The multi-GPU System with ExpEther

Shimpei Nomura[1], **Tetsuya Nakahama**[1], **Junichi Higuchi**[2]
Jun Suzuki[2], **Takashi Yoshikawa**[2] **and Hideharu Amano**[1]
[1]Graduate School of Science and Technology, Keio University, 3-14-1 Hiyoshi Kouhoku-ku
Yokohama, Kanagawa 223-8522, Japan
[2]System Platforms Research Laboratories, NEC Corporation 1753
Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa 211-8666, Japan
Email: cell@am.ics.keio.ac.jp

Abstract— *Clusters using multiple GPUs have been already widespread to build a high performance computer economically. However, since the number of plugged GPUs into a CPU is limited, such clusters are consisting of multiple host PCs each of which has a few GPUs. This conventional multi-GPU cluster requires programmers to learn parallel programming skills for controlling communication between nodes as well as GPU programming.*

In order to show the illusion that a large number of GPUs to a single host, a multi-GPU system with ExpEther is proposed. The multi-GPU system allows interconnecting a single host PC and multiple GPUs by ExpEther which extends PCIe interface to Ethernet.

Execution of the application program with two to six GPUs achieved 1.99, 2.96, 3.92, 4.83 and 5.14 times speedup at most, as those with a single GPU. Also, the influence of the bandwidth of the network used in the multi-GPU system is evaluated quantatively.

Keywords: Graphics processing unit, cluster, Parallel Computing, Scalability

1. Introduction

The GPGPU (General Purpose Computing on Graphic Processing Units) has become a major way for high performance computing. Recent GPUs have multiple SIMD (Single Instruction Multiple Datastreams) units each of which provides more than hundreds processors, and support thousands of concurrent threads. GPUs are much superior to general purpose multi-core CPUs from the viewpoint of both performance per cost and performance per power. Also, their growth of performance per year is also much higher than that of CPUs.

Although the programming of GPUs was difficult for common programmers, it has been rapidly improved. For instance, NVIDIA and AMD support c-like GPU programming languages, Compute Unified Device Architecture (CUDA) [1] and ATI Stream SDK [2], respectively. Open Computing Language (OpenCL) [3] has been widely spread as a programming environment for various accelerators including GPUs. Such programming environment lowers the barrier for introducing of GPU in the many fields.

In order to obtain the performance beyond a single GPU, clusters with GPUs are popularly used in the field of high performance computing. Japanese Tsubame 2.0 [4] system, a supercomputer using GPUs is in the Top500 of would supercomputers ranking. Generally, these multi-GPU systems are consisting of a large number of network connected PCs each of which provides a few GPUs plugged into each slot. This structure comes from that the the GPU needs support of CPU to control data transfer and kernel execution.

This conventional multi-GPU system cause two problems. The first is an increase of the latency of the communication between nodes. The communication between GPUs must be done via its connected CPU, and it often stretches the latency and limits the bandwidth. When the latency of the communication is large compared with the computation time of GPUs, the time for communication can bottleneck the system, especially when the number of nodes becomes large. The second is the programming complexity of parallel processing. In order to use such a cluster efficiently, programmers are required to describe hierarchical parallel programming: that is, the coarse grained parallel programming which controls the communication between GPUs in MPIs or other message passing library and the fine grained parallel programming for intra-GPUs in GPU specialized programming language such as CUDA and OpenCL. Although some tools have been developed to reduce the multi-GPU systems programming difficulty[5][6], they require some performance overhead.

We address these problems by using ExpEther, the Ethernet-based virtualization technology. ExpEther extends PCI Express(PCIe), the standard interface used for connecting hosts and GPU devices to Ethernet. It provides a transport function for PCIe packets by encapsulating them within an Ethernet frame and tunneling between the con-

nected modules.

The proposed multi-GPU system with ExpEther, which is called GPU-BOX, has only a single host PC connected with a large number of GPUs by the Ethernet consisting of conventional switches and cables. GPU-BOX provides PCIe ports and power supply for GPUs together with the function of ExpEther. This system releases programmers from the requirement to learn communication programming between nodes, and enables users to select the number of GPUs independent of a host PC's capacity. Moreover, the latency for communication between GPUs is not so stretched, since the data is communicated between GPUs without using CPUs.

