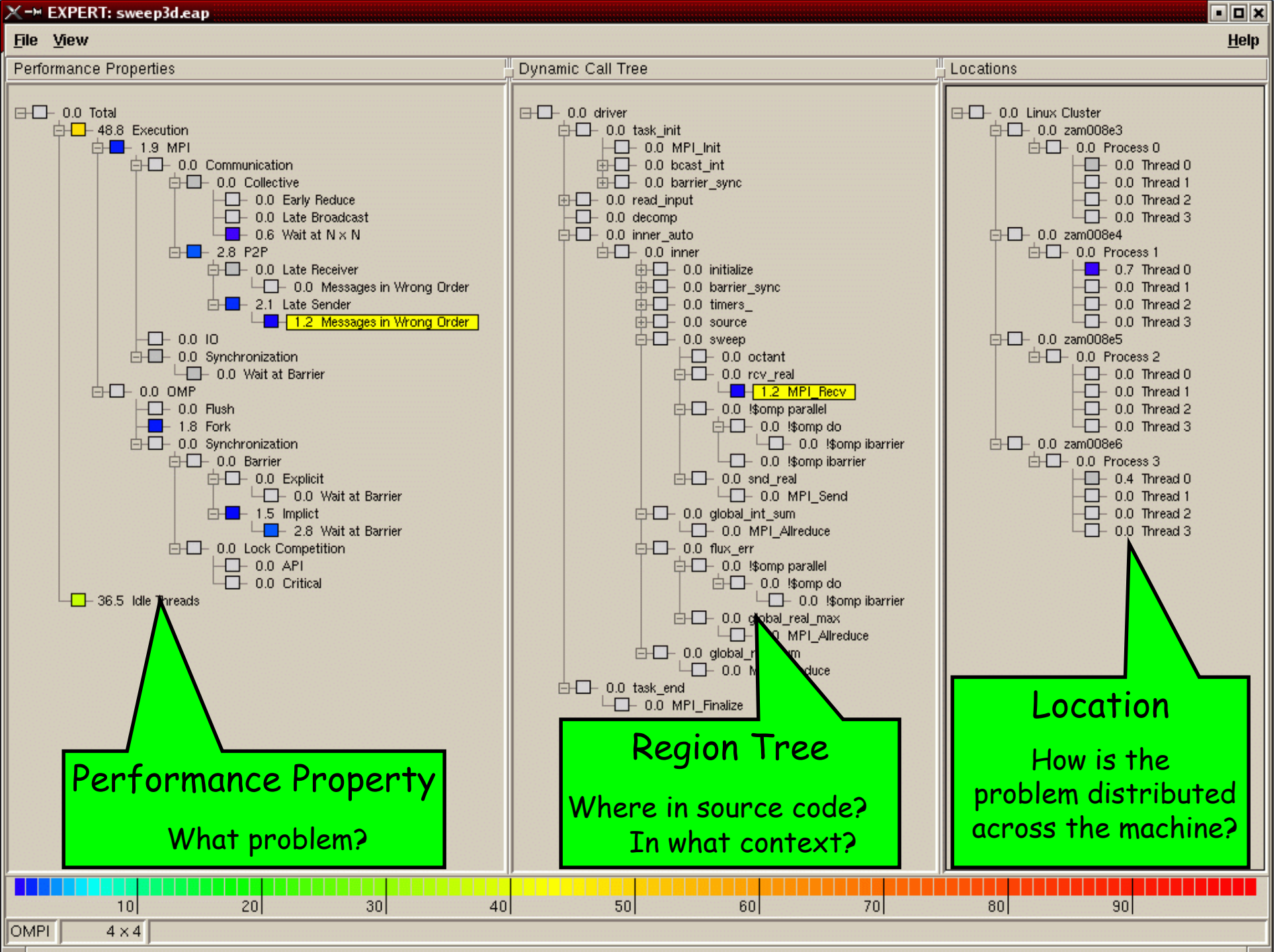
- **Performance behavior**
  - **3 dimensional matrix**
  - **Hierarchical dimensions**
- **Weighted tree**
  - **Tree browser**
  - **Each node has weight**
    - **Percentage of CPU allocation time**
    - **E.g. time spent in subtree of call tree**
  - **Displayed weight depends on state of node**
    - **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



