Cray Performance Tools

Heidi Poxon

Agenda

- Overview
- Using CrayPAT-lite and CrayPAT
- MPI rank reorder
- Cray Apprentice2

Cray Performance Analysis Tools

- **Whole program performance analysis with**
  - Novice and advanced user interfaces
  - Support for MPI, SHMEM, OpenMP, UPC, CAF, OpenACC, CUDA
  - Load imbalance detection
  - HW counter metrics (hit rates, computational intensity, etc.)
  - Observations and inefficiencies
  - Data correlation to user source
  - Minimal program perturbation
- **Sampling, tracing with runtime summarization, full trace (timeline) modes available**
- **Supports CCE, Intel and GCC compilers**

Components

- **CrayPat and CrayPat-lite**
  - Identifies top time consuming routines, work load imbalance, MPI rank placement strategies, etc.
- **PAPI**
- **Cray Apprentice2**
  - Visualize load imbalance, excessive communication, network contention, excessive serialization

Access perftools Software

- **Load perftools-base module and leave it loaded**
  - Provides access to man pages, Reveal, Cray Apprentice2, and the new instrumentation modules
  - Does not alter how programs are built
- **Load desired instrumentation module**

Two Modes of Use

- **CrayPat-lite** for novice users, or convenience
- **CrayPat** for in-depth performance investigation and tuning assistance
- **Both offer:**
  - Whole program analysis across many nodes
  - Indication of causes of problems
  - Suggestions of modifications for performance improvement

Program Instrumentation Modules

- **Instrumentation modules prepare an application for performance data collection**

- **Instrumentation modules available after perftools-base is loaded:**
  - perftools
  - perftools-lite
  - perftools-lite-loops
  - perftools-lite-events
  - perftools-lite-gpu

Use first to get basic program performance profile

“Lite” Mode

Load performance tools instrumentation module (assumes perftools-base is already loaded)

```
$ module load perftools-lite
```

Build program (no modification to makefile)

```
$ 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 Output

```
#####################################################################
#                                                                   #
#            CrayPat-lite Performance Statistics                    #
#                                                                   #
#####################################################################

CrayPat/X:  Version develop.7492 Revision d21de26  04/15/16
14:55:58
Experiment:                    lite  lite/sample_profile
Number of PEs (MPI ranks):   1,024
Numbers of PEs per Node:        64  PEs on each of  16  Nodes
Numbers of Threads per PE:       1
Number of Cores per Socket:     68
Execution start time:  Tue Apr 19 12:47:48 2016
System name and speed:  avalon  1401 MHz (approx)

Avg Process Time:    572.82 secs
High Memory:        325,662 MBytes     318.03 MBytes per PE
I/O Read Rate:     1.753447 MBytes/sec
I/O Write Rate:   15.522255 MBytes/sec
```

Example CrayPat-lite Output

```
Table 1:  Profile by Function Group and Function (top 10 functions shown)
  Samp% |      Samp |     Imb. |  Imb. |Group
        |           |     Samp | Samp% | Function
        |           |          |       |  PE=HIDE
 100.0% | 55,605.7 |       -- |    -- |Total
|-----------------------------------------------------------------------
|  56.5% | 31,412.8 |       -- |    -- |USER
||----------------------------------------------------------------------
||  19.7% | 10,944.1 |    290.9 |  2.6% |create_boundary$boundary_
||  10.7% |  5,937.8 |    214.2 |  3.5% |get_block$blocks_
||   3.9% |  2,194.4 |      7.6 |  0.3% |create_distrb_balanced$distribution_
||   2.0% |  1,135.5 |    137.5 | 10.8% |impvmixt$vertical_mix_
||   1.9% |  1,064.8 |    124.2 | 10.5% |impvmixt_correct$vertical_mix_
||   1.8% |  1,019.5 |    155.5 | 13.2% |global_sum_dbl$global_reductions_
||   1.7% |    952.3 |    395.7 | 29.4% |boundary_2d_dbl$boundary_
||======================================================================
|  22.5% | 12,513.4 |       -- |    -- |ETC
||----------------------------------------------------------------------
||  20.1% | 11,151.4 | 2,758.6 | 19.9% |__cray_memcpy_KNL
||======================================================================
|  20.7% | 11,503.5 |       -- |    -- |MPI
||----------------------------------------------------------------------
||  11.1% |  6,171.6 | 1,785.4 | 22.5% |MPI_ALLREDUCE
||   7.9% |  4,377.8 | 3,254.2 | 42.7% |mpi_waitall
```

Example Output (2)

```
MPI Grid Detection:

    There appears to be point-to-point MPI communication in a 32 X 32
    grid pattern. The 20.7% 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 Hilbert rank order
    from the following table.

