Application of GPU to Three Computational Models

Qiangqiang Shi[1,*], Yiyang Yang[1], Xiaolin Li[1]

[1] *Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY 11794–3600, United States of America.*

Abstract. In this paper, we introduced the application of Graphics Processing Unit (GPU)-based algorithms for high performance computation of mathematical models in FronTier++. In the first case, the one-dimensional gas dynamics problem is solved by Weighed Essentially Non-Oscillatory (WENO) scheme, we achieved 7-20x speedup for different mesh sizes on 1 GPU. In the second case, the spring model for fabric dynamics is studied, the GPU code is about 10-20 times faster than the non-GPU code for different mesh sizes. In the last case, a GPU enhanced numerical algorithm for American option pricing under generalized hyperbolic distribution was studied. Using one GPU, we have achieved 2x speedup for the price of single option and 100x speedup for multiple options.

Key words: GPGPU, gas dynamics, spring model, American option pricing

1 Introduction

1.1 Introduction to GPU computing

Graphics Processing Units (GPU) computing [1] is the use of a GPUs together with CPUs to accelerate general-purpose scientific and engineering applications. Joint CPU/GPU application is a powerful combination because CPUs consist of a few cores optimized for serial processing, while GPUs consist of thoudands of smaller, more efficient cores designed for parallel performance. GPU computing offers unprecedented application performance by offloading compute-intensive portions of the application to the GPU, while the remainder of the code still runs on CPU. Fig 1 demonstrates the structure of hybird GPU/CPU code. From the viewpoint of a user, applications simply run remarkably faster.

*Corresponding author. *Email addresses:* qqshi@ams.sunysb.edu (Qiangqiang Shi), yiyang@ams.sunysb.edu (Yiyang Yang), linli@ams.sunysb.edu (Xiaolin Li)

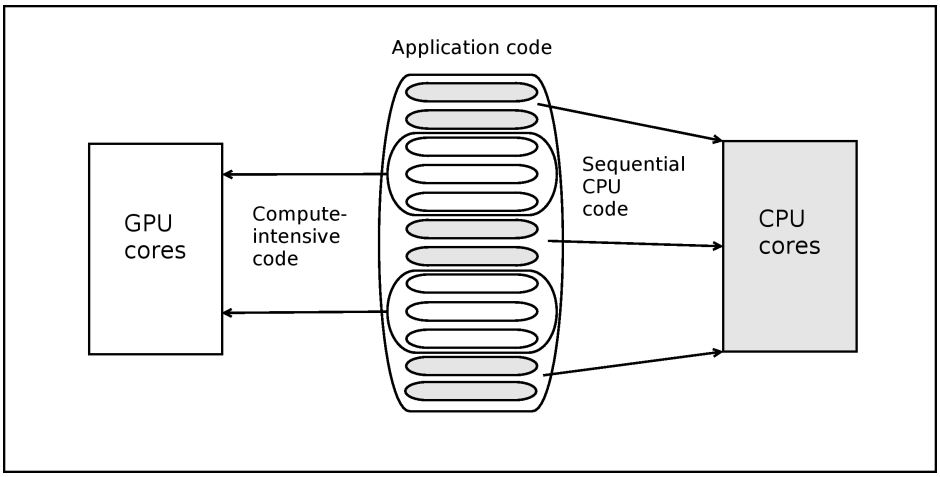

Figure 1: GPU acceleration code stucture

1.2 Related research

GPU computing has become a more and more popular research topic in the past few years and voluminous studies have found that the speedups GPUs delivered are substantial compared to CPUs.

1.3 Experiment platform

Both the CPU and the GPU studies were implemented on a dell precision T7600 Workstation with dual Intel Xeon E5-2687W CPUs and dual NVIDIA Quadro 6000 graphics cards. The Intel Xeon E5-2687W CPU is the latest multi-threaded multi-core Intel-Architecture processor. It offers eight cores on the same die running at 3.10 GHz. The Intel Xeon E5-2687W processor cores feature an out-of-order super-scalar microarchitecture, with newly added 2-way hyper-threading. In addition to scalar units, it also has 4-wide SIMD units that support a wide range of SIMD instuctions. Each has a separate 32KB L1 cache for both instructions and data and a 256 KB unified L2 data cache. All eight cores share an 20 MB L3 data cache. The Intel Xeon E5-2687W processor also features an on-die memory controller that connects to four channels of DDR memory. Each Quadro 6000 graphics card consists of 14 streaming multiprocessors (SMs) running at 1.15 GHz that share a single 768 KB L2 cache and 6 GB global memory on the device. Each SM consists of 32 streaming processors (SPs), a 48 KB shared memory and 32768 32-bit registers. Fedrora 18 with kernel 3.9.2-200, CUDA Toolkit 5.0 and GCC 4.7.2 were used in the computations. Table 1 shows the hardware structure of our computer. Fig 2 illustrates the architecture of the computational platform and Fig 3 demonstrates the architecture of multi-GPU devices.

