Addressing Scalability With the Cray Performance Analysis Toolset

Heidi Poxon & Luiz DeRose
Cray, Inc.

July 21, 2009

CRAY
THE SUPERCOMPUTER COMPANY

Multiple Dimensions of Scalability

- Millions of lines of code
  - Automatic profiling analysis
    - Identifies top time consuming routines
    - Automatically creates instrumentation template customized to your application
- Lots of processes/threads
  - Load imbalance analysis
    - Identifies computational code regions and synchronization calls that could benefit most from load balance optimization
    - Estimates savings if corresponding section of code were balanced
- Long running applications
  - Detection of outliers

Automatic Profiling Analysis

- Analyze the performance data and direct the user to meaningful information
- Simplifies the procedure to instrument and collect performance data for novice users

- Based on a two phase mechanism
  1. Automatically detects the most time consuming functions in the application and feeds this information back to the tool for further (and focused) data collection
     - pat_build –O apa a.out
  2. Provides performance information on the most significant parts of the application
     - pat_build –O <a.out.apa>

APA File Example

```
#  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-
          Oapa.512.quad.cores.seal.090405.1154.mpi.pat_rt_exp=default.pat_rt_hwpc=
          none.14999.xf.xf.apa
#
#  These suggested trace options are based on data from:
#
#
          /home/users/malice/pat/Runs/Runs.seal.pat5001.2009Apr04/./pat.quad/homm
          e/standard.cray-xt.PE-2.1.56HD.pgi-8.0.amd64.pat-5.0.0.2-
          Oapa.512.quad.cores.seal.090405.1154.mpi.pat_rt_exp=default.pat_rt_hwpc=
          none.14999.xf.xf.cdb
# ----------------------------------------------------------------------

#       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_

#  9.76%  5474 bytes
        -T derivative_mod_gradient_str_nonstag_

. . .

#  2.95%  3067 bytes
        -T forcing_mod_apply_forcing_

#  2.93%  118585 bytes
        -T column_model_mod_applycolumnmodel_

#  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/amd6
          4/homme/pgi/pat-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.
```

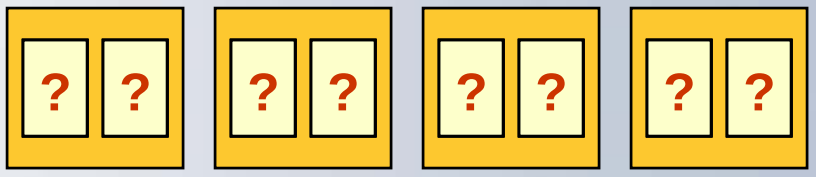
Load Imbalance Analysis

Feedback provided to promote balanced use of requested computing resources

- Profile-guided MPI rank placement suggestions
- Imbalance metrics (both user and MPI functions)

MPI Rank Reorder

- MPI rank placement with environment variable



  - Distributed placement
  - SMP style placement
  - Folded rank placement
  - User provided rank file

Example: -O mpi_rank_order (asura)

```
Notes for table 1:

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

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

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

Table 1:  Sent Message Stats and Suggested MPI Rank Order


                  Sent Msg Total Bytes per MPI rank

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

        378638104    271474542    169280552  56    109

-------------------------------------------------------------
       Quad core:  Sent Msg Total Bytes per node

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

    d   1093188824   1085898170   1071670808  92,124,35,91  86,27,108,63
    u   1093188824   1085898170   1071670808  92,124,35,91  86,27,108,63
    1   1249207480   1085898170    930426320  56,57,58,59   108,109,110,111
    2   1297029256   1085898170    936841176  70,57,71,56   74,53,75,52
    0   1300686504   1085898170    923754472  6,70,7,71     52,116,53,117
```

Example: File MPICH_RANK_ORDER.u (asura)

```
# Suggested custom rank placement:
#
#     pat_report -O mpi_sm_rank_order \
#       /home/crayadm/ldr/ASURA/asura10it.x+apa+4442-824tdt.ap2
#
#   Targets multi-core processors, based on Sent Msg Total Bytes.
#
#   Program:    /work/crayadm/ldr/ASURA/run/asura10it.x
#   Number PEs: 128
#   Cores/Node:  4
#
#   Heuristic: u
#
86,27,108,63,13,67,23,39,70,3,113,17,21,46,40,89
28,36,34,10,7,127,41,105,94,25,12,38,6,75,57,60
56,109,106,68,42,66,43,79,72,45,85,80,33,111,49,107
14,103,114,9,126,52,78,2,55,88,87,118,119,64,15,16
90,102,122,31,37,123,29,59,71,53,98,82,92,124,35,91
5,125,115,11,97,95,30,54,19,4,69,0,62,110,51,112
26,32,121,77,65,100,76,24,58,74,1,18,101,116,84,50
44,96,93,20,83,61,104,47,99,81,120,73,8,117,22,48
```

