Performance Tools and PAPI

Philip J. Mucci

Center for Parallel Computers (PDC),
Royal Institute of Technology (KTH), Stockholm, Sweden
Innovative Computing Laboratory, UT Knoxville

mucci at cs.utk.edu

http://www.cs.utk.edu/~mucci

October 3rd, 2005

PAPI

Parallelldatorcentrum

- **P**erformance **A**pplication **P**rogramming **I**nterface
- The purpose of PAPI is to implement a standardized portable and efficient API to access the hardware performance monitor counters found on most modern microprocessors.
- The goal of PAPI is to facilitate the optimization of parallel and serial code performance by encouraging the development of cross-platform optimization tools.
  - PapiEx
  - PerfSuite
  - HPCToolkit
  - TAU
  - KOJAK
  - non-PAPI: mpiP

PAPI Features

- Preset, Native and Derived Performance Metrics
- Full enumeration of platform-specific metrics
- Multiplexed Event Measurement
- Callbacks on overflow
- Full SVR4 Profiling, plus extensions (32, 64 bit, etc..)
- Bindings for C, Fortran, Matlab, and Java
- Overflow and profiling on multiple events simultaneously
- Complete memory hierarchy information
- Complete executable and shared information
- Thread safe API
- Efficient thread local storage and locking routines
- 2 API's, high (app. eng.) and low level (tool dev.)

PapiEx: PAPI Execute

KTH VETENSKAP OCH KONST
Parallelldatorcentrum

- A simple tool that generates performance measurements for the entire run of a code. No recompilation.
- Monitors all subprocesses/threads.
- Monitor detailed memory usage.
- Automatically detects multi-threaded executables.
- Supports counter multiplexing with -m.
- Provides hooks for simple instrumentation of user source code if desired.

PerfSuite

- Three command-line tools:
  - psrun: obtain performance data
  - psprocess: present/transform data
  - psinv: machine inventory utility
- Command line tool similar to IRIX's perfex command.
- Does aggregate counting of the entire run. Also provides statistical profiling.
- Output is XML or Plain Text.

HPCToolkit from Rice U.

- Use event-based sampling and statistical profiling to profile unmodified applications: hpcrun
- Interpret program counter histograms: hpcprof
- Correlate source code, structure and performance metrics: hpcview/hpcquick
- Explore and analyze performance databases: hpcviewer

TAU Performance System

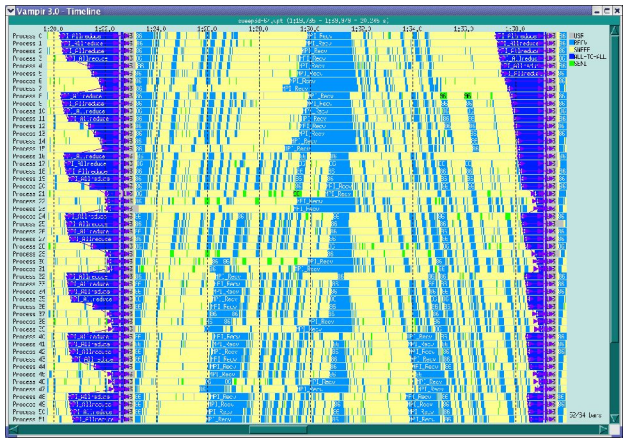
- Tuning and Analysis Utilities (11+ year project effort)
- *Performance system framework* for scalable parallel and distributed high-performance computing
- Targets a general complex system computation model
  - nodes / contexts / threads
  - Multi-level: system / software / parallelism
  - Measurement and analysis abstraction
- *Integrated toolkit* for performance instrumentation, measurement, analysis, and visualization
  - Portable performance profiling and tracing facility
  - Open software approach with technology integration
- University of Oregon , Forschungszentrum Jülich, LANL

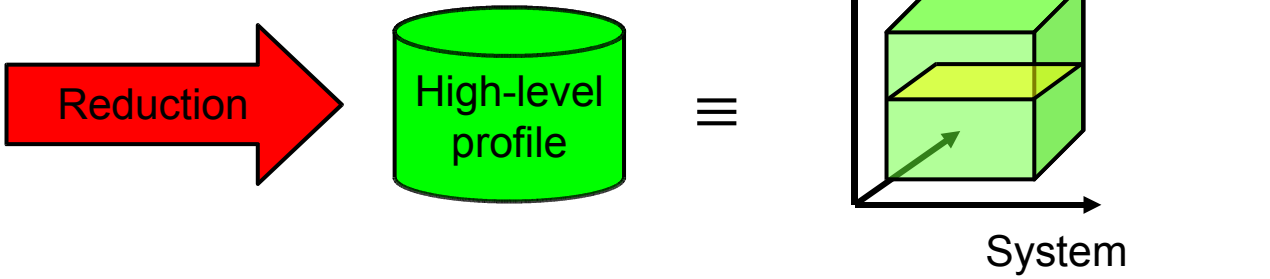
TAU Performance System

- Multi-level performance instrumentation
  - Multi-language automatic source instrumentation
- Flexible and configurable performance measurement
- Widely-ported parallel performance profiling system
  - Computer system architectures and operating systems
  - Different programming languages and compilers
- Support for multiple parallel programming paradigms
  - Multi-threading, message passing, mixed-mode, hybrid
- Support for performance mapping
- Support for object-oriented and generic programming
- Integration in complex software systems and applications

KOJAK

- Joint open-source project between
  - Forschungszentrum Jülich, Germany
  - University of Tennessee, USA
- Performs automatic performance analysis of parallel applications (MPI and/or OpenMP)
- Uses pattern recognition to transform event traces into information about performance bottlenecks relevant to developers





MpiP Overview

- Scalable, light-weight MPI profiling library
  - Generates detailed text summary of MPI behavior
    - Time spent at each MPI function callsite
    - Bytes sent by each MPI function callsite (where applicable)
    - MPI I/O statistics
    - Configurable traceback depth for function callsites
  - Controllable from program using MPI_Pcontrol
    - Allows you to profile just one code module or cycle
    - Allows mpiP profile dumps mid-run
  - Requires only a relink with mpiP libraries

- mpiPview: Qt interface to browse the data

Parallelldatorcentrum

PAPIEX

papiex Usage

Parallelldatorcentrum

```
Usage: papiex [-lihVmukord] [-L event] [-f[prefix]] [-F file] [-e
   event] ... -- <cmd> <cmd options>

 -l              List the available events.

 -L event        List information about specific event.

 -i              Print information about the host machine.

 -h              Print this message.

 -V              Print version information.

 -m              Enable multiplexing of hardware counters.

 -u              Monitor user mode events. (default)

 -k              Monitor kernel mode events.

 -o              Monitor transient mode events.

 -r              Report getrusage() information.

 -x              Report memory information.

 -d              Enable debugging output.

 -f[prefix]      Output to <prefix><cmd>.papiex.<host>.<pid>.<tid>.

 -e event        Monitor this hardware event.
```

PapiEx Output

Parallelldatorcentrum

```
PapiEx Version:         0.99rc2
Executable:             /afs/pdc.kth.se/home/m/mucci/summer/a.out
Processor:              Itanium 2
Clockrate:              900.000000
Parent Process ID:      8632
Process ID:             8633
Hostname:               h05n05.pdc.kth.se
Options:                MEMORY
Start:                  Wed Aug 24 14:34:18 2005
Finish:                 Wed Aug 24 14:34:19 2005
Domain:                 User
Real usecs:             1077497
Real cycles:            969742309
Proc usecs:             970144
Proc cycles:            873129600
PAPI_TOT_CYC:           850136123
PAPI_FP_OPS:            40001767
Mem Size:               4064
Mem Resident:           2000
Mem Shared:             1504
Mem Text:               16
Mem Library:            2992
Mem Heap:               576
Mem Locked:             0
Mem Stack:              32

Event descriptions:
Event: PAPI_TOT_CYC
        Derived: No
        Short Description: Total cycles
        Long Description: Total cycles
        Developer's Notes:
Event: PAPI_FP_OPS
        Derived: No
        Short Description: FP operations
        Long Description: Floating point operations
        Developer's Notes:
```

PapiEx Caliper Fortran Example

Parallelldatorcentrum

```
#include "papiex.h"

      program zero

      real a, b, c;
      a = 0.1
      b = 1.1
      c = 2.1

      PAPIEX_START_ARG(1,"write")

      print *, "Doing 10000000 iters. of a += b * c on doubles."

      PAPIEX_STOP_ARG(1)

      PAPIEX_START_ARG(2,"do loop")

      do i=1,100000000
         a = a + b * c
      end do

      PAPIEX_STOP_ARG(2)

      end
```

PapiEx Caliper C/C++ Example

Parallelldatorcentrum

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include "papiex.h"

volatile double a = 0.1, b = 1.1, c = 2.1;

int main(int argc, char **argv)
{
  int i;

  PAPIEX_START_ARG(1,"printf");

  printf("Doing 100000000 iters. of a += b * c on doubles.\n");

  PAPIEX_STOP_ARG(1);

  PAPIEX_START_ARG(2,"for loop");

  for (i=0;i<100000000;i++)
    a += b * c;

  PAPIEX_STOP_ARG(2);

  exit(0);
}
```

PapiEx Caliper Example

Parallelldatorcentrum

```
bash-3.00$ papiex -e PAPI_L1_DCM ./a.out
 Doing 10000000 iters. of a += b * c on doubles.

[normal papiex output...]
PAPI_L1_DCM:             6864

Caliper 1: write
        Executions:              1
        Real usecs:              42
        Real cycles:             144432
        Proc usecs:              42
        Proc cycles:             144304
        PAPI_L1_DCM:             667
Caliper 2: do loop
        Executions:              1
        Real usecs:              769107
        Real cycles:             2608043669
        Proc usecs:              769019
        Proc cycles:             2607743748
        PAPI_L1_DCM:             4167

Event descriptions:
Event: PAPI_L1_DCM
        Derived: No
        Short Description: L1D cache misses
        Long Description: Level 1 data cache misses
        Developer's Notes:
```

Parallelldatorcentrum

PerfSuite

PSRUN Sample Output

Parallelldatorcentrum

```
PerfSuite Hardware Performance Summary Report
Version      : 1.0
Created      : Mon Dec 30 11:31:53 AM Central Standard Time 2002
Generator    : psprocess 0.5
XML Source   : /u/ncsa/anyuser/performance/psrun-ia64.xml
Execution Information
================================================================================
Date         : Sun Dec 15 21:01:20 2002
Host         : user01
Processor and System Information
================================================================================
Node CPUs    : 2
Vendor       : Intel
Family       : IPF
Model        : Itanium
CPU Revision : 6
Clock (MHz)  : 800.136
                                    Memory (MB)   : 2007.16
Pagesize (KB): 16
Cache Information
================================================================================
Cache levels : 3
--------------------------------
Level 1
Type         : data
Size (KB)    : 16
Linesize (B) : 32
Assoc        : 4
Type         : instruction
Size (KB)    : 16
Linesize (B) : 32
Assoc        : 4
--------------------------------
Level 2
Type         : unified
Size (KB)    : 96
Linesize (B) : 64
Assoc        : 6
--------------------------------
Level 3
Type         : unified
Size (KB)    : 4096
Linesize (B) : 64
Assoc        : 4
```

PSRUN Sample Output


```
Index Description                                                                    Counter Value
================================================================================================
    1 Conditional branch instructions mispredicted.....................              4831072449
    2 Conditional branch instructions correctly predicted..............             52023705122
    3 Conditional branch instructions taken............................             47366258159
    4 Floating point instructions......................................             86124489172
    5 Total cycles.....................................................            594547754568
    6 Instructions completed...........................................           1049339828741
    7 Level 1 data cache accesses......................................             30238866204
    8 Level 1 data cache hits..........................................               972479062
    9 Level 1 data cache misses........................................             29224377672
   10 Level 1 instruction cache reads..................................            221828591306
   11 Level 1 cache misses.............................................             29312740738
   12 Level 2 data cache accesses......................................            129470315862
   13 Level 2 data cache misses........................................             15569536443
   14 Level 2 data cache reads.........................................            110524791561
   15 Level 2 data cache writes........................................             18622708948
   16 Level 2 instruction cache reads..................................               566330907
   17 Level 2 store misses.............................................              1208372120
   18 Level 2 cache misses.............................................             15401180750
   19 Level 3 data cache accesses......................................              4650999018
   20 Level 3 data cache hits..........................................               186108211
   21 Level 3 data cache misses........................................              4451199079
   22 Level 3 data cache reads.........................................              4613582451
   23 Level 3 data cache writes........................................                38456570
   24 Level 3 instruction cache misses.................................                 3631385
   25 Level 3 instruction cache reads..................................                17631093
   26 Level 3 cache misses.............................................              4470968725
   27 Load instructions................................................            111438431677
   28 Load/store instructions completed................................            130391246662
   29 Cycles Stalled Waiting for memory accesses.......................            256484777623
   30 Store instructions...............................................             18840914540
   31 Cycles with no instruction issue.................................             61889609525
   32 Data translation lookaside buffer misses.........................                 2832692

