CRAY The Supercomputer Company

The New Generation of Cray Performance Tools

Luiz DeRose
Sr. Principal Engineer
Tools Manager

Albuquerque, NM

CRAY USER GROUP INCORPORATED

CUG 2005

Outline

- Previous status
- The Cray tools strategy
- Cray Tools
  - HWPC
  - CrayPat
  - Cray Apprentice^2
- Future directions
- Conclusions

The Starting Point

- Missing a complete solution
  - Cray X1
    - Very good measurement infrastructure
    - Very poor user interface
  - XT3 & XD1
    - No in-house solution
      - Dependence on external providers
- Needed integration across platforms

The Cray Tools Strategy

- Must be easy to use
- Integrated performance tools solution
- Strategy based on the three main steps normally used for application optimization and tuning:
  1. Debug application
     - Nobody really cares how fast the program can compute the wrong answer
       - TotalView is our debugger of choice on all platforms
  2. Single processor and vector optimization
     - Make the common case fast
  3. Parallel processing and I/O optimization
     - Communication / barriers / etc.

Single Processor Optimization

- Answer the following questions:

1. Does my program have performance problems?
   - Time / Resource / Hardware Counters measurement
   - Provides overall view of the program execution
     - % pat_hwpc a.out

Single Processor Optimization (2)

- Answer the following questions:

2. Where are the main bottlenecks?
   - Sample based profiler (Flat, Call graph, Call path)
   - Event Profiler with runtime summarization

Single Processor Optimization (3)

- Answer the following questions:

3. What are the causes of performance problems?
   - Compiler feedback
     - Canal / Loopmarks
   - HW counters based Instrumentation library
     - Run-time summarization
     - Hand or automatic instrumentation

Parallel Processing Optimization

- Answer the following questions:

1. Does my program have communication or synchronization problems?
   - Communication Profiler

Parallel Processing Optimization (2)

- Answer the following questions:

2. What are the causes of parallel performance problems?
   - Task/Thread tracing
     - Tracing library
       - » CrayPat
   - Visualization GUI
     - Cray Apprentice^2

Approach

- Take advantage of existing tools infrastructure
  - Improve usability
  - Develop components for missing functionality
  - Port to all Cray platforms
- Close interaction and collaboration with tools builders in Universities, Labs, and Research Centers
- Close interaction with application developers for quick feedback, targeting functionality enhancements

Collaborations

- Interactions with lead researchers in the performance analysis and optimization field
  - Jack Dongarra's group at UTK
    - PAPI port to all Cray systems
  - Bernd Mohr at Forschungszentrum Jülich
    - KOJAK expert system ported to X1
    - Investigation of CAF and UPC measurement and analysis
    - Automatic performance analysis
  - Jeff Vetter's group at ORNL
    - MPI Profiler (mpiP) port to all Cray platforms
    - Data management techniques in performance analysis
      - Development of filters
      - Hierarchical analysis

Cray Performance Tools

- PAT_HWPC
  - Hardware Counters based measurement for whole application
- CrayPat
  - Performance Analysis Toolkit
- Cray Apprentice^2
  - Performance visualization

PAT_HWPC (X1 Series only)

- Collects hardware performance counters information for an application
- No instrumentation required
  - pat_hwpc [options] a.out
  - Accepts various hardware counters groups
- Results in report with raw counts and derived metrics for the whole execution
  - Hardware counters summed across threads per process

LIBHWPC (XT3 and XD1)

- Instrumentation library for Fortran, C, and C++
- For each instrumented region provides:
  - Total count & duration (wall clock time & user time)
  - Hardware performance counters information
  - Derived metrics
- Supports:
  - Multiple instrumentation points
  - Nested instrumentation
  - Multiple calls to an instrumented point
- Uses PAPI Version 3
  - PAPI available on base distribution

CrayPat

- Performance Analysis Toolkit
  - pat_build
    - Utility that automatically instruments the application
      - No source code modification required
  - pat runtime library
    - Transparent to the user
    - Collects performance data during execution
    - Writes performance file
  - pat_report
    - Performance analysis utility
  - pat_run
  - pat_help

pat_build

- pat_build <options> a.out instrumented.a.out
- Automatic instrumentation
  - User functions
  - MPI, SHMEM, OpenMP, UPC, CAF, pThreads
  - System calls
  - Dynamic heap
  - Raw I/O, buffered I/O, flexible file I/O
- API available for fine grain instrumentation
- "instrumented.a.out" is executed the same way as the original application
- No recompilation needed!

CrayPat Functionality: Instrumentation

- Capture of:
  - Software state
    - Thread
    - Call Stack
    - parameter values passed into a function
  - Hardware state
    - Program counter (PC)
    - Hardware performance counters (HWPC)
  - Time stamps
    - Real time clock
    - HWPC Cycle counter

Performance Collection

- Trigger mechanism:
  - Activated by external agent (sampling)
    - Timer
    - Hardware counters overflow
    - (not supported on the XT3 yet)
  - Activated internally (event tracing or synchronous)
    - Code inserted through instrumentation
- Data recording:
  - Summarized during runtime
    - Stored in the form of a profile
  - Trace records for each event
    - Stored in the form of a trace file

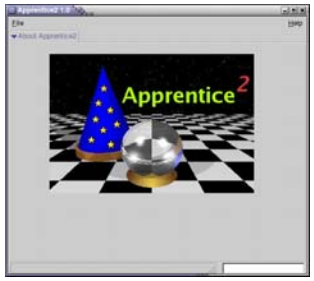
pat_report: Flat Profile

Time%	Cum.Time%	Time	Traces	Function
7.4%	93.8%	3.737219	24000	MPI_Waitall
1.4%	96.7%	0.707859	56383	MPI_Isend
0.9%	98.9%	0.455456	56383	MPI_Irecv
0.1%	99.8%	0.051986	56383	mpi_
0.1%	99.9%	0.040626	4	InitalCalc1Calc2#88
0.1%	100.0%	0.026122	24000	mpi_waitall_
0.0%	100.0%	0.009164	4800	Calc1Calc2#10
0.0%	100.0%	0.007347	4800	Calc2#20
0.0%	100.0%	0.006329	4792	Calc3#30
0.0%	100.0%	0.001980	60	MPI_Reduce
0.0%	100.0%	0.000083	60	mpi_reduce_
0.0%	100.0%	0.000030	60	MPI_Type_get_true_extent
0.0%	100.0%	0.000016	4	main
0.0%	100.0%	0.000011	4	MPI_Finalize
0.0%	100.0%	0.000005	4	mpi_comm_rank_
0.0%	100.0%	0.000005	4	mpi_comm_size_
0.0%	100.0%	0.000003	4	MPI_Comm_size
0.0%	100.0%	0.000003	4	_Exit
0.0%	100.0%	0.000003	4	MPI_Comm_rank
0.0%	100.0%	0.000003	4	MPI_Init

pat_report Call Tree Profile

```
Time% | Cum.Time% |      Time | Traces |Calltree

 100.0% |    100.0% | 50.313549 | 316932 |Total
|-----------------------------------------------
|  98.5% |     98.5% | 49.541495 | 316864 |main
||----------------------------------------------
||  98.5% |     98.5% | 49.541465 | 316852 |MAIN_
|||---------------------------------------------
|||  39.0% |     39.0% | 19.645725 | 115200 |calc2_
||||--------------------------------------------
||||  36.4% |     36.4% | 18.336529 |   4800 |pat_region_begin_
||||        |           |           |        | Do 200#21
||||   1.1% |     37.6% |  0.559553 |  19200 |mpi_waitall_
|||||-------------------------------------------
|||||   1.1% |     37.5% |  0.550431 |   9600 |MPI_Waitall
|||||   0.0% |     37.6% |  0.009122 |   9600 |mpi_waitall_(exclusive)
|||||===========================================
||||   0.6% |     38.2% |  0.300126 |  43200 |mpi_isend_
|||||-------------------------------------------
|||||   0.6% |     38.1% |  0.280955 |  21600 |MPI_Isend
|||||   0.0% |     38.2% |  0.019171 |  21600 |mpi_isend_(exclusive)
|||||===========================================
||||   0.5% |     38.6% |  0.246678 |   4800 |calc2_(exclusive)
||||   0.4% |     39.0% |  0.202839 |  43200 |mpi_irecv_
|||||-------------------------------------------
|||||   0.4% |     39.0% |  0.180679 |  21600 |MPI_Irecv
|||||   0.0% |     39.0% |  0.022161 |  21600 |mpi_irecv_(exclusive)
||||===========================================
|||  29.5% |     68.5% | 14.844131 | 110400 |calc1_
||||--------------------------------------------
||||  23.7% |     62.8% | 11.933617 |   4800 |pat_region_begin_
||||        |           |           |        | Do 100#11
||||   4.5% |     67.3% |  2.282388 |  19200 |mpi_waitall_
|||||-------------------------------------------
```

