# An Optimized Large-Scale Hybrid DGEMM Design for CPUs and ATI GPUs

Jiajia Li, Xingjian Li, Guangming Tan, Mingyu Chen, and Ninghui Sun

Institute of Computing Technology
Chinese Academy of Sciences

2012-06-28

# Outline

- Motivation
- DGEMM on ATI GPU
- Optimization & Performance
- Findings
- Conclusion

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

# Motivation

- **GPU performance**

  GOOD

  - **Intel X5650 V.S. ATI HD5970**
  - **128 GFLOPS 928 GFLOPS (double)**

- **DGEMM**

  Suit to GPU

**C := alpha × A × B + beta × C**

# Top500 (June 2012)

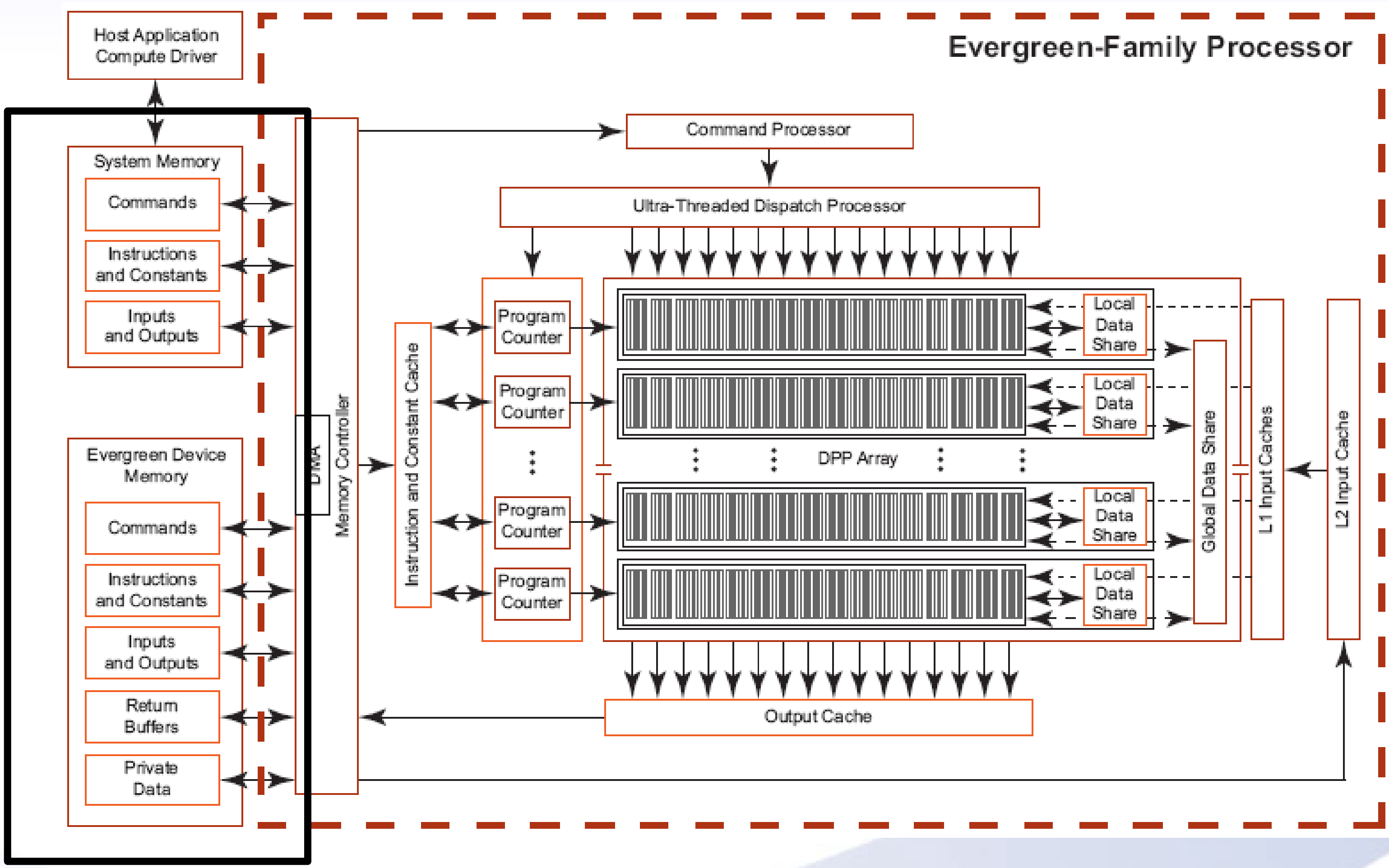
| Rank | Site | Manufacturer | Computer | Cores | Rmax [Pflops] | Rpeak [Pflops] | Efficiency |
|---|---|---|---|---|---|---|---|
| 1 | Lawrence Livermore National Laboratory | IBM | Sequoia<br>BlueGene/Q,<br>Power BQC 16C 1.6GHz, Custom | 1,572,864 | 16.3 | 20.1 | 81% |
| 2 | RIKEN Advanced Institute for Computational Science | Fujitsu | K Computer<br>SPARC64 VIIIfx 2.0GHz,<br>Tofu Interconnect | 795,024 | 10.5 | 11.3 | 93% |
| 3 | Argonne National Laboratory | IBM | Mira<br>BlueGene/Q,<br>Power BQC 16C 1.6GHz, Custom | 786,432 | 8.16 | 10.1 | 81% |
| 4 | Leibniz Rechenzentrum | IBM | SuperMUC<br>iDataPlex DX360M4,<br>Xeon E5 8C 2.7GHz, Infiniband FDR | 147,456 | 2.90 | 3.19 | 91% |
| 5 | National SuperComputer Center in Tianjin | NUDT | Tianhe-1A<br>NUDT TH MPP,<br>Xeon 6C, NVidia, FT-1000 8C | 186,368 | 2.57 | 4.70 | 55% |
| 6 | Oak Ridge National Laboratory | Cray | Jaguar<br>Cray XK6, Opteron 16C 2.2GHz,<br>Gemini, NVIDIA 2090 | 298,592 | 1.94 | 2.63 | 74% |
| 7 | CINECA | IBM | Fermi<br>BlueGene/Q,<br>Power BQC 16C 1.6GHz, Custom | 163,840 | 1.73 | 2.10 | 82% |
| 8 | Forschungszentrum Juelich (FZJ) | IBM | JuQUEEN<br>BlueGene/Q,<br>Power BQC 16C 1.6GHz, Custom | 131,072 | 1.38 | 1.68 | 82% |
| 9 | Commissariat a l'Energie Atomique CEA/TGCC-GENCI | Bull | Curie thin nodes<br>Bullx B510,<br>Xeon E5 8C 2.7GHz, Infiniband QDR | 77,184 | 1.36 | 1.67 | 82% |
| 10 | National Supercomputing Centre in Shenzhen | Dawning | Nebulae<br>TC3600 Blade,<br>Intel X5650, NVidia Tesla C2050 GPU | 120,640 | 1.27 | 2.98 | 43% |

