Cray Performance Measurement and Analysis Tools

Heidi Poxon
Technical Lead
Programming Environment
Cray Inc.

March 2016

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA and YARCDATA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Other names and brands may be claimed as the property of others. Other product and service names mentioned herein are the trademarks of their respective owners.

Topics

- **Introduction to Cray performance tools, including:**
  - CrayPat-lite and CrayPat interfaces
  - Collecting performance counter information
  - MPI rank reorder
  - Cray Apprentice2
- **Recent enhancements**
- **Support for GPUs**

Strengths

- Whole program analysis across many nodes
- Novice and advanced user interfaces
- Support for MPI, SHMEM, OpenMP, UPC, CAF, OpenACC, CUDA
- Load Imbalance detection
- HW counter derived metrics

Strengths (2)

- Performance statistics for libraries called by program (BLAS, LAPACK, PETSc, NetCDF, HDF5, etc.)
- Observations of inefficient performance
- Data correlation to user source (line number, function)
- Minimal program perturbation

Two Interfaces to the Performance Tools

- **Support traditional post-mortem performance analysis**
  - Indication of causes of problems
  - Suggestions of modifications for performance improvement
- **CrayPat-lite for first time users**
- **CrayPat for in-depth performance investigation and tuning assistance**

Example CrayPat-lite Output

```
CrayPat/X:  Version 6.1.4.12457 Revision 12457 (xf 12277)  02/26/14 13:58:24
Experiment:                   lite  lite/sample_profile
Number of PEs (MPI ranks):    8164
Numbers of PEs per Node:        16  PEs on each of  510  Nodes
                                 4  PEs on    1  Node
Numbers of Threads per PE:       1
Number of Cores per Socket:      8
Execution start time:  Fri Feb 28 23:06:31 2014
System name and speed:  hera2 2100 MHz

Wall Clock Time:     999.595275 secs
High Memory:             475.52 MBytes
MFLOPS (aggregate):  806112.33 M/sec
I/O Read Rate:            33.57 MBytes/Sec
I/O Write Rate:          215.40 MBytes/Sec
```

Example CrayPat-lite Output (2)

```
Table 1:  Profile by Function Group and Function (top 7 functions shown)
 Time%  |      Time  |    Imb.   |  Imb.   |    Calls  |Group
        |            |    Time   | Time%   |           | Function
        |            |           |         |           |  PE=HIDE

 100.0% | 101.961423 |       -- |     -- | 5315211.9 |Total
|-------------------------------------------------------------------------------------
|  92.5% |  94.267451 |       -- |     -- | 5272245.9 |USER
||------------------------------------------------------------------------------------
||  75.8% |  77.248585 | 2.356249 |   3.0% |    1001.0 |LAMMPS_NS::PairLJCut::compute
||   6.5% |   6.644545 | 0.105246 |   1.6% |      51.0 |LAMMPS_NS::Neighbor::half_bin_newton
||   4.1% |   4.131842 | 0.634032 |  13.5% |       1.0 |LAMMPS_NS::Verlet::run
||   3.8% |   3.841349 | 1.241434 |  24.8% | 5262868.9 |LAMMPS_NS::Pair::ev_tally
||   1.3% |   1.288463 | 0.181268 |  12.5% |    1000.0 |LAMMPS_NS::FixNVE::final_integrate
||====================================================================================
|   7.0% |   7.110931 |       -- |     -- |   42637.0 |MPI
||------------------------------------------------------------------------------------
||   4.8% |   4.851309 | 3.371093 |  41.6% |   12267.0 |MPI_Send
||   1.5% |   1.536106 | 2.592504 |  63.8% |   12267.0 |MPI_Wait
|====================================================================================
```

Example CrayPat-lite Output (3)

```
Table 2:  File Input Stats by Filename

  Read Time |  Read MBytes |  Read Rate |      Reads |  Bytes/ Call |File Name[max10]
            |              | MBytes/sec |            |              | PE=HIDE

 387.432937 | 13006.522781 |  33.571030 | 41596900.0 |       327.87 |Total
|-------------------------------------------------------------------------------------
| 331.691801 |  1395.829828 |   4.208213 | 13153931.0 |       111.27 |/proc/self/maps
|  13.129507 |  4075.682968 | 310.421627 |      868.0 |   4923575.28 |regional.grid.a
|  12.654338 |  2000.329418 | 158.074605 | 26892862.0 |        77.99 |./patch.input
|   3.924810 |   679.265625 | 173.069704 |        3.0 | 237420544.00 |./forcing.radflx.a
. . .
```

More Information from Same Profile

- You don't need to run again for the following:

```
For a complete report with expanded tables and notes, run:
  pat_report /lus/scratch/heidi/lab/craypat-lite/run/sweep3d.mpi.ap2

For help identifying callers of particular functions:
  pat_report -O callers+src /lus/scratch/heidi/lab/craypat-lite/run/
sweep3d.mpi.ap2

To see the entire call tree:
  pat_report -O calltree+src /lus/scratch/heidi/lab/craypat-lite/run/
sweep3d.mpi.ap2
```

The Cray Performance Analysis Framework

- **Supports traditional post-mortem performance analysis**
  - 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_region API**
  - Provides mechanism to control collection of performance data within source code
- **pat_report performs analysis and generates text reports**
- **pat_help: online help utility**
- **Cray Apprentice2: graphical visualization tool**

New perftools-base and Instrumentation Modules

Access perftools Software

- **Load perftools-base module and leave it loaded**
  - Provides access to man pages, Reveal, Cray Apprentice2, and the new instrumentation modules
  - Can keep loaded with no impact to applications
- **Available starting in perftools/6.3.0 in September 2015**
- **Prior to perftools/6.3.0:**
  - Load perftools module

Program Instrumentation Modules

Instrumentation modules available after perftools-base is loaded:

- perftools
- perftools-lite
- perftools-lite-events
- perftools-lite-gpu
- perftools-lite-loops

What Do the Instrumentation Modules Do?

perftools

- Full access to CrayPat functionality
- Use pat_build to instrument, pat_report to process data and collect reports
- Equivalent to loading perftools module in earlier releases

perftools-lite

- Default CrayPat-lite profiling
- Load before building and running program to get a basic performance profile sent to stdout
- Equivalent to loading perftools-lite module in earlier releases

Tips

- Loading perftools without loading perftools-base first will continue to work as in pre-6.3.0 releases until perftools/6.4.0
- Sites can consider loading the default perftools-base for all users. Cray will look at automatically loading this module in a future release.
- Instrumentation modules can be loaded and unloaded for different performance experiments
- Use the 'module list' command to easily see which type of instrumentation is currently active
- Unload the instrumentation module after performance analysis experiments are complete

How to Use CrayPat-lite

Access performance tools software & instrumentation module

```
> module load perftools-base
> module load perftools-lite
```

Build program

```
> make
```

a.out (instrumented program)

Run program (no modification to batch script)

```
aprun a.out
```

Condensed report to stdout
a.out*.rpt (same as stdout)
a.out*.ap2
files

Example Performance Data

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 Bytes	MPI Msg Count	MsgSz <16B Count	4KB<= MsgSz <64KB Count	Function Caller PE=[mmm]
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_ shalow_
78700800.0	7200.0	2400.0	4800.0	pe.0
78681600.0	4800.0	0.0	4800.0	pe.1
59020800.0	4800.0	1200.0	3600.0	pe.47
59421800.0	3725.0	100.0	3625.0	calc1_ shalow_
78700800.0	7200.0	2400.0	4800.0	pe.0
59011200.0	3600.0	0.0	3600.0	pe.1
59011200.0	3600.0	0.0	3600.0	pe.24

624,3 79%

Automatic Profiling Analysis Custom Template

```
#  You can edit this file, if desired, and use it
#   to reinstrument the program for  tracing like this:
#
#          pat_build -O standard.cray-
    xt.PE-2.1.56HD.pgi-8.0.amd64.pat-5.0.0.2-0apa.512.quad.cores.seal.
    090405.1154.mpi.pat_rt_exp=default:pat_rt_hwpc=none.14999.xf.xf.apa
#
# ----------------------------------------------------------------------
#       HWPC group to collect by default.

  -Drtenv=PAT_RT_HWPC=1  # Summary with TLB metrics.

#
# ----------------------------------------------------------------------
#       Libraries to trace.

  -g mpi

#
# ----------------------------------------------------------------------
#       User-defined functions to trace, sorted by % of samples.
#
#       The way these functions are filtered can be controlled with
#       pat_report options (values used for this file are shown):
#
#       -s apa_max_count=200    No more than 200 functions are listed.
#       -s apa_min_size=800     Commented out if text size < 800 bytes.
#       -s apa_min_pct=1        Commented out if it had < 1% of samples.
#       -s apa_max_cum_pct=90   Commented out after cumulative 90%.
#
#       Local functions are listed for completeness, but cannot be
    traced.

  -w  # Enable tracing of user-defined functions.
      # Note: -u should NOT be specified as an additional option.
```

```
# 31.29%  38517 bytes
         -T prim_advance_mod_preq_advance_exp_

# 15.07%  14158 bytes
         -T prim_si_mod_prim_diffusion_

. . .


#  Functions below this point account for less than 10% of samples.

#  0.66%  4575 bytes
#         -T bndry_mod_bndry_exchangev_thsave_time_

#  0.10%  46797 bytes
#         -T baroclinic_inst_mod_binst_init_state_

#  0.04%  62214 bytes
#         -T prim_state_mod_prim_printstate_

#  0.00%  118 bytes
#         -T time_mod_timelevel_update_

# ----------------------------------------------------------------------

  -o preqx.cray-xt.PE-2.1.56HD.pgi-8.0.amd64.pat-5.0.0.2.x+apa
     # New instrumented program.

  /.AUTO/cray/css.pe_tools/malice/craypat/build/pat/2009Apr03/2.1.56HD/
     amd64/homme/pgi7pat-5.0.0.2/homme/2005Dec08/build.Linux/preqx.cray-
     xt.PE-2.1.56HD.pgi-8.0.amd64.pat-5.0.0.2.x  # Original program.
```

