Cray Performance Tools Enhancements for Next Generation Systems

Heidi Poxon

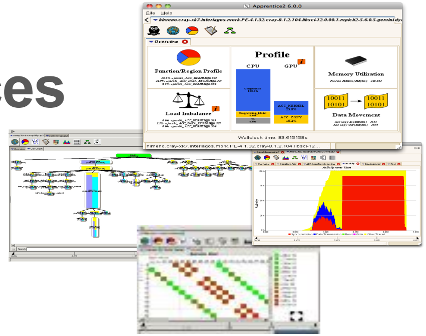
Agenda

- Cray Performance Tools Overview
- Recent Enhancements
- Support for Cray systems with KNL

Cray Performance Analysis Tools Overview

- **Assist the user with application performance analysis and optimization**
  - Help user identify important and meaningful information from potentially massive data sets
  - Help user identify problem areas instead of just reporting data
  - Bring optimization knowledge to a wider set of users

- **Focus on ease of use and intuitive user interfaces**
  - Automatic program instrumentation
  - Automatic analysis



- **Whole program analysis across many nodes**

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

"Lite" Mode

Load performance tools modules

```
> 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 CrayPat-lite Output

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

CrayPat/X:  Version 6.3.2.461 Revision 56930ff  02/01/16 15:31:33
Experiment:                  lite  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 Feb  2 18:53:50 2016
System name and speed:  kay  2301 MHz (approx)

Avg Process Time:          64.38 secs
High Memory:               1,563 MBytes  24.43 MBytes per PE
MFLOPS:            Not supported (see observation below)
I/O Read Rate:         48.514130 MBytes/sec
I/O Write Rate:        22.281350 MBytes/sec
Avg CPU Energy:           41,820 joules 20,910 joules per node
Avg CPU Power:            649.53 watts  324.77 watts per node
```

```
Table 1:  Profile by Function Group and Function (top 10 functions
shown)

  Samp% |    Samp |  Imb. |  Imb. |Group
        |         |  Samp | Samp% | Function
        |         |       |       |  PE=HIDE

 100.0% | 6,156.2 |    -- |    -- |Total
|-----------------------------------------------------------------
|  66.3% | 4,082.5 |    -- |    -- |USER
||----------------------------------------------------------------
||  11.8% |   729.2 |  48.8 |  6.4% |mult_su3_mat_vec_sum_4dir
||  10.2% |   629.4 |  49.6 |  7.4% |mult_adj_su3_mat_4vec
||   6.1% |   377.1 |  28.9 |  7.2% |mult_su3_nn
||   5.9% |   365.4 |  42.6 | 10.6% |mult_su3_na
||   5.4% |   329.4 |  37.6 | 10.4% |scalar_mult_add_lathwvec_proj
||   3.8% |   232.9 |  39.1 | 14.6% |mult_su3_sitelink_lathwvec
||================================================================
|  25.3% | 1,557.0 |    -- |    -- |MPI
||----------------------------------------------------------------
||  12.8% |   789.3 | 163.7 | 17.5% |MPI_Wait
||   6.7% |   411.9 |  74.1 | 15.5% |MPI_Isend
||   4.9% |   300.2 |  95.8 | 24.6% |MPI_Allreduce
||================================================================
|   5.9% |   365.4 |  44.6 | 11.1% |STRING
||---------------------------------------------------------------
||   5.9% |   365.4 |  44.6 | 11.1% |memcpy
|=================================================================
```

