Design and Test Based on Stream Processor Programmable Cluster Architecture

Xu Wen[1], Wu Dongyan[1], Qi Lijun[1], Wang Mingge[2], Xiao Jingxin[1]

[1]Aviation University of Air Force, Jilin, Changchun130022, China

[2]Jilin University, Jilin, Changchun130012, China

huner2011@foxmail.com

Keywords: stream processor, GPU programming model, CUDA technique, stream processor cluster

Abstract. As the representative of common programmable stream processor, the performance of GPU develops rapidly, which has broken Moore's law which is obeyed by CPU. GPU applies the performance of programmability and functional expansibility to support complicated computation and process, and the feature has been acknowledged in the industry. The paper deeply researches programmable model CUDA based on NVIDIA GPU, and analyzes CUDA technique. And the paper applies MPI+CUDA hybrid programming model and common stream processor to load the model, which not only separates the control of stream processor cluster from computation, and optimizes multi-data stream processing strategy, but also promotes the performance of stream processor cluster system compared with traditional x86 cluster system.

Introduction

With the maturity of stream system structure such as dedicated stream system structure represented by Imagine, common stream system structure represented by TRIP and RAW, and more complicated dedicated stream system structure represented by well-known CELL processor, and the development of stream process and stream computation, the common stream processor represented by GPU has attracted the attention of people with the characteristics of high popularity, popularized prices and great computation capacity. The computing technology based on GPU may replace clustering computing technology to become the mainstream technology with high-performance computation. It fully uses large-scale thread parallel processing capacity of GPU, and GPU can be used as common computation platform based on CPU to provide supplement for high-performance computation capability, which implements cost-effective and high-performance computation HPC solution based on the existing common computing platform.

Key to Making GPU Become high-performance Programmable Stream Processor

The development of GPGPU represented by NVIDIA GPU, and the emergence of Tesla system structure [8] makes GPU develop towards high-performance computation field. The present hardware architecture of GPU inherits traditional stream system structure, and the hardware architecture uses the design idea of stream system structure. Compared with the early GPU, the hardware architecture of new GPU changes greatly.

The first step to establish a high-performance GPU is to map the nucleus in graphics pipeline into independent function unit of single chip. So each nuclear which is implemented in different areas of chip is called task parallelism in the organization. Not only task parallelism allows parallelism of task level (because all nucleus are executed at the same time), but also functional units make hardware specialization on the given nucleus. The organization of task parallelism allows effective communication between nucleus, the reason for which is that the adjacent nucleus implemented by functional unit in graphics stream pipeline adjoin the chip, and they can effectively communicate under the condition without requiring global memory access. ‘

In each stage of graphics pipeline of processing units mapping to the chip, GPU applies the independence of flow elements by processing several data elements in parallel. The synchronous combination of task level and data level allows GPU to effectively use many function units at the same time.

The input of graphics pipeline must be processed by each nuclear orderly. It may require thousands of cycles to process an element. If a high-latency memory reference is required in processing elements, processing unit only needs to work for other elements when data is achieved. Therefore, the depth pipeline of GPU effectively tolerates high-latency operation.

MPI+CUDA Hybrid Programming Technique

Message passing interface, MPI. MPI is the standard specification of message passing function library. MPI is developed by MPI forum which is an alliance consisting of paralleled computer provider, library author and application specialist [26] [27]. MPI achieves the objective of portability by providing a standard method of non-patented message passing library which is independent of platform. MPI explains the library with the form without relation to language. And it provides the binding of Fortran and C. The standard doesn't include any features of suppliers, operation system or hardware. For the above reasons, MPI is widely accepted in computation filed. Meanwhile, MPI provides more than two hundreds of routines. The provided powerful functionality is based on four orthogonal notions, message data type, communication subsystem, communication operation and virtual topology. MPI indicates the combination can obey well defined semantic pattern.

Execution process of typical CUDA application. The program execution of CUDA has certain flow. The following is the implementation step of typical CUDA. (1) Distributing the occupation time of data in device. (2) Transmitting the data from Host to Device. (3) If there is need, the memory of Device is initialized. (4) Determining execution configuration according to the processed data. (5) Executing Kernel, storing the results on Device. (6) Outputting the result data from Device to Host.

