Performance Measurement and Analysis Tools for the Cray XK System

Heidi Poxon
Cray Inc.

Strengths

Provide a complete solution from instrumentation to measurement to analysis to visualization of data

- **Performance measurement and analysis on large systems**
  - Automatic Profiling Analysis
  - Load Imbalance
  - HW counter derived metrics
  - Predefined trace groups provide performance statistics for libraries called by program (blas, lapack, pgas runtime, netcdf, hdf5, etc.)
  - Observations of inefficient performance
  - Data collection and presentation filtering
  - Data correlates to user source (line number info, etc.)
  - Support MPI, SHMEM, OpenMP, UPC, CAF, OpenACC
  - Access to network counters
  - Minimal program perturbation

The Cray Performance Analysis Framework

- **Supports traditional post-mortem performance analysis**
  - Automatic identification of performance problems
    - Indication of causes of problems
    - Suggestions of modifications for performance improvement

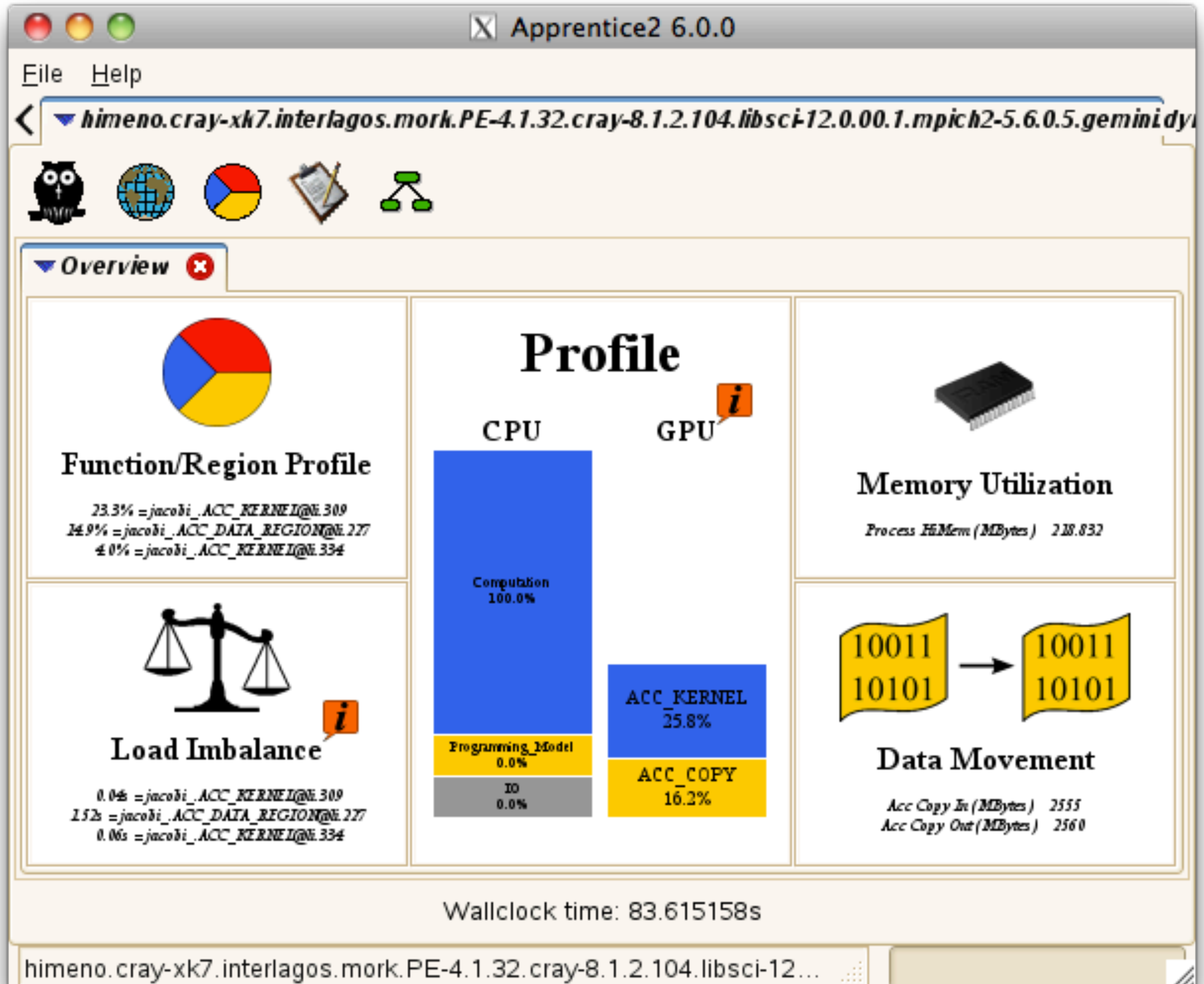
  - pat_build: provides automatic instrumentation
  - CrayPat run-time library collects measurements (transparent to the user)
  - pat_report performs analysis and generates text reports
  - pat_help: online help utility
  - Cray Apprentice2: graphical visualization tool

- **To access software:**
  - module load perftools

Where to Run Instrumented Application

- **By default, data files are written to the execution directory**
- **Default behavior requires file system that supports record locking, such as Lustre (/mnt/snx3/… , /lus/…, /scratch/ …,etc.)**
  - Can use PAT_RT_EXPFILE_DIR to point to existing directory that resides on a high-performance file system if not execution directory
- **Number of files used to store raw data**
  - 1 file created for program with 1 – 256 processes
  - $\sqrt{n}$ files created for program with 257 – *n* processes
  - Ability to customize with PAT_RT_EXPFILE_MAX
- **See intro_craypat(1) man page**

Apprentice2 Overview



Sampling with Line Number information

heidi@limited: /h/heidi — ssh — 81×26

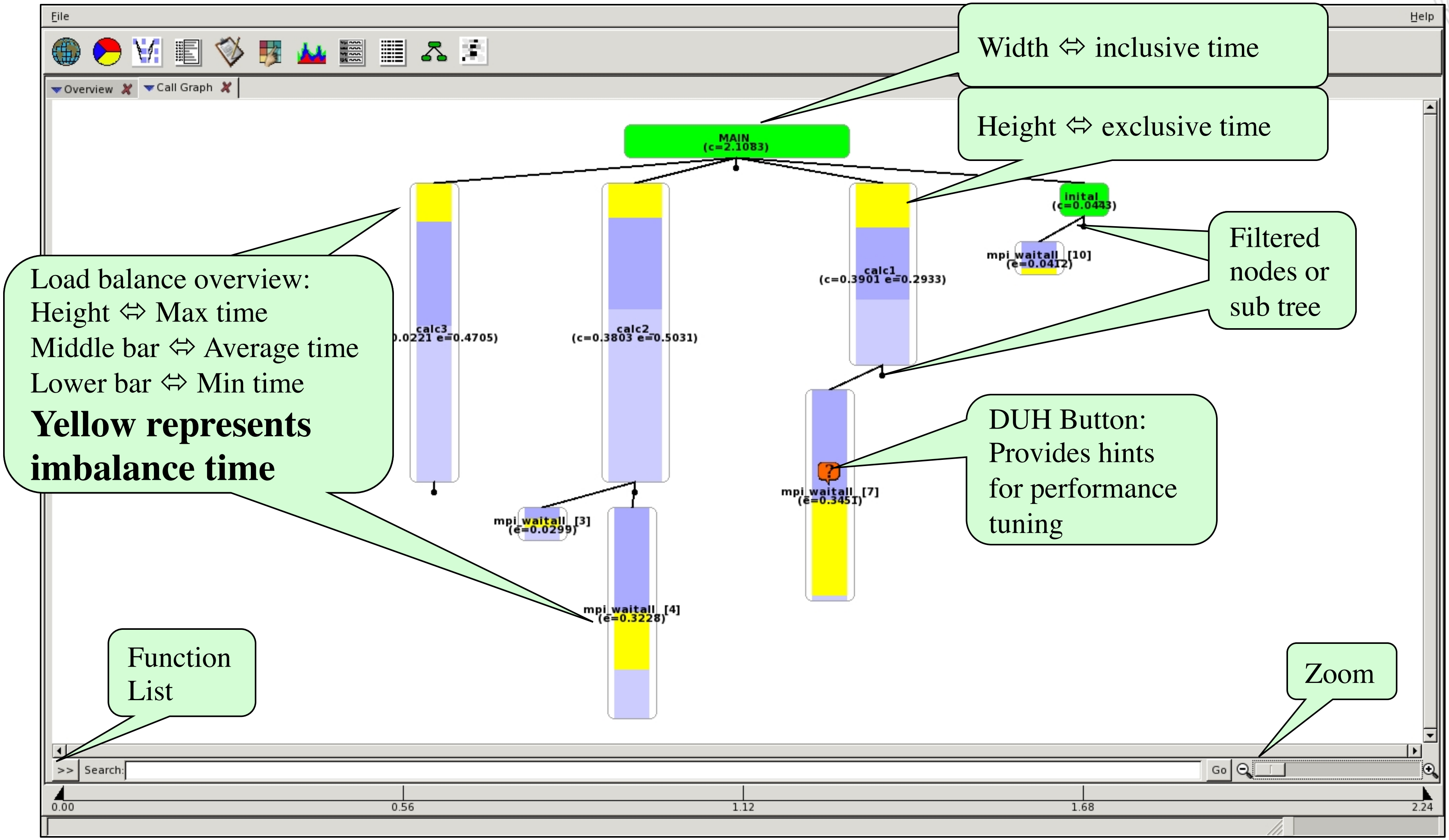
```
Table 2:  Profile by Group, Function, and Line

 Samp%  |   Samp  | Imb.   |  Imb.   |Group
        |         | Samp   | Samp%   | Function
        |         |        |         |  Source
        |         |        |         |   Line
        |         |        |         |    PE=HIDE

 100.0% | 8376.9 |     -- |      -- |Total
|------------------------------------------------------------------------------
|  93.2% | 7804.0 |     -- |      -- |USER
||-----------------------------------------------------------------------------
||  51.7% | 4328.7 |     -- |      -- |calc3_
3|        |        |        |        | heidi/DARPA/cache_util/calc3.do300-ijswap.F
||||---------------------------------------------------------------------------
4|||   15.7% | 1314.4 |  93.6 |   6.8% |line.78
4|||   13.9% | 1167.7 |  98.3 |   7.9% |line.79
4|||   14.5% | 1211.6 |  97.4 |   7.6% |line.80
4|||    1.2% |  103.1 |  26.9 |  21.2% |line.93
4|||    1.1% |   88.4 |  22.6 |  20.8% |line.94
4|||    1.0% |   84.5 |  17.5 |  17.6% |line.95
4|||    1.0% |   86.8 |  33.2 |  28.2% |line.96
4|||    1.3% |  105.0 |  23.0 |  18.4% |line.97
4|||    1.4% |  116.5 |  24.5 |  17.7% |line.98
||||===========================================================================
                                                           144,1         38%
```