# Top500 (June 2012)

| Rank | Site | Manufacturer | Computer | Cores | Rmax [Pflops] | Rpeak [Pflops] | Efficiency |
|---|---|---|---|---|---|---|---|
| 1 | Lawrence Livermore National Laboratory | IBM | Sequoia<br>BlueGene/Q,<br>Power BQC 16C 1.6GHz, Custom | 1,572,864 | 16.3 | 20.1 | 81% |
| 2 | RIKEN Advanced Institute for Computational Science | Fujitsu | K Computer<br>SPARC64 VIIIfx 2.0GHz,<br>Tofu Interconnect | 795,024 | 10.5 | 11.3 | 93% |
| 3 | Argonne National Laboratory | IBM | Mira<br>BlueGene/Q,<br>Power BQC 16C 1.6GHz, Custom | 786,432 | 8.16 | 10.1 | 81% |
| 4 | Leibniz Rechenzentrum | IBM | SuperMUC<br>iDataPlex DX360M4,<br>Xeon E5 8C 2.7GHz, Infiniband FDR | 147,456 | 2.90 | 3.19 | 91% |
| 5 | National SuperComputer Center in Tianjin | [illegible] | Tianh[illegible]1A<br>[illegible] | [illegible]6,368 | 2.57 | 4.70 | 55% |
| 6 | Oak Ridge National Laboratory | Cra[illegible] | [illegible] | [illegible] | [illegible] | 2.63 | 74% |
| 7 | CINECA | IBM | Fermi<br>BlueGene/Q,<br>Power BQC 16C 1.6GHz, Custom | 163,840 | 1.7[illegible] | 2.10 | 82% |
| 8 | Forschungszentrum Juelich (FZJ) | IBM | JuQUEEN<br>BlueGene/Q,<br>Power BQC 16C 1.6GHz, Custom | 131,072 | 1.38 | 1.68 | 82% |
| 9 | Commissariat a l'Energie Atomique CEA/TGCC-GENCI | Bull | Curie thin nodes<br>Bullx B510,<br>Xeon E5 8C 2.7GHz, Infiniband QDR | 77,184 | 1.36 | 1.67 | 82% |
| 10 | National Supercomputing Centre in Shenzhen | Dawning | Nebulae<br>TC3600 Blade,<br>Intel X5650, NVidia Tesla C2050 GPU | 120,640 | 1.27 | 2.98 | 43% |

