Performance of 3-D FFT Using Multiple GPUs with CUDA 4

Akira Nukada
Tokyo Institute of Technology
GPGPU

- High performance, high memory bandwidth
- High efficiencies in power, space, price
- Complicated programming
- Capacity of device memory
- Requires parallelism
- Need optimizations in algorithm, data placement, ….
Next stage of GPGPU

We need multiple GPUs for

• More computing power
• More device memory

There is no SLI for GPGPU, so we have to manage data distribution, load-balancing, and synchronizations between GPUs.
Multi-GPU programming

• Similar to MPI applications (distributed memory)
  – Easy for developers with MPI experience
  – Not easy to achieve expected performance
CUDA 4.0

- New features for multi-GPU applications
  - Direct P2P via PCI-E network
  - NIC interoperability with InfiniBand HCAs

These improve the data transfer speed between GPUs
Intra-node

CUDA 3.X

- DDR3 RAM
- DDR3 RAM
- DDR3 RAM

CPU 0
Westmere

QPI

IO-Hub
Tylersburg

GPU 0
Fermi

PCI-E x16

CUDA 4.X

- DDR3 RAM
- DDR3 RAM
- DDR3 RAM

CPU 0
Westmere

QPI

IO-Hub
Tylersburg

GPU 0
Fermi

PCI-E x16

GPU 1
Fermi

PCI-E x16
Inter-node: CUDA 3.X

Memory copy on host

CUDA pinned ➔ IB pinned ➔ QDR x4 IB ➔ GPU 0 (Fermi)

CUDA pinned ➔ QDR x4 IB ➔ GPU 0 (Fermi)

Disable DMA access by IB

CUDA pinned ➔ IB pinned ➔ QDR x4 IB ➔ GPU 0 (Fermi)
Inter-node: CUDA 4.X

CPU 0

CUDA pinned

IB pinned

GPU 0

Fermi

QDR x4 IB

Allows DMA access by IB
3-D FFT

Performs 1-D FFT for each dimension of 3-D data.

Memory-intensive computation
• \( O(N \log N) \) ops for \( O(N) \) data

Several FFT implementations for CUDA
• NVIDIA CUFFT library
• Our NukadaFFT library
• others...
3-D FFT on distributed memory

1-D FFTs for dim. X & Y on each device

all-to-all comm.

1-D FFTs for dim. Z on each device
All-to-all comm. in FFT

Parallel FFT typically requires 1~3 all-to-all comm. The number depends on the data distribution before and after the transform.

\[(NX, NY, NZ/P) \Leftrightarrow (NX, NY/P, NZ)\]

before \hspace{2cm} after

In such a case, only 1 all-to-all comm. is required.

Many applications allow this:

• 3D-RISM, Molecular simulation. (Motivation of this work)
• Convolution based computations.
TSUBAME 2.0 Compute Node

Thin node (HP ProLiant SL390s G7)
- Intel Xeon X5670 (Westmere-EP) x2
- NVIDIA Tesla M2050 x3
- Mellanox QDR x4 InfiniBand HCA x2

Medium node (HP ProLiant DL580 G7)
- Intel Xeon X7550 (Nehalem-EX) x4
- NVIDIA Tesla M2070 x4
  (NextIO vCORE Express 2070)
- Mellanox QDR x4 InfiniBand HCA x1
SINGLE NODE MULTI GPUS....
Four scheduling via host memory

1. Default
2. Exclusive
3. Overlapped
4. P2P

PCI-E & QPI are bi-directional; Important to fill both direction.
Performance of 3-D FFT using different scheduling policies (256x256x256, DP)

- default
- exclusive
- "+overlap"
- "+P2P"
Performance of 3-D FFT using P2P

![Graph showing performance comparison between CUFFT, NukadaFFT, 2GPU, 3GPU, and 4GPU for different data sizes (256, 384, 512). The x-axis represents the number of GPUs, and the y-axis represents performance in GFLOPS.](image-url)
Breakdown of $256^3$ 3-D FFT

