An Update on the Cray Tools Activities for Extreme Scale Computing

Luiz DeRose and Heidi Poxon
Cray Inc.
Moving from X86 to Hybrid Multi-core Systems

- Running MPI only on a node will not work well
  - Too much memory used, even if on-node shared communication is available
  - As the number of MPI ranks increases, more off-node communication can result, creating a network injection issue

- Focus on where MPI starts leveling off

- Address by adding additional levels of parallelism, reducing MPI ranks per node
  - MPI -> MPI + OpenMP
  - MPI + OpenMP -> MPI + OpenMP GPU extensions
Steps to Porting to Hybrid Multi-core Systems

- Maximize on-node communication if MPI point-to-point communication is dominant in the program
  - Auto-grid detection and placement suggestions

- Determine where to add additional levels of parallelism
  - Find top time consuming loops with enough work for GPU
    - Loop statistics

- Do parallel analysis and restructuring on targeted high level loops
  - Scoping assistance
Steps to Porting to Hybrid Multi-core Systems (2)

- Add parallel directives and acceleration extensions
  - OpenMP extensions

- Run on X86 + GPU and get performance feedback
- Optimize for data locality and copies to the GPU
- Optimize kernel on GPU
  - Cray performance tools statistics
Automatic Communication Grid Detection

- Analyze runtime performance data to identify grids in a program to maximize on-node communication
  - Example: nearest neighbor exchange in 2 dimensions
    - Sweep3d uses a 2-D grid for communication

- Determine whether or not a custom MPI rank order will produce a significant performance benefit

- Grid detection is helpful for programs with significant point-to-point communication

- Produce a custom rank order if it’s beneficial based on grid size, grid order and cost metric
Example summary for sweep3d (pat_report table Notes)

This application appears to use point-to-point MPI communication at least partly organized into a 8 X 6 grid pattern. Time spent in MPI routines accounted for over 63.1% of the execution time. A portion of this time could potentially be saved by utilizing a rank order that maximizes the fraction of communication that is between ranks on the same node. The following table estimates this fraction for several rank orders.

An MPICH_RANK_ORDER file was generated along with this report and contains the Custom rank order from the following table. This file also contains usage instructions and a table of alternative rank orders.
Table 4: Sent Message Stats for Selected MPI Rank Orders

<table>
<thead>
<tr>
<th>Rank Order</th>
<th>On-Node</th>
<th>On-Node %</th>
<th>Options for grid_order utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom</td>
<td>1.30e+07</td>
<td>50.00%</td>
<td>-R -P -m 48 -n 4 -g 8,6 -c 2,1</td>
</tr>
<tr>
<td>SMP</td>
<td>8.10e+06</td>
<td>31.25%</td>
<td></td>
</tr>
<tr>
<td>Fold</td>
<td>6.75e+05</td>
<td>2.60%</td>
<td></td>
</tr>
<tr>
<td>RoundRobin</td>
<td>0.00e+00</td>
<td>0.00%</td>
<td></td>
</tr>
</tbody>
</table>

CScADS, Aug 2011

Cray Inc.
# The 'Custom' rank order in this file targets nodes with multi-core
# processors, based on Sent Msg Total Bytes collected for:
#
# Program: /lus/nid00030/heidi/sweep3d/mod/sweep3d.mpi
# Ap2 File: sweep3d.mpi+pat+27054-89t.ap2
# Number PEs: 48
# Max PEs/Node: 4
#
# To use this file, set the environment variable
# MPICH_RANK_REORDER_METHOD to 3 prior to executing the program.
#
# The following table lists rank order alternatives and the grid_order
# command-line options that can be used to generate a new order.
...

CScADS, Aug 2011
Cray Inc.
Loop Statistics

- Helps identify loops to move to GPU:
  - Loop timings approximate how much work exists within a loop
  - Trip counts can be used to help carve up loop on GPU

- Enabled with CCE –h profile_generate option

- Loop statistics reported by default in pat_report table
Notes for table 2:
Table option:
-0 loops

... 
The Function value for each data item is the avg of the PE values.
(To specify different aggregations, see: pat_help report options s1)

This table shows only lines with Loop Incl Time / Total > 0.0095.
(To set thresholds to zero, specify: -T)

Loop data version: L.12.2:B.3.1

Loop instrumentation can interfere with optimizations, so time reported here may not reflect time in a fully optimized program.