Imbalance Time

- Metric based on execution time
- It is dependent on the type of activity:
  - User functions

    Imbalance time = Maximum time – Average time
  - Synchronization (Collective communication and barriers)

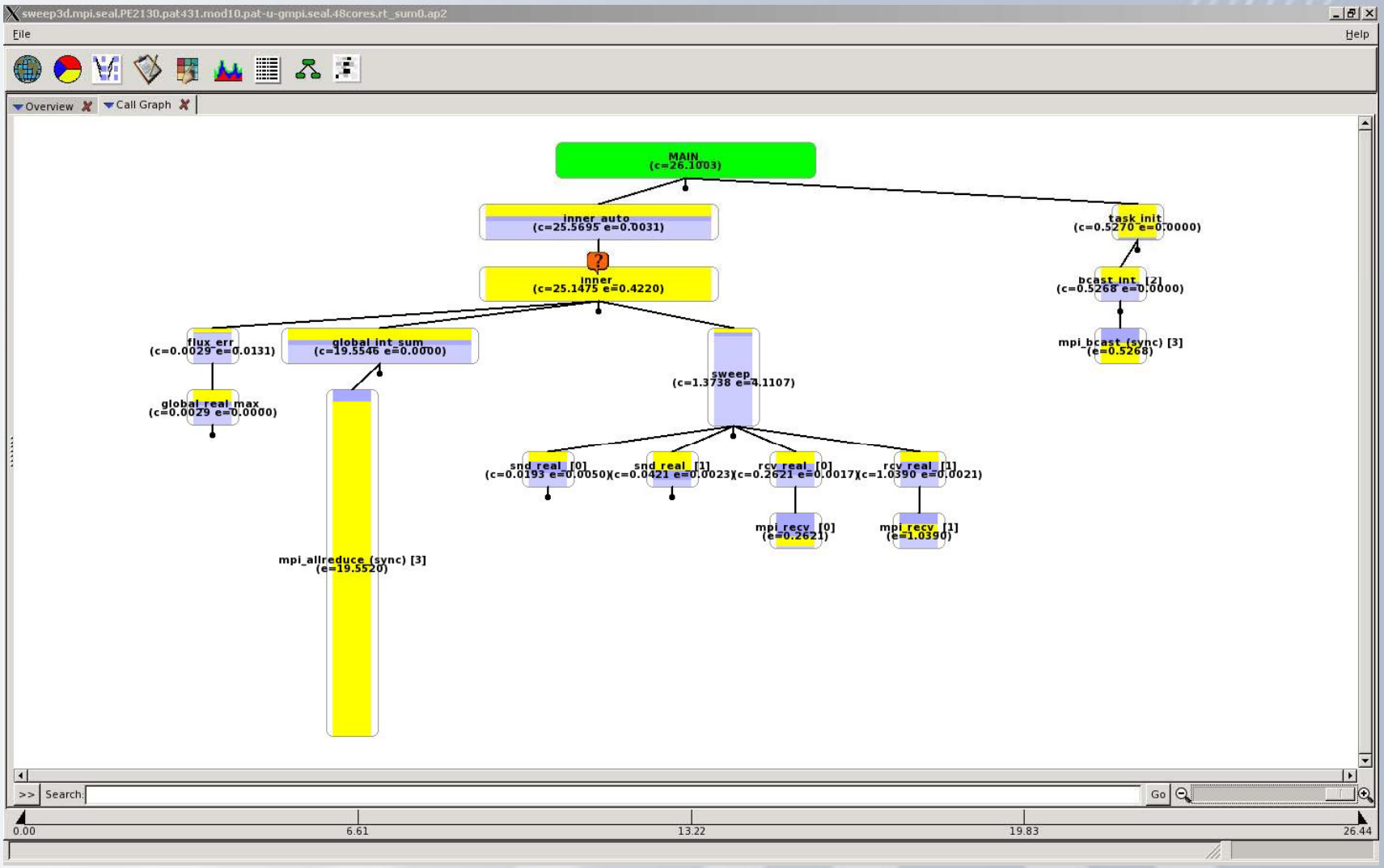
    Imbalance time = Average time – Minimum time
- Identifies computational code regions and synchronization calls that could benefit most from load balance optimization
- Estimates how much overall program time could be saved if corresponding section of code had a perfect balance
  - Represents upper bound on “potential savings”
  - Assumes other processes are waiting, not doing useful work while slowest member finishes

