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.

Copyright 2016 Cray Inc.
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>
<thead>
<tr>
<th>Time%</th>
<th>Time</th>
<th>Imb.</th>
<th>Imb.</th>
<th>Calls</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>Time%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>--------------------------------------------</td>
</tr>
</tbody>
</table>

Table 1: Profile by Function Group and Function (top 7 functions shown)

<table>
<thead>
<tr>
<th>Time%</th>
<th>Time</th>
<th>Imb.</th>
<th>Imb.</th>
<th>Calls</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>--------------------------------------------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time%</th>
<th>Time</th>
<th>Imb.</th>
<th>Imb.</th>
<th>Calls</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>--------------------------------------------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>100.0%</th>
<th>101.961423</th>
<th>--</th>
<th>--</th>
<th>5315211.9</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>---</td>
<td>---</td>
<td>-------------</td>
<td>-------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>92.5%</th>
<th>94.267451</th>
<th>--</th>
<th>--</th>
<th>5272245.9</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>---</td>
<td>---</td>
<td>-------------</td>
<td>-------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>75.8%</th>
<th>77.248585</th>
<th>2.356249</th>
<th>3.0%</th>
<th>1001.0</th>
<th>LAMMPS_NS::PairLJCut::compute</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5%</td>
<td>6.644545</td>
<td>0.105246</td>
<td>1.6%</td>
<td>51.0</td>
<td>LAMMPS_NS::Neighbor::half_bin_newton</td>
</tr>
<tr>
<td>4.1%</td>
<td>4.131842</td>
<td>0.634032</td>
<td>13.5%</td>
<td>1.0</td>
<td>LAMMPS_NS::Verlet::run</td>
</tr>
<tr>
<td>3.8%</td>
<td>3.841349</td>
<td>1.241434</td>
<td>24.8%</td>
<td>5262868.9</td>
<td>LAMMPS_NS::Pair::ev_tally</td>
</tr>
<tr>
<td>1.3%</td>
<td>1.288463</td>
<td>0.181268</td>
<td>12.5%</td>
<td>1000.0</td>
<td>LAMMPS_NS::FixNVE::final_integrate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7.0%</th>
<th>7.110931</th>
<th>--</th>
<th>--</th>
<th>42637.0</th>
<th>MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>---</td>
<td>---</td>
<td>-----------</td>
<td>-----</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4.8%</th>
<th>4.851309</th>
<th>3.371093</th>
<th>41.6%</th>
<th>12267.0</th>
<th>MPI_Send</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5%</td>
<td>1.536106</td>
<td>2.592504</td>
<td>63.8%</td>
<td>12267.0</td>
<td>MPI_Wait</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 2: File Input Stats by Filename

<table>
<thead>
<tr>
<th>Read Time</th>
<th>Read MBytes</th>
<th>Read Rate</th>
<th>Reads</th>
<th>Bytes/ Call</th>
<th>File Name[max10]</th>
<th>PE=HIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>387.432937</td>
<td>13006.522781</td>
<td>33.571030</td>
<td>41596900.0</td>
<td>327.87</td>
<td>Total</td>
<td></td>
</tr>
<tr>
<td>331.691801</td>
<td>1395.829828</td>
<td>4.208213</td>
<td>13153931.0</td>
<td>111.27</td>
<td>/proc/self/maps</td>
<td></td>
</tr>
<tr>
<td>13.129507</td>
<td>4075.682968</td>
<td>310.421627</td>
<td>868.0</td>
<td>4923575.28</td>
<td>regional.grid.a</td>
<td></td>
</tr>
<tr>
<td>12.654338</td>
<td>2000.329418</td>
<td>158.074605</td>
<td>26892862.0</td>
<td>77.99</td>
<td>./patch.input</td>
<td></td>
</tr>
<tr>
<td>3.924810</td>
<td>679.265625</td>
<td>173.069704</td>
<td>3.0</td>
<td>237420544.00</td>
<td>./forcing.radflx.a</td>
<td></td>
</tr>
</tbody>
</table>