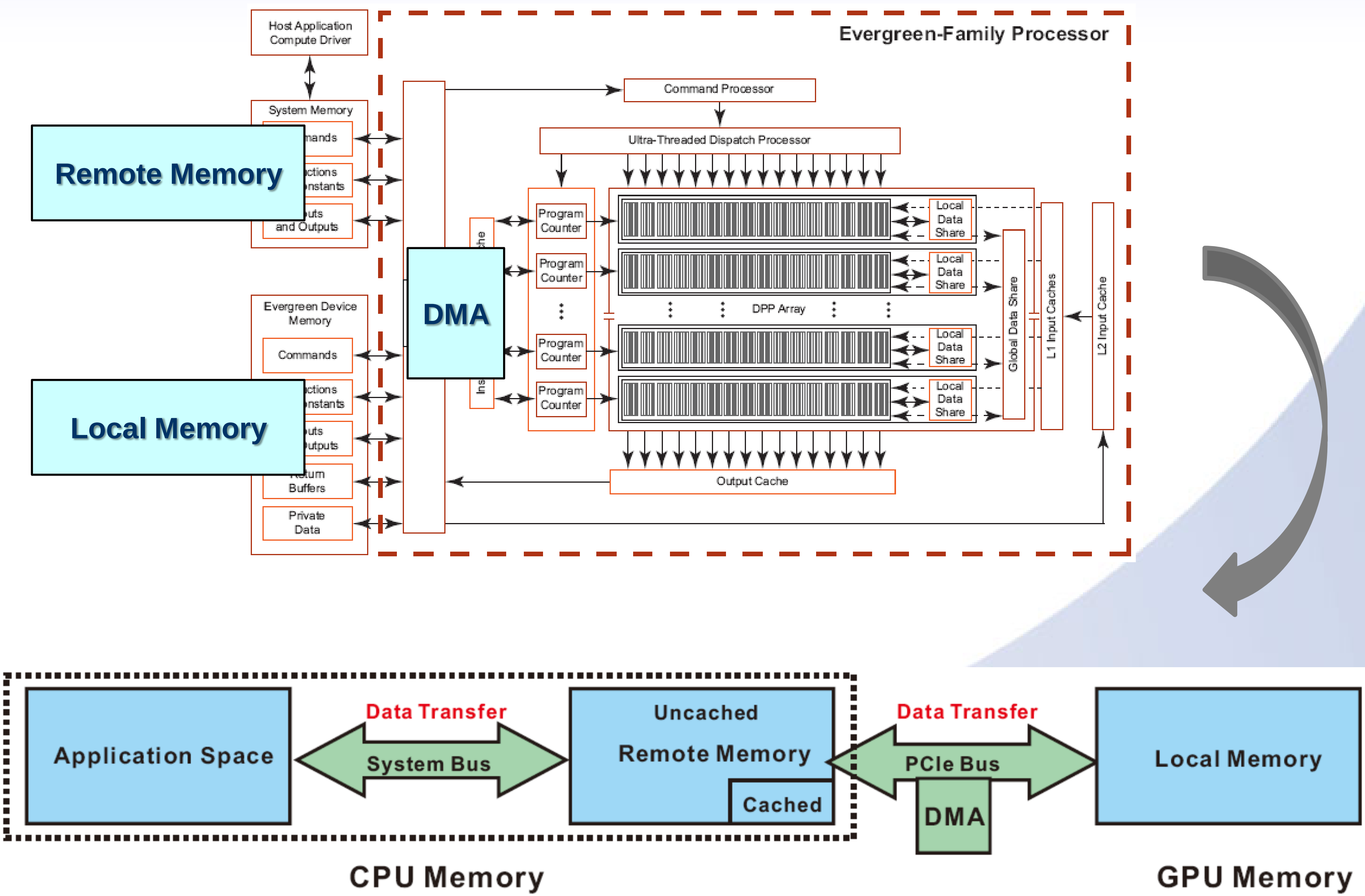
**Data Transfer**

# ATI GPU architecture




**Memory Hierarchy in ATI Compute Abstraction Layer (CAL)**

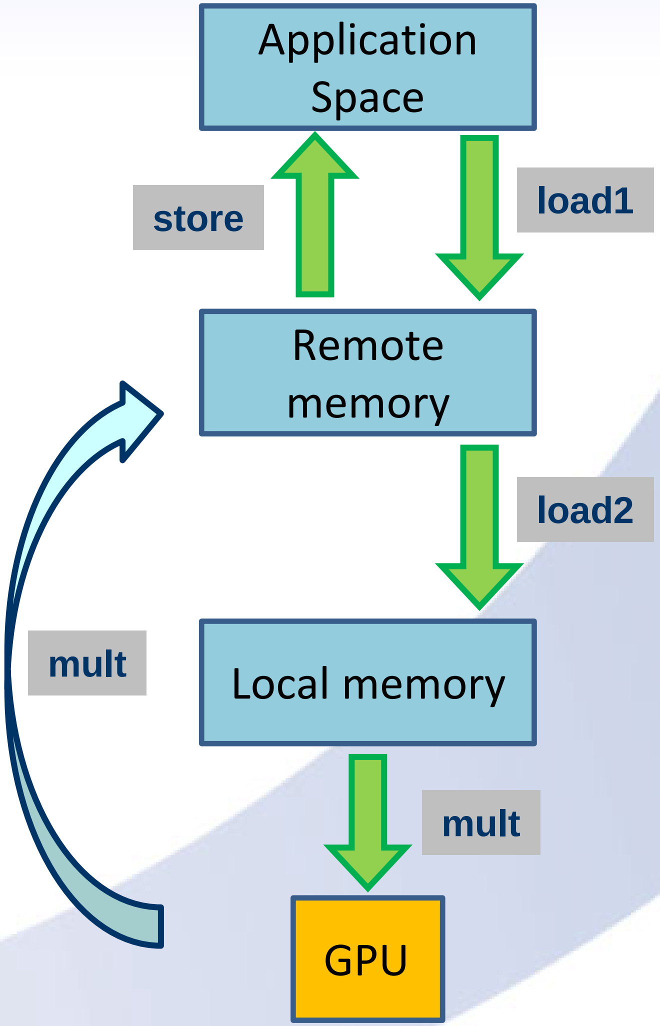
# ATI GPU architecture



# Prior DGEMM implementation

**Four-stage pipelining**

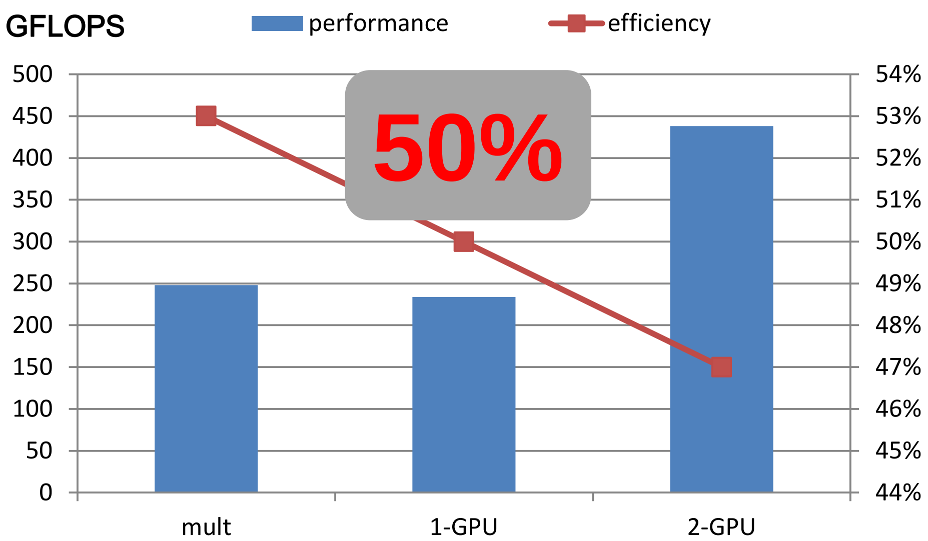
*Partition:* $A = \{A_1, A_2, \cdots, A_p\}, B = \{B_1, B_2, \cdots, B_q\},$
$C = \{C_1, C_2, \cdots, C_{p \times q}\}$
*Work units:* $WU = \{C_1 = A_1 \times B_1, C_2 = A_1 \times B_2, \ldots\}$
$C_{i,j}$: *the sub-matrices of* $C_j$
//////////////////////////////////////////////////////
1. bind remote memory for sub-matrices A,B,C
//*pre-processing*
Allocate workunits using the "bounce corner turn" for exploiting data reuse
//*the for-loop is pipelined*
2. *for* each workunit $wu_i$ do  $//i = 1, 2, \cdots, p \times q$
//*load1*
3. copy either $A_i$ or $B$ **load1** plication space to remote memory
//*load2*
4. copy either $A_i$ or $B_i$ from rem **load2** y to local memory
//*mult*
5. calculate $C_{i,1}$ on GPU device and output it to remote memory
6. *for* each block $C_{i,j}$ do  $//j = 2, 3, \cdots$
//*store*
7. copy $C_{i,j-1}$ from remote memory to application space
(also multiplied by beta)
//*mult*
8. calculate $C_{i,j}$ on GPU device and output it **mult** e memory
9. *endfor*
//*store*
10. copy the last $C_{i,j}$ from remot **store** to application space
(also multiplied by beta)
11. *endfor*




[C. Yang et. al. "Adaptive optimization for petascale heterogeneous CPU/GPU computing." 2010]

