A Stream Processor Cluster Architecture Model with the Hybrid Technology of MPI and CUDA

Qing-kui Chen, Jia-kang Zhang
School of Optical-Electrical and Computer Engineering
University of Shanghai for Science and Technology
Shanghai, P.R.C
chenqingk@tom.com, shinson@126.com

***Abstract*—Nowadays, the compute capability of traditional cluster system can't keep up with the computing needs of a practical application, and these aspects of energy, space technology, etc. have become a huge problem. However, as parallel computing equipment, the stream processor (SP) has a high performance of floating-point operations. NVIDIA GPUs is a typical stream processor device, CUDA technology enables the way to develop a better parallel program on GPUs to become flexible. In this paper, we make use of the hybrid parallel computing programming environment (HPCPE) with MPI and CUDA technology to build the simple CPU + GPU-based stream processor cluster system. In addition, we also proposed the "Two Level Model (TLM)" to separate the intensive computing tasks and controlling tasks, and exploit the compute capability of contemporary GPUs to accelerate computing tasks. Finally, we conducted a relevant experiment about the calculation of N-Body problem, and verified the better performance that stream processor cluster system has than the traditional one.**

***Keywords*-SP cluster system, MPI, CUDA, HPCPE, TLM, N-Body**

I. INTRODUCTION

As we all know, in the field of high-performance computing, the traditional CPU-based cluster system has been in a dominant position. In November 2008, the statistics of "Top 500 Supercomputers" show that more of 90% cluster systems are composed of CPUs as core components for calculation, such as IBM's "Roadrunner", Blue Gene/L Supercomputer, Dawn 5000A in China, etc. However, because of these restrictions to many problems such as compute capability of multi-core CPU, system energy consumption, space technology, etc., they hindered the development of the traditional cluster system.

In recent years, with the rapid development of the core technologies of stream processors, stream process, stream compute create a new era for the field of parallel compute. GPU as a typical stream processor equipment has higher floating-point compute capability than the CPU. The "Table Super PC" built with GPU as core hold more prominent compute capability than the small cluster. If these super PC is built a PC cluster system, it will be a tremendous impact in the case of the traditional CPU-based clusters.

Therefore, in this paper, we proposed a CPU+GPU-based streaming processor cluster structure model, and make use of the hybrid programming technologies with Message Passing Interface (MPI) [9] and Compute Unified Device Architecture (CUDA) [4] to build a parallel computing environment so as to try to open a breakthrough in the field of parallel compute for traditional high-performance computing cluster system.

II. STREAM PROCESSOR CLUSTER SYSTEM

Definition 1. Stream Processor Cluster System. Using both CPU and GPU to build a cluster system, GPU for each node is used to take on the core computing tasks, and CPU as a coprocessor is in this system. Unlike the traditional x86 architecture cluster system, GPU for each node is based on a unified stream computing architecture (GPU computing) [2] [5].

A. *SP Cluster Architecture Model*

Cluster is the collection of computer resources. By Definition 1, on the way that GPU is as the core, CPU collaborates with GPU to processing data, control program, we build a SP cluster architecture model, and it is a new generation cluster system that is different from the typical traditional cluster architecture. This system architecture is shown in Figure 1.

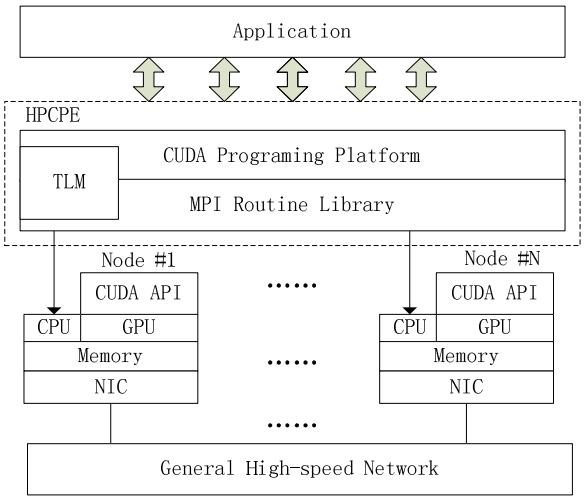

Figure 1. SP cluster architecture model based on CPU+GPU

The work of Qingkui Chen and Jiakang Zhang was supported by National Nature Science Foundation of China (No.60573108), the Innovation Program of Shanghai Municipal Education Commission (No. 08ZZ76, 07ZZ92), and Shanghai leading academic discipline project (S30501).

