Parallel Performance Analysis and Visualization on the Cray XT

Luiz DeRose
Programming Environment Director
Cray Inc.
ldr@cray.com

University of Bergen
Bergen, Norway
March 11-14, 2008

Detecting Application Load Imbalance

- Current trend in high end computing is to have systems with tens of thousands of processors
  - This is being accentuated with multi-core processors
- Applications have to be very well balanced In order to perform at scale on these MPP systems
- Very few performance tools focus on load imbalance
  - Need standard metrics
  - Need intuitive way of presentation

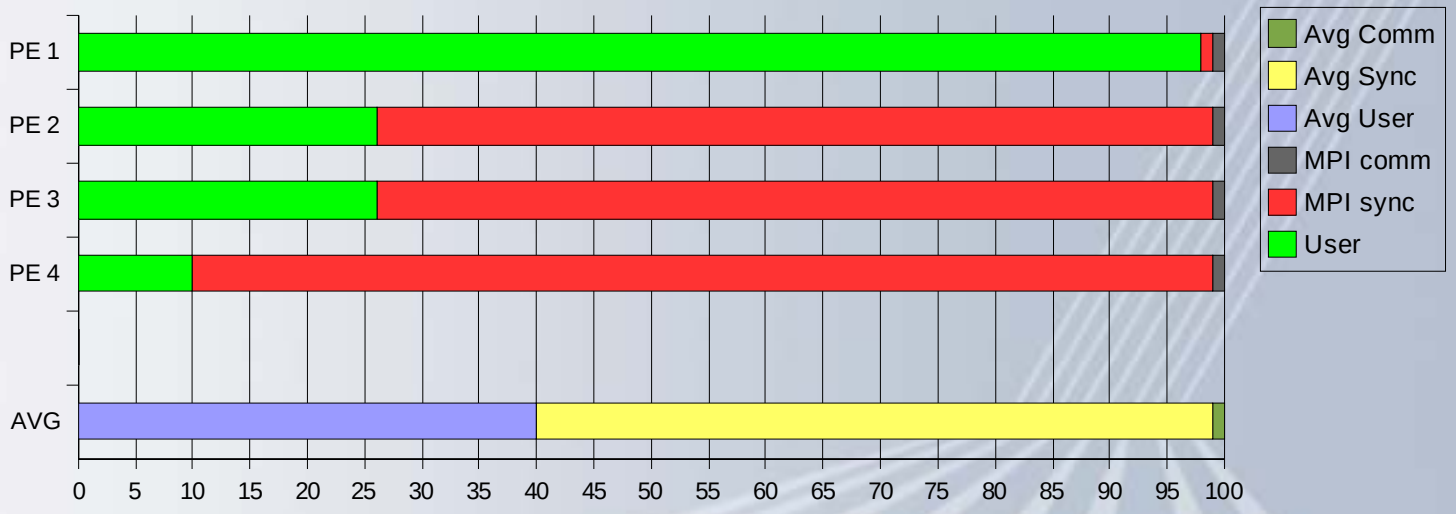
Load balance metric - rationale

Between two barriers

User: Imb = Max-Avg = 99-40 = 59

MPI Sync: Avg = 59

MPI Sync+Comm: Avg-Min = 60-1 = 59



Imbalance Time

- We define an imbalance metric, based on execution times, to identify computational code regions that could benefit most from optimization of load balance:

  Imbalance time = Max Time – Avg time

- The imbalance time provides an estimation to the user of how much time in the overall program would be saved if the corresponding section of the code had a perfect balance
  - Represents an upper bound on the “potential saving”
    - Assumes that other PEs are simply waiting without doing any useful work while the slowest member finishes

Imbalance %

- Goal is to provide an idea of the “badness” of the imbalance
  - Defined to be in the range of 0 to 100
    - A perfectly balance code segment would have 0 imbalance %
    - Serial portion of code segment would have imbalance% of 100

$$\text{Imbalance\%} = 100 \times \frac{\text{Max Time} - \text{Avg time}}{\text{Max Time}} \times \frac{N}{N - 1}$$

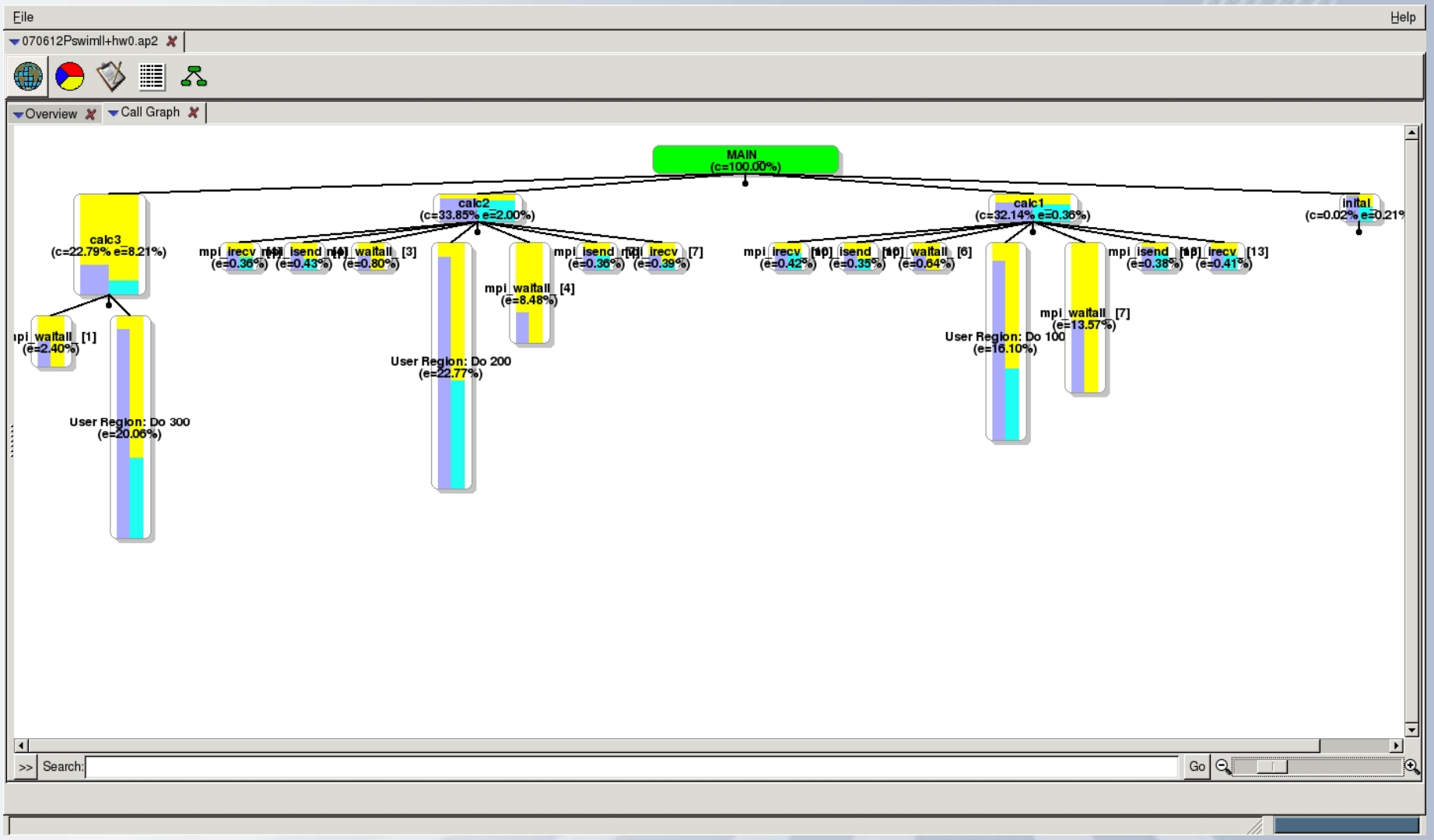
- The imbalance% corresponds to the percentage of the time that the rest of the team, excluding the slowest PE is not engaged in useful work on the given function
  - “Percentage of resources available for parallelism” that is wasted

Profile with Load Distribution by Groups

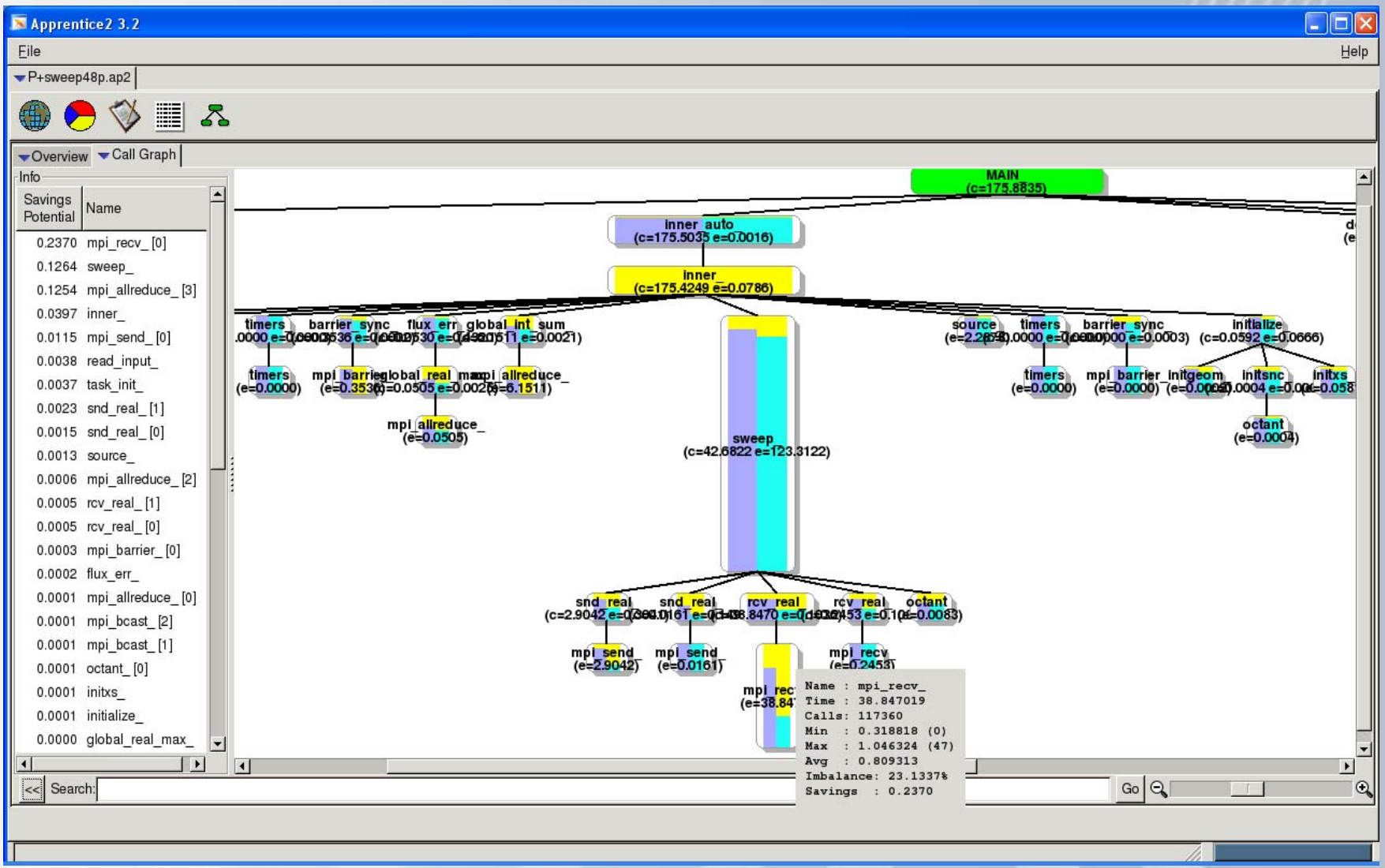
Table 1: Profile by Function Group and Function

Time %	Time	Imb. Time	Imb. Time %	Calls	Group Function PE='HIDE'
100.0%	4.609555	--	--	11947	Total
72.5%	3.340820	--	--	5112	USER
97.8%	3.267208	0.045394	1.4%	12	sweep_
1.5%	0.049875	0.000763	1.5%	12	source_
0.3%	0.009005	0.000226	2.5%	12	flux_err_
0.2%	0.007557	0.000917	11.0%	2460	snd_real_
0.1%	0.003155	0.000552	15.2%	2460	rcv_real_
22.9%	1.053745	--	--	4963	MPI
94.2%	0.992757	0.287228	22.9%	2460	mpi_recv_
5.6%	0.058613	0.010351	15.3%	2460	mpi_send_
0.2%	0.002107	0.000663	24.5%	32	mpi_allreduce_
4.5%	0.205414	--	--	39	MPI_SYNC
81.5%	0.167507	0.183974	53.5%	32	mpi_allreduce_(sync)
13.7%	0.028122	0.000784	2.8%	3	mpi_barrier_(sync)
4.8%	0.009785	0.000263	2.7%	4	mpi_bcast_(sync)
0.1%	0.004985	--	--	1825	HEAP
58.2%	0.002901	0.001326	32.0%	914	malloc
38.4%	0.001914	0.000685	26.9%	910	free
3.4%	0.000169	0.000001	0.7%	1	calloc
0.1%	0.004591	--	--	8	IO
89.3%	0.004098	0.036538	91.8%	6	fwrite
7.2%	0.000331	0.015561	100.0%	0	fputc
3.2%	0.000149	0.007006	100.0%	1	getc
0.2%	0.000009	0.000400	100.0%	0	fopen
0.1%	0.000003	0.000156	100.0%	0	fclose