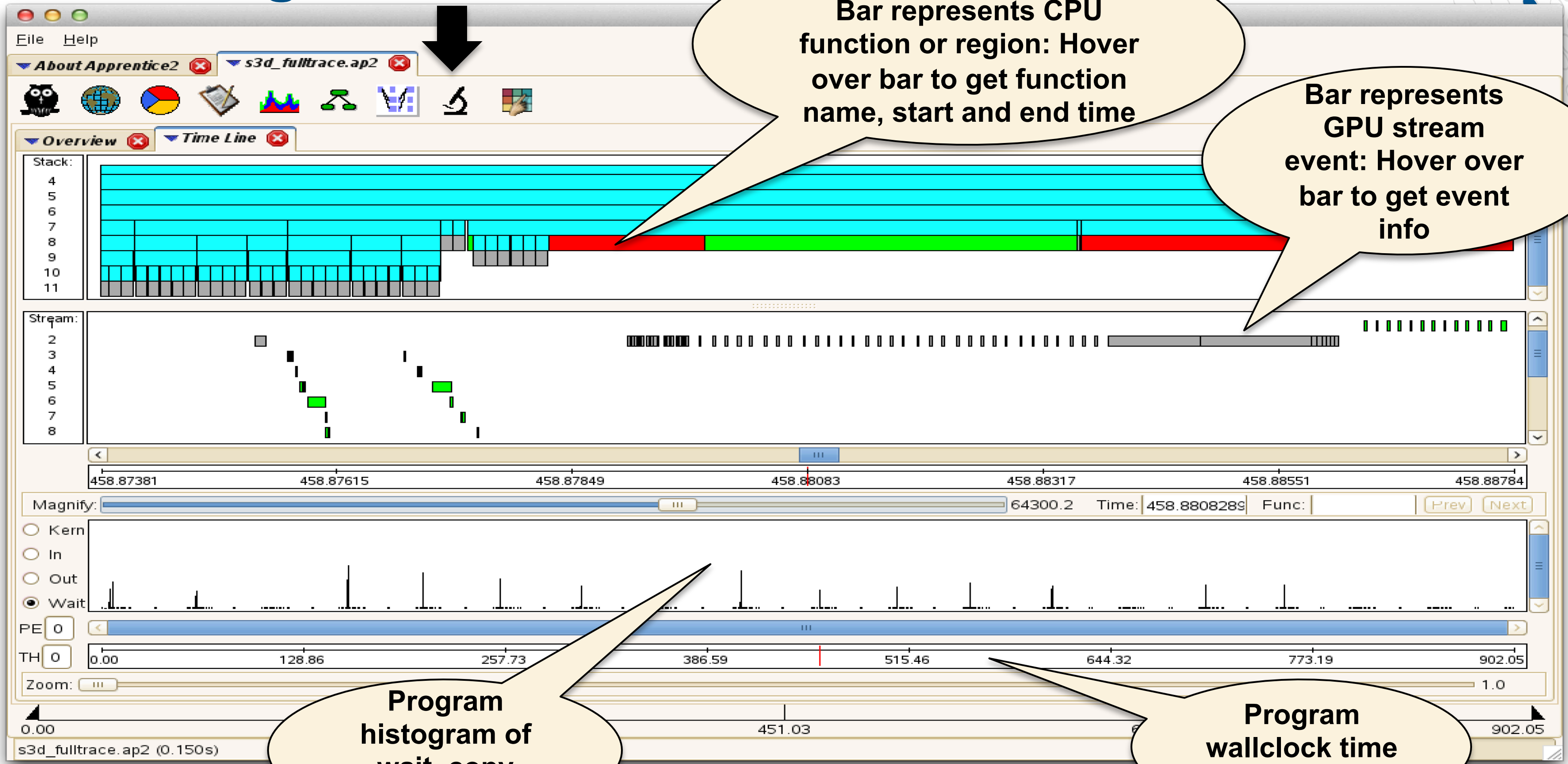
Guidance: How Can I Learn More?

```
MPI utilization:

    The time spent processing MPI communications is relatively high.
    Functions and callsites responsible for consuming the most time can
    be found in the table generated by pat_report -O callers+src (within
    the MPI group).
```

Guidance: Reduce Shared Resource Contention

```
Metric-Based Rank Order:

    When the use of a shared resource like memory bandwidth is unbalanced
    across nodes, total execution time may be reduced with a rank order
    that improves the balance.

    A file named MPICH_RANK_ORDER.USER_Time was generated
    along with this report and contains usage instructions and the
    custom rank order from the following table.

       Rank     Node Reduction    Maximum  Average
      Order   Metric     in Max     Value  Value
                Imb.      Value
    Current   15.46%             1.134e+03  9.588e+02
     Custom    1.46%    14.202%  9.731e+02  9.588e+02
```