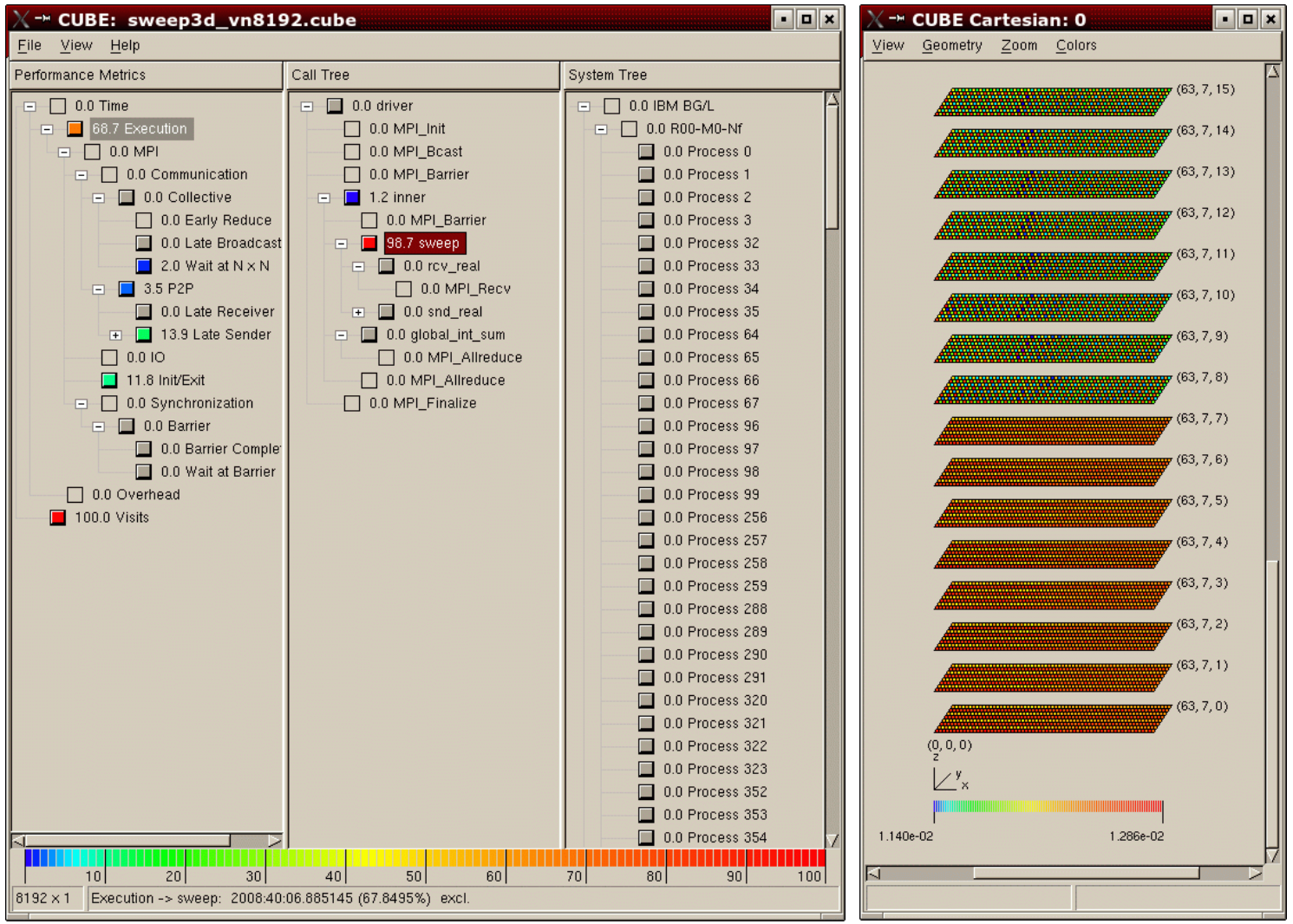


[2003]

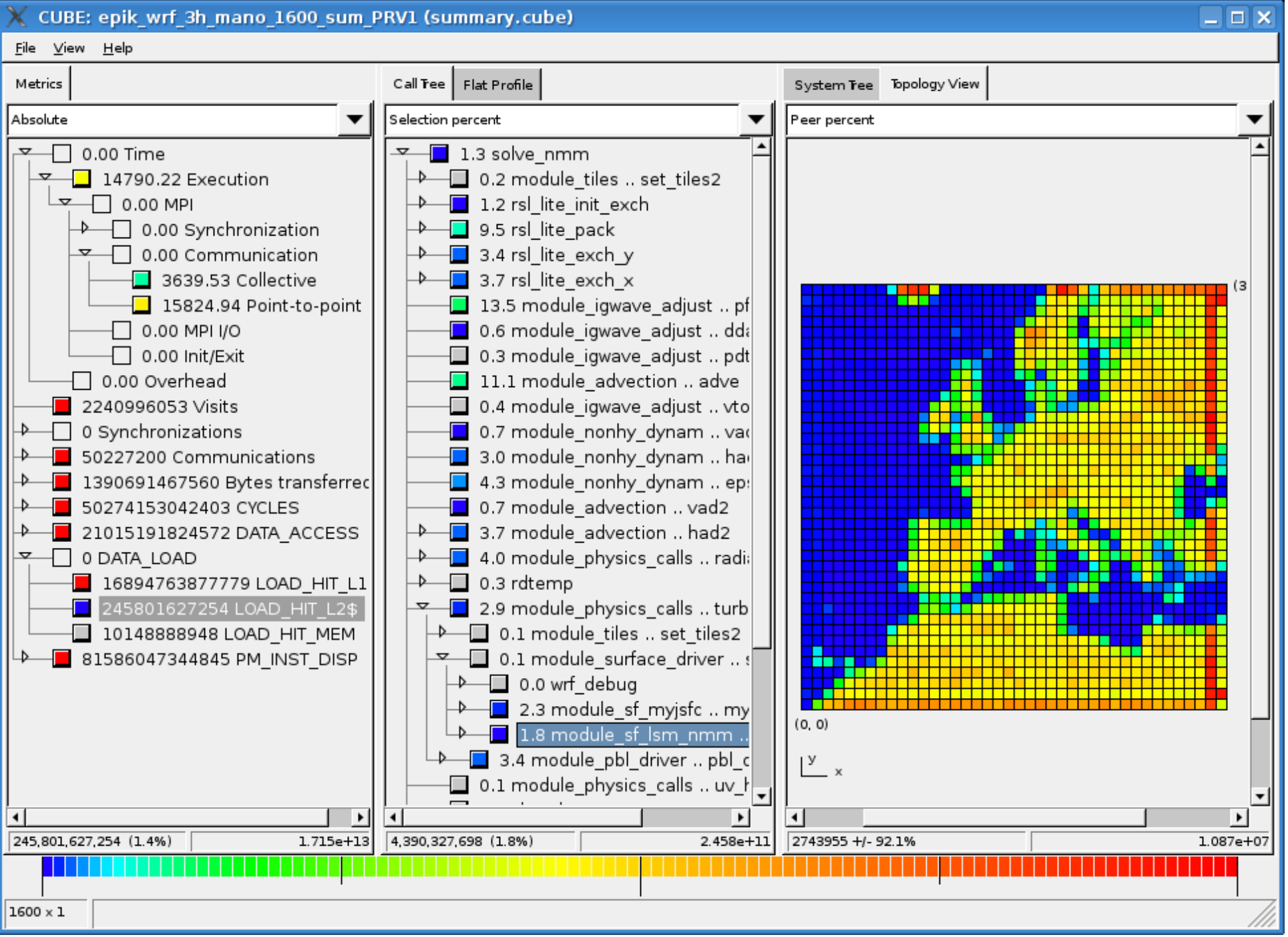
JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

EXAMPLE: SWEEP3D ON 8192 BG/L PES

[2007]



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



[2009]