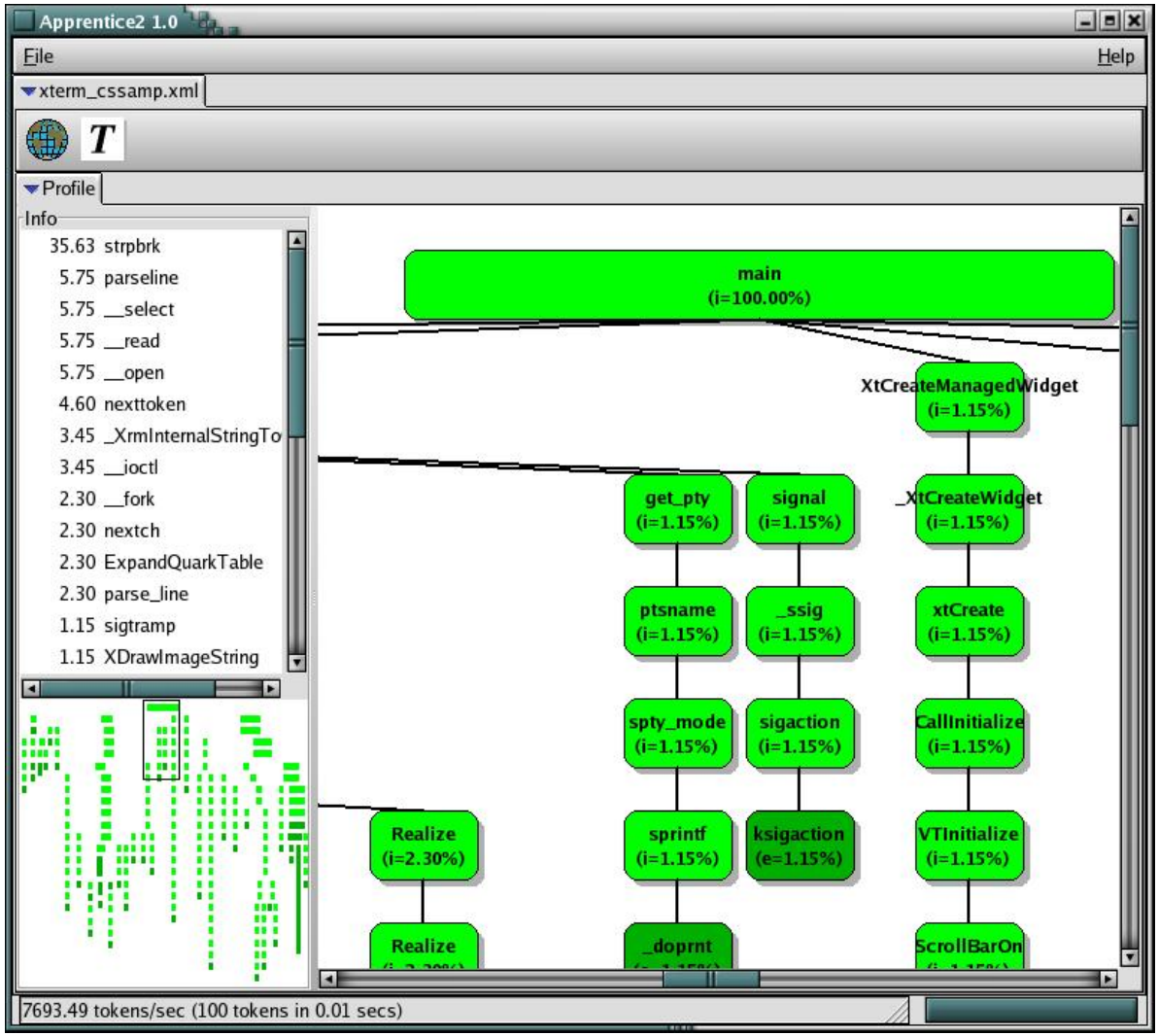
Cray Apprentice^2



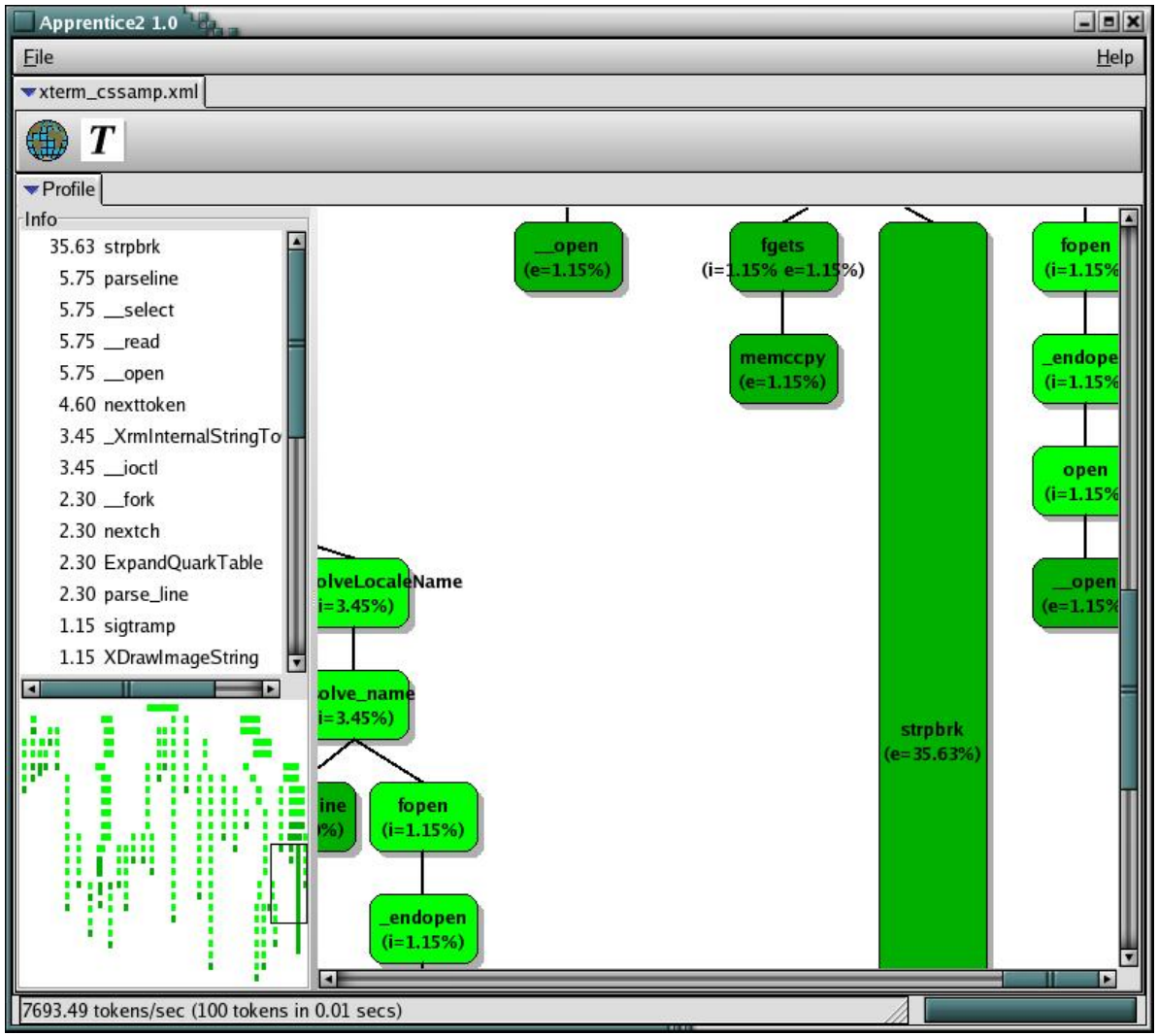
- Call graph profile
- Communication statistics
- Time-line view
  - Communication
  - I/O
- Activity view
- Pair-wise communication statistics
- Text reports
- Source code mapping

- Target to help identify and correct:
  - Load imbalance
  - Excessive serialization
  - Excessive communication
  - Network contention
  - Poor use of the memory hierarchy
  - Poor functional unit use