Event Index
================================================================================
    1: PAPI_BR_MSP       2: PAPI_BR_PRC       3: PAPI_BR_TKN       4: PAPI_FP_INS
    5: PAPI_TOT_CYC      6: PAPI_TOT_INS      7: PAPI_L1_DCA       8: PAPI_L1_DCH
    9: PAPI_L1_DCM      10: PAPI_L1_ICR      11: PAPI_L1_TCM      12: PAPI_L2_DCA
   13: PAPI_L2_DCM      14: PAPI_L2_DCR      15: PAPI_L2_DCW      16: PAPI_L2_ICR
   17: PAPI_L2_STM      18: PAPI_L2_TCM      19: PAPI_L3_DCA      20: PAPI_L3_DCH
   21: PAPI_L3_DCM      22: PAPI_L3_DCR      23: PAPI_L3_DCW      24: PAPI_L3_ICM
   25: PAPI_L3_ICR      26: PAPI_L3_TCM      27: PAPI_LD_INS      28: PAPI_LST_INS
   29: PAPI_MEM_SCY     30: PAPI_SR_INS      31: PAPI_STL_ICY     32: PAPI_TLB_DM
```

PSRUN Sample Output

Parallelldatorcentrum

```
Statistics
==========================================================================================
Graduated instructions per cycle.....................................  1.765
Graduated floating point instructions per cycle......................  0.145
% graduated floating point instructions of all graduated instructions..  8.207
Graduated loads/stores per cycle.....................................  0.219
Graduated loads/stores per graduated floating point instruction.....  1.514
Mispredicted branches per correctly predicted branch.................  0.093
Level 1 data cache accesses per graduated instruction................  2.882
Graduated floating point instructions per level 1 data cache access....  2.848
Level 1 cache line reuse (data).....................................  3.462
Level 2 cache line reuse (data).....................................  0.877
Level 3 cache line reuse (data).....................................  2.498
Level 1 cache hit rate (data).......................................  0.776
Level 2 cache hit rate (data).......................................  0.467
Level 3 cache hit rate (data).......................................  0.714
Level 1 cache miss ratio (instruction)..............................  0.003
Level 1 cache miss ratio (data).....................................  0.966
Level 2 cache miss ratio (data).....................................  0.120
Level 3 cache miss ratio (data).....................................  0.957
Bandwidth used to level 1 cache (MB/s).............................  1262.361
Bandwidth used to level 2 cache (MB/s).............................  1326.512
Bandwidth used to level 3 cache (MB/s).............................  385.087
% cycles with no instruction issue..................................  10.410
% cycles stalled on memory access...................................  43.139
MFLOPS (cycles).....................................................  115.905
MFLOPS (wallclock)..................................................  114.441
MIPS (cycles).......................................................  1412.190
MIPS (wallclock)....................................................  1394.349
CPU time (seconds)..................................................  743.058
Wall clock time (seconds)...........................................  752.566
% CPU utilization...................................................  98.737
```

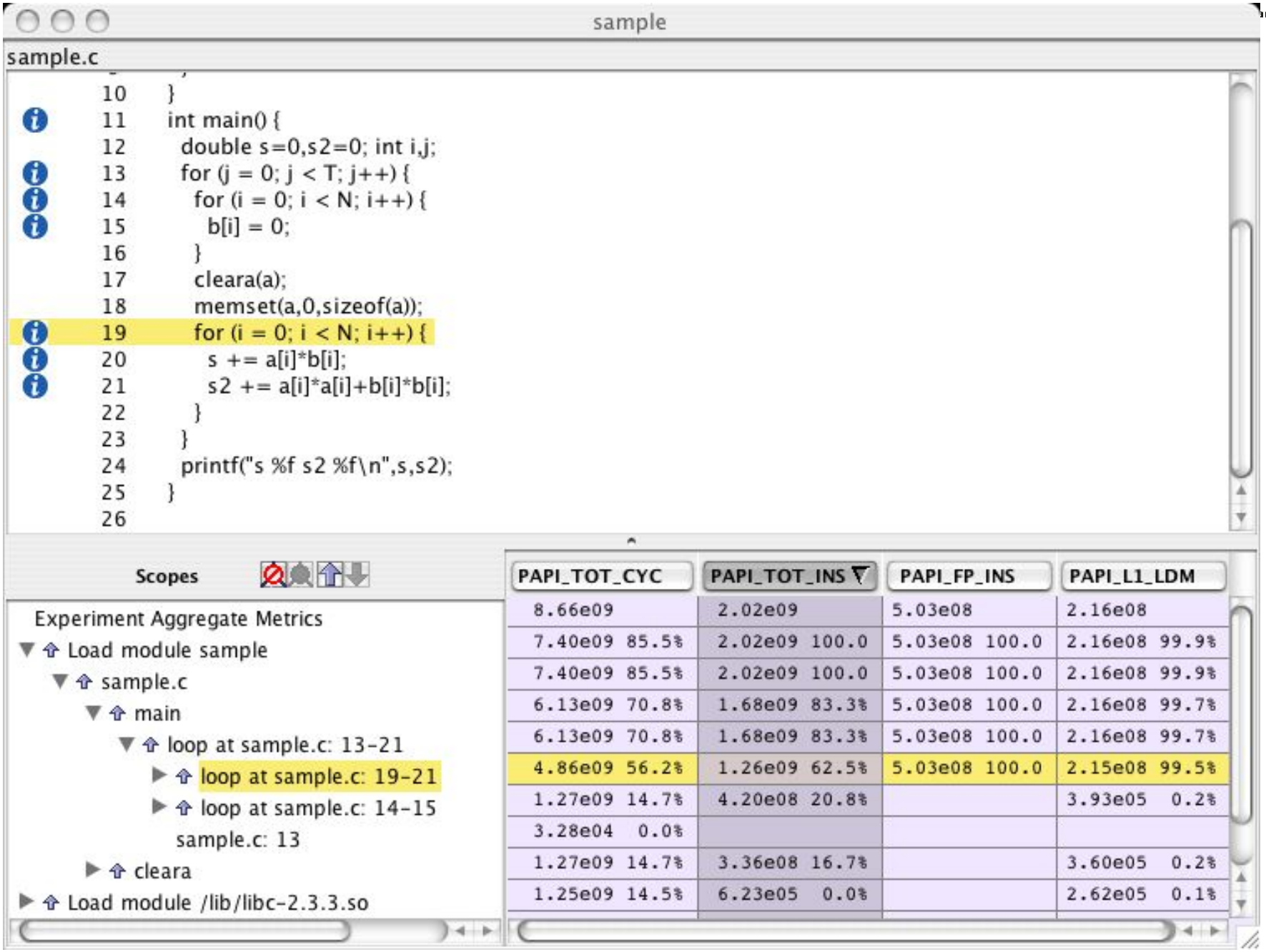
Parallelldatorcentrum

HPCToolkit

HPCToolkit

- A statistical profiling package based on interrupts from the performance monitoring hardware.
- No instrumentation required, but compiling with -g helps.
- Does not work on statically linked programs.
- 2 phase, collections and visualization.

HPCToolkit Sample Output



hpcrun Usage


```
hpcrun [-lLVhr] [-t each,all] [-e event:[period]] [-o path] [-f flag]
   <cmd> -- <cmd options>

 -l              List the available events.

 -L              List detailed information about all events.

 -V              Print version information.

 -h              Print this message.

 -r              Do not follow subprocesses.

 -t[each,all]    Profile threaded applications.

 -e event[:period] Sample event every period counts.

 -o path         Directory for output data.

 -f flag         PAPI profile mode.


Default is to profile every 32768 cycles (PAPI_TOT_CYC).
```

hpcprof Usage


```
hpcprof [-hefrl] [-H dir] <cmd> <profile> ... <profile>
 -h              Print this message.
 -e              Dump all information.
 -f              Dump results by file.
 -r              Dump results by function.
 -l              Dump results by line.
 -H dir        Dump HTML into dir.

There are more options.
```

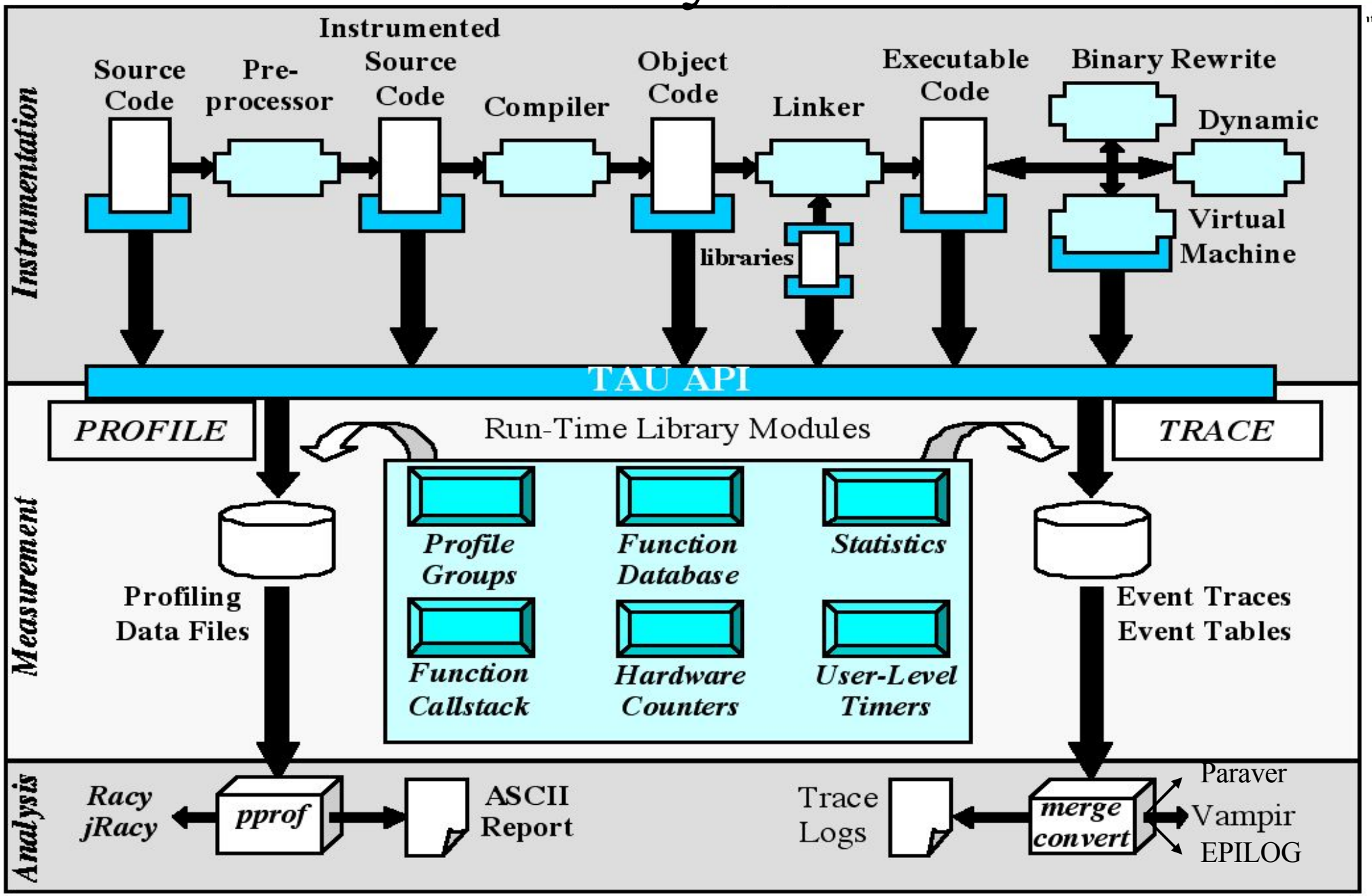
hpcprof Output


```
[mucci@h05n05:~]$ hpcprof -e ./a.out ./a.out.PAPI_TOT_CYC.h05n05.pdc.kth.se.13255.0
Columns correspond to the following events [event:period (events/sample)]
  PAPI_TOT_CYC:32767 - Total cycles (24755 samples)

Load Module Summary:
 100.0% /afs/pdc.kth.se/home/m/mucci/a.out

File Summary:
 100.0% <</afs/pdc.kth.se/home/m/mucci/a.out>>/afs/pdc.kth.se/home/m/mucci/main.c

Function Summary:
 100.0% <</afs/pdc.kth.se/home/m/mucci/a.out>>main

Line Summary:
  90.0% <</afs/pdc.kth.se/home/m/mucci/a.out>>/afs/pdc.kth.se/home/m/mucci/main.c:7
   5.8% <</afs/pdc.kth.se/home/m/mucci/a.out>>/afs/pdc.kth.se/home/m/mucci/main.c:6
   4.2% <</afs/pdc.kth.se/home/m/mucci/a.out>>/afs/pdc.kth.se/home/m/mucci/main.c:4