...
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 perftoo ls Software

● Load `perftoo ls-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 `perftoo ls/6.3.0` in September 2015

● Prior to `perftoo ls/6.3.0`:
  ● Load `perftoo ls` 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

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

Table 2: Profile by Group, Function, and Line

<table>
<thead>
<tr>
<th>Samp%</th>
<th>Samp</th>
<th>Imb.</th>
<th>Imb.</th>
<th>Group</th>
<th>Function</th>
<th>Source</th>
<th>Line</th>
<th>PE=HIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0%</td>
<td>8376.9</td>
<td>--</td>
<td>--</td>
<td>Total</td>
<td>--</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93.2%</td>
<td>7804.0</td>
<td>--</td>
<td>--</td>
<td>USER</td>
<td>--</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>51.7%</td>
<td>4328.7</td>
<td>--</td>
<td>--</td>
<td>calc3</td>
<td>--</td>
<td>--</td>
<td>heidi/DARPA/cache_util/calc3.do300-ijswap.F</td>
<td></td>
</tr>
<tr>
<td>15.7%</td>
<td>1314.4</td>
<td>93.6</td>
<td>6.8%</td>
<td>line.78</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.9%</td>
<td>1167.7</td>
<td>98.3</td>
<td>7.9%</td>
<td>line.79</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.5%</td>
<td>1211.6</td>
<td>97.4</td>
<td>7.6%</td>
<td>line.80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2%</td>
<td>103.1</td>
<td>26.9</td>
<td>21.2%</td>
<td>line.93</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1%</td>
<td>88.4</td>
<td>22.6</td>
<td>20.8%</td>
<td>line.94</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0%</td>
<td>84.5</td>
<td>17.5</td>
<td>17.6%</td>
<td>line.95</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0%</td>
<td>86.8</td>
<td>33.2</td>
<td>28.2%</td>
<td>line.96</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3%</td>
<td>105.0</td>
<td>23.0</td>
<td>18.4%</td>
<td>line.97</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4%</td>
<td>116.5</td>
<td>24.5</td>
<td>17.7%</td>
<td>line.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total: 144,1 38%
## MPI Messages By Caller

### Table 4: MPI Message Stats by Caller

<table>
<thead>
<tr>
<th>MPI Msg Bytes</th>
<th>MPI Msg</th>
<th>MsgSz</th>
<th>4KB≤</th>
<th>4KB≤</th>
<th>Function</th>
<th>Count</th>
<th>Count</th>
<th>&lt;64KB</th>
<th>PE=[mmm]</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>140166953.8</td>
<td>8890.6</td>
<td>339.8</td>
<td>8550.8</td>
<td></td>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>140166833.8</td>
<td>8875.6</td>
<td>324.8</td>
<td>8550.8</td>
<td></td>
<td>MPI_ISEND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>78272400.0</td>
<td>4850.0</td>
<td>75.0</td>
<td>4775.0</td>
<td>calc2_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>78272400.0</td>
<td>4850.0</td>
<td>75.0</td>
<td>4775.0</td>
<td>calc2_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>78700800.0</td>
<td>7200.0</td>
<td>2400.0</td>
<td>4800.0</td>
<td>pe.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>78681600.0</td>
<td>4800.0</td>
<td>0.0</td>
<td>4800.0</td>
<td>pe.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59020800.0</td>
<td>4800.0</td>
<td>1200.0</td>
<td>3600.0</td>
<td>pe.47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59421800.0</td>
<td>3725.0</td>
<td>100.0</td>
<td>3625.0</td>
<td>calc1_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59001200.0</td>
<td>3600.0</td>
<td>0.0</td>
<td>3600.0</td>
<td>pe.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59001200.0</td>
<td>3600.0</td>
<td>0.0</td>
<td>3600.0</td>
<td>pe.24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

624,3 79%
# 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.
```

# HWPC group to collect by default.

