Enhancements to the Cray Performance Measurement and Analysis Tools

Heidi Poxon
Technical Lead & Manager, Performance Tools
Cray Inc.
Strengths

Provide a complete solution from instrumentation to measurement to analysis to visualization of data

- Performance measurement and analysis on large systems
  - Automatic Profiling Analysis
  - Load Imbalance
  - HW counter derived metrics
  - Predefined trace groups provide performance statistics for libraries called by program (blas, lapack, pgas runtime, netcdf, hdf5, etc.)
  - Observations of inefficient performance
  - Data collection and presentation filtering
  - Data correlates to user source (line number info, etc.)
  - Support MPI, SHMEM, OpenMP, UPC, CAF, OpenACC
  - Access to network counters
  - Minimal program perturbation
The Cray Performance Analysis Framework

● **Supports traditional post-mortem performance analysis**
  ● Automatic identification of performance problems
    ● 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_report** performs analysis and generates text reports
● **pat_help**: online help utility
● **Cray Apprentice2**: graphical visualization tool

● **To access software:**
  ● module load perftools
Recent Enhancements

- Apprentice2 for the Mac
- Aries™ network counters
- PAPI Cray network component
- Apprentice2 application performance summary
- Reveal 1.0 released
- CrayPat-lite
Application Performance Summary with GPUs
New to Reveal!
Try "Getting Started" in the "Help" Menu

ISOMP parallel do (declare none)
ISOMP S shared (gamma, sendi, z00, z01, zp0, zp1, zv0, zv1, zv2, zv3) &
ISOMP S lastprivate (dx, dy0, dx, f, p, t, r, x, k, xk, xk0)

Cray Inc.
● Message filtering (allows you to identify all loops that didn’t vectorize or which functions were not inlined, etc.)

● New “Insert All Valid Directives” menu option (which inserts OpenMP directives for all “green” loops scoped successfully)

● Usability enhancements (Example: scoping loop selection and results windows combined into one tabbed window (to reduce the number of additional windows that “pop-up”))
Message Filtering

Default filter: Loops that didn’t vectorize. Can select other filters.
CrayPat-lite
CrayPat-lite Goals

- Provide automatic application performance statistics at the end of a job
  - Focus is to offer a simplified interface to basic application performance information for users not familiar with the Cray performance tools and perhaps new to application performance analysis
  - Provides a simple performance summary mechanism for Cray performance tools users before they move on to more detailed analysis with classic perftools
  - Gives sites the option to enable/disable application performance data collection for all users for a period of time

- Keep traditional or “classic” perftools working the same as before

- Provide a simple way to transition from perftools-lite to perftools to encourage further tool use for performance analysis
Steps to Using CrayPat “classic”

Access performance tools software

> module load perftools

Build program, retaining .o files

> make

Instrument binary

> pat_build –O apa a.out

Modify batch script and run program

aprun a.out+pat

Process raw performance data and create report

> pat_report a.out+pat*.xf

a.out+pat*.ap2
Text report to stdout
a.out+pat*.apa
MPICH_RANK_XXX
Steps to Using CrayPat-lite

Access light version of performance tools software

> 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
MPICH_RANK_XXX files
Benefits of CrayPat-lite

- Program is automatically relinked to add instrumentation in a.out (pat_build step done for the user)
- .o files are automatically preserved
- No modifications are needed to a batch script to run instrumented binary, since original binary is replaced with instrumented version
- pat_report is automatically run before job exits
- Performance statistics are issued to stdout
- User can use “classic” CrayPat for more in-depth performance investigation
Performance Statistics Available

- Job information
  - Number of MPI ranks
  - Number of PEs per node
  - Number of threads
  - Number of cores per socket
  - Execution start time
  - System name and speed

- Wallclock
- High memory water mark
- Aggregate MFLOPS (CPU only)
Performance Statistics Available (2)

- Profile of top time consuming routines with load balance information by group (user functions, MPI, etc.)