File <</afs/pdc.kth.se/home/m/mucci/a.out>>/afs/pdc.kth.se/home/m/mucci/main.c with profile
annotations.
     1        main()
     2        {
     3        int i;
     4   4.2% for (i=0;i<10000000;i++)
     5        {
     6   5.8% double a = 1.0, b = 2.0, c = 3.0;
     7  90.0% a += b * c + (double)i;
     8        }}
     9
```

Parallelldatorcentrum

TAU

TAU Performance System Architecture



Strategies for Empirical Performance Evaluation

- Empirical performance evaluation as a series of performance experiments
  - Experiment trials describing instrumentation and measurement requirements
  - Where/When/How axes of empirical performance space
    - where are performance measurements made in program
      - routines, loops, statements…
    - when is performance instrumentation done
      - compile-time, while pre-processing, runtime…
    - how are performance measurement/instrumentation chosen
      - profiling with hw counters, tracing, callpath profiling…

TAU Instrumentation Approach

- Support for standard program events
  - Routines
  - Classes and templates
  - Statement-level blocks
- Support for user-defined events
  - Begin/End events (“user-defined timers”)
  - Atomic events (e.g., size of memory allocated/freed)
  - Selection of event statistics
- Support definition of “semantic” entities for mapping
- Support for event groups
- Instrumentation optimization

TAU Instrumentation

- Flexible instrumentation mechanisms at multiple levels
  - Source code
    - manual
    - automatic
      - C, C++, F77/90/95 (Program Database Toolkit (*PDT*))
      - OpenMP (directive rewriting (*Opari*), *POMP spec*)
  - Object code
    - pre-instrumented libraries (e.g., MPI using *PMPI*)
    - statically-linked and dynamically-linked
  - Executable code
    - dynamic instrumentation (pre-execution) (*DynInstAPI*)
    - virtual machine instrumentation (e.g., Java using *JVMPI*)

Multi-Level Instrumentation

- Targets common measurement interface
  - *TAU API*
- Multiple instrumentation interfaces
  - Simultaneously active
- Information sharing between interfaces
  - Utilizes instrumentation knowledge between levels
- Selective instrumentation
  - Available at each level
  - Cross-level selection
- Targets a common performance model
- Presents a unified view of execution
  - Consistent performance events

Program Database Toolkit (PDT)

- Program code analysis framework
  - develop source-based tools
- *High-level interface* to source code information
- *Integrated toolkit* for source code parsing, database creation, and database query
  - Commercial grade front-end parsers
  - Portable IL analyzer, database format, and access API
  - Open software approach for tool development
- Multiple source languages
- Implement automatic performance instrumentation tools
  - *tau_instrumentor*

TAU Performance Measurement

- TAU supports profiling and tracing measurement
- Robust timing and hardware performance support using PAPI
- Support for online performance monitoring
  - Profile and trace performance data export to file system
  - Selective exporting
- Extension of TAU measurement for multiple counters