Loop stats can safely be used in the compiler directives:
!PGO$ loop_info est_trips(Avg) min_trips(Min) max_trips(Max)
#pragma pgo loop_info est_trips(Avg) min_trips(Min) max_trips(Max)

Explanation of Loop Notes (P=1 is highest priority, P=0 is lowest):
novec (P=0.5): Loop not vectorized (see compiler messages for reason).
sunwind (P=1): Loop could be vectorized and unwound.
vector (P=0.1): Already a vector loop.
Table 2: Loop Stats from -hprofile_generate

<table>
<thead>
<tr>
<th>Loop</th>
<th>Loop Incl</th>
<th>Loop Incl</th>
<th>Loop</th>
<th>Loop</th>
<th>Loop</th>
<th>Function=/&gt;.LOOP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incl</td>
<td>Time</td>
<td>Time /</td>
<td>Hit</td>
<td>Trips</td>
<td>Notes</td>
<td>PE='HIDE'</td>
</tr>
<tr>
<td>Time /</td>
<td></td>
<td>Hit</td>
<td></td>
<td>Avg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| | 24.6% | 0.057045 | 0.000570 | 100 | 64.1 | novec |calc2_.LOOP.0.li.614 |
| | 24.0% | 0.055725 | 0.000009 | 6413 | 512.0 | vector |calc2_.LOOP.1.li.615 |
| | 18.9% | 0.043875 | 0.000439 | 100 | 64.1 | novec |calc1_.LOOP.0.li.442 |
| | 18.3% | 0.042549 | 0.000007 | 6413 | 512.0 | vector |calc1_.LOOP.1.li.443 |
| | 17.1% | 0.039822 | 0.000406 | 98 | 64.1 | novec |calc3_.LOOP.0.li.787 |
| | 16.7% | 0.038883 | 0.000006 | 6284 | 512.0 | vector |calc3_.LOOP.1.li.788 |
| | 9.7% | 0.022493 | 0.000230 | 98 | 512.0 | vector |calc3_.LOOP.2.li.805 |
| | 4.2% | 0.009837 | 0.000098 | 100 | 512.0 | vector |calc2_.LOOP.2.li.640 |

CScADS, Aug 2011

Cray Inc.
Source Code – Loopmark

```
DO 200 I=1,M
  DO 200 J=js,je
    UNEW(I+1,J) = UOLD(I+1,J) +
    TDTS8*(Z(I+1,J+1)+Z(I+1,J))*(CV(I+1,J+1)+CV
    +CV(I+1,J)) - TDTSDX*(H(I+1,J)-H(I,J))
    if(j.gt.1)then
      VNEW(I,J) = VOLD(I,J) - TDTS8*(Z(I+1,J)+Z(I,J))
      *(CU(I+1,J)+CU(I,J)+CU(I,J-1)+CU(I+1,J-1))
      - TDTSDY*(H(I,J)-H(I,J-1))
      endif
    if(j.eq.n)then
      VNEW(I,J+1) = VOLD(I,J+1) - TDTS8*(Z(I+1,J+1)+Z(I,J))
      *(CU(I+1,J+1)+CU(I,J+1)+CU(I,J)+CU(I+1,J))
      - TDTSDY*(H(I,J+1)-H(I,J))
      endif
  PNEW(I,J) = POLD(I,J) - TDTSDX*(CU(I+1,J)-CU(I,J))
    1 - TDTSDY*(CV(I,J+1)-CV(I,J))
  200 CONTINUE
```

Info
Line 66:
Loop unrolled 2 times.
Loop interchanged with loop at line 67.
Display Scoping Information for Selected Loop

```plaintext
1064    do k=0, lz-1
1065    do j=0, local_ly-1
1066    do i=0, local_lx-1
1067        if (cell(i,j,k)==4) then
1068            rho_tmp = 0.000
1069            rho_rtmp = 0.000
1070            rho_btmp = 0.000
1071            ux_tmp = 0.000
1072            uy_tmp = 0.000
1073            uz_tmp = 0.000
1074            rho_rtmp = R(i,j,k, 0)+ R(i,j,k, 1)+ R(i,j,k, 2)&
1075                  + R(i,j,k, 3)+ R(i,j,k, 4)+ R(i,j,k, 5)&
1076                  + R(i,j,k, 6)+ R(i,j,k, 7)+ R(i,j,k, 8)&
1077                  + R(i,j,k, 9)+ R(i,j,k,10)+ R(i,j,k,11)&
1078                  + R(i,j,k,12)+ R(i,j,k,13)+ R(i,j,k,14)
1079            rho_btmp = B(i,j,k,0)+ B(i,j,k,1)+ B(i,j,k,2)&
1080                  + B(i,j,k,3)+ B(i,j,k,4)+ B(i,j,k,5)&
1081                  + B(i,j,k,6)+ B(i,j,k,7)+ B(i,j,k,8)&
1082                  + B(i,j,k,9)+ B(i,j,k,10)+B(i,j,k,11)&
1083                  +B(i,j,k,12)+B(i,j,k,13)+B(i,j,k,14)
1084            ux_tmp = R(i,j,k, 1)+ B(i,j,k,1)&
1085                  - R(i,j,k, 4)+ B(i,j,k,4)&
1086                  + R(i,j,k, 7)+ B(i,j,k,7)&
1087                  - R(i,j,k, 9)+ B(i,j,k,9)&
1088                  + R(i,j,k,10)+B(i,j,k,10)&
1089                  + R(i,j,k,12)+ B(i,j,k,13)+ B(i,j,k,14)
```