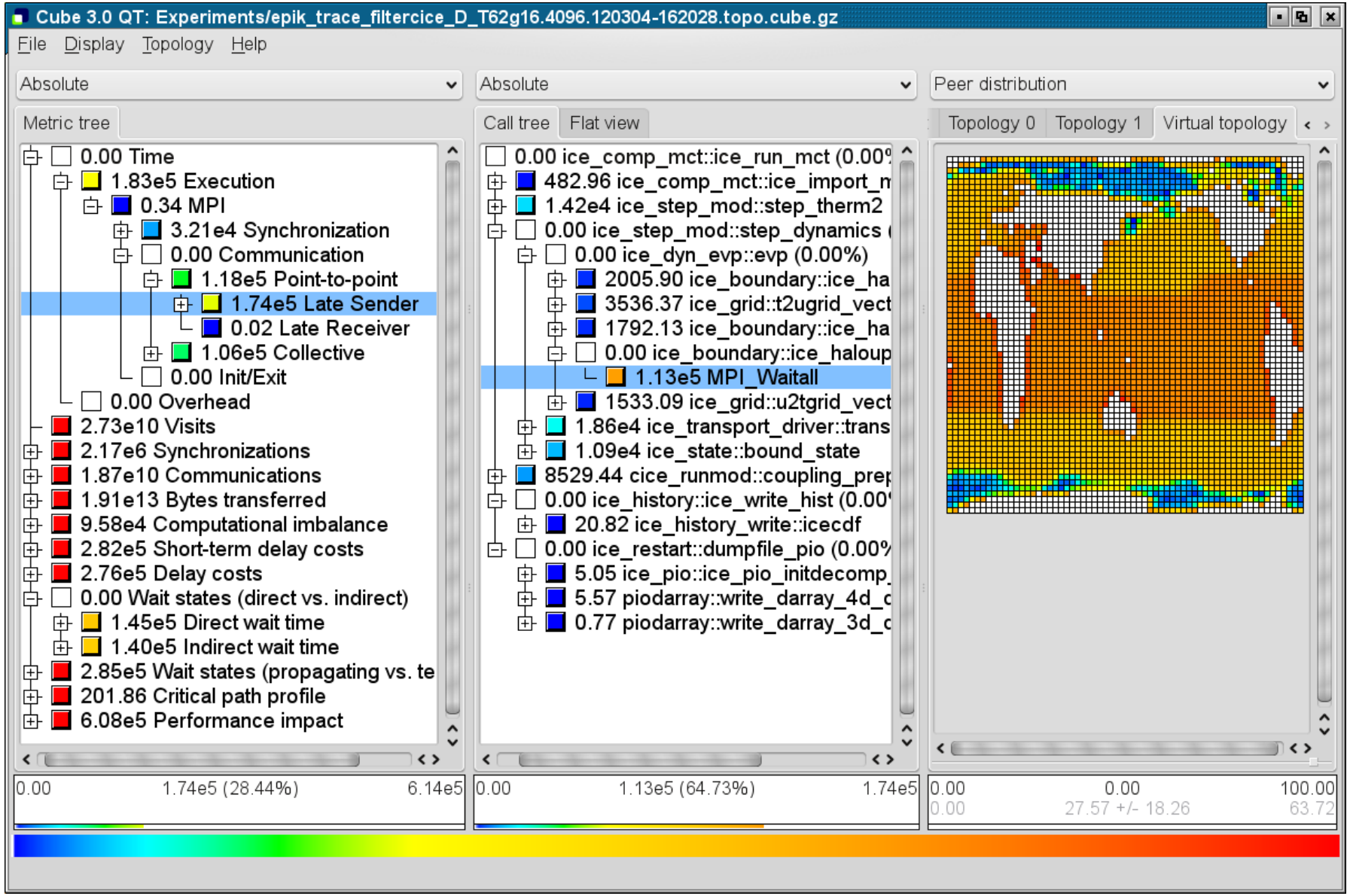
- Application topologies

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

SCALASCA EXAMPLE: CESM SEA ICE MODULE

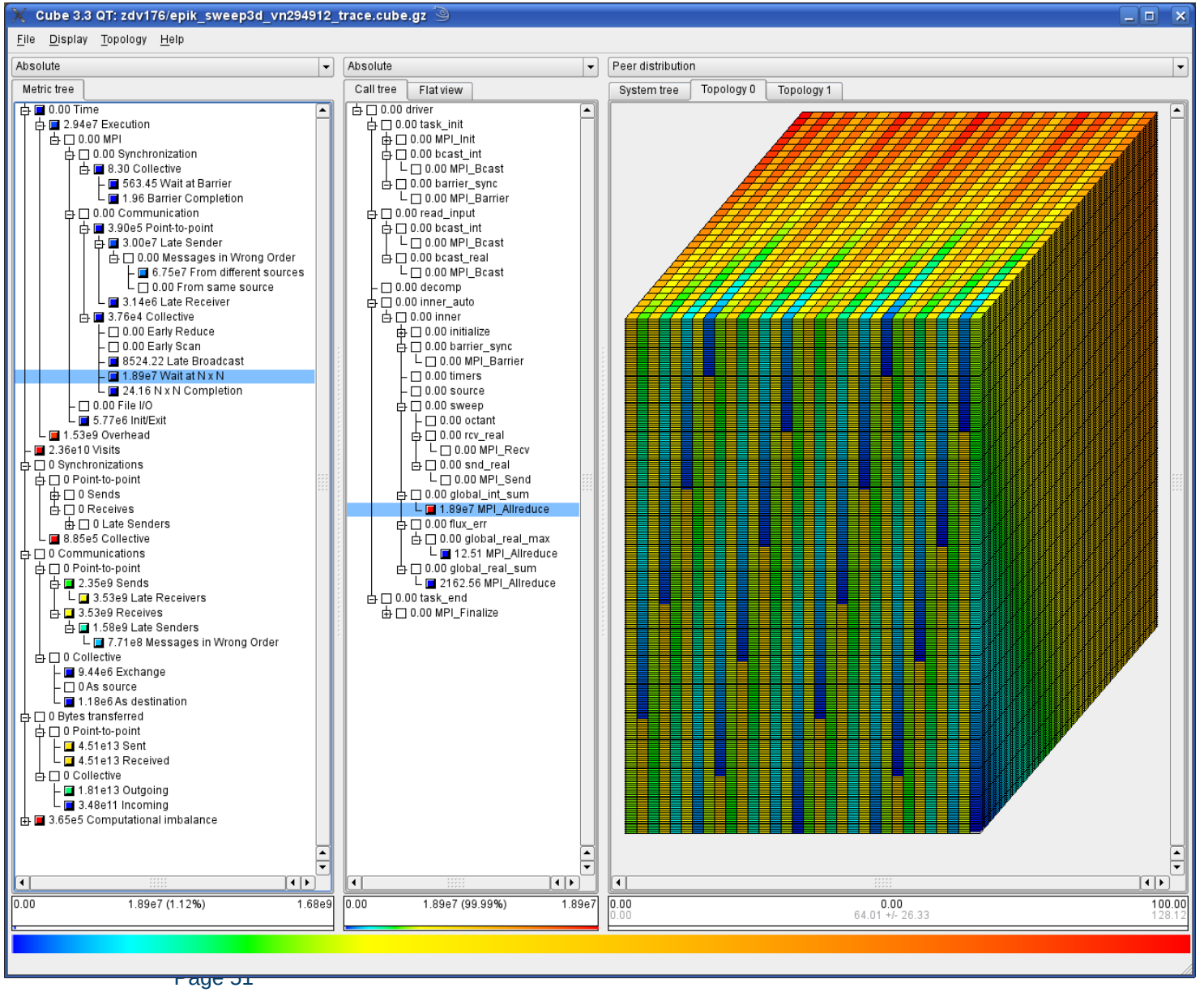
Late Sender Analysis + Application Topology