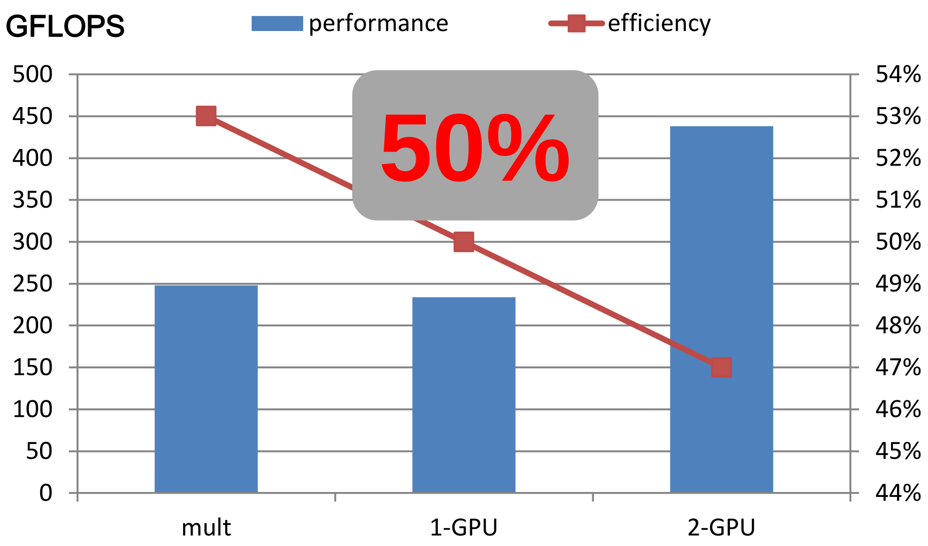
# Prior DGEMM performance

## Performance

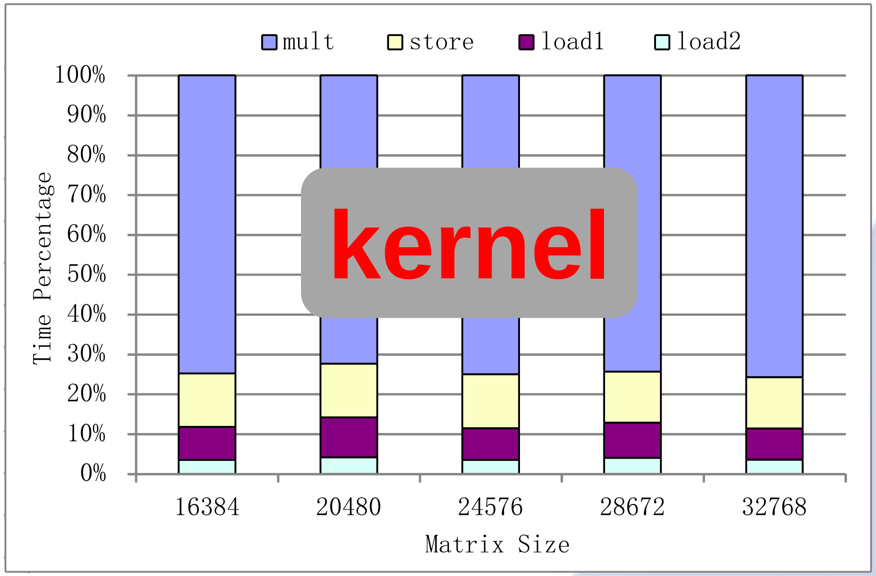


# Prior DGEMM performance

## Performance

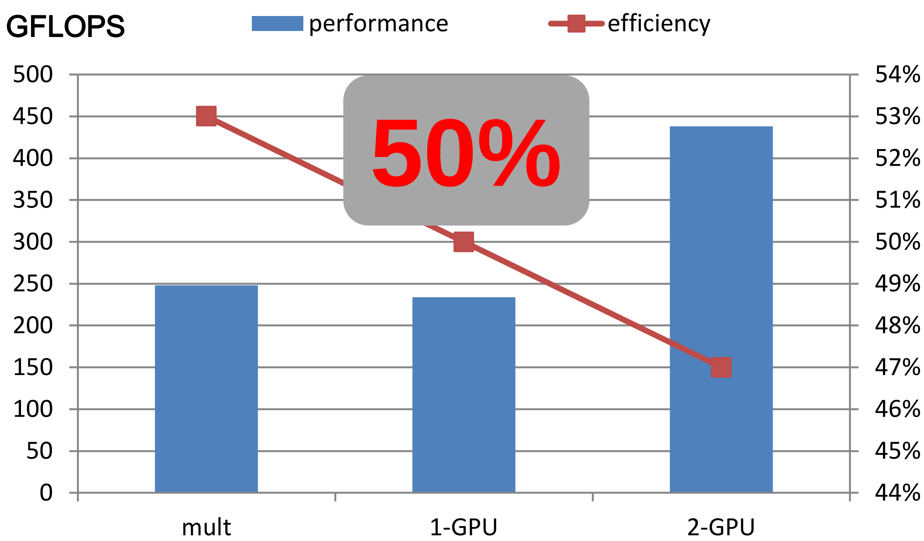



## Profile

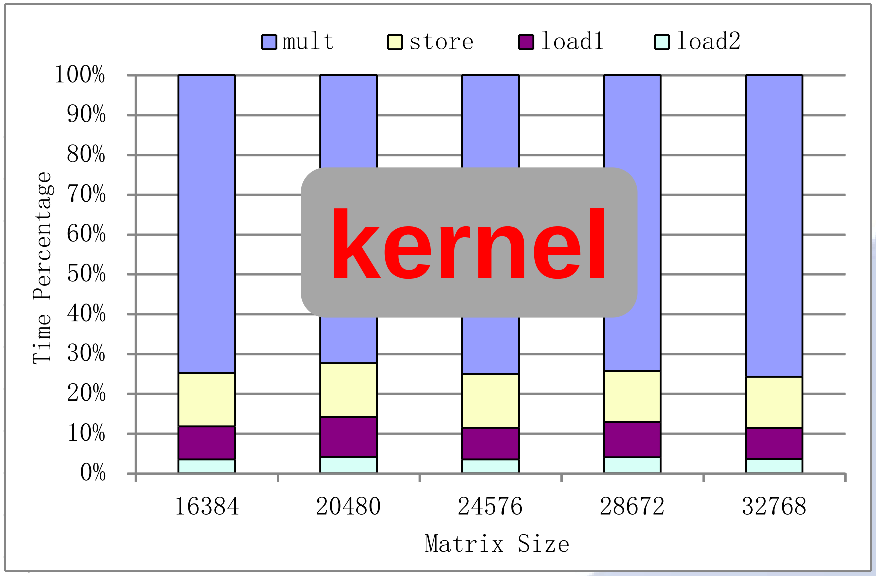


# Prior DGEMM performance

## Performance



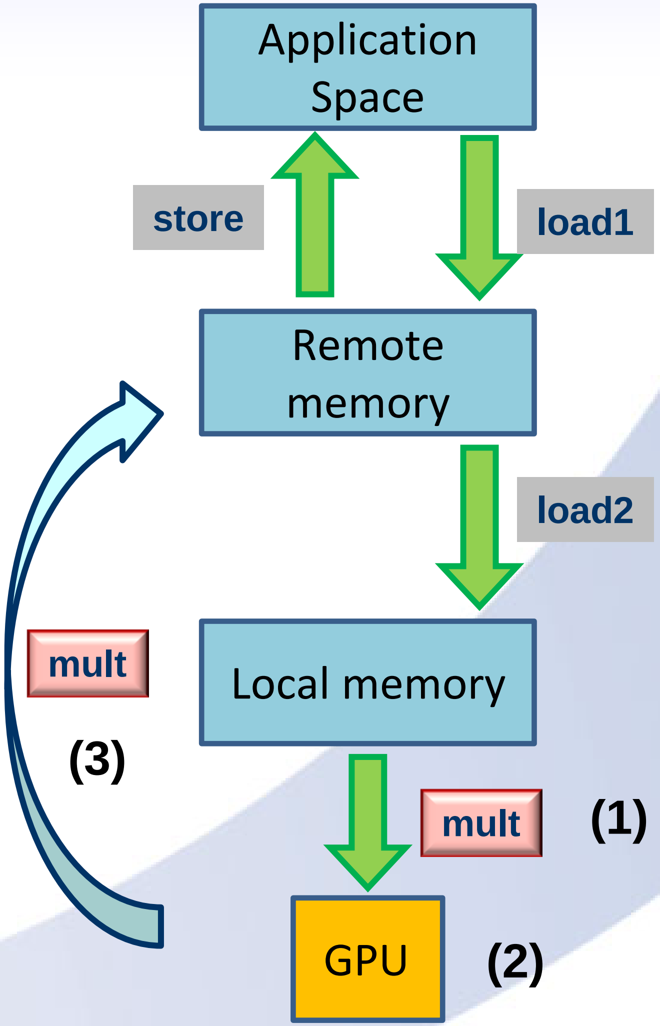

## Profile




**Bottleneck: kernel**

# Optimizations

- **Image addressing mode**
- **Five-stage pipelining**




**Stage (3) – long latency**

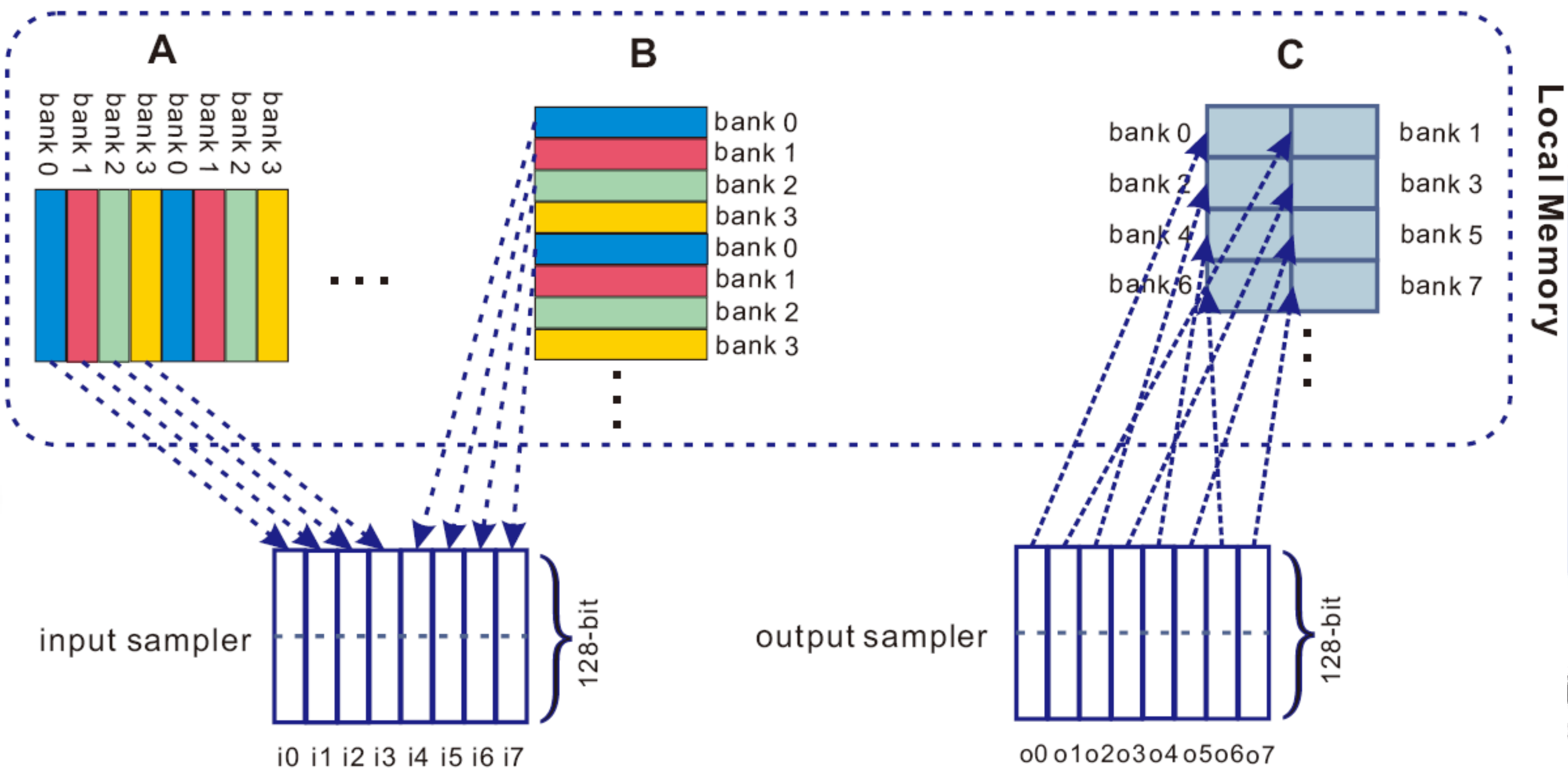
# Addressing mode

| | Image Addressing | V.S. | Global Buffer Addressing |
|---|---|---|---|
| Location | Pre-allocated segments | | Arbitrary |
| Program | Complicated | | easy |
| Latency | Short | | Long |