Maximize On-node Communication by Reordering MPI ranks

MPI Rank Reorder – 2 Interfaces Available

- **CrayPat**
  - Run program and let CrayPat determine if communication is dominant, detect communication pattern and suggest MPI rank order if applicable

- **grid_order utility**
  - User knows communication pattern in application and wants to quickly create a new MPI rank placement file
  - Available when perftools module is loaded

When Is Rank Re-ordering Useful?

- Maximize on-node communication between MPI ranks
- Physical system topology agnostic
- Grid detection and rank re-ordering is helpful for programs with significant point-to-point communication
- 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

Automatic Communication Grid Detection

- Cray performance tools produce a custom rank order if it's beneficial based on grid size, grid order and cost metric
- 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
```

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_METHOD to 3 prior to
# executing the program.
#
0,532,64,564,32,572,96,540,85,596,72,524,40,604,24,588
104,556,16,628,80,636,56,620,48,516,112,580,88,548,120,612
1,403,65,435,33,411,97,443,95,467,25,499,105,507,41,475
73,395,81,427,57,459,17,419,53,399,85,431,21,463,61,391,113,491,49,387,89,451,121,483
6,436,102,468,70,404,38,412,2,530,34,562,66,538,98,522,14,444,46,476,110,508,78,500
86,396,30,428,62,460,54,492,18,514,74,586,58,626,82,546,118,420,22,452,94,388,126,484
129,563,193,531,161,571,225,135,315,167,339,199,347,259,539,241,595,233,523,249,603,307,231,371,239,379,191,331,185,555
153,587,169,627,137,635,201,175,363,159,323,143,355,255,619,177,515,145,579,209,547,291,207,275,183,283,151,267,217,611
7,405,71,469,39,437,103,413,133,406,197,438,165,470,229,47,445,15,509,79,477,31,501
111,397,63,461,55,429,87,421,23,493,119,389,95,453,127,485
134,402,198,434,166,410,230,442,238,466,174,506,158,394,246,474
190,498,254,426,142,458,150,386,182,418,206,490,214,450,222,482
128,533,192,541,160,565,232,525,224,573,240,597,184,557,248,605
168,589,200,517,152,629,136,549,176,637,144,621,208,581,216,613
95,439,37,407,69,447,101,415,13,471,45,503,29,479,77,511
3,440,35,432,67,400,99,408,1,464,43,496,27,472,51,504
19,392,75,424,59,456,83,384,107,416,91,488,115,448,123,480
1132,401,196,441,164,409,228,433,236,465,204,473,244,393,569,561
188,497
252,505,140,425,212,457,156,385,172,417,180,449,148,489,277,365
220,481
131,534,195,542,163,566,227,526,235,574,203,598,243,558,336,344
187,606
251,590,211,630,179,638,139,622,155,550,171,518,219,582,678,362
147,614
761,660,737,652,705,668,745,692,673,700,641,684,713,644,694,654
753,724
729,732,681,756,721,716,764,676,697,748,689,657,740,665,279,374
649,708
760,528,736,536,704,560,744,520,672,568,712,592,752,552,302,334
640,600
728,584,680,624,720,512,696,632,688,616,664,544,608,656,679,703
648,576
762,659,738,651,706,667,746,643,714,691,674,699,754,683,645,759
730,723
722,731,763,658,642,755,739,675,707,650,682,715,698,666,690,747
257,345,265,313,281,305,273,
545,297,633,361,625,321,585,537,601,289,553,353,593,521,513,529
256,373,261,341,264,349,280,304,360,312,376,293,296,368,317,272,381,269,309,285,333,
352,301,320,325,288,357,328,370,766,306,710,378,742,330,
646,298,750,322,718,354,758,290,734,662,686,670,726,702,
262,375,263,343,270,311,271,351,286,319,278,342,287,350,
294,318,358,383,359,310,295,382,326,303,327,367,366,335,
765,661,709,663,741,653,711,669,767,655,743,671,749,695,
677,727,751,693,647,701,717,687,757,685,733,725,719,735,
```

grid_order Utility

- No need to run application if you already know program's data movement pattern
- Originally designed for MPI programs, but also works for SHMEM and PGAS programming models that use Cray PMI
- Utility available if perftools-base modulefile is loaded
- See grid_order(1) man page or run grid_order with no arguments to see usage information

Using New Rank Order

- Save grid_order output to file called MPICH_RANK_ORDER
- Export MPICH_RANK_REORDER_METHOD=3
- Run non-instrumented binary with and without new rank order to check overall wallclock time for performance improvements
- Can be used for all subsequent executions of same job size

Using Performance Counters

Performance Counters

- **Cray supports raw counters, derived metrics and thresholds for:**
  - Processor (core and uncore)
  - Network
  - Accelerator
  - Power
- **Predefined groups**
  - Groups together suggested counters for experiments
- **Single interface to access counters**
  - PAT_RT_PERFCTR environment variable
- **See *hwpc, nwpc, accpc, rapl* man pages**

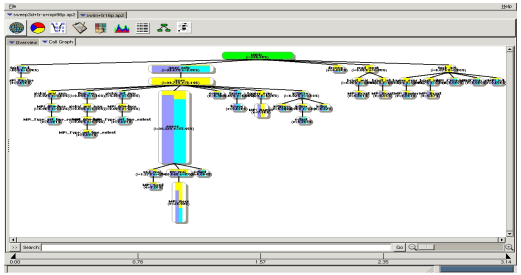
Example: HW counter data and Derived Metrics

```
  PAPI_TLB_DM  Data translation lookaside buffer misses
  PAPI_L1_DCA  Level 1 data cache accesses
  PAPI_FP_OPS  Floating point operations
  DC_MISS      Data Cache Miss
  User_Cycles  Virtual Cycles
========================================================================
USER
------------------------------------------------------------------------
  Time%                                            98.3%
  Time                                         4.434402 secs
  Imb.Time                                           -- secs
  Imb.Time%                                          --
  Calls                      0.001M/sec          4500.0 calls
  PAPI_L1_DCM               14.820M/sec        65712197 misses
  PAPI_TLB_DM                0.902M/sec         3998928 misses
  PAPI_L1_DCA              333.331M/sec      1477996162 refs
  PAPI_FP_OPS              445.571M/sec      1975672594 ops
  User time (approx)         4.434 secs     11971868993 cycles  100.0%Time
  Average Time per Call                        0.000985 sec
  CrayPat Overhead : Time       0.1%
  HW FP Ops / User time    445.571M/sec      1975672594 ops    4.1%peak(DP)
  HW FP Ops / WCT          445.533M/sec
  Computational intensity     0.17 ops/cycle     1.34 ops/ref
  MFLOPS (aggregate)       1782.28M/sec
  TLB utilization           369.60 refs/miss    0.722 avg uses
  D1 cache hit,miss ratios   95.6% hits         4.4% misses
  D1 cache utilization (misses)  22.49 refs/miss   2.811 avg hits
========================================================================
```

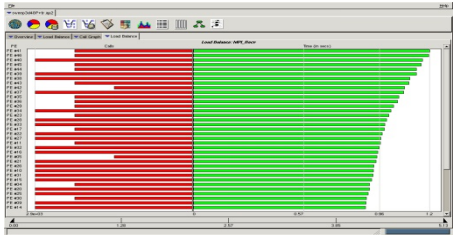
PAT_RT_PERFCTR=1
Flat profile data
Raw counts
Derived metrics

