Performance analysis of the multi-GPU System with ExpEther

Shimpei Nomura
Keio University
3-14-1 Hiyoshi, Yokohama,
223-8522, Japan
nomura@am.ics.keio.ac.jp

Takuji Mitsuishi
Keio University
3-14-1 Hiyoshi, Yokohama,
223-8522, Japan
mits@am.ics.keio.ac.jp

Jun Suzuki
NEC Corporation
1753, Shimonumabe,
Nakahara-ku, Kawasaki,
Kanagawa 211-8666, Japan
j-suzuki@ax.jp.nec.com

Yuki Hayashi
NEC Corporation
1753, Shimonumabe,
Nakahara-ku, Kawasaki,
Kanagawa 211-8666, Japan
y-hayashi@kv.jp.nec.com

Masaki Kan
NEC Corporation
1753, Shimonumabe,
Nakahara-ku, Kawasaki,
Kanagawa 211-8666, Japan
kan@bq.jp.nec.com

Hideharu Amano
Keio University
3-14-1 Hiyoshi, Yokohama,
223-8522, Japan
asap@am.ics.keio.ac.jp

Abstract

A GPU cluster in which each node provides a few GPUs connected with PCIe (PCI Express) is commonly used for acceleration of a large application program requiring the performance beyond a single GPU. However, in such a system, programmers are required to describe two parallel programming between nodes in MPIs or other message passing library as well as the fine grained parallel programming for intra-GPUs. As a cost effective alternative of such clusters, we propose a novel multi-GPU system with ExpEther, a virtualization technique which extends PCIe of a host CPU to Ethernet. All devices connected by ExpEther can be treated as if they were directly connected to the host. Evaluation with two application programs with and without GPU-GPU communication revealed that the proposed system with four GPUs achieved 3.88 and 3.29 times performance improvement respectively compared with a single GPU system. Compared with GPU cluster system in which each node provides a GPU, the proposed system achieved about 7% and 30% performance improvement, respectively.

Keywords

GPU, Cluster, ExpEther, Parallel Computing, Scalability

1. INTRODUCTION

In the field of high performance computing, acceleration with GPUs has become popular. The growing GPU computing in recent years is supported by their rapid performance improvement, higher energy efficiency than common multi-core CPUs, and improved development environment such as CUDA[7] and OpenCL[5].

When the performance supported by a single GPU is not enough, acceleration using multiple GPUs is required. In such cases, a cluster in which each node provides a few GPUs connected with PCIe (PCI Express) is commonly used. Japanese Tsubame 2.5 [2] system is at the 11th place in the Top500 of the world super computers ranking, and Tsubame-KFC GCIC center won the top of Green500. Smaller clusters using GPUs have been widely utilized in universities or research laboratories as cost efficient supercomputers.

In such a system, as shown in Figure 1 (a), the number of GPUs connected in a node is limited by the number of PCIe ports of the host, power supply and cable length limitation of PCIe. Thus, in most of such systems, two GPUs are connected to a single host with PCIe, and such a node is connected with each other by using Infiniband or Ethernet.

However, such a hierarchical structure introduces the following problems. First, the programming complexity of parallel processing is increased. 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 nodes in MPIs or other message passing library, and the fine grained parallel programming for intra-GPUs in a specialized programming language such as CUDA and OpenCL. Although some tools have been developed to reduce such programming difficulty[11][1][9], they suffer a certain performance overhead.

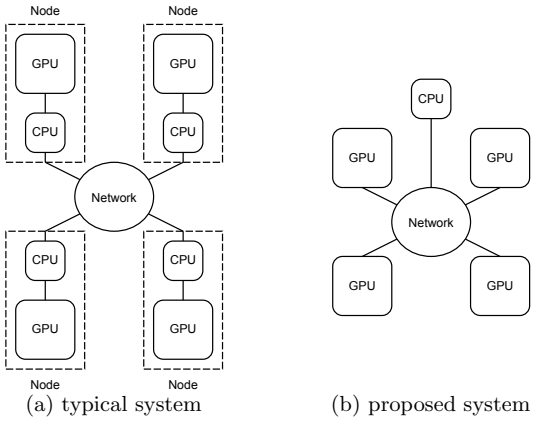
The second is the latency of communication between nodes tends to be large. Since GPUs connected to the different host must communicate via hosts, and it often stretches the latency and limits the bandwidth when the number of nodes becomes large.