MPI Messages By Caller

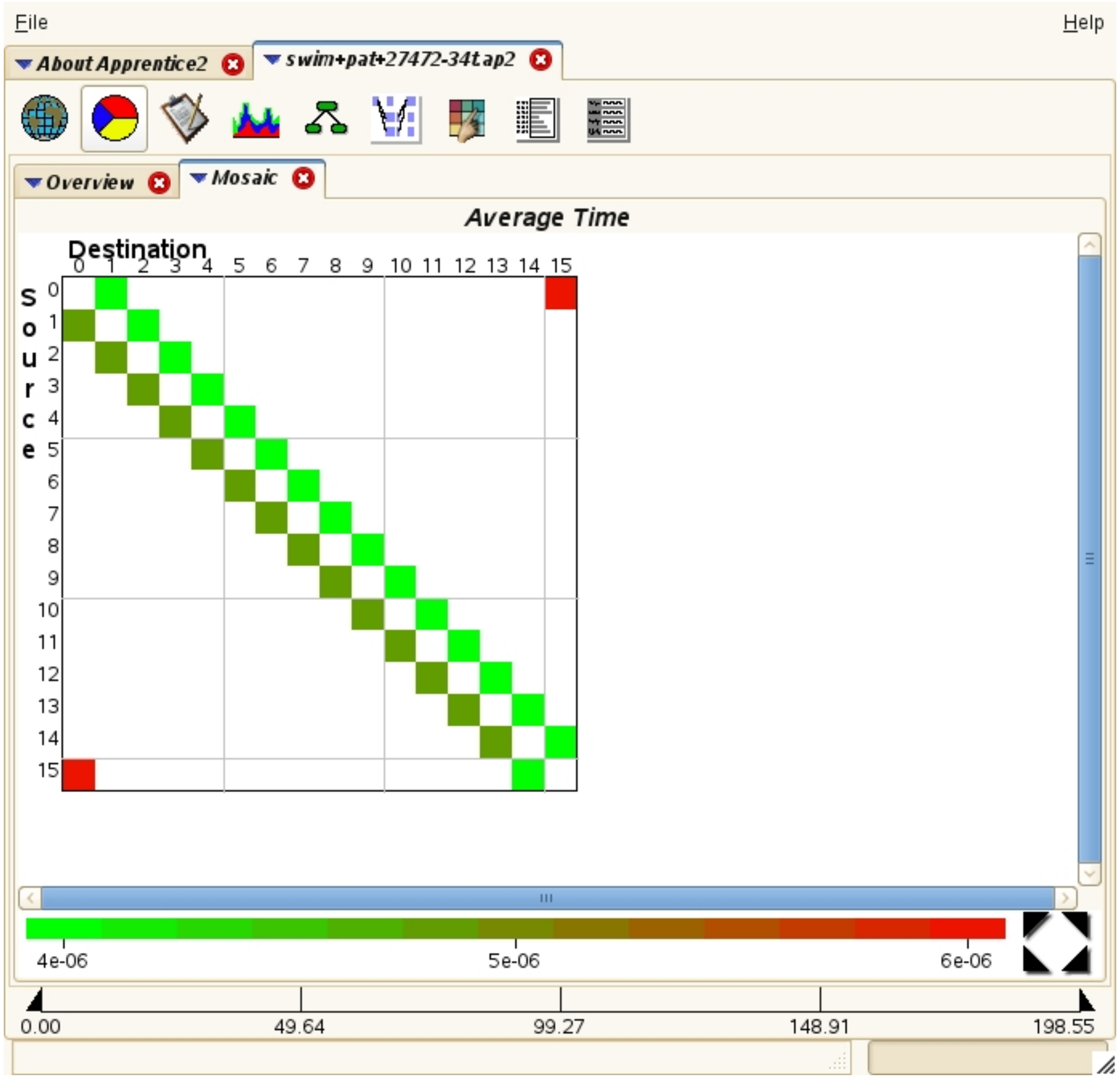
heidi@limited: /h/heidi — ssh — 81×26

```
Table 4:  MPI Message Stats by Caller

    MPI Msg   |     MPI  | MsgSz   | 4KB<=   |Function
      Bytes   |     Msg  |  <16B   | MsgSz   | Caller
              |   Count  | Count   | <64KB   |   PE=[mmm]
              |          |         | Count   |

 140166953.8 |   8890.6 |   339.8 | 8550.8 |Total
|-----------------------------------------------------
| 140166833.8 |   8875.6 |   324.8 | 8550.8 |MPI_ISEND
||----------------------------------------------------
||   78272400.0 |   4850.0 |    75.0 | 4775.0 |calc2_
3|              |          |         |         | shalow_
||||--------------------------------------------------
4|||   78700800.0 |   7200.0 | 2400.0 | 4800.0 |pe.0
4|||   78681600.0 |   4800.0 |     0.0 | 4800.0 |pe.1
4|||   59020800.0 |   4800.0 | 1200.0 | 3600.0 |pe.47
||||==================================================
||   59421800.0 |   3725.0 |   100.0 | 3625.0 |calc1_
3|              |          |         |         | shalow_
||||--------------------------------------------------
4|||   78700800.0 |   7200.0 | 2400.0 | 4800.0 |pe.0
4|||   59011200.0 |   3600.0 |     0.0 | 3600.0 |pe.1
4|||   59011200.0 |   3600.0 |     0.0 | 3600.0 |pe.24
||||==================================================
                                                          624,3          79%
```