Visualizing Performance of Your Application Through Cray Apprentice2

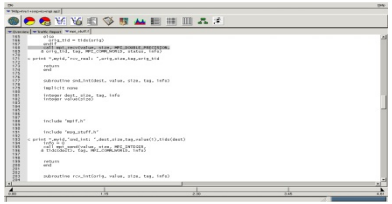
Cray Apprentice²

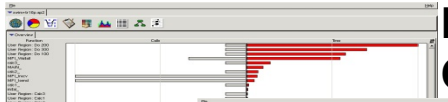


Load balance views

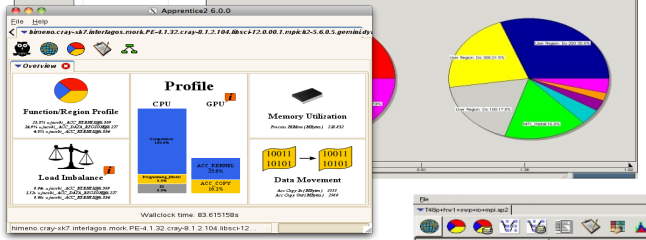


Source code mapping

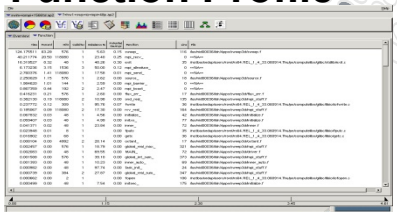


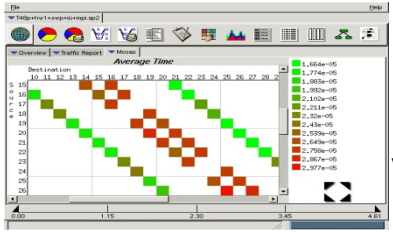


Program Overview

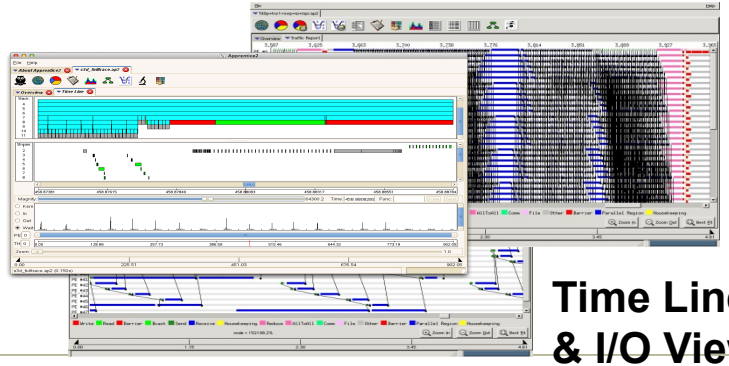


Function Profile

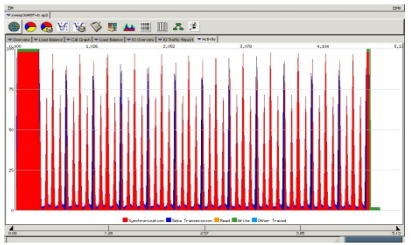




Pair-wise Communication View



Time Line & I/O Views



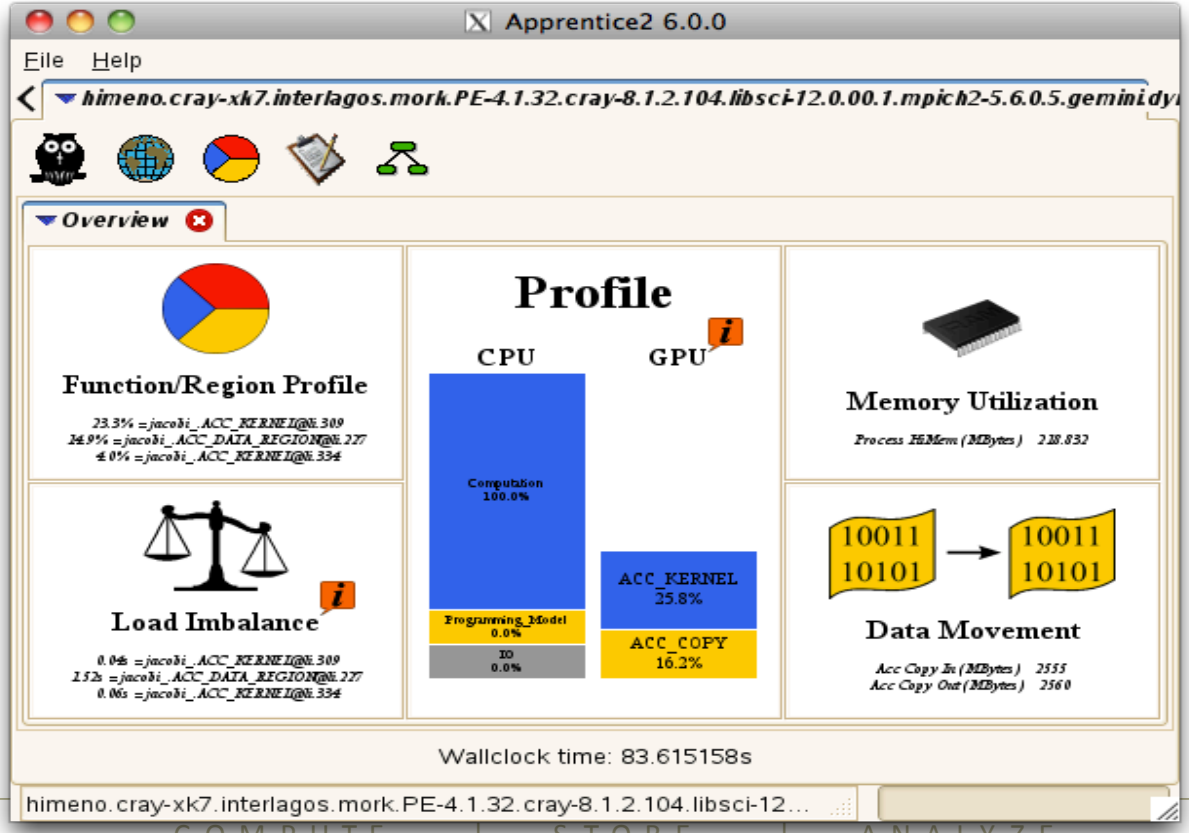
Communication & I/O Activity View

Installing Apprentice2 on Laptop

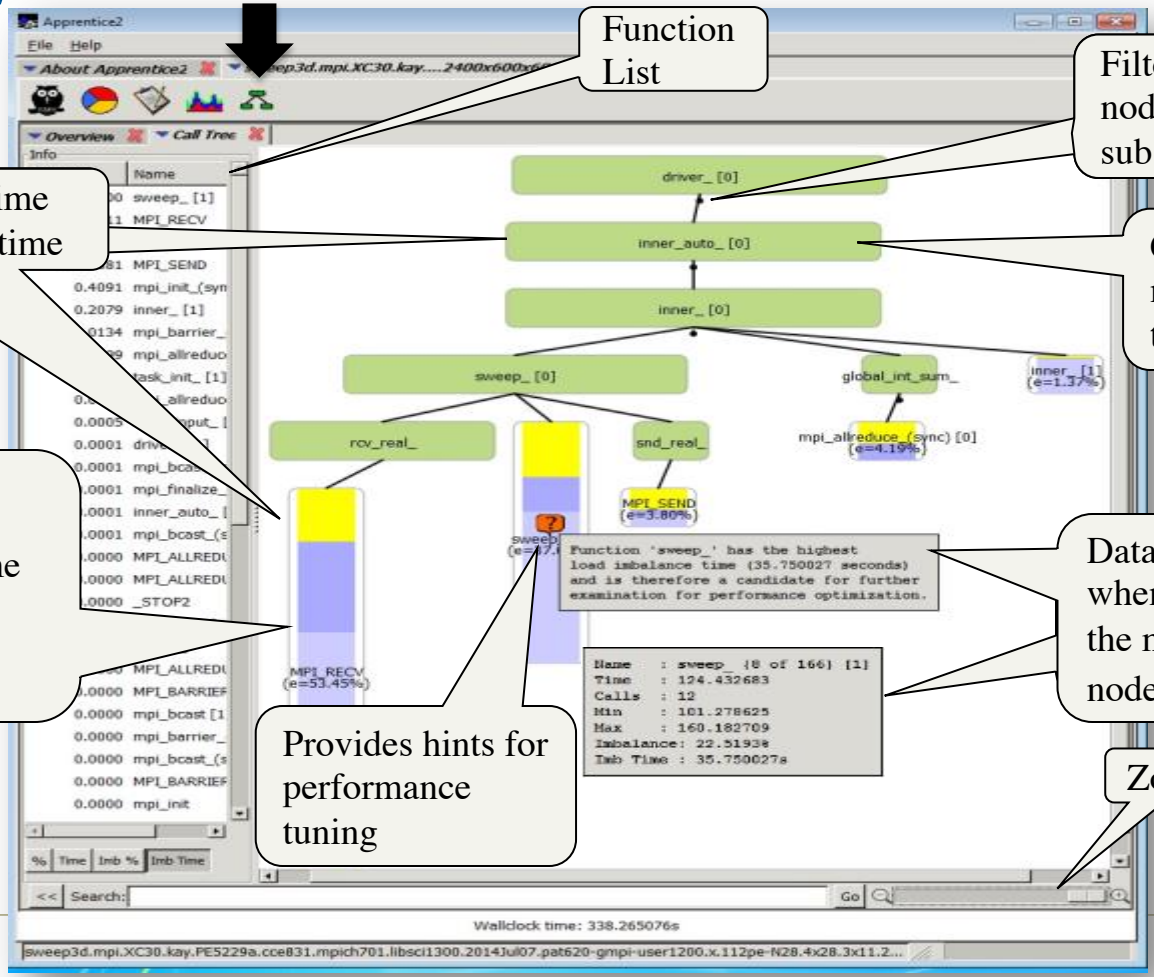
From a Cray login node

- `> module load perftools`
- **Go to:**
  - $CRAYPAT_ROOT/share/desktop_installers/
- **Download .dmg or .exe installer to laptop**
- **Double click on installer and follow directions to install**