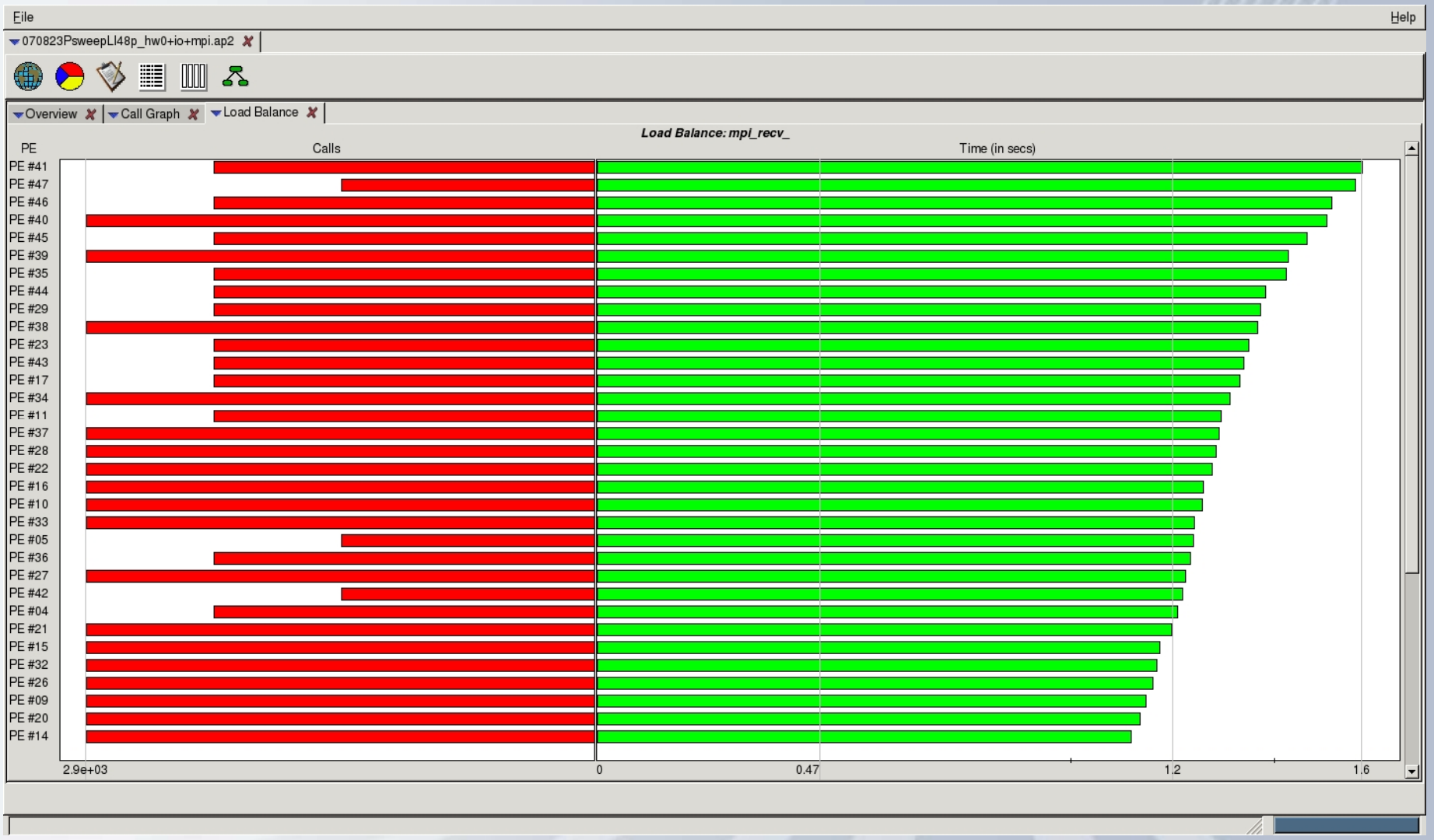
Call Tree Visualization



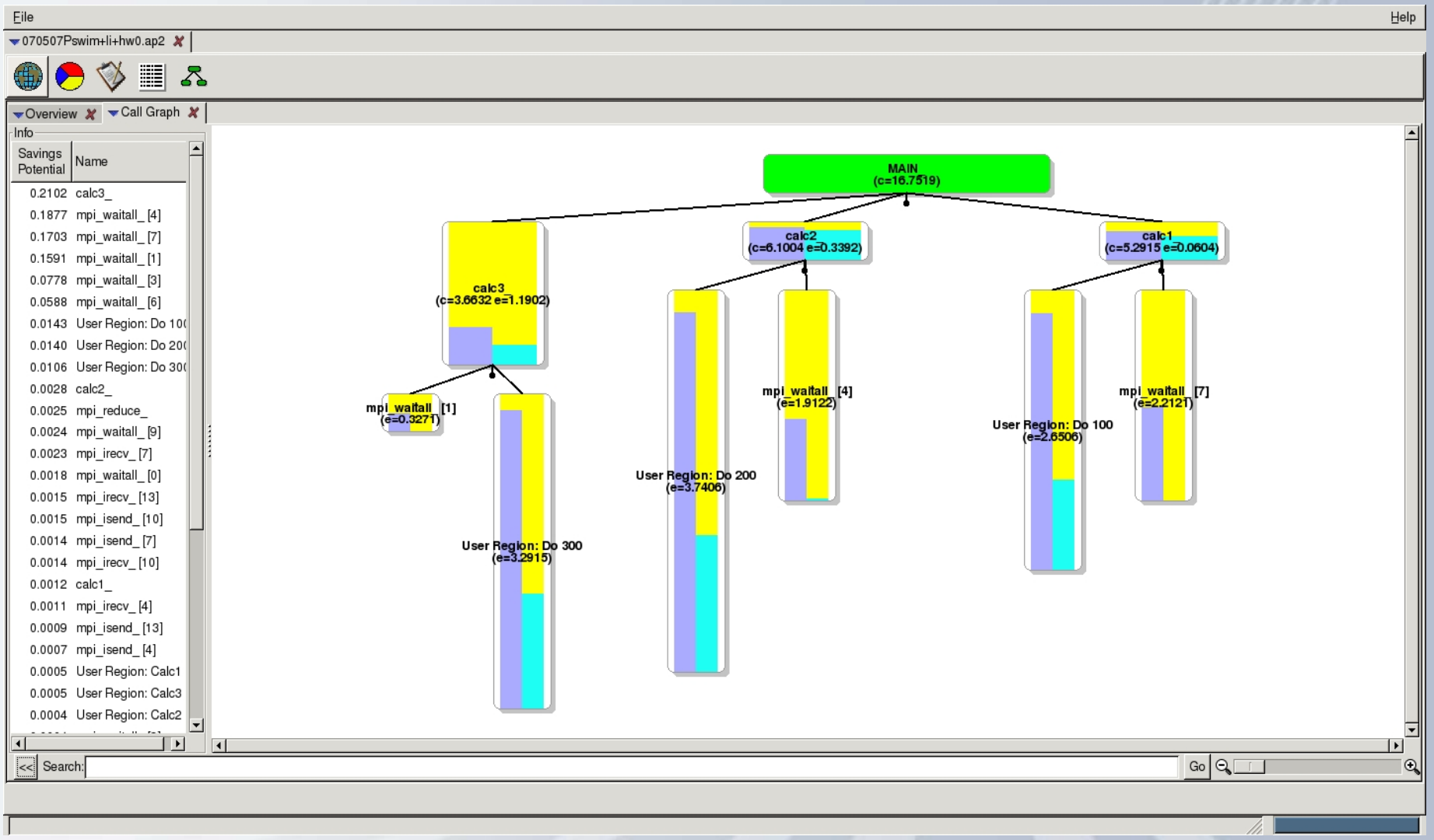
Load Imbalance Detection



Load Distribution

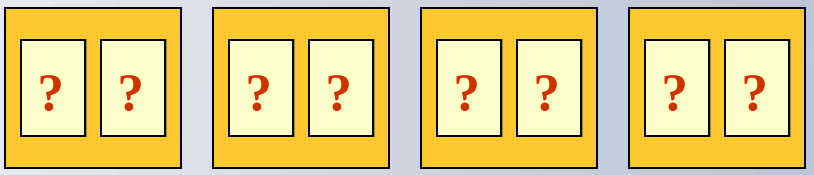


Example: Swim Benchmark



MPI Rank Reorder

- MPI rank placement with environment variable



  - Distributed placement
  - SMP style placement
  - Folded rank placement
  - User provided rank file
- Performance improvements depend on application
  - DLPOLY – SMP style – 13% gain (512pes)
  - HPCC(PTRANS) – custom placement -> 2x gain (10000pes)
  - Rank reordering can also improve performance on imbalanced codes

Rank Reorder Example - hycom

```
pat_report -O load_balance

Table 2:  Load Balance across PE's by FunctionGroup

  Time % |   Cum. |       Time |      Calls |Group
         | Time % |            |            | PE[mmm]

  100.0% | 100.0% | 482.705844 | 7446623155 |Total
|-----------------------------------------------------
|  57.7% |  57.7% | 278.657370 | 7329740077 |USER
||----------------------------------------------------
||   0.5% |   0.5% | 361.310805 |   33130409 |pe.201
||   0.4% |  58.2% | 311.898417 |   34020074 |pe.45
||   0.0% | 100.0% |  23.780267 |     320096 |pe.184
||====================================================
|  42.3% | 100.0% | 204.048383 |  116783478 |MPI
||----------------------------------------------------
||   0.9% |   0.9% | 476.662251 |     399087 |pe.184
||   0.3% |  61.9% | 167.921814 |     422197 |pe.37
||   0.2% | 100.0% | 119.123503 |     514637 |pe.201
|====================================================
```

Rank Reorder Example - hycom

```
pat_report -O load_balance -s pe=ALL

Table 2:  Load Balance across PE's by FunctionGroup

  Time % |   Cum. |       Time |      Calls |Group
         | Time % |            |            | PE

  100.0% | 100.0% | 482.705844 | 7446623155 |Total
|-----------------------------------------------
|  57.7% |  57.7% | 278.657370 | 7329740077 |USER
||----------------------------------------------
||   0.5% |   0.5% | 361.310805 |   33130409 |pe.201
||   0.5% |   1.0% | 349.849557 |   30460022 |pe.182
||   0.5% |   1.5% | 346.919713 |   33685730 |pe.200
||   0.5% |   2.0% | 342.844256 |   34879988 |pe.188
||   0.5% |   2.5% | 342.308415 |   34913960 |pe.172
. . .
||   0.1% |  99.8% |  45.464691 |    3000260 |pe.248
||   0.1% |  99.9% |  35.970972 |     399401 |pe.213
||   0.0% |  99.9% |  27.431543 |     340673 |pe.232
||   0.0% | 100.0% |  25.142167 |     117620 |pe.240
||   0.0% | 100.0% |  23.780267 |     320096 |pe.184
||==============================================
```

Rank Reorder Example - hycom

After custom placement (10% performance improvement):

```
Table 2:  Load Balance with MPI Sent Message Stats

  Time % |       Time | Sent Msg |     Sent Msg | Avg Sent |Group
         |            |    Count |  Total Bytes | Msg Size | PE[mmm]

  100.0% | 437.418783 | 17161829 | 289328285840 | 16858.83 |Total
|-------------------------------------------------------------------
|  60.2% | 263.211966 |       -- |           -- |       -- |USER
||------------------------------------------------------------------
||   0.5% | 322.019049 |       -- |           -- |       -- |pe.158
||   0.4% | 286.179471 |       -- |           -- |       -- |pe.126
||   0.0% |  23.318648 |       -- |           -- |       -- |pe.184
||==================================================================
|  39.8% | 174.206510 | 17161829 | 289328285840 | 16858.83 |MPI
||------------------------------------------------------------------
||   1.0% | 414.091071 |    62224 |    635942368 | 10220.21 |pe.184
||   0.3% | 151.242560 |    68002 |   1039329136 | 15283.80 |pe.126
||   0.3% | 115.396258 |    68002 |   1039329136 | 15283.80 |pe.158
|==================================================================
```

Profile Guided Rank Placement Suggestion

- From the sampling profile:
  - If point to point MPI functions use a significant fraction of the program time and if they have a significant imbalance, then:
    - pat_report -O mpi_sm_rank_order ...
      - "sm" stands for "sent message“
  - If, there seems to be a load imbalance of another type, then you can get a report and suggested rank order file based on user time:
    - pat_report -O mpi_rank_order ...
  - If you have a different metric than user time that you want to balance, then:
    - pat_report -O mpi_rank_order -s mro_metric=DATA_CACHE_MISSES ...
      - "mro" stands for mpi_rank_order

Example: -O mpi_sm_rank_order (sweep3d)

```
Table 1:  Sent Message Stats and Suggested MPI Rank Order


                Communication Partner Counts

        Number   Rank
      Partners  Count      Ranks

             2      4      0   7  40  47
             3     20      1   2   3   4  ...
             4     24      9  10  11  12  ...
-----------------------------------------------------------
                   Sent Msg Total Bytes per MPI rank

              Max           Avg           Min  Max   Min
      Total Bytes   Total Bytes   Total Bytes  Rank  Rank

         60825600      51840000      29721600  9     7
-----------------------------------------------------------
       Dual core:  Sent Msg Total Bytes per node

 Rank           Max           Avg           Min  Max Node  Min Node
Order   Total Bytes   Total Bytes   Total Bytes  Ranks     Ranks

    1     87091200      69120000      42163200  10,11     6,7
    u     87091200      71884800      42163200  18,19     46,47
    d     87091200      72633600      42163200  17,18     46,47
    0    121651200     103680000      71884800  9,33      7,31
    2    121651200     103680000      60134400  26,21     40,7
```

Example: -O mpi_sm_rank_order (sweep3d)

```
Notes for table 1:

    To maximize the locality of point to point communication, choose
    and specify a Rank Order with small Max and Avg Sent Msg Total Bytes
    per node for the target number of cores per node.

    To specify a Rank Order with a numerical value, set the environment
    variable MPICH_RANK_REORDER_METHOD to the given value.

    To specify a Rank Order with a letter value 'x', set the environment
    variable MPICH_RANK_REORDER_METHOD to 3, and copy or link the file
    MPICH_RANK_ORDER.x to MPICH_RANK_ORDER.
```