  - Creation of user-defined TAU counters
  - Access to system-level metrics
- Support for callpath measurement
- Integration with system-level performance data
  - Linux MAGNET/MUSE (Wu Feng, LANL)

TAU Measurement

- Performance information
  - Performance events
  - High-resolution timer library (real-time / virtual clocks)
  - General software counter library (user-defined events)
  - Hardware performance counters
    - *PAPI* (Performance API) (UTK, Ptools Consortium)
    - consistent, portable API
- Organization
  - Node, context, thread levels
  - Profile groups for collective events (runtime selective)
  - Performance data mapping between software levels

TAU Measurement Options

- Parallel profiling
  - Function-level, block-level, statement-level
  - Supports user-defined events
  - TAU parallel profile data stored during execution
  - Hardware counts values
  - Support for multiple counters
  - Support for callgraph and callpath profiling
- Tracing
  - All profile-level events
  - Inter-process communication events
  - Trace merging and format conversion

Grouping Performance Data in TAU

- Profile Groups
  - A group of related routines forms a profile group
  - Statically defined
    - TAU_DEFAULT, TAU_USER[1-5], TAU_MESSAGE, TAU_IO, …
  - Dynamically defined
    - group name based on string, such as "adlib" or "particles"
    - runtime lookup in a map to get unique group identifier
    - uses *tau_instrumentor* to instrument
  - Ability to change group names at runtime
  - Group-based instrumentation and measurement control

TAU Analysis

- Parallel profile analysis
  - *Pprof*
    - parallel profiler with text-based display
  - *ParaProf*
    - Graphical, scalable, parallel profile analysis and display
- Trace analysis and visualization
  - Trace merging and clock adjustment (if necessary)
  - Trace format conversion (ALOG, SDDF, VTF, Paraver)
  - Trace visualization using *Vampir* (Pallas/Intel)

Pprof Output (NAS LU Parallel)

- Intel Quad PIII Xeon
- F90 + MPICH
- Profile
  - Node
  - Context
  - Thread
- Events
  - code
  - MPI

emacs@neutron.cs.uoregon.edu

Buffers Files Tools Edit Search Mule Help

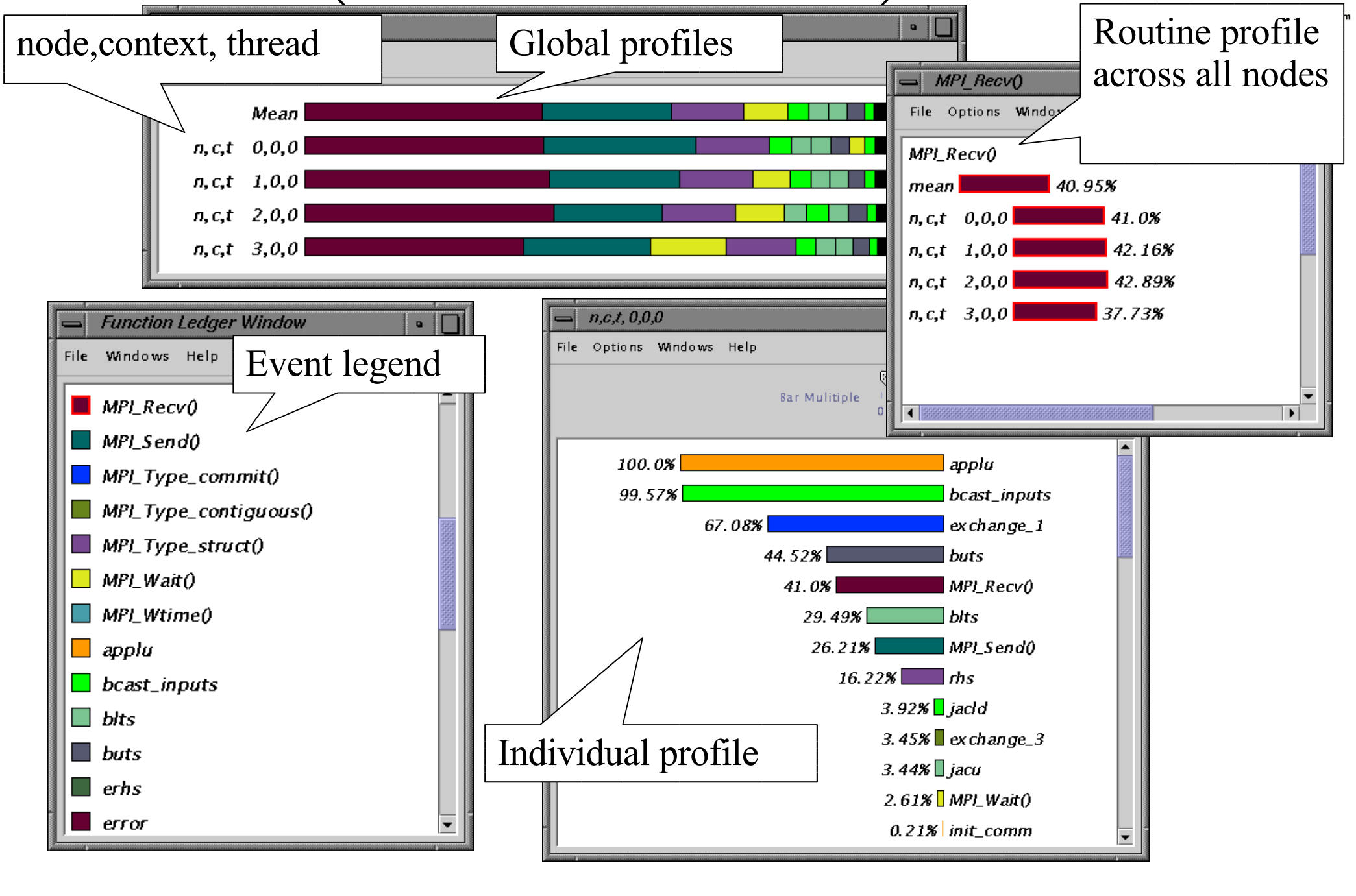
Reading Profile files in profile.*

NODE 0;CONTEXT 0;THREAD 0:

%Time	Exclusive msec	Inclusive total msec	#Call	#Subrs	Inclusive usec/call	Name
100.0	1	3:11.293	1	15	191293269	applu
99.6	3,667	3:10.463	3	37517	63487925	bcast_inputs
67.1	491	2:08.326	37200	37200	3450	exchange_1
44.5	6,461	1:25.159	9300	18600	9157	buts
41.0	1:18.436	1:18.436	18600	0	4217	MPI_Recv()
29.5	6,778	56,407	9300	18600	6065	blts
26.2	50,142	50,142	19204	0	2611	MPI_Send()
16.2	24,451	31,031	301	602	103096	rhs
3.9	7,501	7,501	9300	0	807	jacld
3.4	838	6,594	604	1812	10918	exchange_3
3.4	6,590	6,590	9300	0	709	jacu
2.6	4,989	4,989	608	0	8206	MPI_Wait()
0.2	0.44	400	1	4	400081	init_comm
0.2	398	399	1	39	399634	MPI_Init()
0.1	140	247	1	47616	247086	setiv
0.1	131	131	57252	0	2	exact
0.1	89	103	1	2	103168	erhs
0.1	0.966	96	1	2	96458	read_input
0.0	95	95	9	0	10603	MPI_Bcast()
0.0	26	44	1	7937	44878	error
0.0	24	24	608	0	40	MPI_Irecv()