The third is the cost of the hierarchical heterogeneous architecture. Generally, most of host processors only manage the communication between GPUs during execution. It means that computational power of powerful host CPUs is mostly wasted.

In order to address such problems, we propose a novel multi-GPU system with ExpEther, a virtualization technique[10] which extends PCIe of a host CPU to Ethernet. All devices connected by ExpEther can be treated as if they were directly connected to a host computer.

Figure 1 (b) shows an overview of the proposed system. A number of GPUs can be easily connected to a single host di-

This work was presented in part at the international symposium on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART2014), Sendai, Japan, June 9-11, 2014.



(a) typical system (b) proposed system

Figure 1: Comparing typical and proposed systems

rectly by using ExpEther. This structure gives programmers a flat view of multiple GPUs which enables programmers to describe parallel programming without using MPI or other message passing library for inter-node communication. Also, the cost and power consumption of multiple host CPUs in the traditional system shown in (a) are not needed.

In the proposed system (b), however, the latency of ExpEther might degrade the performance, although the latency through the host in (a) can be omitted. Thus, the performance analysis of the real system using micro-benchmark and practical application benchmarks are required.

The contribution of this paper is as follows:

- A prototype multi-GPU architecture using ExpEther called GPU-BOX is proposed.
- The performance analysis using micro-benchmark and practical application programs are shown, and it is demonstrated that GPU-BOX can achieve better performance than a corresponding cluster for applications which require a limited inter-GPU data exchange.

2. RELATED WORK

In order to reduce the load of programmers of GPU clusters, tools[1][9] [11] to hide the communication between nodes from programmers have been developed. They give programmers of GPU clusters flat vision like Figure 1 (b) by software environment so that communication part using MPI is not necessary. However, such an approach using software tools suffer a certain performance degradation.

Extending PCIe also enables to build a flat multi-GPU system. Although PCIe is commonly used in a cabinet of computers, it can be extended using cables specified in PCIe External Cabling Spec.2.0[8] and PCIe switches[4]. By using such PCIe equipments, a number of GPUs can be connected with a single host. For example, in DEGIMA-2[3], more than hundred FPGAs and GPUs are connected to a single host by using original PCIe switches. Unlike software approach, GPUs and host are tightly connected without performance degradation. However, since PCIe is originally designed as a standard interface inside the cabinet, there is a limitation in a cable length which makes interconnection of a large number of GPUs difficult. The cost of PCIe switches is much higher than that for Ethernet, since they are not so commonly used.

3. EXPETHER

ExpEther[10] is an Ethernet based virtualization technique, which is developed by NEC[6]. 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 term of performance, DMA transfer based on PCIe specification is available without modification for communication between a host and a device. A latency overhead for communication via Ethernet is also decreased by using an original protocol for retransmission and congestion control.

3.1 ExpEther NIC

3.1.1 PEB

ExpEther NIC is consisting of bridges (PEB :PCI Express-to-Ethernet Bridge) between PCIe and Ethernet. PEB encapsulates Transaction Layer Packet (TLP) used in PCIe into Ethernet frame, and decapsulates it for extending PCIe to Ethernet. PEB is implemented with common hardware/software components including a conventional operating system, device driver, PCIe interface, Ethernet cables and Ethernet Switch.

3.1.2 Specialized Protocol for ExpEther

ExpEther uses delay-based congestion control on Ethernet to reduce the latency of communication unlike TCP with loss-based protocol. In this protocol, a certain number of probe packets are sent, when communication starts. Then, the initial transmission bandwidth is decided based on the acknowledging packets for the probe packets. After this, the transmission bandwidth is adjusted according to RTT (Round Trip Time). EFE (Ether Forwarding Engine) implemented in ExpEther NIC PEB manages the congestion control.