The rest of this papers is organized as follows. We introduce some related work about multi-GPU systems in Section 2. The key technology, ExpEther is explained in Section 3, and then a multi-GPU system by using GPU-BOX is proposed in Section 4. In Section 5, our experimental results are shown. Finally, we conclude this study with future work in Section 6.

2. Related Works

There are some equipments to increase the number of GPUs which a single host PC provide. PCI Express Switches provided by some vendors such as IDT[7] increase PCIe slots of the host PCs. They enable host PCs to connect GPUs over the number of slots provided with host PCs. Also, PCI-SIG announced the availability of the PCI Express External cabling 1.0 specification[8]. It focuses on the implementation of cabled PCIe. Based on the specification, there are external expansion units of PCIe, for instance CONTEC provides some of bus expansion units for PCI Express[9]. However, compared with the system connected by network, extending with these equipments has less flexibility and extensibility.

Some tools for reduction of the complexity of parallel programming are also developed. Vegeta[5] and Hybrid OpenCL[6] virtualize the communication between nodes. By using them, programmers can use multi GPUs without describing communication program in message passing library, but they require some performance overhead. FLAT[10] also has the same policy, though the target program is described in CUDA instead of OpenCL which is the target of Vegeta and Hybrid.

Our proposed multi-GPU system has only a single host and is connected by Ethernet. It doesn't require to describe communication program between nodes since a single host is used. The flexibility and extensibility are not degraded compared with conventional systems.

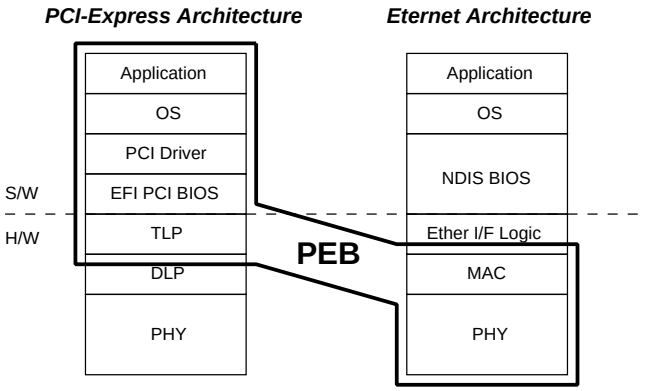

Fig. 1: Overview of PEB

3. ExpEther

ExpEther[11], a key technology of multi-GPU system, is the Ethernet based Virtualization technology developed by NEC[12]. It extends PCIe which is high performance I/O network but limited to small area around the cabinet to much wider area by using Ethernet. Various types of I/O devices on distant location can be connected as if they were inside the cabinet.

In this section, we describe the function of ExpEther, PCI Express-to-Ethernet bridge (PEB) and and Ether-Forwarding-Engine (EFE).

3.1 PEB

As Fig. 1 shows, PEB is the function of ExpEther to bridge the TLP layer in PCIe and the MAC layer in Ethernet. PEB encapsulates a PCIe packet, Transaction Layer Packet (TLP) into Ethernet frame, and decapsulates it for extending PCIe to Ethernet. Moreover, the target Ethernet is virtualized so that the connected devices can be treated as if they were plugged into host PCs without Ethernet. PEB is implemented with a conventional operating system, device driver, PCIe interface, Ethernet Switch and others. That is, the ExpEther technology can be easily introduced into the systems only with PCIe interfaces.

3.2 EFE

EFE is the delay-based protocol which supports repeat and congestion control instead of TPC, the loss-based protocol. The congestion control system in EFE consists of the following steps.

1) It sends a certain number of probe packets, when communication starts.
2) It decides the initial transmission bandwidth based on the acknowledging packets for the probe packets.