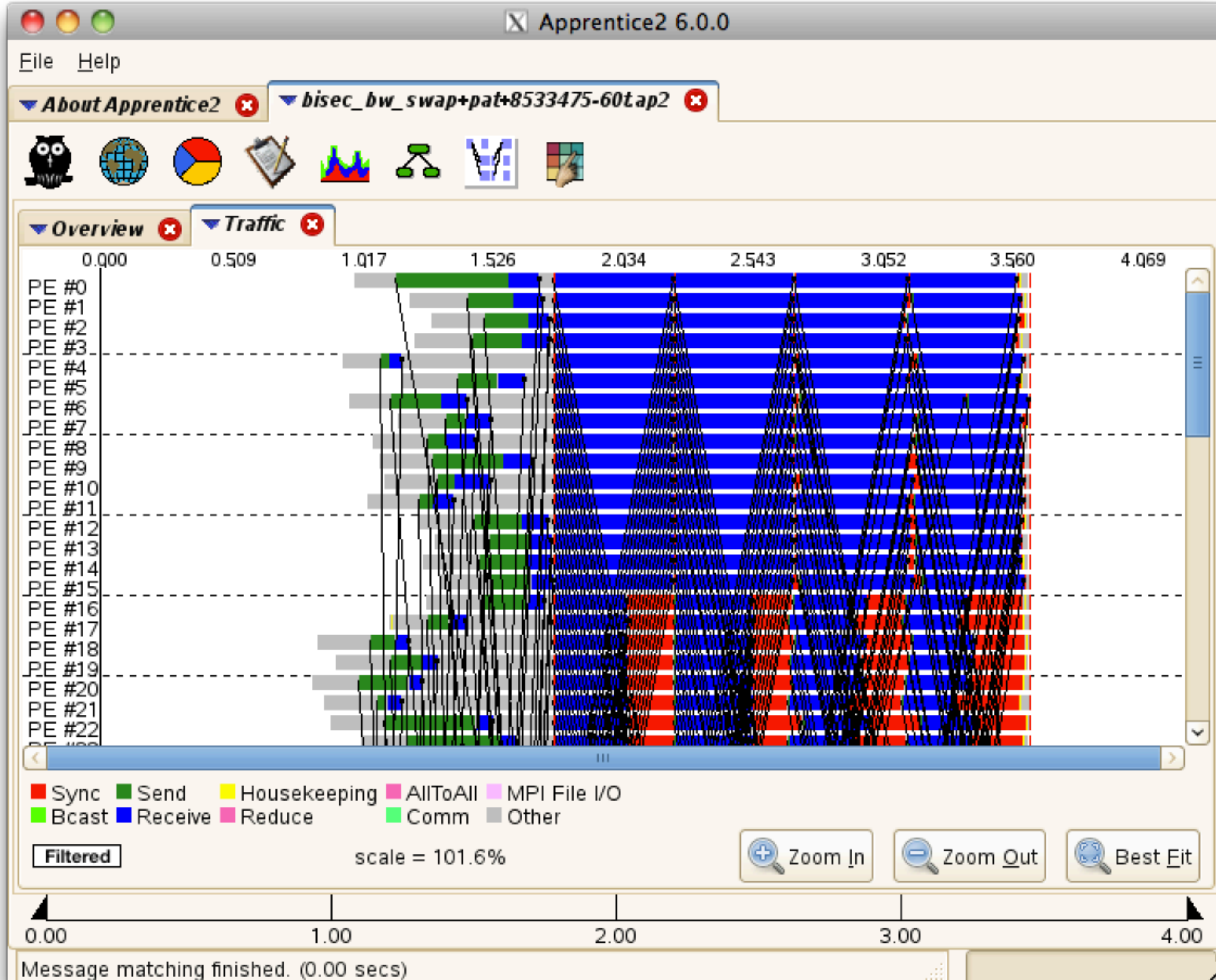
Call Tree View



Mosaic View – Shows Communication Pattern



Traffic Report – MPI Communication Timeline



Porting to a Hybrid or Many-core System

A Porting and Optimization Strategy for Hybrid and Many-core Systems

- Maximize on-node communication between MPI ranks
- Relieve on-node shared resource contention by pairing threads or processes that perform different work (for example computation with off-node communication) on the same node
- Add parallelism to MPI ranks to take advantage of cores within a node while minimizing network injection contention
- Accelerate work intensive parallel loops

Grid Detection and Rank Reordering Suggestions

Automatic Communication Grid Detection

- **Analyze runtime performance data to identify grids in a program to maximize on-node communication**
  - Example: nearest neighbor exchange in 2 dimensions
    - Sweep3d uses a 2-D grid for communication
- **Determine whether or not a custom MPI rank order will produce a significant performance benefit**
- **Grid detection is helpful for programs with significant point-to-point communication**
- **Doesn't interfere with MPI collective communication optimizations**

Automatic Communication Grid Detection (2)

- **Tools produce a custom rank order if it's beneficial based on grid size, grid order and cost metric**
- **Heuristics available for:**
  - MPI sent message statistics
  - User time (time spent in user functions) – can be used for PGAS codes
  - Hybrid of sent message and user time)
- **Summarized findings in report**
- **Available with sampling or tracing**
- **Describe how to re-run with custom rank order**

MPI Rank Order Observations

```
Table 1:  Profile by Function Group and Function

 Time%  |       Time |     Imb. |  Imb.  |   Calls |Group
        |            |     Time |  Time% |         | Function
        |            |          |        |         |  PE=HIDE

 100.0% | 463.147240 |       -- |     -- | 21621.0 |Total
|-----------------------------------------------------------------------
|  52.0% | 240.974379 |       -- |     -- | 21523.0 |MPI
||----------------------------------------------------------------------
||  47.7% | 221.142266 | 36.214468 |  14.1% | 10740.0 |mpi_recv
||   4.3% |  19.829001 | 25.849906 |  56.7% | 10740.0 |MPI_SEND
||======================================================================
|  43.3% | 200.474690 |       -- |     -- |    32.0 |USER
||----------------------------------------------------------------------
||  41.0% | 189.897060 | 58.716197 |  23.6% |    12.0 |sweep_
||   1.6% |   7.579876 |  1.899097 |  20.1% |    12.0 |source_
||======================================================================
|   4.7% |  21.698147 |       -- |     -- |    39.0 |MPI_SYNC
||----------------------------------------------------------------------
|   4.3% |  20.091165 | 20.005424 |  99.6% |    32.0 | mpi_allreduce_(sync)
||======================================================================
|   0.0% |   0.000024 |       -- |     -- |    27.0 |SYSCALL
|=======================================================================
```

MPI Rank Order Observations (2)

```
MPI Grid Detection:

    There appears to be point-to-point MPI communication in a 96 X 8
    grid pattern. The 52% of the total execution time spent in MPI
    functions might be reduced with a rank order that maximizes
    communication between ranks on the same node. The effect of several
    rank orders is estimated below.

    A file named MPICH_RANK_ORDER.Grid was generated along with this
    report and contains usage instructions and the Custom rank order
    from the following table.

         Rank     On-Node      On-Node   MPICH_RANK_REORDER_METHOD
        Order    Bytes/PE    Bytes/PE%
                              of Total
                              Bytes/PE

        Custom   2.385e+09      95.55%  3
           SMP   1.880e+09      75.30%  1
          Fold   1.373e+06       0.06%  2
    RoundRobin   0.000e+00       0.00%  0
```

MPICH_RANK_ORDER File

```
# The 'Custom' rank order in this file targets nodes with multi-core
# processors, based on Sent Msg Total Bytes collected for:
#
# Program:      /lus/nid00030/heidi/sweep3d/mod/sweep3d.mpi
# Ap2 File:     sweep3d.mpi+pat+27054-89t.ap2
# Number PEs:   48
# Max PEs/Node: 4
#
# To use this file, make a copy named MPICH_RANK_ORDER, and set the
# environment variable MPICH_RANK_REORDER_METHOD to 3 prior to
# executing the program.
#
# The following table lists rank order alternatives and the grid_order
# command-line options that can be used to generate a new order.
…
```

Auto-Generated MPI Rank Order File

```
# The
'USER_Time_hybrid'
rank order in this
file targets nodes
with multi-core
# processors, based on
Sent Msg Total Bytes
collected for:
#
# Program:      /lus/
nid00023/malice/
craypat/WORKSHOP/bh2o-
demo/Rank/sweep3d/src/
sweep3d
# Ap2 File:
sweep3d.gmpi-u.ap2
# Number PEs:   768
# Max PEs/Node: 16
#
# To use this file,
make a copy named
MPICH_RANK_ORDER, and
set the
# environment variable
MPICH_RANK_REORDER_MET
HOD to 3 prior to
# executing the
program.
#
0,532,64,564,32,572,96
,540,8,596,72,524,40,6
04,24,588
104,556,16,628,80,636,
56,620,48,516,112,580,
88,548,120,612
1,403,65,435,33,411,97
,443,9,467,25,499,105,
507,41,475
73,395,81,427,57,459,1
7,419,113,491,49,387,8
9,451,121,483
6,436,102,468,70,404,3
8,412,14,444,46,476,11
0,508,78,500
86,396,30,428,62,460,5
4,492,118,420,22,452,9
4,388,126,484
129,563,193,531,161,57
1,225,539,241,595,233,
523,249,603,185,555
153,587,169,627,137,63
5,201,619,177,515,145,
579,209,547,217,611
7,405,71,469,39,437,10
3,413,47,445,15,509,79
,477,31,501
111,397,63,461,55,429,
87,421,23,493,119,389,
95,453,127,485
134,402,198,434,166,41
0,230,442,238,466,174,
506,158,394,246,474
190,498,254,426,142,45
8,150,386,182,418,206,
490,214,450,222,482
128,533,192,541,160,56
5,232,525,224,573,240,
597,184,557,248,605
168,589,200,517,152,62
9,136,549,176,637,144,
621,208,581,216,613
5,439,37,407,69,447,10
1,415,13,471,45,503,29
,479,77,511
53,399,85,431,21,463,6
1,391,109,423,93,455,1
17,495,125,487
2,530,34,562,66,538,98
,522,10,570,42,554,26,
594,50,602
18,514,74,586,58,626,8
2,546,106,634,90,578,1
14,618,122,610
135,315,167,339,199,34
7,259,307,231,371,239,
379,191,331,247,299
175,363,159,323,143,35
5,255,291,207,275,183,
283,151,267,215,223
133,406,197,438,165,47
0,229,414,245,446,141,
478,237,502,253,398
157,510,189,462,173,43
0,205,390,149,422,213,
454,181,494,221,486
130,316,260,340,194,37
2,162,348,226,308,234,
380,242,332,250,300
202,364,186,324,154,35
6,138,292,170,276,178,
284,210,218,268,146
4,535,36,543,68,567,10
0,527,12,599,44,575,28
,559,76,607
52,591,20,631,60,639,8
4,519,108,623,92,551,1
16,583,124,615
3,440,35,432,67,400,99
,408,11,464,43,496,27,
472,51,504
19,392,75,424,59,456,8
3,384,107,416,91,488,1
15,448,123,480
132,401,196,441,164,40
9,228,433,236,465,204,
473,244,393,188,497
252,505,140,425,212,45
7,156,385,172,417,180,
449,148,489,220,481
131,534,195,542,163,56
6,227,526,235,574,203,
598,243,558,187,606
251,590,211,630,179,63
8,139,622,155,550,171,
518,219,582,147,614
761,660,737,652,705,66
8,745,692,673,700,641,
684,713,644,753,724
729,732,681,756,721,71
6,764,676,697,748,689,
657,740,665,649,708
760,528,736,536,704,56
0,744,520,672,568,712,
592,752,552,640,600
728,584,680,624,720,51
2,696,632,688,616,664,
544,608,656,648,576
762,659,738,651,706,66
7,746,643,714,691,674,
699,754,683,730,723
722,731,763,658,642,75
5,739,675,707,650,682,
715,698,666,690,747
257,345,265,313,281,30
5,273,337,609,369,577,
377,617,329,513,529
545,297,633,361,625,32
1,585,537,601,289,553,
353,593,521,569,561
256,373,261,341,264,34
9,280,317,272,381,269,
309,285,333,277,365
352,301,320,325,288,35
7,328,304,360,312,376,
293,296,368,336,344
258,338,266,346,282,31
4,274,370,766,306,710,
378,742,330,678,362
646,298,750,322,718,35
4,758,290,734,662,686,
670,726,702,694,654
262,375,263,343,270,31
1,271,351,286,319,278,
342,287,350,279,374
294,318,358,383,359,31
0,295,382,326,303,327,
367,366,335,302,334
765,661,709,663,741,65
3,711,669,767,655,743,
671,749,695,679,703
677,727,751,693,647,70
1,717,687,757,685,733,
725,719,735,645,759
```