Example – Swim 16 Processors

```
-------------------------------------------------------------

       Dual core:  Sent Msg Total Bytes per node

 Rank          Max          Avg          Min  Max Node  Min Node
Order  Total Bytes  Total Bytes  Total Bytes  Ranks     Ranks

    d     28736208     28736208     28736208  7,8       7,8
    1     41054984     31815902     28736208  0,1       2,3
    u     41054984     31815902     28736208  0,1       10,11
    2     57472416     50288364     28736208  9,6       8,7
    0     69791192     60552110     57472416  0,8       1,9

-------------------------------------------------------------

       Quad core:  Sent Msg Total Bytes per node

 Rank          Max          Avg          Min  Max Node   Min Node
Order  Total Bytes  Total Bytes  Total Bytes  Ranks      Ranks

    1     41054984     34895596     28736208  0,1,2,3    4,5,6,7
    2     57472416     43104312     28736208  10,5,11,4  8,7,9,6
    d     57472416     43104312     28736208  5,6,11,12  7,8,9,10
    u     69791192     56447752     28736208  0,1,4,5    10,11,8,9
    0     69791192     63631804     57472416  0,8,1,9    2,10,3,11
```

Example: -O mpi_rank_order (sweep3d)

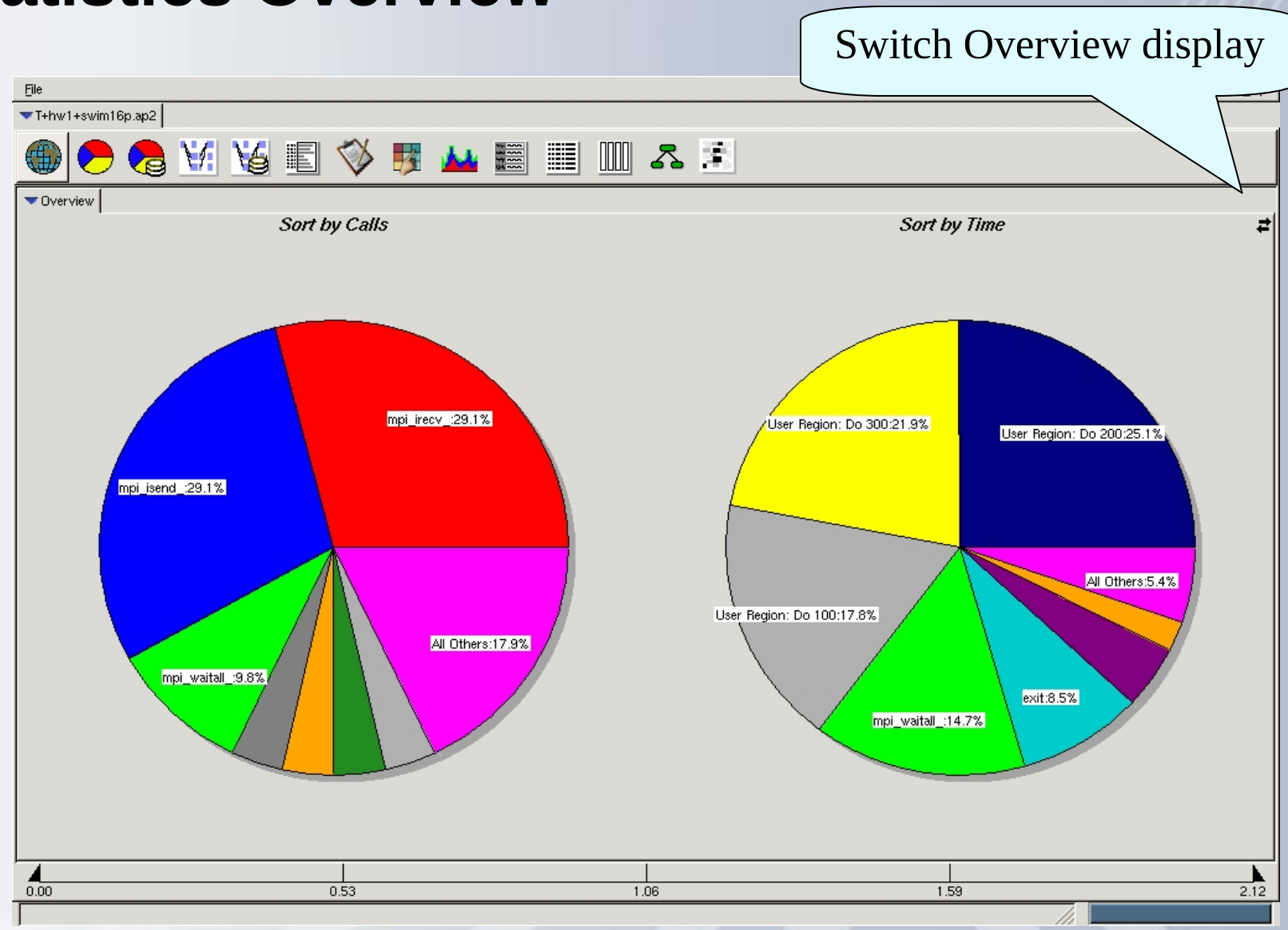
```
Table 1:  Suggested MPI Rank Order

                  USER Time per MPI rank

             Max          Avg          Min  Max   Min
       USER Time    USER Time    USER Time  Rank  Rank

      7385065415  7260112829  7055902995  36    14
-----------------------------------------------------------

       Dual core:  USER Time per node

 Rank          Max          Avg          Min  Max Node  Min Node
Order    USER Time    USER Time    USER Time  Ranks     Ranks

    d  14578808583  14520225658  14440968410  40,43     36,14
    1  14677232995  14520225658  14191401210  36,37     46,47
    0  14748640489  14520225658  14171663668  11,35     22,46
    2  14756120062  14520225658  14287395096  36,11     46,1
-----------------------------------------------------------

       Quad core:  USER Time per node

 Rank          Max          Avg          Min  Max Node      Min Node
Order    USER Time    USER Time    USER Time  Ranks         Ranks

    d  29062641310  29040451316  29019776993  28,33,9,6     40,43,36,14
    1  29327030609  29040451316  28732792941  8,9,10,11     44,45,46,47
    0  29329831324  29040451316  28442065777  8,32,9,33     22,46,23,47
    2  29334330628  29040451316  28641644420  36,11,37,10   46,1,47,0
```

Cray Apprentice^2

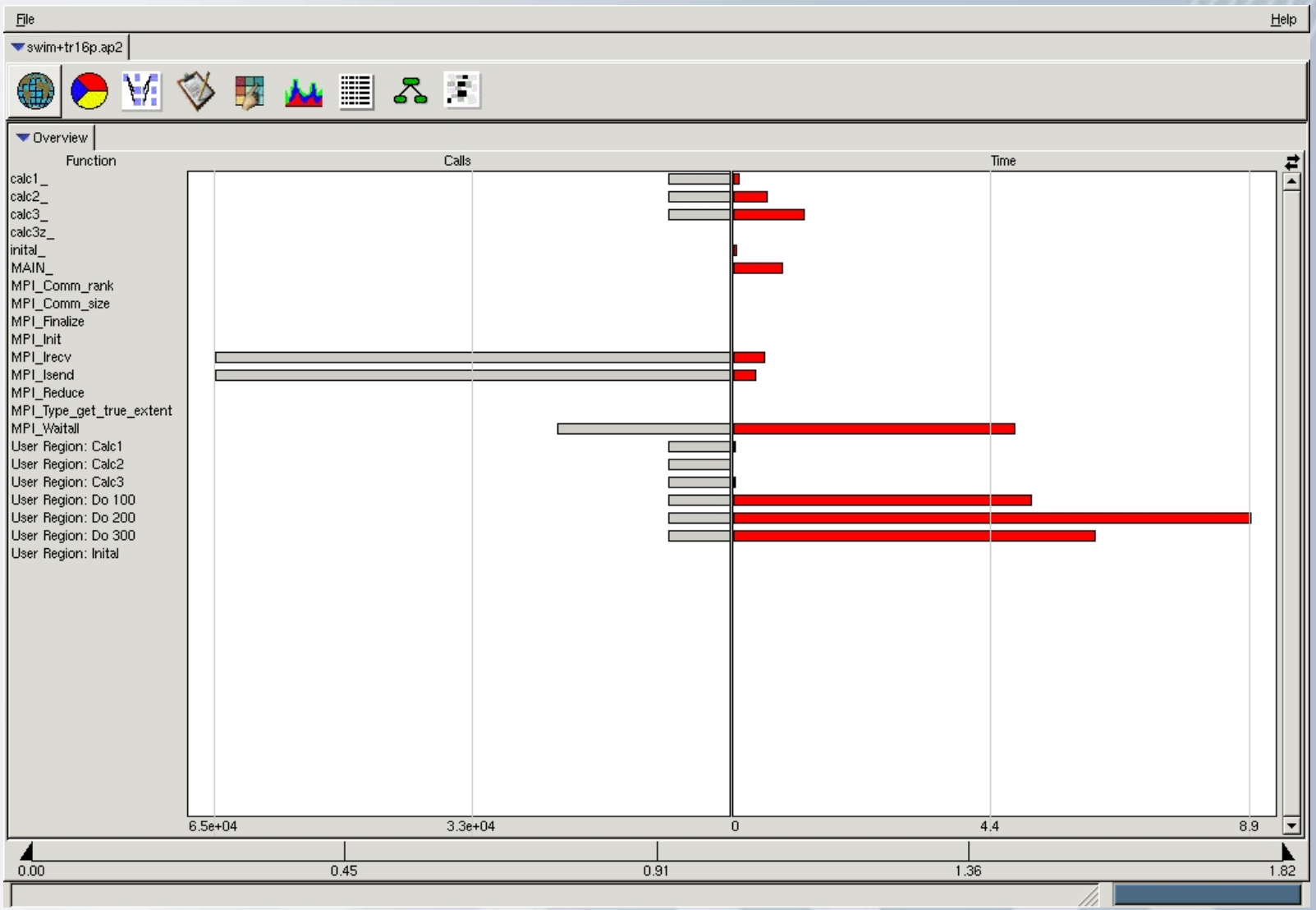
- Call graph profile
- Communication statistics
- Time-line view
  - Communication
  - I/O
- Activity view
- Pair-wise communication statistics
- Text reports
- Source code mapping

- Cray Apprentice^2
- is target to help and correct:
  - Load imbalance
  - Excessive communication
  - Network contention
  - Excessive serialization
  - I/O Problems

