Optimization of Matrix Multiplication for CPU-GPU Systems

Kazuya Matsumoto, Naohito Nakasato, Stanislav G. Sedukhin
The University of Aizu, Japan, e-mail: d8121101@u-aizu.ac.jp

Introduction

General matrix-matrix multiply (GEMM) is a fundamental linear algebra routine in Level 3 BLAS. GEMM is used in many important numerical algorithms such as other Level 3 BLAS routines and one-sided factorizations. The computational intensity and the regular memory access pattern of GEMM is well suited for acceleration by GPUs.
This poster presents results of our study on DGEMM (double-precision GEMM) and SGEMM (single-precision GEMM) for hybrid CPU-GPU systems. We utilized the DGEMM and SGEMM kernels previously implemented for GPU from AMD, and have implemented the GEMM routines for large matrices where whole data cannot be allocated in GPU memory at the same time. The achieved performance of the routines are up to 0.47 TFlop/s in DGEMM and 2 TFlop/s in SGEMM.

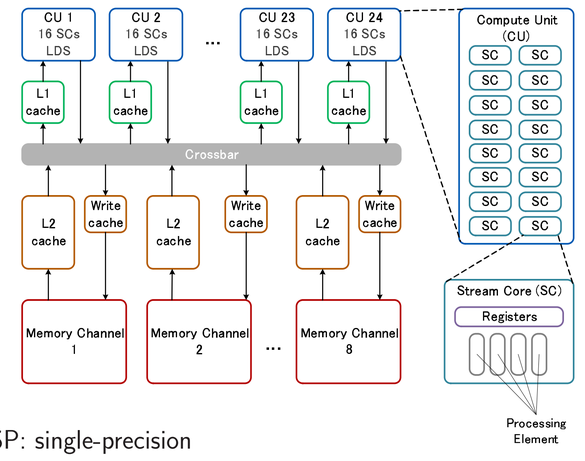
GEMM (General Matrix-Matrix Multiply)

- $C \leftarrow \alpha \mathrm{op}(A)\mathrm{op}(B) + \beta C$
  - $\mathrm{op}(A)$ is an $m \times k$ matrix, $\mathrm{op}(B)$ is a $k \times n$ matrix, and C is an $m \times n$ matrix.
  - α and β are scalars.
  - $\mathrm{op}(X)$ is X (nontransposed) or X^T (transposed).
    1. $C \leftarrow AB + C$, where $c(i,j) \leftarrow \sum_{p=1}^{k} a(i,p) \cdot b(p,j) + c(i,j)$;
    2. $C \leftarrow A^T B + C$, where $c(i,j) \leftarrow \sum_{p=1}^{k} a(p,i) \cdot b(p,j) + c(i,j)$;
    3. $C \leftarrow AB^T + C$, where $c(i,j) \leftarrow \sum_{p=1}^{k} a(i,p) \cdot b(j,p) + c(i,j)$;
    4. $C \leftarrow A^T B^T + C$, where $c(i,j) \leftarrow \sum_{p=1}^{k} a(p,i) \cdot b(j,p) + c(i,j)$.

GPU Specification

Architecture	Cayman	Cypress
Board name	Radeon HD 6970	Radeon HD 5870
Num. of SP cores	1536	1600
Num. of DP cores	384	320
VLIW width	4	5
Num. of CU	24	20
Core clock [GHz]	0.88	0.85
SP peak perf. [GFlop/s]	2703	2720
DP peak perf. [GFlop/s]	676	544
Memory clock [GHz]	1.375	1.2
Memory size [GB]	2	1
L2 cache size [kB]	512	512
L1 cache size / CU [kB]	8	8
LDS size / CU [kB]	32	32
Size of registers / CU [kB]	256	256
Memory BW [GB/s]	176	154
L2 read BW [GB/s]	451	435
L1 read BW [GB/s]	1352	1088
LDS read BW [GB/s]	2703	2176