GPU Program Timeline

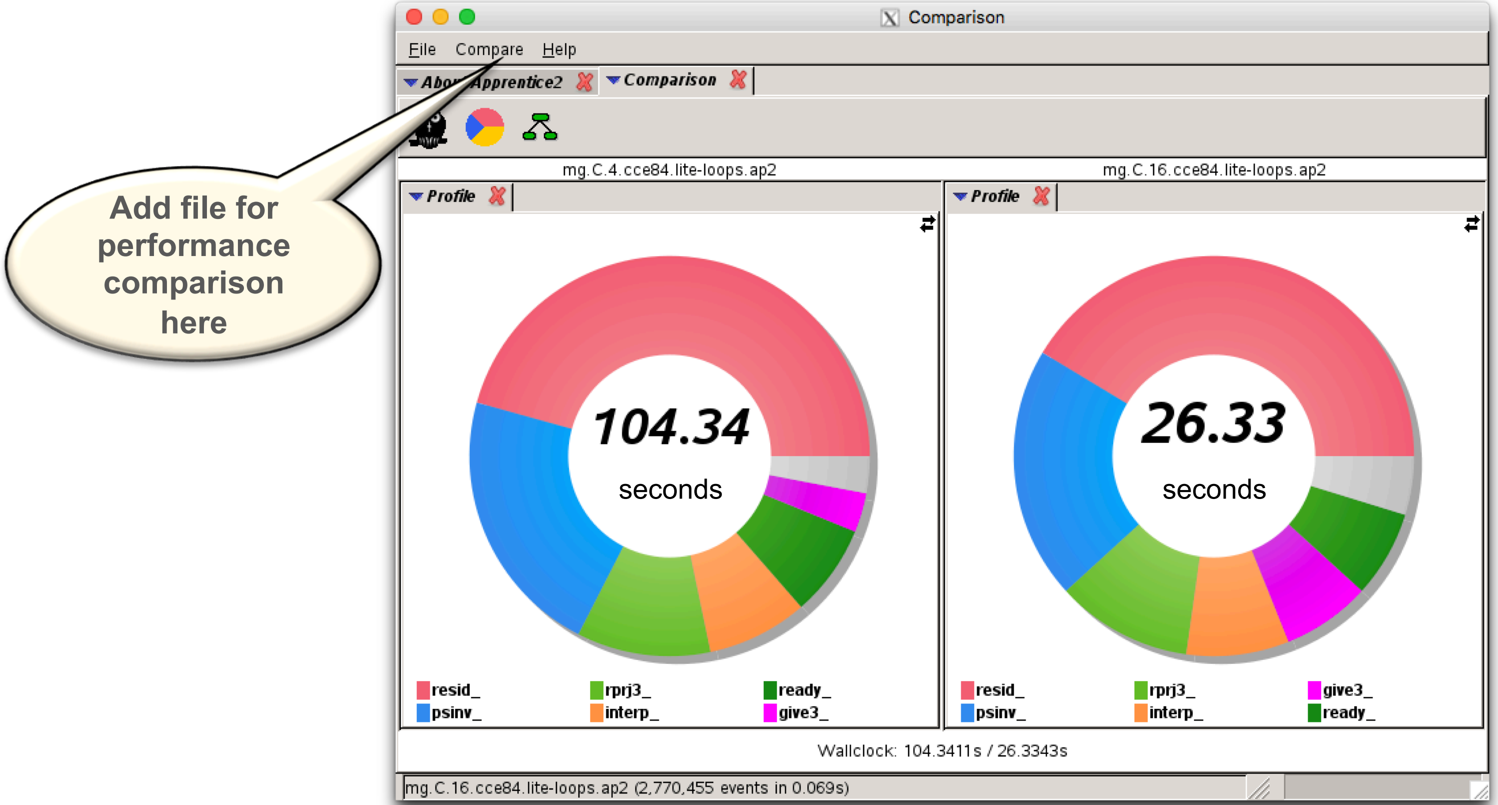


Recent Enhancements through perftools/6.3.2

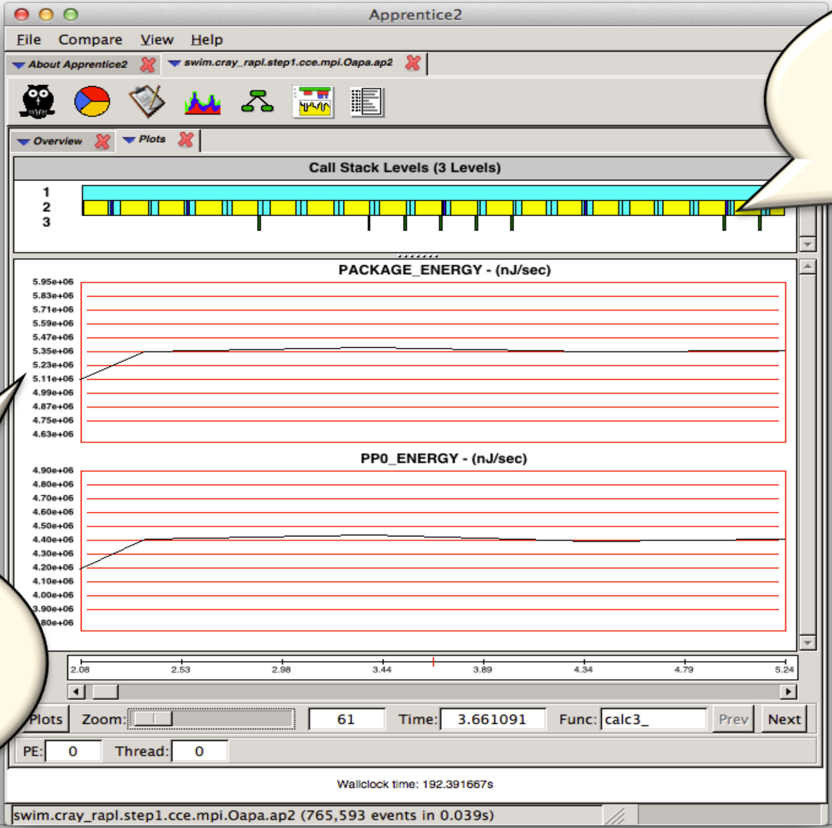
Highlights Since Last CUG

- **New perftools-base and instrumentation modules (*6.3.0*)**
- **Sampling over time and gnuplots (*6.2.3*)**
- **Apprentice2 sampling over time plots with call stack (*6.3.0*)**
- **Apprentice2 MPI communication pattern in summary mode (*6.3.0*)**
- **Observation for helper threads in reports (*6.3.0*)**
- **Performance data comparison in Apprentice2 (*6.3.1*)**

Evaluate Scaling with Cray Apprentice2



Energy Consumption Over Time (XC Systems)

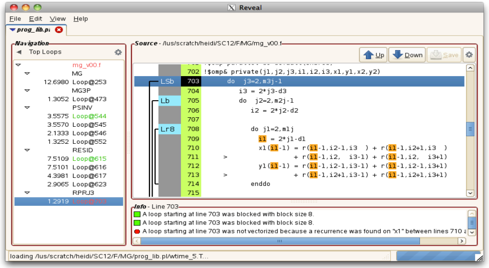


Call stack:
Bar represents function or region: Hover over bar to get function name, start and end time