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

       Hilbert  1.413e+12     81.94%  3
           SMP  1.053e+12     61.04%  1
          Fold  9.405e+11     54.53%  2
    RoundRobin  8.962e+11     51.96%  0
```

More Information from Same Profile

- You don't need to run again for more details:

```
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
```

How to Use CrayPat

- **Make sure the following modules are loaded:**
  - $ module load perftools-base perftools
- **2 instrumentation examples:**
  - $ pat_build my_program
  - $ pat_build –u –g mpi my_program

Same sampling experiment is used when perftools-lite module is loaded

- **Run application**
  - $ aprun -n … my_program+pat
- **Create report**
  - $ pat_report my_program.xf > my_report

CrayPat Runtime Options

- **Runtime controlled through PAT_RT_XXX environment variables**
- **See intro_craypat(1) man page**
- **Examples of control**
  - Enable full trace
  - Change number of data files created
  - Enable collection of HW, network or power counter events
  - Enable tracing filters to control trace file size (max threads, max call stack depth, etc.)

Performance Counters

- **Cray supports raw counters, derived metrics and thresholds for:**
  - Processor (core and uncore)
  - Network
  - Accelerator
  - Power

- **Predefined groups**
  - Groups together counters for experiments

- **See *hwpc, nwpc, accpc, and rapl* man pages**

How to Get List of Events for a Processor

- **Run the following utilities on a compute node:**
  - papi_avail
  - papi_native_avail

- **Set PAT_RT_PERFCTR environment variable to list of events or group prior to execution**

pat_report's Role

- Combines information from binary with raw performance data
- Performs analysis on data
- Generates text report of performance results
- Generates customized instrumentation template for automatic profiling analysis
- Formats data for input into Cray Apprentice[2]

Data from pat_report

- **Default reports are intended to be useful for most applications**
- **Don't need to rerun program to get more detailed data**
- **Different aggregations, or levels of information available**
  - Get fined-grained thread-imbalance information for OpenMP program built with CCE
    - $ pat_report -s pe=ALL –s th=ALL
- **Get list of tables available:**
  - $ pat_report –O –h

A Useful Tip When Viewing Reports. . .

If you don't see the function you are looking for in a report, use pat_report utility to:

- **Disable pruning: "pat_report –P . . ."**
  - This will disable the attempt to associate lower level library routines back to calls from user source
- **Disable thresholding: "pat_report –T . . ."**
  - Include all functions, even those that don't take much time

Sampling with Line Number Information

```
Table 2:  Profile by Group, Function, and Line
  Samp% |      Samp |     Imb. |   Imb. |Group
        |           |     Samp | Samp% | Function
        |           |          |       |  Source
        |           |          |       |   Line
        |           |          |       |    PE=HIDE
 100.0% | 55,605.7 |       -- |    -- |Total
|-----------------------------------------------------------------------------
|  56.5% | 31,412.8 |       -- |    -- |USER
||----------------------------------------------------------------------------
||  19.7% | 10,944.1 |       -- |    -- |create_boundary$boundary_
3|        |          |          |       | source.omp_removed/test/compile/boundary.f90
||||--------------------------------------------------------------------------
4|||   7.8% |  4,355.9 |   175.1 |  3.9% |line.265
4|||   1.8% |    977.0 |    98.0 |  9.1% |line.268
4|||   1.1% |    617.0 |    94.0 | 13.2% |line.273
4|||   2.0% |  1,133.6 |   101.4 |  8.2% |line.549
4|||   1.1% |    590.0 |    66.0 | 10.1% |line.557
||||==========================================================================
||  10.7% |  5,937.8 |       -- |    -- |get_block$blocks_
3|        |          |          |       | source.omp_removed/test/compile/blocks.f90
||||--------------------------------------------------------------------------
4|||   4.1% |  2,305.7 |   145.3 |  5.9% |line.221
4|||   1.0% |    569.5 |    77.5 | 12.0% |line.243
4|||   2.9% |  1,610.1 |   134.9 |  7.7% |line.246
```

Imbalance

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

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

 100.0% | 1.957703 |       -- |     -- | 42,970.8 |Total
|-----------------------------------------------------------------
|  60.0% | 1.174021 |       -- |     -- |  3,602.0 |USER
||----------------------------------------------------------------
||  30.8% | 0.603850 | 0.176924 |  23.0% |  1,198.0 |calc3_
||  19.2% | 0.375117 | 0.128748 |  26.0% |  1,200.0 |calc2_
||   9.1% | 0.178111 | 0.081880 |  32.0% |  1,200.0 |calc1_
||================================================================
|  36.0% | 0.704928 |       -- |     -- |  9,613.0 |MPI_SYNC
||----------------------------------------------------------------
||  25.8% | 0.505174 | 0.385130 |  76.2% |  9,596.0 |mpi_barrier_(sync)
||  10.2% | 0.199537 | 0.199518 | 100.0% |      1.0 |mpi_init_(sync)
||================================================================
|   4.0% | 0.078736 |       -- |     -- | 29,754.8 |MPI
||----------------------------------------------------------------
||   2.3% | 0.045351 | 0.003531 |   7.3% |  9,596.0 |MPI_BARRIER
||   1.1% | 0.021520 | 0.051295 |  71.6% |  8,756.9 |MPI_ISEND
|=================================================================
```