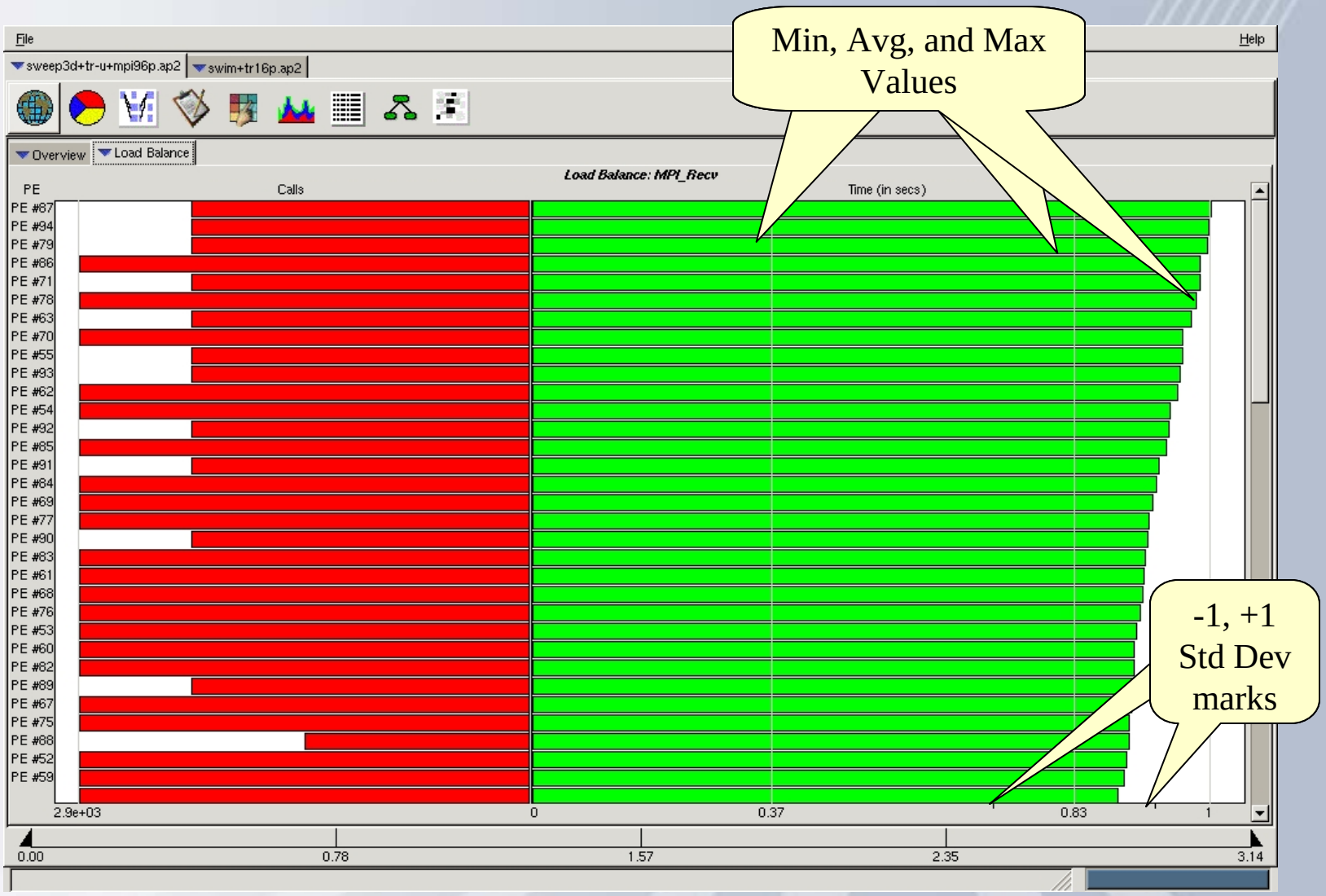
Statistics Overview



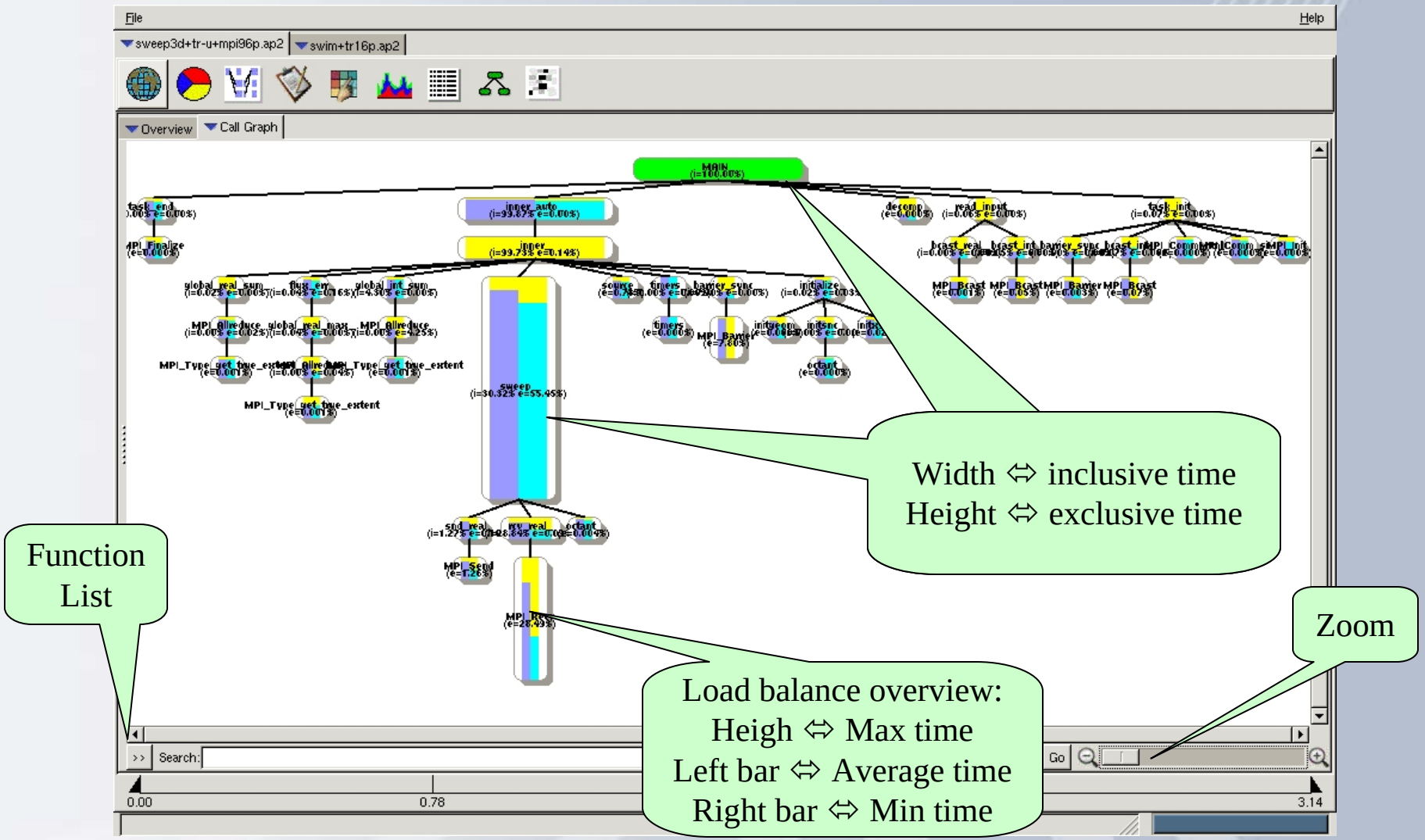
Function Profile



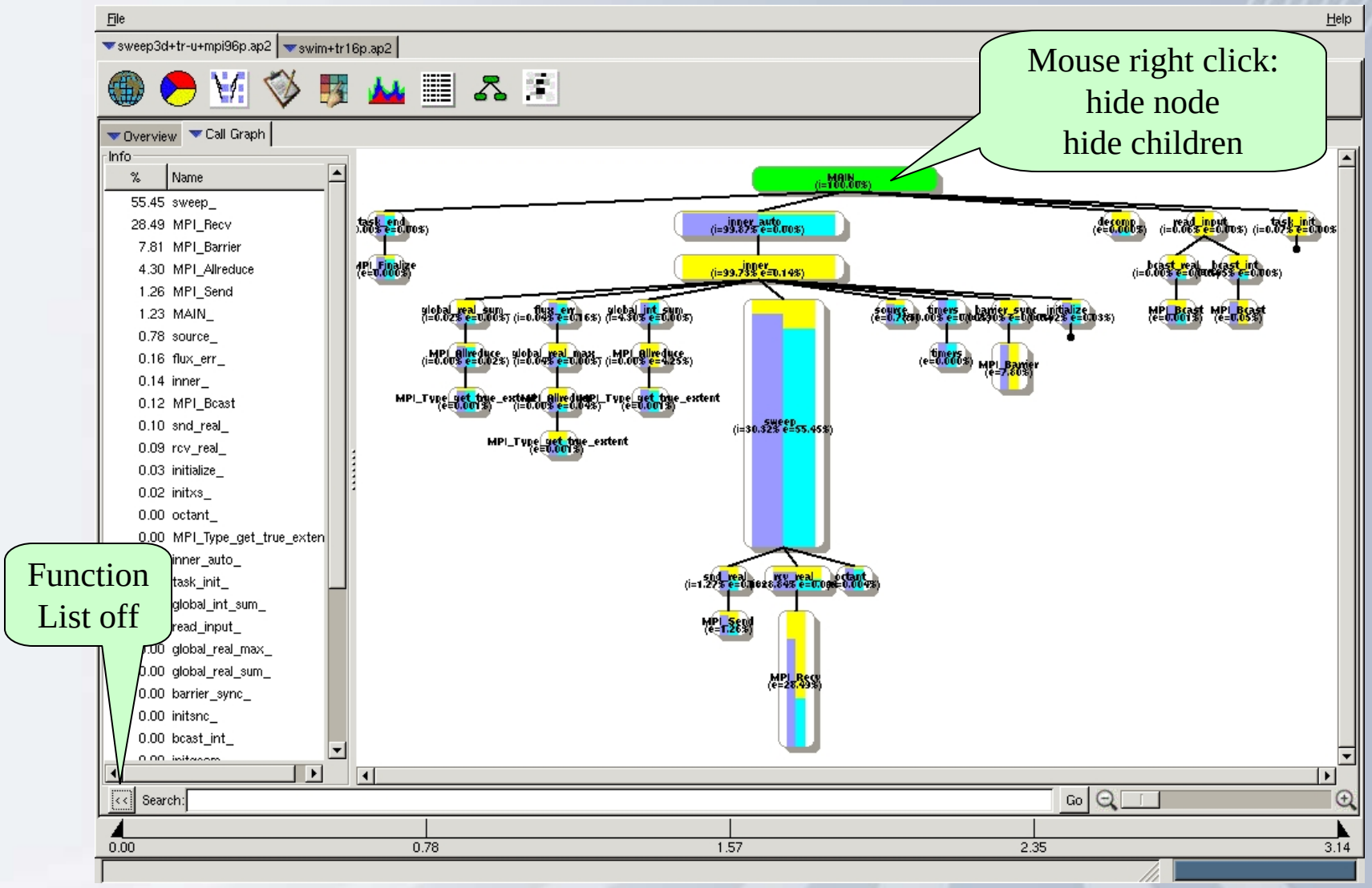
Load Balance View (Aggregated)



