Memory Analysis Assistance

Heidi Poxon
Sr. Principal Engineer
Cray Programming Environment
Managing Multi-tiered Memory

- Future systems are likely to include a high performance tier (for bandwidth or latency) and a high capacity tier at a lower cost.

- Our goals to assist users with using more than one type of explicitly addressable on-node memory:
  - Provide easy-to-use interface for user to allocate data into HBM
  - Provide assistance with making best use of limited capacity
Identifying Arrays Is Difficult

subroutine ax_e()

do i=1,n
    wr = g(1,i)*ur(i) + g(2,i)*us(i) + g(3,i)*ut(i)
    ws = g(2,i)*ur(i) + g(4,i)*us(i) + g(5,i)*ut(i)
    wt = g(3,i)*ur(i) + g(5,i)*us(i) + g(6,i)*ut(i)
    ur(i) = wr
    us(i) = ws
    ut(i) = wt
endo

subroutine glsc3()

do i=1,n
    tmp = tmp + a(i)*b(i)*mult(i)
    continue
endo

Arrays a, b, and mult have a higher bandwidth sensitivity than array g
MCDRAM Usage Assistance

- Combination of CCE, CrayPat and Reveal are used to analyze application memory traffic
  - Available initially with
    - cce/8.5.6
    - perftools-base/6.4.4

- Analysis can be performed on Haswell/Broadwell or KNL processors

- Easy-to-use CrayPat-lite experiment to collect information on arrays that contribute most to memory bandwidth
Memory Traffic Analysis Overhead

- We have seen low data collection overhead
  - < 1% to a few %
  - Should be able to run large jobs
  - Functions that are entered and exited often and allocate arrays big enough to be tracked can increase runtime

- Most of the analysis is done post execution

- Lite mode post processing occurs on compute node
  - Can be slow on KNL because of KNL processor speed
  - User can defer pat_report to after job completes and run on login node
  - Scaling improvements for pat_report coming in 2017
Memory Analysis Tool Usage

Similar to other Lite mode experiments. . .

- Load needed modules
- Build application with CCE’s program library
- Run program to generate memory tracking data
- Analyze memory traffic and build memory directives using Reveal
Tips When Collecting Data

- Run similar problem size as you would in “production”
- Fill up nodes to maximize contention
- Run on multiple nodes
Example Usage on Haswell

- $ module swap PrgEnv-intel PrgEnv-cray
- $ module swap craype-ivybridge craype-haswell
- $ module load perftools-lite-hbm

- $ cc -h pl=/full_path/my_program.pl -dynamic my_program.c

- Run program (no batch script modifications required) to create .ap2 file

- $ reveal my_program.pl traffic_results.ap2
Viewing Ranked Arrays and Allocation Sites
Viewing Ranked Arrays and Allocation Sites

Inlining call tree
Matching Frees() to Mallocs()
Handling C++ new

- Reveal shows how STL objects rank with other arrays

- Reveal currently points to allocation site
  - Allocation directive must be at declaration site

- Users must find declaration site, modify declaration to point to an hbw-aware allocator

- We are working on the ability to identify declaration sites in Reveal
Building Allocation Directive
What the Tool Doesn’t Know

- No view into which arrays are alive when
  - You may be able to pack more arrays into MCDRAM than the experiment suggests because not all arrays may be active at the same time
  - User can review source code at memory activity hot spots to better understand which arrays are concurrently active
  - Use preferred clause on directive to prevent allocation failure
Tip

- Ranked list of arrays from CrayPat / Reveal may be better than using `numactl` if user wants application data allocated into HBM
  - User only gets objects that are allocated within their program (no objects from libraries like MPI are included)
CrayPAT MCDRAM Configuration Information

CrayPat/X:  Version 6.4.2.36 Revision 8374f24  08/08/16  14:59:22
Experiment:  lite  lite/sample_profile
Number of PEs (MPI ranks):  2,048
Numbers of PEs per Node:  32  PEs on each of  64  Nodes
Numbers of Threads per PE:  1
Number of Cores per Socket:  512  PEs on sockets with  34  Cores
                              1,536  PEs on sockets with  68  Cores
...

MCDRAM:  7.2 GHz, 16 GiB available as snc2, cache (100% cache)  for 512 PEs
MCDRAM:  7.2 GHz, 16 GiB available as quad, cache (100% cache)  for 1536 PEs

Avg Process Time:  3,251 secs
High Memory:  9,651,837.1 MBytes  4,712.8 MBytes per PE
I/O Read Rate:  23.645208 MBytes/sec
I/O Write Rate:  6.836539 MBytes/sec
Avg CPU Energy:  43,899,476 joules  685,929 joules per node
Avg CPU Power:  13,504 watts  211.00 watts per node
Per-numanode Memory High Water Mark

● **Available with Lite and full modes**
  ● $ module load perftools-lite  
  ● $ module load perftools

● **3 examples:**
  ● 1) $ aprun -m1800m -n 2048 -N 16 -d 1 ./cpmd.x
  ● 2) $ aprun -m1800m -n 2048 -N 16 -d 1 \ 
    numactl --preferred=1 ./cpmd.x
  ● 3) $ aprun -m1800m -n 2048 -N 64 -d 1 \ 
    numactl --preferred=1 ./cpmd.x
1) Flat Mode, No Allocation