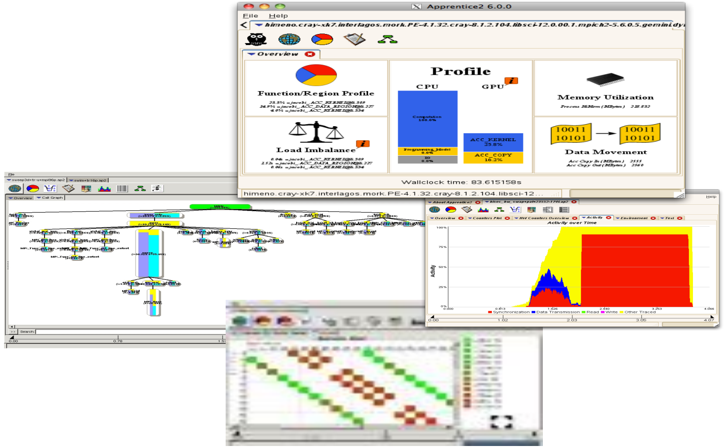
Plots of energy consumed by the socket and by the cores within a socket over time. Can also show memory high water mark, etc.

Support for Cray Systems with KNL

Port



Analyze



Overview of Support for KNL

- **CrayPat and CrayPat-lite**
  - Identifies top time consuming routines, work load imbalance, MPI rank placement strategies, etc.
  - Enhanced program memory high water mark
    - Broken down into DDR and MCDRAM memory
  - Report active allocations at samples or during tracing at the function level
- **PAPI**
- **Cray Apprentice2**
  - Helps identify load imbalance, excessive communication, network contention, excessive serialization
- **Reveal support for adding OpenMP and allocating in MCDRAM**