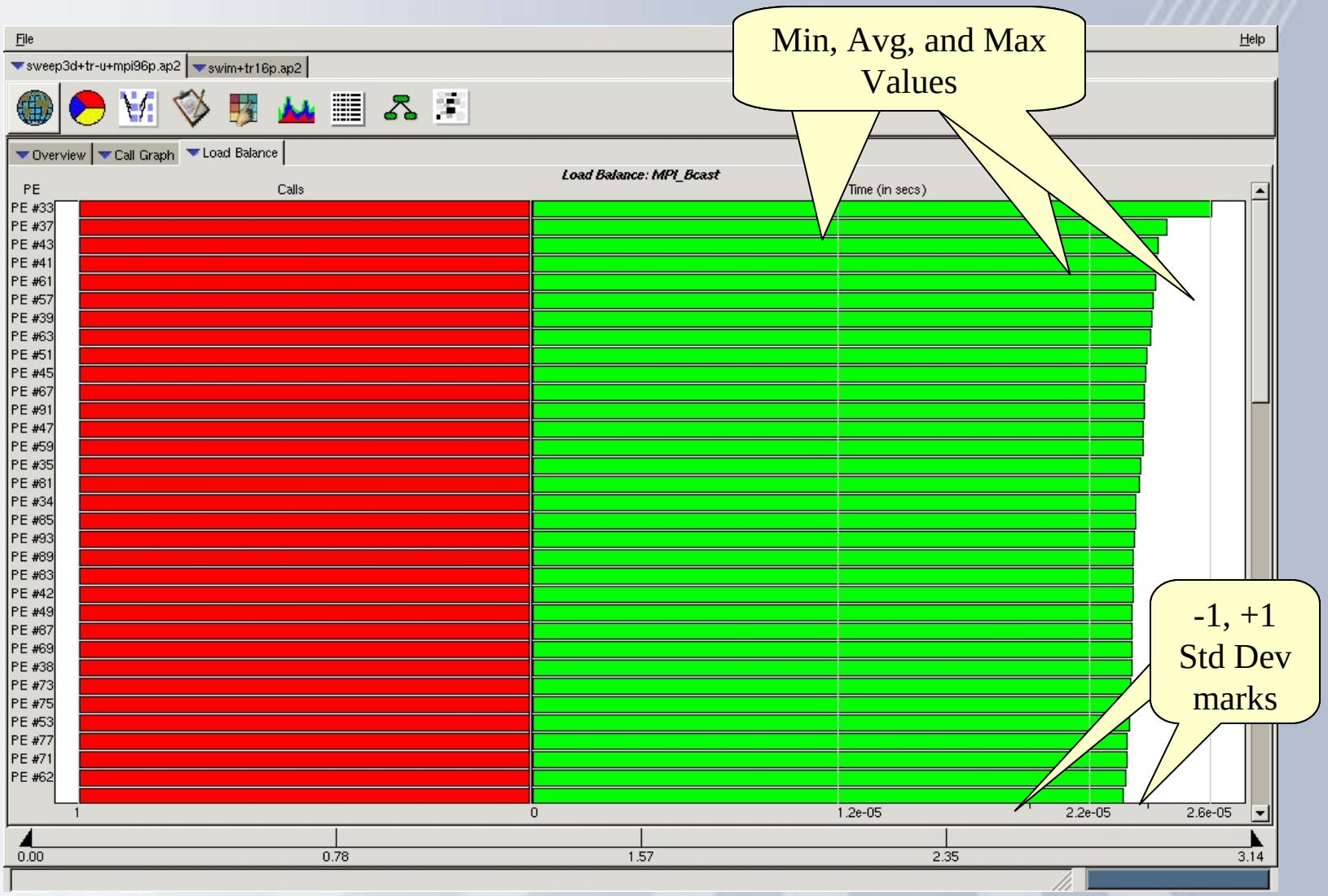
Call Graph View - Zoom



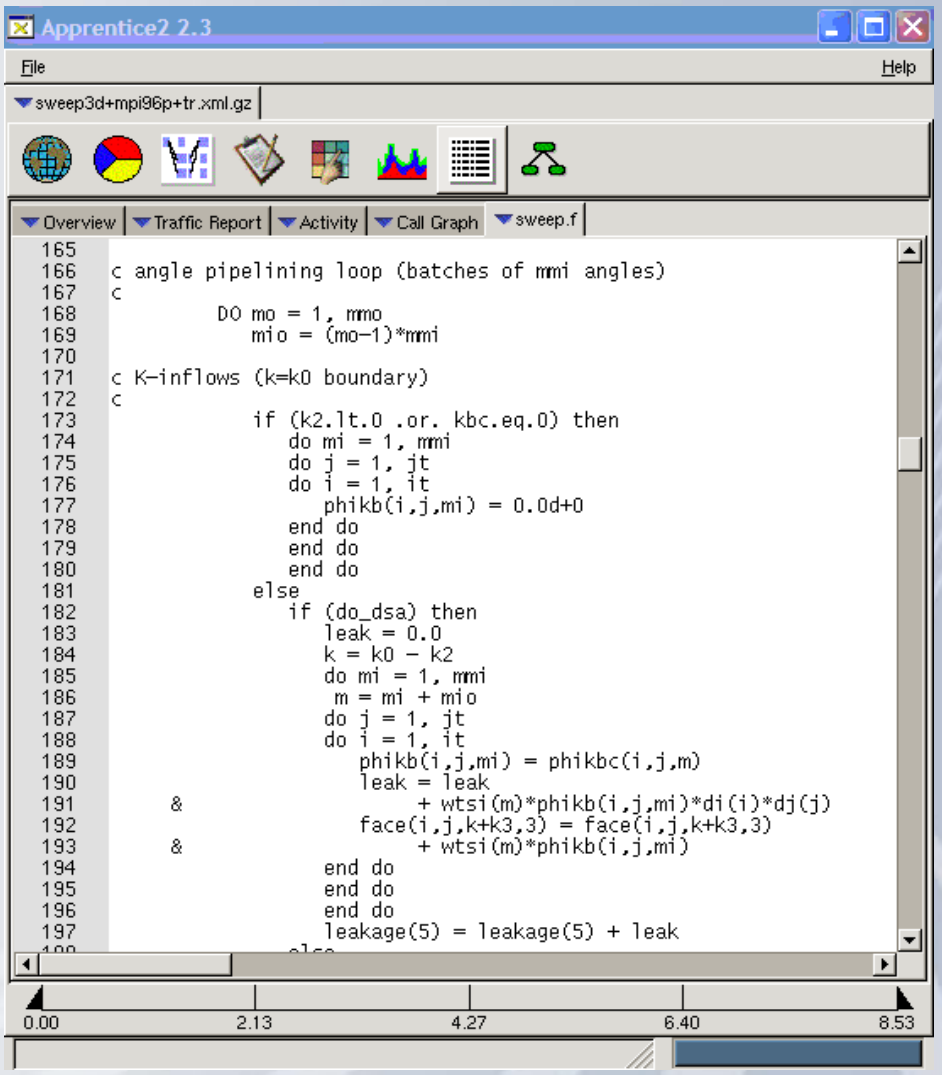
Call Graph View – Function List



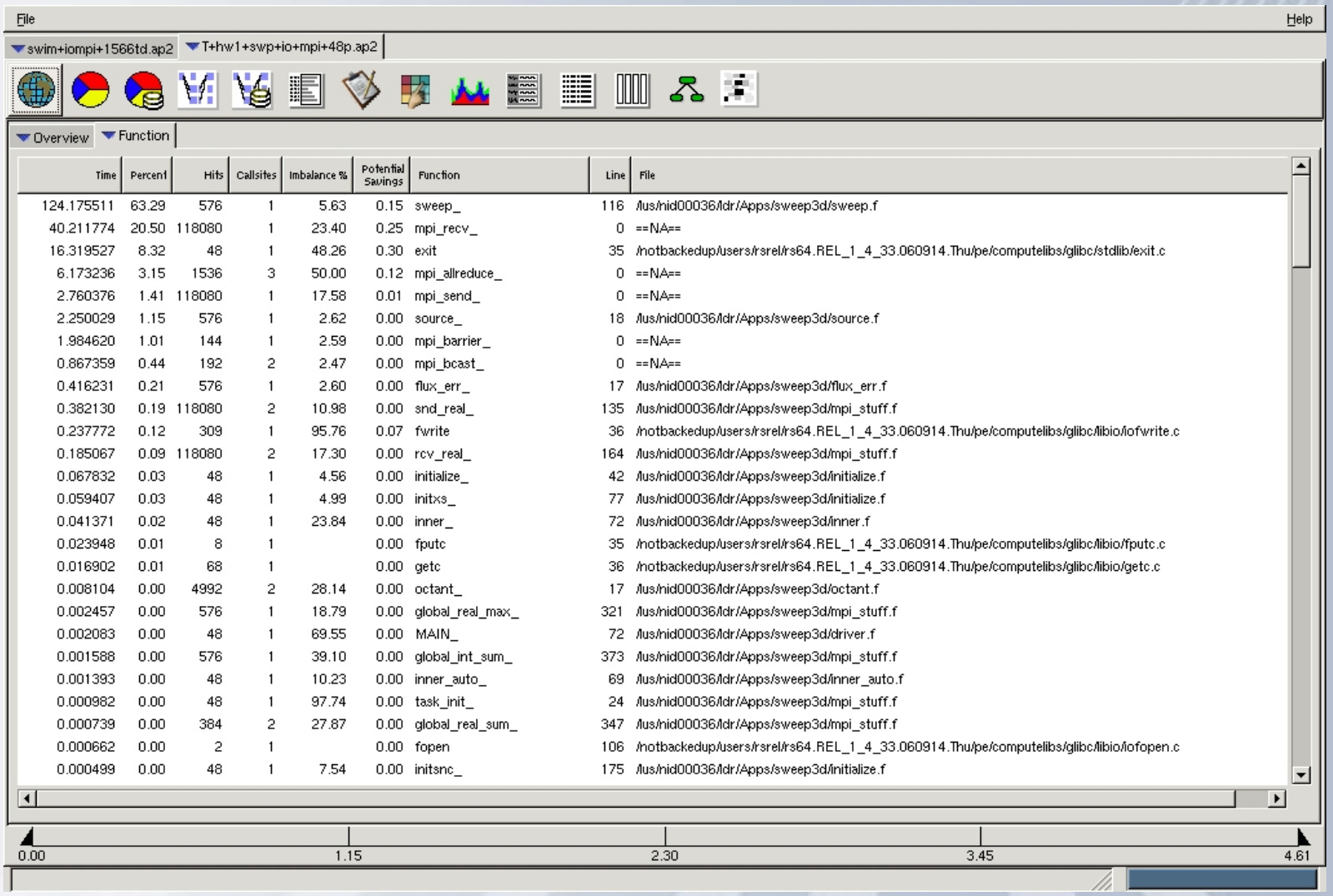
Load Balance View (from Call Graph)



Source Mapping from Call Graph

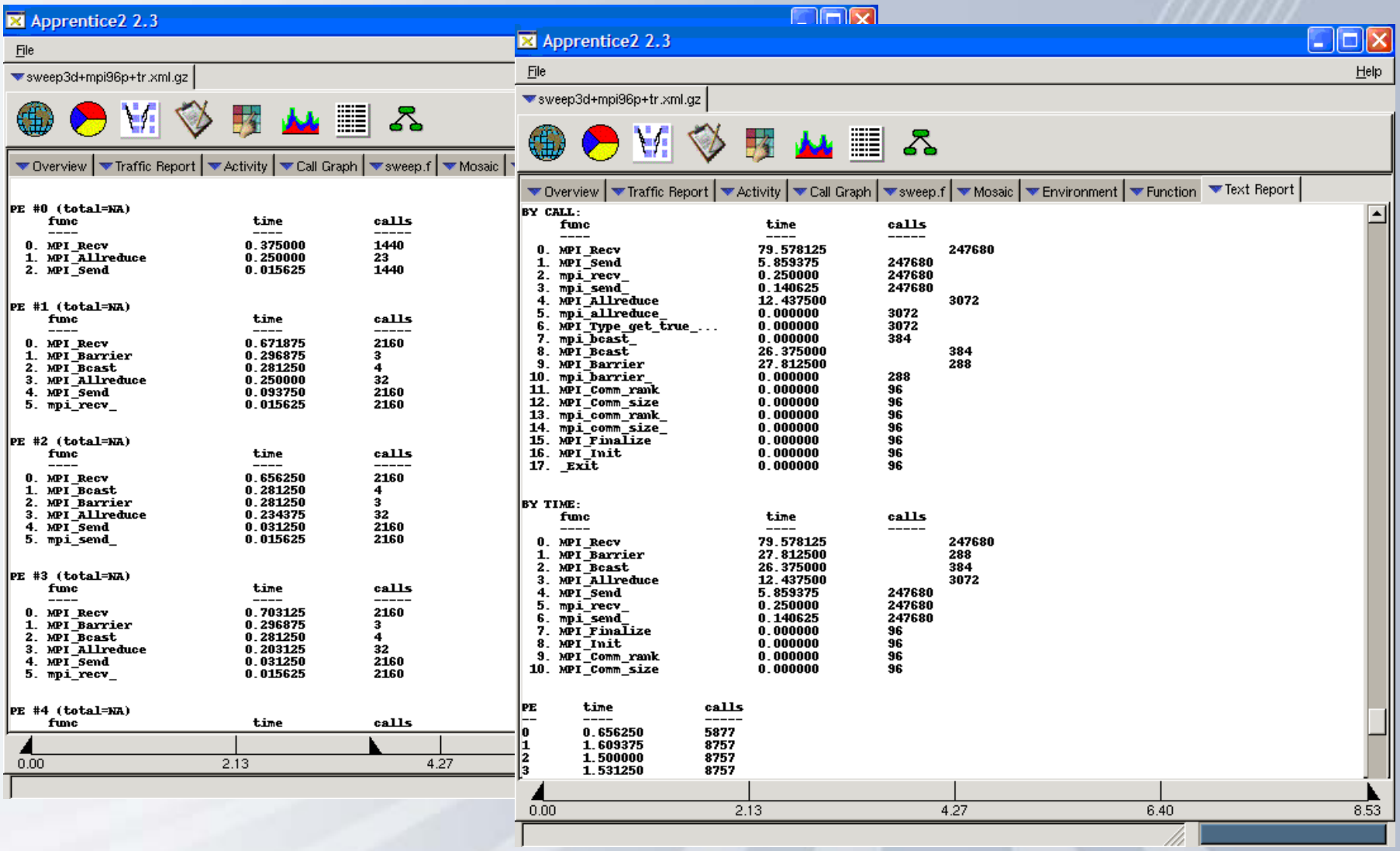


Function Profile

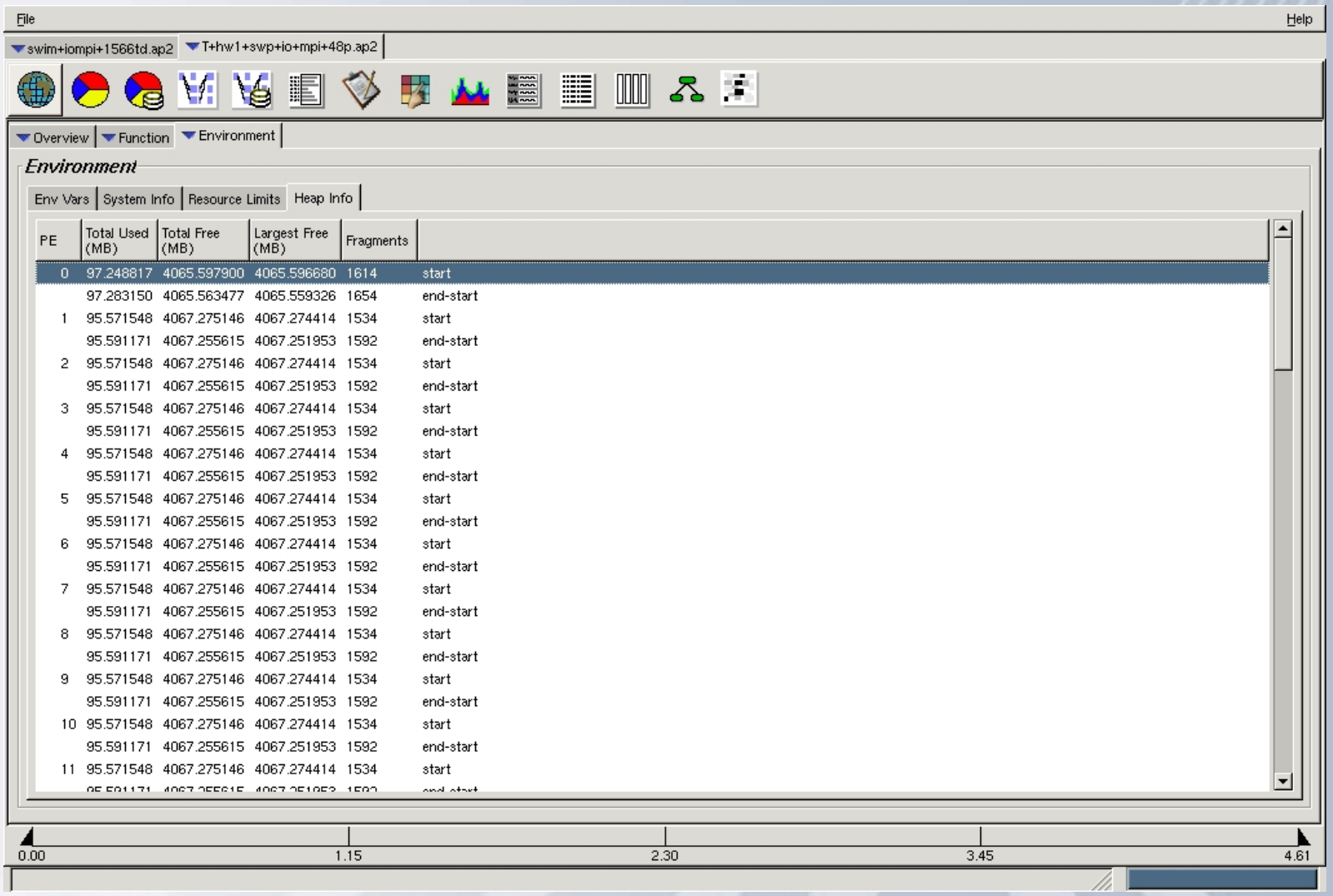


Time	Percent	Hits	Callsites	Imbalance %	Potential Savings	Function	Line	File
124.175511	63.29	576	1	5.63	0.15	sweep_	116	/lus/nid00036/ldr/Apps/sweep3d/sweep.f
40.211774	20.50	118080	1	23.40	0.25	mpi_recv_	0	==NA==
16.319527	8.32	48	1	48.26	0.30	exit	35	/notbackedup/users/rsrel/rs64.REL_1_4_33.060914.Thu/pe/computelibs/glibc/stdlib/exit.c
6.173236	3.15	1536	3	50.00	0.12	mpi_allreduce_	0	==NA==
2.760376	1.41	118080	1	17.58	0.01	mpi_send_	0	==NA==
2.250029	1.15	576	1	2.62	0.00	source_	18	/lus/nid00036/ldr/Apps/sweep3d/source.f
1.984620	1.01	144	1	2.59	0.00	mpi_barrier_	0	==NA==
0.867359	0.44	192	2	2.47	0.00	mpi_bcast_	0	==NA==
0.416231	0.21	576	1	2.60	0.00	flux_err_	17	/lus/nid00036/ldr/Apps/sweep3d/flux_err.f
0.382130	0.19	118080	2	10.98	0.00	snd_real_	135	/lus/nid00036/ldr/Apps/sweep3d/mpi_stuff.f
0.237772	0.12	309	1	95.76	0.07	fwrite	36	/notbackedup/users/rsrel/rs64.REL_1_4_33.060914.Thu/pe/computelibs/glibc/libio/iofwrite.c
0.185067	0.09	118080	2	17.30	0.00	rcv_real_	164	/lus/nid00036/ldr/Apps/sweep3d/mpi_stuff.f
0.067832	0.03	48	1	4.56	0.00	initialize_	42	/lus/nid00036/ldr/Apps/sweep3d/initialize.f
0.059407	0.03	48	1	4.99	0.00	initxs_	77	/lus/nid00036/ldr/Apps/sweep3d/initialize.f
0.041371	0.02	48	1	23.84	0.00	inner_	72	/lus/nid00036/ldr/Apps/sweep3d/inner.f
0.023948	0.01	8	1		0.00	fputc	35	/notbackedup/users/rsrel/rs64.REL_1_4_33.060914.Thu/pe/computelibs/glibc/libio/fputc.c
0.016902	0.01	68	1		0.00	getc	36	/notbackedup/users/rsrel/rs64.REL_1_4_33.060914.Thu/pe/computelibs/glibc/libio/getc.c
0.008104	0.00	4992	2	28.14	0.00	octant_	17	/lus/nid00036/ldr/Apps/sweep3d/octant.f
0.002457	0.00	576	1	18.79	0.00	global_real_max_	321	/lus/nid00036/ldr/Apps/sweep3d/mpi_stuff.f
0.002083	0.00	48	1	69.55	0.00	MAIN_	72	/lus/nid00036/ldr/Apps/sweep3d/driver.f
0.001588	0.00	576	1	39.10	0.00	global_int_sum_	373	/lus/nid00036/ldr/Apps/sweep3d/mpi_stuff.f
0.001393	0.00	48	1	10.23	0.00	inner_auto_	69	/lus/nid00036/ldr/Apps/sweep3d/inner_auto.f
0.000982	0.00	48	1	97.74	0.00	task_init_	24	/lus/nid00036/ldr/Apps/sweep3d/mpi_stuff.f
0.000739	0.00	384	2	27.87	0.00	global_real_sum_	347	/lus/nid00036/ldr/Apps/sweep3d/mpi_stuff.f
0.000662	0.00	2	1		0.00	fopen	106	/notbackedup/users/rsrel/rs64.REL_1_4_33.060914.Thu/pe/computelibs/glibc/libio/iofopen.c
0.000499	0.00	48	1	7.54	0.00	initsnc_	175	/lus/nid00036/ldr/Apps/sweep3d/initialize.f

