Comparison and Analysis of GPGPU and Parallel Computing on Multi-Core CPU

Hong Zhang, Da-Fang Zhang, and Xia-An Bi

***Abstract*—There are two ways to improve the performance of the algorithm computing, which are general purpose of computation and parallel computation of multi-core CPU. By comparison and analysis, contrast the main difference between them, we reach a conclusion that GPU is suitable for processing large-scale data-parallel load of high-density computing but relatively simple branching logic , however, the CPU is more suitable for processing complex logic computation. Now, the appearance of the CUDA makes GPU architecture more suitable for general purpose of computation. Cryptographic algorithm is typical compute-intensive algorithm, this paper take the modular exponentiation of RSA algorithms for example, through the comparison and analysis of GPU implementation and CPU implementation, the experiment results show: that the GPU implementation can achieve more than 45 times speedup in comparison with multi-core CPU implementation of RSA.**

***Index Terms*—GPU, GPU multi-core, CUDA, RSA, cryptographic algorithm.**

I. Introduction

Traditional CPU parallel levels only allow instruction-level parallelism, can only improve the performance of the CPU through improving working efficiency of processor. A simulation test of U.S. Sandia national laboratory shows, under the traditional architecture, due to the limitations of storage mechanism and memory bandwidth, the processors that above 16 nuclear can't bring performance improvements for super computer, might even lead to efficiency dropped substantially [1]. Intel, AMD and other manufacturers turn to focus on improving CPU architecture, on the single chip integrate more processor core, make CPU turn to the development direction of Multi-Core [2].

At the same time, the GPU technology is also in constant development. The birth and development of the GPU technology, can be divided into Ex-GPU era, fixed function pipeline era, programmable shader pipeline era and unified programmable shaders era. In November 2006, the NVIDIA company released G80-parallel-programming-model and C-language-based development environment-CUDA (Compute Unified Device Architecture) [3].

This article will take the architecture and parallel level as the main line to discuss the difference between the GPU general-purpose computing and CPU multi-core parallel.

Manuscript received March 10, 2012; revised April 20, 2012.

The authors are with School of information Science and Engineering Hunan University, China (e-mail: zhanghonghaiqin@163.com).

II. Related Work

A. GPU Architecture

GPU architecture in the continuous development, choose NVIDIA Tesla GT200 architecture as a representative to the GPU parallel levels for analysis [4]. Tesla GT200 were composed of two parts, Scalable Streaming Array

(SPA) and the memory system, they are connected by an on-chip interconnect network. As shown in Fig 1, the scalable stream processor array is formed by a number of Thread Processing Cluster(SM).

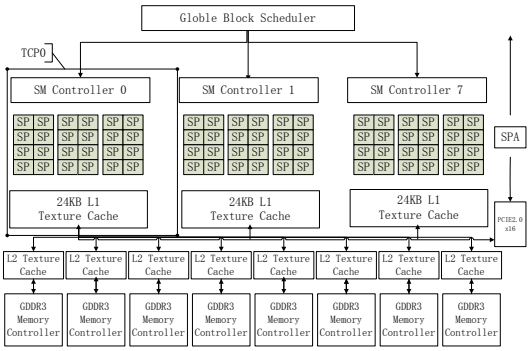
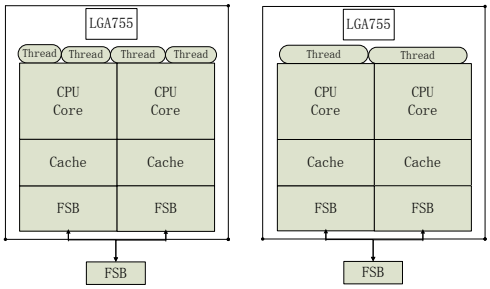


Fig. 1. GT200 architecture

B. Multi-Core CPU Architecture

By integrating multiple cores on one processor, the CPU can support thread level / process level parallelism [5]. Generally, a CPU core can run only one thread / process at the same time (Fig. 2 (b) mode), However, through threads technology, you can support a CPU core run two Hyper-Threading at the same time (Fig. 2 (a) mode).