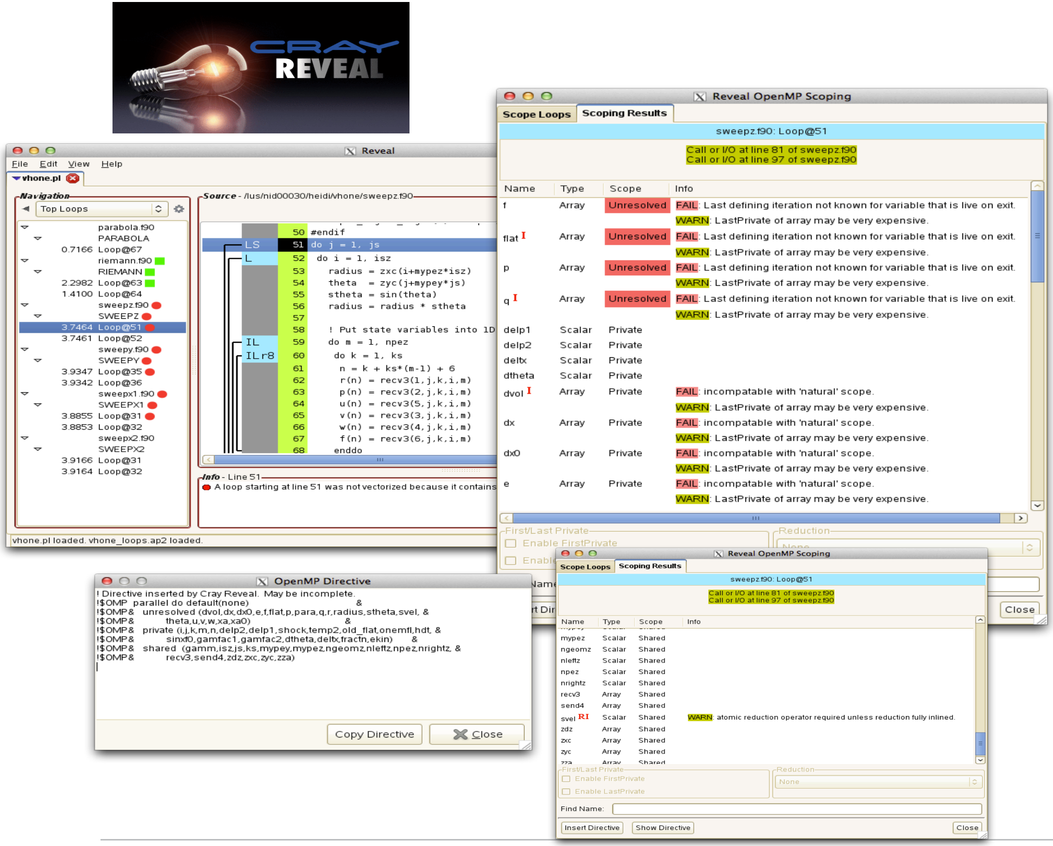
Functionality Coming in 2016

- **MCDRAM configuration information**
- **New trace groups for**
  - MemKind, HBW, CrayMem
  - OpenCL
  - Lustre API
  - Parallel NetCDF
- **Support for Charm++**

Example: MCDRAM Configuration Information

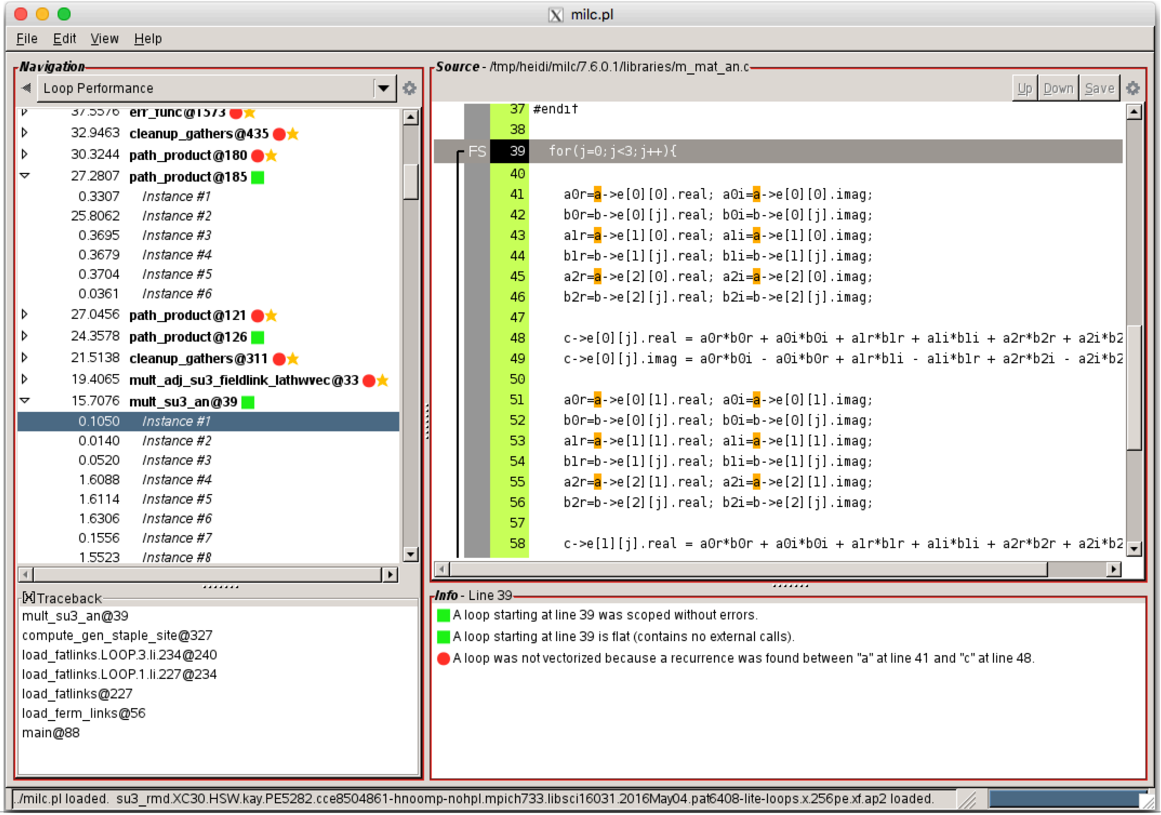
```
CrayPat/X:  Version 6.4.X Revision e82c848  04/29/16 14:13:55

Number of PEs (MPI ranks):   64

Numbers of PEs per Node:      1  PE on each of        64  Nodes

Numbers of Threads per PE:    1

...

Execution start time:  Mon May  2 15:54:21 2016

Intel knl CPU  ...

MCDRAM: 16 GiB available as snc2, cache (100% cache)  for 16 PEs
MCDRAM: 16 GiB available as snc2,  flat (  0% cache)  for 16 PEs
MCDRAM: 16 GiB available as snc4,  flat (  0% cache)  for 1 PE
MCDRAM: 16 GiB available as quad,  flat (  0% cache)  for 15 PEs
MCDRAM: 16 GiB available as quad, equal ( 50% cache)  for 16 PEs
```