MPI Messages by Caller

```
    MPI |      MPI Msg |  MPI Msg |   MsgSz |  4KiB<= |Function
    Msg |        Bytes |    Count |     <16 |   MsgSz | Caller
 Bytes% |              |          |   Count |  <64KiB |  PE=[mmm]
        |              |          |         |   Count |
 100.0% | 34,940,767.4 |  8,771.9 |   258.6 | 8,513.3 |Total
|-----------------------------------------------------------------
| 100.0% | 34,940,647.4 |  8,756.9 |   243.6 | 8,513.3 |MPI_ISEND
||----------------------------------------------------------------
||  56.2% | 19,622,700.0 |  4,837.5 |    56.2 | 4,781.2 |calc2_
3|        |              |          |         |         | shalow_
||||--------------------------------------------------------------
4|||  56.4% | 19,718,400.0 |  7,200.0 | 2,400.0 | 4,800.0 |pe.0
4|||  56.4% | 19,699,200.0 |  4,800.0 |     0.0 | 4,800.0 |pe.32
4|||  42.3% | 14,784,000.0 |  4,800.0 | 1,200.0 | 3,600.0 |pe.63
||||==============================================================
||  42.5% | 14,851,950.0 |  3,693.8 |    75.0 | 3,618.8 |calc1_
3|        |              |          |         |         | shalow_
||||--------------------------------------------------------------
4|||  56.4% | 19,718,400.0 |  7,200.0 | 2,400.0 | 4,800.0 |pe.0
4|||  42.3% | 14,774,400.0 |  3,600.0 |     0.0 | 3,600.0 |pe.31
4|||  42.3% | 14,774,400.0 |  3,600.0 |     0.0 | 3,600.0 |pe.62
```

MPI Load Imbalance and Message Size

```
Table 2:  Load Balance with MPI Message Stats (limited entries shown)

  Time% |       Time |     MPI Msg |    MPI Msg Bytes |  Avg MPI |Group
        |            |       Count |                  | Msg Size | PE=[mmm]

 100.0% | 262.744329 | 2,460,676.0 | 13,968,292,736.0 | 5,676.61 |Total
|-----------------------------------------------------------------------------
|  91.8% | 241.239771 |         0.0 |              0.0 |       -- |USER
||----------------------------------------------------------------------------
||  93.0% | 244.307970 |         0.0 |              0.0 |       -- |pe.14
||  91.7% | 240.956438 |         0.0 |              0.0 |       -- |pe.21
||  90.8% | 238.650074 |         0.0 |              0.0 |       -- |pe.0
||============================================================================
|   5.7% |  14.851858 | 2,460,676.0 | 13,968,292,736.0 | 5,676.61 |MPI
||----------------------------------------------------------------------------
||   6.2% |  16.358772 | 2,460,676.0 | 13,968,292,736.0 | 5,676.61 |pe.0
||   5.6% |  14.805505 | 2,460,676.0 | 13,968,292,736.0 | 5,676.61 |pe.40
||   4.9% |  12.910010 | 2,460,676.0 | 13,968,292,736.0 | 5,676.61 |pe.14
||============================================================================
|   2.5% |   6.652700 |         0.0 |              0.0 |       -- |MPI_SYNC
||----------------------------------------------------------------------------
||   2.9% |   7.623723 |         0.0 |              0.0 |       -- |pe.43
||   2.6% |   6.726732 |         0.0 |              0.0 |       -- |pe.46
||   1.9% |   5.095840 |         0.0 |              0.0 |       -- |pe.48
```