### Display Scoping Information for Selected Loop (2)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Scope</th>
<th>F</th>
<th>L</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Scalar</td>
<td>Shared</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cell</td>
<td>Scalar</td>
<td>Shared</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>local_lx</td>
<td>Scalar</td>
<td>Shared</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>local_ly</td>
<td>Scalar</td>
<td>Shared</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lz</td>
<td>Scalar</td>
<td>Shared</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>Scalar</td>
<td>Shared</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rho</td>
<td>Scalar</td>
<td>Shared</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rho_btmp</td>
<td>Scalar</td>
<td>Private</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>rho_rtmp</td>
<td>Scalar</td>
<td>Private</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>rho_tmp</td>
<td>Scalar</td>
<td>Private</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>ux_tmp</td>
<td>Scalar</td>
<td>Private</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>uxyz</td>
<td>Scalar</td>
<td>Shared</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uz_tmp</td>
<td>Scalar</td>
<td>Private</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>
Table 1: Profile by Function Group and Function

<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100.0%</td>
<td>18.113521</td>
<td>--</td>
<td>--</td>
<td>6.0</td>
<td>Total</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>
<tr>
<td>100.0%</td>
<td>18.113443</td>
<td>--</td>
<td>--</td>
<td>5.0</td>
<td>USER</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>
<tr>
<td>90.6%</td>
<td>18.113000</td>
<td>0.000000</td>
<td>0.0%</td>
<td>1.0</td>
<td>acc_sample_.ACC_DATA_REGION@li.23</td>
</tr>
<tr>
<td>9.4%</td>
<td>0.000443</td>
<td>0.000000</td>
<td>0.0%</td>
<td>1.0</td>
<td>acc_sample_.ACC_REGION@li.24</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>
<tr>
<td>0.0%</td>
<td>0.000078</td>
<td>0.000000</td>
<td>0.0%</td>
<td>1.0</td>
<td>ETC</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>
<tr>
<td>0.0%</td>
<td>0.000078</td>
<td>0.000000</td>
<td>0.0%</td>
<td>1.0</td>
<td>exit</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>

CScADS, Aug 2011
### Example Performance Statistics

#### Table 2: Time and Bytes Transferred for Accelerator Regions

<table>
<thead>
<tr>
<th>Host</th>
<th>Host</th>
<th>Acc</th>
<th>Acc Copy</th>
<th>Acc Copy</th>
<th>Calls</th>
<th>Calltree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time%</td>
<td>Time</td>
<td>Time</td>
<td>In</td>
<td>Out</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(MBytes)</td>
<td>(MBytes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100.0%</td>
<td>18.113</td>
<td>18.112</td>
<td>209.808</td>
<td>209.808</td>
<td>4</td>
<td>Total</td>
</tr>
<tr>
<td>100.0%</td>
<td>18.113</td>
<td>18.112</td>
<td>209.808</td>
<td>209.808</td>
<td>4</td>
<td>acc_sample_</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>acc_sample_.ACC_DATA_REGION@li.23</td>
</tr>
<tr>
<td>3</td>
<td>90.6%</td>
<td>16.418</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>9.4%</td>
<td>1.695</td>
<td>1.695</td>
<td>209.808</td>
<td>209.808</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0.0%</td>
<td>0.000</td>
<td>16.418</td>
<td>0.000</td>
<td>0.000</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>async_kernel</td>
</tr>
</tbody>
</table>
Systems with hundreds of thousands of threads of execution need a new debugging paradigm

- Innovative techniques for productivity and scalability
  - Scalable Solutions based on MRNet from University of Wisconsin
    - STAT - Stack Trace Analysis Tool
      - Scalable generation of a single, merged, stack backtrace tree
        - running at 216K back-end processes
    - ATP - Abnormal Termination Processing
      - Scalable analysis of a sick application, delivering a STAT tree and a minimal, comprehensive, core file set.

- Comparative debugging
  - A data-centric paradigm instead of the traditional control-centric paradigm
  - Collaboration with Monash University and University of Wisconsin for scalability

- Fast Track Debugging
  - Debugging optimized applications
  - Added to Allinea's DDT 2.6 (June 2010)