```
-Drtenv=PT_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
-s apa_min_size=800
-s apa_max_cum_pct=90
```

Local functions are listed for completeness, but cannot be
traced.

```
-w
```

# 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
```

# Original instrumented program.

```
./AUTO/cray/ccs.pe/tools/malice/craypat/build/pat/2009Apr03/2.1.56HD/xt.PE-2.1.56HD.pgi-8.0.amd64.pat-5.0.0.2.x # Original program.
```

# New instrumented program.

```
amd64/home/pga7/pat-5.0.0.2/pt_home/2005bec08/builid.Linux/preqx.cray-
```
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
<table>
<thead>
<tr>
<th>Time%</th>
<th>Time</th>
<th>Imb.</th>
<th>Imb.</th>
<th>Calls</th>
<th>Group</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>Time</td>
<td></td>
<td></td>
<td>PE=HIDE</td>
</tr>
<tr>
<td>100.0%</td>
<td>463.147240</td>
<td>--</td>
<td>--</td>
<td>21621.0</td>
<td>Total</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>------------------</td>
</tr>
<tr>
<td>52.0%</td>
<td>240.974379</td>
<td>--</td>
<td>--</td>
<td>21523.0</td>
<td>MPI</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>------------------</td>
</tr>
<tr>
<td>47.7%</td>
<td>221.142266</td>
<td>36.214468</td>
<td>14.1%</td>
<td>10740.0</td>
<td>mpi_recv</td>
<td></td>
</tr>
<tr>
<td>4.3%</td>
<td>19.829001</td>
<td>25.849906</td>
<td>56.7%</td>
<td>10740.0</td>
<td>MPI_SEND</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>------------------</td>
</tr>
<tr>
<td>43.3%</td>
<td>200.474690</td>
<td>--</td>
<td>--</td>
<td>32.0</td>
<td>USER</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>------------------</td>
</tr>
<tr>
<td>41.0%</td>
<td>189.897060</td>
<td>58.716197</td>
<td>23.6%</td>
<td>12.0</td>
<td>sweep_</td>
<td></td>
</tr>
<tr>
<td>1.6%</td>
<td>7.579876</td>
<td>1.899097</td>
<td>20.1%</td>
<td>12.0</td>
<td>source_</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>------------------</td>
</tr>
<tr>
<td>4.7%</td>
<td>21.698147</td>
<td>--</td>
<td>--</td>
<td>39.0</td>
<td>MPI_SYNC</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>------------------</td>
</tr>
<tr>
<td>4.3%</td>
<td>20.091165</td>
<td>20.005424</td>
<td>99.6%</td>
<td>32.0</td>
<td>mpi_allreduce_(sync)</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>------------------</td>
</tr>
<tr>
<td>0.0%</td>
<td>0.000024</td>
<td>--</td>
<td>--</td>
<td>27.0</td>
<td>SYSCALL</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Profile by Function Group and Function
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.

<table>
<thead>
<tr>
<th>Rank Order</th>
<th>On-Node Bytes/PE</th>
<th>On-Node Bytes/PE% of Total Bytes/PE</th>
<th>MPICH_RANK_REORDER_METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom</td>
<td>2.385e+09</td>
<td>95.55%</td>
<td>3</td>
</tr>
<tr>
<td>SMP</td>
<td>1.880e+09</td>
<td>75.30%</td>
<td>1</td>
</tr>
<tr>
<td>Fold</td>
<td>1.373e+06</td>
<td>0.06%</td>
<td>2</td>
</tr>
<tr>
<td>RoundRobin</td>
<td>0.000e+00</td>
<td>0.00%</td>
<td>0</td>
</tr>
</tbody>
</table>
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:
# 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.
# 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.
# 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.
# 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.
# 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.
# 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.
# 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.
# 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.
# 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.
# 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.
# 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.
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