MPI Message Sizes

```
Total
-----------------------------------------------------------------
  MPI Msg Bytes%                          100.0%
  MPI Msg Bytes                  4,465,684,125.8
  MPI Msg Count                        13,057.0 msgs
  MsgSz <16 Count                        719.0 msgs
  16<= MsgSz <256 Count                   28.0 msgs
  256<= MsgSz <4KiB Count                  0.7 msgs
  4KiB<= MsgSz <64KiB Count              279.8 msgs
  64KiB<= MsgSz <1MiB Count            12,029.6 msgs
=================================================================
  MPI_Send
-----------------------------------------------------------------
  MPI Msg Bytes%                          100.0%
  MPI Msg Bytes                  4,465,680,353.8
  MPI Msg Count                        12,318.0 msgs
  MsgSz <16 Count                          8.0 msgs
  16<= MsgSz <256 Count                    0.0 msgs
  256<= MsgSz <4KiB Count                  0.7 msgs
  4KiB<= MsgSz <64KiB Count              279.8 msgs
  64KiB<= MsgSz <1MiB Count            12,029.6 msgs
```

```
MPI_Send / LAMMPS_NS::Comm::reverse_comm
-----------------------------------------------------------------
  MPI Msg Bytes%                           48.6%
  MPI Msg Bytes                  2,171,466,150.3
  MPI Msg Count                         6,006.0 msgs
  MsgSz <16 Count                           0.0 msgs
  16<= MsgSz <256 Count                     0.0 msgs
  256<= MsgSz <4KiB Count                   0.0 msgs
  4KiB<= MsgSz <64KiB Count                 0.0 msgs
  64KiB<= MsgSz <1MiB Count             6,006.0 msgs
=================================================================
  MPI_Send / LAMMPS_NS::Comm::reverse_comm / LAMMPS_NS::Verlet::run
-----------------------------------------------------------------
  MPI Msg Bytes%                           48.6%
  MPI Msg Bytes                  2,169,218,110.3
  MPI Msg Count                         6,000.0 msgs
  MsgSz <16 Count                           0.0 msgs
  16<= MsgSz <256 Count                     0.0 msgs
  256<= MsgSz <4KiB Count                   0.0 msgs
  4KiB<= MsgSz <64KiB Count                 0.0 msgs
  64KiB<= MsgSz <1MiB Count             6,000.0 msgs
=================================================================
```

MCDRAM Configuration Information

```
CrayPat/X:  Version 6.4.2.36 Revision 8374f24  08/08/16 14:59:22
Experiment:                    lite  lite/sample_profile
Number of PEs (MPI ranks):   2,048
Numbers of PEs per Node:        32  PEs on each of  64  Nodes
Numbers of Threads per PE:       1
Number of Cores per Socket:    512  PEs on sockets with  34  Cores
                             1,536  PEs on sockets with  68  Cores
…

MCDRAM: 7.2 GHz, 16 GiB available as snc2, cache (100% cache)  for 512 PEs
MCDRAM: 7.2 GHz, 16 GiB available as quad, cache (100% cache)  for 1536 PEs

Avg Process Time:       3,251 secs
High Memory:      9,651,837.1 MBytes     4,712.8 MBytes per PE
I/O Read Rate:      23.645208 MBytes/sec
I/O Write Rate:      6.836539 MBytes/sec
Avg CPU Energy:    43,899,476 joules     685,929 joules per node
Avg CPU Power:         13,504 watts       211.00 watts per node
```

Per-numanode Memory High Water Mark

- `$ module load perftools-lite`
- Run program:
  - `1) $ aprun -m1800m -n 2048 -N 16 -d 1 ./cpmd.x`
  - `2) $ aprun -m1800m -n 2048 -N 16 -d 1 \`
    `numactl --preferred=1 ./cpmd.x`
  - `3) $ aprun -m1800m -n 2048 -N 64 -d 1 \`
    `numactl --preferred=1 ./cpmd.x`
- Run pat_report to get memory high water mark (not in reports by default)
  - `$ pat_report -O himem cpmd.x.*ap2 > rpt`
  - or
  - `$ pat_report -O himem –s pe=ALL cpmd.x.*ap2 > rpt_by_pe`

1) Flat Mode, No Allocation

```
MCDRAM: 7.2 GHz, 16 GiB available as quad,  flat (  0% cache)

  Process |    HiMem |Numanode
    HiMem |     Numa | PE=ALL
 (MBytes) |   Node 0 |
          | (MBytes) |

    827.3 |    827.3 |Total
|-------------------------------
|    827.3 |    827.3 |numanode.0
||------------------------------
||    870.5 |    870.5 |pe.1904
||    870.0 |    870.0 |pe.2000
||    869.8 |    869.8 |pe.1776
||    868.6 |    868.6 |pe.1008
||    868.1 |    868.1 |pe.1264
```