Approach to Adding Parallelism

1. **Identify possible accelerator kernels**
   - Determine where to add additional levels of parallelism
     - Assumes MPI application is functioning correctly on X86
     - Find top serial work-intensive loops (perftools + CCE loop work estimates)
2. **Perform parallel analysis, scoping and vectorization**
   - Split loop work among threads
     - Do parallel analysis and restructuring on targeted high level loops
     - Use CCE loopmark feedback, Reveal loopmark and source browsing
3. **Move to OpenMP and then to OpenACC**
   - Add parallel directives and acceleration extensions
     - Insert OpenMP directives (Reveal scoping assistance)
     - Run on X86 to verify application and check for performance improvements
     - Convert desired OpenMP directives to OpenACC
4. **Analyze performance from optimizations**

Step 1 - Identify possible accelerator kernels

Loop Work Estimates

- **Helps identify high-level serial loops to parallelize**
  - Based on runtime analysis, approximates how much work exists within a loop
  - Provides min, max and average trip counts that can be used to approximate work and help carve up loop on GPU

Collecting Loop Statistics

- Load PrgEnv-cray module
- Load perftools module

- Compile **AND** link with –h profile_generate

- Instrument binary for tracing
  - pat_build –w my_program

- Run application

- Create report with loop statistics
  - pat_report my_program.xf > loops_report

Example Report – Inclusive Loop Time

Table 2: Loop Stats by Function (from -hprofile_generate)

Loop Incl Time Total	Loop Hit	Loop Trips Avg	Loop Trips Min	Loop Trips Max	Function=/.LOOP[.] PE=HIDE
8.995914	100	25	0	25	sweepy_.LOOP.1.li.33
8.995604	2500	25	0	25	sweepy_.LOOP.2.li.34
8.894750	50	25	0	25	sweepz_.LOOP.05.li.49
8.894637	1250	25	0	25	sweepz_.LOOP.06.li.50
4.420629	50	25	0	25	sweepx2_.LOOP.1.li.29
4.420536	1250	25	0	25	sweepx2_.LOOP.2.li.30
4.387534	50	25	0	25	sweepx1_.LOOP.1.li.29
4.387457	1250	25	0	25	sweepx1_.LOOP.2.li.30
2.523214	187500	107	0	107	riemann_.LOOP.2.li.63
1.541299	20062500	12	0	12	riemann_.LOOP.3.li.64
0.863656	1687500	104	0	108	parabola_.LOOP.6.li.67

Step 2 - Perform parallel analysis, scoping and vectorization
&
Step 3 - Move to OpenMP and then to OpenACC