0.0	15	15	1	5	15630	MPI_Finalize()
0.0	4	12	1	1700	12335	setbv
0.0	7	8	3	3	2893	l2norm
0.0	3	3	8	0	491	MPI_Allreduce()
0.0	1	3	1	6	3874	pintgr
0.0	1	1	1	0	1007	MPI_Barrier()
0.0	0.116	0.837	1	4	837	exchange_4
0.0	0.512	0.512	1	0	512	MPI_Keyval_create()
0.0	0.121	0.353	1	2	353	exchange_5
0.0	0.024	0.191	1	2	191	exchange_6
0.0	0.103	0.103	6	0	17	MPI_Type_contiguous()

--:-- NPB_LU.out (Fundamental)--L8--Top

ParaProf (NAS LU Parallel)

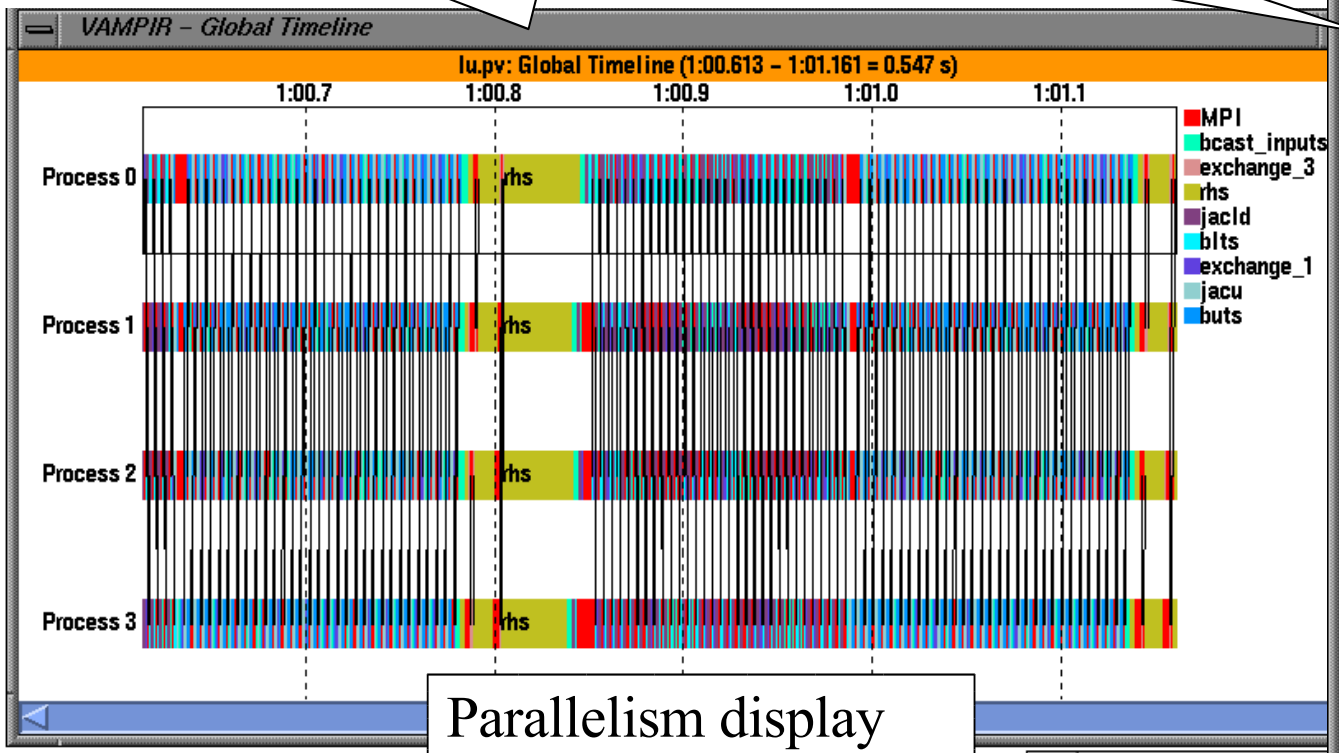


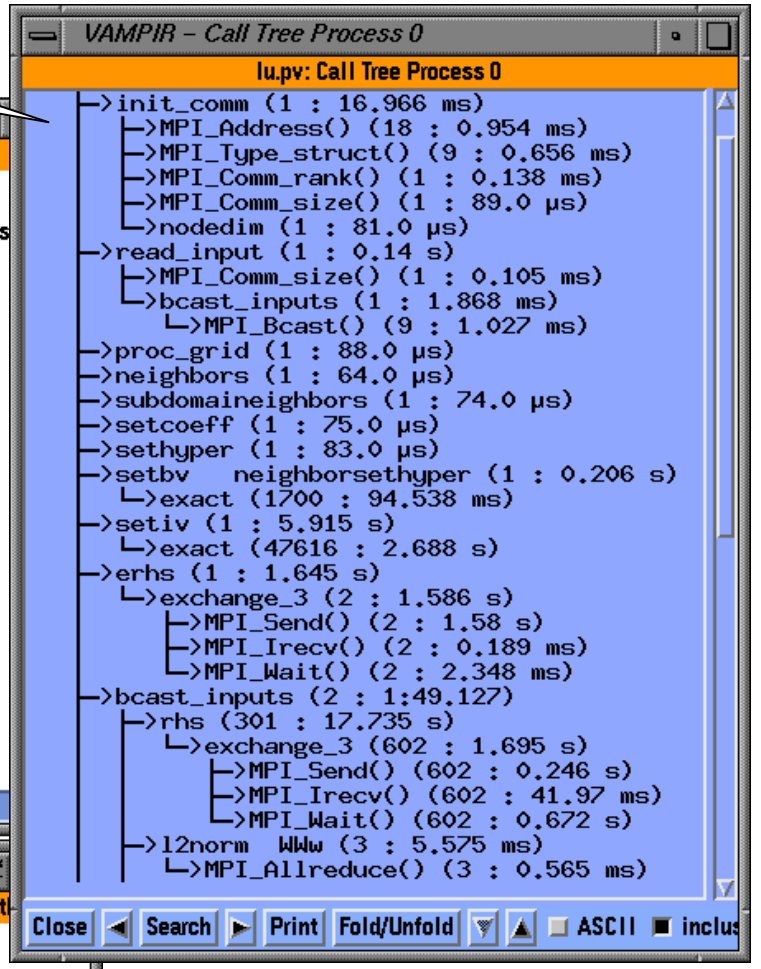
TAU + Vampir (NAS Parallel Benchmark – LU)

Parallelldatorcentrum

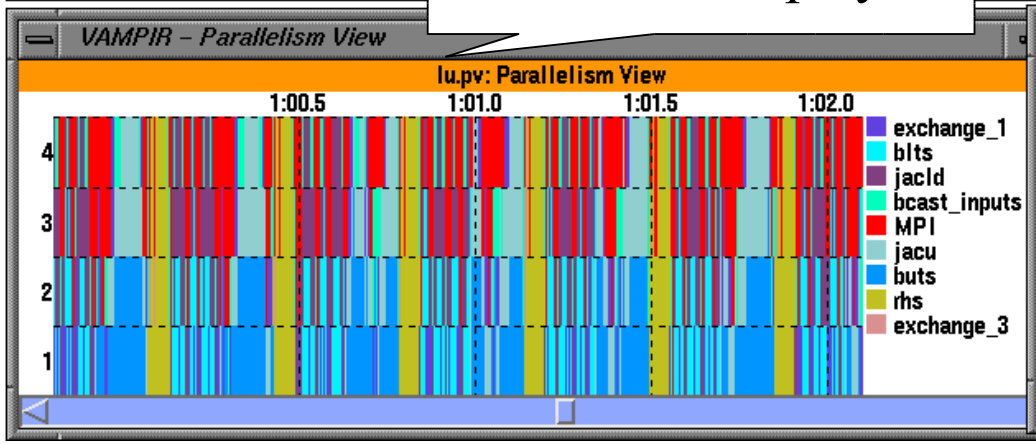
Timeline display

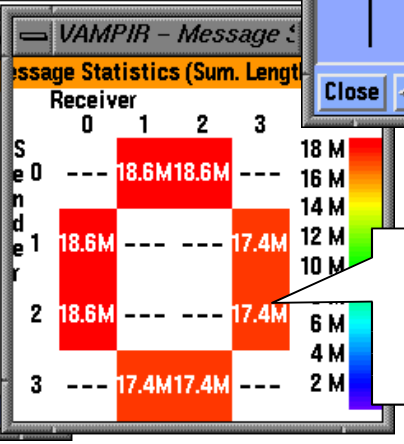
Callgraph display





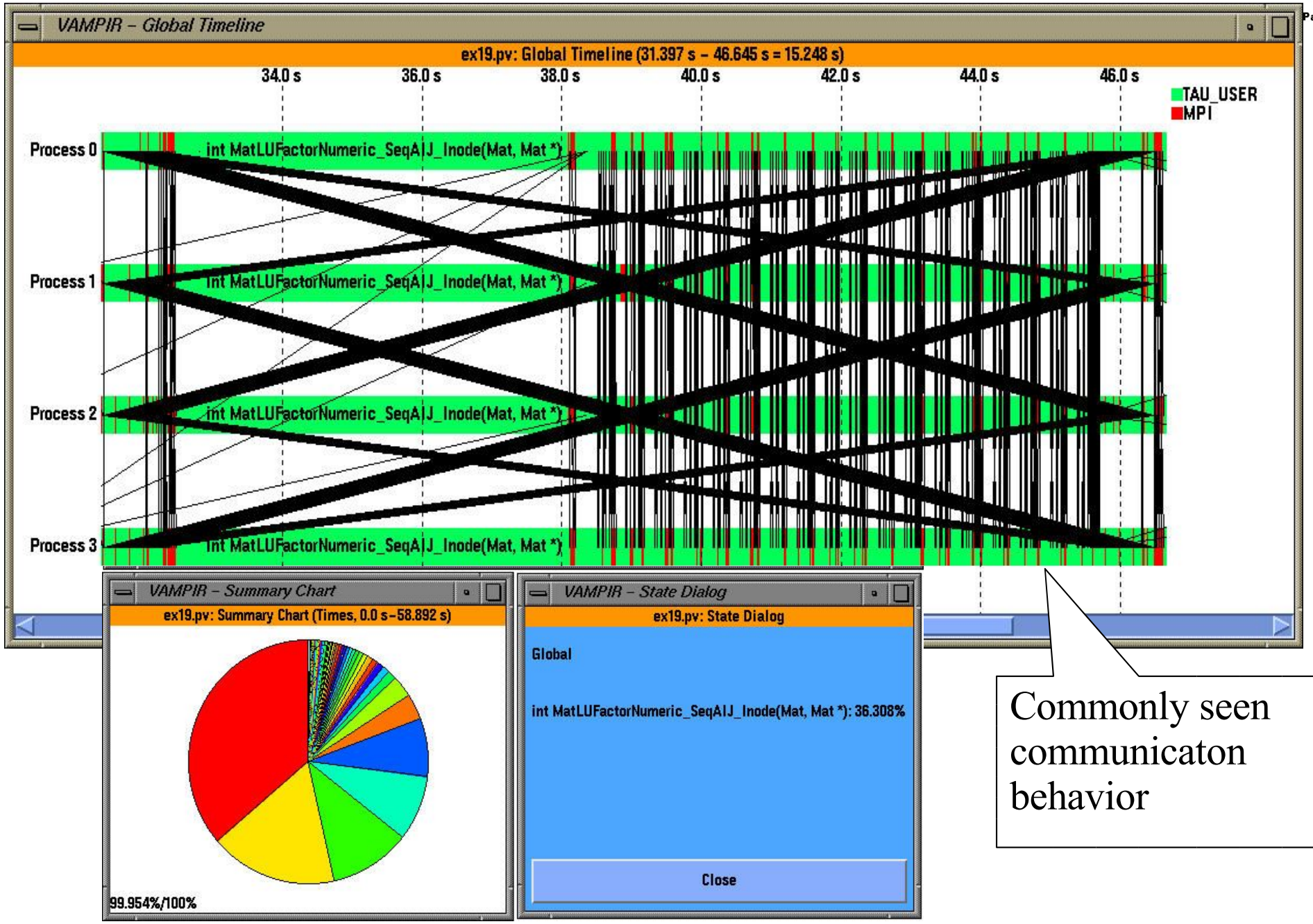
Parallelism display





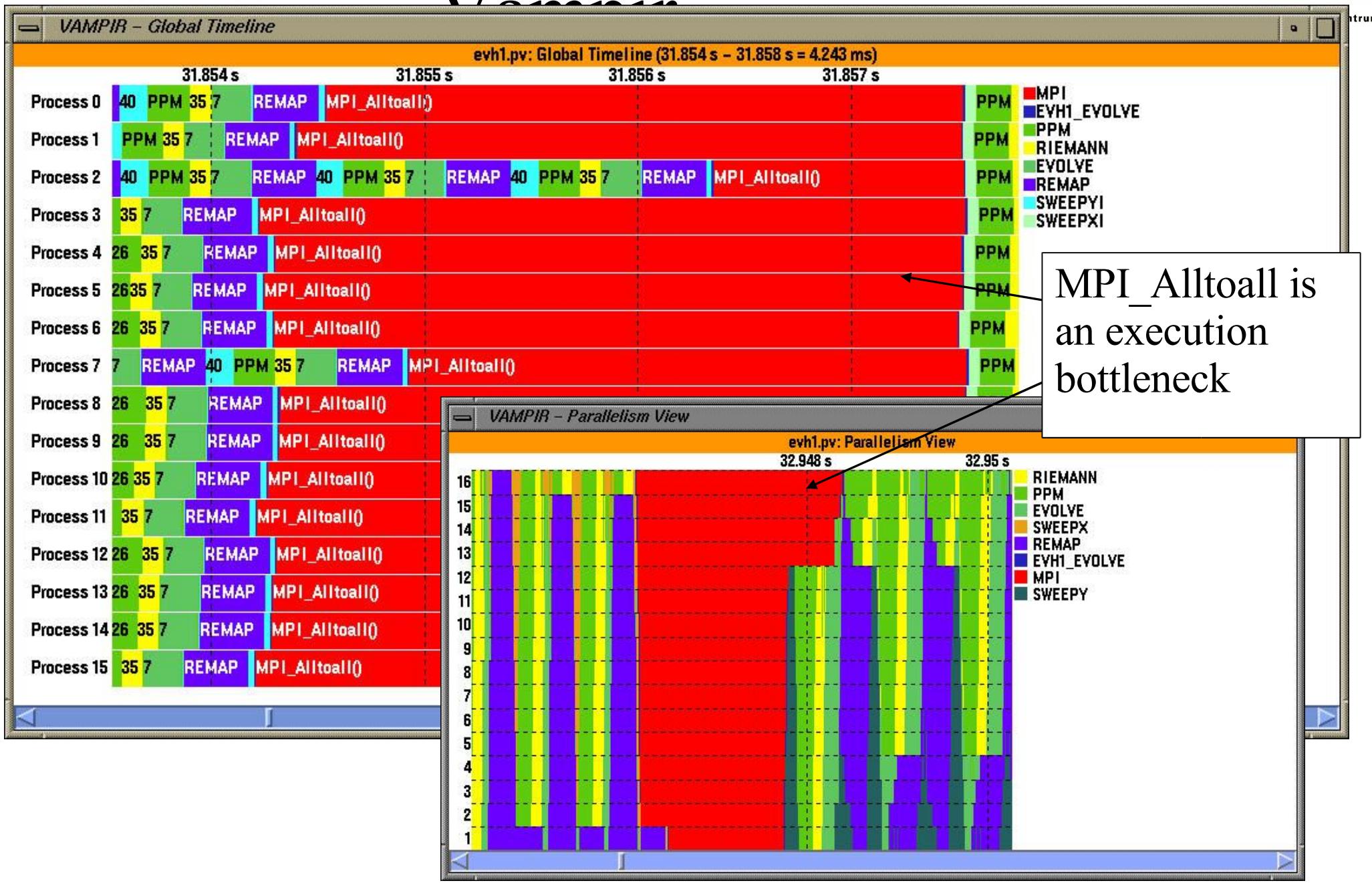
Communications display

PETSc ex19 (Tracing)



Commonly seen communicaton behavior

TAU's EVH1 Execution Trace in Vampir

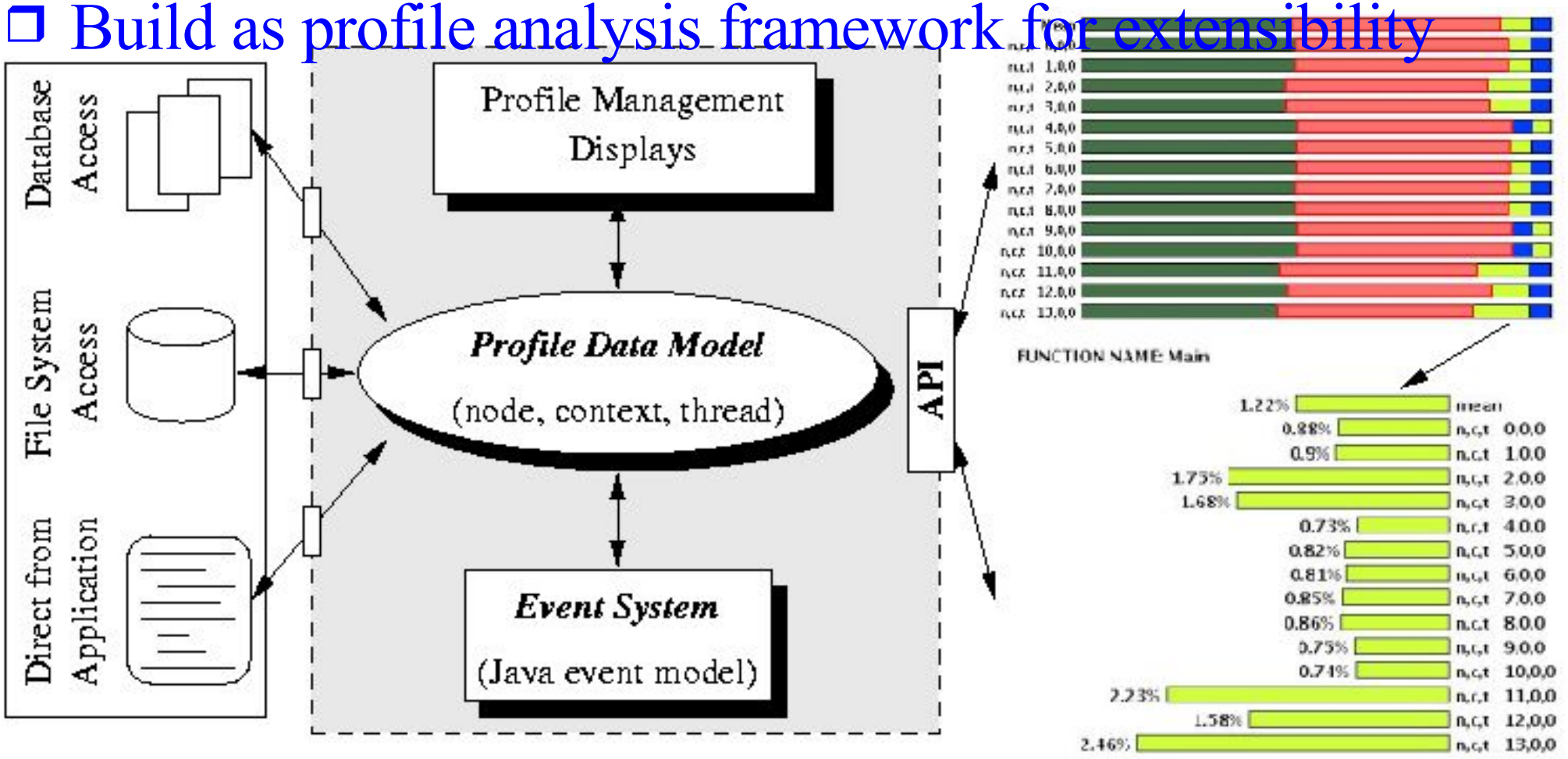


Performance Analysis and Visualization

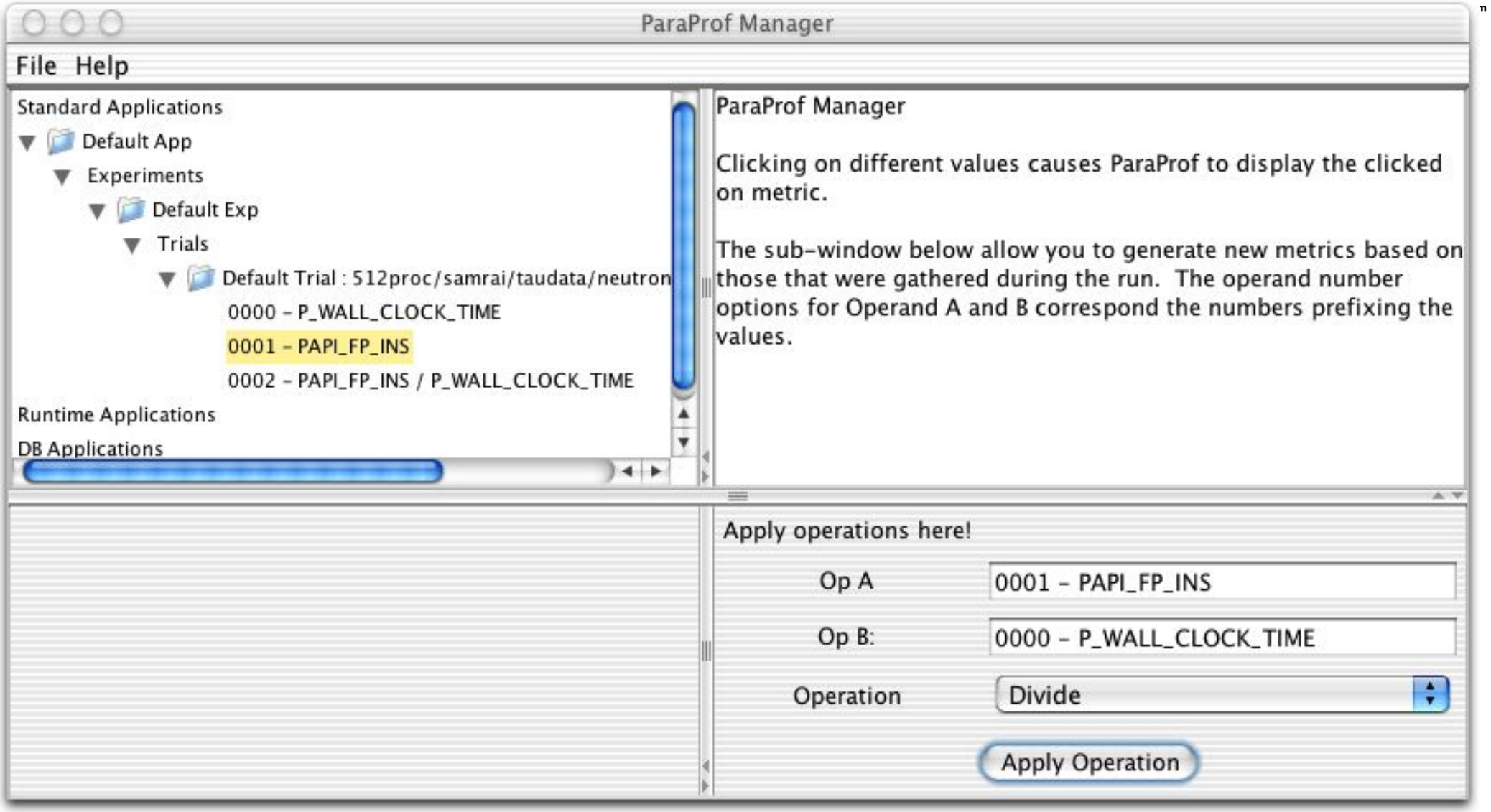
- Analysis of parallel profile and trace measurement
- Parallel profile analysis
  - ParaProf
  - Profile generation from trace data
- Performance database framework (PerfDBF)
- Parallel trace analysis
  - Translation to VTF 3.0 and EPILOG
  - Integration with VNG (Technical University of Dresden)
- Online parallel analysis and visualization

ParaProf Framework Architecture

- Portable, extensible, and scalable tool for profile analysis
- Try to offer "best of breed" capabilities to analysts
- Build as profile analysis framework for extensibility

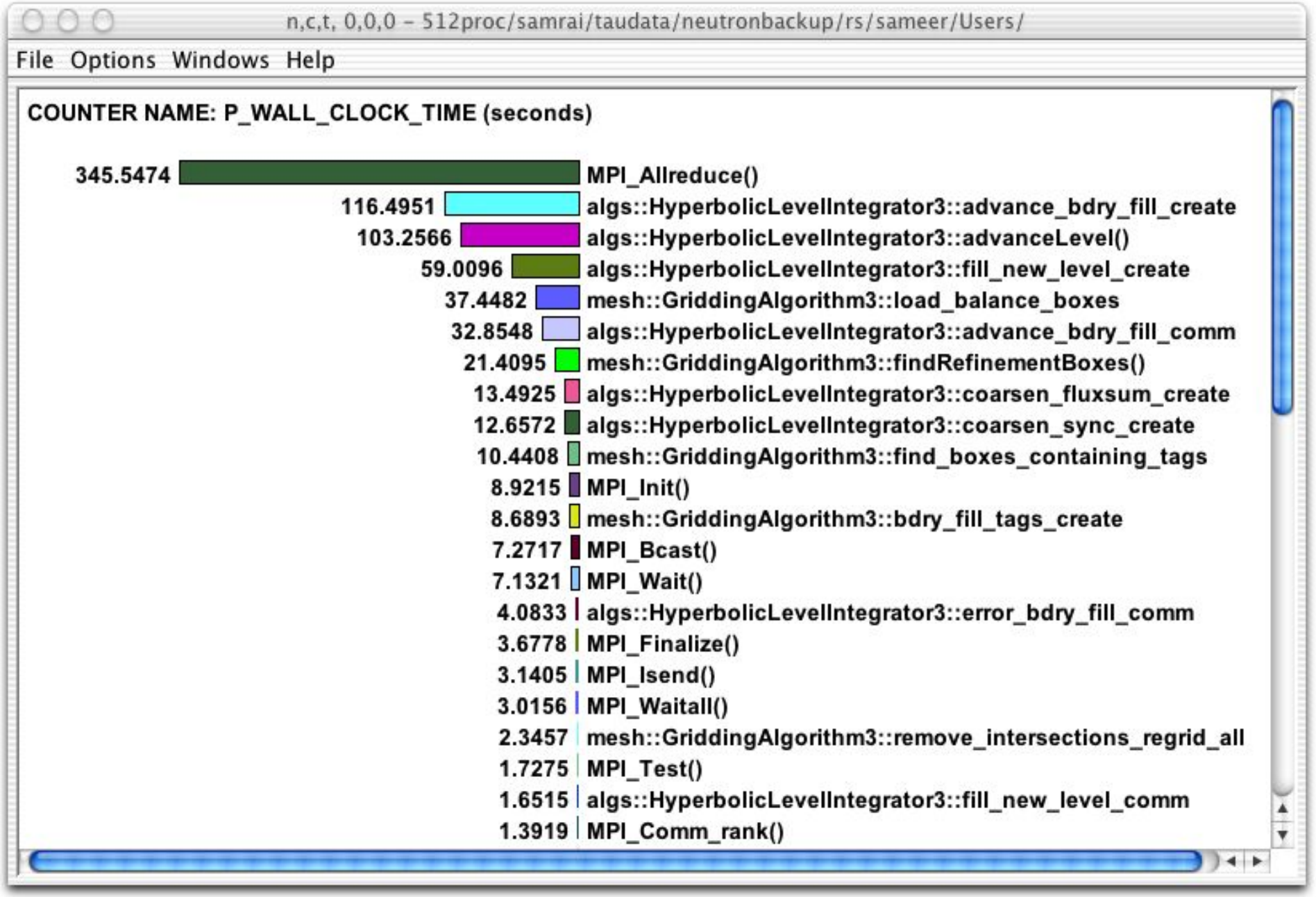


Profile Manager Window

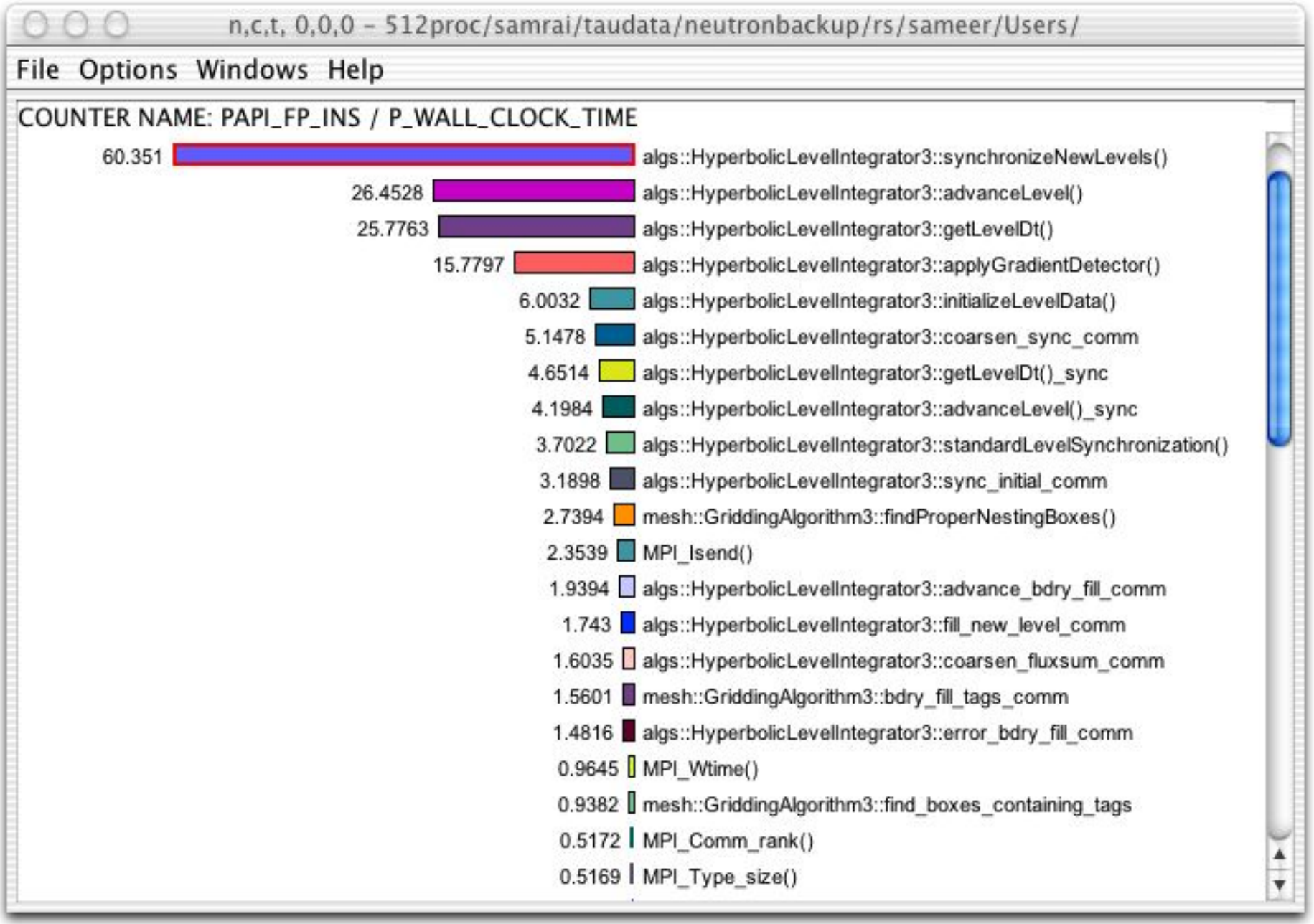


- Structured AMR toolkit (SAMRAI++), LLNL

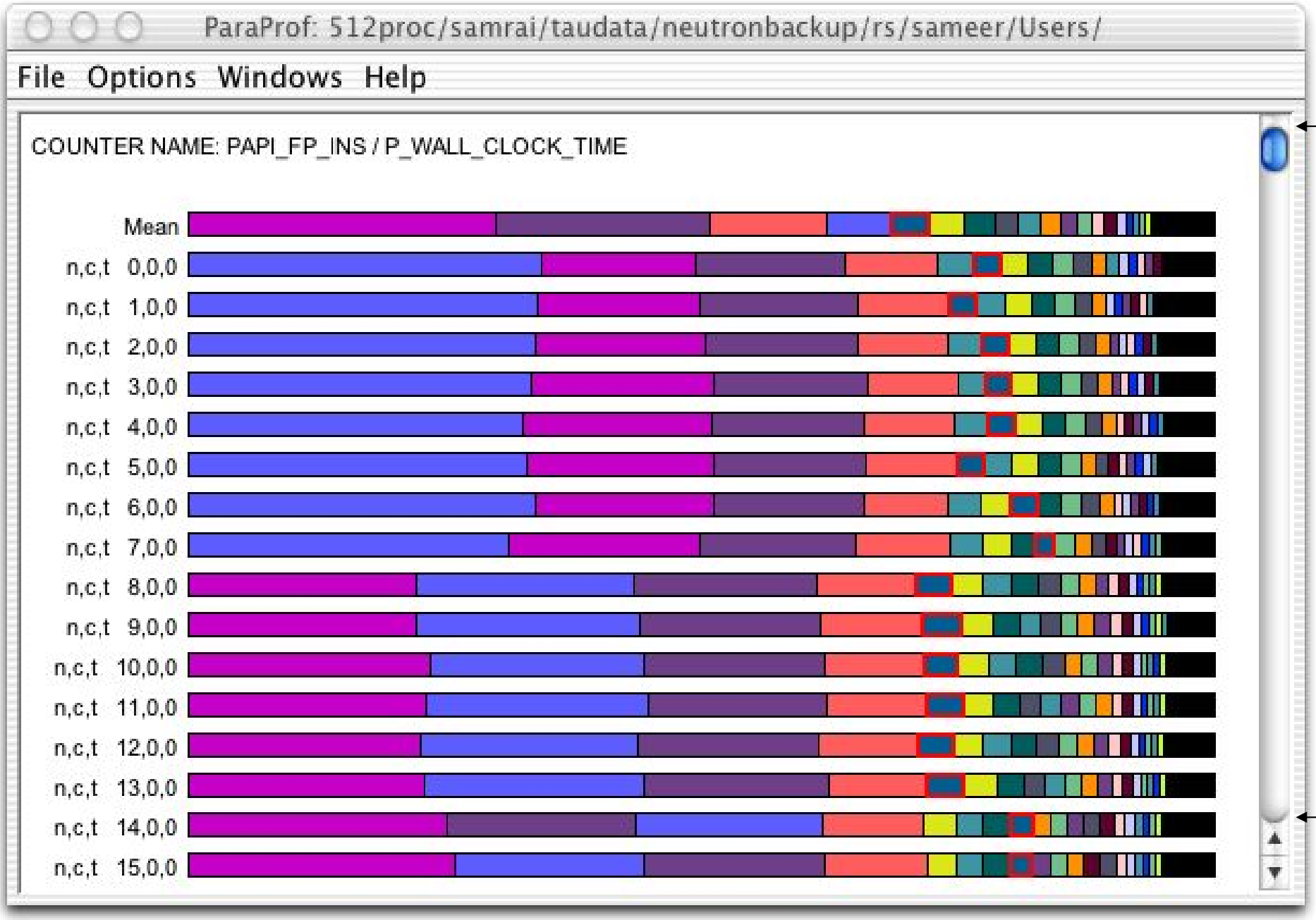
Node / Context / Thread Profile Window



Derived Metrics



Full Profile Window (Metric-specific)

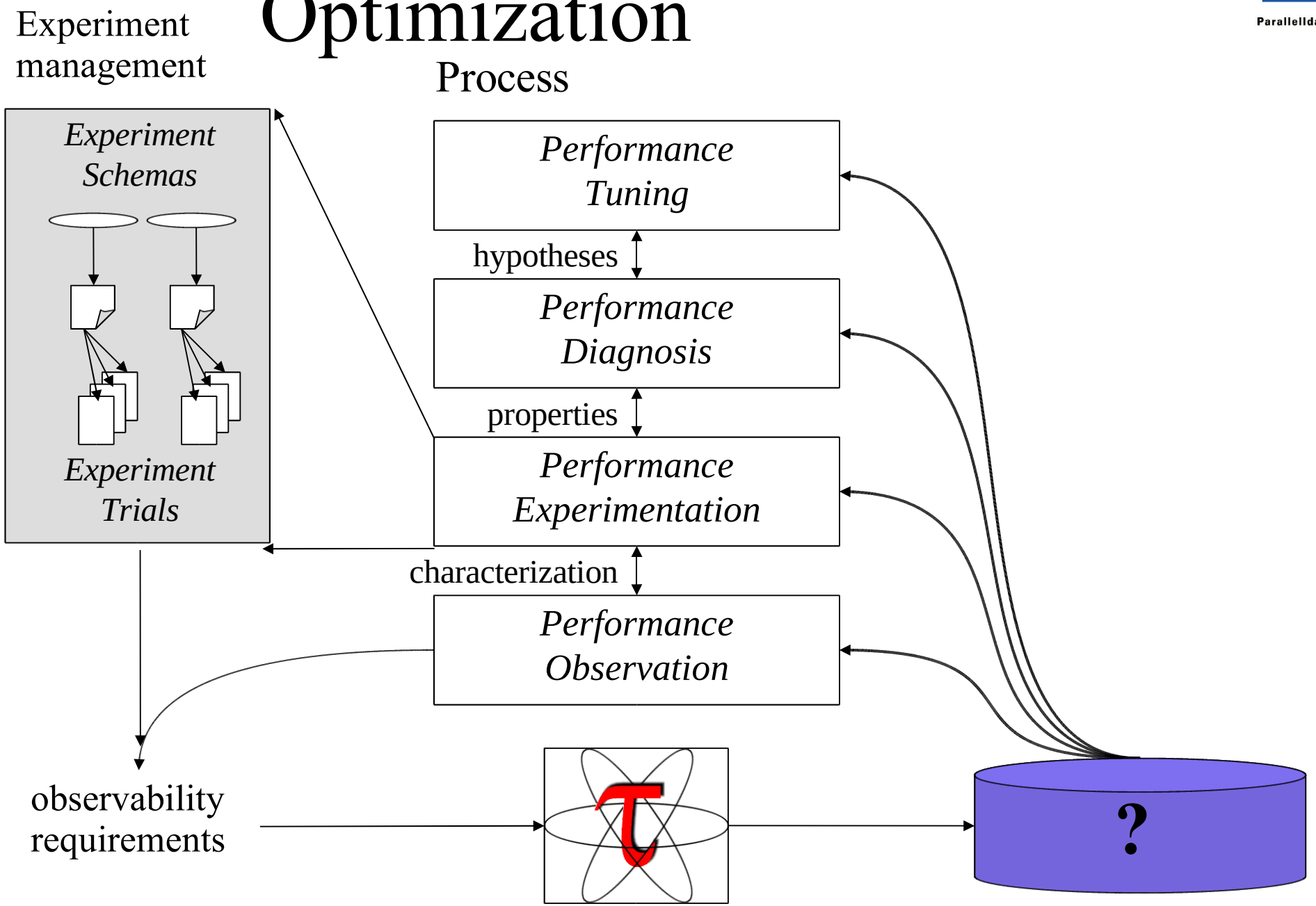


ParaProf Enhancements

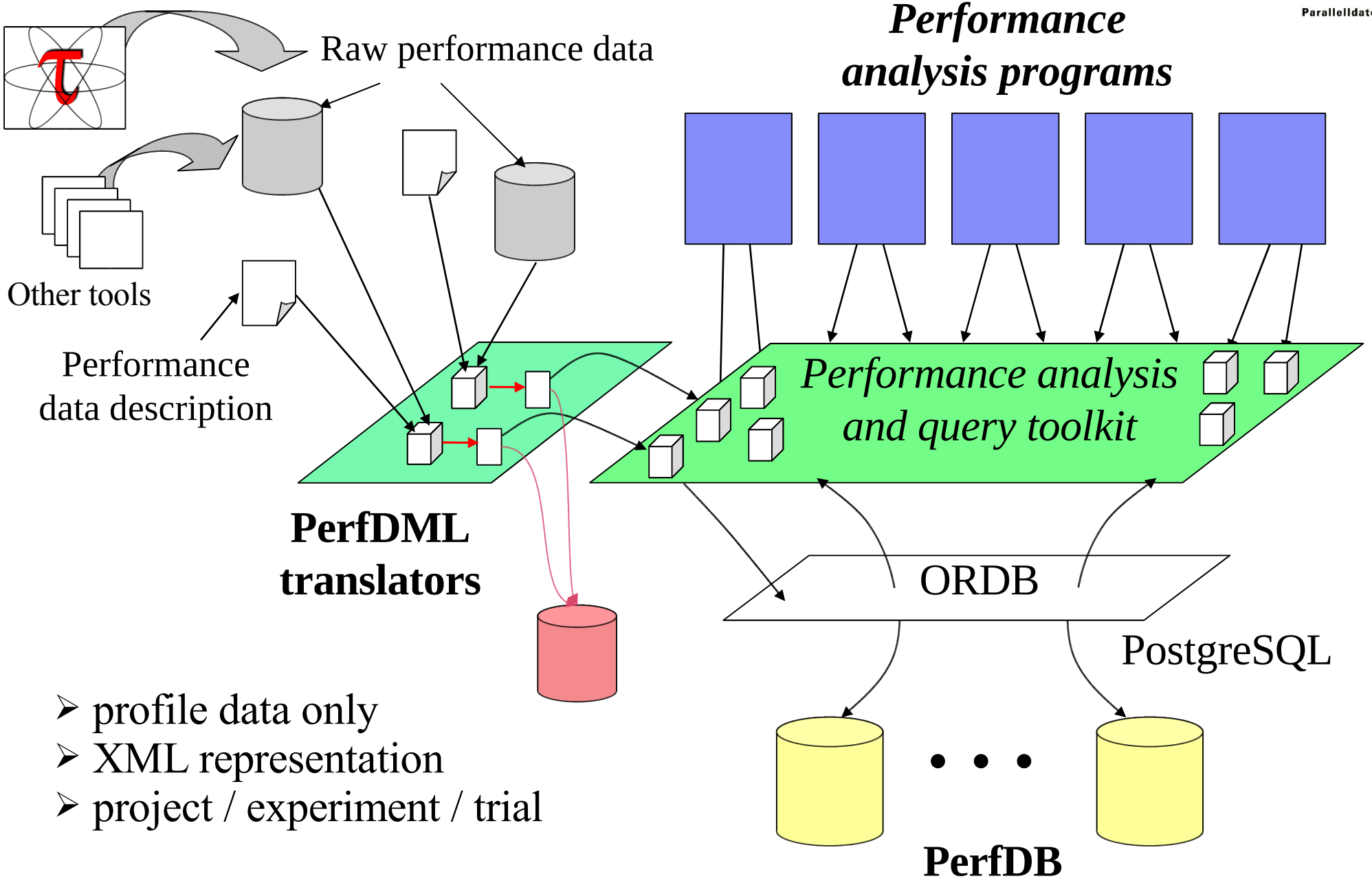
- Readers completely separated from the GUI
- Access to performance profile database

- Profile translators
  - mpiP, papiprof, dynaprof
- Callgraph display
  - prof/gprof style with hyperlinks
- Integration of 3D performance plotting library
- Scalable profile analysis
  - Statistical histograms, cluster analysis, …
- Generalized programmable analysis engine
- Cross-experiment analysis

Empirical-Based Performance Optimization

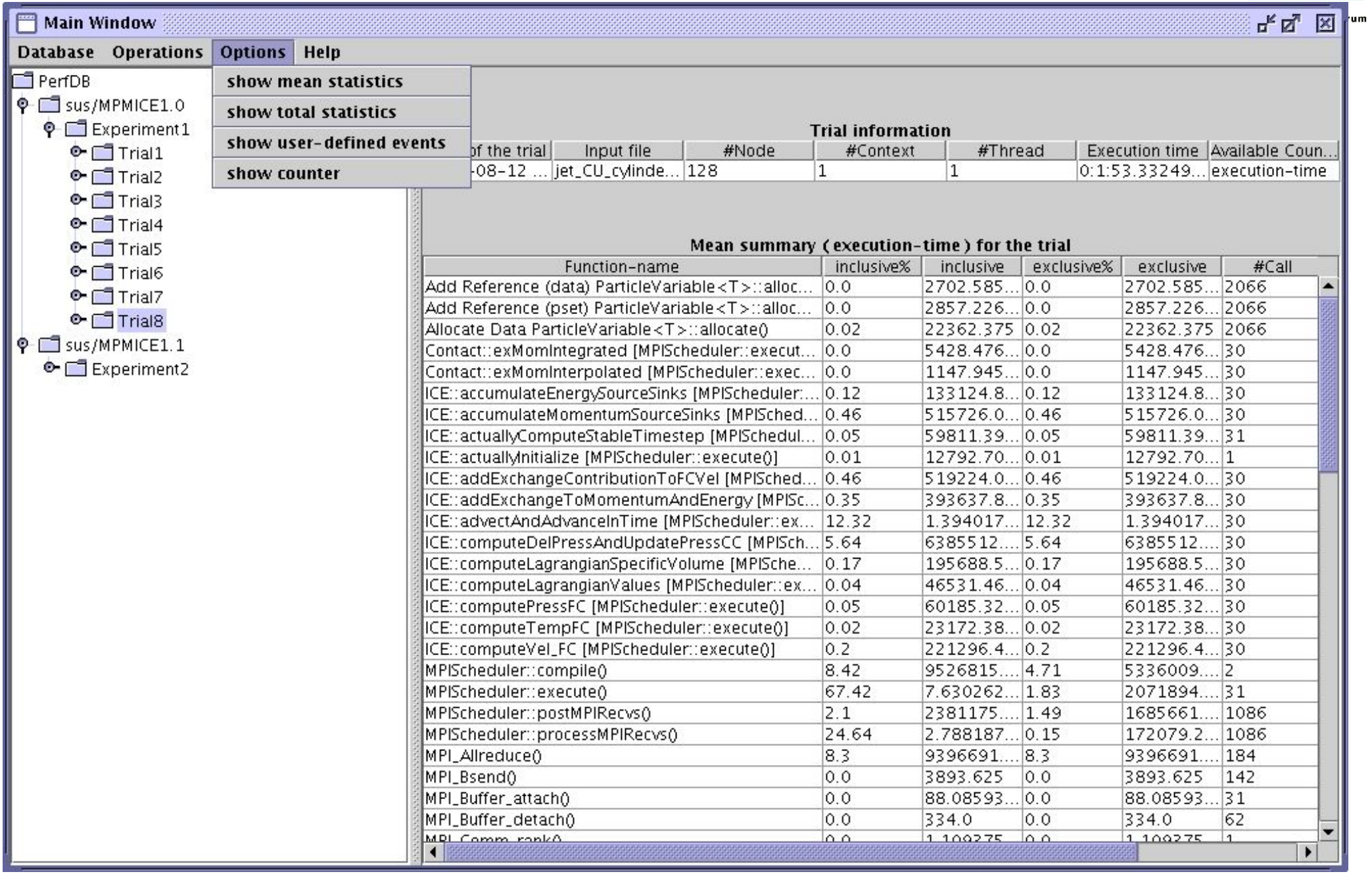


TAU Performance Database Framework

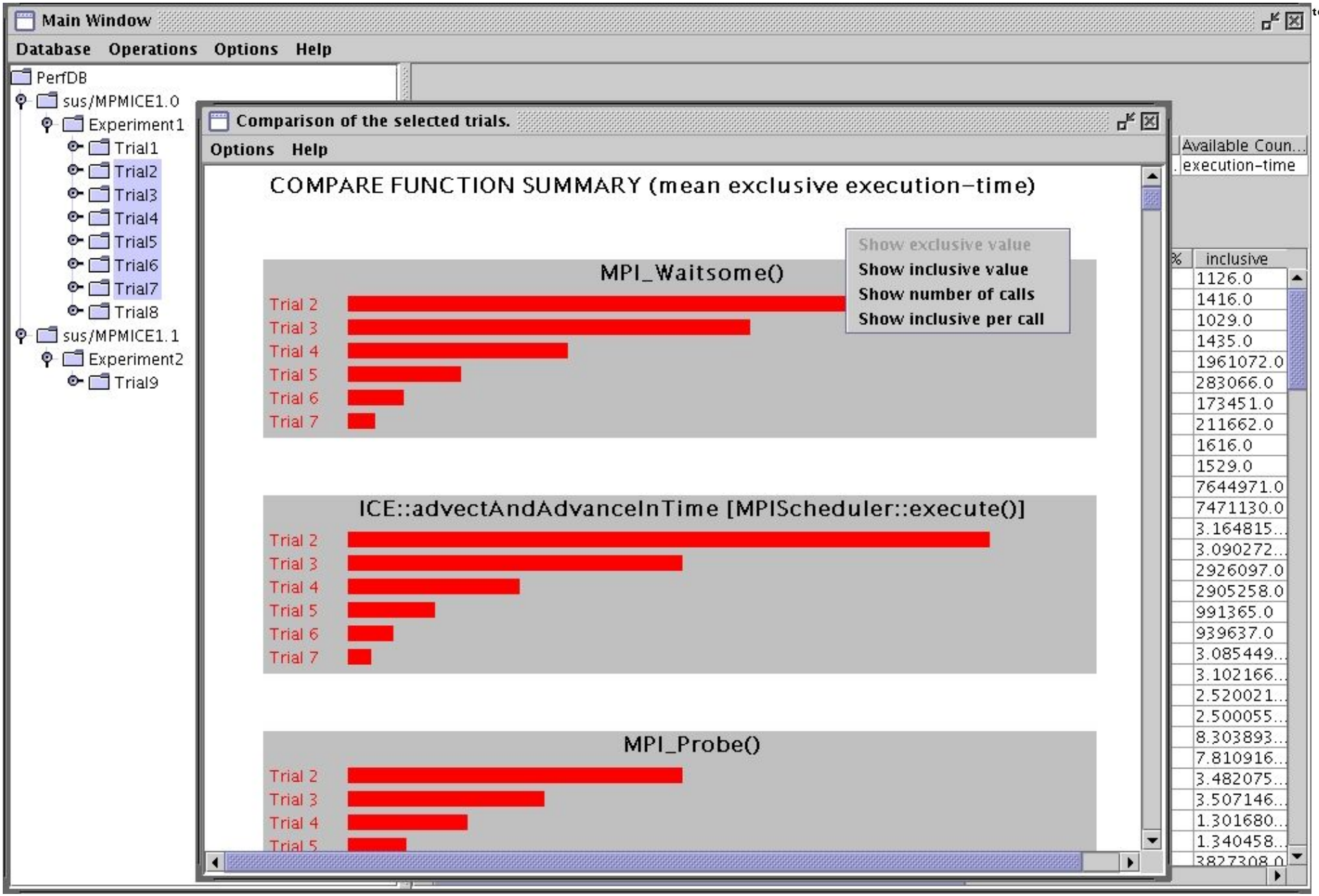


- profile data only
- XML representation
- project / experiment / trial

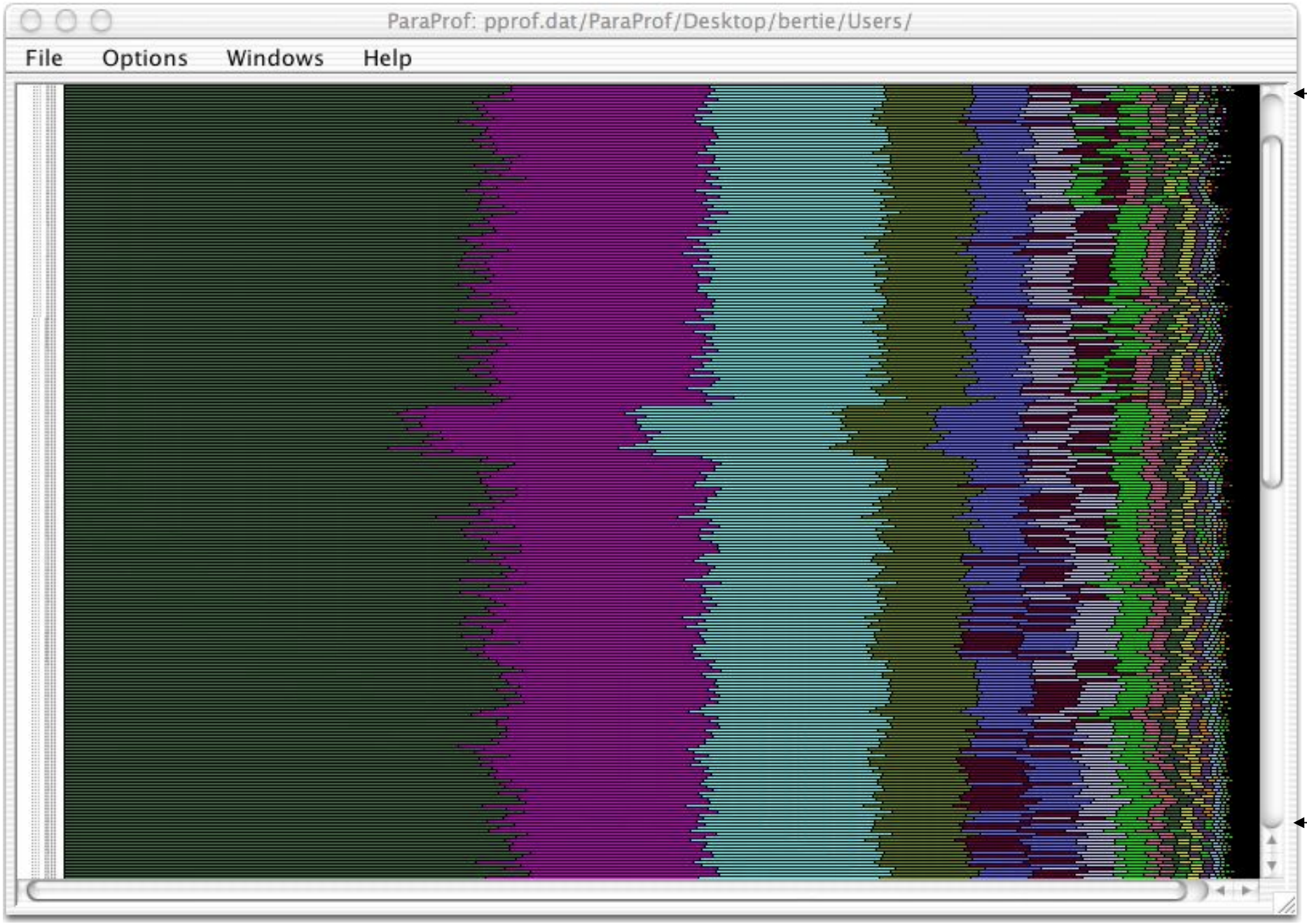
PerfDBF Browser



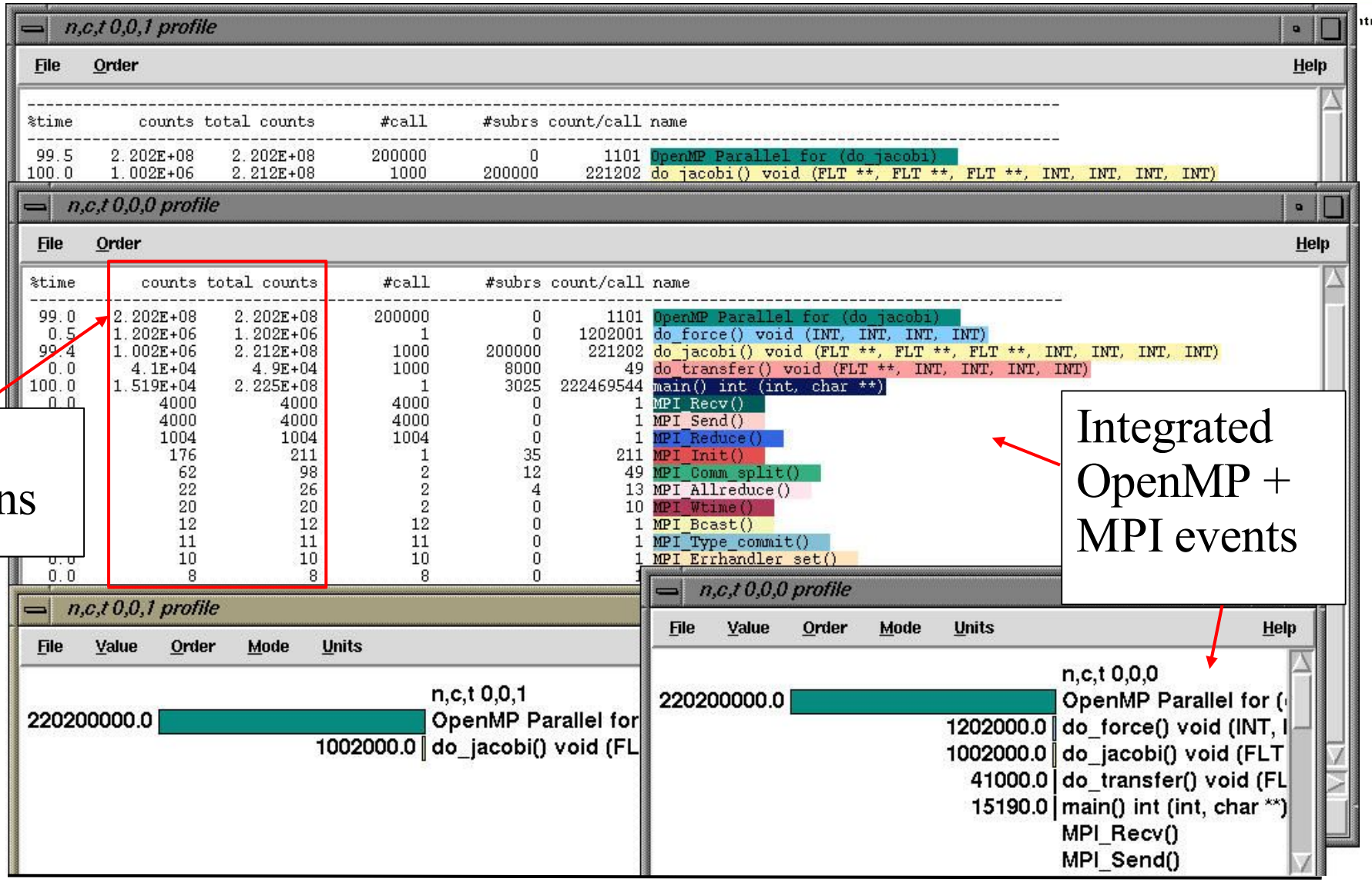
PerfDBF Cross-Trial Analysis



Full Profile Window



OpenMP + MPI Ocean Modeling



Parallelldatorcentrum

KOJAK

KOJAK Analysis process

Parallelldatorcentrum

Source code

Automatic multilevel instrumentation

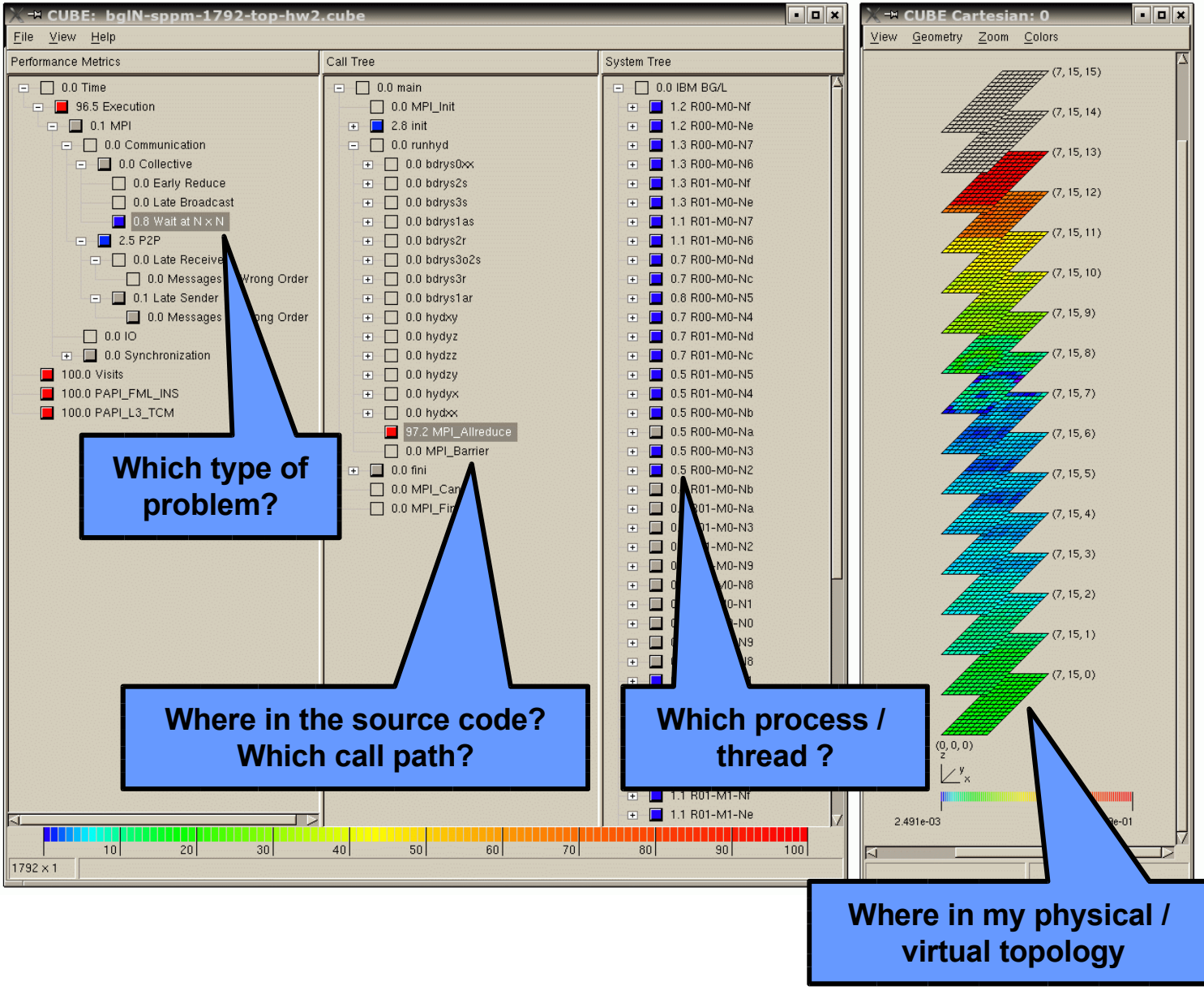
Executable

Execution on parallel machine

Event Trace

Automatic pattern analysis

High-Level Profile



Parallelldatorcentrum

MPIP

MPIP Output