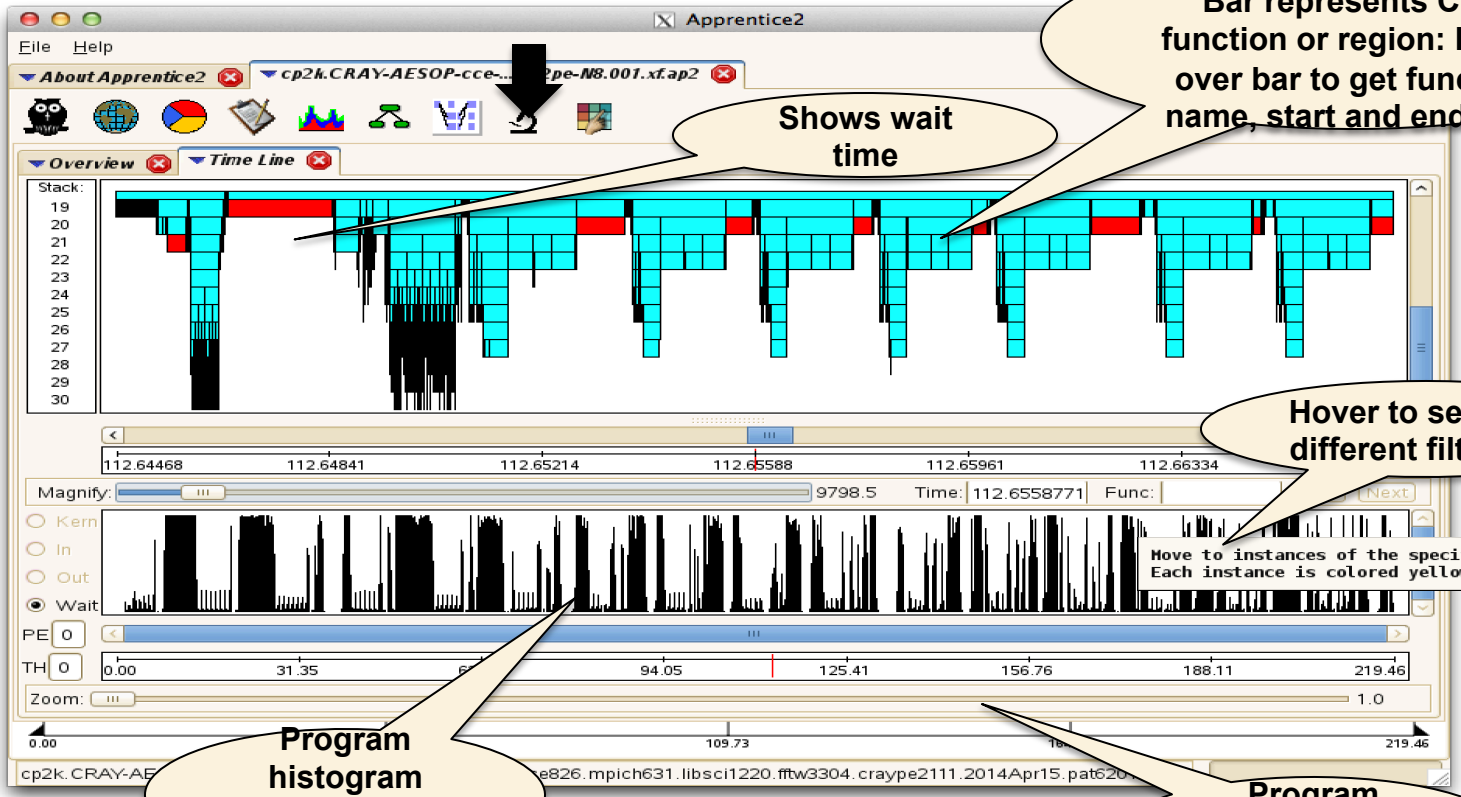
Apprentice2 Overview with GPU Data



Call Tree View



CPU Program Timeline: 36GB CP2K Full Trace

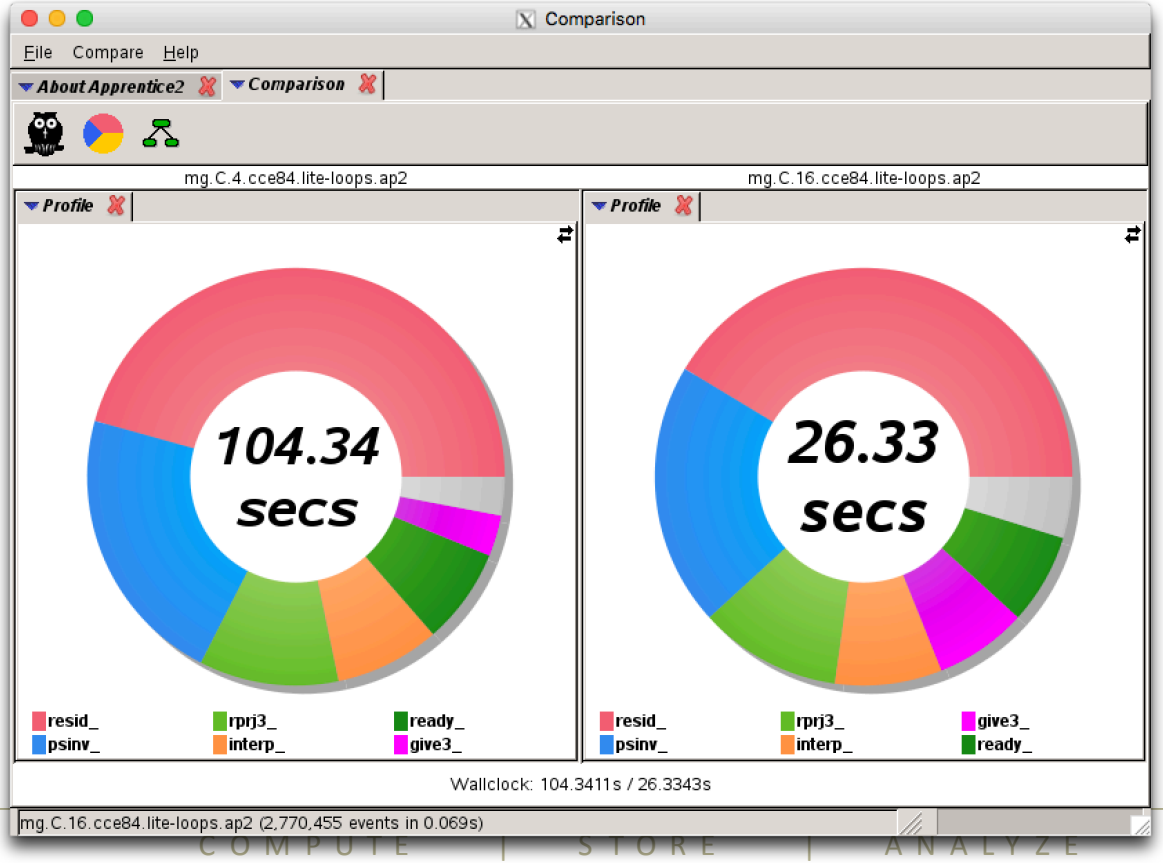


What's New?