Programming environment and the hardware structure of cluster node is very different from traditional one. First, it is hybrid architecture model based on the CPU and GPU. Unlike traditional cluster system, this model separate process control tasks from data computing tasks. Secondly, this complement of separation depends on HPCPE based on MPI and CUDA.

SP cluster architecture model is described as follows:

- Developers develop some application with HPCPE (Dashed Border) based on MPI and CUDA. HPCPE is based on TLM, which corresponds to top-level of the traditional cluster architecture [7].
- The hardware structure on each node differs from the traditional cluster node. GPU will be added into the cluster node. In the upper level, i.e. HPCPE, Intensive computing tasks will be taken on by GPU with CUDA. However, adopt MPI Process to control program, data migration and other adjective tasks on CPU. This kind of program execution model is called "Two Level Model", i.e. TLM in Figure 1.
- Every node connects to each other through a general high-speed network.

B. *Traditional Cluster Node and SP Cluster Node*

Mainly, CPU and a group of memory constitute a traditional cluster node, and in the structure of SP cluster nodes, GPU is regarded as a core computing components, it has its own independent memory system. GPU as a collection of SPs is based on stream computing model [1] [3]. Stream computing model is a new framework with SP devices as the carrier.

SP can be regarded as the specific realization of stream computing model [1]. NVIDIA GPUs are the most typical SP devices, and have a high floating-point operations power. GPU provides a more powerful than CPU in the field of large-scale parallel computing. Generally consider the availability and compute capability of cluster system, using cheap NVIDIA GeForce GPUs is enough to build SP cluster system. Of course, the GPU products of the same architecture in the NVIDIA products, such as Tesla also removed a series of the graphics interface, but it's very expensive. On the contrary, its compute capability gets high upgrade, and it is dedicated to scientific computing.

Some characteristics of SP cluster node can be summarized into the following four points:

a) Strong compute capability: For an example, NVIDIA's GTX280's theoretical peak of floating-point operations is 933GFlops, and double-precision computing is about 90GFlops.

b) Large memory capacity, multi-level of memory structure, independence, the interaction with the Host Memory may become a bottleneck: The latest NVIDIA Tesla S1070 has 16GB memory, NVIDIA GPUs have adopted three memory hierarchy [4] , which expands memory architecture of the traditional cluster node, data exchange model between nodes will change. Due to the independence of GPU memory, the device memory on GPU interacts to the host memory on CPU through PCI 16X bus, and its maximum transfer rate is only 4GB/s. With the increase in compute capability, interaction of device memory and host memory will become frequent.

c) The complex programming environment: Cluster nodes' communication, data transmission will be supported by MPI library or other routine library, but computing tasks of node need be supported by the CUDA platform. Mixing two types of programming environments to a whole one, but no conflict, that is a key problem developers need to consider.

d) The performance of the network may become a bottleneck: The quality of network performance is an important indicator of clusters. Similarly, because SP has a strong compute capability, so the cluster system requires much more amount of data than the traditional one; That single-node computing speed enhancement means that the data exchange will short the time interval, the frequency of data collection and distribution increases.

C. *HPCPE based on MPI+CUDA*

All in the current message passing software, the most important and popular is MPI [8] [9]. MPI is a message passing library standard based on the consensus of the MPI Forum. However, as to GPU in SP cluster on each node, MPI does not adapt to it and cannot program on GPU with MPI, so now we change the task object of MPI to program control, and assign CUDA the computing tasks. Figure 2 show the hybrid programming collaborative process based on MPI and CUDA in single-node, i.e. TLM.