16 ranks per core

No allocation to MCDRAM, only to DRAM so one Numanode is used

2) Flat Mode, Full Allocation

```
MCDRAM: 7.2 GHz, 16 GiB available as quad,  flat (  0% cache)

  Process |    HiMem |    HiMem |Numanode
    HiMem |     Numa |     Numa | PE=ALL
 (MBytes) |   Node 0 |   Node 1 |
          | (MBytes) | (MBytes) |

    827.7 |      0.0 |    827.7 |Total
|-----------------------------------------
|    827.7 |      0.0 |    827.7 |numanode.0
||----------------------------------------
||    970.7 |      0.0 |    970.6 |pe.18
||    970.4 |      0.0 |    970.4 |pe.60
||    970.3 |      0.0 |    970.3 |pe.242
||    970.2 |      0.0 |    970.2 |pe.351
||    969.8 |      0.0 |    969.8 |pe.562
||    870.1 |      0.0 |    870.1 |pe.1904
```

16 ranks per core

With numactl --preferred=1, all allocations end up on MCDRAM Numanode 1. No allocations end up on DRAM Numanode 0.

3) Flat Mode, Partial Allocation

```
MCDRAM: 7.2 GHz, 16 GiB available as quad,  flat (  0% cache)

  Process |    HiMem |    HiMem |Numanode
    HiMem |     Numa |     Numa | PE=ALL
 (MBytes) |   Node 0 |   Node 1 |
          | (MBytes) | (MBytes) |

    786.9 |    534.5 |    252.4 |Total
|-----------------------------------------
|    786.9 |    534.5 |    252.4 |numanode.0
||----------------------------------------
||    794.3 |    538.9 |    255.4 |pe.0
||    791.3 |    537.6 |    253.8 |pe.64
||    791.2 |    537.3 |    253.9 |pe.128
||    791.1 |    537.3 |    253.8 |pe.192
||    791.0 |    537.1 |    253.9 |pe.256
||    790.7 |    537.4 |    253.4 |pe.136
||    790.7 |    538.0 |    252.8 |pe.55
```

Using 64 cores on a node creates less available space per MPI rank.

With numactl --preferred=1, after MCDRAM Numanode 1 is full, data is allocated to DRAM Numanode 0.

Questions About the Data?

- **See Job summary information at top of report**
- **See Details section at bottom of report (may include warnings from CrayPat)**
  - Includes information about helper threads found
  - Includes overhead incurred during measurement
- **Check man pages**
- **Check Notes before Tables for data aggregation information, source of data, etc.**

Notes Section

- Check the Notes before each table in the text report

```
Notes for table 5:

  The Total value for Process HiMem (MBytes), Process Time is the avg
  for the PE values.

  The value shown for Process HiMem is calculated from information in
  the /proc/self/numa_maps files captured near the end of the program.
  It is the total size of all pages, including huge pages, that were
  actually mapped into physical memory from both private and shared
  memory segments.

  This table shows only the maximum, median, minimum PE entries,
    sorted by Process Time.
```

Maximize On-node Communication by Reordering MPI ranks

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

MPI Rank Reorder – Two Interfaces Available

- **CrayPat**
  - Available with sampling or tracing
  - Include –g mpi when instrumenting program
  - 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-base module is loaded

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
```

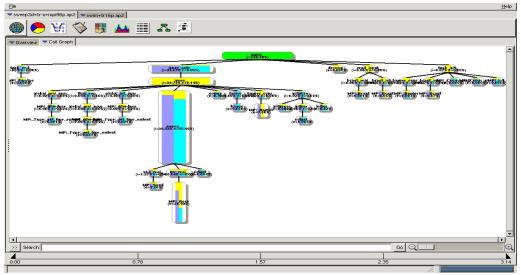
Auto-Generated MPI Rank Order File