Adding OpenMP with Reveal

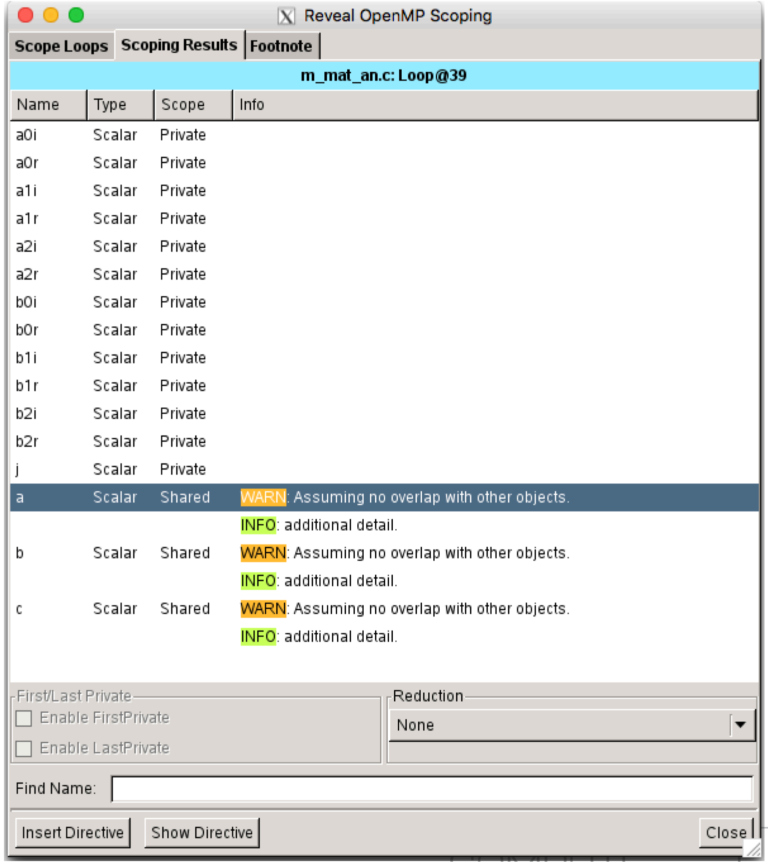


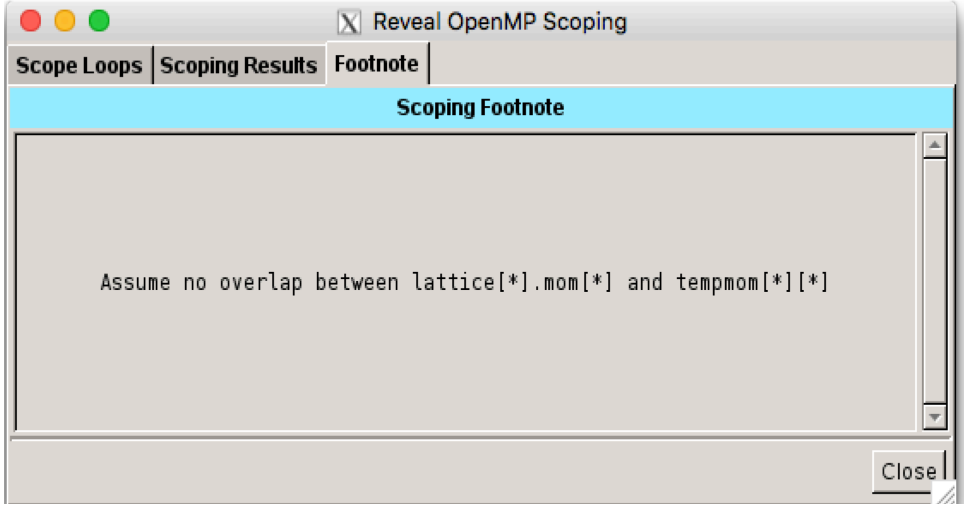
- Navigate to relevant loops to parallelize
- Identify parallelization and scoping issues
- Get feedback on issues down the call chain (shared reductions, etc.)
- Insert parallel directives into source for performance portable code
- Validate scoping correctness on existing directives