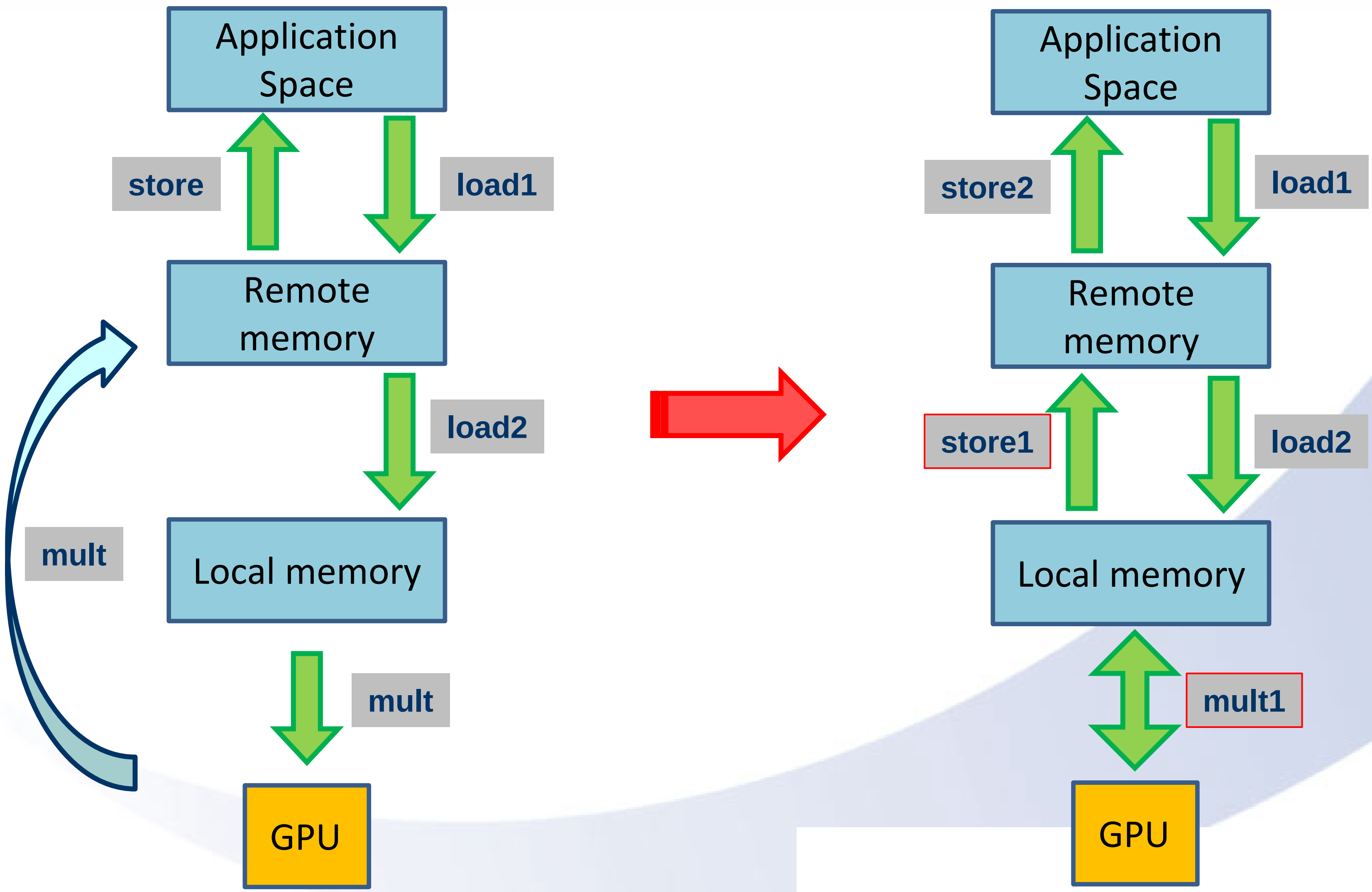
中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

# Addressing mode

| | Image Addressing | V.S. | Global Buffer Addressing |
|---|---|---|---|
| Location | Pre-allocated segments | | Arbitrary |
| Program | Complicated | | easy |
| Latency | Short | | Long |



# Five-stage pipelining



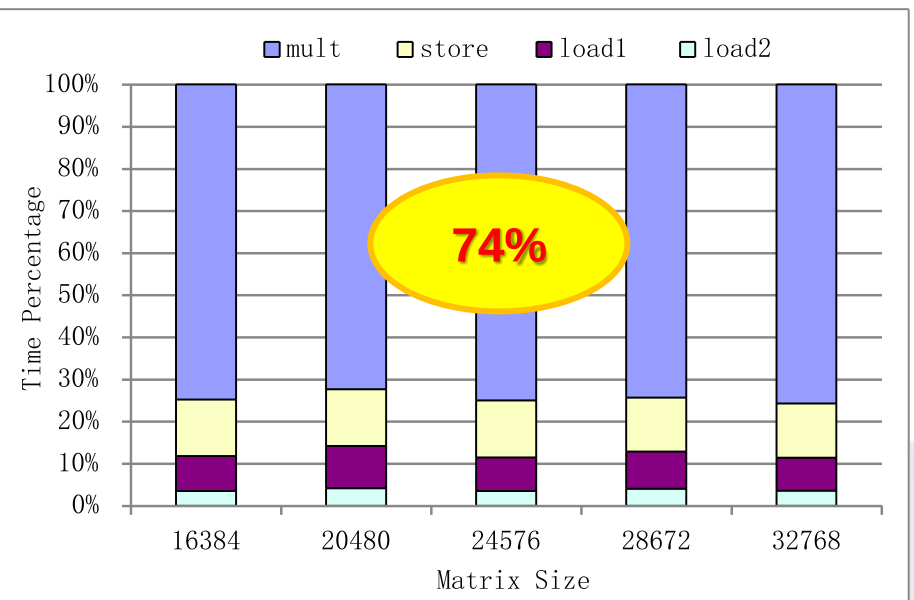
# Five-stage pipelining (cont.)

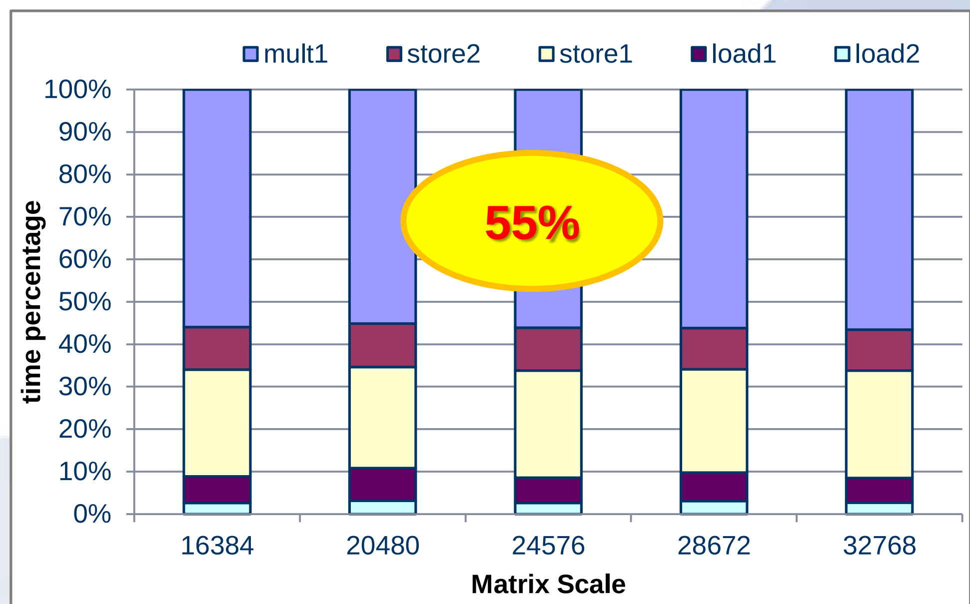
## Resource allocation

| | Host Memory | GPU | PCIe Bus |
|---|---|---|---|
| load1 | X | | |
| load2 | | | X |
| mult | | X | X |
| store | X | | |

| | Host memory | GPU | PCIe Bus |
|---|---|---|---|
| load1 | X | | |
| load2 | | | X |
| mult1 | | X | |
| store1 | | | X |
| store2 | X | | |