- Shows distribution of imbalance over topology
- MPI topologies are automatically captured



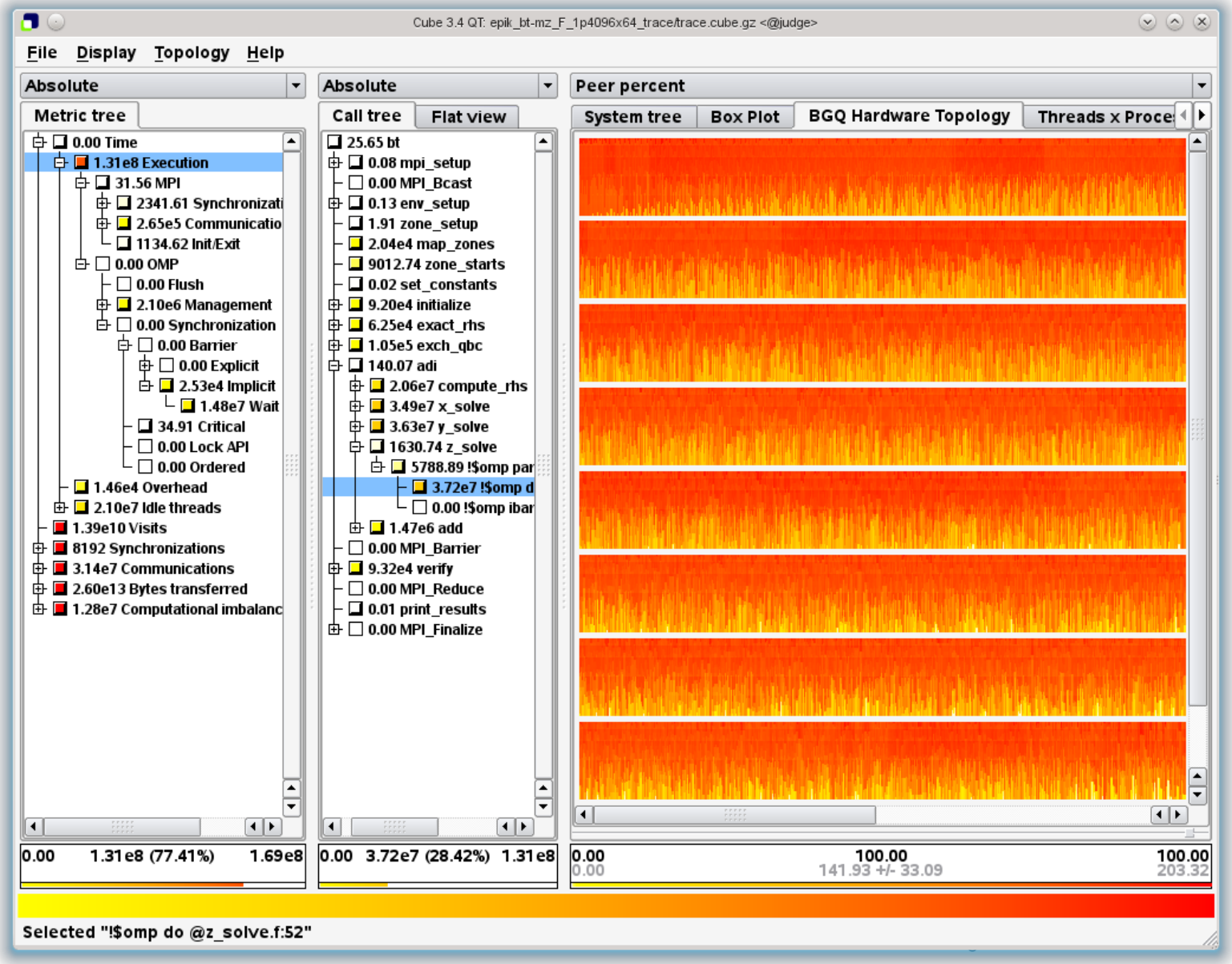
SCALASCA TRACE ANALYSIS SWEEP3D@294,912 BGP [2010]

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



B. J. N. Wylie, M. Geimer, B. Mohr, D. Böhme, Z.Szebenyi, F. Wolf: Large-scale performance analysis of Sweep3D with the Scalasca toolset. Parallel Processing Letters, 20(4):397-414, 2010.