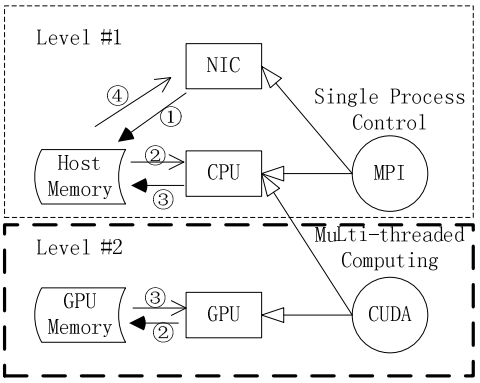

Figure 2. The hybrid programming collaborative process of TLM

The number of processes in single-node launched by MPI depends on the number of GPUs in single-node. We assume that a node is with a GPU, so each node launched single MPI process. Four digits in Figure 2 indicate that the following four processes:

① *Data are collected.*

② *Host Memory dump data on GPU Memory to perform compute.*

③ *Finish computing tasks and return data from the GPU to the Host.*

④ *Nodes communicate for each other, and Data are distributed, broadcasted.*

HPCPE is based on TLM. From Figure 2, we can see that these two levels.

- In level 1, we use MPI to control program, the communication between nodes, data schedule, and interaction with CPU, rather than intensive computing tasks. Single node is assigned a CPU process.
- In level 2, CUDA technology will be used to build a parallel computing environment on single node, take on computing tasks and interact with CPU and GPU. CUDA create a large number of lightweight threads on GPU. From the threads' creation, management, implementation to destruction, GPU load above all. These threads execute intensive computing tasks in Single-Instruction Multiple-Data (SIMD).

In TLM, level 1 and 2 are interrelated and interact on each other. This design will due to such a powerful GPU compute capability.

1) CUDA

CUDA programming model can be seen in [4]. It drives GPU to work. It owns two standard mathematical libraries, CUFFT and CUBLAS. CUDA's program code is divided into two types [6]. One is called "Host Code" compiled on CPU, and other one is called "Device Code" compiled on GPU.

On CUDA, threads are organized through "Thread Block" [4]. A Thread Block can have up to 512 threads. Threads belong to the same Thread Block can share data through shared memory. Thread Blocks can re-organize into "Thread Grid" [4]. CUDA owns a specific index mechanism for data addressing. There is a set of index for Block and Thread respectively.

CUDA extends C by allowing the programmer to define C functions, called "kernels" [4], that, when called, are executed N times in parallel by N different CUDA threads, as opposed to only once like regular C functions.

2) The pattern of MPI communication based on CUDA

Because of the characteristics of GPU's hardware, CUDA programming model organize threads into Thread-Block, and a number of Thread-Blocks are re-organized into a Thread-Grid. Therefore, we used some corresponding data structures to build some special send/receive pools during MPI communication. We create Data-Grid (DG) and Data-Block (DB) used to organize data so as to build the send/receive pool. Figure 3 describes that the structures of these pools with Data-Grid and Data-Block in three kinds of cases.

Definition 2. $DG=\{DB\#1, DB\#2, \ldots, DB\#n\}$, $n \geqslant 1$; $DB=\{\cup \text{Data}\}$.

Figure 3 (a) describes the case for one-to-many communications. *DG* is broken down into *p DG*s, and each *DG* will be sent to a host.

Figure 3 (b) describes the case for many-to-one communications. Each node sends *DG* by itself. The main process receives these *DGs* from all sub-processes to merge.

Figure 3 (c) describes the case for many-to-many communications. All processes respectively broadcast and receive own *DG*.