Reveal 0.1
File
Help
About Reveal
CRAY
REVEAL

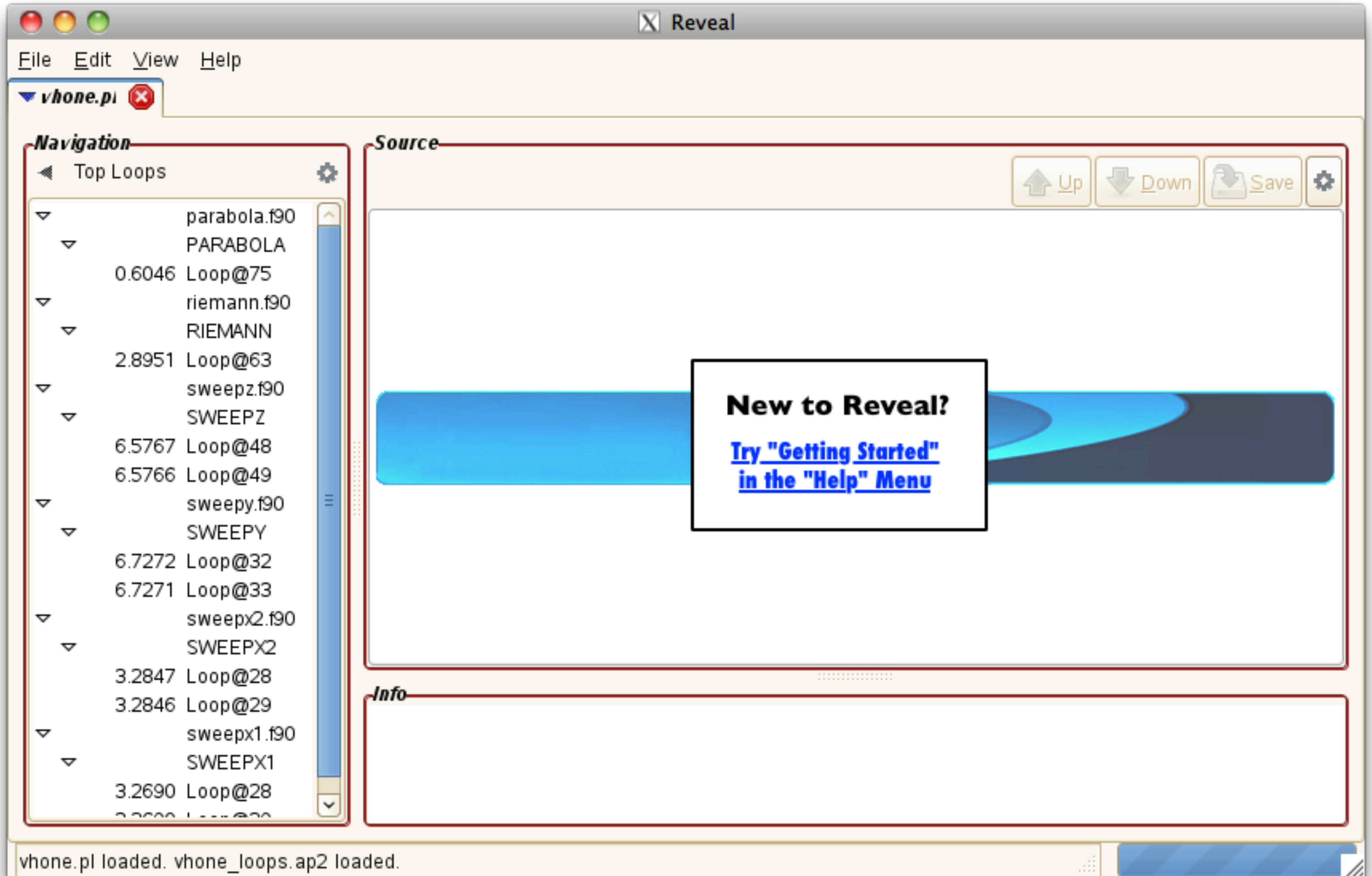
Reveal

New code analysis and restructuring assistant…

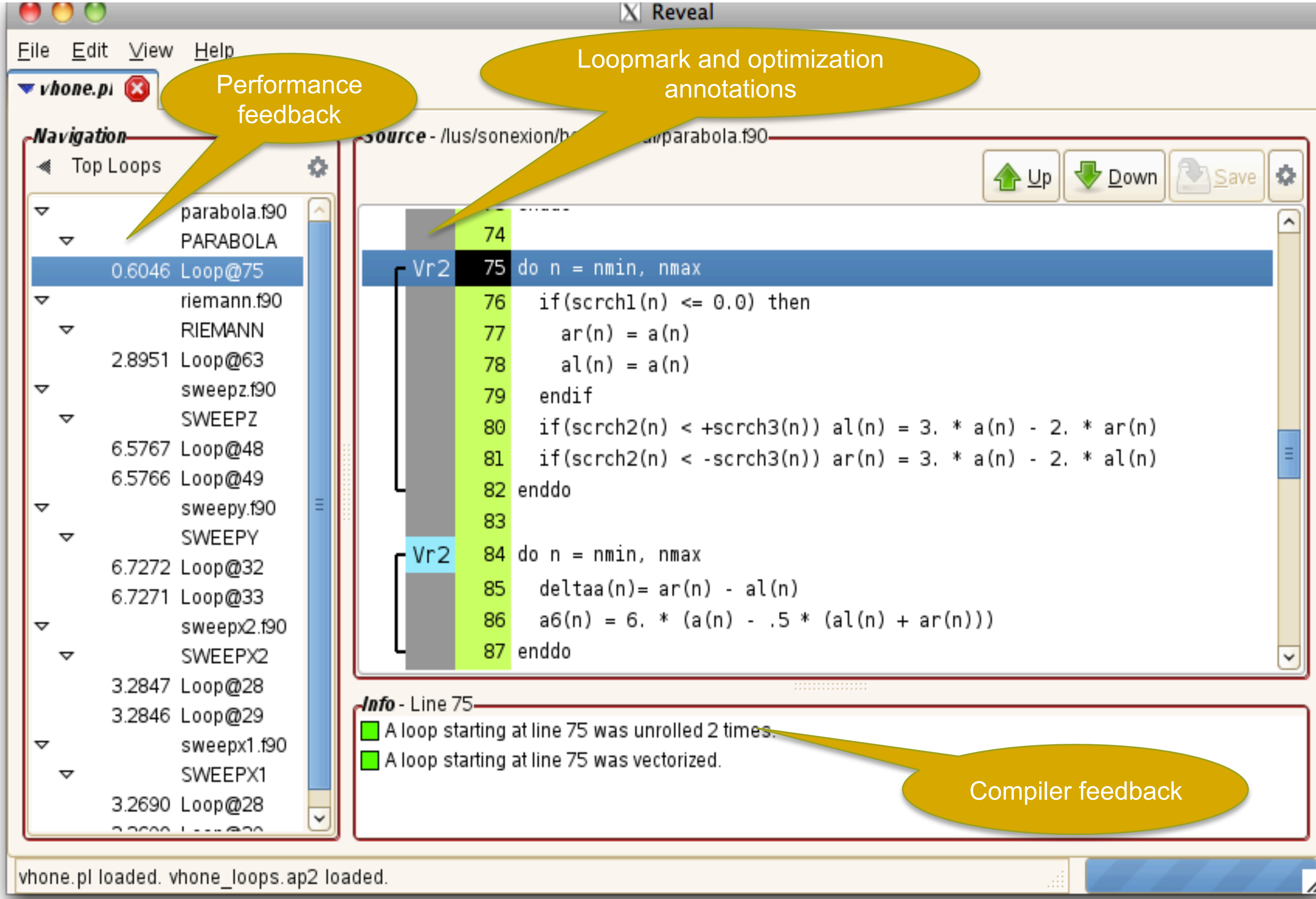
- **Uses both the performance toolset and CCE's program library functionality to provide static and runtime analysis information**

- **Key Features**
  - Annotated source code with compiler optimization information
    - Feedback on critical dependencies that prevent optimizations
  - Scoping analysis
    - Identify, shared, private and ambiguous arrays
      - Allow user to privatize ambiguous arrays
      - Allow user to override dependency analysis
  - Source code navigation based on performance data collected through CrayPat

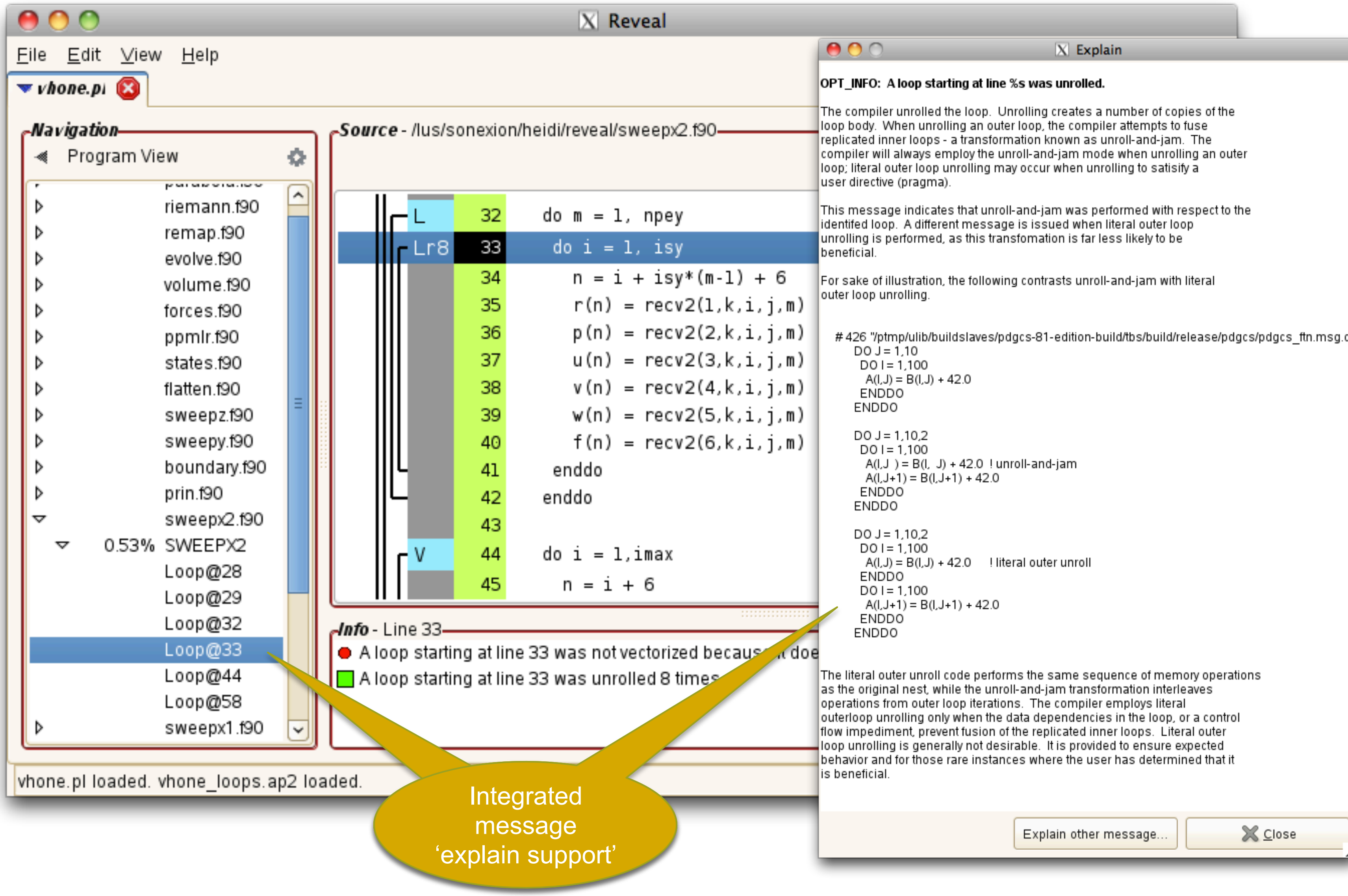
Reveal with Loop Work Estimates



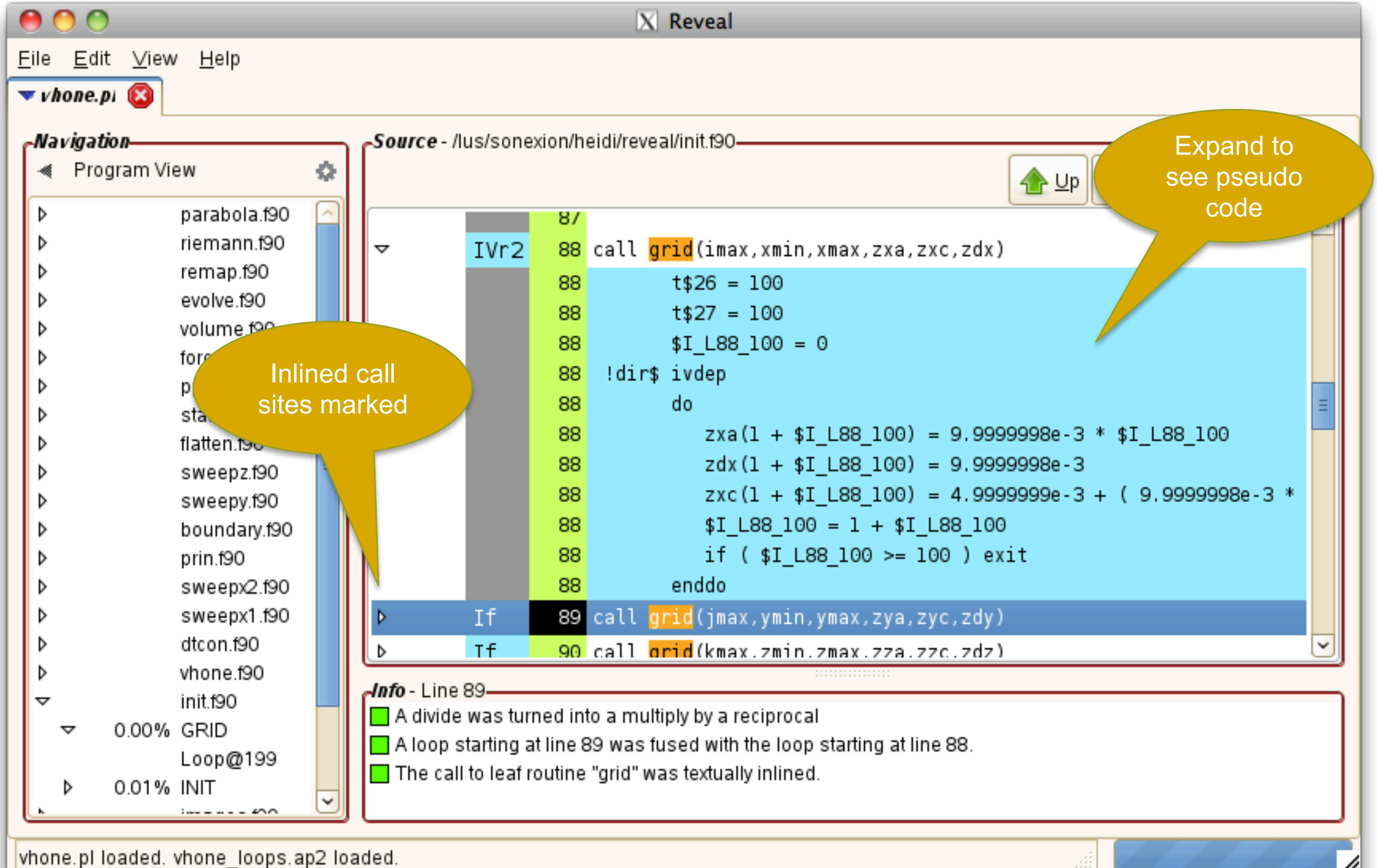
Visualize Loopmark with Performance Information



