Performance Tools Documentation and Tips

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.
Cray PE Documentation Available

- **Release Notes**
  - `> module help product/product_version`

- **User Guides**
  - [http://docs.cray.com](http://docs.cray.com)

- **Man pages, for example:**
  - `cc`
  - `crayftn`
  - `intro_directives`
  - `Intro_openacc`
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](http://docs.cray.com)

- **pat_help** – interactive help utility on the Cray Performance toolset

- **Man pages**
Man pages

- **intro_craypat(1)**
  - Introduces the craypat performance tool
  - Runtime environment variables (enable full trace, etc.)

- **pat_build(1)**
  - Instrument a program for performance analysis

- **pat_help(1)**
  - Interactive online help utility

- **pat_report(1)**
  - Generate performance report in both text and for use with GUI

- **app2 (1)**
  - Describes how to launch Cray Apprentice2 to visualize performance data
Man pages (2)

- **hwpc(5)**
  - describes predefined hardware performance counter groups

- **nwpc(5)**
  - Describes predefined network performance counter groups

- **accpc(5) / accpc_k20(5), etc.**
  - Describes predefined GPU performance counter groups

- **intro_papi(3)**
  - Lists PAPI event counters
  - Use papi_avail or papi_native_avail utilities to get list of events when running on a specific architecture
Reveal Usage Recipe

● Access Cray compiler
  ● > module load PrgEnv-cray

● Access perftools
  ● > module load perftools-base

● Enable loop work estimates program instrumentation
  ● > module load perftools-lite-loops

● Build program (make)

● Run program to get loop work estimates in file with .ap2 suffix
Reveal Usage Recipe (2)

● Disable loop work estimates program instrumentation so we can get fully optimized program now
  ● > module unload perftools-lite-loops

● Create program library with CCE:
  ● Add –h pl=/full_path/my_program.pl to program’s Makefile

● Rebuild application with full optimization
  ● > make clean
  ● > make

● Launch Reveal
  ● > reveal /full_path/my_program.pl loop_work_estimates.ap2
How to Install Apprentice2 on Your Laptop

● > module load perftools

● Go to:
  ● $CRAYPAT_ROOT/share/desktop_installers/

● Download .dmg or .exe installer

● Double click on installer and follow directions to install
Why Should I generate a “.ap2” file?

- The “.ap2” file is a self contained compressed performance file
- Normally it is about 5 times smaller than the “.xf” file
- Contains the information needed from the application binary
  - Can be reused, even if the application binary is no longer available or if it was rebuilt
- It is the only input format accepted by Cray Apprentice $^2$
### Files Generated and the Naming Convention

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

● Get list of tables available:
  ● > pat_report –O –h

● Other formats available (txt, html, csv, xml)
A Useful Tip. . .

If you don’t see the function you are looking for in a report:

- Disable pruning: “pat_report –P . . .”
- Disable thresholding: “pat_report –T . . .”
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.
Questions About the Data?

● See Job summary information at top of report

● See Details section at bottom of report (may include warnings from CrayPat)

● Check `pat_help`

● Check man pages
Pat_help

● > pat_help environment . . .

pat_help environment (.=quit ,=back ^=up /=top =~search)
=> PAT_RT_SAMPLING_DATA

Specifies additional data to collect during a sampling experiment. The valid values are shown below.

The value may be followed by '@ratio' which indicates the frequency at which the data is sampled. By default the data is sampled once for every 100 sampled program counter addresses. For example, if 'ratio' is '1', the additional data requested would be collected each time the program counter is sampled. If the 'ratio' is '1000', the additional data requested would be collected once every 1000 program counter samples.

Collecting additional data during sampling is only supported in full-trace mode (see PAT_RT_SUMMARY).

Additional topics that may follow "PAT_RT_SAMPLING_DATA":

cray_pm      perfctr

cray_rapl    rusage

heap         sheep

memory
> pat_help environment PAT_RT_SAMPLING_DATA memory

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>himem</td>
<td>memory high water mark</td>
</tr>
<tr>
<td>rss</td>
<td>resident set size</td>
</tr>
<tr>
<td>peak</td>
<td>maximum virtual memory used</td>
</tr>
<tr>
<td>priv</td>
<td>private resident memory</td>
</tr>
<tr>
<td>shared</td>
<td>shared resident memory</td>
</tr>
<tr>
<td>proportional</td>
<td>proportional resident memory</td>
</tr>
</tbody>
</table>