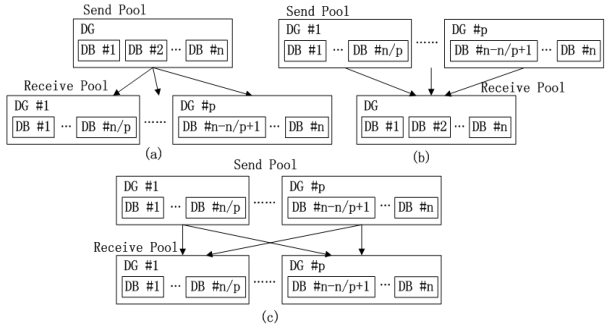

Figure 3. Send/Receive pools' structure of MPI communication based on CUDA

III. CLUSTER SYSTEM BUILDING

A. *Hardware Environment*

We use three PCs with NVIDIA GPU to build a SP cluster system and a traditional cluster. SP cluster: Each PC is equipped with a Intel Core2 E8400 3.0GHz CPU, 2GB 800MHz host memory, 500GB hard disk and an NVIDIA GTX280 graphics cards with 1GB global memory, and its core frequency is 602MHz, main frequency is 1296MHz. The version of GPU's compute capability is v1.3 [4], also support for double precision of IEEE754 standard. GTX280 has 30 multi-processor (MP) units, and 8 SP per MP, i.e. 240 stream processors.

B. *Software Environment*

Use HPCPE based on MPI and CUDA. Table I shows details as follows:

TABLE I. SOFTWARE CONFIGURATION

OS	Windows XP 32-bit SP2
Programming Platform	MPICH2 1.8.0 release
	CUDA 2.0 release

C. *System Performance*

High Performance LINPACK (HPL). The result of HPL is an important indicator for the TOP500 ranking. Unfortunately, the official also does not release LINPACK in the version of CUDA.

Therefore, we prepared a test program to replace HPL test. It is a program about N-Body problem by MPI and CUDA. N-Body problem involves many fields of science and engineer. Its major characteristic is that its algorithm complexity is $O(N^2)$. The N-Body program is based on particle Atom Decomposition Algorithm (ADA) [10]. The basic idea of ADA is that all atoms are assigned to the processors on average. The number of atoms each processor got is random, and there is no correlation between these atoms in space. Each processor is only responsible for the "Force Calculating" of its N/P atoms assigned (N is the total number of atoms, P is the number of processors), then updates their information of displacement and velocity.

TABLE II. EXECUTION CONFIGURATION ON CUDA

n	Kernel #1		Kernel #2	
	Grid	*Block*	*Grid*	*Block*
1024	（24，64）	（16，16）	1	384
2048	（48，128）	（16，16）	3	256
4096	（96，256）	（16，16）	6	256
8192	（192，512）	（16，16）	12	256
16384	（384，1024）	（16，16）	24	256

Table Ⅱ gives us the dimensions of Grid and Block on CUDA, i.e. thread assignment configuration. Kernel #1 is "Force Calculation" kernel, Eq. (1) and Eq. (2) is used to configure the Grid.

$$GridDim.x = \frac{3}{2} \times \frac{n}{(numprocs + 1) \times BLOCK\_SIZE} \quad (1)$$

$$GridDim.y = \frac{n}{BLOCK\_SIZE} \quad (2)$$

n is the size of a problem, *numprocs* is the number of MPI processes, i.e. the number of all nodes (three) , BLOCK_SIZE is a constant. Block dimension is two dimensions, i.e. (*BLOCK_SIZE*, *BLOCK_SIZE*).

Kernel #2 is "Velocity and Displacement Calculation" kernel, Eq. (3) and Eq. (4) is used to configure the Grid.

$$GridDim.x = \frac{3}{2} \times \frac{n}{(numprocs + 1) \times 256} \quad (3)$$

$$GridDim.y = 1 \quad (4)$$

Eq. (3) and Eq. (4) are the same as Kernel #1. In addition to n = 1024, above equations are not applied, Block dimensions is 384. These data in Table Ⅱ are given by Eq. (1) ~ (4).