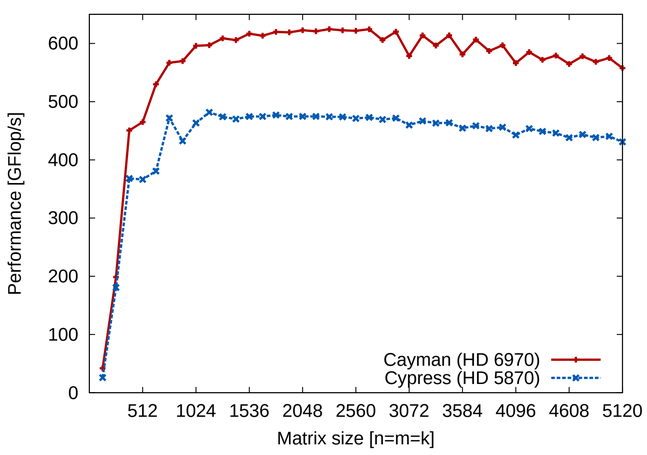
Cayman Architecture



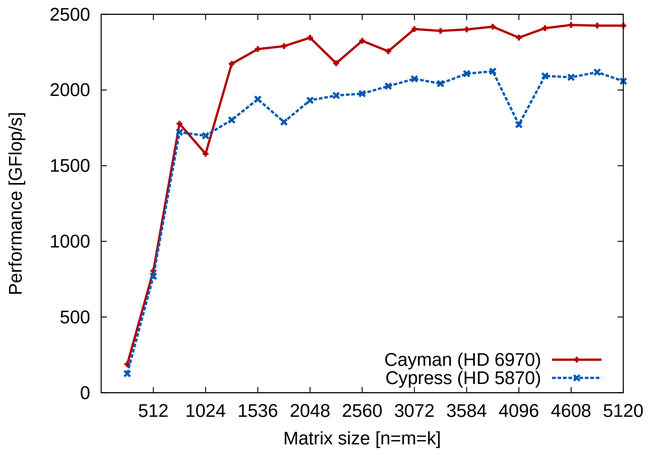
- SP: single-precision
- DP: double-precision
- VLIW: Very Long Instruction Word
- CU: Compute Unit
- LDS: Local Data Share (shared memory in each CU)
- BW: bandwidth

GEMM Kernels on GPUs

Performance of DGEMM Kernel $C \leftarrow A^T B$



Performance of SGEMM Kernel $C \leftarrow A^T B$



- Kernels were written in assembler-like language called *Intermediate Language* (IL).
- $C \leftarrow A^T B$ kernel is the fastest among GEMM variants in row-major layout.
  - Ref.: N. Nakasato, *A fast GEMM implementation on the Cypress GPU*, ACM SIGMETRICS Performance Evaluation Review, vol. 38, no. 4, pp. 50-55, Mar. 2011.

Maximum Performance

		Cayman (HD 6970)		Cypress (HD 5870)	
	Variant	Perf. [GFlop/s]	Efficiency	Perf. [GFlop/s]	Efficiency
DGEMM	$C \leftarrow A^T B$	624	92%	482	88%
	$C \leftarrow AB$	449	66%	360	66%
	$C \leftarrow A^T B^T$	448	66%	360	66%
	$C \leftarrow AB^T$	337	50%	270	50%
SGEMM	$C \leftarrow A^T B$	2432	90%	2137	79%
	$C \leftarrow AB$	1799	67%	1428	53%
	$C \leftarrow A^T B^T$	1794	66%	1409	52%
	$C \leftarrow AB^T$	1348	50%	1067	39%