```
# 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.
#
0,532,64,564,32,572,96,540,8
,596,72,524,40,604,24,588
104,556,16,628,80,636,56,620
,48,516,112,580,88,548,120,6
12
1,403,65,435,33,411,97,443,9
,467,25,499,105,507,41,475
73,395,81,427,57,459,17,419,
113,491,49,387,89,451,121,48
3
6,436,102,468,70,404,38,412,
14,444,46,476,110,508,78,500
86,396,30,428,62,460,54,492,
118,420,22,452,94,388,126,48
4
129,563,193,531,161,571,225,
539,241,595,233,523,249,603,
185,555
153,587,169,627,137,635,201,
619,177,515,145,579,209,547,
217,611
7,405,71,469,39,437,103,413,
47,445,15,509,79,477,31,501
111,397,63,461,55,429,87,421
,23,493,119,389,95,453,127,4
85
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
5,439,37,407,69,447,101,415,
13,471,45,503,29,479,77,511
53,399,85,431,21,463,61,391,
109,423,93,455,117,495,125,4
87
2,530,34,562,66,538,98,522,1
0,570,42,554,26,594,50,602
18,514,74,586,58,626,82,546,
106,634,90,578,114,618,122,6
10
135,315,167,339,199,347,259,
307,231,371,239,379,191,331,
247,299
175,363,159,323,143,355,255,
291,207,275,183,283,151,267,
215,223
133,406,197,438,165,470,229,
414,245,446,141,478,237,502,
253,398
157,510,189,462,173,430,205,
390,149,422,213,454,181,494,
221,486
130,316,260,340,194,372,162,
348,226,308,234,380,242,332,
250,300
202,364,186,324,154,356,138,
292,170,276,178,284,210,218,
268,146
4,535,36,543,68,567,100,527,
12,599,44,575,28,559,76,607
52,591,20,631,60,639,84,519,
108,623,92,551,116,583,124,6
15
3,440,35,432,67,400,99,408,1
1,464,43,496,27,472,51,504
19,392,75,424,59,456,83,384,
107,416,91,488,115,448,123,4