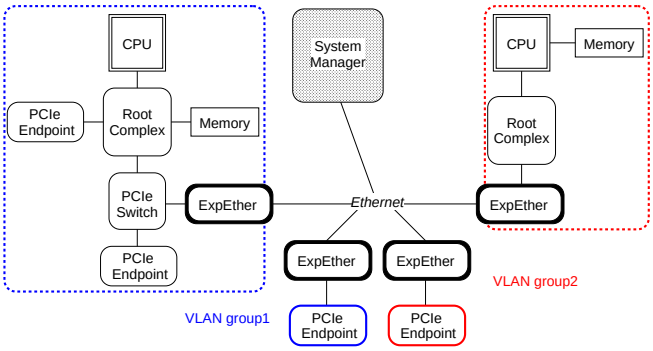

Fig. 2: Example of the System with ExpEther

3) After this, it adjusts the transmission bandwidth for RTT.

EFE employs Go-back-N as a flow-control method. Compared with Selective-Repeat, Go-back-N has the advantage of hardware cost which does not need the reorder buffer. Also, although a number of retransmitted packets are required in the environments of low round-trip propagation delay which ExpEther targets, packet retransmission would not frequently happen.

3.3 System Example using ExpEther

Fig. 2 shows an example of the basic system with ExpEther. On the system of ExpEther, servers and PCIe endpoints are managed and grouped by VLAN ID, which each server or endpoint has in the register of ExpEther bridges. The VLAN ID group consists of a single server and multiple PCIe endpoints. The system manager allocates servers and endpoints the VLAN ID, and can flexibly constitute the groups. On the data transfer, the Ethernet frame has the VLAN ID tag which is referenced to distinguish the group to which the frame belongs.

Though ExpEther doesn't support direct CPUs connection, it can be done by using Remote Direct Memory Access (RDMA). By using the RDMA, CPUs can communicate each other by accessing memory of another CPU directly. It provides low latency and high throughput communications, and suppresses the CPU workload.

4. Design and Implementation

In this section, we propose a multi-GPU system with ExpEther, and show an implementation using GPU-BOX.

4.1 System Design

Fig. 3 shows an overview of the multi-GPU system with ExpEther. It allows to interconnect a single host PC and multiple GPU devices with a common Ethernet through the

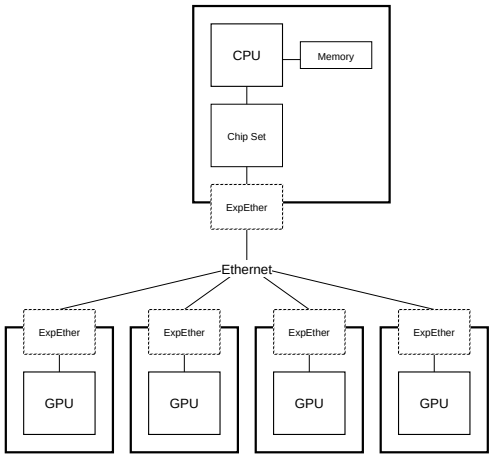

Fig. 3: Multi-GPU System with GPU-BOX

PEB. The network consists of the common Ethernet equipments such as switches and cables. Programmers can treat the multi-GPU system as if all of GPU devices were directly ported into a single host PC, since the PEB encapsulates communication for the control via the network. On the multi-GPU system with only a single host, programmers can make the use of computing power of multiple GPUs without learning programming skill of communication between nodes such as MPI.

The following is a list of the benefits and advantages that the multi-GPU system with ExpEther provides.

- Performance: Stretching the latency of communication between GPUs is suppressed by using ExpEther.
- Programmability: Even programmers who can't describe communication between GPUs can use the system with multiple GPUs.
- Portability: The existing GPU program can run with only a small change about the number of devices in GPU programming language for instance CUDA.
- Flexibility: When the number of GPUs is changed, the application on multiple GPUs can run with only program modification of device size and extending network with Ethernet switches.
- Compatibility: GPUs are accessible to a conventional operating system, device driver, PCIe interface, and Ethernet switch. Thus, they can be used without special modification.
- Future Prospect: The proposed multi-GPU system can receive the benefit of Ethernet technology improvement which will constantly continue in future.