(a)Support hyper-threading (b)Don't support hyper-threading

Fig. 2. Schematic diagram of dual-core CPU architecture

C. The Difference between Multi-Core CPU and GPU

Due to the GPU and CPU in the design goals of different, lead to both in architecture, parallel levels and performance of the large difference. GPU thread is fine-grained thread in the hardware management, while the CPU thread is coarse-grained heavy thread in the software management;

Each streaming multiprocessor of GPU is seen more like a single core of CPU, and the capacity of single-precision floating-point processing has reached the same period ten times as much as CPU; CPU memory controller is usually based on three-channel or dual-channel technology, while the GPU has several memory control unit; GPU is suitable for processing the calculation of high density, simple branching logic and large-scale data parallel load. While the CPU is suitable for complex logic operations [6].

III. Parallel Implementation of RSA Algorithm Based on CUDA

A. CUDA Parallel Computing Architecture

CUDA has brought a new dawn for GPU-based general purpose computation.

In the CUDA computing architecture, the implementation of procedures is organized in the form of grid, the smallest unit of execution is thread, by a number of threads to form a thread block, thread blocks execute the same program can be composed of grid blocks. Thread in each thread block can access the same piece of shared memory, and can proceed synchronous operation quickly [7]. Specifically, by calling the built-in function syncthreads() to specified synchronization in the thread grid. Fig 3 shows the Thread-level architecture of CUDA [8].

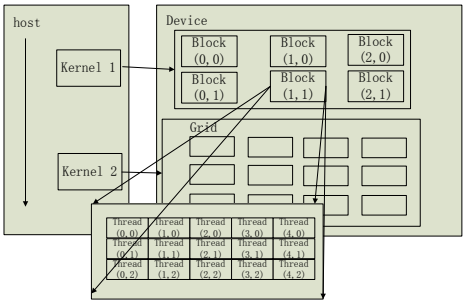


Fig. 3. CUDA thread-level architecture

B. Parallel Implementation of RSA Algorithm Based on CUDA

Cryptographic algorithm is typical of compute-intensive algorithms[9]. Take RSA as an example, give the RSA algorithm implementations on the GPU.

In the GPU implementation of RSA algorithm, y, n, s and the intermediate results q and a, are frequently used during the process of threads execution, so they will be stored in shared memory, in order to reduce global memory access, which can significantly improve the computational performance.

Thread mapping scheme is as follows: Each thread can independently undertake a modular exponentiation. Each thread performs the same control flow, by considering the allocation strategy of memory, further optimize the algorithm performance. Thread mapping scheme described in the following:

1) thread _index =threadId .x +(blockId .x *blockDim. x);
2) state =pt [thread _index];
3) ct [thread _index]=state.

The blockDim represents the number of CUDA thread block in the CUDA grid, blockId represents the index number of thread block which the current CUDA thread in, threadId represents the index number of current thread in current CUDA thread block, thread_index signify the index number of the current thread in the global thread environment, state sign signify the data block that is being handled by the current thread.

IV. Performance Evaluation

A. Experiment Settings

In order to compare the performance of GPU implementation with CPU implementation of RSA algorithm, the experimental Settings as follows:

1) CPU:AMD Athlon 64 Processor 3800+2.4 GHz;
2) Memory: 2.0GB;
3) GPU: NVIDIA GeForce 9800 GTX+, CUDA version is 2.1.
4) System: Windows XP sp3.

Accordance with the above implementation, complete the GPU implementation of RSA algorithm. In the NVIDIA GeForce 9800 GTX +, measured the throughput computing performance of the RSA algorithm, the results shown in Table 1.

TABLE I: Modular Exponentiation of Large Integer Of GPU Implementation

	512bits	1024bits	2048bits
1 operation/s	0.0230	0.1122	0.4610
128 operations/s	0.0562	0.2830	1.0352
1024 operations/s	0.1232	0.6020	2.0323
2048 operations/s	0.2520	1.1800	4.2130
4096 operations/s	0.5102	2.4600	8.3010

B. Performance Comparison

When performing 1024 group of concurrent large integer modular exponentiation operations, both GPU and CPU to achieve the performance comparison shown in Table 2 and Table 3.