## Time percentage



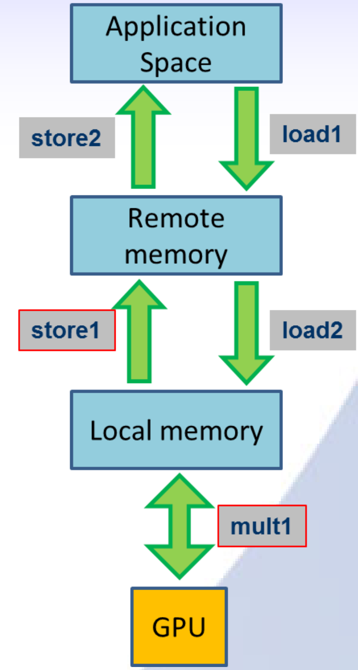



# Optimized DGEMM

**Algorithm 3** The five-stage software-pipelining DGEMM

*Partition:* $A = \{A_1, A_2, \cdots, A_p\}, B = \{B_1, B_2, \cdots, B_q\},$
$C = \{C_1, C_2, \cdots, C_{p\times q}\}$
*Work units:* $WU = \{C_1 = A_1 \times B_1, C_2 = A_1 \times B_2, \cdots\}$
$C_{i,j}$: *the sub-matrices of* $C_j$
////////////////////////////////////////////////////////////////////
1. bind remote memory for sub-matrices A,B,C
*//pre-processing*
Allocate workunits using the "bounce corner turn"
*//the for-loop is pipelined*
2. *for* each workunit $wu_i$ do  $//i = 1, 2, \cdots, p \times q$
*//load1*
3. copy either $A_i$ or $B_i$ from application space to remote memory
*//load2*
4. copy either $A_i$ or $B_i$ from remote memory to local memory
*//mult*
5. *DMA Pipeline($C_{i,1}$)*
6. *for* each block $C_{i,j}$ do  $//j = 2, 3, \cdots$
*//store2*
7. copy $C_{i,j-1}$ from remote memory to application space (also multiplied by beta)
*//mult*
8. *DMA Pipeline($C_{i,j}$)*
9. *endfor*
*//store2*
10. copy the last $C_{i,j}$ from remote memory to application space (also multiplied by beta)
11. *endfor*

*Algorithm: DMAPipeline($C_{i,j}$)*
$C_{i,j,k}$: *the sub-blocks of* $C_{i,j}$
////////////////////////////////////////////////////////////////////
*//the for-loop is pipelined*
*//mult1*
1. calculate $C_{i,j,1}$ in local memory
2. *for* each sub-block $C_{i,j,k}$ do  $//k = 2, 3, \cdots$
*//store1*
3. DMA transfer $C_{i,j,k-1}$ from local memory to remote memory
*//mult1*
4. calculate $C_{i,j,k}$ in local memory
5. *endfor*
*re1*
DMA transfer the last $C_{i,j,k}$ from local memory to remote memory




**Benefits:**

- Faster kernel
- DMA usage
- Better overlap

# Experimental platform & problem size

## Platform configuration

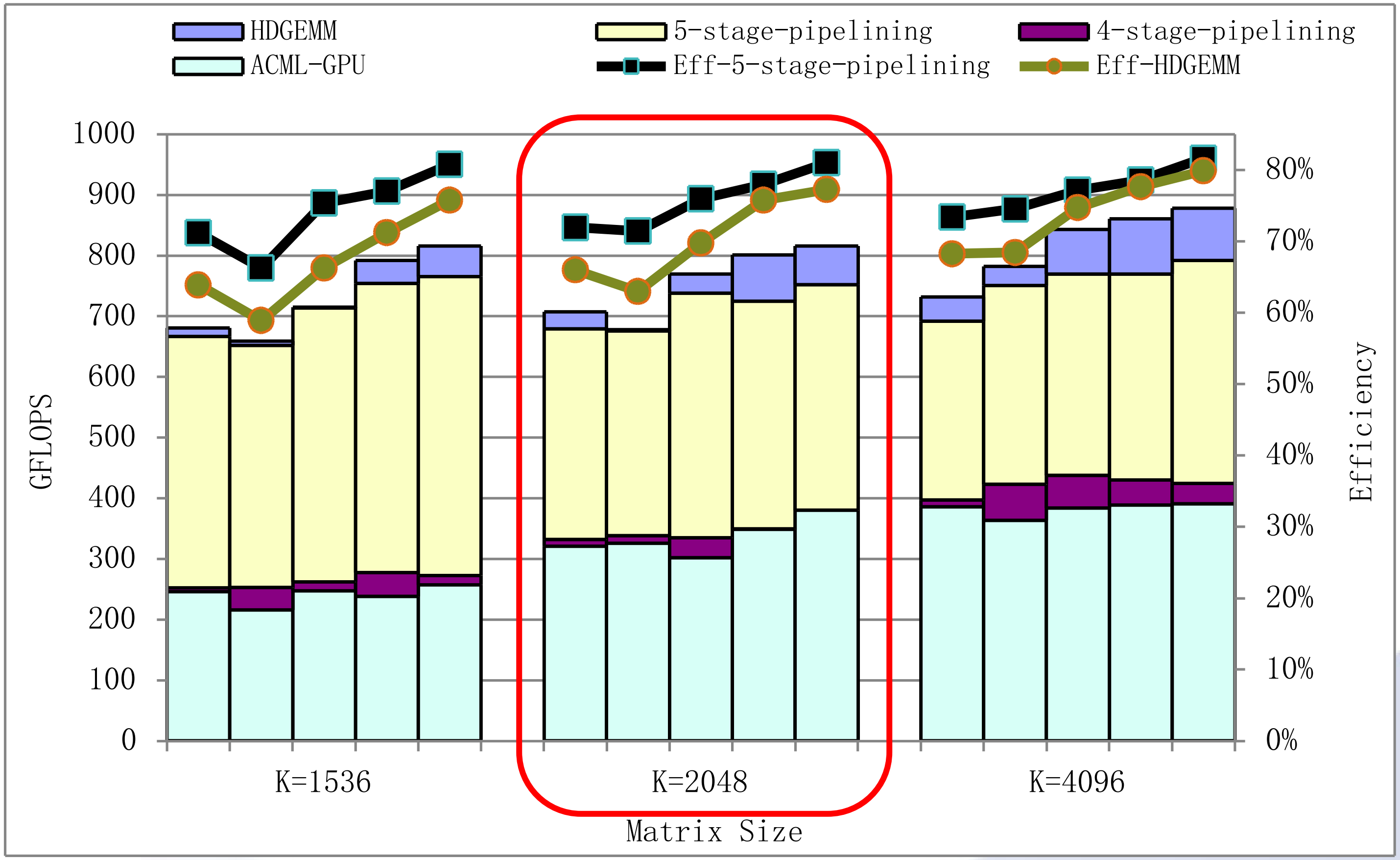
| | | |
|---|---|---|
| Processors | Xeon X5650 | $Radeon^{TM}$ HD5970 |
| Model | Westmere-EP | Cypress |
| Frequency | 2.66GHz | 725MHz |
| #chips | 2 | 2 |
| DP | 128 GFLOPS | 928 GFLOPS |
| DRAM type | DDR3 1.3GHz | GDDR5 1.0GHz |
| DRAM size | 24GB | 2GB |
| DRAM bandwidth | 31.2 GB/s | 256 GB/s |
| PCIe2.0 | x16, 8 GB/s | |
| Programming | icc + openmpi | ATI Stream SDK 2.2 |

## Matrix size

| GB / k | m=n | | | | |
|---|---|---|---|---|---|
| | 16384 | 20480 | 24576 | 28672 | 32768 |
| 1536 | 2.55 | 3.86 | 5.44 | 7.28 | 9.40 |
| 2048 | 2.68 | 4.03 | 5.64 | 7.52 | 9.66 |
| 4096 | 3.22 | 4.70 | 6.44 | 8.46 | 10.74 |