SCALASCA TRACE ANALYSIS BT-MZ@1,048,704 BGQ [2013]



JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

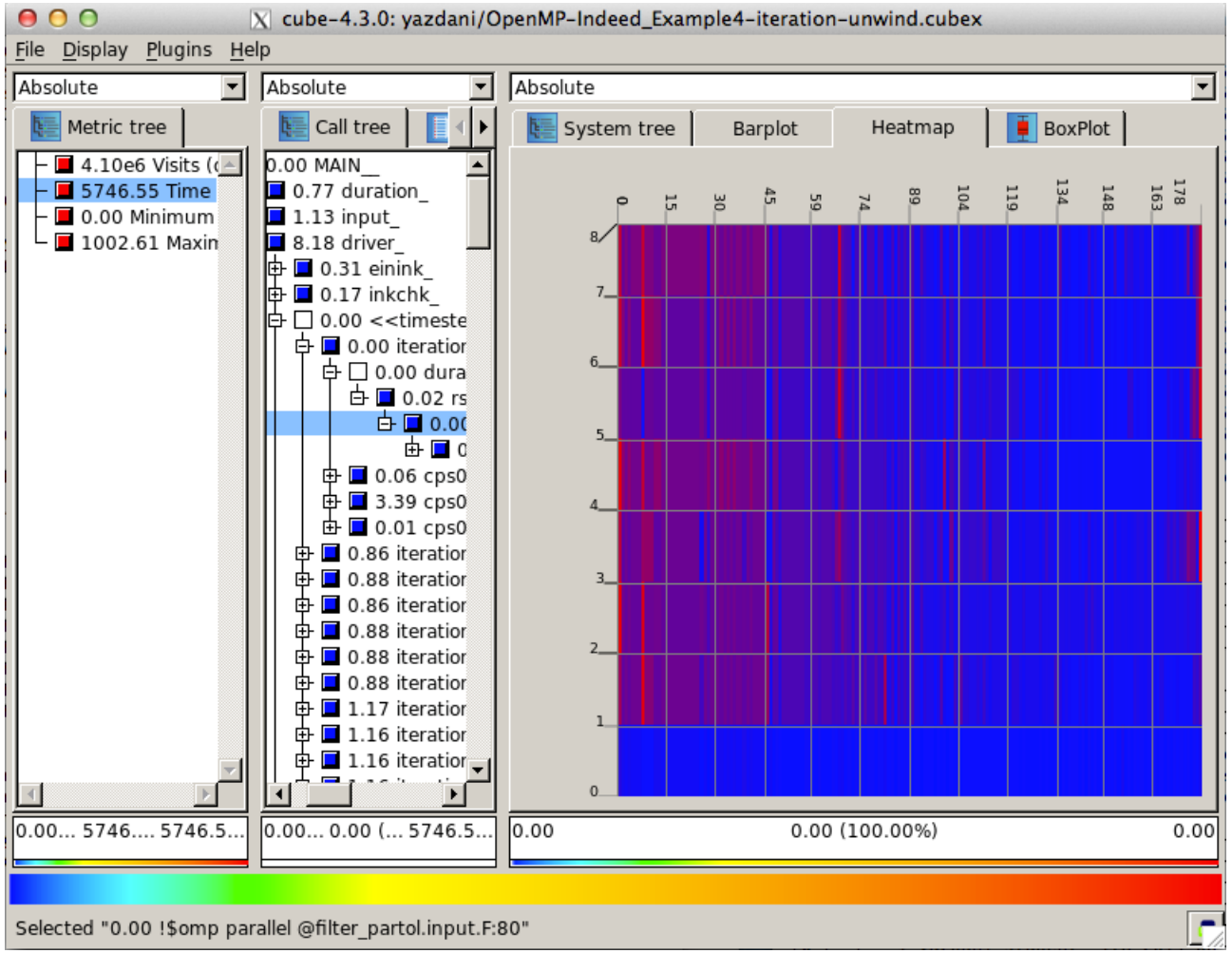
CUBE VIZ PLUGINS: PHASE HEATMAP

[2015]