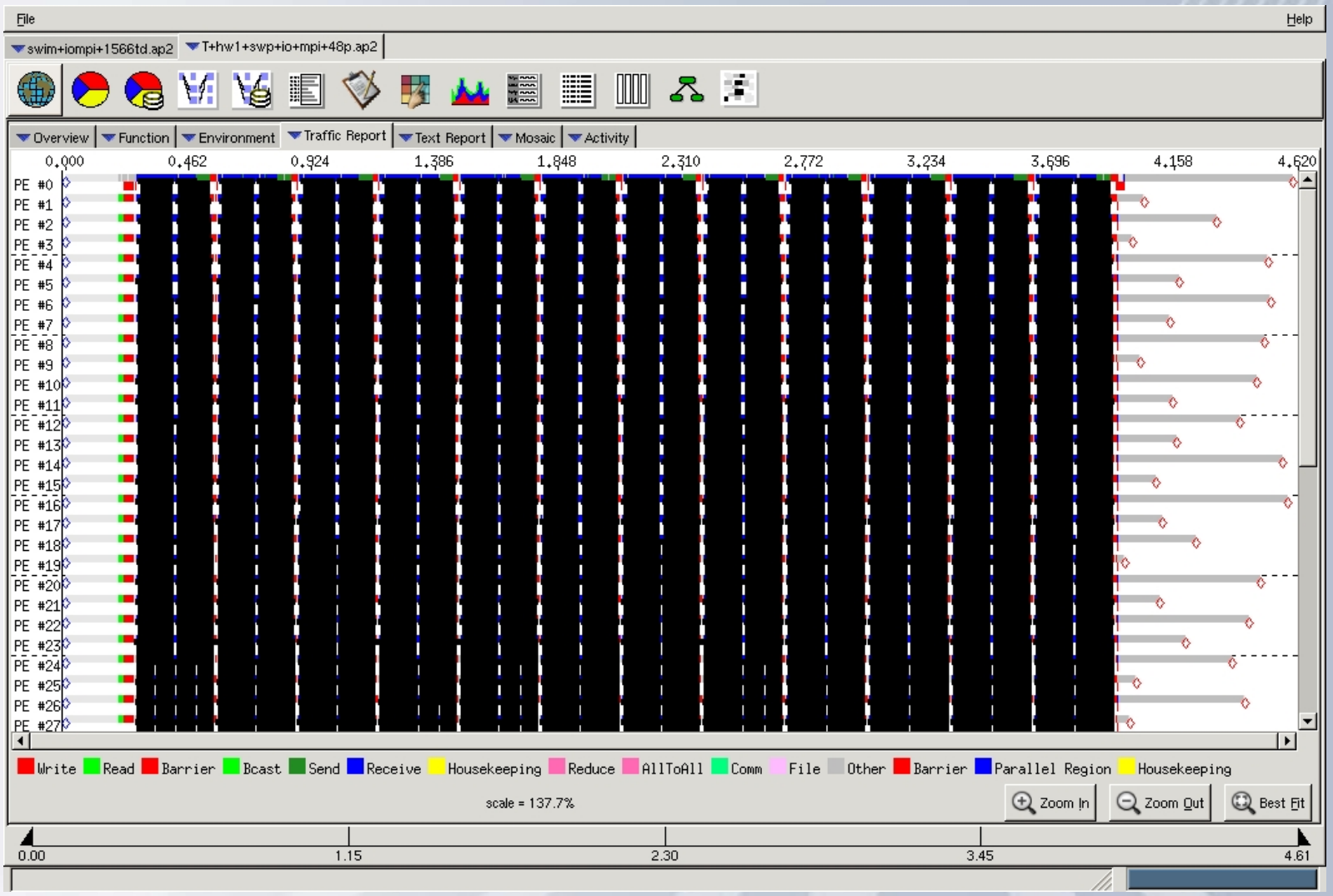
Distribution by PE, by Call, & by Time



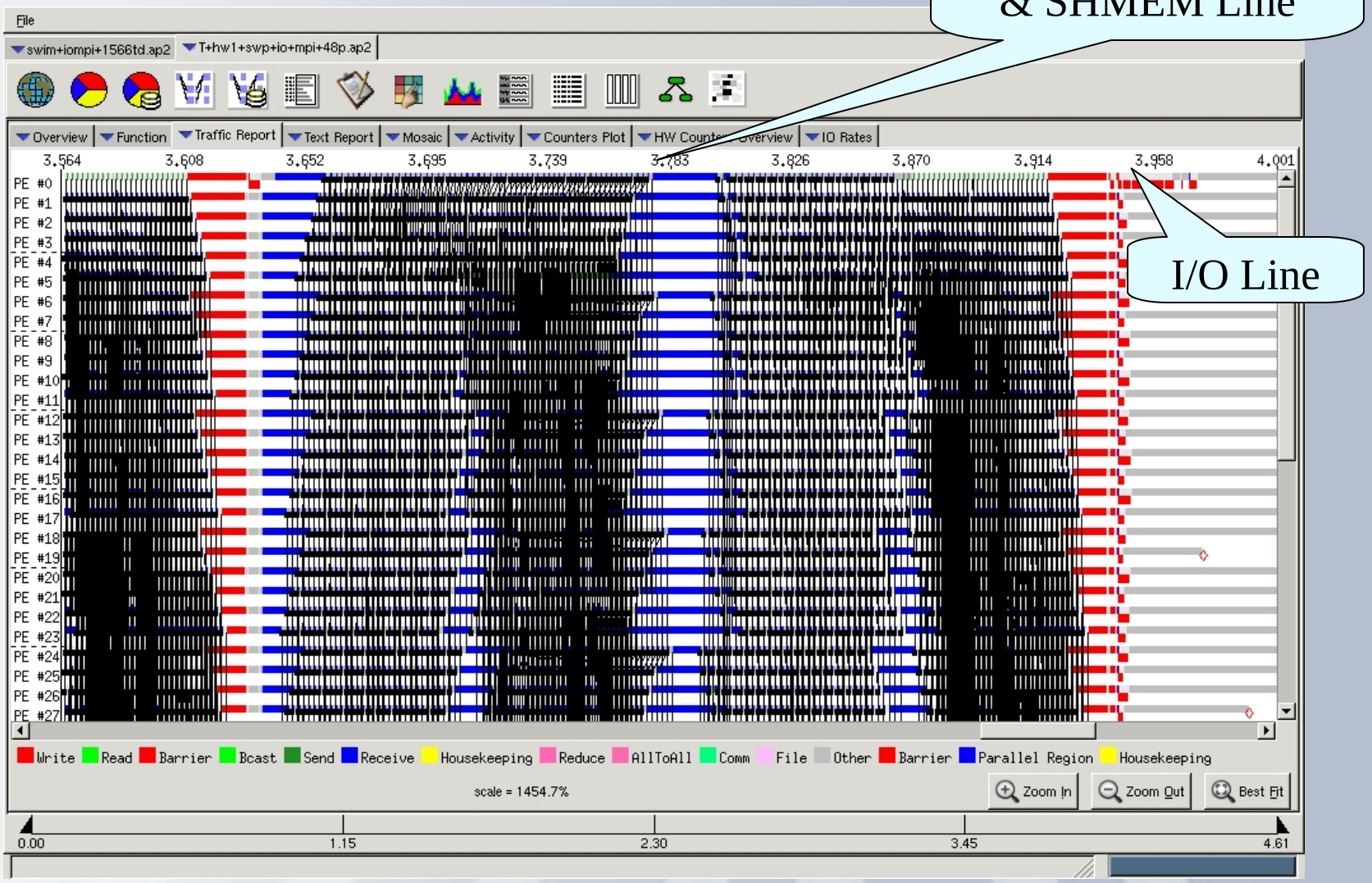
Environment & Execution Details



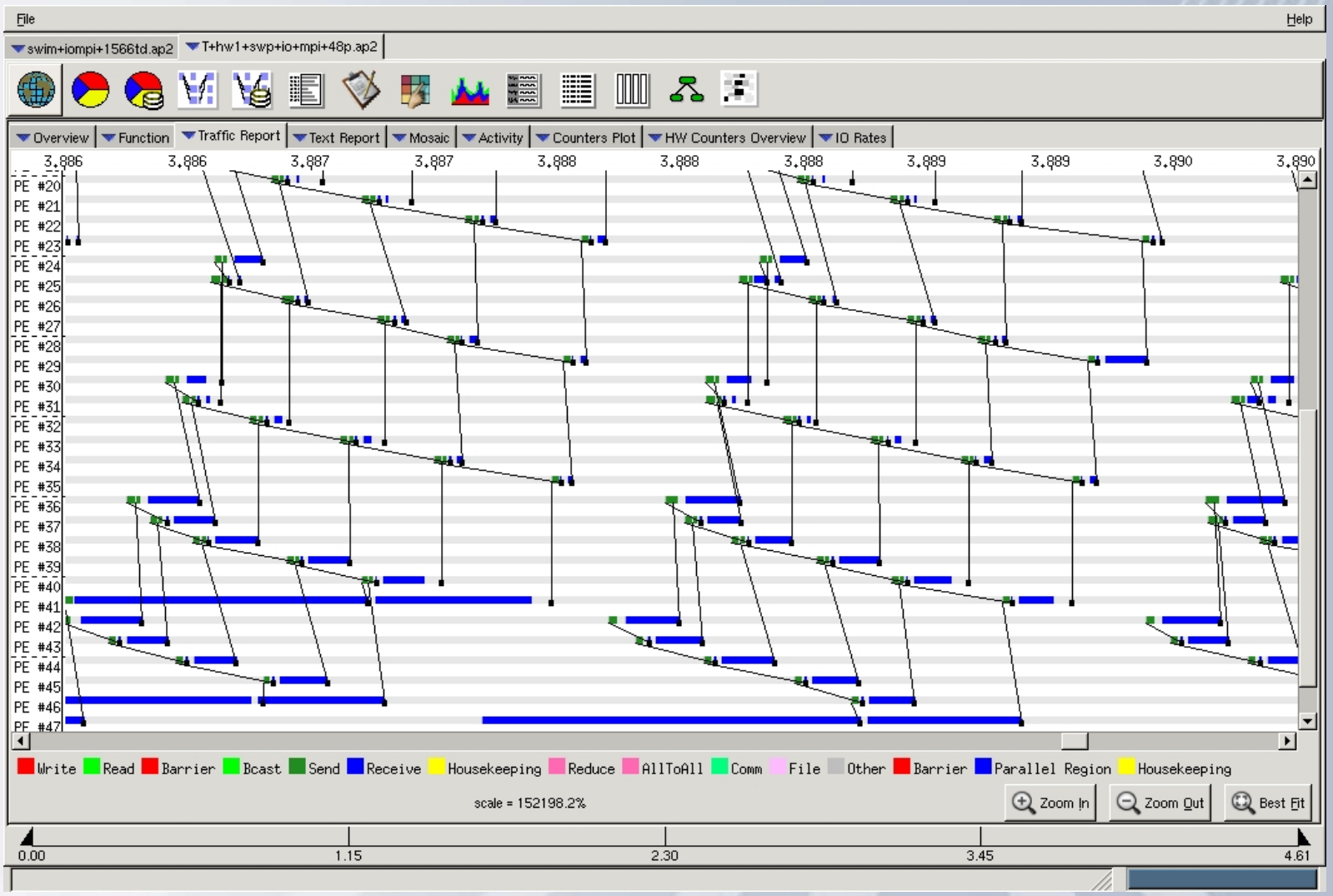
Time Line View (Sweep3D)