<table>
<thead>
<tr>
<th>Counter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPI_TLB_DM</td>
<td>Data translation lookaside buffer misses</td>
</tr>
<tr>
<td>PAPI_L1_DCA</td>
<td>Level 1 data cache accesses</td>
</tr>
<tr>
<td>PAPI_FP_OPS</td>
<td>Floating point operations</td>
</tr>
<tr>
<td>DC_MISS</td>
<td>Data Cache Miss</td>
</tr>
<tr>
<td>User_Cycles</td>
<td>Virtual Cycles</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>USER</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Time%</td>
<td>98.3%</td>
</tr>
<tr>
<td>Time</td>
<td>4.434402 secs</td>
</tr>
<tr>
<td>Imb.Time</td>
<td>-- secs</td>
</tr>
<tr>
<td>Imb.Time%</td>
<td>--</td>
</tr>
<tr>
<td>Calls</td>
<td>0.001M/sec 4500.0 calls</td>
</tr>
<tr>
<td>PAPI_L1_DCM</td>
<td>14.820M/sec 65712197 misses</td>
</tr>
<tr>
<td>PAPI_TLB_DM</td>
<td>0.902M/sec 3998928 misses</td>
</tr>
<tr>
<td>PAPI_L1_DCA</td>
<td>333.331M/sec 1477996162 refs</td>
</tr>
<tr>
<td>PAPI_FP_OPS</td>
<td>445.571M/sec 1975672594 ops</td>
</tr>
<tr>
<td>User time (approx)</td>
<td>4.434 secs 11971868993 cycles 100.0% Time</td>
</tr>
<tr>
<td>Average Time per Call</td>
<td>0.000985 sec</td>
</tr>
<tr>
<td>CrayPat Overhead : Time</td>
<td>0.1%</td>
</tr>
<tr>
<td>HW FP Ops / User time</td>
<td>445.571M/sec 1975672594 ops 4.1% peak(DP)</td>
</tr>
<tr>
<td>HW FP Ops / WCT</td>
<td>445.533M/sec</td>
</tr>
<tr>
<td>Computational intensity</td>
<td>0.17 ops/cycle 1.34 ops/ref</td>
</tr>
<tr>
<td>MFLOPS (aggregate)</td>
<td>1782.28M/sec</td>
</tr>
<tr>
<td>TLB utilization</td>
<td>369.60 refs/miss 0.722 avg uses</td>
</tr>
<tr>
<td>D1 cache hit,miss ratios</td>
<td>95.6% hits 4.4% misses</td>
</tr>
<tr>
<td>D1 cache utilization (misses)</td>
<td>22.49 refs/miss 2.811 avg hits</td>
</tr>
</tbody>
</table>

**Derived Metrics**

- **PAT_RT_PERFCTR=1**
- Flat profile data
- Raw counts
- Derived metrics

---

*March 2016 © Cray Inc. Proprietary*
Visualizing Performance of Your Application Through Cray Apprentice2
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

Node width \(\approx\) inclusive time
Node height \(\approx\) exclusive time

Load balance overview:
Height \(\approx\) Max time
Upper bar \(\approx\) Average time
Lower bar \(\approx\) Min time
Yellow represents imbalance time

Provides hints for performance tuning

Filtered node or sub tree
Green colored nodes are not traced.
Data displayed when hovering the mouse over nodes or “?”. Zoom
CPU Program Timeline: 36GB CP2K Full Trace

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

Shows wait time

Program histogram showing wait time

Program wallclock time line

Hover to see what different filters do
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

![Comparison Diagram]

- **104.34 secs** for mg.C.4.cce84.lite-loops_ap2
- **26.33 secs** for mg.C.16.cce84.lite-loops_ap2
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)
NumHPC: 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

© Cray Inc. Proprietary
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
  - \texttt{PAT\_RT\_SUMMARY} = 0
  - \texttt{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

© Cray Inc. Proprietary
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)