- **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

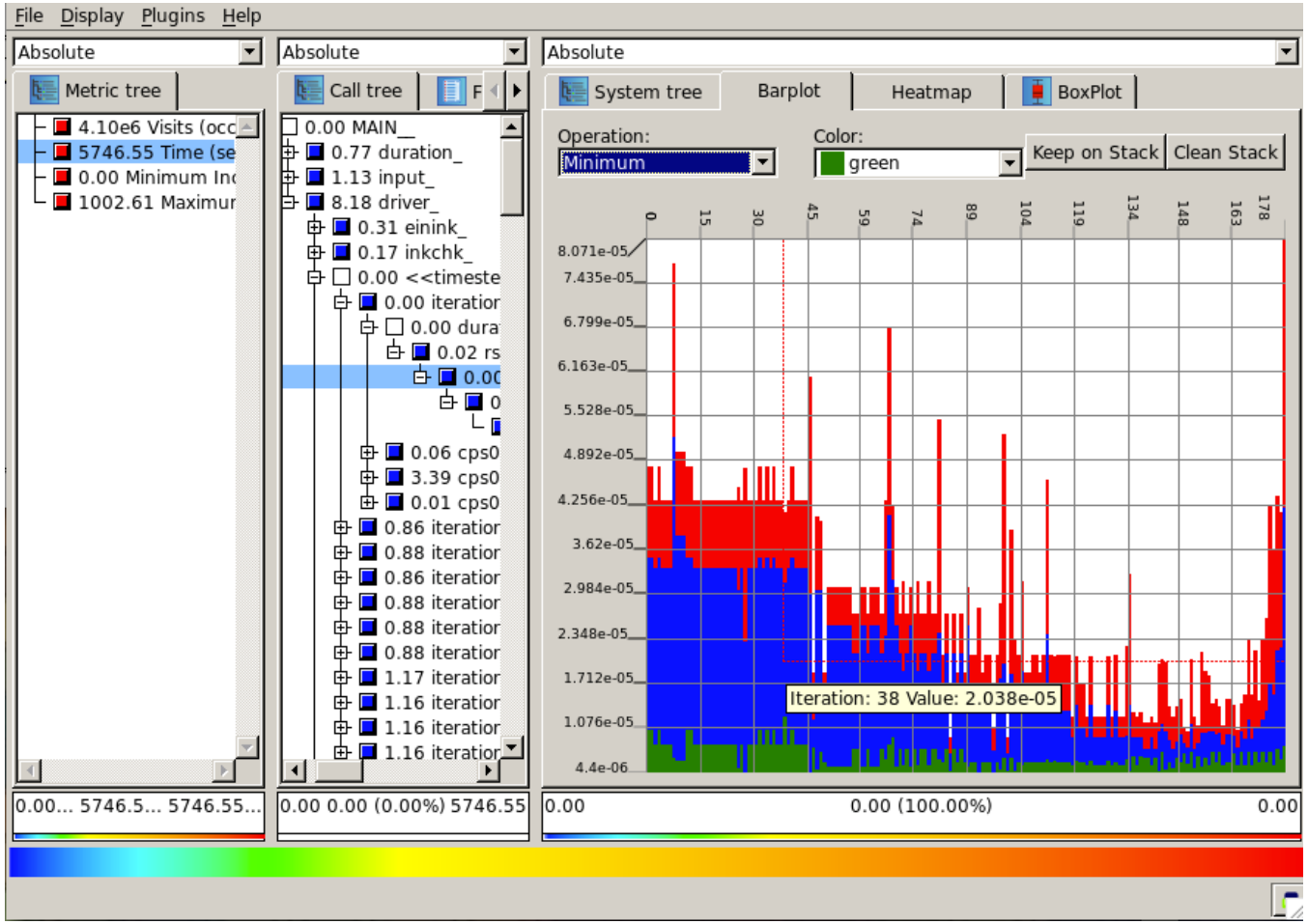


CUBE VIZ PLUGINS: PHASE BARPLOT [2015]

- **Phase profiling**

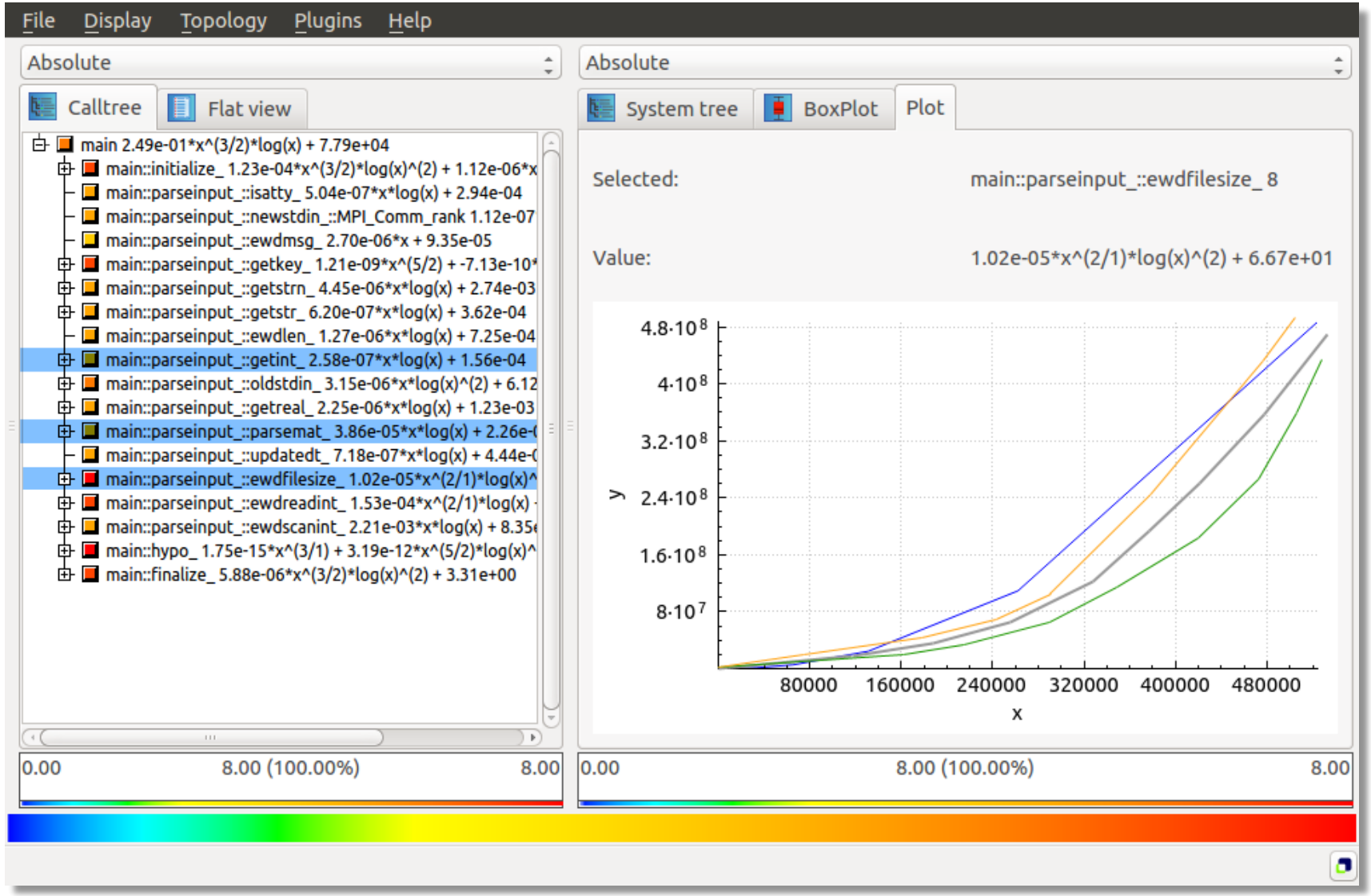
- Collects data for each instance of phases marked in program instead of aggregating it