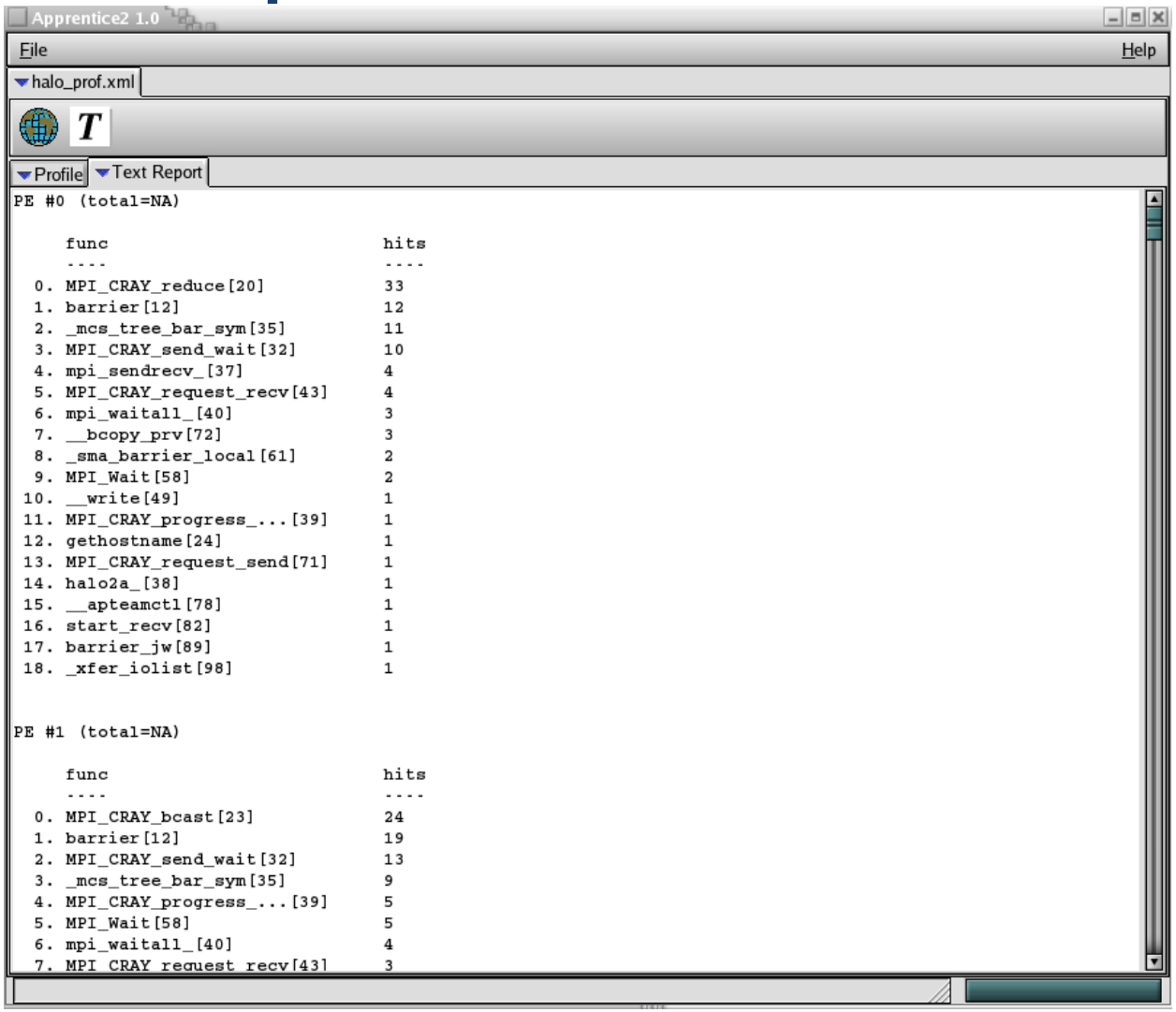
Call Graph Profile View



Call Graph Profile View

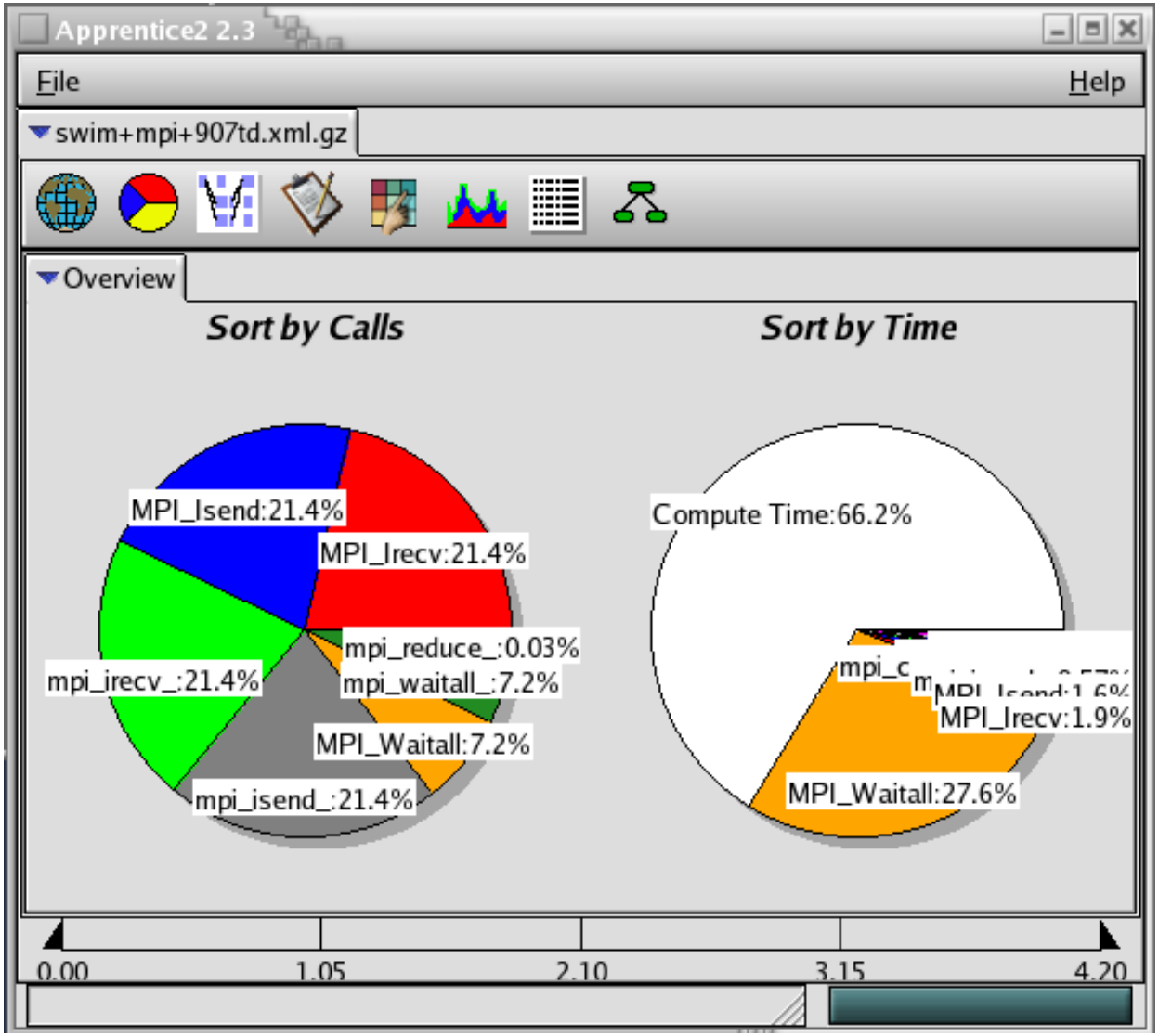


Call Graph Profile View

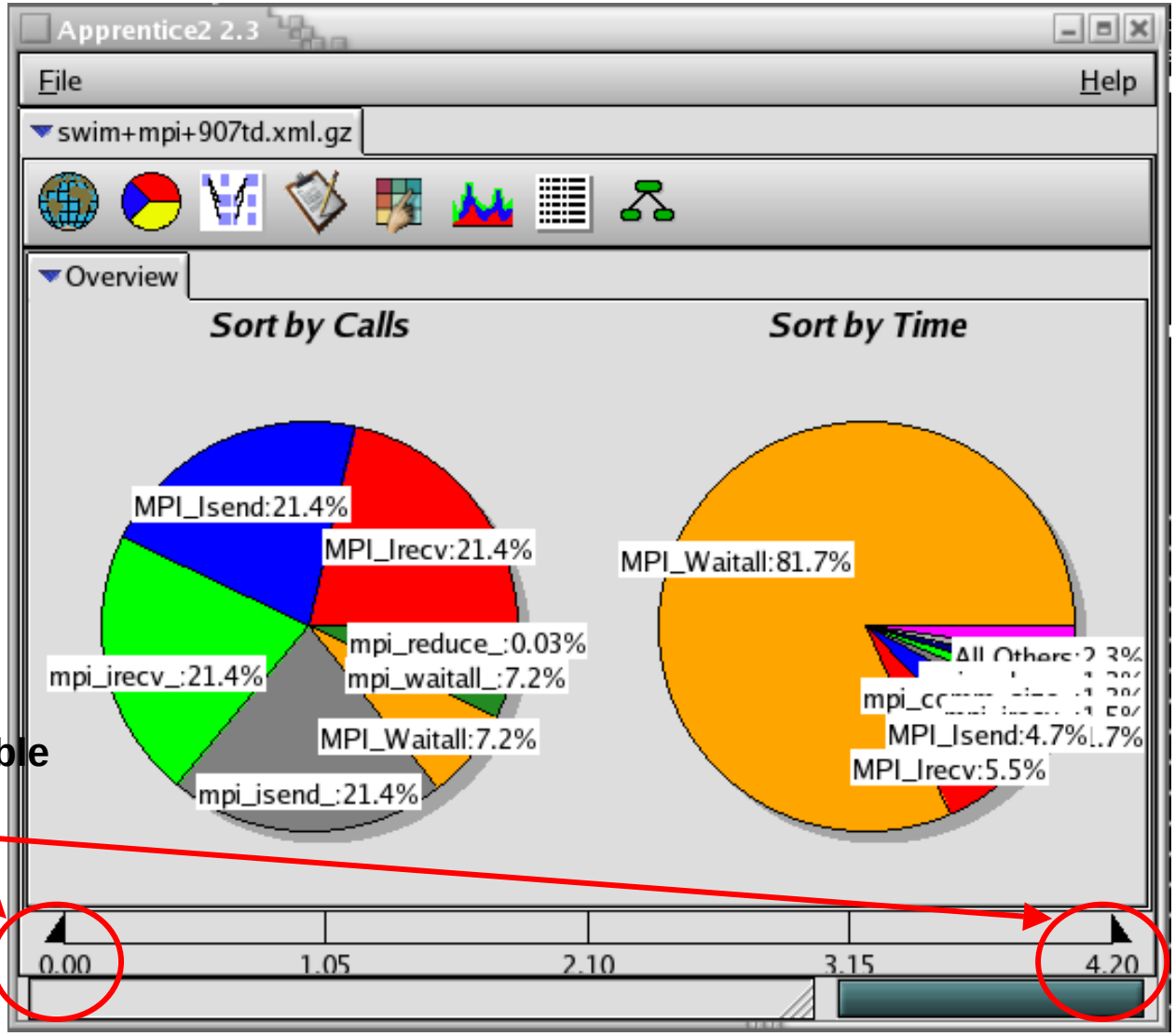



```
PE #0 (total=NA)

     func                          hits
     ----                          ----
  0. MPI_CRAY_reduce[20]           33
  1. barrier[12]                   12
  2. _mcs_tree_bar_sym[35]         11
  3. MPI_CRAY_send_wait[32]        10
  4. mpi_sendrecv_[37]             4
  5. MPI_CRAY_request_recv[43]     4
  6. mpi_waitall_[40]              3
  7. __bcopy_prv[72]               3
  8. _sma_barrier_local[61]        2
  9. MPI_Wait[58]                  2
 10. __write[49]                   1
 11. MPI_CRAY_progress_...[39]     1
 12. gethostname[24]               1
 13. MPI_CRAY_request_send[71]     1
 14. halo2a_[38]                   1
 15. __apteamctl[78]               1
 16. start_recv[82]                1
 17. barrier_jw[89]                1
 18. _xfer_iolist[98]              1


PE #1 (total=NA)

     func                          hits
     ----                          ----
  0. MPI_CRAY_bcast[23]            24
  1. barrier[12]                   19
  2. MPI_CRAY_send_wait[32]        13
  3. _mcs_tree_bar_sym[35]         9
  4. MPI_CRAY_progress_...[39]     5
  5. MPI_Wait[58]                  5
  6. mpi_waitall_[40]              4
  7. MPI_CRAY_request_recv[43]     3
```