```
@ Command : /afs/pdc.kth.se/home/m/mucci/mpiP-2.7/testing/./sweep-ops-stack.exe
/tmp/SPnodes-mucci-0
@ Version                  : 2.7
@ MPIP Build date          : Aug 17 2004, 17:04:36
@ Start time               : 2004 08 17 17:08:48
@ Stop time                : 2004 08 17 17:08:48
@ MPIP env var             : [null]
@ Collector Rank           : 0
@ Collector PID            : 17412
@ Final Output Dir         : .
@ MPI Task Assignment      : 0 h05n05-e.pdc.kth.se
@ MPI Task Assignment      : 1 h05n35-e.pdc.kth.se
@ MPI Task Assignment      : 2 h05n05-e.pdc.kth.se
@ MPI Task Assignment      : 3 h05n35-e.pdc.kth.se

@--- MPI Time (seconds) ---------------------------------------------------
--------------------------------------------------------------------------
Task    AppTime    MPITime     MPI%
   0      0.084     0.0523    62.21
   1     0.0481      0.015    31.19
   2      0.087     0.0567    65.20
   3     0.0495     0.0149    29.98
   *      0.269      0.139    51.69

@--- Aggregate Time (top twenty, descending, milliseconds) ----------------
--------------------------------------------------------------------------
Call                 Site       Time    App%    MPI%
Barrier                 1        112   41.57   80.42
Recv                    1       26.2    9.76   18.89
Allreduce               1      0.634    0.24    0.46
Bcast                   1        0.3    0.11    0.22
Send                    1      0.033    0.01    0.02