Table 1: A Dell Precision T7600 Workstation with dual NVIDIA Quadro 6000 graphics cards was used to set up the test environmet.

Hardware	CPU	Dual Eight Core XEON E5-2687W, 3.1GHz 64GB DDR3 32KB x 16 L1 Cache, 256KB x 16 L2 Cache, 20MB x 2 L3 Cache
	GPU	Dual Quadro 6000 with 14 multiprocessor, 448 cores, 1.15Hz 6GB global memory, 64 KB constant memory 48KB shared memory and 32768 registers per multiprocessor
Software	OS Compiler CUDA	Fedora 18 with kernel 3.9.2-200.fc18.x86_64 gcc version 4.7.2 CUDA Toolkit 5.0

2 GPU application to gas dynamics

3 GPU application to spring model

4 GPU application to American option pricing

5 Numerical results

5.1 Gas dynamics results

5.2 Spring model results

5.3 American option pricing results

6 Conclusion

7 Acknowledgement

References

[1] David B Kirk and W Hwu Wen-mei. *Programming massively parallel processors: a hands-on approach*. Morgan Kaufmann, 2010.

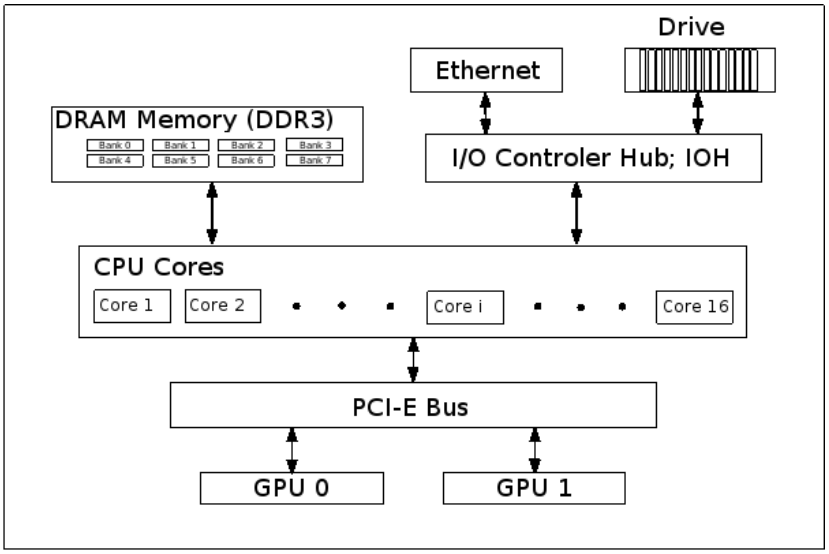

Figure 2: Architecture of the experiment platform. The Host part has 16 hyper-threading supported CPU cores and 64GB DRAM. The Device part consists of double NVIDIA Quadro 6000 graphics cards. The host part and the device part were connected by the PCIe Bus.

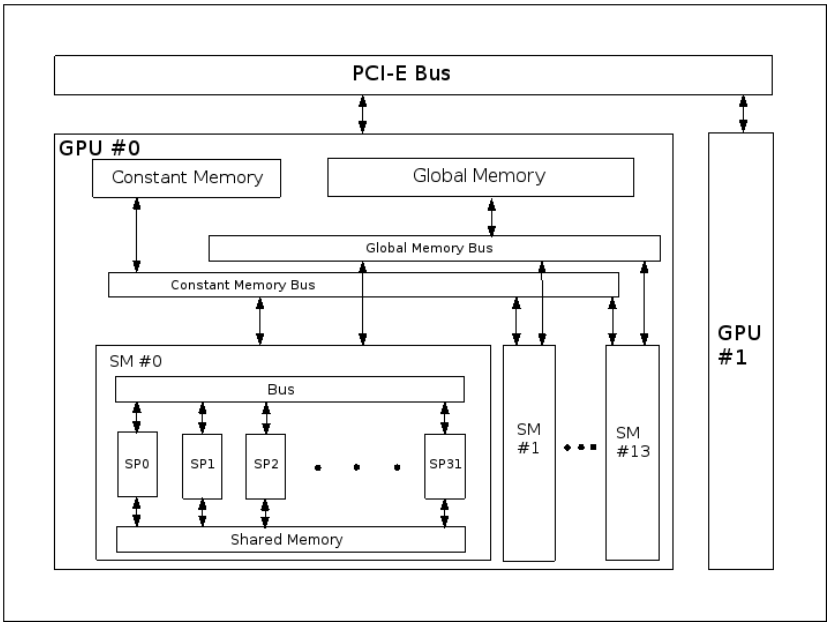

Figure 3: Architecture of multi-GPU device. Each GPU hardware consists of memory (global, constant, shared) and 14 SMs. Each SM consists of 32 SPs and can run 1536 threads simultaneously.