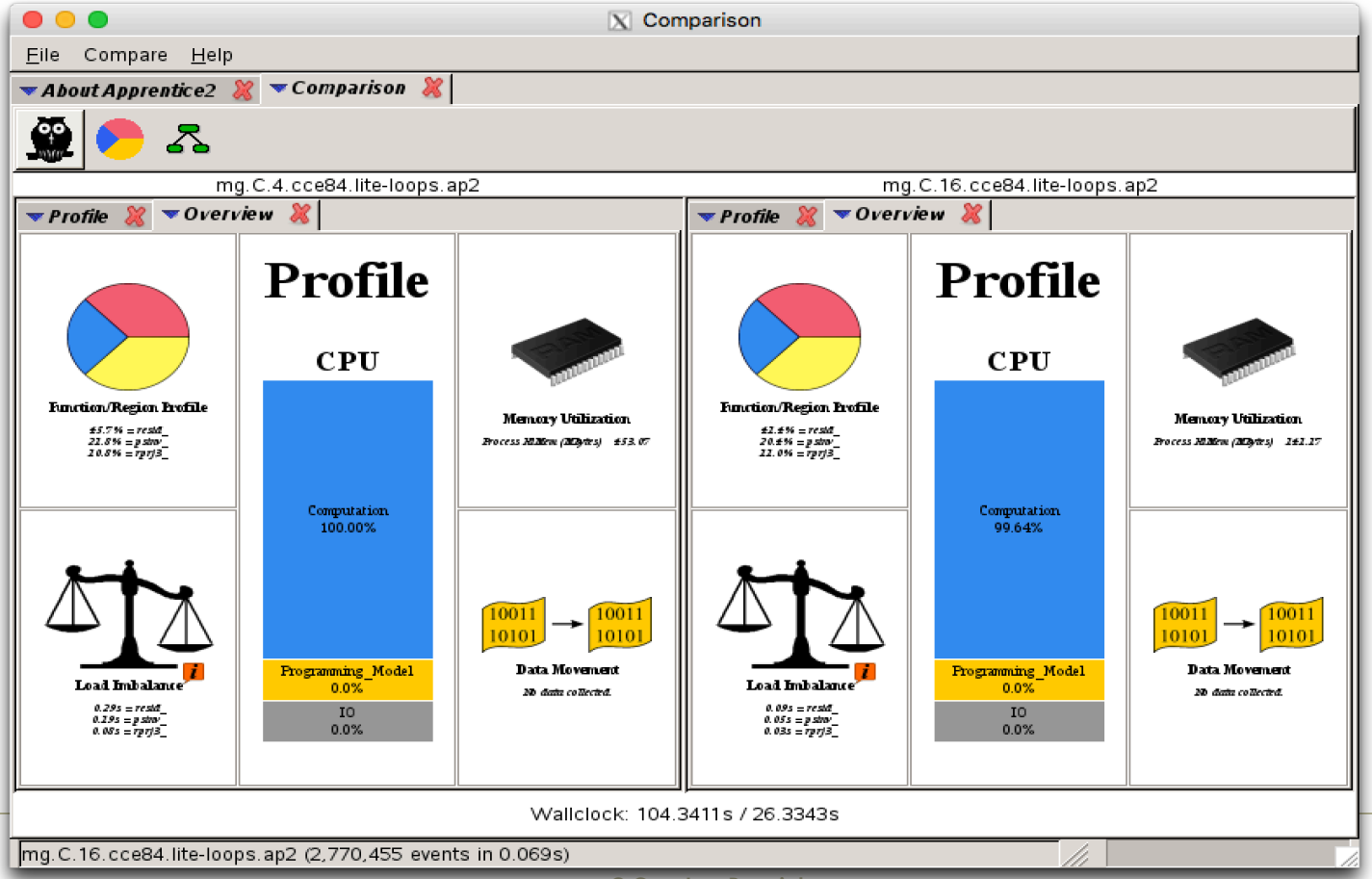
Recent Enhancements

- **Improved ease of use:**
  - perftools-base module, pat_info utility
- **Profile comparison in Cray Apprentice2**
  - Useful for comparing MPI vs MPI+OpenMP, scaling bottlenecks, etc.
- **2D communication heat map (Cray Apprentice2 Mosaic) in summarized mode**
- **Visualize sampling data over time with associated call stack**

Apprentice2 Comparison



Apprentice2 Comparison (2)



Pat_info Utility

- Can be used to generate a quick summary statement regarding the contents of a CrayPat .ap2 file or set of files without running pat_report or Apprentice2
- Useful if you have multiple .ap2 files in a directory or if you want to review what experiments you have already performed
- Works on a single .ap2 file or a directory of .ap2 files
- When invoked with no arguments, the command looks in the current directory for .ap2 files
- When invoked with a directory argument, information about all .ap2 files in that directory are displayed

Example pat_info Utility Output

```
# When given a single .ap2 file argument, it defaults to the long form
# (counter lists are added with the -c option):

kay-esl$ pat_info -c
sweep3d.mpi+17552-12s.ap2
ap2:                                sweep3d.mpi+17552-12s.ap2
ap2_size:           289792
RTS:                                yes
Experiment:         samp_cs_time
PE:                                 CRAY
NumPEs:                             96
NumThreads:         0
NumLeafNodes:       928
OpenMP:                             yes
Original prog:      /lus/scratch/clark/sweep3d/sweep3d.mpi+orig
prog_size:          (not available)
NumHWPC:            16
  CYCLES_RTC
  L1D:REPLACEMENT
  L2_RQSTS:ALL_DEMAND_DATA_RD
  FP_COMP_OPS_EXE:SSE_SCALAR_DOUBLE
  FP_COMP_OPS_EXE:SSE_FP_SCALAR_SINGLE
  FP_COMP_OPS_EXE:X87
  FP_COMP_OPS_EXE:SSE_PACKED_SINGLE
  SIMD_FP_256:PACKED_SINGLE
  FP_COMP_OPS_EXE:SSE_FP_PACKED_DOUBLE
  SIMD_FP_256:PACKED_DOUBLE
  L2_RQSTS:DEMAND_DATA_RD_HIT
  CPU_CLK_UNHALTED:THREAD_P
  CPU_CLK_UNHALTED:REF_P
  MEM_UOPS_RETIRED:ALL_LOADS
  DTLB_LOAD_MISSES:MISS_CAUSES_A_WALK
  DTLB_STORE_MISSES:MISS_CAUSES_A_WALK
NumCPMC:            2
  PM_ENERGY:NODE
  PM_ENERGY:ACC
```

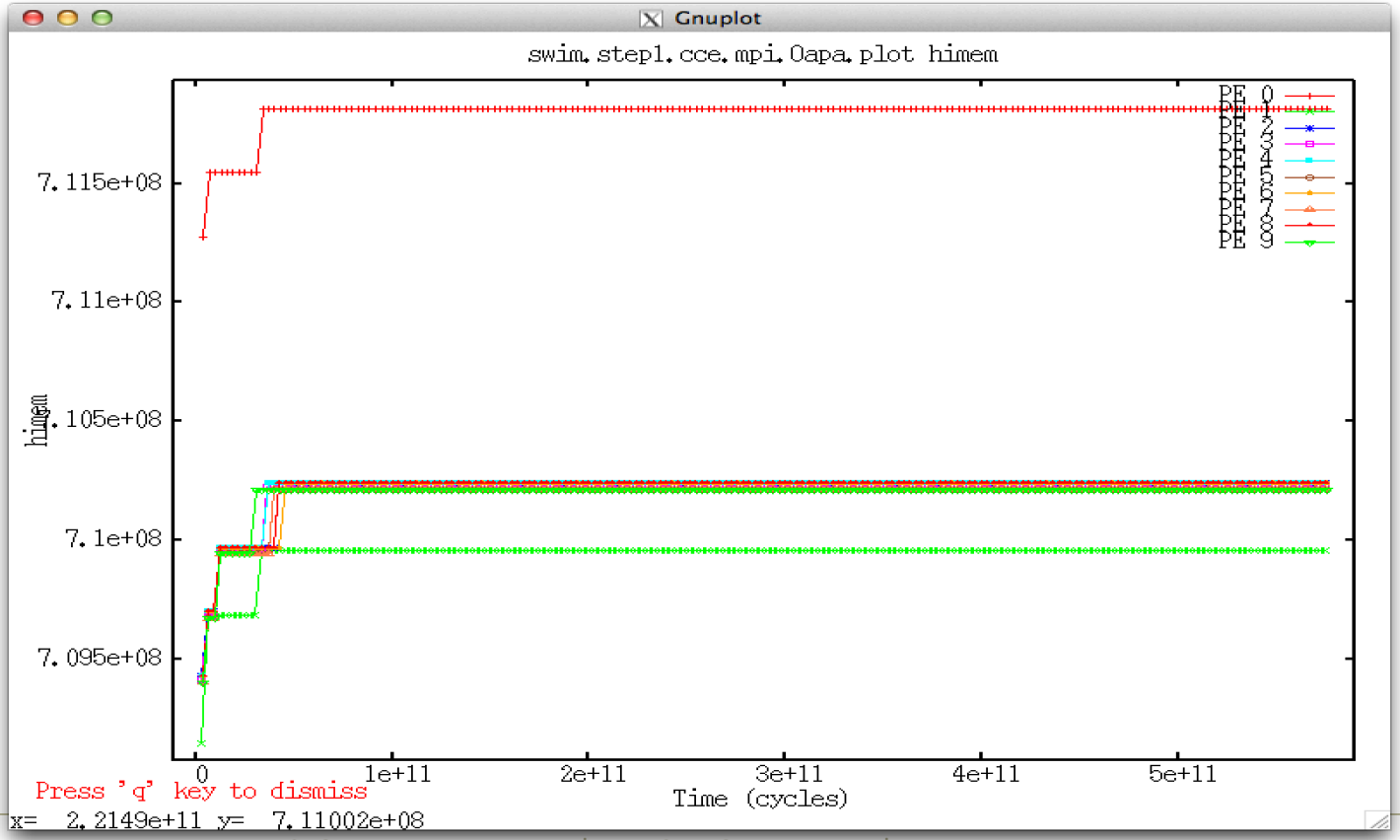
Sampling Over Time

- Available in perftools/6.2.3 (available in April 2015)
- Intended for collecting higher overhead performance data
- Sampling experiment in non-summary mode
  - `PAT_RT_SUMMARY=0`
  - `PAT_RT_SAMPLING_DATA=cray_pm`
- Records data every 100 Program Counter addresses by default (user can adjust)
- Examples:
  - Heap, shared heap
  - Perfctr (selected performance counters)
  - Rusage (resource usage (getrusage)
  - Cray PM, RAPL