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

# Optimized DGEMM performance




**844 GFLOPS**
**Eff: 80%**

**758 GFLOPS**
**Eff: 82%**

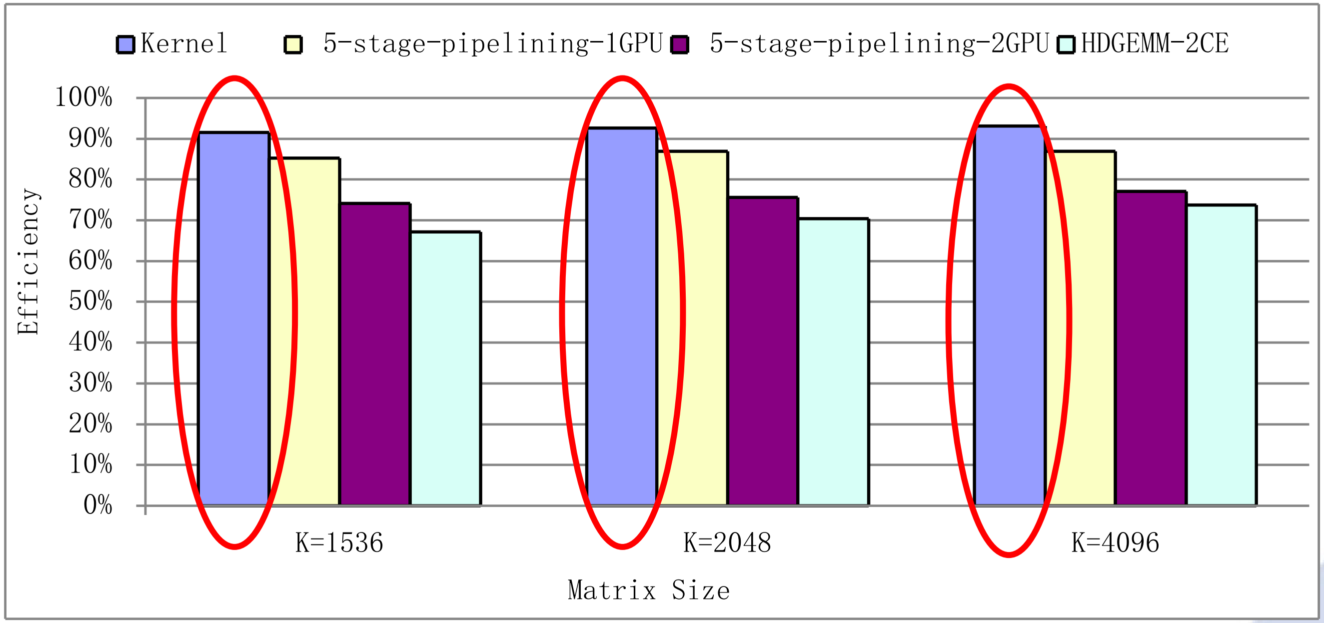
**Speedup: 2X**

# Findings

- We CAN achieve high efficiency on GPU !
- Contention means a LOT !
  - PCIe bus contention
  - Host memory contention