More Information for C/C++ Programs



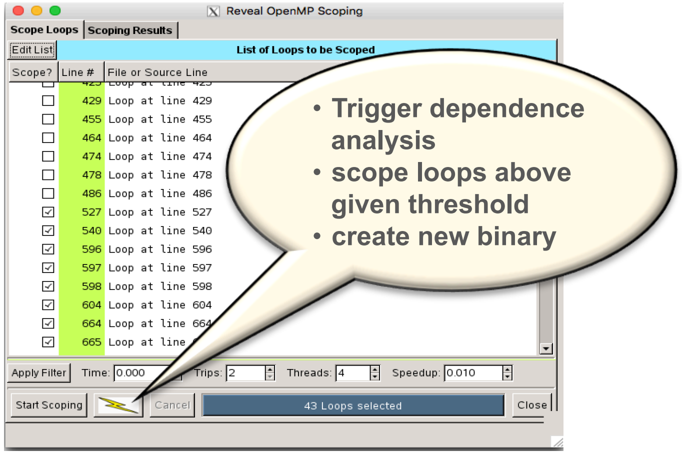
More Information for C/C++ Programs (2)





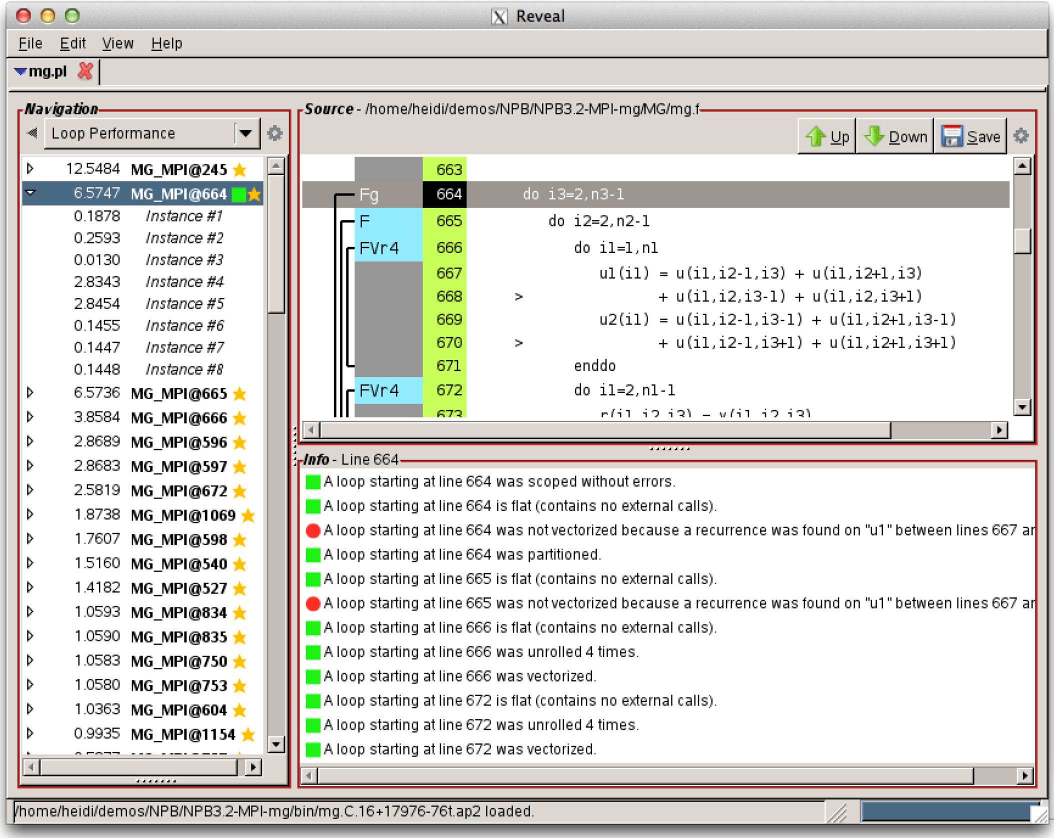
Reveal Auto-Parallelization

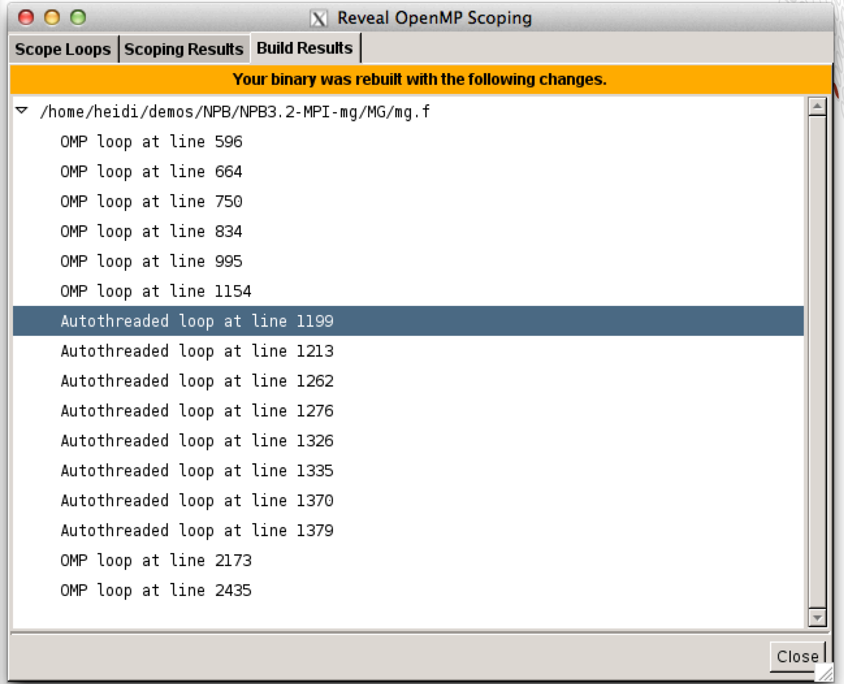
- Build an ***experimental binary*** **that includes automatic runtime-assisted parallelization**
- **No source code changes required to see if high level loops that contain calls can be automatically parallelized**
- **Result includes parallelization of serial loops via traditional OpenMP as well as more extensive loop optimizations**



- **User Workflow:**
  1. Obtain **loop work estimates** using CCE 8.5 and perftools-lite-loops from perftools/6.4.0
  2. Use Reveal and CCE's program library to **parallelize loops and create experimental binary**
  3. Run experimental binary and compare performance against baseline
  4. If auto-parallelization is successful, **use Reveal to insert parallel directives into source**

Examples of Reveal Analysis Feedback





Reveal Auto-Parallelization Recap

- **Minimal user time investment includes time to set up and run optimization experiment**
  - Collect loop work estimates
  - Build program library
  - Click button in Reveal
  - Run experimental binary and compare against original program
- **Even if experiment does not yield a performance improvement, Reveal will provide insight into parallelization issues**
- **Targeted for KNL, where a pure MPI solution cannot utilize all cores on a node**
- **Can be used on existing hardware (AMD Interlagos, Intel Haswell, etc.)**
- **Infrastructure will allow different optimization experiments in the future**

Coming Soon…

Identifying Objects for Allocation in KNL MCDRAM

FLOPS vs Data Movement / Data Locality

- **Next generation memory systems will be more complicated**
  - Multi-tier hierarchies
  - NUMA domains
  - Complicated caches