TABLE II: Modular Exponentiation of Large Integer of CPU Implementation

	512bits	1024bits	2048bits
1 operation/s	1.0580	4.6002	21.667
128 operations/s	2.5852	9.7651	47.743
1024 operations/s	5.6500	24.820	94.250
2048 operations/s	11.466	49.560	202.10
4096 operations/s	24.990	103.88	398.45

TABLE III: Performance Comparison of the CPU And GPU Implementation

	512bits	1024bits	2048bits
CPU Computing performance/s	5.6500	24.820	94.250
GPU Computing performance/s	0.1232	0.6020	2.0323
Speedup	45.8600	41.2300	47.3600

Experimental results show that in large-scale data

concurrency of the key operation in the RSA algorithm, GPU implementation compared with the CPU implementation, the ratio is 40 times more.

V. Summaries

The ideal of general purpose computation on GPU used mode is: with CPU to control the main process, through the decomposition of the problem, handle the compute-intensive tasks that need massively parallel processing on the GPU to deal with. Compared with the CPU multi-core computing, its advantage is that high performance price ratio, low power dissipation, good portability, direct visualization [10]–[12]. GPU-based general-purpose computing research has played a linking role, Academia and industry are all of great deal of enthusiasm to the GPU-based parallel high performance algorithm.

Acknowledgements

This work is supported by the National Science Foundation of China under Grant No.61173167.

References

[1] J. Gazolla and J. Delgado *et al.*, "An Incremental Approach to Porting Complex Scientific Applications to GPU/CUDA," *IEEE Trans. on Neural Networks*, vol. 4, pp. 570-578, July 2010.

[2] Y. Sun, Y. Tong, and Z. Wang, "CUDA based high performance implementation of RSA algorithm," *Computer Engineering and Applications*, vol. 47, no. 2, pp. 84-87, 98. 11 Jan 2011.

[3] K. Jang and S. Han *et al.*, "Accelerating SSL with GPUs," *ACM SIGCOMM Computer Communication Review*, vol. 41, issue 1, January 2011.

[4] J. N. William and J. Dally, "The GPU Computing Era," *IEEE Trans. Image Process*, vol. 10, no. 5, pp. 767-782, May 2010.

[5] K. Korotaev, "Hierarchical CPU Schedulers for Multiprocessor Systems,Fair CPU Scheduling and Processes Isolation," *IEEE Trans. Electron Devices*, vol. ED-11, pp. 34-39,Jan 2005.

[6] C. Mei, H. Jiang, and J. Jenness, "CUDA-based AES Parallelization with Fine-Tuned GPU Memory Utilization," *IEEE Trans. on Neural Networks*, vol. 4, pp. 102-106, July 2010.

[7] C. Dai and J. Yang, "Research on Orthorectification of Remote Sensing Images Using GPU-CPU Cooperative Processing," *IEEE Trans. Computer Engineering*, vol. 4, pp. 223-226,Apr 2011.

[8] H. Patel, "GPU Accelerated Real Time Polarimetric Image Processing through the use of CUDA," *IEEE Trans. Applications*, vol. 4, pp. 48-55, Mar 2010.

[9] D. Hara and Y. Nakayama, "Secure and High-performance Web Server System for Shared Hosting Servic," *IEEE Trans, IEEE Trans. Electron Devices*, vol. ED-11, pp. 156-160, Jan 2005.

[10] Q. Hou, X. Sun, and K. Zhou *et al.*, "Memery-Scalable GPU Spatial Hierarchy Construction," *IEEE Trans.Computer Engineering*, vol. 4, pp. 189-193, Apr 2011.

[11] F. Cui and C. Cheng *et al.*, "Accelerated GPU Computing Technology for Parallel Management Systems," *IEEE Trans. Image Process*, vol. 10, no. 5, pp. 255-259, May 2010.

[12] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU Computing," *IEEE Trans. Neural Networks*, vol. 5, pp. 334-339, Oct 2010.