Communications Issues Running the Non-hydrostatic Icosahedral Model (NIM) on GPUs

September 19, 2013
Jacques Middlecoff

2012 NCAR workshop on heterogeneous multi-core platforms
Topics

- Zero copy using mapped memory
- Run setup
- Platforms
- NIM dynamics scaling
- Communication time breakdown
- Paths to interconnect
- HyperQ
- GPUDirect
Zero Copy Using Mapped Memory

• Mapped memory
  – Host pinned memory that is mapped to device memory
  – Eliminates the need to copy between host and device
  – Limited amount of mapped memory

• We used mapped memory for the MPI send and receive buffers
Zero Copy Using Mapped Memory

- Mapped memory helped on both titan and jet
- On titan for NIM it’s best to define the buffer memory as Mapped-WriteCombined
  - Pack and unpack is slower than regular GPU memory
  - No copy savings is more that the pack-unpack slowdown
- Jet (Fermi GPU C2090) is different for the send buffer
  - Mapped-WriteCombined greatly slows down the pack
  - Mapped is best
    - Still do a copy
    - The copy is faster with mapped memory
Run Setup

• All runs were made using NIM
  – 14 KM resolution (2,621,442 columns)
  – 96 vertical levels per column
  – 100 time steps
  – All physics runs used GFS physics
  – Timings did not include input or output
Platforms

• All runs were made at ORNL and TACC
  – ORNL titan
    • CPU: AMD Opteron 6774 (AMD)
    • GPU: Nvidia Tesla K20X (K20)
  – TACC stampede
    • CPU: Intel Xeon E5-2680 Sandy Bridge (SB)

• All comparisons were socket to socket
  – 1 GPU socket = 1 GPU
  – 1 AMD CPU socket = 16 CPU cores
  – 1 SB CPU socket = 8 CPU cores
## NIM GPU vs CPU Communication Time by Function

<table>
<thead>
<tr>
<th>Function</th>
<th>320 K20 GPUs</th>
<th>320 AMD CPU cores</th>
<th>320 AMD CPU sockets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns/core</td>
<td>8192</td>
<td>8192</td>
<td>512</td>
</tr>
<tr>
<td>Initialization</td>
<td>.5</td>
<td>.1</td>
<td>.1</td>
</tr>
<tr>
<td>Pack</td>
<td>2.4</td>
<td>5.1</td>
<td>1.1</td>
</tr>
<tr>
<td>Barrier</td>
<td>.4</td>
<td>.8</td>
<td>.8</td>
</tr>
<tr>
<td>MPI_Waitall</td>
<td>1.6</td>
<td>6.2</td>
<td>2.8</td>
</tr>
<tr>
<td>Unpack</td>
<td>2.4</td>
<td>3.6</td>
<td>.7</td>
</tr>
<tr>
<td>Exchange</td>
<td>6.8</td>
<td>15.8</td>
<td>5.7</td>
</tr>
</tbody>
</table>
Paths to interconnect from socket

To interconnect

K20 socket (1 GPU)

AMD socket (16 cores)
Hyper-Q

- Provides 32 work queues between CPU and GPU
- Each work queue can be assigned to a different MPI process
- “Maximizes GPU utilization and increases overall GPU performance”
NIM with GFS physics: Hyper-Q scaling on 80 GPUs

CPU time on 80 sockets = 302 seconds

<table>
<thead>
<tr>
<th>Work Queues</th>
<th>MPI Tasks</th>
<th>Cols/Task</th>
<th>Runtime (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
<td>32768</td>
<td>1270</td>
</tr>
<tr>
<td>2</td>
<td>160</td>
<td>16384</td>
<td>814</td>
</tr>
<tr>
<td>4</td>
<td>320</td>
<td>8192</td>
<td>445</td>
</tr>
<tr>
<td>8</td>
<td>640</td>
<td>4096</td>
<td>285</td>
</tr>
<tr>
<td>16</td>
<td>1280</td>
<td>2048</td>
<td>192</td>
</tr>
</tbody>
</table>
NIM with GFS physics: AMD CPU vs K20 GPU

Physics runs on the CPUs via 16 Hyper-Q work queues each an MPI process

<table>
<thead>
<tr>
<th>Sockets</th>
<th>Cols/core</th>
<th>AMD Runtime</th>
<th>GPU Cols/queue</th>
<th>GPU Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>2048</td>
<td>302</td>
<td>2048</td>
<td>192</td>
</tr>
<tr>
<td>160</td>
<td>1024</td>
<td>153</td>
<td>1024</td>
<td>129</td>
</tr>
<tr>
<td>320</td>
<td>512</td>
<td>77</td>
<td>512</td>
<td>89</td>
</tr>
<tr>
<td>640</td>
<td>256</td>
<td>39</td>
<td>256</td>
<td>74</td>
</tr>
<tr>
<td>1280</td>
<td>128</td>
<td>21</td>
<td>128</td>
<td>69</td>
</tr>
</tbody>
</table>
NIM dynamics with and without Hyper-Q using 16 work queues

Run time (sec)

Sockets utilized
### Hyper-Q

NIM dynamics on 80 nodes
Hyper-Q with 16 MPI-processes per GPU
With and without the exchange time included

<table>
<thead>
<tr>
<th>Function</th>
<th>Points/MPI-Process</th>
<th>Including Exchange</th>
<th>No Exchange Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>2048</td>
<td>203</td>
<td>193</td>
</tr>
<tr>
<td>GPU</td>
<td>32768</td>
<td>72</td>
<td>60</td>
</tr>
<tr>
<td>GPU Hyper-Q</td>
<td>2048</td>
<td>100</td>
<td>48</td>
</tr>
</tbody>
</table>
GPUDirect

- GPUDirect plus GPU-aware MPI provides direct MPI connection from GPU to GPU
- Jeff Larkin: “GPUDirect works on titan”
  - Not yet fully optimized
    - Still has copies between GPU and CPU
  - Can be a savings when combined with pipelining
    - Forward-looking optimization (XC30)
- Requires a port to the Cray compiler which we are working on.
QUESTIONS?