- **Data movement through the memory hierarchy is critical to performance**

- **Compared to the price of data movement, flops are “free”**

- **Monitoring and minimizing data movement within node and across nodes is key**

Identifying Objects for Allocation in MCDRAM

- Is application predominantly memory intensive?
- Does application data all fit within MCDRAM?
- What data contributes most to memory bandwidth?
- Where is data allocated within program?

Reveal Data Allocation Assistance

- Use combination of CrayPat and Reveal to identify data most relevant for allocation in MCDRAM

```
subroutine sweep (it,jt, kt, nm, isct, mm,mmo,mmi, mk, myid,
     1  hi, hj, hk, di, dj, dk, Phi, Phii,
     2  Src, Flux, Sigt,
     3  w,mu,eta,tsi, wmu,weta,wtsi, pn,
…
      do i = 1, it
         phi(i) = src(i,j,k,1)
      end do
      do n = 2, nm
         do i = 1, it
            phi(i) = phi(i) + pn(m,n,iq)*src(i,j,k,n)
         end do
      end do
```

```
c...AUTOMATIC ARRAYS
 double precision Src(it,jt,kt,nm)
 double precision pn(mm,nm,8)
 double precision Phi(it)
```

Example Workflow

- Enable data collection for memory analysis experiment and build program with CCE's program library feature
- Run program to collect data
- Pass program library and memory traffic data to Reveal
- Reveal shows best candidates for allocation in MCDRAM and identifies points of allocation
- Reveal helps insert allocation directives into source

Summary

- Users continue to need tools to help find critical performance bottlenecks within a program

- Cray performance tools offer functionality that reduces the time investment associated with porting and tuning applications on new and existing Cray systems

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, and URIKA. The following are trademarks of Cray Inc.: 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.