MPI+CUDA hybrid programming model. The task objects of MPI are transferred to be controlled by program, and the computation task is delivered to CUDA. Figure 1 shows hybrid programming collaboration flow of single-node MPI and CUDA.

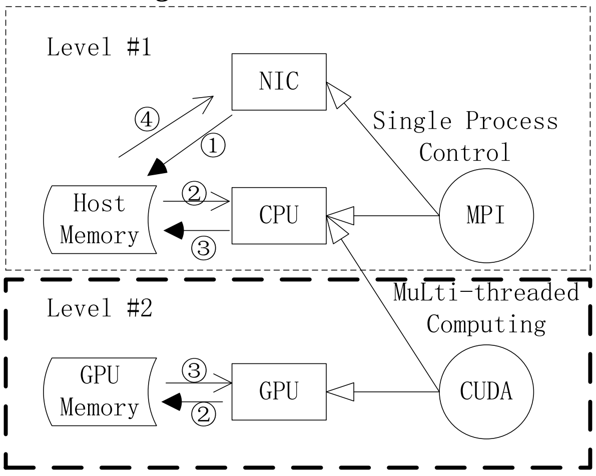

Figure 1 Single-node MPI+CUDA hybrid programming collaboration flow

The number of processes launched by MPI on each node is determined by the number of GPU carried on nodes. If a node carries a GPU, the number of processes launched by MPI on each node is 1. The number in Figure 1 means four processes.

Data collection.

(1)The data of Host Memory is restored into GPU Memory to execute computation.

(2)After computing the data, the data is recycled to Host from GPU.

(3)Node communication, data distribution and broadcast.

Although hierarchical mode independents control from computation, layer 1 is not completely independent of layer 2. On layer 2, the nodes are independent, and they independently compute the data which is distributed and collected by layer 1 on GPU. Then, the data results are returned to the main memory by GPU. The control process of GPU of layer 1 accepts the data, and makes collection and broadcast. The design owes to great computation capability of GPU.

Establishment of Stream Processor Cluster

Hardware environment. Three PC carrying NVIDIA GPU are used to compose a stream processor clustering system. Each PC is equipped with CPU of Intel Core2 E8400 3.0GHz, host memory 2GB, 800MHz, hard disk 500GB, NVIDIA GTX280 display card, and GT200 architecture. The computation capability of the device is version 1.3. It supports IEEE754 double-precision standard, and has 30 multinulear processor units, and 240 stream processors. Internet uses DLink 16-port switcher to compose a group of Gigabit LAN.

Software environment. The paralleled programming environment of MPI and CUDA is used, as shown in Table 2.

Table 2 Programming environment configuration information of nodes

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

Cluster performance test. LINPACK is the most internationally popular benchmark to test the floating-point performance of high-performance computer system. LINPACK is the linear algebra package which is written by Jack Dongarra in Tennessee University. The high-performance computer uses Gaussian elimination method to solve the test of unitary N dense linear algebraic equation set, to evaluate floating-point performance of high-performance computer. LINPACK test is divided into LINPACK100, LINPACK1000 and HPL, in which HPL（High Performance Linpack) is the important basis of TOP500 ranking. And the version of LINPACK in CUDA is not given officially. So the testing program which is written by ourselves has replaced HPL test. It is CUDA multi-machine program solving N-Body celestial body problem. N-Body problem involves many fields of science and engineer. The main characteristic is the calculated amount of O (N2). Applying paralleled computation method of stream processor cluster is one of ways solving huge computation amount of N-Body problem.

N-Body program is based on ADA(Atom Decomposition Algorithm)[28]. ADA averagely distributes all atoms to each processor. Before computation, each processor distributes atoms randomly, and there is no any relationship in space. Each processor only computes the stress of atoms in its N/P (N is the total number of atoms, and P is the number of processors), and it is used to update the displacement and speed information.