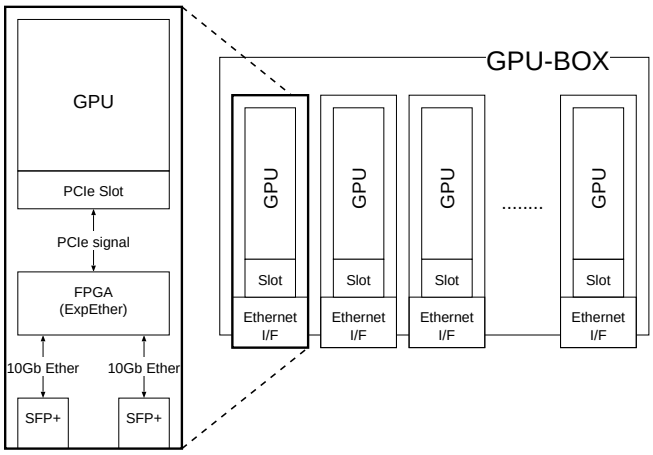

Fig. 4: Overview of GPU-BOX

Table 1: Evaluation Environment

CPU	Intel Core i7 (2.67 GHz)
GPU	NVIDIA Tesla C2050 x6
Host Memory	16 GB
OS	Scientific Linux 6.0
Host Compiler	gcc4.4
CUDA	Toolkit 4.0
Network	10Gb Ethernet x2
Switch	Fulcrum Microsystems Monaco

Table 2: Problem Size of Particle Motion

Number	Particles	Steps
1	1x1024x1024	100
2	1x1024x1024	1000
3	10x1024x1024	1000
4	10x1024x1024	2000
5	10x1024x1024	4000

4.2 GPU-BOX

Here, multi-GPUs with ExpEther is implemented in GPU-BOX which provides PCIe ports and power supply together with the function of ExpEther for extending PCIe interface of GPUs to Ethernet network.

The target GPU-BOX in this paper has slots for eight GPUs. That is, it provides the environment of eight GPUs in term of PCIe slots, Ethernet interface, power supply and space. The power supply of the GPU-BOX is up to 3000W. The Ethernet interfaces are two SFP+ interfaces per a GPU, thus 20 Gbps bandwidth is available in total. Each PCIe in GPU-BOX is extended to Ethernet by ExpEther NIC implemented on FPGAs.

5. Evaluation

In this section, we evaluate the performance of the multi-GPU system with GPU-BOX by executing programs in CUDA.

5.1 Experimental Environment

Table 1 shows the multi-GPU environment used in the evaluation. The evaluated environment uses six GPUs, each of which is NVIDIA Tesla C2050, while the number of slots in the target GPU-BOX is eight.

5.2 Application

For performance evaluation, we implemented two application programs, the simulation of particles motion and the calculation of Advection term.

One has no communication between devices, while the other needs a considerable amount of communication.

These application programs are mainly consisting of following three parts;

- computing in GPUs,
- data transfer between host and GPU, and
- data exchange between GPUs.

The calculation of particle has only two parts, computing and data transfer between host and GPU, while the calculation of Advection term includes all of them.

5.2.1 Simulation of Particle Motion by the Runge-Kutta Method

This application simulates particle motion when initial particle distribution and velocity field are given and there is no interference between particles. It divides time into several steps, and on each step, each particle position is updated with the velocity field by the Runge-Kutta method. For the motion of each particle is independent from another, it is possible to perform every particle motion computation in parallel, and there is no communication between GPUs in multi-GPU environment.

Here, five combinations of parameters are executed for evaluation. Table 2 shows the combinations of problem size and steps.