Communication Statistics

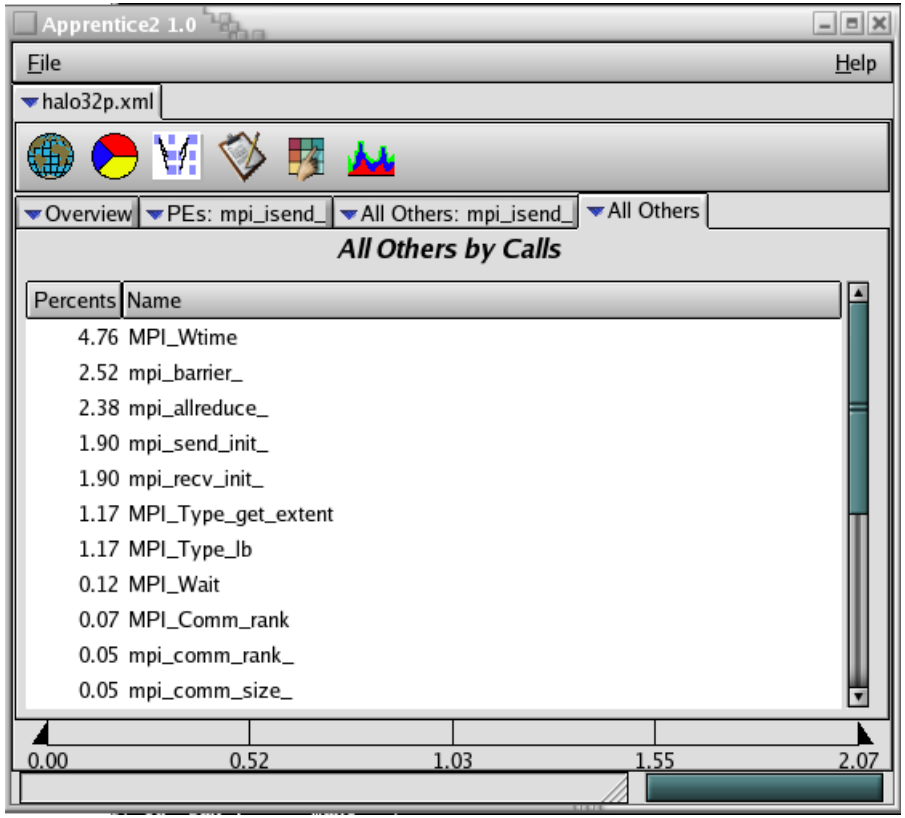


Communication Statistics



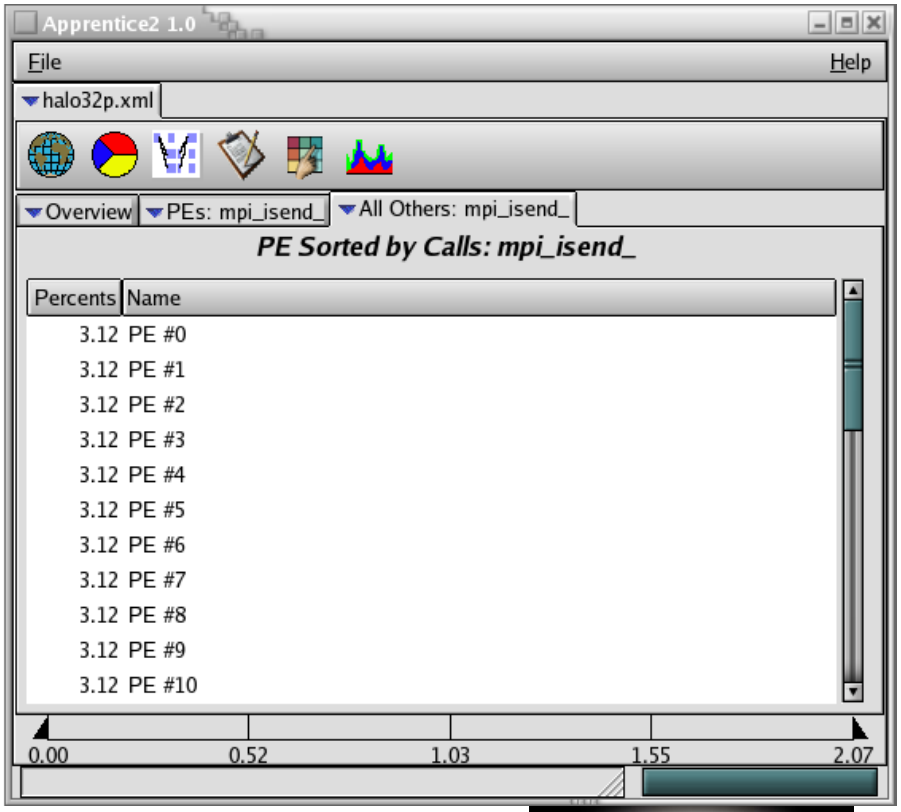
Adjustable calipers

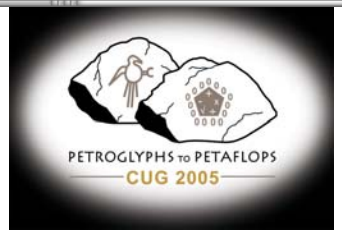
Communication Statistics



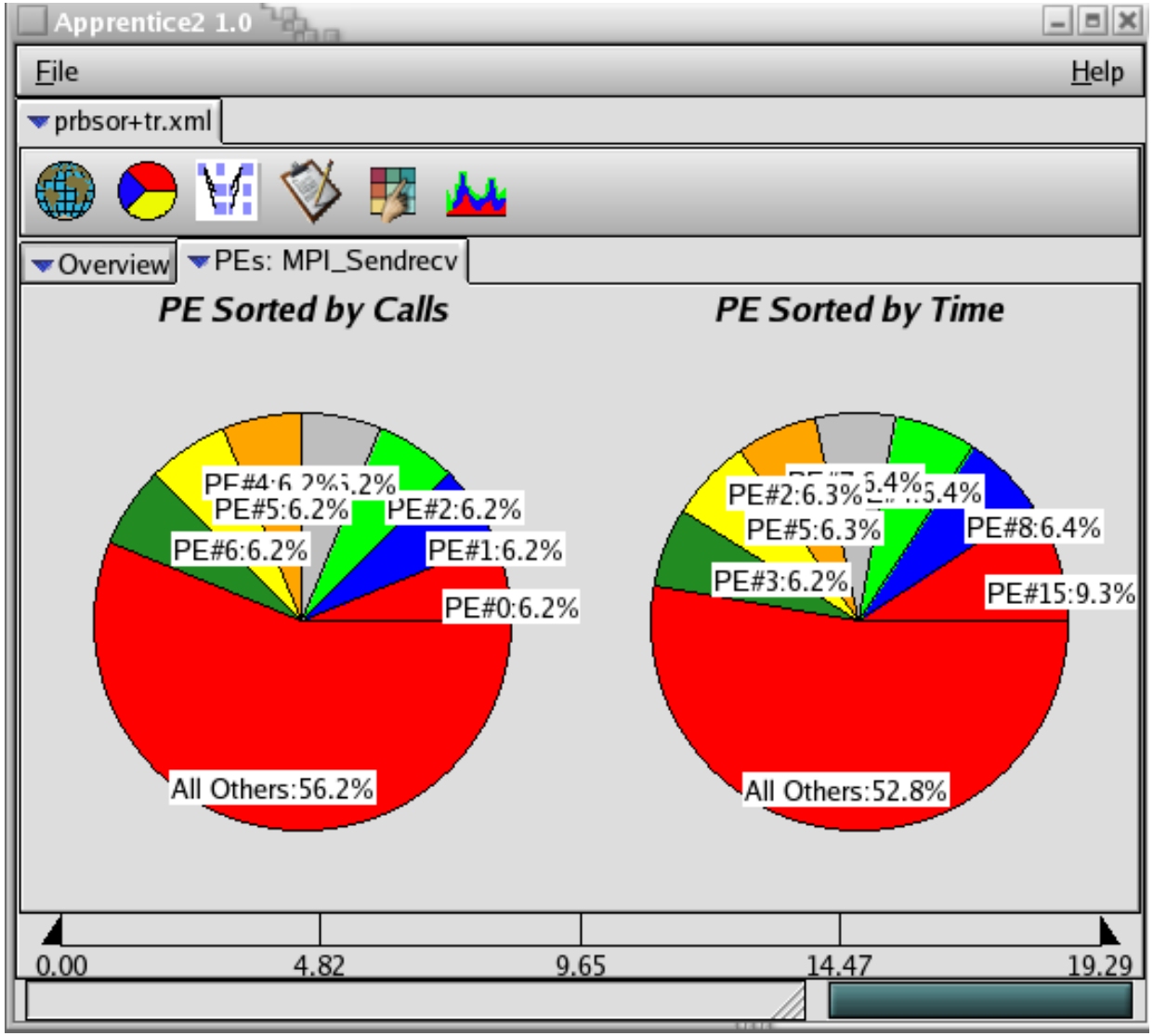
All Others %

PE Distribution (mpi_isend)