80
132,401,196,441,164,409,228,
433,236,465,204,473,244,393,
188,497
252,505,140,425,212,457,156,
385,172,417,180,449,148,489,
220,481
131,534,195,542,163,566,227,
526,235,574,203,598,243,558,
187,606
251,590,211,630,179,638,139,
622,155,550,171,518,219,582,
147,614
761,660,737,652,705,668,745,
692,673,700,641,684,713,644,
753,724
729,732,681,756,721,716,764,
676,697,748,689,657,740,665,
649,708
760,528,736,536,704,560,744,
520,672,568,712,592,752,552,
640,600
728,584,680,624,720,512,696,
632,688,616,664,544,608,656,
648,576
762,659,738,651,706,667,746,
643,714,691,674,699,754,683,
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,
337,609,369,577,377,617,329,
513,529
545,297,633,361,625,321,585,
537,601,289,553,353,593,521,
569,561
256,373,261,341,264,349,280,
317,272,381,269,309,285,333,
277,365
352,301,320,325,288,357,328,
304,360,312,376,293,296,368,
336,344
258,338,266,346,282,314,274,
370,766,306,710,378,742,330,
678,362
646,298,750,322,718,354,758,
290,734,662,686,670,726,702,
694,654
262,375,263,343,270,311,271,
351,286,319,278,342,287,350,
279,374
294,318,358,383,359,310,295,
382,326,303,327,367,366,335,
302,334
765,661,709,663,741,653,711,
669,767,655,743,671,749,695,
679,703
677,727,751,693,647,701,717,
687,757,685,733,725,719,735,
645,759
```

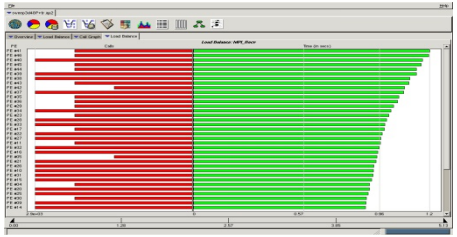
Using New MPI Rank Order

- 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

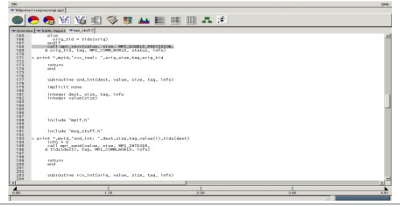
Cray Apprentice^2

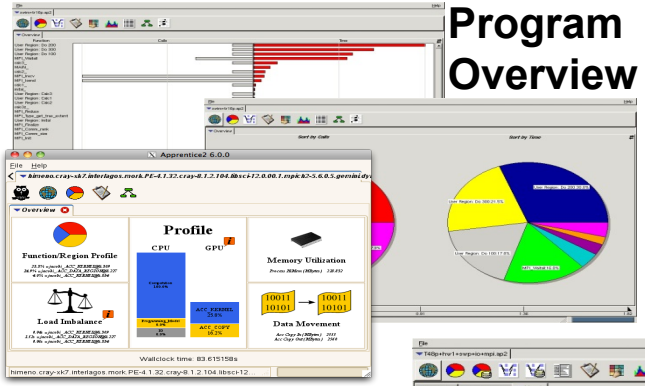


Load balance views



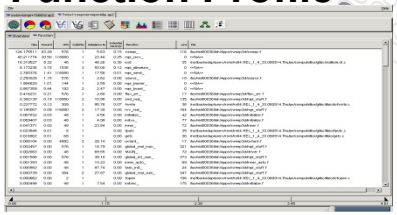