# DGEMM kernel performance

## Intra-node scalability

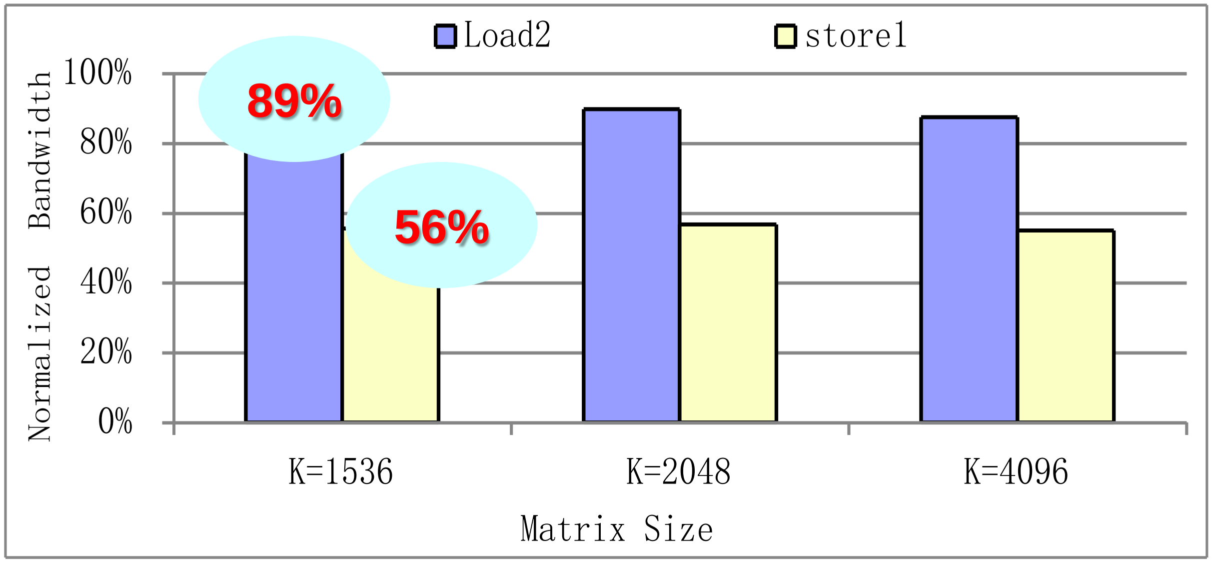


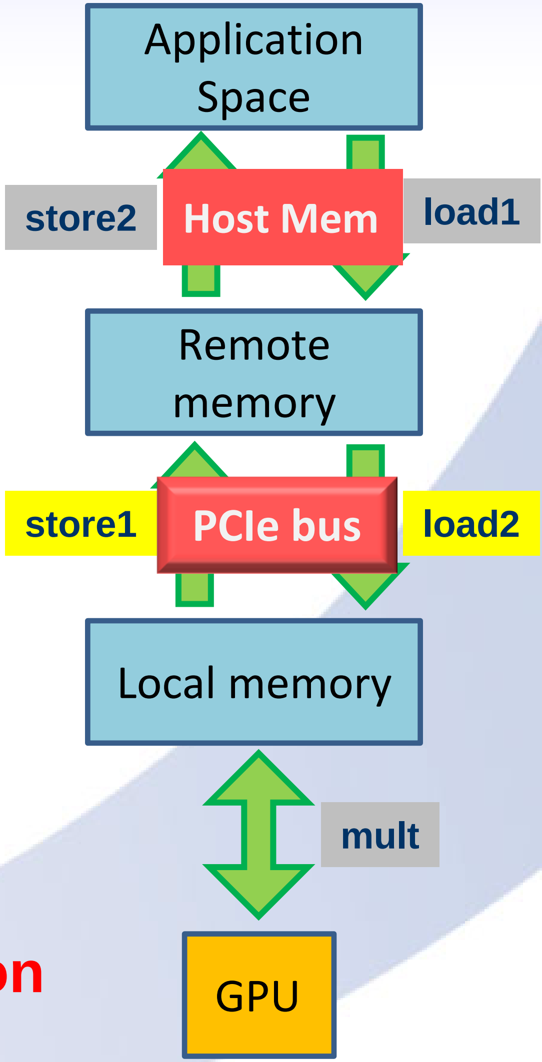

**Kernel performance: 94% (max)**

**Efficiency down, due to contention.**

# Contention on PCIe bus

## 1 GPU chip V.S. 2 GPU chips



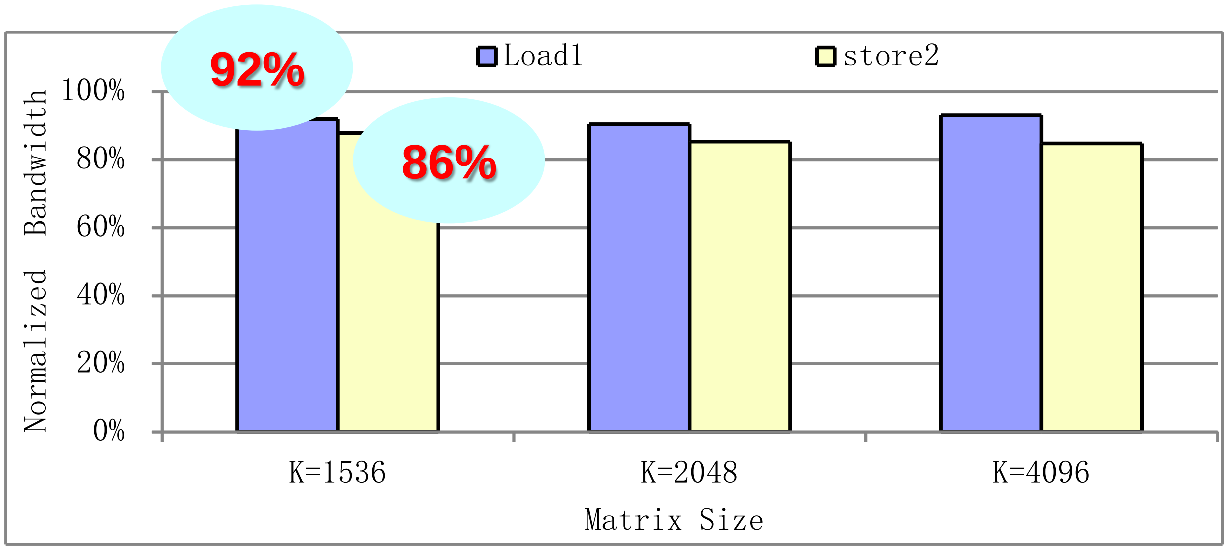

**Observation 1**

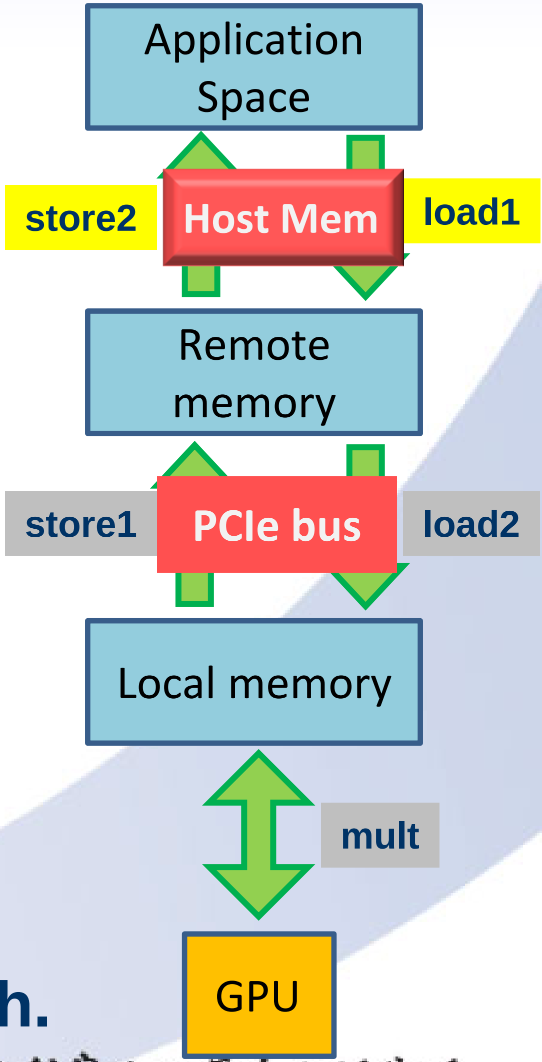
Contention on PCIe bus is an un-trivial limitation on multiple GPUs with restricted number of lanes.

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

# Contention on host memory (1)

## 1 GPU chip V.S. 2 GPU chips



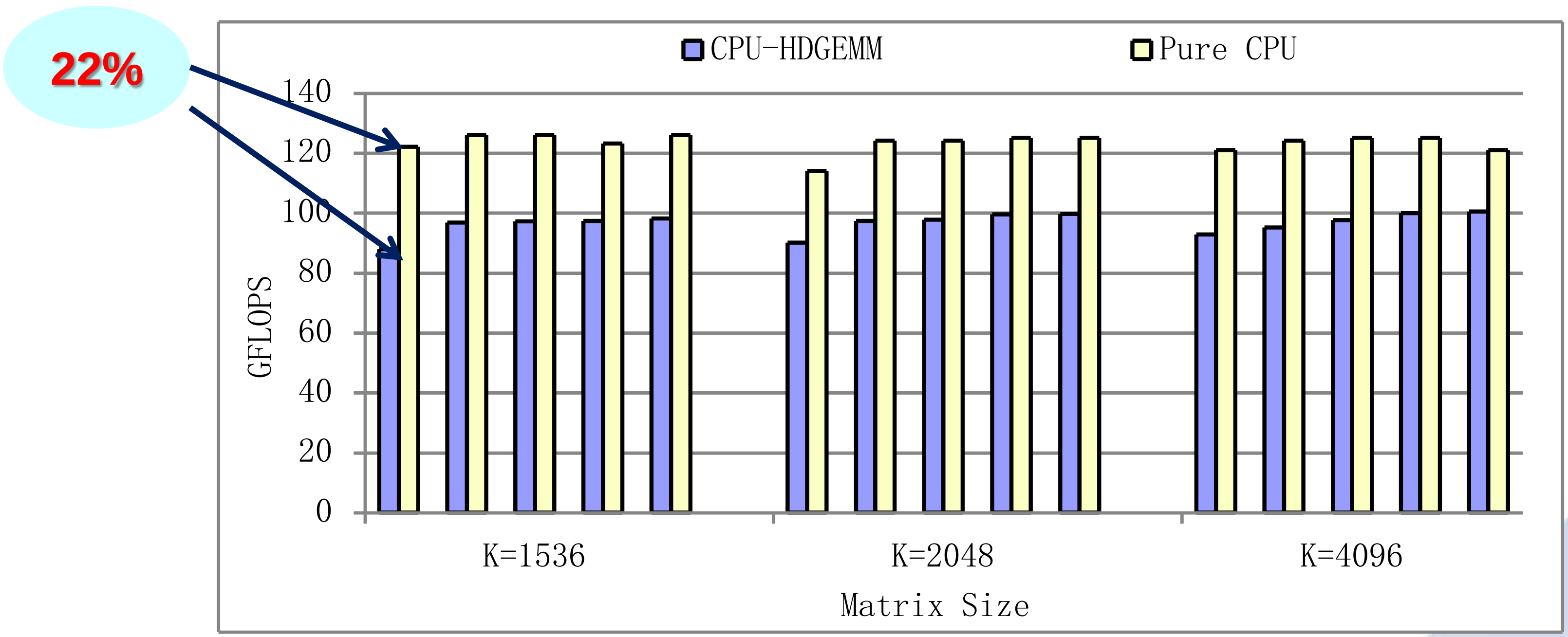




**Observation 2**

**DGEMM on multiple GPUs will not benefit much by improving host memory bandwidth.**

# Contention on host memory (2)

## Hybrid DGEMM V.S. CPU-only DGEMM




**Observation 3**

**Host memory bandwidth is important to Hybrid DGEMM.**

# Conclusion

- DGEMM optimization
  - Image addressing mode
  - Five-stage pipelining
  - Good Performance!
- Three observations
  - PCIe bus contention
  - Host memory contention
  - Give the reference for programmers and hardware designers

# Thanks!

# Questions?

http://asl.ncic.ac.cn/projects/dgemm

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

# ATI V.S. NV – Peak Performance

- **ATI HD7970 (Latest)**
  - **3.79 TFLOPS** Single Precision compute power
  - **947 GFLOPS** Double Precision compute power
- **NV Tesla K10 (Latest)**
  - **4.58 Gigaflops** Peak single precision floating point
  - **190 Gigaflops** Peak double precision floating point
- ATI still has higher performance for double precision.

# HD5970 V.S. Latest ATI GPU

- **ATI HD7970 (Latest)**
  - **3.79 TFLOPS** Single Precision compute power
  - **947 GFLOPS** Double Precision compute power
- **HD5970**
  - **4.64 TFLOPS** Single Precision compute power
  - **928 GFLOPS** Double Precision compute power
- HD5970 performance is acceptable.

# New AMD Math Library -- APPML

- APPML
  - Base on OpenCL
  - Provide GPU-only DGEMM kernel
  - Our kernel achieves higher performance than it.
- While our DGEMM
  - Run on heterogeneous CPU-GPU system

# Compared to MAGMA

- MAGMA
  - Base on NVIDIA GPUs
  - The memory hierarchy is different from ATI CAL.
- While our DGEMM
  - Base on ATI GPUs, is a complement for it.