<table>
<thead>
<tr>
<th>Time (msec.)</th>
<th>Single GPU</th>
<th>Total</th>
<th>1-D FFT for dim.Z</th>
<th>2-D FFT &amp; Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUFFT</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NukadaFFT</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2GPU</td>
<td>20</td>
<td>15</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3GPU</td>
<td>18</td>
<td>10</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4GPU</td>
<td>16</td>
<td>12</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Minimum time for 2-D FFT

- Single GPU: CUFFT
- 2 GPU: Total
- 3 GPU: 2-D FFT & Transfer
- 4 GPU: 2-D FFT & Transfer
Comm. between GPUs on multi-node

Three steps of DMA transfers
(1) Device-to-host by CUDA API
(2) Between node by MPI
(3) Host-to-device by CUDA API

All-to-all between P nodes performs \((P - 1)\) Comm. between GPUs.
Those steps should be pipelined, because all DMA controllers can work simultaneously.
On node #0

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>D2H</td>
<td>0-&gt;1</td>
<td>MPI_Send</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0-&gt;1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPI_Recv</td>
</tr>
<tr>
<td></td>
<td>99-&gt;0</td>
<td></td>
</tr>
<tr>
<td>D2H</td>
<td>0-&gt;2</td>
<td>MPI_Send</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0-&gt;2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPI_Recv</td>
</tr>
<tr>
<td></td>
<td>98-&gt;0</td>
<td></td>
</tr>
<tr>
<td>D2H</td>
<td>0-&gt;3</td>
<td>MPI_Send</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0-&gt;3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPI_Recv</td>
</tr>
<tr>
<td></td>
<td>97-&gt;0</td>
<td></td>
</tr>
<tr>
<td>D2H</td>
<td>0-&gt;4</td>
<td>MPI_Send</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0-&gt;4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPI_Recv</td>
</tr>
<tr>
<td></td>
<td>96-&gt;0</td>
<td></td>
</tr>
<tr>
<td>D2H</td>
<td>0-&gt;5</td>
<td>MPI_Send</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0-&gt;5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPI_Recv</td>
</tr>
<tr>
<td></td>
<td>95-&gt;0</td>
<td></td>
</tr>
<tr>
<td>D2H</td>
<td>0-&gt;6</td>
<td>MPI_Send</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0-&gt;6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPI_Recv</td>
</tr>
<tr>
<td></td>
<td>94-&gt;0</td>
<td></td>
</tr>
<tr>
<td>D2H</td>
<td>0-&gt;7</td>
<td>MPI_Send</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0-&gt;7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPI_Recv</td>
</tr>
<tr>
<td></td>
<td>93-&gt;0</td>
<td></td>
</tr>
</tbody>
</table>
3-D FFT using MPI (1GPU/node)

Performance (GFLOPS) vs. # of node for different data sizes:
- 256
- 384
- 512
- 1024
Multi-GPU per node, multiple nodes.

(1) Hybrid: multiple GPU per process

1GPU (0)  2GPU (0)  3GPU (0)

QDR x4 IB
CPU0
CPU1

GPU0  GPU1  GPU2

(2) Flat: one GPU per process

2GPU (0-0)  2GPU (0-1)

3GPU (0-0-0)  3GPU (0-1-1)  3GPU (0-0-1)
Performance with 64 nodes

Performance (GFLOPS)

1GPU(0)  2GPU(0)  3GPU(0)  2GPU(0-0)  2GPU(0-1)  3GPU(0-0-0)  3GPU(0-1-1)  3GPU(0-0-1)

256  384  512  1024
Summary

- Speed-up of 3-D FFT using multi-GPU
  - Up to 2X using 4 GPUs (single node)
  - Up to 4X using 32 GPUs (32 nodes)

- Small message in all-to-all limits the scalability

- Use of multi-GPU per node
  - Decrease performance for small data size
  - Increase performance for large data size