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



CATWALK: MODELING RESULT VISUALIZATION

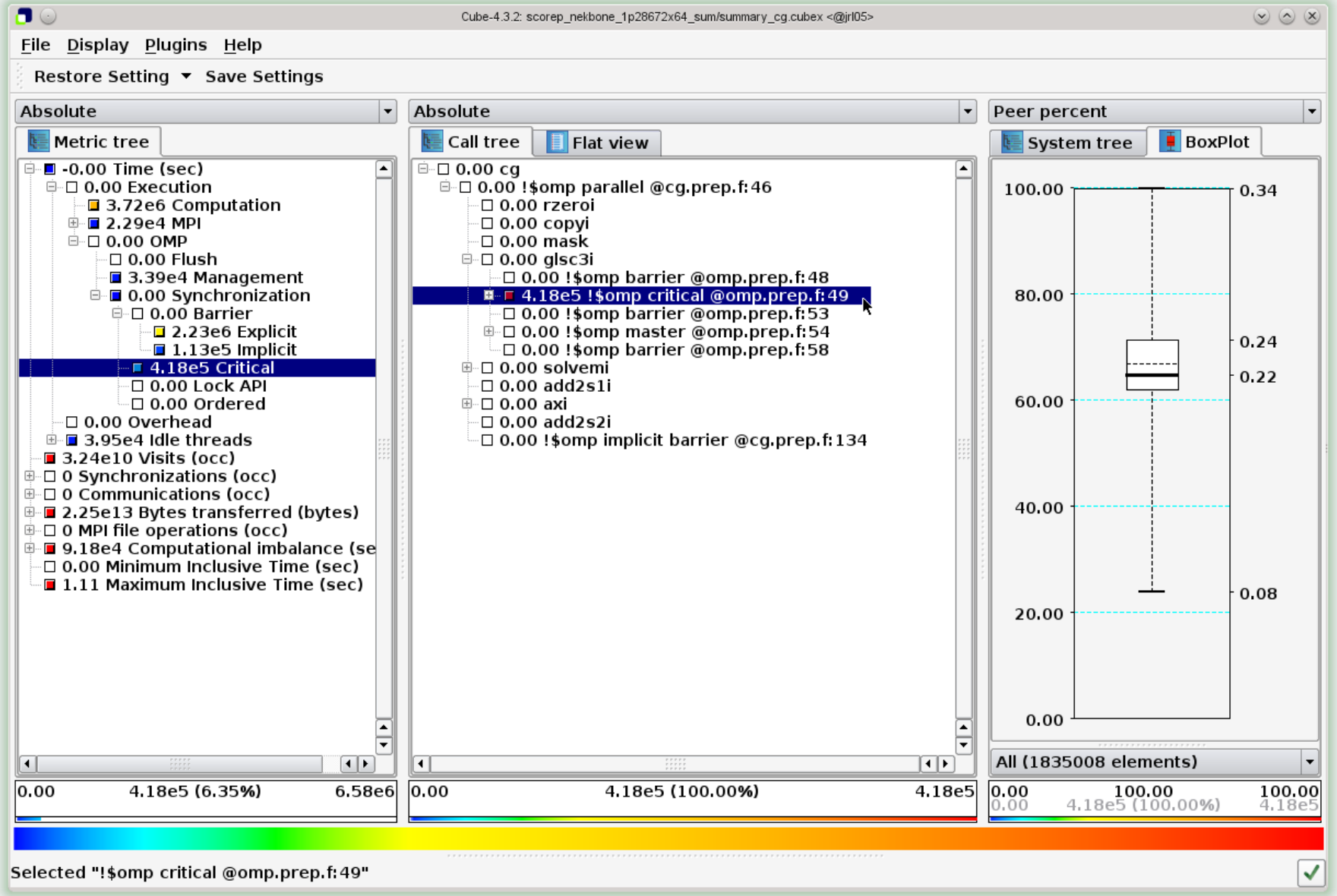
[2015]



SCALASCA: 1,835,008 THREADS TEST CASE [2016]

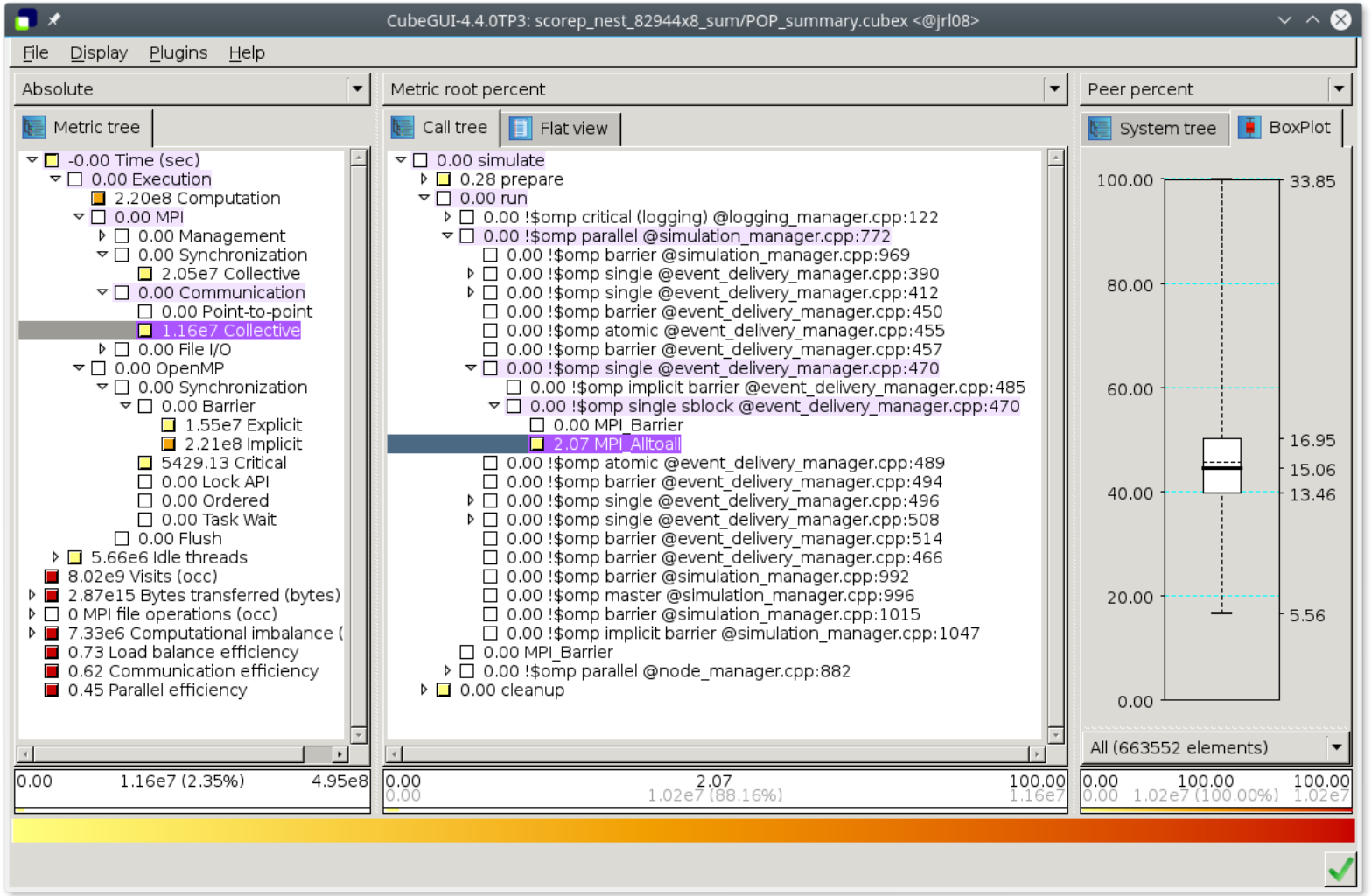
- Nekbone
- CORAL benchmark
- JuQueen experiment
- 28,672 x 64 = 1,835,008 threads