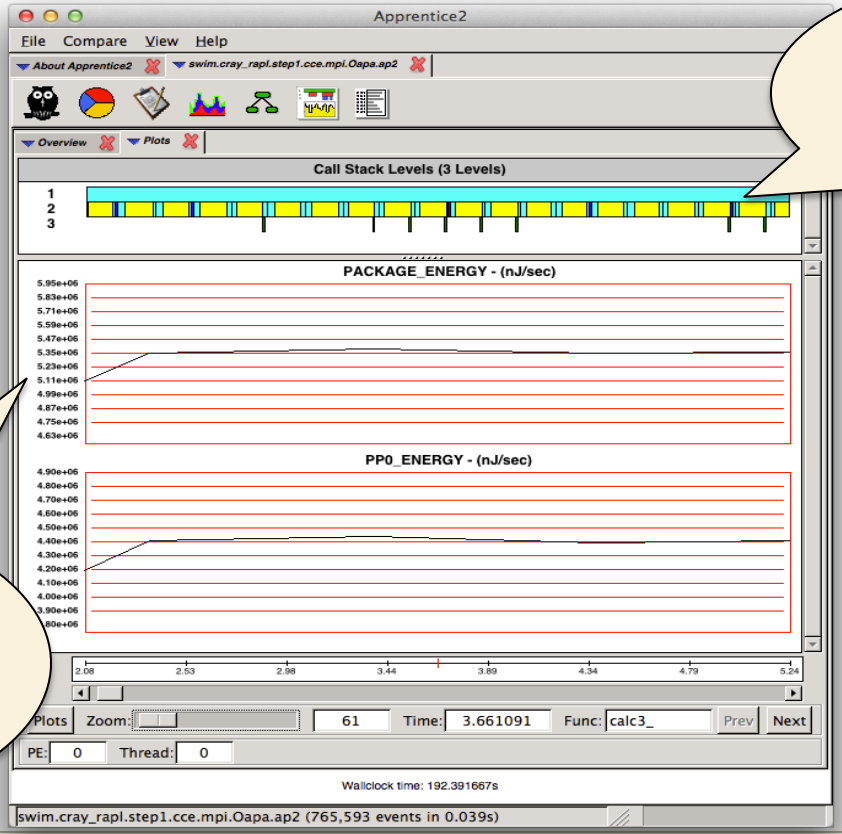
Visualize Samples Over Time

- **Plots show activity over time**
- **pat_report generates gnuplot files**
  - > pat_report [-r] -f plot $some.xf
  - > pat_report [-r] -f plot $some.ap2
- **Visualize (pat_report launches gnuplot)**
  - > pat_report $some.plot
  - > pat_report $some.plot/himem.gp
  - > pat_report -s pe=N
    - plot data only for pe N
  - > pat_report -s filter_input='pe<10'
    - specify a subset of pe values for which to plot data
- **Run "pat_help plots" or see craypat(1) man page for more info**

Memory High Water Mark



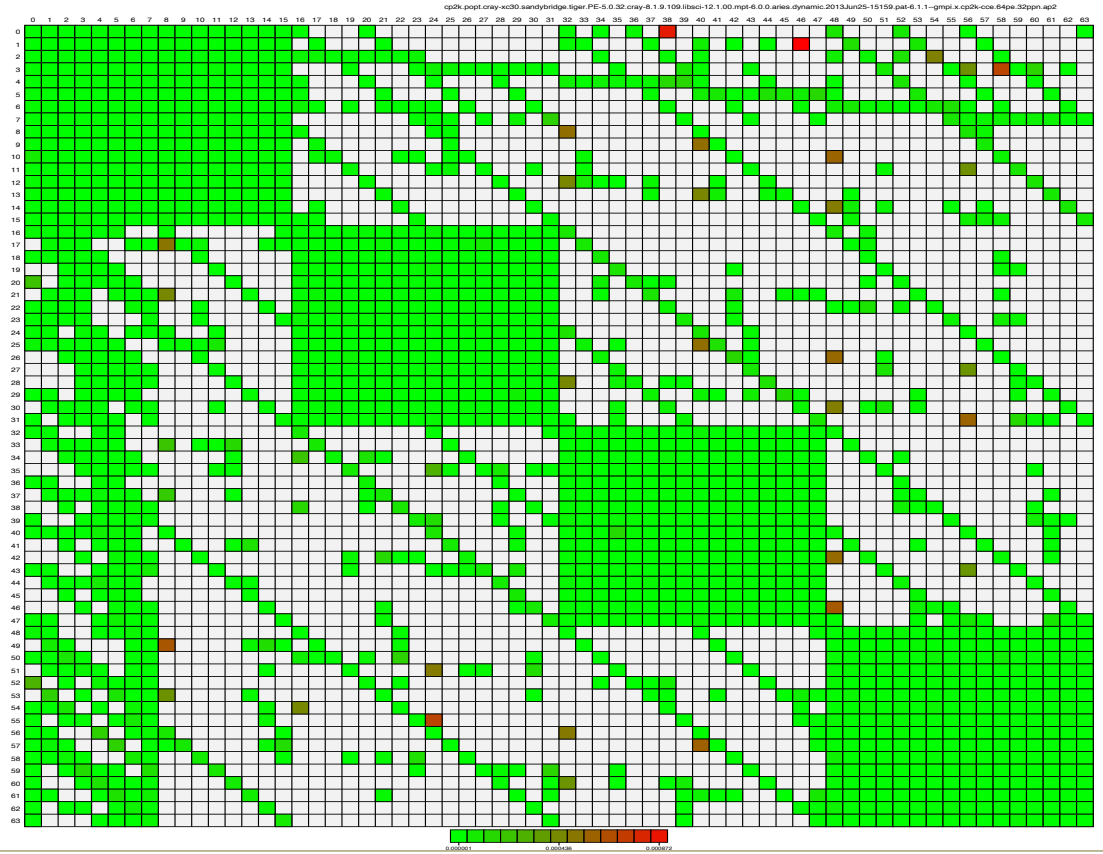
Energy Consumption Over Time (XC Systems)



Call stack:
Bar represents function or region: Hover over bar to get function name, start and end time

Plots of energy consumed by the socket and by the cores within a socket over time. Can also show memory high water mark, etc.

What is My Program's Communication Pattern?



GPU Support

What to Determine About Your Application

- Where are the dominant bottlenecks (excessive communication, excessive data movement, etc.)
- What parts of my code ran on the GPU?
- Was the GPU used efficiently?
- Was the host stalled, waiting for completion of accelerated regions?