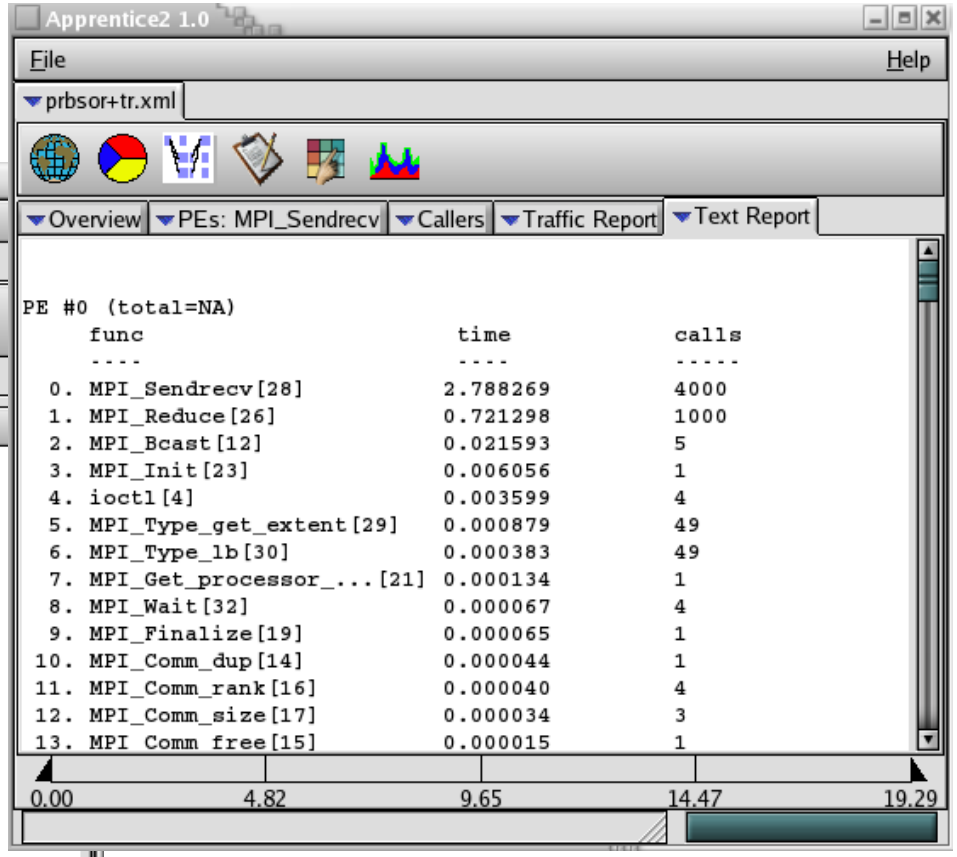


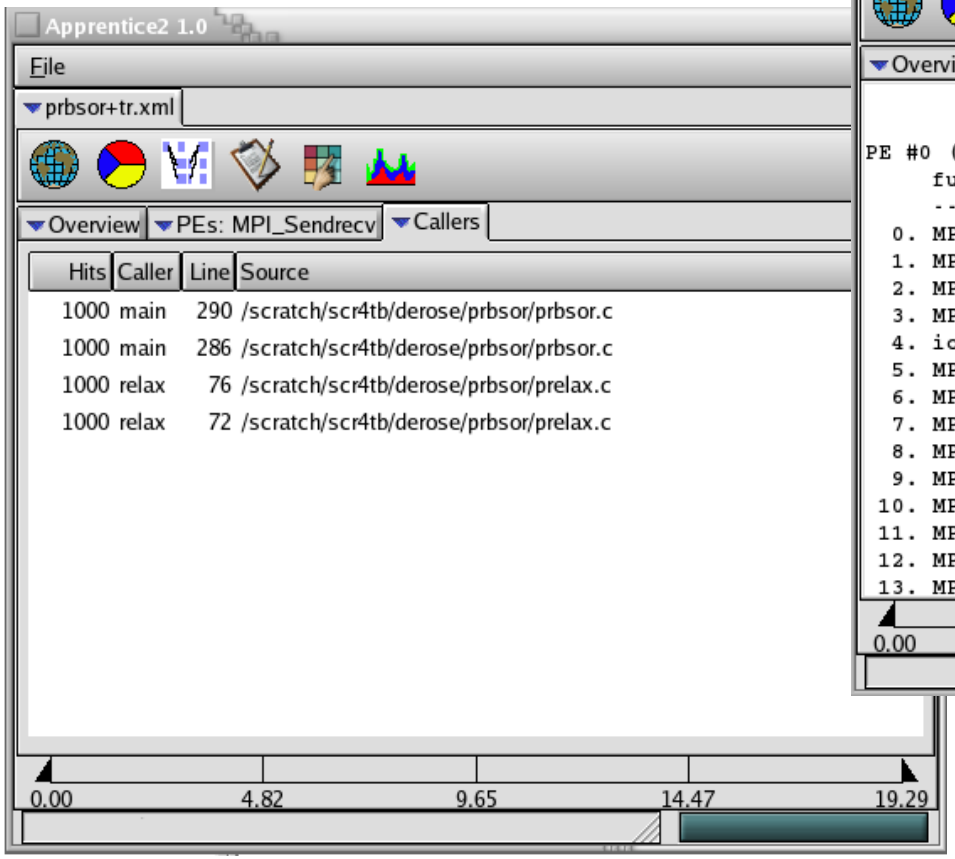
Statistics by PE



Statistics by PE

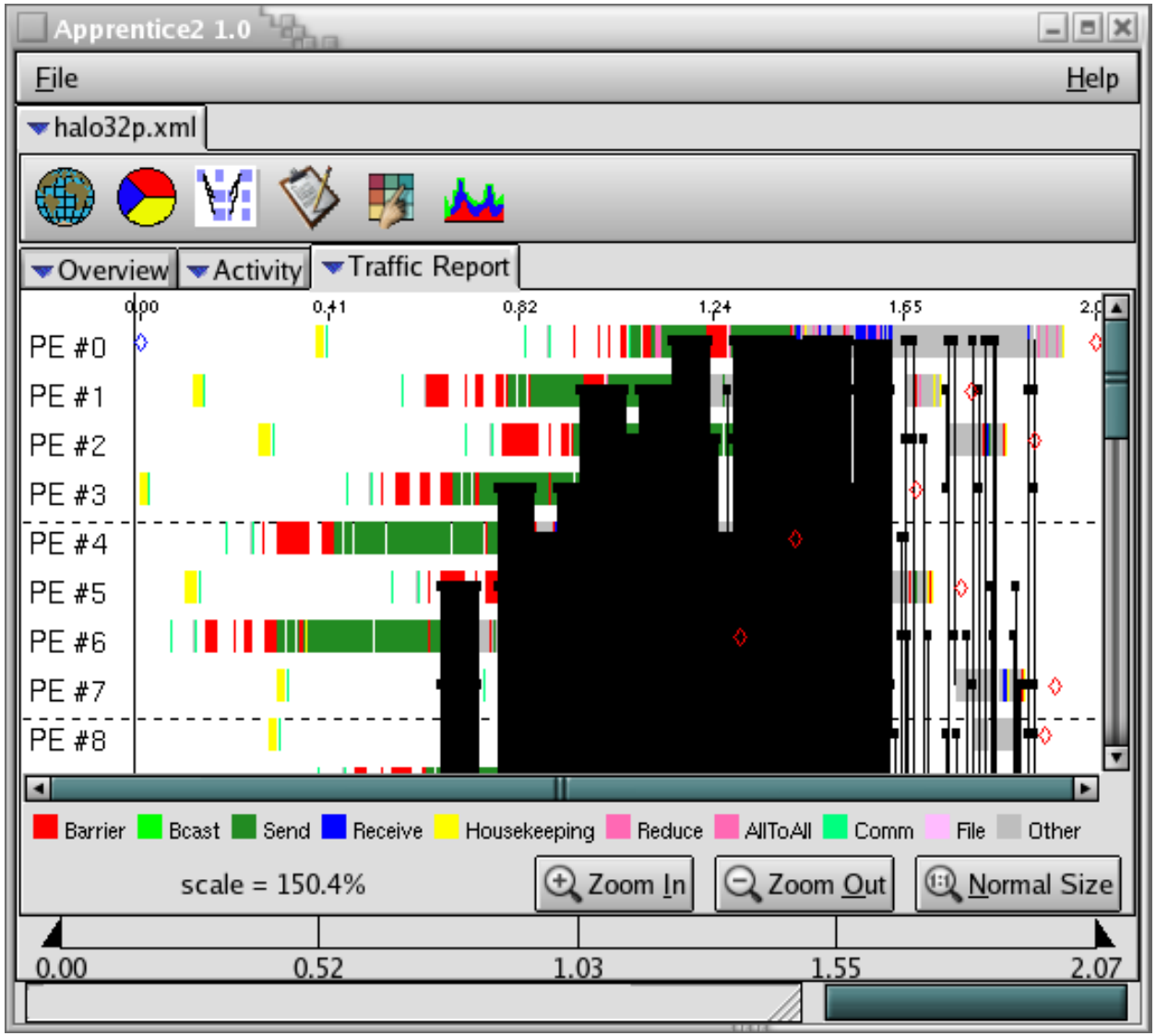
Profile



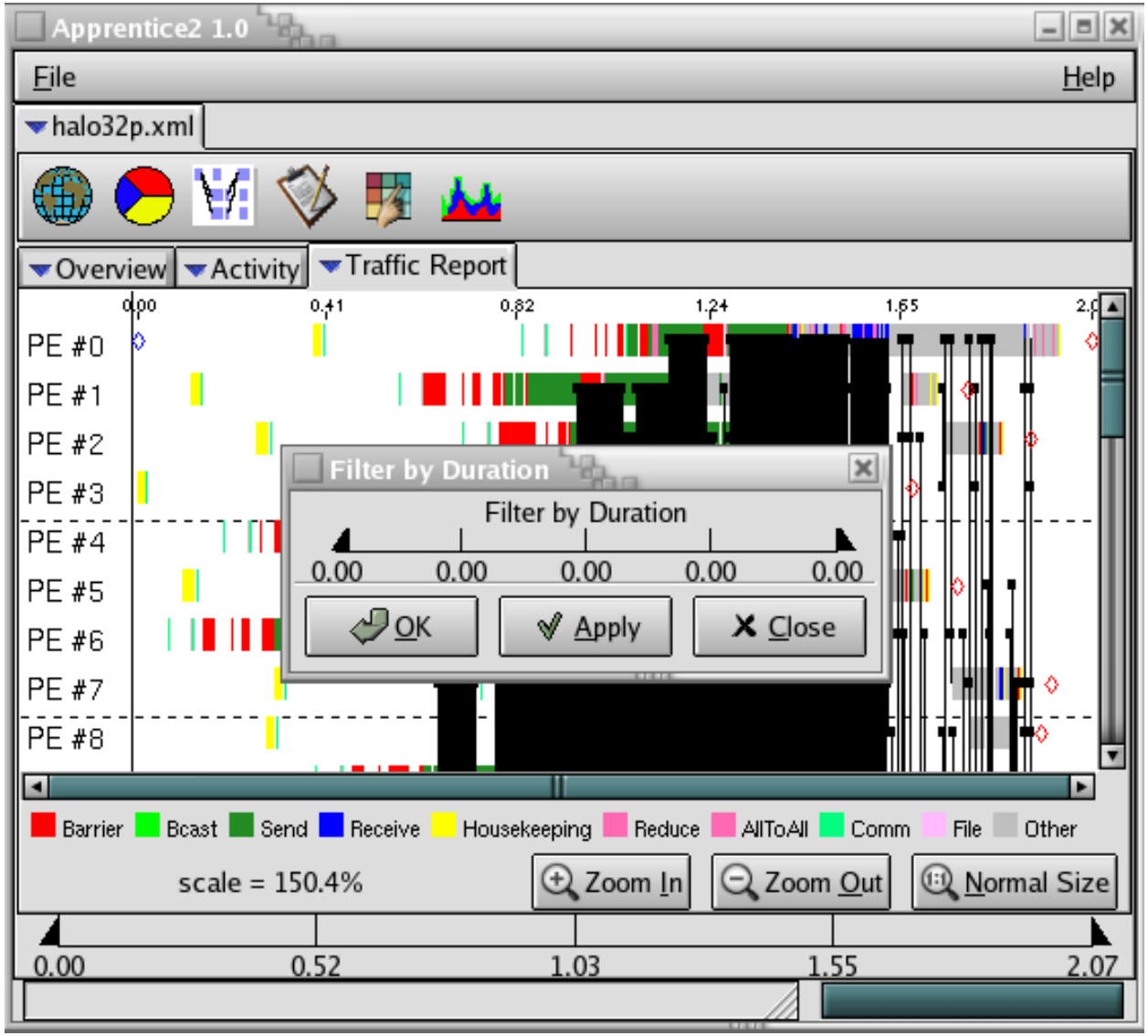


Source code mapping

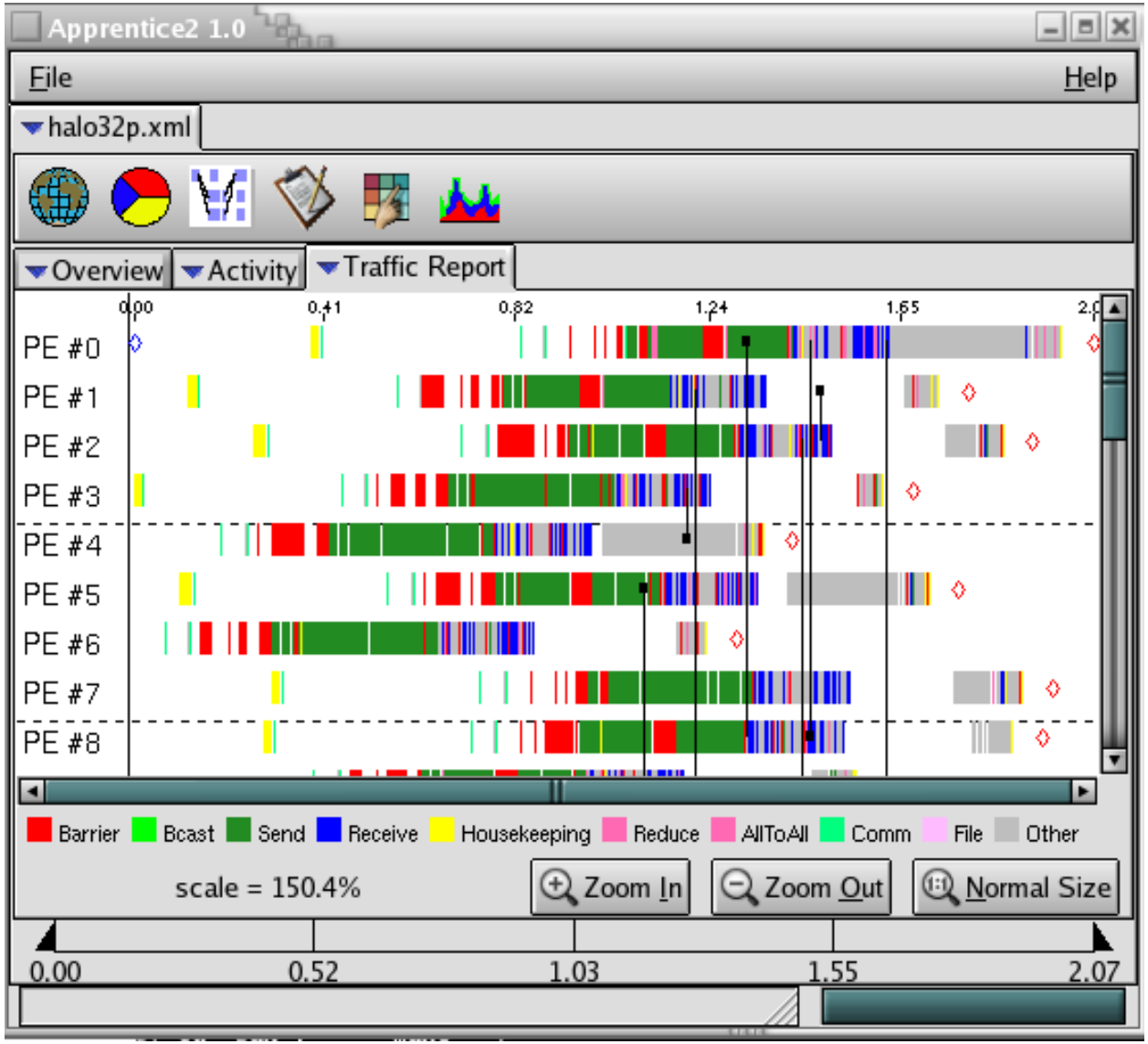
Timeline View



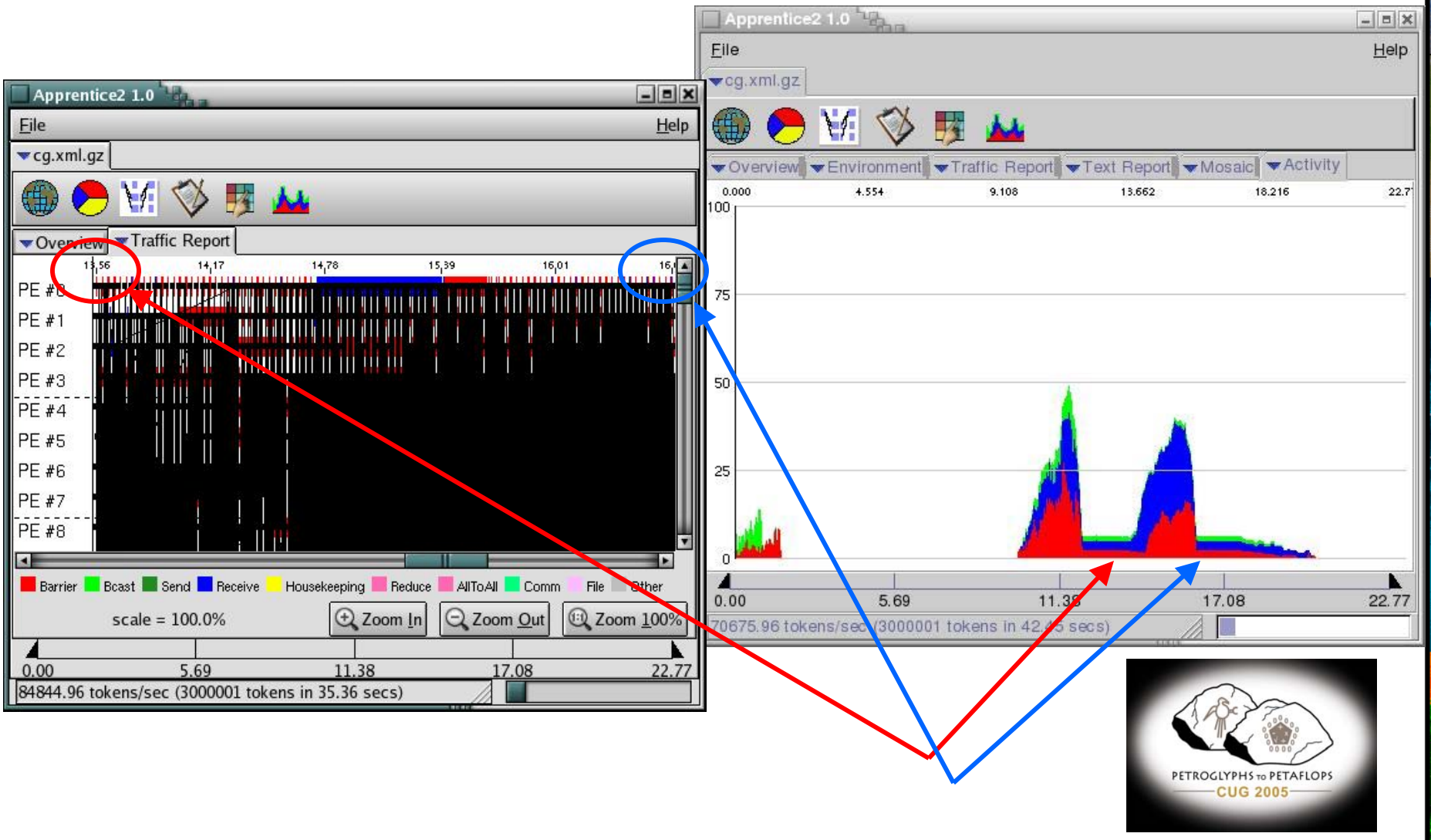
Timeline View



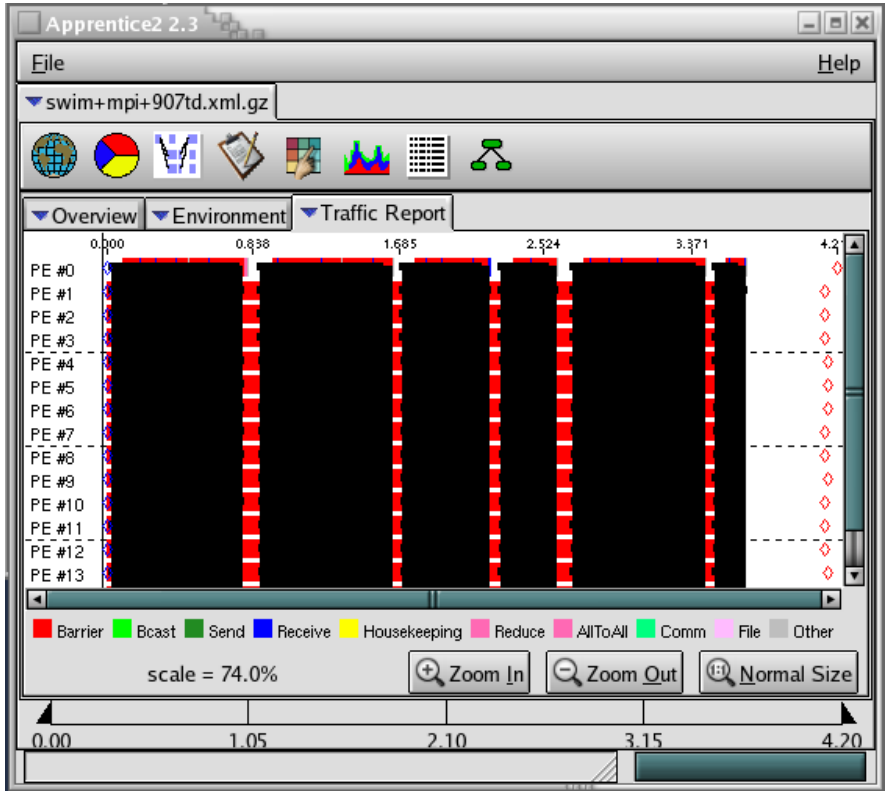