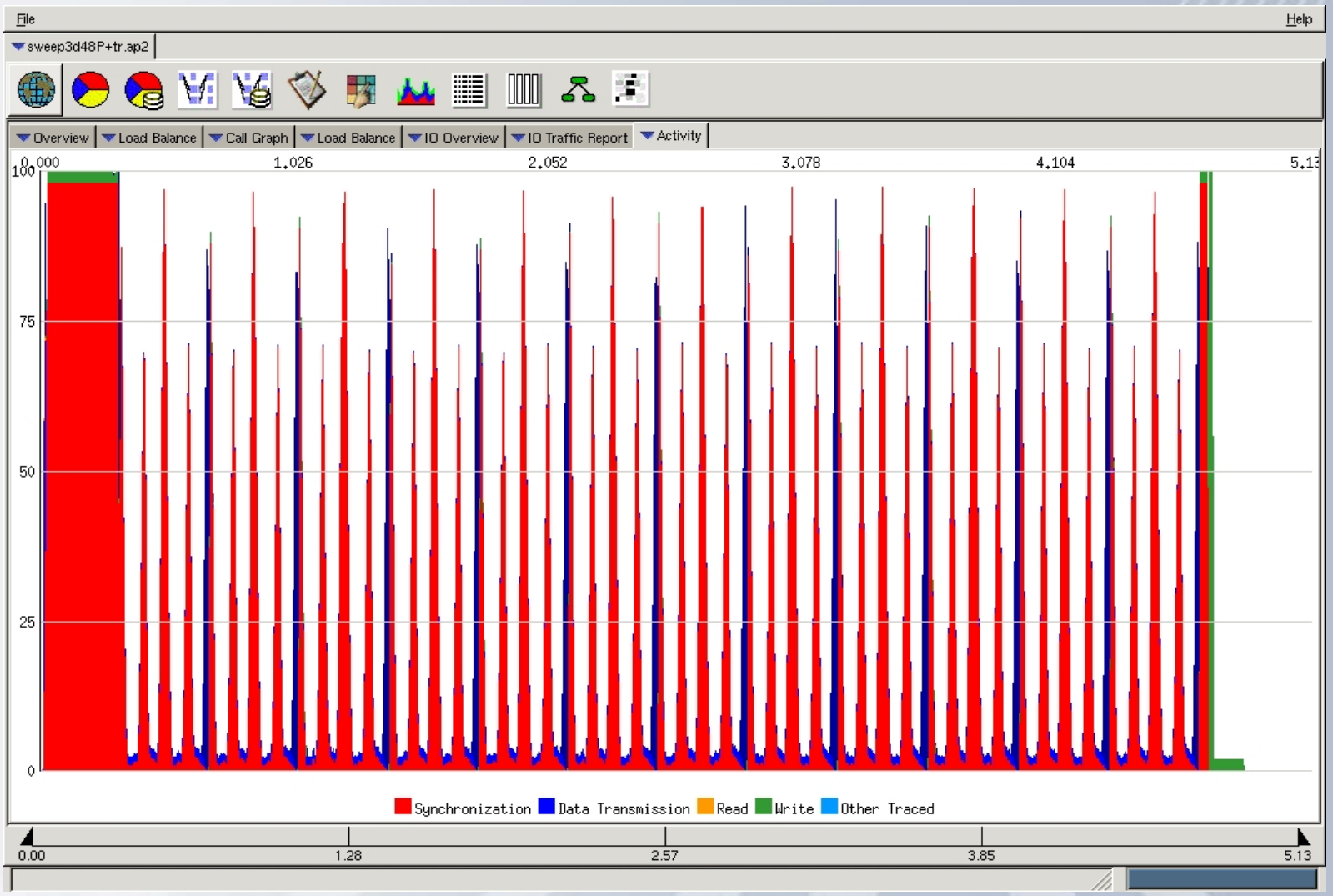
Time Line View (Zoom)



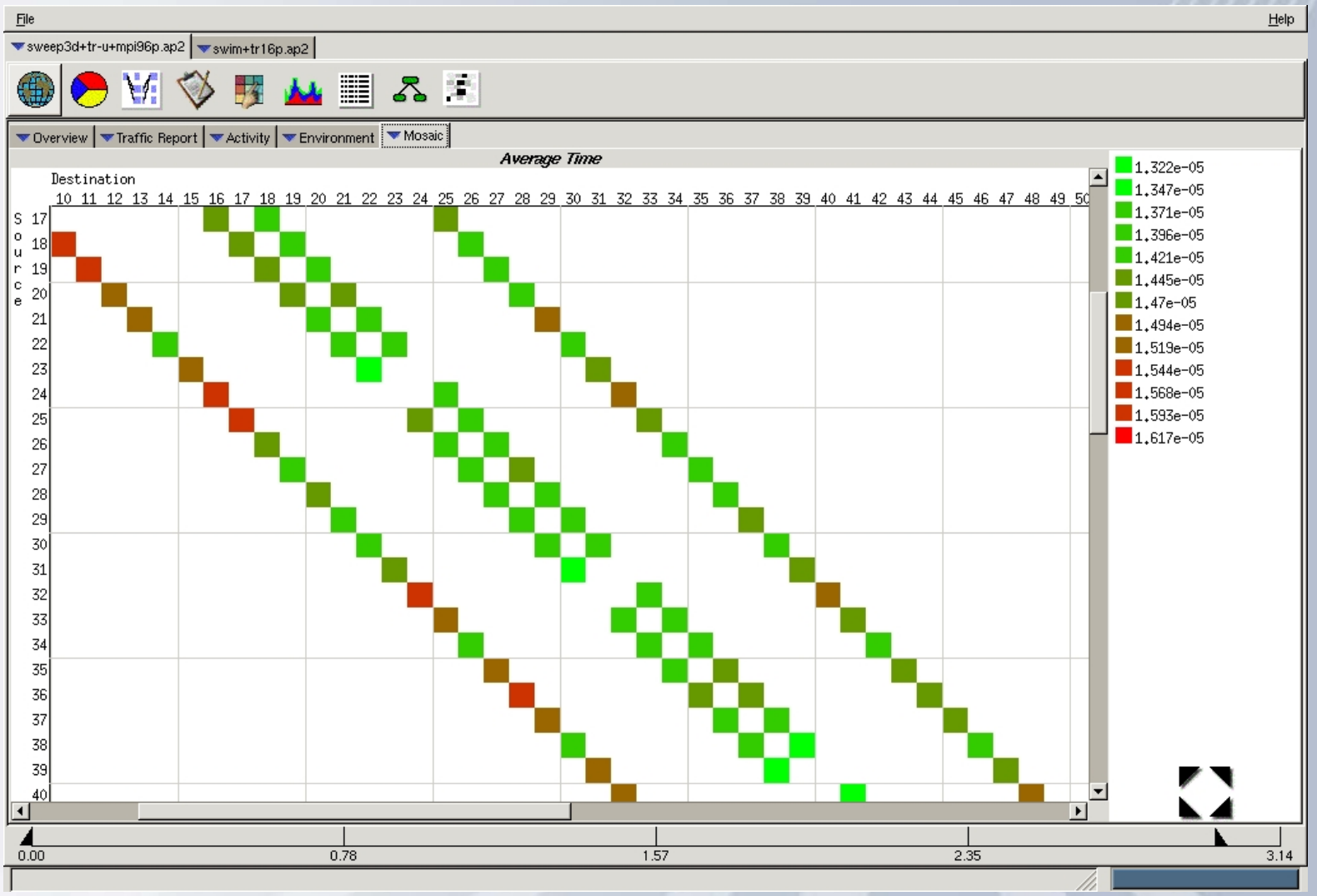
Time Line View (Fine Grain Zoom)