Performance Feedback Available

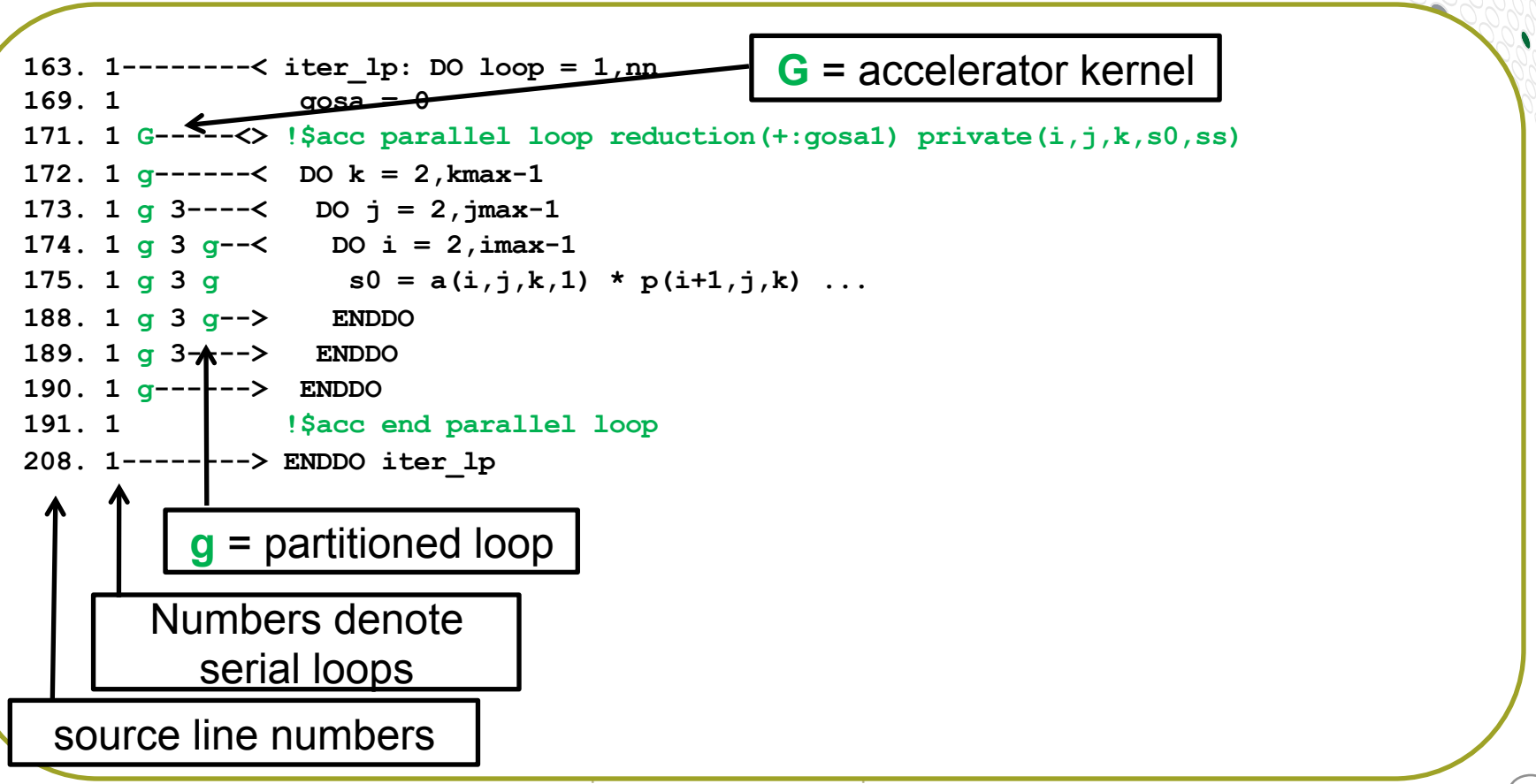
OpenACC and OpenMP 4.0 accelerator directives:

- **Compiler feedback through listings (static analysis)**
  - CCE: -hlist=a
- **Runtime commentary of accelerator activity**
  - CCE: setenv CRAY_ACC_DEBUG <1,2,3>
- **Application profiling**
  - Cray Performance Tools

Compiler feedback – Extremely Useful

- **Did the compiler recognize the accelerator directives?**
  - A good sanity check
- **How will the compiler move data?**
  - First major code optimization: removing unnecessary data movements
- **How will the compiler schedule loop iterations across GPU threads?**
  - Did it parallelize the loopnests?
  - Did it schedule the loops sensibly?
  - Another main optimization: correcting poor loop scheduling

Acceleration Feedback



Data Movement Feedback

```
163. 1--------< DO loop = 1,nn
169. 1           gosa1 = 0
171. 1 G-----<> !$acc parallel loop reduction(+:gosa1) private(i,j,k,s0,ss)
172. 1 g------<  DO k = 2,kmax-1
173. 1 g 3----<   DO j = 2,jmax-1
174. 1 g 3 g--<    DO i = 2,imax-1
175. 1 g 3 g         s0 = a(i,j,k,1) * p(i+1,j,k) ...
188. 1 g 3 g-->    ENDDO
189. 1 g 3---->   ENDDO
190. 1 g------>  ENDDO
191. 1          !$acc end parallel loop
208. 1--------> ENDDO
```

Over-cautious: compiler worried about halos; could specify copyout(wrk2)

```
ftn-6418 ftn: ACCEL File = himeno_F_v02.F90, Line = 171
  If not already present: allocate memory and copy whole array "p" to accelerator, free
   at line 191 (acc_copyin).

ftn-6416 ftn: ACCEL File = himeno_F_v02.F90, Line = 171
  If not already present: allocate memory and copy whole array "wrk2" to accelerator,
   copy back at line 191 (acc_copy).
```

Loop Scheduling

```
163. 1--------< DO loop = 1,nn
169. 1           gosa1 = 0
171. 1 G-----<> !$acc parallel loop reduction(+:gosa1) private(i,j,k,s0,ss)
172. 1 g------<  DO k = 2,kmax-1
173. 1 g 3----<   DO j = 2,jmax-1
174. 1 g 3 g--<    DO i = 2,imax-1
175. 1 g 3 g        s0 = a(i,j,k,1) * p(i+1,j,k) ...
188. 1 g 3 g-->    ENDDO
189. 1 g 3---->   ENDDO
190. 1 g------>  ENDDO
191. 1          !$acc end parallel loop
208. 1--------> ENDDO

ftn-6430 ftn: ACCEL File = himeno_F_v02.F90, Line = 172
  A loop starting at line 172 was partitioned across the thread blocks.

ftn-6509 ftn: ACCEL File = himeno_F_v02.F90, Line = 173
  A loop starting at line 173 was not partitioned because a better candidate was found at line 174.

ftn-6412 ftn: ACCEL File = himeno_F_v02.F90, Line = 173
  A loop starting at line 173 will be redundantly executed.

ftn-6430 ftn: ACCEL File = himeno_F_v02.F90, Line = 174
  A loop starting at line 174 was partitioned across the 128 threads within a threadblock.
```

CCE Runtime Activity Commentary

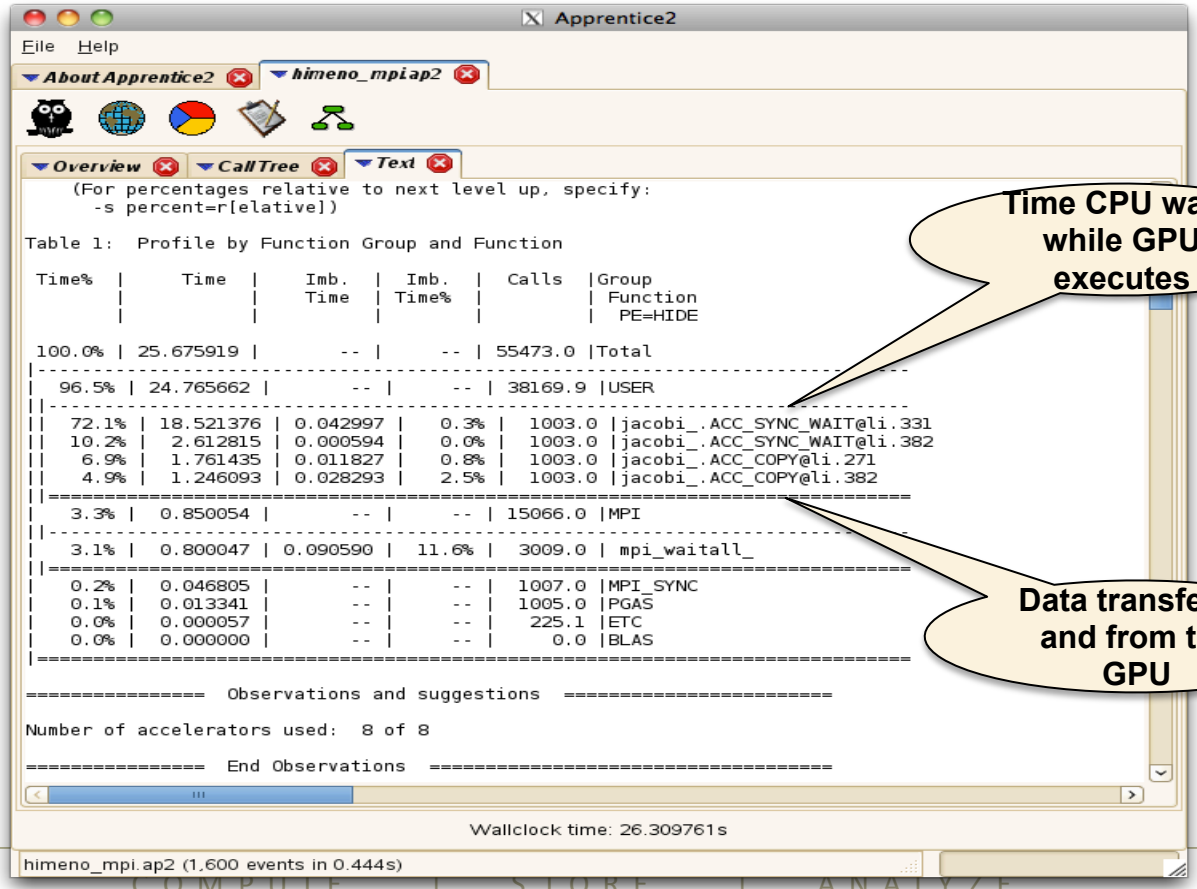
- Setenv CRAY_ACC_DEBUG <1,2,3>

```
ACC: Initialize CUDA
ACC: Get Device 0
ACC: Create Context
ACC: Set Thread Context
ACC: Start transfer 2 items from saxpy.c:17
ACC:       allocate, copy to acc 'x' (4194304 bytes)
ACC:       allocate, copy to acc 'y' (4194304 bytes)
ACC: End transfer (to acc 8388608 bytes, to host 0 bytes)
ACC: Execute kernel saxpy$ck_L17_1 blocks:8192 threads:128 async(auto) from saxpy.c:17
ACC: Wait async(auto) from saxpy.c:18
ACC: Start transfer 2 items from saxpy.c:18
ACC:       free 'x' (4194304 bytes)
ACC:       copy to host, free 'y' (4194304 bytes)
ACC: End transfer (to acc 0 bytes, to host 4194304 bytes)
```