@--- Aggregate Sent Message Size (top twenty, descending, bytes) ----------
--------------------------------------------------------------------------
Call                 Site      Count      Total       Avrg  Sent%
Allreduce               1          8    4.8e+03        600  46.15
Bcast                   1          8    4.8e+03        600  46.15
Send                    1          2        800        400   7.69
```

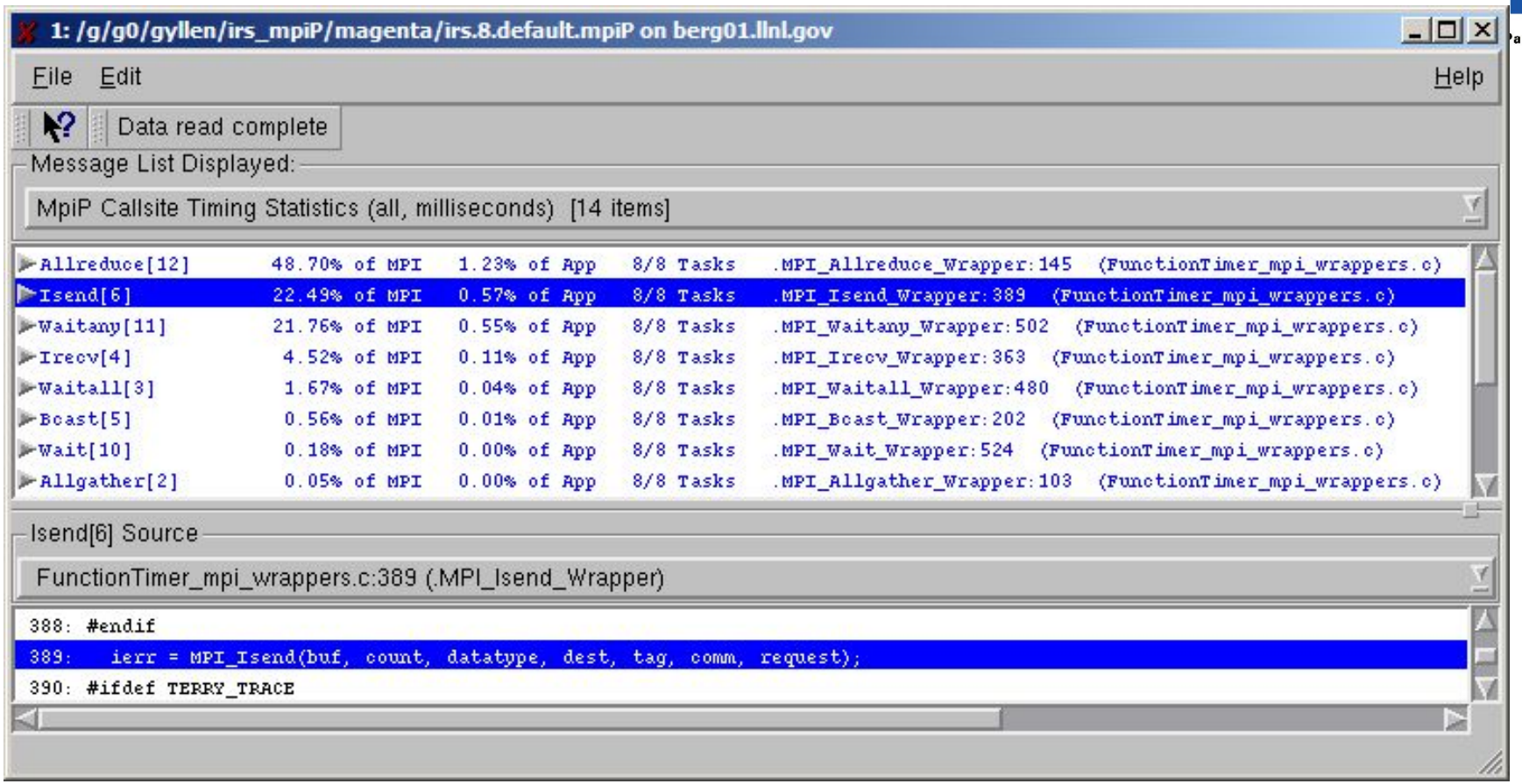
MPIP Output (2)