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

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

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

G = accelerator kernel

g = partitioned loop

Numbers denote serial loops

source line numbers
Data Movement Feedback

163. 1------< DO loop = 1,nn
169. 1 gosal = 0
171. 1 G-----< !$acc parallel loop reduction(+:gosal) private(i,j,k,s0,ss)
172. 1 g------< DO k = 2,kmax-1
173. 1 g 3-----< DO j = 2,jmax-1
174. 1 g 3 g--< DO i = 2,imax-1
175. 1 g 3 g  s0 = a(i,j,k,1) * p(i+1,j,k) ...
188. 1 g 3 g--> ENDDO
189. 1 g 3----> ENDDO
190. 1 g------> ENDDO
191. 1 !$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 gosal = 0
171. 1 G-----> !$acc parallel loop reduction(+:gosal) private(i,j,k,s0,ss)
172. 1 g-------< DO k = 2,kmax-1
173. 1 g 3----< DO j = 2,jmax-1
174. 1 g 3 g--< DO i = 2,imax-1
175. 1 g 3 g--> s0 = a(i,j,k,1) * p(i+1,j,k) ...
178. 1 g 3 g--> ENDDO
179. 1 g 3----> ENDDO
180. 1 g------> ENDDO
181. 1 g------> !$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.
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

Table 1: Profile by Function Group and Function

<table>
<thead>
<tr>
<th>Time's</th>
<th>Time</th>
<th>Inb.</th>
<th>Inb.</th>
<th>Calls</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0%</td>
<td>25.675616</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>Total</td>
</tr>
<tr>
<td>96.5%</td>
<td>24.769662</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>USER</td>
</tr>
<tr>
<td>72.1%</td>
<td>18.521376</td>
<td>0.042997</td>
<td>0.3%</td>
<td>10.03</td>
<td>jacobi_ACC_SYNC_WAIT@1.331</td>
</tr>
<tr>
<td>10.2%</td>
<td>2.612815</td>
<td>0.000694</td>
<td>0.0%</td>
<td>10.03</td>
<td>jacobi_ACC_SYNC_WAIT@1.382</td>
</tr>
<tr>
<td>5.9%</td>
<td>1.761435</td>
<td>0.011827</td>
<td>0.0%</td>
<td>10.03</td>
<td>jacobi_ACC_COPY@1.271</td>
</tr>
<tr>
<td>4.9%</td>
<td>1.246693</td>
<td>0.028293</td>
<td>2.5%</td>
<td>10.03</td>
<td>jacobi_ACC_COPY@1.382</td>
</tr>
</tbody>
</table>

3.3% 0.050654 ... 150.055.0 MPI

3.1% 0.000000 ... 3009.0 mpi_waitall 1.0 1.0

Observations and suggestions

Number of accelerators used: 8 of 8

Wallclock time: 25.309761 s

Time CPU waits while GPU executes

Data transfer to and from the GPU
### Example Accelerator Statistics

<table>
<thead>
<tr>
<th>Table 1: Time and Bytes Transferred for Accelerator Regions</th>
</tr>
</thead>
<tbody>
<tr>
<td>**Host</td>
</tr>
<tr>
<td>Time%</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>100.0%</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>63.5%</td>
</tr>
<tr>
<td>22.1%</td>
</tr>
<tr>
<td>20.6%</td>
</tr>
<tr>
<td>18.8%</td>
</tr>
<tr>
<td>1.6%</td>
</tr>
<tr>
<td>1.1%</td>
</tr>
<tr>
<td>1.1%</td>
</tr>
</tbody>
</table>
Apprentice2 Overview with GPU Data

![Image of Apprentice2 Overview with GPU Data]

- **Overview**
- **Profile**
  - CPU: 100.0%
  - GPU: ACC_KERNEL 35.8%, ACC_COPY 16.2%
- **Memory Utilization**
  - Process Heap (2GB): 21.832
- **Data Movement**
  - Acc Copy In (2GB): 2155
  - Acc Copy Out (2GB): 2300

Wallclock time: 83.615158s

© Cray Inc. Proprietary
GPU Program Timeline

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

Bar represents GPU stream event: Hover over bar to get event info

Navigation assistance

Program histogram of wait, copy kernel time

Program wallclock time line