Collecting GPU Statistics for OpenACC / OpenMP

- **Make sure the following modules are loaded:**
  - craype-accel-nvidia35 accelerator module
  - PrgEnv-cray module
  - perftools module (perftools-base is already loaded)
- **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

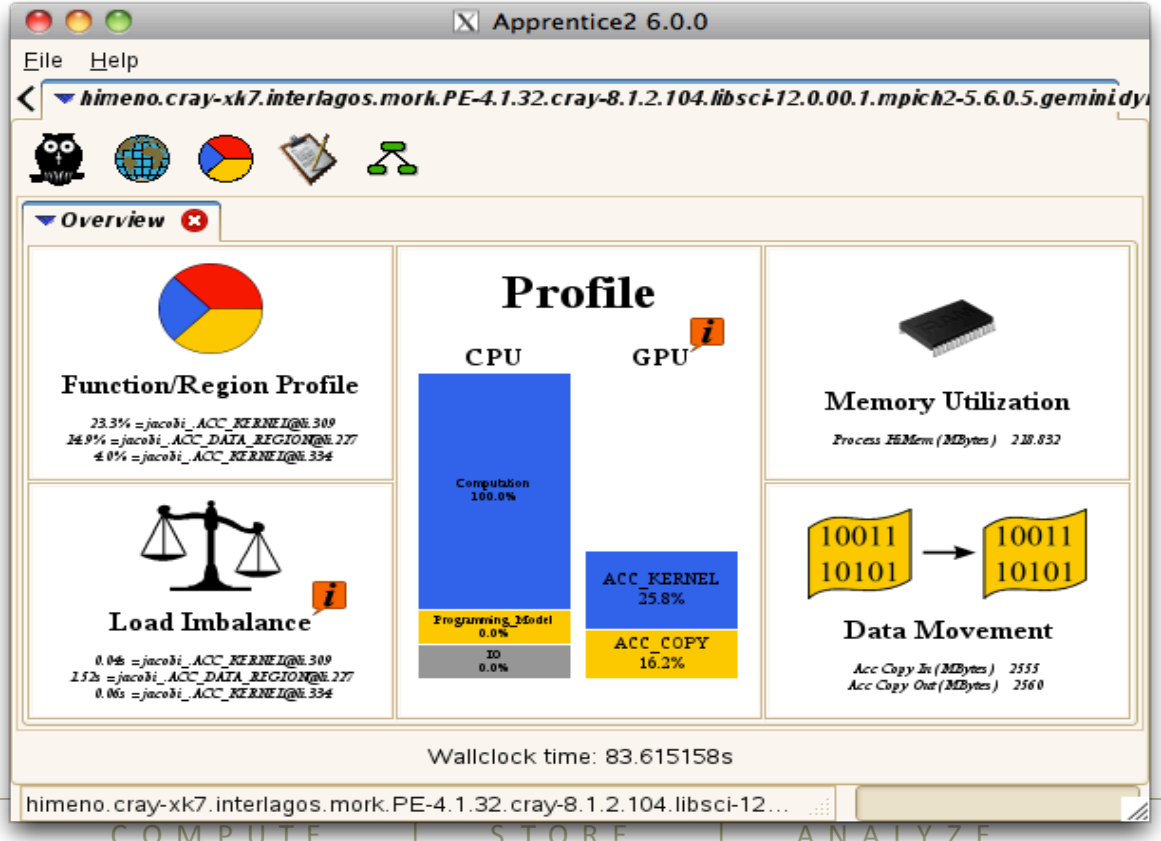
Profile with GPU Information



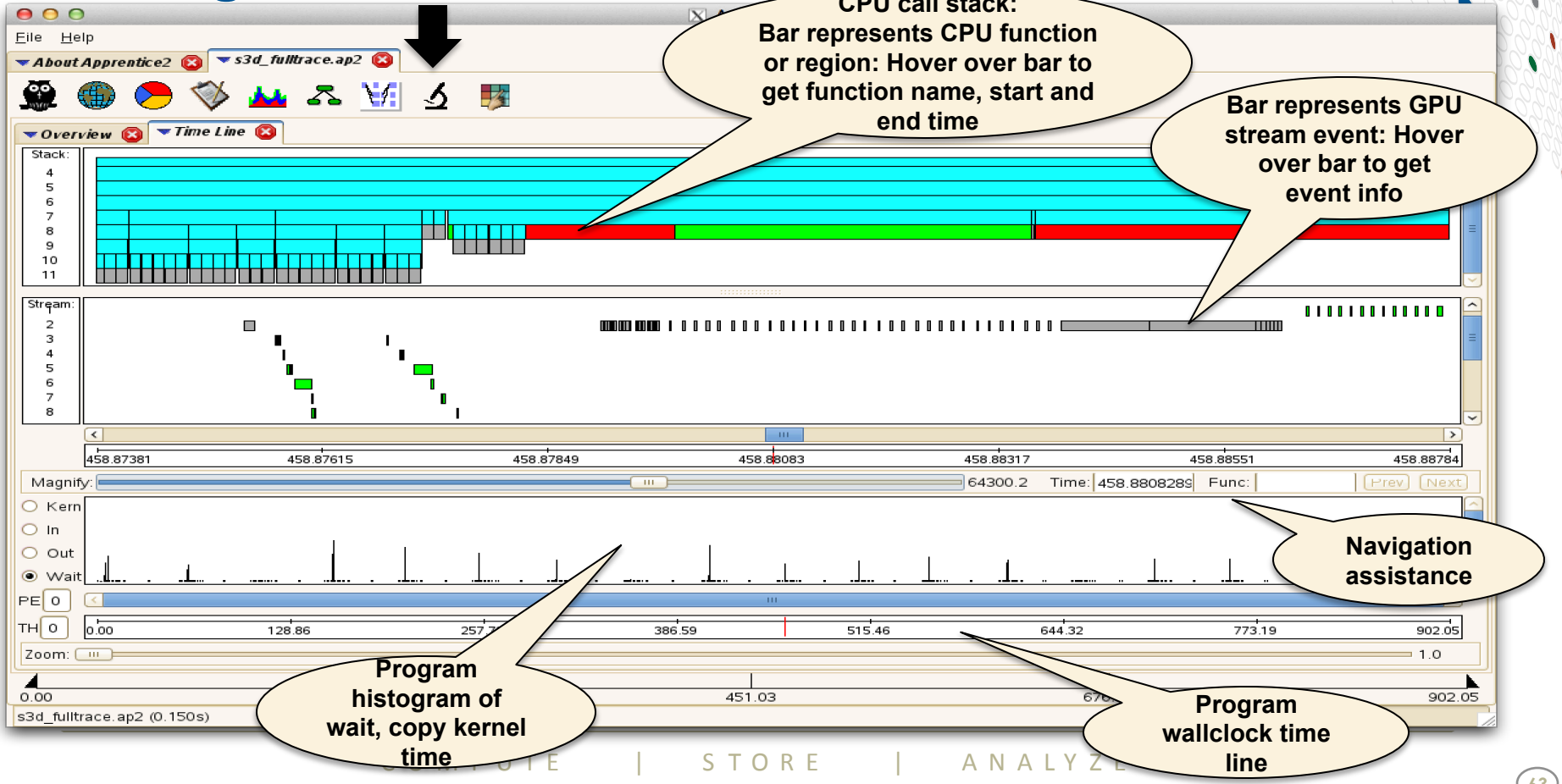
Example Accelerator Statistics

```
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)
||||=====================================================================================================
```

Apprentice2 Overview with GPU Data



GPU Program Timeline




CRAY
COMPUTE | STORE | ANALYZE
Thank You