- Observations
  - Currently reporting MPI rank reorder suggestions if applicable

- Instructions on how to access additional information that is available
Predefined Set of Performance Experiments

- Set of predefined experiments, enabled with the CRAYPAT_LITE environment variable
  - sample_profile
  - event_profile
  - GPU

What do the predefined events mean to someone familiar with the Cray performance tools?
CRAYPAT_LITE=sample_profile

- Default experiment

- Equivalent to “pat_build -O apa a.out”

- Provides profile based on sampling
  - Includes collection of summary CPU performance counters around MAIN (for MFLOPS)
  - Includes Imbalance information

- More information available in .ap2 file
  - Can get classic report by running pat_report
CRAYPAT_LITE=event_profile

- Provides profile based on summarization of events
- Includes OpenMP and OpenACC information if these models are used within program

Equivalent to "pat_build -u -gmpi a.out" +
  - Collection of summary CPU performance counters
  - Filter to only trace functions above 1200 bytes
    - In most cases, omits tiny repetitive functions that can perturb results (like ranf())
    - Can give coarser granularity results over classic perftools

- More information available in .ap2 file
CRAYPAT_LITE=GPU

- Provides more detailed OpenACC GPU statistics
- Equivalent to "pat_build -w a.out" (coarsest granularity tracing, around MAIN)
- Output similar to classic perftools accelerator table
  - Includes host and device time
  - Bytes transferred between host and device
  - Time to transfer data between host and device
- More information available in .ap2 file
Default Output – Job Summary Info

CrayPat/X:  Version 6.1.0.10929 Revision 10929... 03/04/13 23:51:00
Experiment:                  lite  sample_profile
Number of PEs (MPI ranks):     64
Numbers of PEs per Node:       32  PEs on each of  2  Nodes
Numbers of Threads per PE:      1
Number of Cores per Socket:    16
Execution start time:  Tue Mar  5 18:17:03 2013
System name and speed:  mork 2100 MHz

Wall Clock Time:    75.432429 secs
High Memory:            43.96 MBytes
MFLOPS (aggregate):  25718.61 M/sec
## Table 1: Profile by Function Group and Function (top 7 functions shown)

<table>
<thead>
<tr>
<th>Samp %</th>
<th>Samp</th>
<th>Imb.</th>
<th>Imb.</th>
<th>Group</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Samp</td>
<td>Samp</td>
<td></td>
<td>PE=HIDE</td>
</tr>
</tbody>
</table>

| | | | | | |
| --- | --- | --- | --- | --- | |
| 100.0% | 7422.6 | -- | -- | Total | |

| | | | | | |
| 88.7% | 6585.5 | -- | -- | USER | |

| | | | | | |
| 71.7% | 5325.1 | 111.9 | 2.1% | LAMMPS_NS::PairLJCut::compute | |
| 8.9%  | 659.2  | 17.8  | 2.7% | LAMMPS_NS::Neighbor::half_bin_newton | |
| 3.1%  | 227.7  | 67.3  | 23.2%| LAMMPS_NS::FixNVE::initial_integrate | |
| 1.7%  | 124.9  | 27.1  | 18.1%| LAMMPS_NS::FixNVE::final_integrate | |
| 1.5%  | 112.0  | 16.0  | 12.7%| LAMMPS_NS::Verlet::run | |

| | | | | | |
| 11.1% | 825.0 | -- | -- | MPI | |

| | | | | | |
| 7.2% | 534.0 | 199.0 | 27.6% | MPI_Send | |
| 3.0% | 226.1 | 135.9 | 38.1% | MPI_Wait | |
Default Output – For More Information…

Program invocation:
```
lammps.x -var x 4 -var y 2 -var z 8
```

For more detailed performance reports, run:
```
pat_report /lus/scratch/test/lammps.x.lj.64pe.32ppn.ap2
```

For interactive performance analysis, run:
```
app2 /lus/scratch/test/lammps.x.lj.64pe.32ppn.ap2
```

End of CrayPat output.
Event Profile Output - Observations

================= Observations and suggestions =================

MPI Grid Detection:

There appears to be point-to-point MPI communication in a 4 X 2 X 8 grid pattern. The 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.

<table>
<thead>
<tr>
<th>Rank Order</th>
<th>On-Node Bytes/PE</th>
<th>On-Node Bytes/PE% of Total</th>
<th>MPICH_RANK_REORDER_METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hilbert</td>
<td>5.533e+10</td>
<td>90.66%</td>
<td>3</td>
</tr>
<tr>
<td>Fold</td>
<td>4.907e+10</td>
<td>80.42%</td>
<td>2</td>
</tr>
<tr>
<td>SMP</td>
<td>4.883e+10</td>
<td>80.02%</td>
<td>1</td>
</tr>
<tr>
<td>RoundRobin</td>
<td>3.740e+10</td>
<td>61.28%</td>
<td>0</td>
</tr>
</tbody>
</table>
What’s Next...
User inserted directive with mis-scoped variable ‘l’
GPU Timeline

Host call chain for time segment (MAIN is top bar). Hover to see function name.

Device stream activity for time segment. Hover to see GPU event (copy, etc.)

PE host and device activity for timeline

Segment from timeline
Questions