Imbalance %

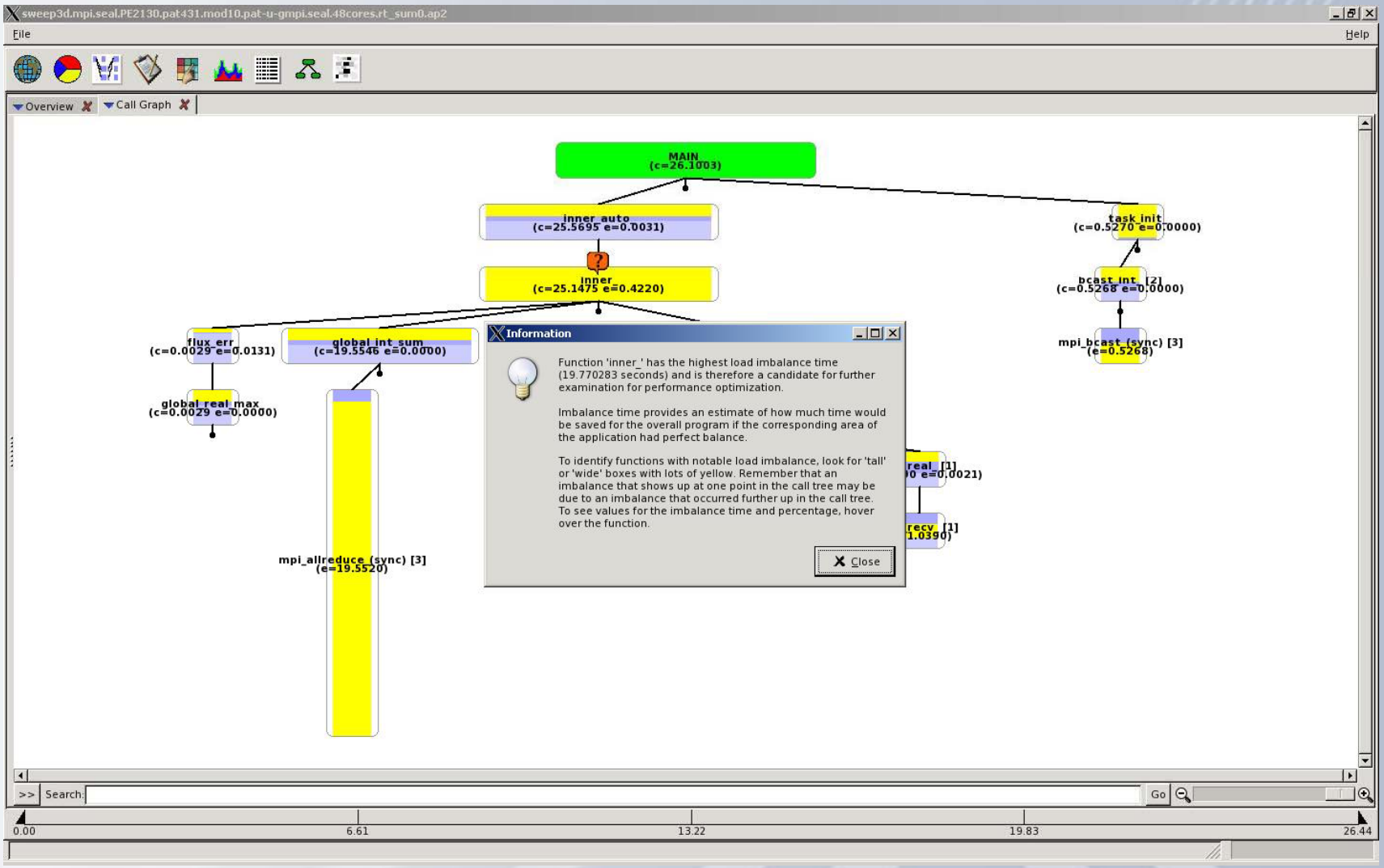
$$\text{Imbalance\%} = 100 \times \frac{\text{Imbalance time}}{\text{Max Time}} \times \frac{N}{N - 1}$$

- Represents % of resources available for parallelism that is “wasted”
- Corresponds to % of time that rest of team is not engaged in useful work on the given function
- Perfectly balanced code segment has imbalance of 0%
- Serial code segment has imbalance of 100%

Call Tree Visualization (Swim3d)



Discrete Unit of Help (DUH Button)



Detection of Outliers

- Adds to automatic profiling analysis functionality (event trace phase)
- Allows the user to detect traveling hot spots
- Focuses on max + std deviation of functions over time
- Will also support memory traffic outliers

Addressing Scalability With the Cray Performance Analysis Toolset

Questions / Comments

Thank You!

July 21, 2009

CRAY
THE SUPERCOMPUTER COMPANY