5.2.2 Calculation of Advection term by Cubic Lagrange Interpolation

Calculation for Advection term of Cartesian grid method is a kind of fluid dynamics computation. It simulates the movement of ink when initial concentration, distribution, and velocity field are given. On this calculation, it separates the entire surface into grid and updates each value of the grid using values of the surrounding grids in a certain time step. On each step, updating of the grid value is independent from other computations. However, in case of computing with

Table 3: Problem Size of Advection Term

Number	X	Y	Steps
1	256	256	1024
2	1024	1024	10280
3	4096	2048	10280

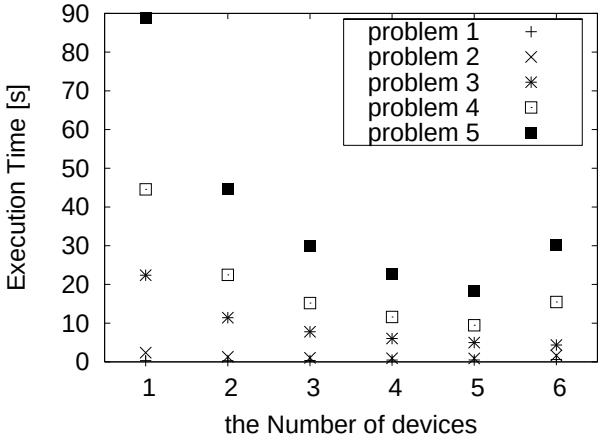

Fig. 5: Execution Time of Calculation of Particle Motion

multiple GPUs, we have to exchange data around memory-boundary between GPUs. Problem size of Cartesian grid method on this evaluation is shown in Table 3.

5.3 Performance versus the number of GPUs

5.3.1 Simulation of Particle Motion

Fig. 5 shows relationship between execution time and the number of GPUs when the particle motion simulation is executed. It is found that the execution time decreases as increasing the number of GPUs. However the execution of the case with six GPUs in the GPU-BOX spent larger time than one with five GPUs. That is caused by the EFE protocol tuning which is not adequate for the system with more than five GPUs.

Fig. 5 shows the performance speedup of the different number of GPUs over a single device. Respectively, the multi-GPU system provided speedup of 1.99, 2.96, 3.92, 4.83 and 5.14 times at most for execution on two to six devices. While performance speedup is directly proportional to the number of devices roughly in case of large problem size, there are cases to get no performance improvement by increasing the number of GPUs. It is mainly caused by the overhead of data transfer between the host processor.

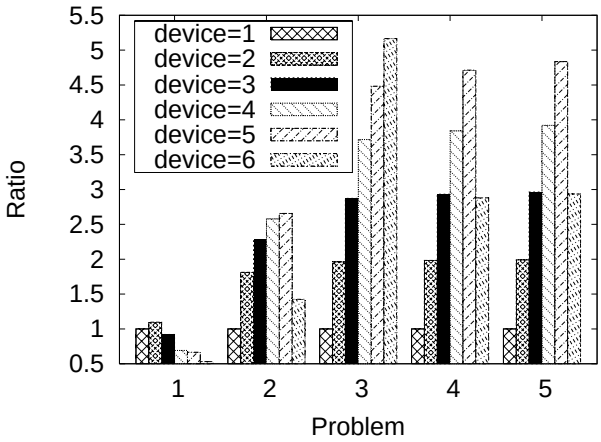

Fig. 6: Performance rate of Calculation of Particle Motion

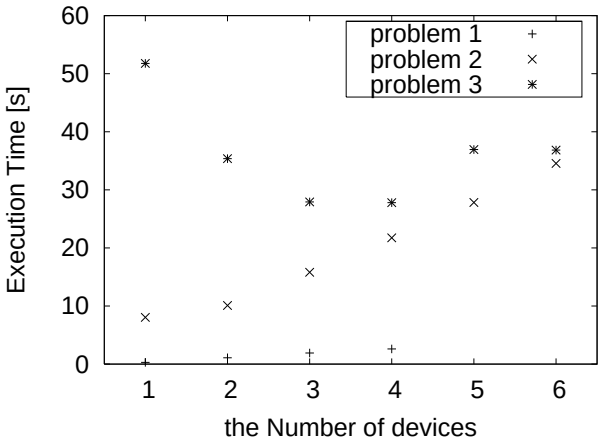

Fig. 7: Execution Time of Calculation of Advection term

5.3.2 Calculation of Advection term

Fig. 7 shows relationship between execution time and the number of GPUs when the Advection term is calculated. In the problem 1, the cases with five and six devices can't be evaluated, since the problem size is so small that there is no data allocated to fifth and sixth device in this application program.