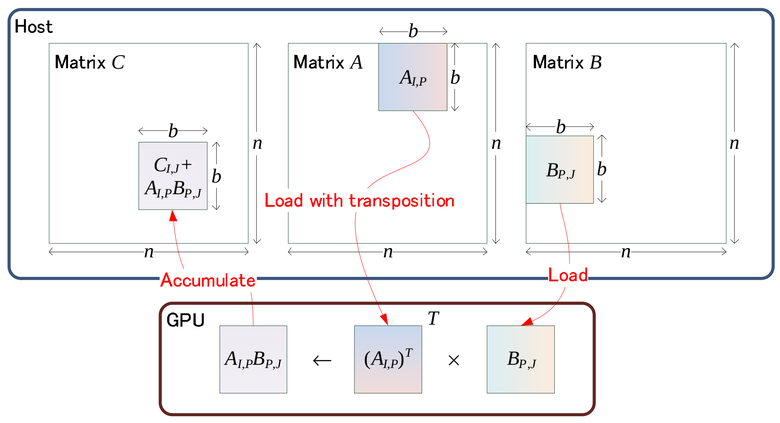
GEMM for Large Matrices on CPU-GPU Systems

System Configurations

	System A	System B	System C
GPU	Cayman (AMD Radeon HD 6970)	Cypress (ATI Radeon HD 5870)	
CPU	Sandy Bridge (Intel Core i7 2600K; 4 cores at 3.4 GHz)	Gulftown (Intel Core i7 970; 6 cores at 3.2 GHz)	Bloomfield (Intel Core i7 960; 4 cores at 3.2 GHz)
Display driver	AMD Catalyst 11.3		ATI Catalyst 10.12
SDK	AMD APP SDK v2.5		ATI Stream SDK v2.2

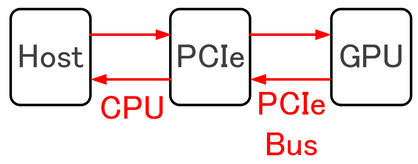
Using $C \leftarrow A^T B$ Kernel

- Fastest kernel.
- No need to send matrix C to GPU.
- Load matrices with transposition by CPU if necessary.
  - Example: $C \leftarrow AB + C$

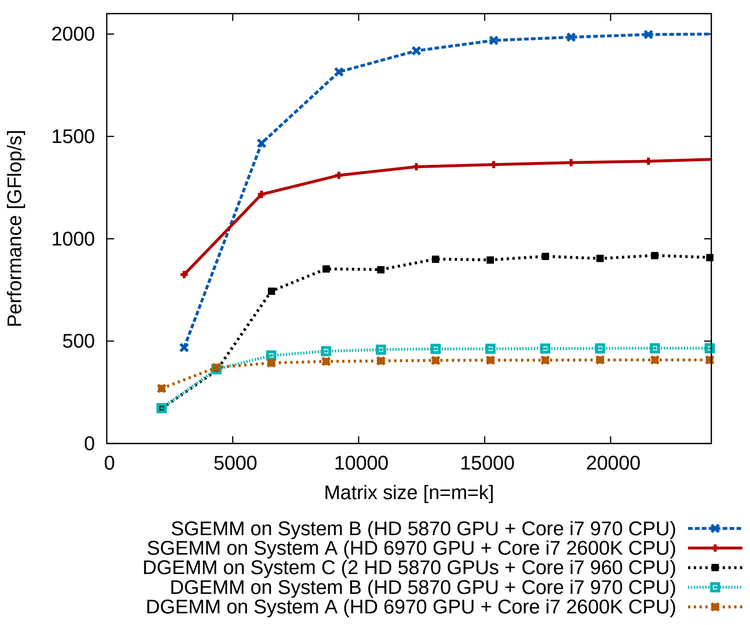


Hiding Data Communication Latency

- Reuses block matrix data within the GPU and minimizing the amount of communication.
  - Asymptotically requires sending 1 matrix block and receiving 1 matrix block during a GEMM kernel execution on the GPU.
- Explicitly loads/stores matrix blocks to/from PCI-Express (PCIe) memory by the CPU.
  - Two communication processes are required to send/receive data between the host memory and the GPU memory.
    1. Communication between the host memory and the PCIe memory.
    2. Communication between the PCIe memory and the GPU memory.
  - This explicit loading/sending virtually parallelizes the communication processes.