Timeline View

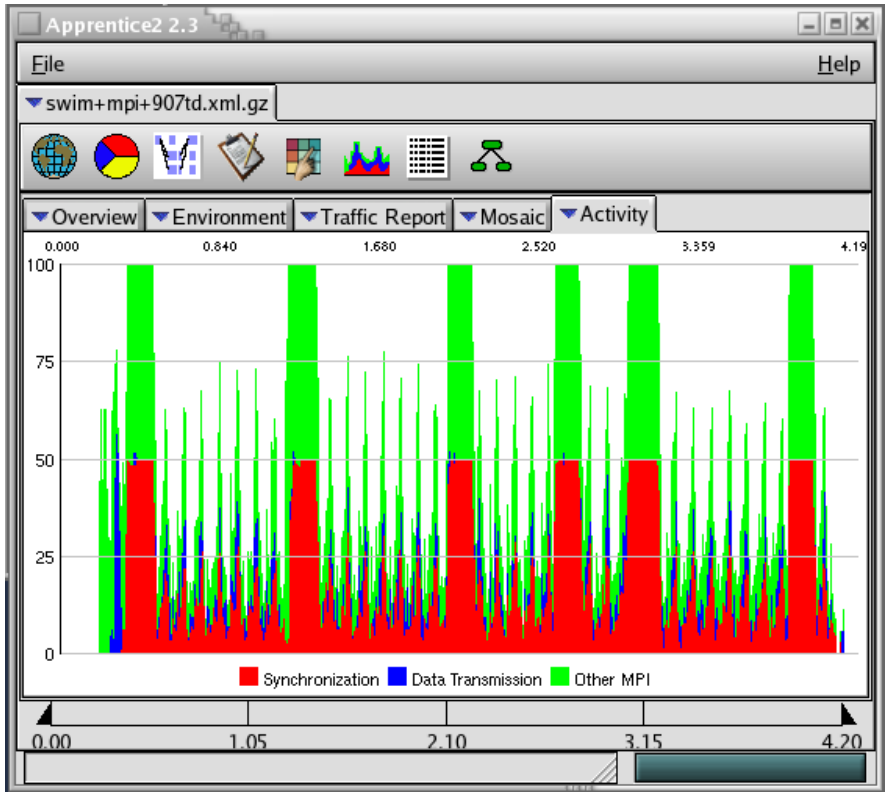


Activity Panel

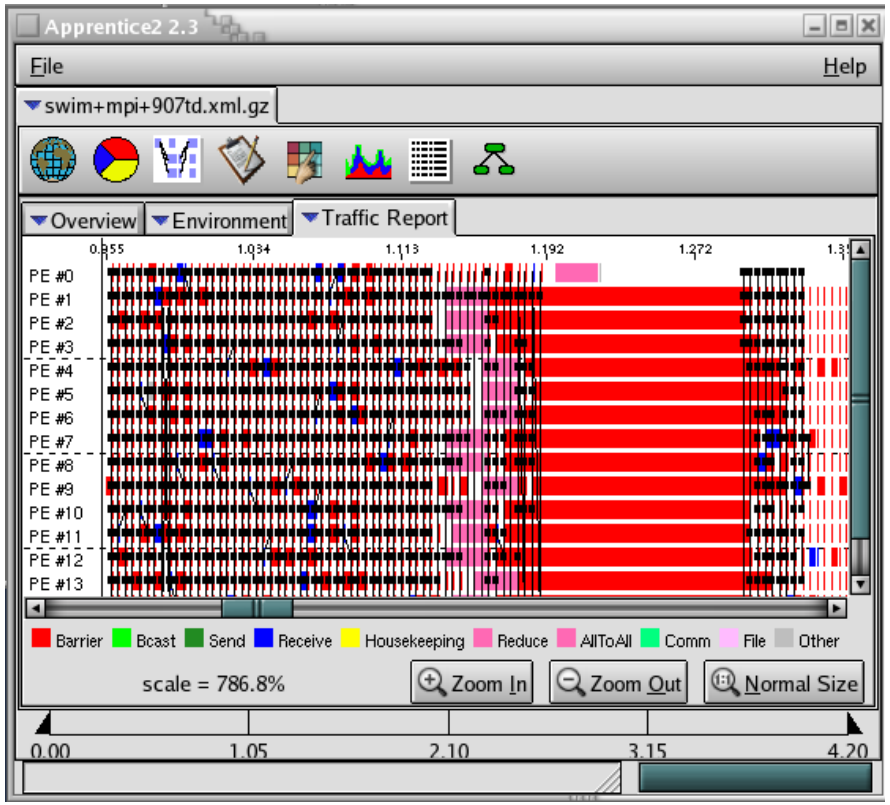


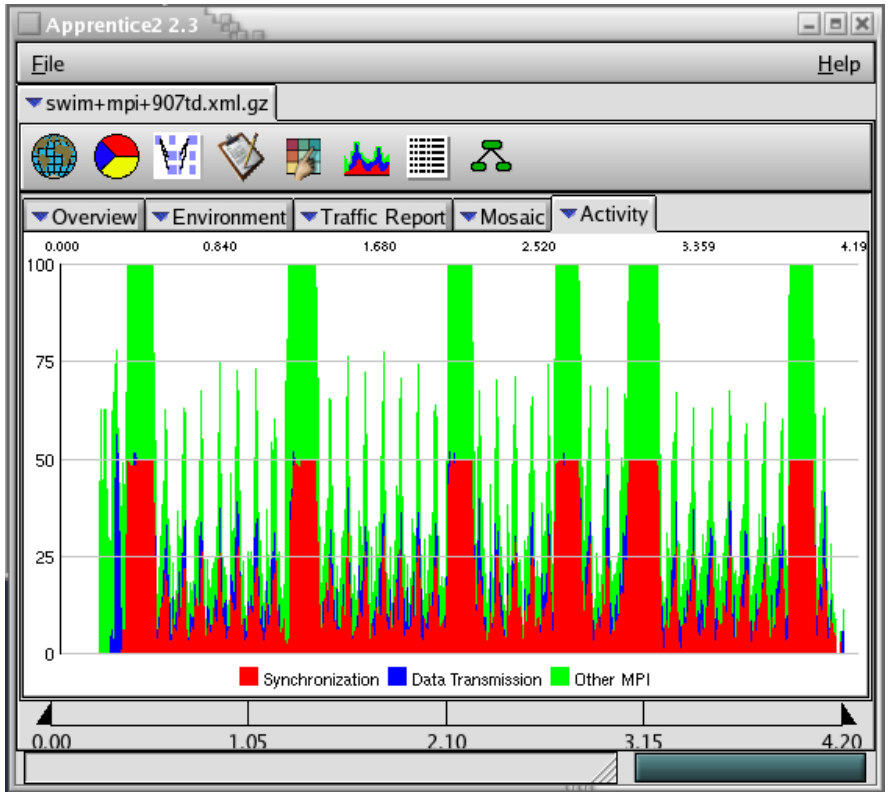
Timeline View



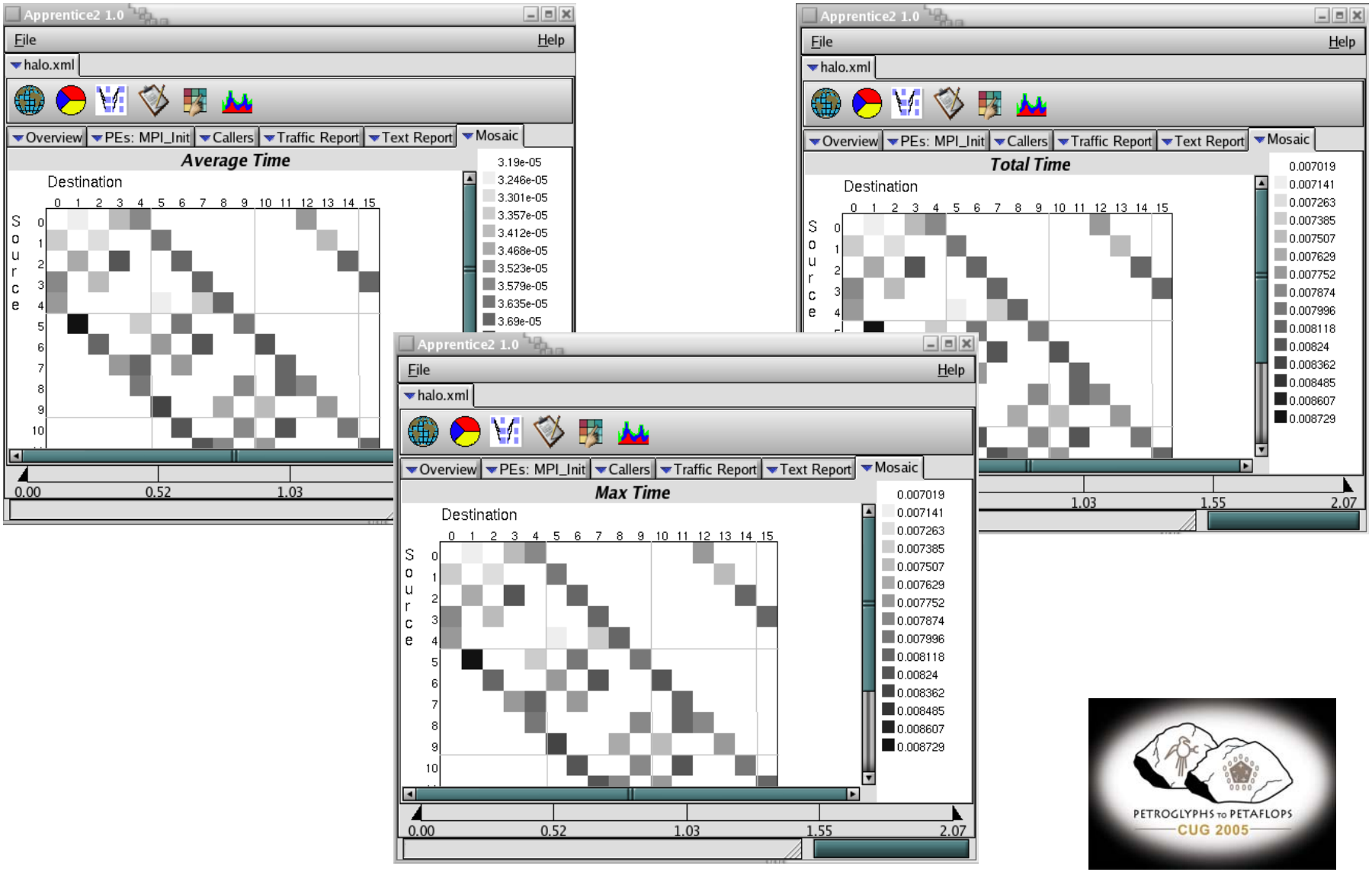


Timeline View

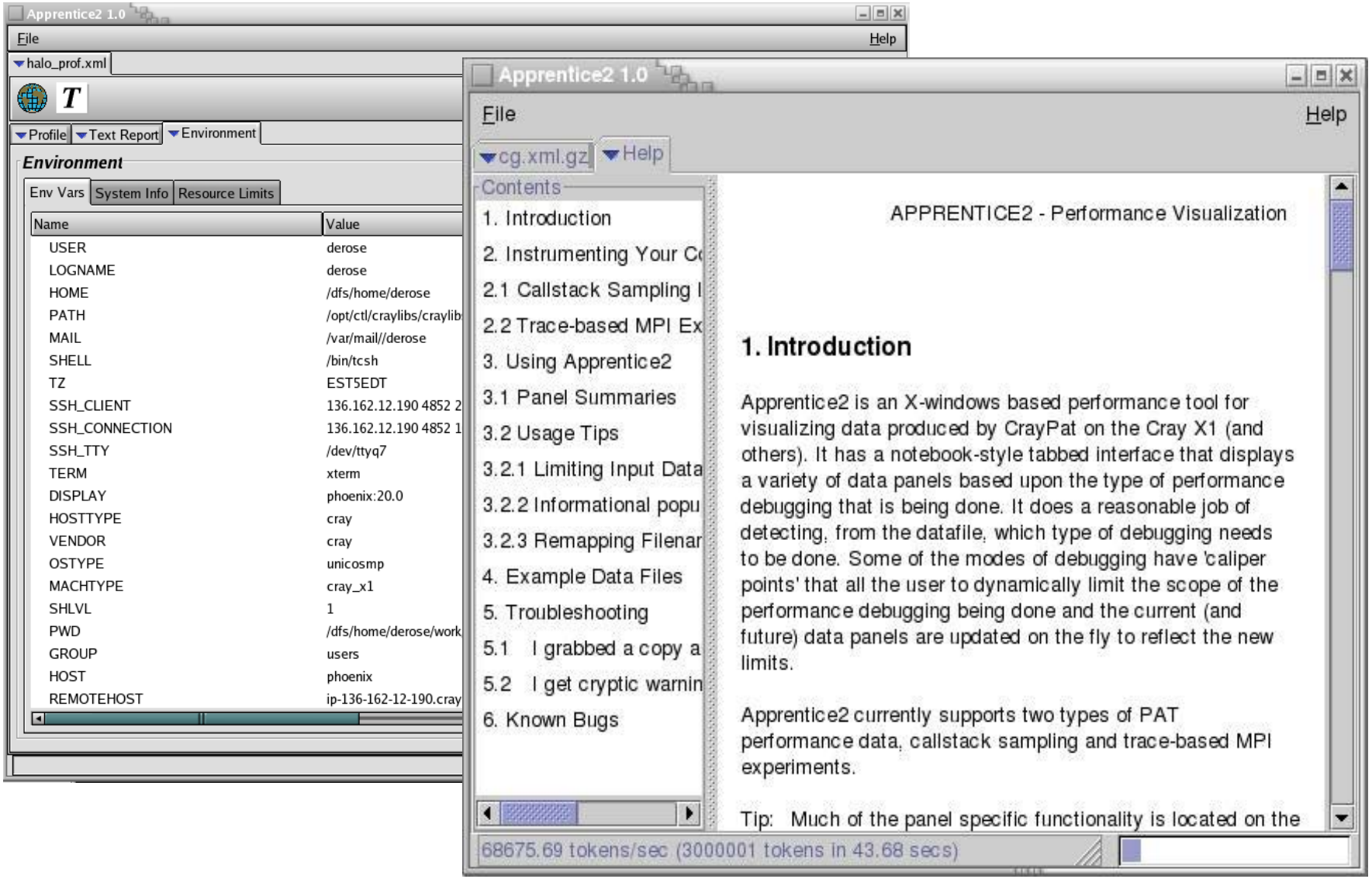




Pair-wise Statistics



Run Information and On-line Help



New Features Summary

- Improved usability of Cray PAT on the X1 Series
  - Cray Apprentice^2
  - pat_run
  - Linux Cross Compiler environment support
  - Improved performance of trace generation
  - Runtime summarization
  - Better OpenMP support
- Cray XT3 and XD1
  - Port of Cray PAT to both platforms
    - Beta available by the end of the year
  - Cray Apprentice^2

Conclusions

- Previous status
  - Good infrastructure
    - Lacking user interface
    - Lacking support on all platforms
- Current status
  - State of the art tools on all Cray systems
    - Functionality to answer all questions in the performance tuning and optimization cycle
- Near future
  - Best performance analysis tools in the industry

The New Generation of Cray Performance Tools

Questions? / Comments!

Thank You!

Luiz DeRose

ldr@cray.com