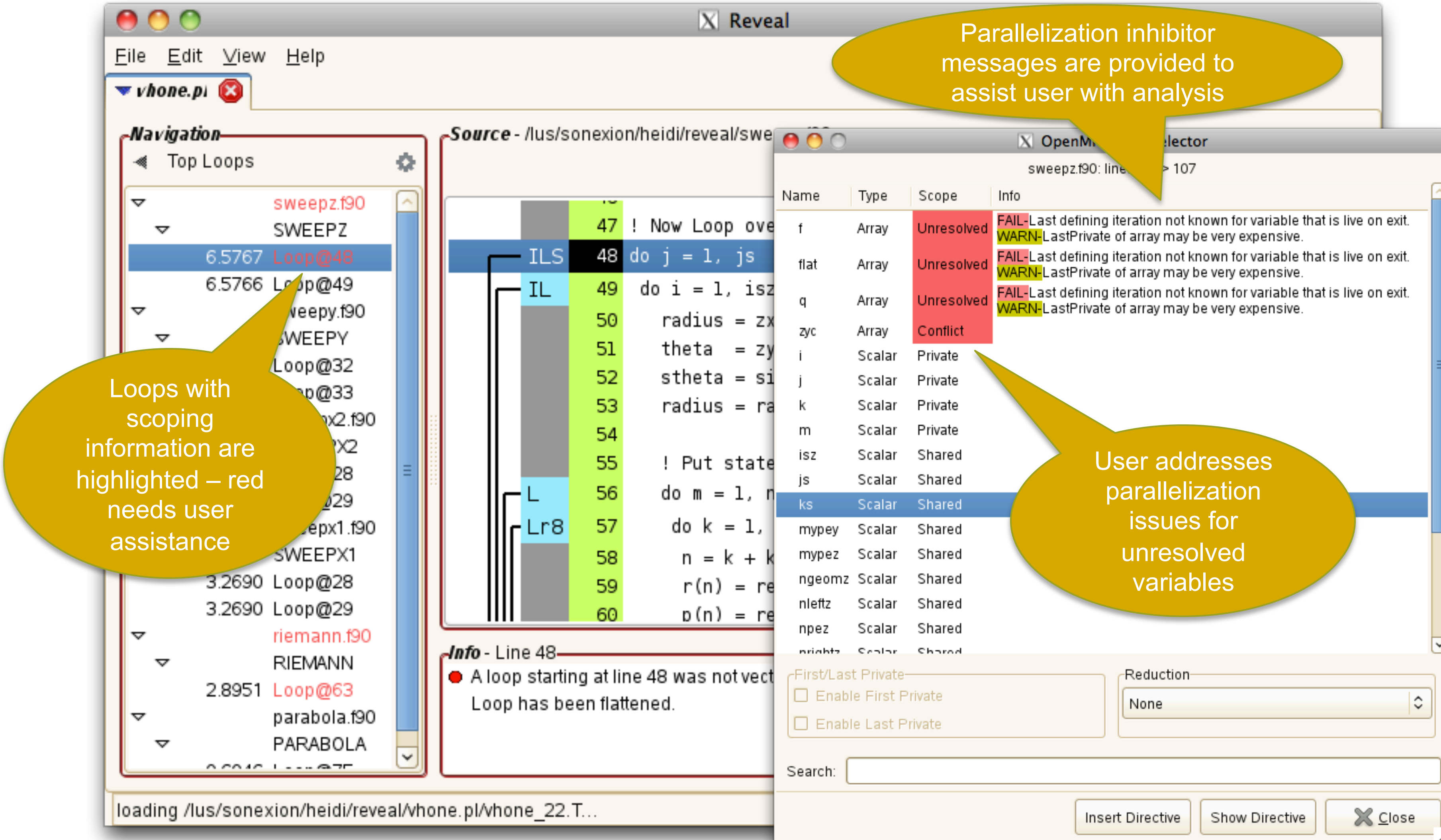
Visualize CCE's Loopmark with Performance Profile (2)