Fig. 8 shows the performance enhancement of the different number of GPUs over a single one. Respectively, the multi-GPU system achieved speedup of 1.45, 1.85, 1.86, 1.40 and 1.41 times in problem 3 for execution with two to six devices. On the other hand, we can see performance degradation on calculating small size problems; problem 1 and 2. The performance degradation is caused by the frequently executed data exchange part for the calculation of

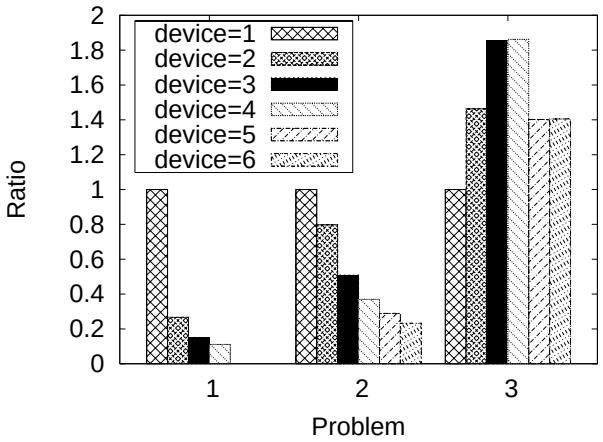

Fig. 8: Performance Rate of Calculation of Advection term

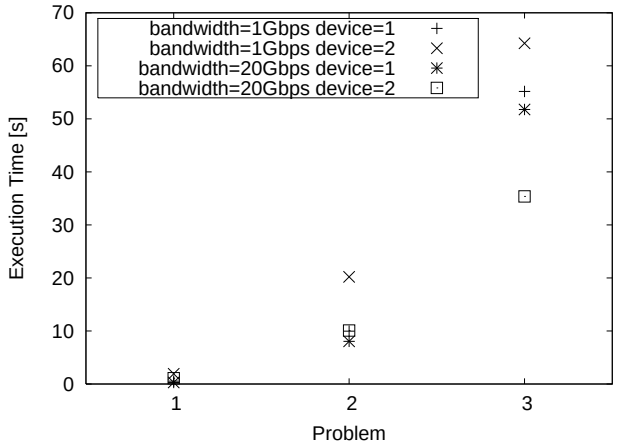

Fig. 9: Relationship between Network and Execution Time

Table 4: Data Transfer Bandwidth in CUDA

Ethernet	1 Gbps	20 Gbps
Host to Device	0.320 Gbps	7.66 Gbps
Device to Host	0.445 Gbps	9.86 Gbps
Device to Device	0.355 Gbps	7.92 Gbps

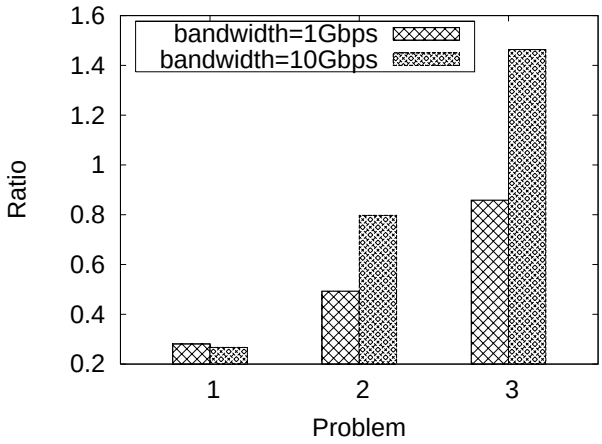

Fig. 10: Relationship between Network and Performance Improvement

Advection term. The data exchange speed mainly depends on the bandwidth of the network, while the exchanged data amount in this application is proportional to the number of GPUs. Although it is difficult to enhance performance in such small problems, their execution time is not so large and the effect of applying multi-GPU system is originally limited. Improving the performance of Ethernet will stretch the target which can be accelerated to smaller size problems.