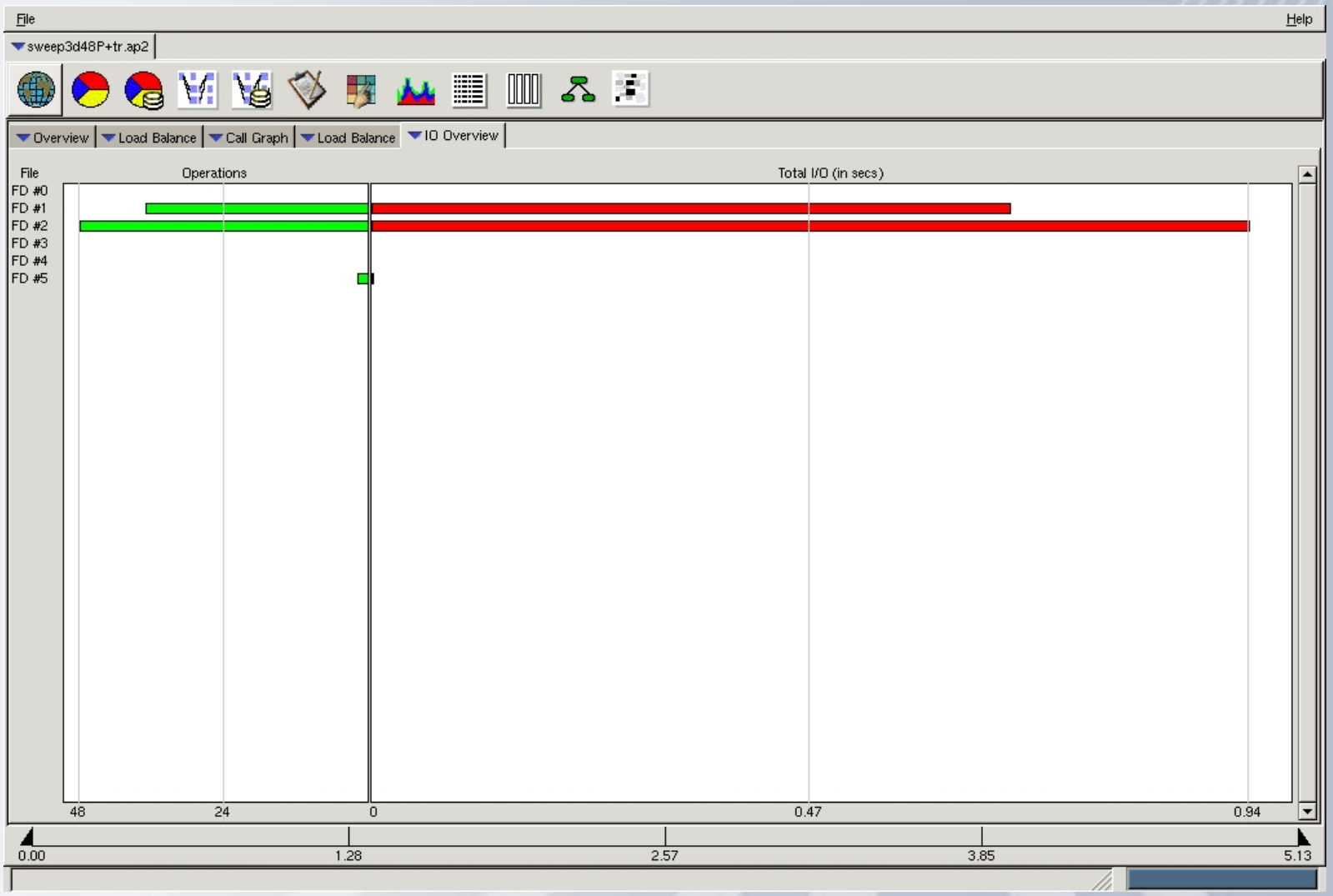
Activity View



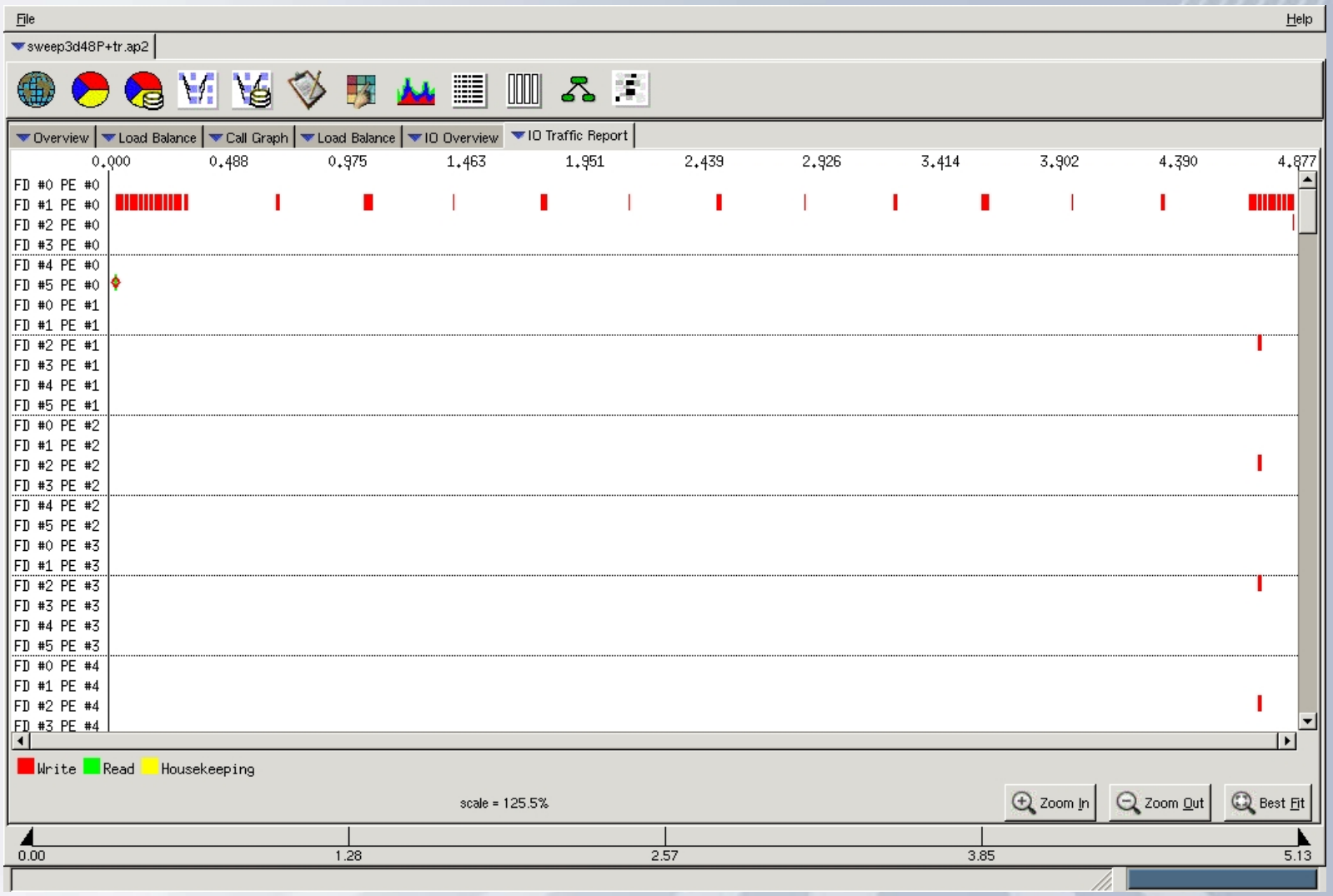
Pair-wise Communication



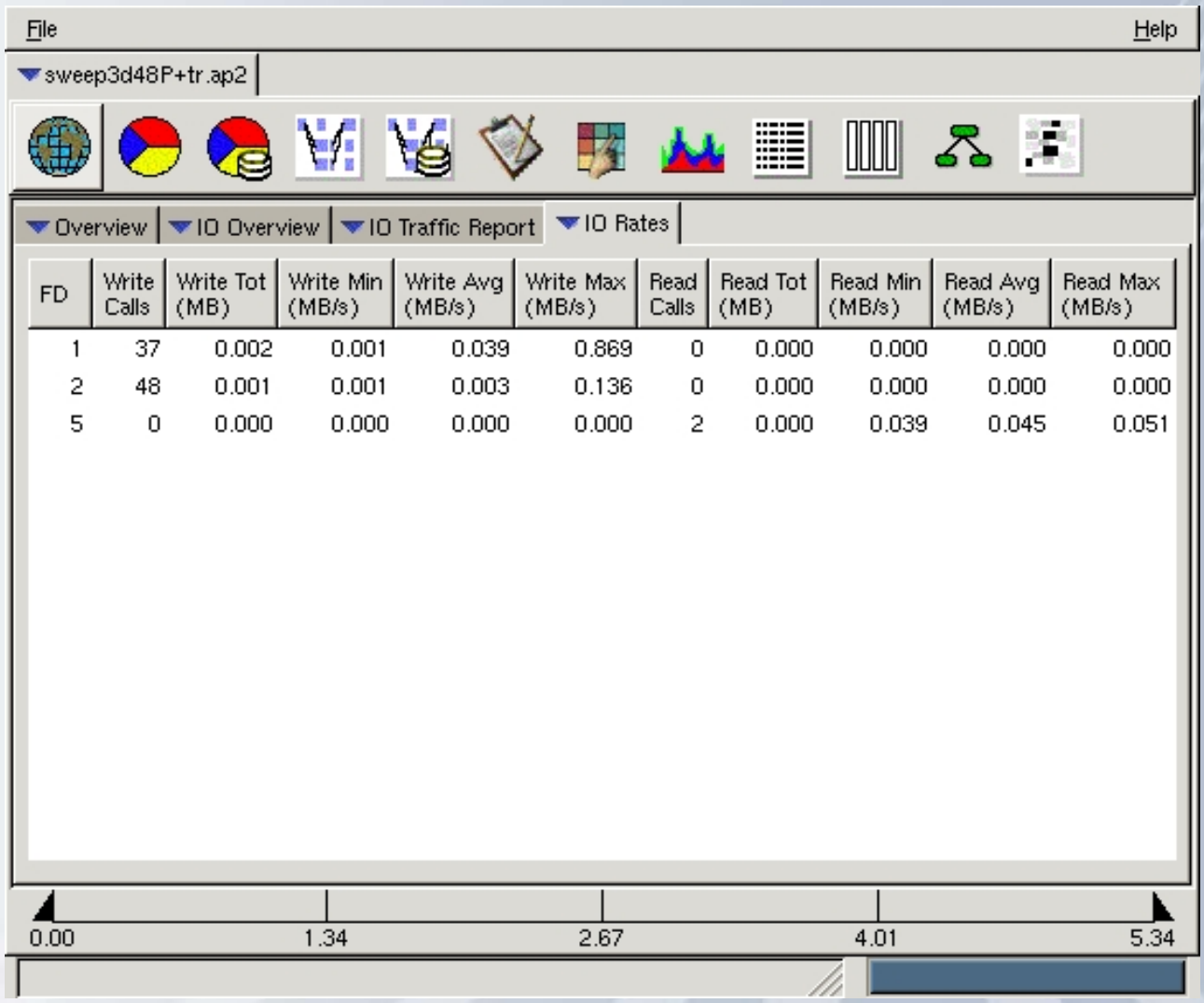
I/O Overview


File
Help
sweep3d48P+tr.ap2
Overview
Load Balance
Call Graph
Load Balance
IO Overview
File
Operations
Total I/O (in secs)
FD #0
FD #1
FD #2
FD #3
FD #4
FD #5
48
24
0
0.47
0.94
0.00
1.28
2.57
3.85
5.13

I/O Traffic Report

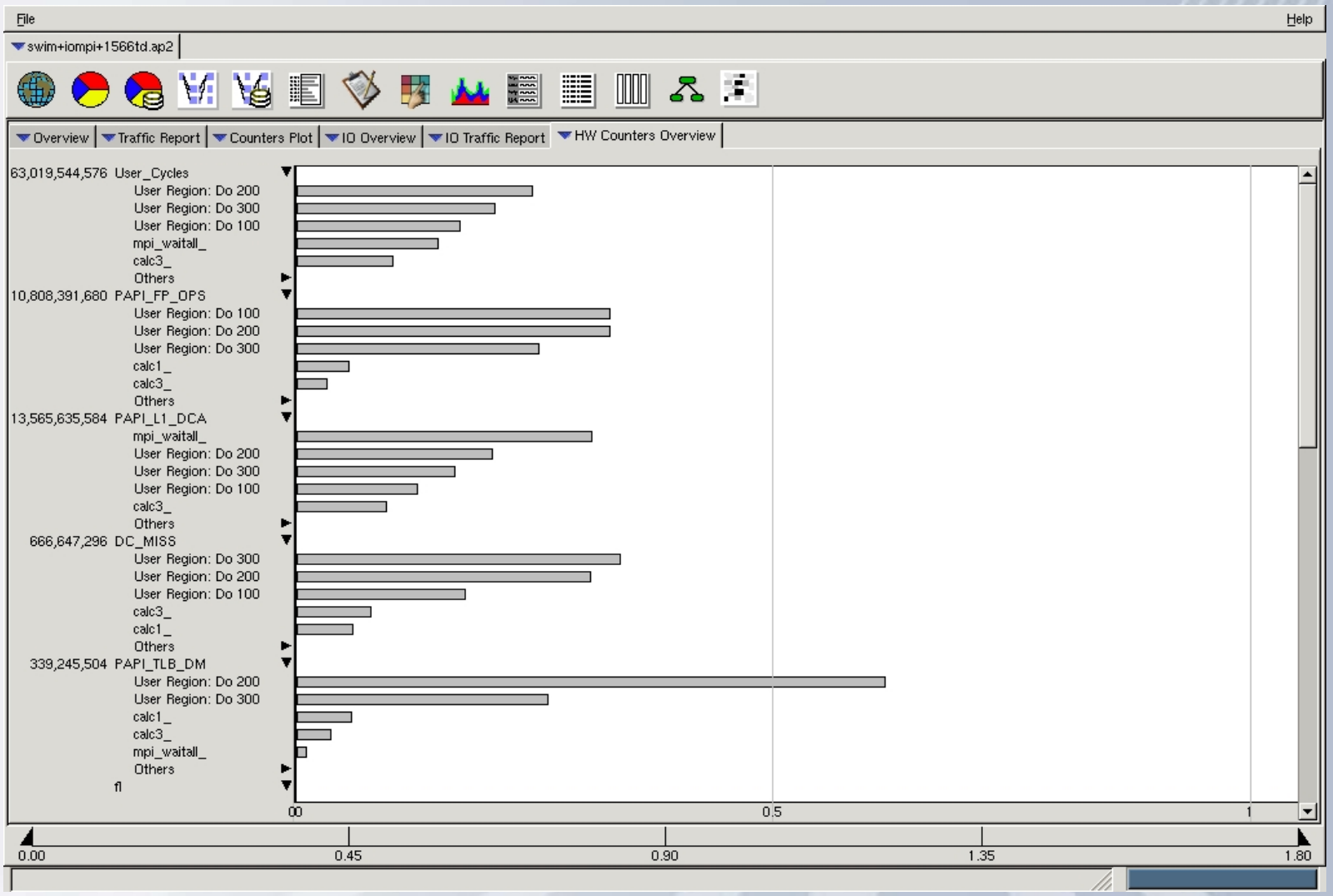


I/O Rates

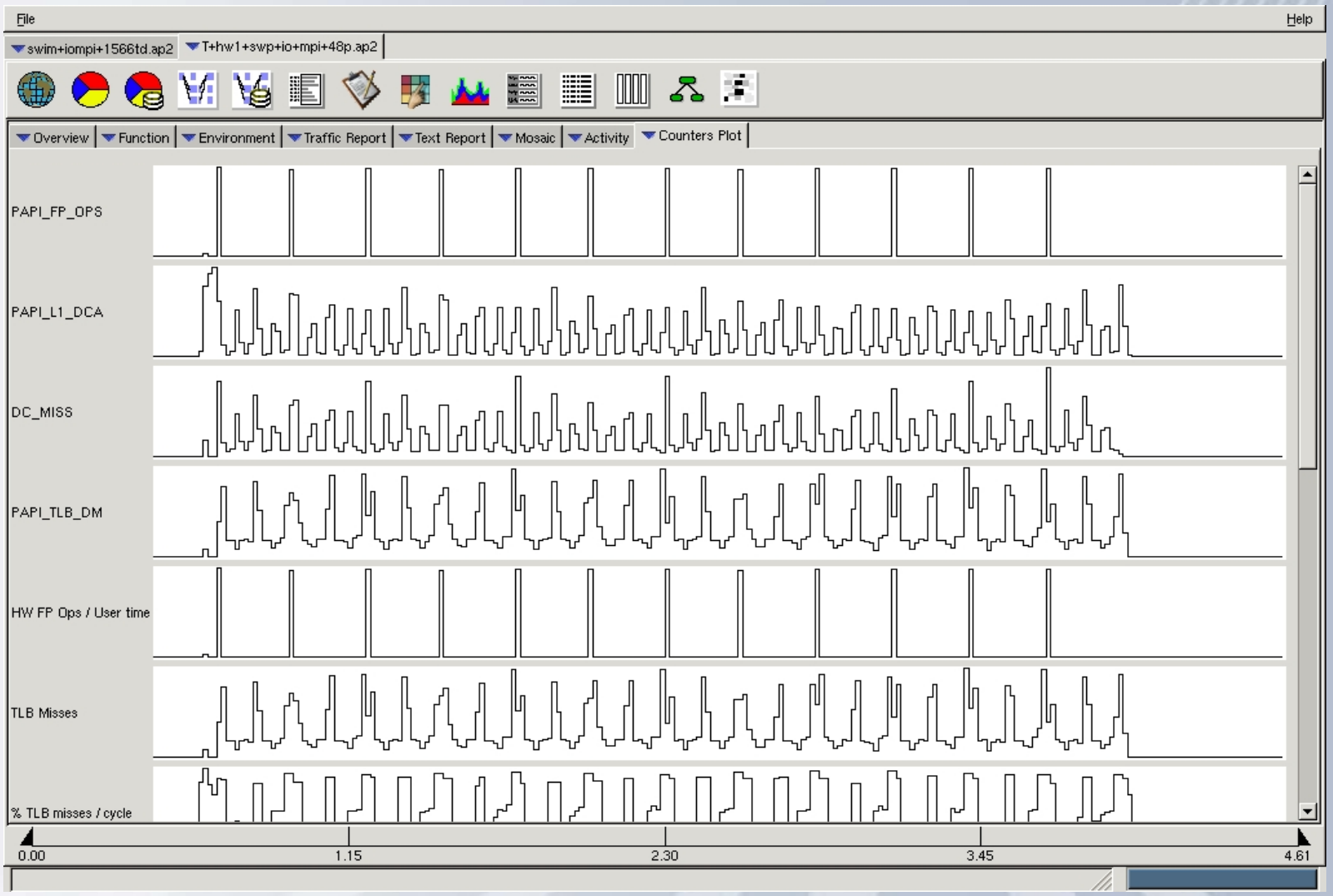


FD	Write Calls	Write Tot (MB)	Write Min (MB/s)	Write Avg (MB/s)	Write Max (MB/s)	Read Calls	Read Tot (MB)	Read Min (MB/s)	Read Avg (MB/s)	Read Max (MB/s)
1	37	0.002	0.001	0.039	0.869	0	0.000	0.000	0.000	0.000
2	48	0.001	0.001	0.003	0.136	0	0.000	0.000	0.000	0.000
5	0	0.000	0.000	0.000	0.000	2	0.000	0.039	0.045	0.051

Hardware Counters Overview



Hardware Counters Time Line



Controlling Trace File Size

- Several environment variables are available to limit trace files to a reasonable size:
  - PAT_RT_CALLSTACK
    - Limit the depth to trace the call stack
  - PAT_RT_HWPC
    - Avoid collecting hardware counters (unset)
  - PAT_RT_RECORD_PE
    - Collect trace for a subset of the PEs
  - PAT_RT_TRACE_FUNCTION_ARGS
    - Limit the number of function arguments to be traced
  - PAT_RT_TRACE_FUNCTION_LIMITS
    - Avoid tracing indicated functions
  - PAT_RT_TRACE_FUNCTION_MAX
    - Limit the maximum number of traces generated for all functions for a single process
- Use CrayPat API to toggle trace state (on / off) or to select functions to trace
  - int PAT_tracing_state (int state)
  - int PAT_trace_function (const void *addr, int state)
- Use the limit built-in command for ksh(1) or csh(1) to control how much disk space the trace file can consume

Additional API Functions

- int PAT_profiling_state (int state)
- int PAT_record (int state)
- int PAT_sampling_state (int state)
- int PAT_tracing_state (int state)
- int PAT_trace_function (const void *addr, int state)
- State can have one of the following:
  - PAT_STATE_ON
  - PAT_STATE_OFF
  - PAT_STATE_QUERY
- int PAT_flush_buffer (void)

Parallel Performance Analysis and Visualization on the Cray XT

Questions / Comments

Thank You!

University of Bergen
Bergen, Norway
March 11-14, 2008

Luiz DeRose (ldr@cray.com)

CRAY
THE SUPERCOMPUTER COMPANY