MCDRAM: 7.2 GHz, 16 GiB available as quad, flat (0% cache)

<table>
<thead>
<tr>
<th>Process</th>
<th>HiMem</th>
<th>Numanode</th>
</tr>
</thead>
<tbody>
<tr>
<td>HiMem</td>
<td>Numa</td>
<td>PE=ALL</td>
</tr>
<tr>
<td>(MBytes)</td>
<td>Node 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(MBytes)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>827.3</td>
<td>827.3</td>
<td>Total</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>----------</td>
</tr>
<tr>
<td>827.3</td>
<td>827.3</td>
<td>numanode.0</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>870.5</td>
<td>870.5</td>
<td>pe.1904</td>
</tr>
<tr>
<td>870.0</td>
<td>870.0</td>
<td>pe.2000</td>
</tr>
<tr>
<td>869.8</td>
<td>869.8</td>
<td>pe.1776</td>
</tr>
<tr>
<td>868.6</td>
<td>868.6</td>
<td>pe.1008</td>
</tr>
<tr>
<td>868.1</td>
<td>868.1</td>
<td>pe.1264</td>
</tr>
</tbody>
</table>

16 ranks per node

No allocation to MCDRAM, only to DRAM so one Numanode is used
2) Flat Mode, Full Allocation

MCDRAM: 7.2 GHz, 16 GiB available as quad, flat (0% cache)

<table>
<thead>
<tr>
<th>Process</th>
<th>HiMem</th>
<th>HiMem</th>
<th>Numanode</th>
</tr>
</thead>
<tbody>
<tr>
<td>HiMem</td>
<td>Numa</td>
<td>Numa</td>
<td>PE=ALL</td>
</tr>
<tr>
<td>(MBytes)</td>
<td>Node 0</td>
<td>Node 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(MBytes)</td>
<td>(MBytes)</td>
<td></td>
</tr>
<tr>
<td>827.7</td>
<td>0.0</td>
<td>827.7</td>
<td>Total</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>827.7</td>
<td>0.0</td>
<td>827.7</td>
<td>numanode.0</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>970.7</td>
<td>0.0</td>
<td>970.6</td>
<td>pe.18</td>
</tr>
<tr>
<td>970.4</td>
<td>0.0</td>
<td>970.4</td>
<td>pe.60</td>
</tr>
<tr>
<td>970.3</td>
<td>0.0</td>
<td>970.3</td>
<td>pe.242</td>
</tr>
<tr>
<td>970.2</td>
<td>0.0</td>
<td>970.2</td>
<td>pe.351</td>
</tr>
<tr>
<td>969.8</td>
<td>0.0</td>
<td>969.8</td>
<td>pe.562</td>
</tr>
<tr>
<td>870.1</td>
<td>0.0</td>
<td>870.1</td>
<td>pe.1904</td>
</tr>
</tbody>
</table>

16 ranks per node

With numactl --preferred=1, all allocations end up on MCDRAM Numanode 1. No allocations end up on DRAM Numanode 0.
3) Flat Mode, Partial Allocation

MCDRAM: 7.2 GHz, 16 GiB available as quad, flat (0% cache)

<table>
<thead>
<tr>
<th>Process</th>
<th>HiMem</th>
<th>HiMem</th>
<th>Numanode</th>
</tr>
</thead>
<tbody>
<tr>
<td>HiMem</td>
<td>Numa</td>
<td>Numa</td>
<td>PE=ALL</td>
</tr>
<tr>
<td>(MBytes)</td>
<td>Node 0</td>
<td>Node 1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(MBytes)</th>
<th>(MBytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>786.9</td>
<td>534.5</td>
</tr>
<tr>
<td>252.4</td>
<td>Total</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>786.9</td>
<td>534.5</td>
</tr>
<tr>
<td>252.4</td>
<td>numanode.0</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>794.3</td>
<td>538.9</td>
</tr>
<tr>
<td>255.4</td>
<td>pe.0</td>
</tr>
<tr>
<td>791.3</td>
<td>537.6</td>
</tr>
<tr>
<td>253.8</td>
<td>pe.64</td>
</tr>
<tr>
<td>791.2</td>
<td>537.3</td>
</tr>
<tr>
<td>253.9</td>
<td>pe.128</td>
</tr>
<tr>
<td>791.1</td>
<td>537.3</td>
</tr>
<tr>
<td>253.8</td>
<td>pe.192</td>
</tr>
<tr>
<td>791.0</td>
<td>537.1</td>
</tr>
<tr>
<td>253.9</td>
<td>pe.256</td>
</tr>
<tr>
<td>790.7</td>
<td>537.4</td>
</tr>
<tr>
<td>253.4</td>
<td>pe.136</td>
</tr>
</tbody>
</table>

Using 64 cores on a node creates less available space per MPI rank.

With numactl --preferred=1, after MCDRAM Numanode 1 is full, data is allocated to DRAM Numanode 0.
MCDRAM Allocation Assistance Recap

- Cray Tools track requests to memory and evaluate the bandwidth contribution of objects within a program

- Helpful for memory-intensive programs that cannot fit within MCDRAM

- Reduces time investment associated with selectively allocating data into KNL’s MCDRAM

- The result is performance portable code
  - CCE memory allocation directives are ignored on X86 processors
Thank You!
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 publicly 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 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.

Copyright 2017 Cray Inc.