Table 3 Linear distribution of each node processor

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

By setting Grid dimension and Block dimension, Figure 2 shows linear distribution of each node stream processor. Kernel #1 means Kernel computing stress, and the computation formula of two-dimensional Grid is

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

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

In the formula, n is the size of problem scale, and numproc is the process of MPI, or node number, which is 3. BLOCK_SIZE is a constant, 16. The dimension value of Block is （BLOCK_SIZE,BLOCK_SIZE）. Kernel #2 means Kernel computing displacement, and the computation formula of one-dimensional Grid is

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

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

For the variable Kernel #1, except for n=1024, the dimension of Block of the other scales is fixed value, 256. When n=1024, the above formula is not applicable. Block dimension is 384. And the data in Table 2 is achieved by computing formula (1)-(4).

When the scale of the problem is n, the times of operating floating points should meet the following formulas.

Computing stress：$20n^2$ (5)

Computing speed and displacement：$2n^2+14n$ (6)

In the computation process, it is iterative for 1000 times, so the total floating point operation times is

$$1000\,(22n^2+14n) \quad (7)$$

Therefore, the problem scale n is given, the system operation time T is measured, peak=floating-point operation times /computation time I, and the testing result is Flops.

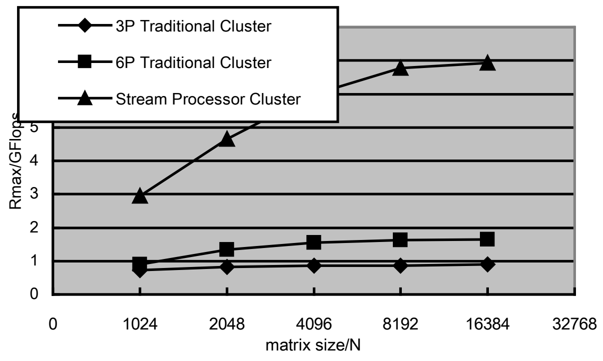

Figure 2 Testing results of N-Body on cluster

Figure 2 is the testing result of N-Body on cluster. 3-thread and 6-thread experiment is done on traditional cluster. From Figure 2, we can see that compared with traditional clustering system, with the increase of problem scale, the performance of cluster increases, and the maximum execution times of stream processor cluster is more evident. The performance of traditional cluster 6-thread is worse than that of 3-thread. Compared with stream processor, it is 7.8 times better than three-thread, and it is 4.2 times higher than 6-thread.

Conclusions

GPGPU depends on graphics API interface, which restricts the great parallel processing capability of GPU. CUDA technique provides direct access interface of hardware, which avoids the implementation of indirect GPGPU using API interface to access GPU.

The paper analyzes the design of MKSD stream processor cluster system, proposes hybrid scheme of MPI+CUDA programming model, and establishes a cluster example and traditional cluster to compare the performance. The experiment indicates that the performance of stream processor cluster is better than traditional cluster clustering system, and applying GPU as computation co-processor will become high-performance computation solution.

References

[1]NVIDIA Corporation. Whitepaper"NVIDIA's Next Generation CUDA Compute Architecture:Fermi"[EB/OL]. http://www.nvidia.com. 2009-11-1.

[2] Jing Du, Xuejun Yang, Guibin Wang. Scientific Computing Applications on the Imagine Stream Processor[C]. Proc. of the 11th ACSAC. Shanghai, China, Sep. 2006.

[3] David Luebke. CUDA: Scalable Parallel Programming for High-performance Scientific Computing[C]. Proc. of the 5th IEEE Int’l Symp. on Biomedical Imaging: From Nano to Macro. Paris, France, 2008, :836-838.

[4] Jayanth Gummaraju, Mendel Rosenblum, "Stream Programming on General-Purpose Processors,"[C] . Proc. of the 38th annual IEEE/ACM MICRO, Barcelona, Spain, Nov. 2005.

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

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

[7] Message Passing Interface Forum Document for a Standard, “Message Passing Interface,” [J]. TechRep:CS-94-230, University of Tennessee, 1994.

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

[9] Xiong Sheng-wu, Wang Lu, Yang Jie. Key Technologies Used For Construct High-performance Computer Cluster System[J]. Micro-computer Information, 2006, 26(13):86-88.