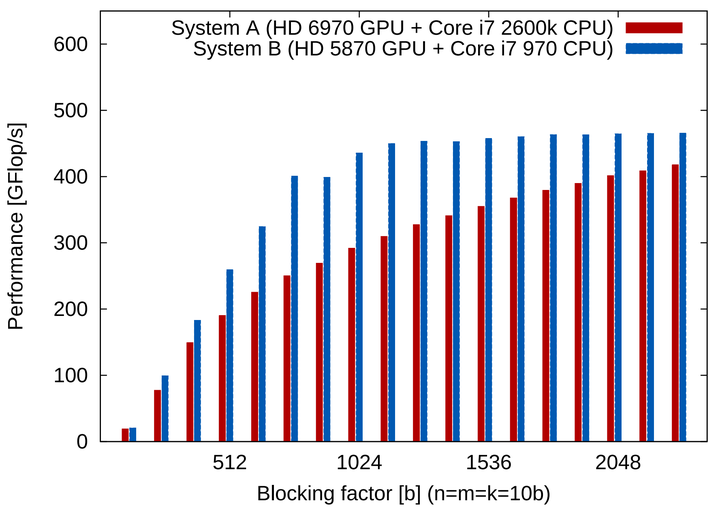


Performance of $C \leftarrow AB + C$ on CPU-GPU Systems

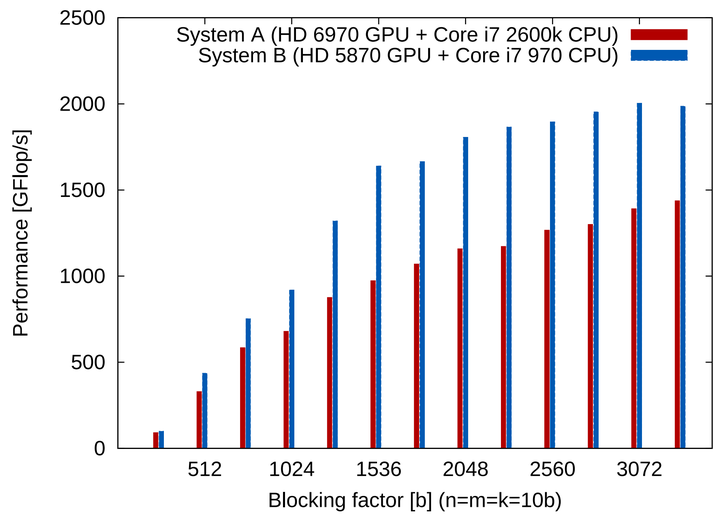


- Blocking factor b is 2176 in DGEMM and 3072 in SGEMM.
  - Largest available block factor is determined by the capacity of PCIe memory (around 500 MB in our systems).
- Overlapping the computation with the communication has not perfectly been implemented in System A containing Radeon HD 6970.

Performance of DGEMM $C \leftarrow AB + C$ with Different Blocking Factors



Performance of SGEMM $C \leftarrow AB + C$ with Different Blocking Factors



Maximum Performance

		System A	System B
	Variant	Perf. [GFlop/s]	Perf. [GFlop/s]
DGEMM	$C \leftarrow A^T B + C$	419	467
	$C \leftarrow AB + C$	417	467
	$C \leftarrow A^T B^T + C$	418	467
	$C \leftarrow AB^T + C$	400	466
SGEMM	$C \leftarrow A^T B + C$	1455	2010
	$C \leftarrow AB + C$	1436	2010
	$C \leftarrow A^T B^T + C$	1442	2010
	$C \leftarrow AB^T + C$	1301	1577

Reference: *Multi-level Optimization of Matrix Multiplication for GPU-equipped Systems*, K. Matsumoto, N. Nakasato, T. Sakai, H. Hideki, S. G. Sedukhin, In Proc. the 11th International Conference on Computational Science (ICCS 2011), Volume 4, pp. 342-351, Singapore, June 2011.