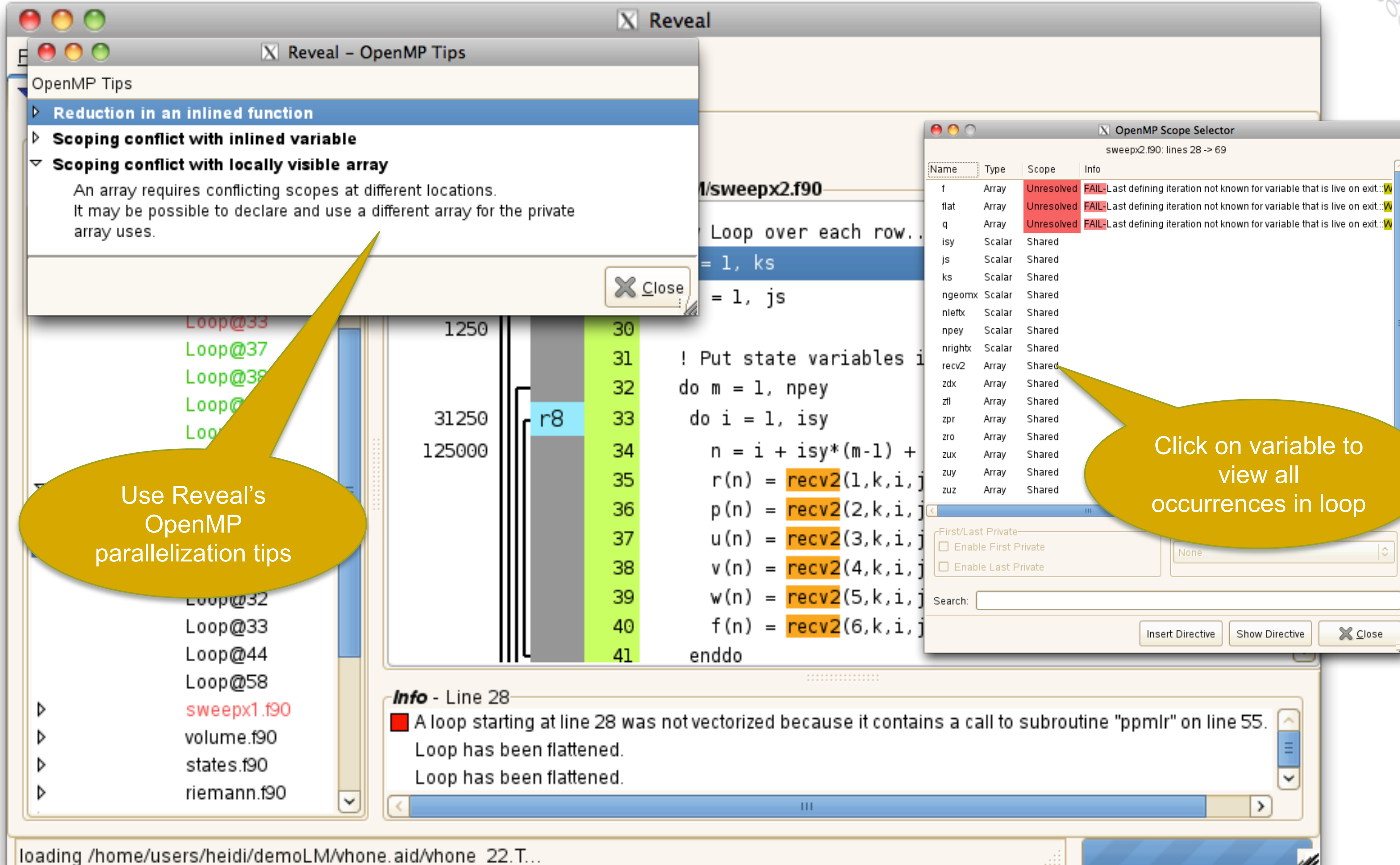
View Pseudo Code for Inlined Functions



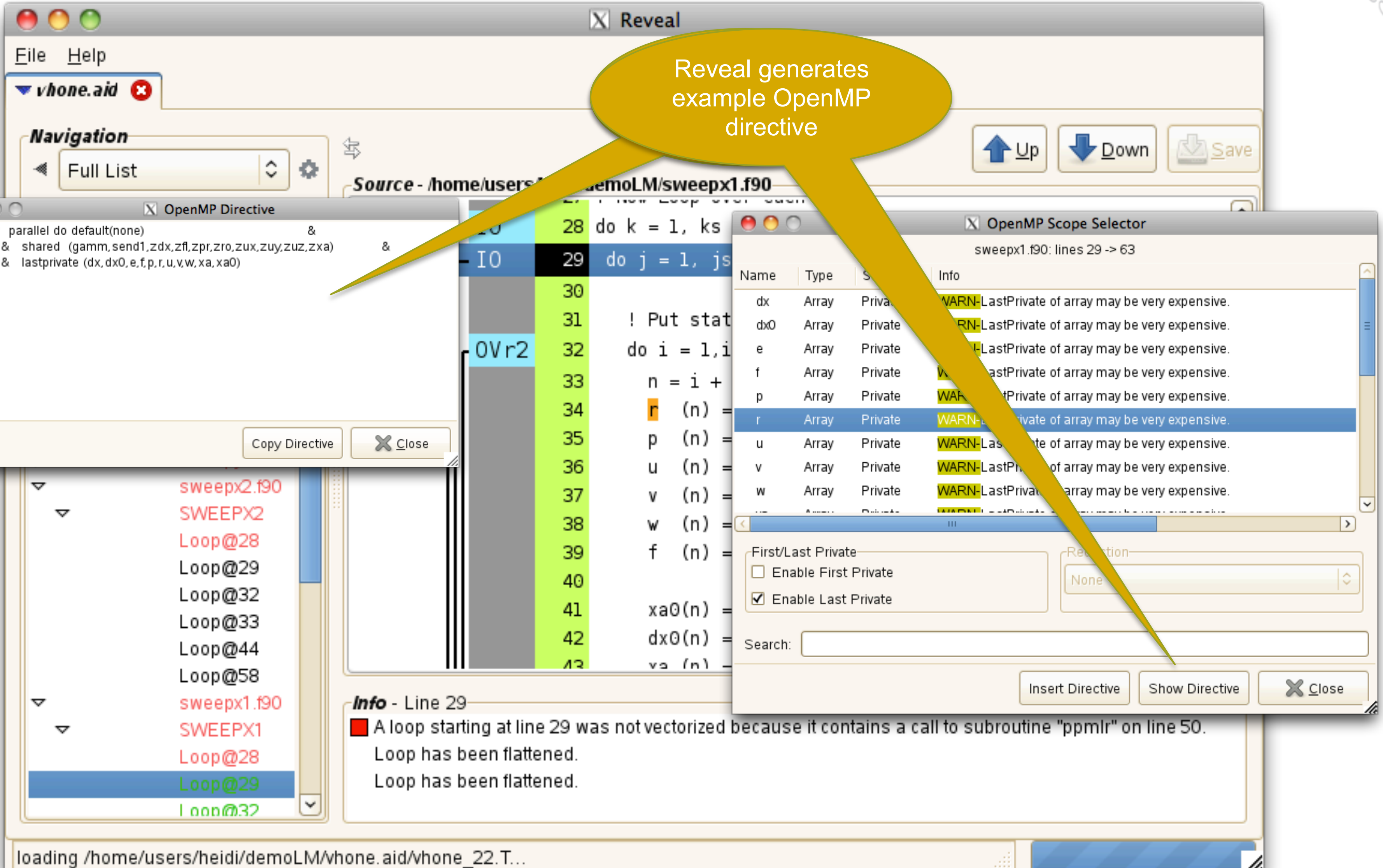
Scoping Assistance – Review Scoping Results



Scoping Assistance – User Resolves Issues



Scoping Assistance – Generate Directive



Step 4 – Analyze performance of accelerator regions

Programming Models Supported for the GPU

- Goal is to provide whole program analysis for programs written for x86 or hybrid x86 + GPUs
- Development focus is on support of CCE with OpenACC directives
- Cray XK programming models supported
  - OpenACC, CUDA, PGI acc (or OpenACC) directives

Collecting GPU Statistics for OpenACC

- **Load PrgEnv-cray module**
- **Load perftools module**

- **Instrument binary for tracing and collecting GPU statistics**
  - pat_build –u –g mpi,blas my_program

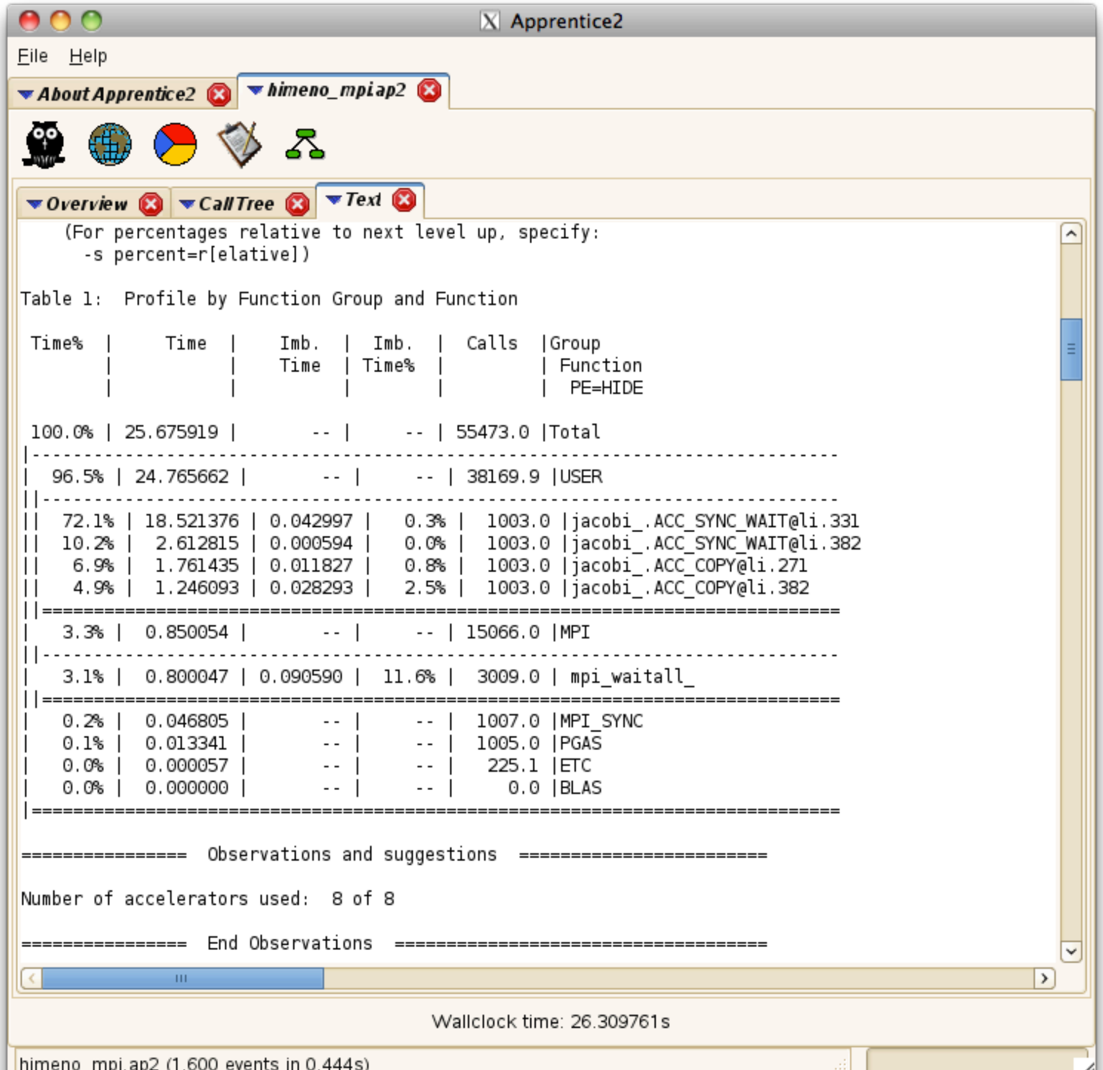
- **Run application**

- **Create report with GPU statistics**
  - pat_report my_program.xf > GPU_stats_report