```
---------------------------------------------------------------------------
@--- Callsite Time statistics (all, milliseconds): 16 ---------------------
---------------------------------------------------------------------------
Name                Site Rank  Count      Max     Mean      Min   App%   MPI%
Allreduce              1    0      2    0.105    0.087    0.069   0.21   0.33
Allreduce              1    1      2    0.118     0.08    0.042   0.33   1.07
Allreduce              1    2      2     0.11    0.078    0.046   0.18   0.27
Allreduce              1    3      2    0.102    0.072    0.042   0.29   0.97
Barrier                1    0      3     51.9     17.3    0.015  61.86  99.44
Barrier                1    1      3    0.073   0.0457    0.016   0.29   0.91
Barrier                1    2      3     54.9     18.8    0.031  64.90  99.53
Barrier                1    3      3     1.56     1.02    0.035   6.20  20.68
Bcast                  1    0      2    0.073   0.0535    0.034   0.13   0.20
Bcast                  1    1      2    0.037    0.023    0.009   0.10   0.31
Bcast                  1    2      2    0.084    0.046    0.008   0.11   0.16
Bcast                  1    3      2     0.03   0.0275    0.025   0.11   0.37
Recv                   1    1      1     14.6     14.6     14.6  30.48  97.71
Recv                   1    3      1     11.6     11.6     11.6  23.37  77.98
Send                   1    0      1    0.013    0.013    0.013   0.02   0.02
Send                   1    2      1     0.02     0.02     0.02   0.02   0.04
Send                   1    *     32     54.9     4.34    0.008  51.69 100.00
---------------------------------------------------------------------------
@--- Callsite Message Sent statistics (all, sent bytes) -------------------
---------------------------------------------------------------------------
Name                Site Rank   Count       Max      Mean       Min       Sum
Allreduce              1    0       2       800       600       400      1200
Allreduce              1    1       2       800       600       400      1200
Allreduce              1    2       2       800       600       400      1200
Allreduce              1    3       2       800       600       400      1200
Bcast                  1    0       2       800       600       400      1200
Bcast                  1    1       2       800       600       400      1200
Bcast                  1    2       2       800       600       400      1200
Bcast                  1    3       2       800       600       400      1200
Send                   1    0       1       400       400       400       400
Send                   1    2       1       400       400       400       400
Send                   1    *      18       800     577.8       400  1.04e+04
---------------------------------------------------------------------------
@--- End of Report --------------------------------------------------------
```

Mpipview: An MpiP Output Viewer

- Organizes and condenses mpiP output
  - Allow users to find key mpiP data quickly
  - Hides complexity of large scale runs until needed
  - Shows source code for the MPI callsites reported on
  - Design based on our experience using mpiP on ASC apps
- Easy to use - parses mpiP text output file
  - mpipview irs.8.default.mpiP

MPI Callsite Timing Summaries



- Shows timing stats summaries, sorted by % of MPI
- Clicking on summary displays callsite's source code
  - Callsites indicate where an MPI call was called from
  - Isend[6] indicates the 6th MPI callsite reached was an MPI_Isend