- Load imbalance at OpenMP critical
- section



SCALASCA: USER ANALYSIS OF NEST ON K COMPUTER

- Jülich nest:: neural network simulator code
- Measurement of **full system K computer run**
  - 82,944 nodes
  - 663,552 threads
- Performance analyst
  - Itaru Kitayama (RIKEN)
- Analysis of MPI and OpenMP communication and synchronization at large scale



JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

You KNOW YOU made it …

… WHEN LARGE COMPANIES “COPY” YOUR STUFF

[2014]

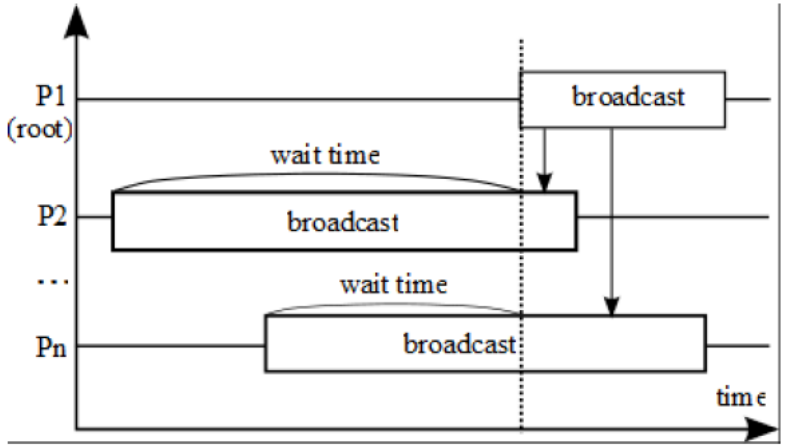
Introducing the Intel® Trace Analyzer and Collector Performance Assistant

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



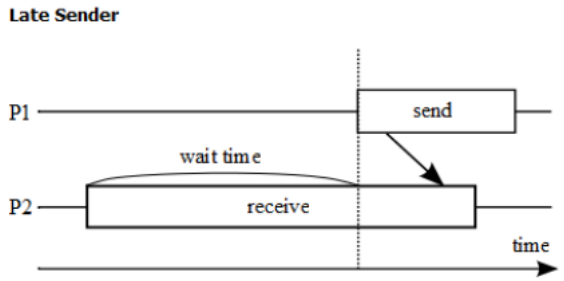
Source:
https://software.intel.com/en-us/videos/quickly-discover-performance-issues-with-the-intel-trace-analyzer-and-collector-90-beta

[2014]

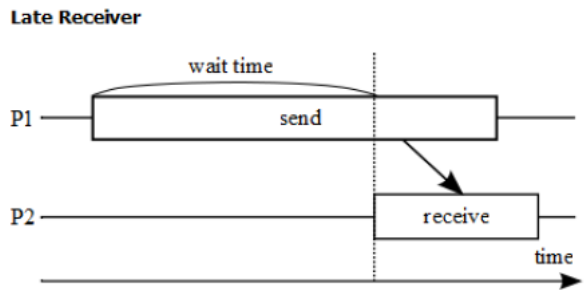
Which Performance Issues are automatically identified?

Point-to-point exchange problems:

- Late Sender

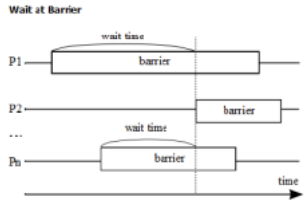


- Late Receiver

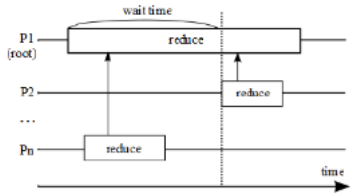


Problems with global collective operation performance:

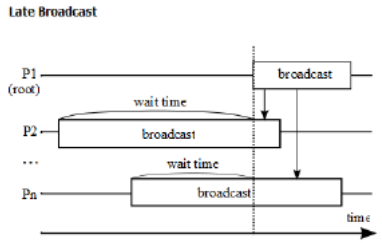
- Wait at Barrier



- Early Reduce



- Late Broadcast



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

ASSESSMENT

- Score-P / Scalasca installed on many HPC sites world-wide
- Used in daily work by performance analysts (e.g. POP CoE)
- User interface consistent for 23 years (but many enhancements)
- Support by vendors: Intel, AMD, (Siemens)

- Lots of **work** behind the scenes
  - Scalasca1 (Epilog) ⇨ scalasca2 (Score-P)
  - **Constant** bug fixing
  - Constant scaling improvements
  - Lately: MPI 3+4, OpenMP 3+4+5, OMPT, F2008 MPI interface, Pthreads, …
  - **GPU support**: OpenMP target, OpenACC, CUDA HIP/ROCm, Kokkos, …

FUTURE WORK

- **Memory and vectorization** performance analysis
  - Hard to capture performance data
    - Only possible if suitable hardware counters are provided
    - VERY processor specific ⇨ hard for open-source portable tools
- Trend towards **task-based / asynchronous programming models**
  - Very dynamic execution might be non reproducible ⇨ off-line tools fail
  - Hard to get the “big picture” ⇨ good high-level metrics still missing here
  - **3-pane Cube display shows its limitations here … ?!**
- Trend towards more **modern programming languages (Python, C++)**
  - How to automatically instrument template-based frameworks and programming styles?
  - How to present the data on Python level (and not on the interpreter low-level)?
  - Performance assessment of **data analytics codes**

QUESTIONS ?