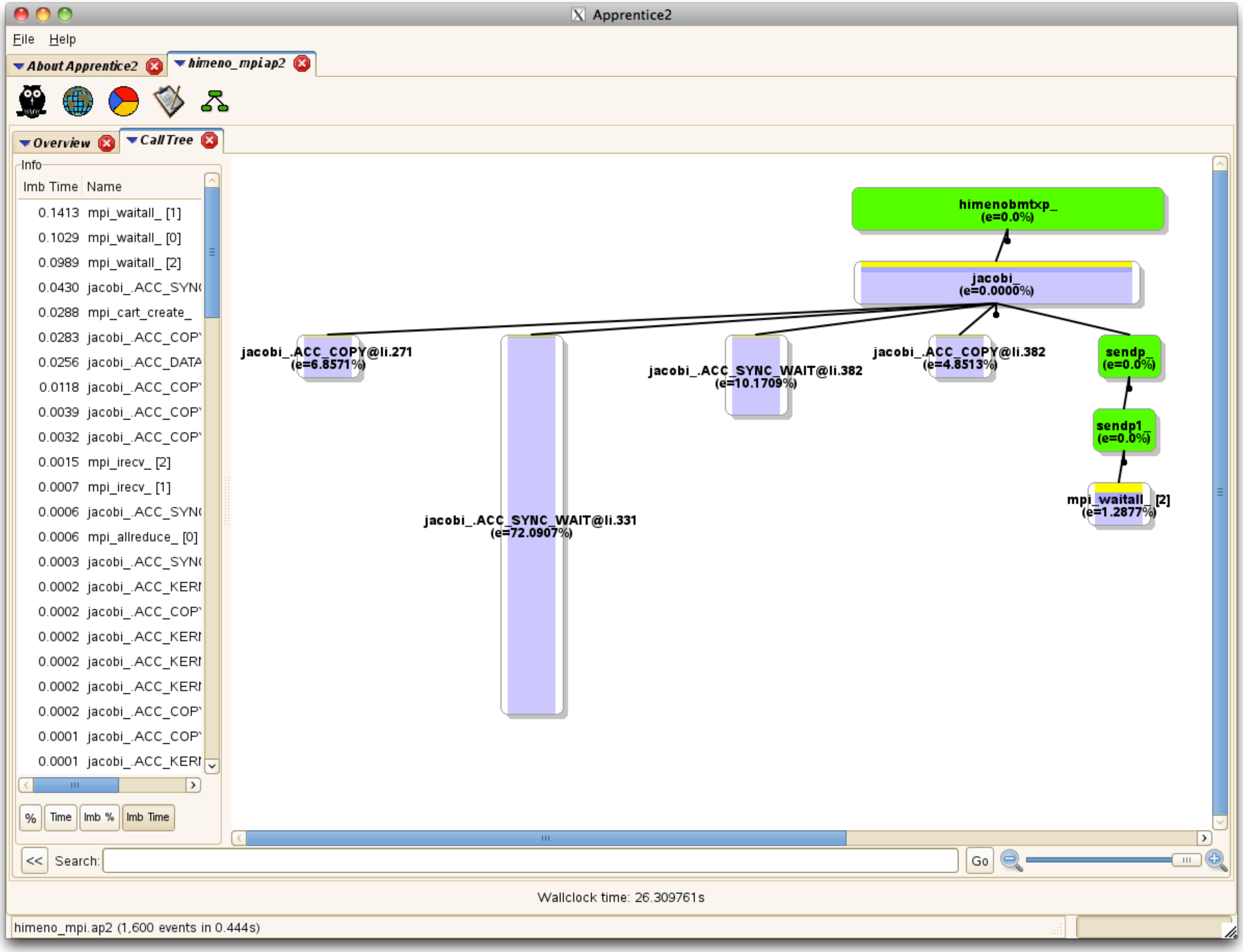
Analyze Performance of Accelerated Program

- **Statistics collected for programs with OpenACC directives**
  - Number of GPUs used in the job
  - Host time for kernel launches, data copies and synchronization with the accelerator
  - Accelerator time for kernel execution and data copies
  - Data copy size to and from the accelerator
  - Kernel grid size
  - Block size
  - Amount of shared memory dynamically allocated for kernel
  - GPU performance counters
  - Derived metrics based on performance counters

Profile with GPU



Call Tree with GPU regions



Example Accelerator Statistics

```
Table 1:  Time and Bytes Transferred for Accelerator Regions
  Host  |     Host  |      Acc  | Acc Copy  | Acc Copy  | Calls   |Calltree
 Time%  |     Time  |     Time  |       In  |      Out  |         | PE=HIDE
        |           |           | (MBytes)  | (MBytes)  |         |
 100.0% |    2.750 |    2.015 |  2812.760 |    13.568 |    103 |Total
|------------------------------------------------------------------------------------------------------------------------
| 100.0% |    2.750 |    2.015 |  2812.760 |    13.568 |    103 |lbm3d2p_d_
|        |          |          |           |           |        | lbm3d2p_d_.ACC_DATA_REGION@li.104
|||----------------------------------------------------------------------------------------------------------------------
3||  63.5% |    1.747 |    1.747 |  2799.192 |        -- |      1 |lbm3d2p_d_.ACC_COPY@li.104
3||  22.1% |    0.609 |    0.088 |    12.304 |    12.304 |     36 |streaming_
||||---------------------------------------------------------------------------------------------------------------------
4|||  20.6% |    0.566 |    0.046 |    12.304 |    12.304 |     27 |streaming_exchange_
5|||        |          |          |           |           |        | streaming_exchange_.ACC_DATA_REGION@li.526
6|||  18.8% |    0.517 |       -- |        -- |        -- |      1 |  streaming_exchange_.ACC_DATA_REGION@li.526(exclusive)
4|||   1.6% |    0.043 |    0.042 |        -- |        -- |      9 |streaming_.ACC_DATA_REGION@li.907
5|||   1.1% |    0.031 |    0.031 |        -- |        -- |      4 | streaming_.ACC_REGION@li.909
6|||   1.1% |    0.031 |       -- |        -- |        -- |      1 |  streaming_.ACC_REGION@li.909(exclusive)
||||=====================================================================================================================

...
```

Example Kernel Statistics

```
Table 2:  Kernel Stats for Accelerator Regions
  Avg   |  Avg   |  Avg   |   Avg    |   Avg    |   Avg    |Function
 Grid   | Grid   | Grid   | Block    | Block    | Block    |
    X   |    Y   |    Z   | X Dim    | Y Dim    | Z Dim    |
  Dim   |  Dim   |  Dim   |          |          |          |
|-----------------------------------------------------------------------------------------------
| 62163 |      1 |      1 |   1024 |        1 |        1 |streaming_.ACC_KERNEL@li.909
|   402 |      1 |      1 |    128 |        1 |        1 |grad_exchange_.ACC_KERNEL@li.443
|   402 |      1 |      1 |    128 |        1 |        1 |grad_exchange_.ACC_KERNEL@li.467
|   402 |      1 |      1 |    128 |        1 |        1 |grad_exchange_.ACC_KERNEL@li.476
|   402 |      1 |      1 |    128 |        1 |        1 |grad_exchange_.ACC_KERNEL@li.500
|   400 |      1 |      1 |    512 |        1 |        1 |cal_velocity_.ACC_KERNEL@li.1126
|   400 |      1 |      1 |    512 |        1 |        1 |collisiona_.ACC_KERNEL@li.474
|   400 |      1 |      1 |    128 |        1 |        1 |collisionb_.ACC_KERNEL@li.597
|   400 |      1 |      1 |    128 |        1 |        1 |wall_boundary_.ACC_KERNEL@li.973
|   400 |      1 |      1 |    128 |        1 |        1 |collisionb_.ACC_KERNEL@li.629
|   400 |      1 |      1 |    512 |        1 |        1 |recolor_.ACC_KERNEL@li.823
|   128 |      1 |      1 |     64 |        1 |        1 |injection_.ACC_KERNEL@li.1281
|   128 |      1 |      1 |    128 |        1 |        1 |streaming_exchange_.ACC_KERNEL@li.829
|   128 |      1 |      1 |    128 |        1 |        1 |streaming_exchange_.ACC_KERNEL@li.729
|   128 |      1 |      1 |    128 |        1 |        1 |streaming_exchange_.ACC_KERNEL@li.641
|   128 |      1 |      1 |    128 |        1 |        1 |streaming_exchange_.ACC_KERNEL@li.538
|   101 |      1 |      1 |    128 |        1 |        1 |collisionb_.ACC_KERNEL@li.612
|   101 |      1 |      1 |    128 |        1 |        1 |set_boundary_micro_press_.ACC_KERNEL@li.299
|   101 |      1 |      1 |    128 |        1 |        1 |set_boundary_macro_press2_.ACC_KERNEL@li.259
|    14 |      1 |      1 |    256 |        1 |        1 |streaming_.ACC_KERNEL@li.919
|===============================================================================================
```

Questions
?

Files Generated and the Naming Convention

File Suffix	Description
a.out+pat	Program instrumented for data collection
a.out…s.xf	Raw data for sampling experiment, available after application execution
a.out…t.xf	Raw data for trace (summarized or full) experiment, available after application execution
a.out…st.ap2	Processed data, generated by pat_report, contains application symbol information
a.out…s.apa	Automatic profiling pnalysis template, generated by pat_report (based on pat_build –O apa experiment)
a.out+apa	Program instrumented using .apa file
MPICH_RANK_ORDER.Custom	Rank reorder file generated by pat_report from automatic grid detection an reorder suggestions

GPU HW Performance Counters

Accelerator Hardware Performance Counters

- **Enable collection similarly to CPU counter collection:**
  - CPU: PAT_RT_HWPC=*group or events*
  - GPU: PAT_RT_ACCPC=*group or events*

- **Enabling GPU counters causes change in behavior of application:**
  - Host needs to synchronize with the accelerator at each event (since accelerator executes asynchronously with the host)

  - Can be seen through accelerator table
    - No counters: time spent waiting for kernel to complete is shown with ACC_SYNC_WAIT (a synchronization created by the compiler)

    - Counters: perftools syncs with accelerator with each event so Host Time is exclusive time for the containing region (since waiting occurs within the event's trace point instead of in the compiler sync)

Accelerator HW Counter Groups

- **A predefined set of groups has been created for ease of use**
  - Combines events that can be counted together
- **ACCPC groups start at 1000, and will be incremented by 100 as new families of accelerators are supported**
- **Specify group by number or name**
  - PAT_RT_ACCPC=1000 *OR*
  - PAT_RT_ACCPC=inst_exec_gst
- **See accpc(5) and accpc_k20(5) man pages for list of groups and their descriptions**