5.4 Influence of the Network Performance

In this subsection, we evaluate the influence of the network equipment in the GPU-BOX. Table 4 shows the bandwidth of data transfer from the host to the GPU, from the GPU to the host, and between GPUs in CUDA by using Ethernet 1 Gbps and 20 Gbps.

5.4.1 Influence to Application performance

Fig. 9 shows performance of Advection term calculation the case when 20 Gbps and 1 Gbps Ethernet are used with a single and two GPUs. When a single GPU is used for calculation, the difference of the execution time with 1 Gbps and 20 Gbps Ethernet is small. However, 1 Gbps Ethernet increases the execution time on calculating with two GPUs, while 20 Gbps Ethernet can decrease it.

Fig. 10 shows the speedup with two GPUs normalized to that with a single GPU. As the problem size becomes larger, the difference of achieved speedup becomes large between two Ethernets, although the multi-GPU system provides more performance in both. As a result, the speedup of the multi-GPU system is 0.86 with 1 Gbps network, and 1.45 with 20 Gbps network. These results indicate that the multi-GPU system with GPU-BOX obtains benefits of developing Ethernet technology, and GPU-BOX enables user to select network construction according to the application.

6. Conclusion

In this study, we proposed and evaluated multi-GPU system with ExpEther. It allows to interconnect a single host PC and multiple GPU devices with ExpEther which extends PCIe interface to Ethernet.

For evaluating the multi-GPU system, two application programs are used. First, we evaluated performance on the different number of devices. For the program without inter-GPU communication, a system with six GPUs achieved 5.14 times performance as that with a GPU.

On executing the application including data exchange between GPUs, the largest performance improvement was 1.86 times with four GPUs. Then, we evaluated performance using the different networks. From the results, it appears that the bandwidth of network used in GPU-BOX greatly affects the performance of the multi-GPU system.

The following is a list of future work:

- The performance of multi-GPU system with ExpEther must be compared with conventional multi-GPU clusters,
- The performance must be evaluated with the other applications and more number of GPU.
- A larger system which uses multiple switches must be evaluated.

References

[1] NVIDIA, "NVIDIA CUDA Compute Unified Device Architecture," http://developer.nvidia.com/object/cuda.html.

[2] A. M. Devices, "Ati stream sdk getting started guide (v2.3)," http://developer.amd.com/GPU/ATISTREAMSDK/DOCUMENTATION/Pages/default.aspx.

[3] NVIDIA, "The OpenCL Specification Version: 1.0," 09.

[4] T. I. of Technology Global Scientific Information and C. Center, "Tsubame2," http://www.gsic.titech.ac.jp/tsubame2.

[5] A. Shitara, T. Nakahama, M. Yamada, T. Kamata, Y. Nishikawa, M. Yoshimi, and H. Amano, "Vegeta: An implementation and evaluation of development-support middleware on multiple opencl platform," in *The Second International Conference on Networking and Computing*, November 30 - December 2, 2011.

[6] R. Aoki, S. Oikawa, T. Nakamura, and S. Miki, "Hybrid opencl: Enhancing opencl for distributed processing," in *Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium on*, 26-28 May 2011, pp. pp.149 – 154.

[7] Integrated Device Technology, "Pci express switches," http://www.idt.com/products/interface-connectivity/pci-express-solutions/pci-express-switches.

[8] PCI-SIG, "Pci express external cable 1.0 specification."

[9] CONTEC, "PCI Express External Cabling (PCISIG) Compliant Expansion Units," http://www.contec.com/products/bus_exp/pcie.php.

[10] T. Miyoshi, H. Irie, K. Shima, H. Honda, M. Kondo, and T. Yoshinaga, "Flat: a gpu programming framework to provide embedded mpi," in *GPGPU-5 Proceedings of the 5th Annual Workshop on General Purpose Processing with Graphics Processing Units*, 2012, pp. pp. 20–29.

[11] J. Suzuki, Y. Hidaka, J. Higuchi, T. Yoshikawa, and A. Iwata, "Expressether - ethernet-based virtualization technology for reconfigurable hardware platform," in *High Performance Interconnects*, 2006, pp. pp.45–51.

[12] NEC Corporation, http://www.nec.co.jp.