Source code mapping

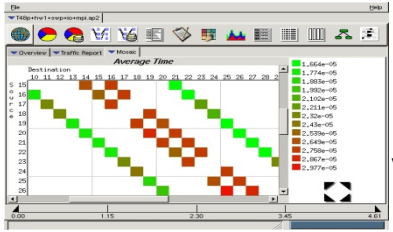




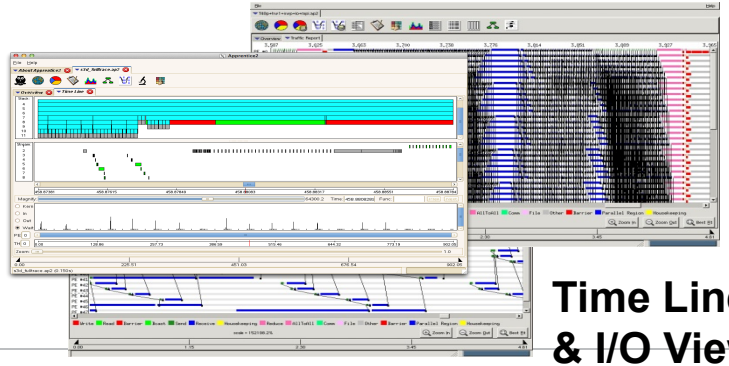
Program Overview

Function Profile

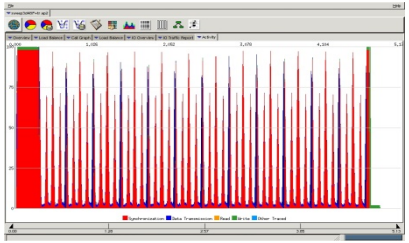




Pair-wise Communication View



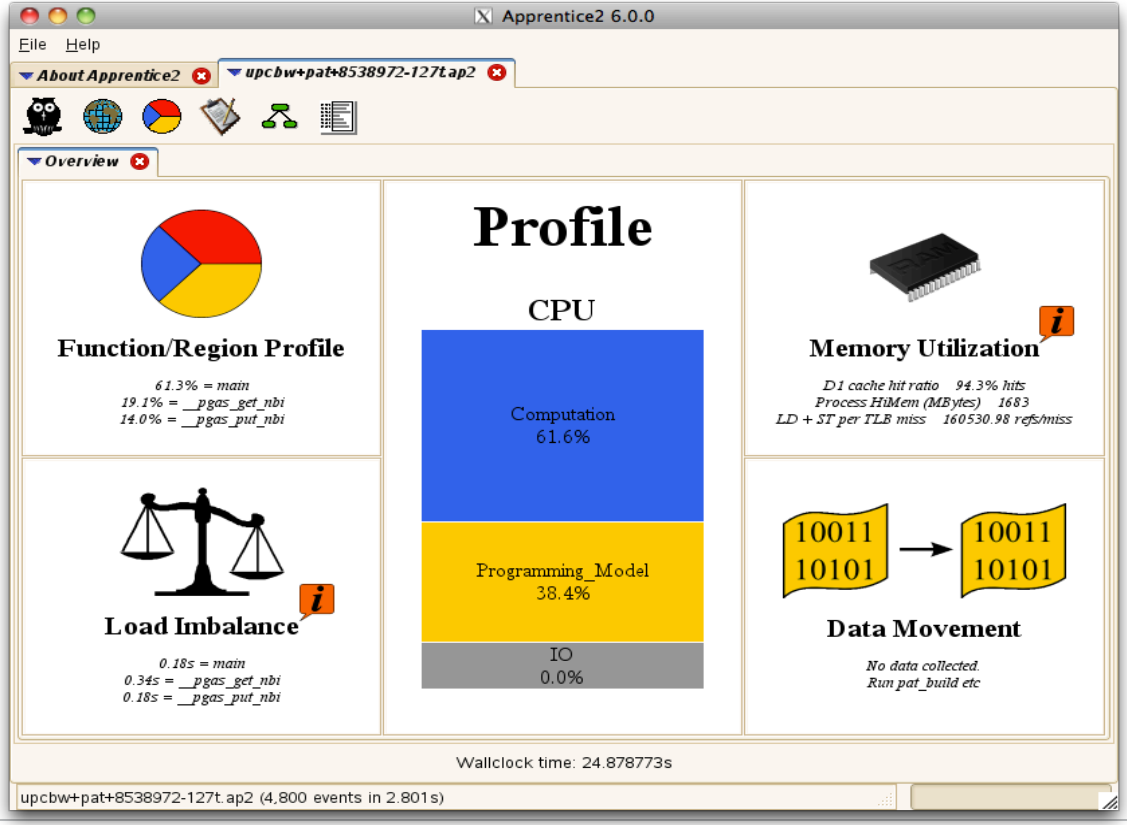
Time Line & I/O Views



Communication & I/O Activity View

COMPUTE | STORE | ANALYZE

Application Performance Summary



Call Tree View

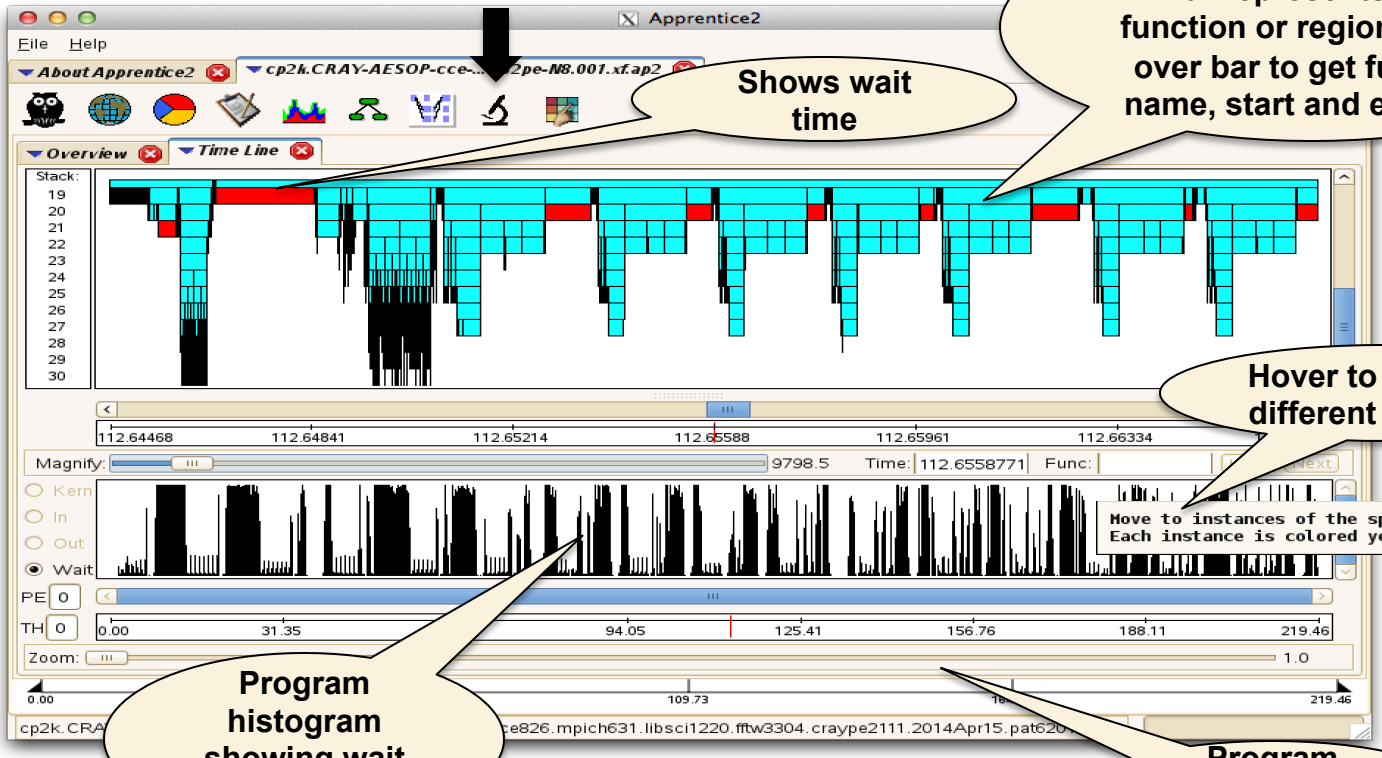
Node width ⇔ inclusive time
Node height ⇔ exclusive time

Load balance overview:
Height ⇔ Max time
Upper bar ⇔ Average time
Lower bar ⇔ Min time
Yellow represents imbalance time

Green colored nodes are not traced.

Data displayed when hovering the mouse over nodes or "**?**".

CPU Program Timeline: 36GB CP2K Full Trace



Shows wait time

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

Hover to see what different filters do

Program histogram showing wait time

Program wallclock time line

Perftools Documentation Available

- **Release Notes**
  - `$ module help perftools/`*`version_number`*
- **User manual "Using the Cray Performance Measurement and Analysis Tools" available at http://docs.cray.com**
- **pat_help – interactive help utility on the Cray Performance toolset**
- **Man pages**
  - craypat, pat_build, pat_report, grid_order

Q&A

Heidi Poxon
heidi@cray.com

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 associated model number marks, are 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 trademarks used in this document are the property of their respective owners.