Multi-level Optimization of Matrix Multiplication for GPU-equipped Systems

K. Matsumoto\textsuperscript{a}, N. Nakasato\textsuperscript{a}, T. Sakai\textsuperscript{a},
H. Yahagi\textsuperscript{b}, S. G. Sedukhin\textsuperscript{a}

\textsuperscript{a}University of Aizu, \textsuperscript{b}Kyoto University; Japan

ICCS 2011, June 1, 2011
Outline

• Matrix Multiplication Problem
• Motivation
• AMD Cypress GPU
• DGEMM Implementation
  – Part 1: DGEMM Kernel
  – Part 2: DGEMM for Large Matrices
• Conclusions
Matrix-Matrix Multiplication

• GEMM (General Matrix Multiplication) is a fundamental linear algebra routine

\[ C \leftarrow AB + C, \quad C \leftarrow A^T B + C, \quad C \leftarrow AB^T + C, \quad C \leftarrow A^T B^T + C \]

• Existing DGEMM (Double-precision GEMM) implementations on GPU
  – On Cypress: 472 GFlop/s; 87% of peak (544 Gflop/s) [Nakasato]
  – On Fermi: 362 GFlop/s; 70% of peak (515 GFlop/s) [Tan]

Motivation

- DGEMM kernel with higher performance stability by considering memory access patterns.
- DGEMM for matrices whose required data size is larger than GPU memory capacity.

<table>
<thead>
<tr>
<th>Optimization</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st level</td>
<td>Building block for computation in each GPU thread</td>
</tr>
<tr>
<td>2nd level</td>
<td>Optimizing the memory access pattern in a group of GPU threads</td>
</tr>
<tr>
<td>3rd level</td>
<td>Maximizing the GPU utilization</td>
</tr>
</tbody>
</table>
AMD Cypress GPU (Radeon HD 5870)

• Evergreen GPU family launched in 2009

• 320 double-precision (DP) cores
  – 640 DP op/clock, using MAD (fused multiply-add) instruction

• DP peak perf.: 544 GFlop/s
  (=640 [op/clock] * 0.85 [GHz])
• Cypress GPU schedules a range of threads onto a group of DP cores.
  – 16 DP cores / group

• 64 threads are a unit of workload on a group.
  – Changing the order of thread assignment changes memory access patterns.
PART 1: DGEMM KERNEL
Layout Functions

• map a thread ID to the memory index.

• are based on space filling curves.

Z-Morton  X-Morton  U-Morton  Hilbert
Performance of $C \leftarrow A^T B + C$

Perf. [GFlop/s] = $\frac{2n^3 \text{ [Flops]}}{\text{Exec. time [sec]}}$
Cache hit rate of $C \leftarrow A^T B + C$

- Stability of performance is related to the cache hit rate.

- When $n = 4096$

<table>
<thead>
<tr>
<th></th>
<th>X-Morton</th>
<th>Z-Morton</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perf. [GFlop/s]</td>
<td>437</td>
<td>404</td>
</tr>
<tr>
<td>Cache hit rate [%]</td>
<td>28.8</td>
<td>20.0</td>
</tr>
</tbody>
</table>
Performance of $C \leftarrow A^T B$

- No prominent difference
- Max perf.: 480 GFlop/s (88% of 544 GFlop/s)
PART 2: DGEMM FOR LARGE MATRICES
Using $C \leftarrow A^T B$ kernel

- Fastest kernel among all tested kernels.
- No need to send a matrix $C$ to GPU.
- Load with transposition by CPU if necessary.
  - Example: $C \leftarrow AB$
• Reusing matrix blocks within the GPU and minimizing the amount of communication.
   – asymptotically sending 1 matrix block and receiving 1 matrix block during a DGEMM kernel execution on the GPU.

• Explicitly loading/storing matrix blocks to/from PCI-Express memory.
Maximum Performance of $C \leftarrow AB + C$

**CPU + 1 GPU**

![Graph showing performance for CPU + 1 GPU](image1)

**CPU + 2 GPUs**

![Graph showing performance for CPU + 2 GPUs](image2)

Core i7 960 CPU (4-core at 3.2 GHz) + Radeon HD 5870 GPUs
Core i7 920 CPU (4-core at 2.67 GHz) + Radeon HD 5870 GPUs
Performance of $C ← AB + C$

Max perf.: 921 GFlop/s

Max perf.: 472 GFlop/s
Conclusions

- DGEMM for systems which contain AMD Cypress GPUs
- Effects of memory access patterns to the DGEMM kernel’s performance
  - $C \leftarrow A^{T}B + C$ kernel with X-Morton layout shows the superior performance
- DGEMM for large matrices on hybrid CPU-GPU systems
  - Max perf. ($n \approx 30000$): 472 GFlop/s (on CPU + 1 GPU), 921 GFlop/s (on CPU + 2 GPUs)
- Future work: optimizing DGEMM for non-square matrices and utilizing it to other linear algebra problems