When the size of problem is *n*, the compute of FLOPS meets the following equations:

$$\text{Force Calculation: } 20n^2 \quad (5)$$

$$\text{Velocity and Displacement Calculation: } 2n^2+14n \quad (6)$$

Eq. (5) plus Eq. (6) is the total number of floating-point operations. Then 1000 iteration, as long as time T is measured, the peak of FLOPS is:

$$1000 \times (22n^2+14n) \ / \ T \quad (7)$$

Figure 4 is test results of N-Body on different cluster systems. In traditional cluster system, we make experiments with 3 and 6 process respectively. From Figure 4 we can see that FLOPS is increasing with the increase of n, and the performance of SP cluster is always higher than traditional cluster. Moreover, it is obvious for the increase rate of SP cluster. The 6 process traditional cluster is the double performance of the 3 process one. Compared with SP cluster, the peak FLOPS of SP cluster is about 7.8 times higher than the 3 process traditional cluster, and 4.2 times higher than the 6 process traditional cluster.

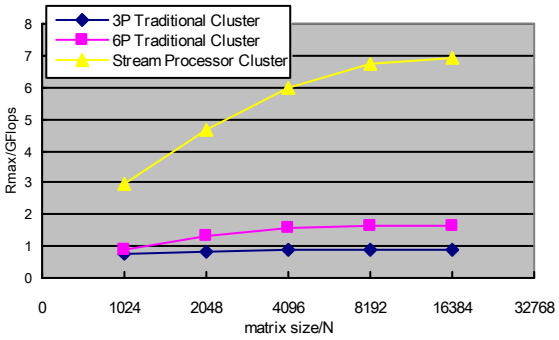

Figure 4. Test Results of N-Body on different Cluster Systems

IV. CONCLUSIONS AND FUTURE WORK

The requirement for computing capability has become more and more giant in some practical science problems. In this paper, we combined CPU with GPU, owned a integration of multi-programming environment based on MPI and CUDA, built a SP cluster system and took advantage of TLM to separate computing tasks from scheduling tasks. On small-scale SP cluster system, we get the higher performance than the same scale of the traditional one. As to the traditional cluster, the SP cluster system is an innovation, an impact and a challenge.

Our future work includes how to build a SP cluster system that own a good scalability and further excavate the compute capability of SP in the basic of better HPCPE.

REFERENCES

[1] Ying Zhang, Xuejun Yang and Guibin Wang, "Scientific Computing Applications on a Stream Processor," in Proceedings of IEEE ISPASS, Austin, Texas, April 2008.

[2] John D. Owens, Mike Houston, David Luebke, Simon Green,John E. Stone, and James C. Phillips, "GPU Computing" in Proceedings of the IEEE, vol. 96, no. 5, pp. 879-889, May 2008.

[3] Jayanth Gummaraju, Mendel Rosenblum, "Stream programming on general-purpose processors," in Proceedings of the 38th annual IEEE/ACM MICRO, Barcelona, Spain, November 2005.

[4] NVIDIA Corporation, "NVIDIA CUDA Compute Unified Device Architecture Programming Guide 2.0," see http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf, June 2008.

[5] Xiaowen Chu, Kaiyong Zhao, Mea Wang, "Massively Parallel Network Coding on GPUs," in Proceedings of IEEE IPCCC, Austin, Texas, USA, December 2008.

[6] David Luebke, "CUDA: Scalable parallel programming for high-performance scientific computing," in Proceedings of the 5th IEEE ISBI, Paris, France, May 2008.

[7] Xiong Sheng-wu, Wang Lu, Yang Jie, "Key Technologies Used For Construct High-performance Computer Cluster System," Micro-computer Information, vol. 26, no. 13, pp. 86-88, 2006.

[8] The MPI Forum, "MPI: A message passing interface," in Proceedings of Supercomputing '93, Portland Oregon, November 1993.

[9] Message Passing Interface Forum Document for a Standard, "Message Passing Interface," TechRep:CS-94-230, University of Tennessee, 1994.

[10] WANG Xiao-Wei, GUO Li, YAN Zhang-Yuan, "N-body algorithms and parallelization of them," Computers and Applied Chemistry, vol. 20, no. 2, pp. 195-200, 2003.