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.

3.2 A system example using ExpEther

Figure 2 shows an example of the basic system with ExpEther. It consists of servers and devices connected by Ethernet. On systems with ExpEther, multiple computer systems can exist together on a single Ethernet. Servers and PCIe endpoints are grouped by the GID (Group ID), and it is registered in each ExpEther NIC. A system manager can change GID in ExpEther NICs and flexibly constitute the groups of computer systems. Other detail of ExpEther is shown in [6].

4. MULTI-GPU SYSTEM WITH EXPETHER

In this section, a multi-GPU system with ExpEther is proposed with its prototype called GPU-BOX.

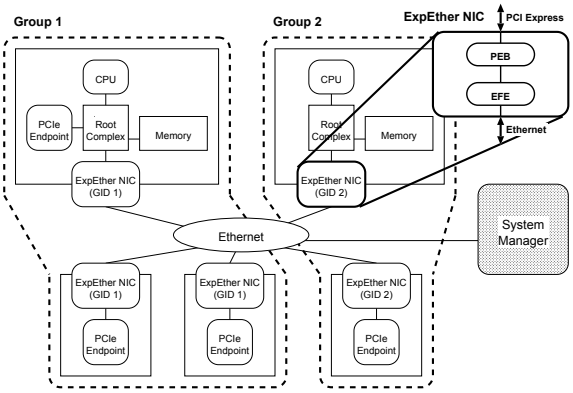


Figure 2: A system example using ExpEther

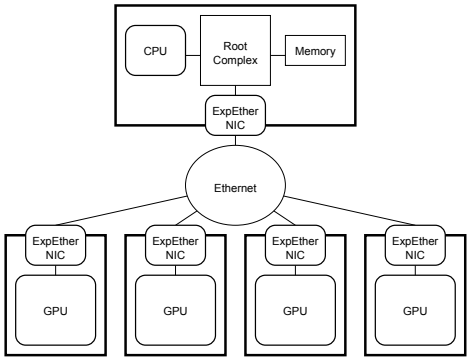


Figure 3: A multi-GPU System with ExpEther

4.1 System Design

ExpEther enables to connect a single host node with a large number of GPUs via Ethernet network switch and control them as if they were connected to PCIe of the host.

Figure 3 shows an overview of the multi-GPU system with ExpEther. The host and GPUs are connected with common Ethernet cables and switches by using ExpEther NIC attached to their PCIe port. Data are transferred directly between GPUs without using host memory by using GPUDirect in CUDA. Although the number of GPUs connected with a single Ethernet switch is limited, the number of GPUs can be increased by using a common extension technique used in Ethernet, that is, a tree like hierarchical structure with multiple switches. Note that the original low latency communication protocol is used in "Ethernet" shown in Figure 3, while it is built from common Ethernet cables and switches. Since Ethernet can be much more extendable than PCIe, GPUs located distant place can be connected and treated together.

Although the proposed system much improves the programmability and flexibility of the traditional GPU clusters, a certain overhead in ExpEther NIC will stretch the latency between GPUs and the host. In order to make the influence of using ExpEther clear, a comprehensive analysis is done in this paper.

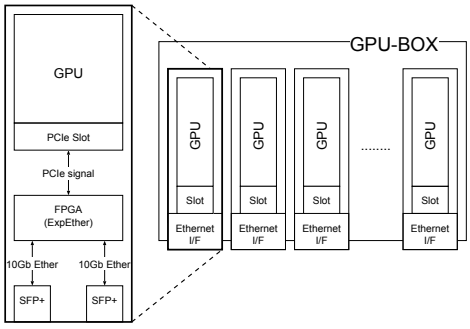


Figure 4: Block Diagram of GPU-BOX

Table 1: Evaluation Environment

	PCIe Direct	GPU cluster	Proposed
CPU	Xeon E5-1650 @3.20 GHz		
GPU	NVIDIA Tesla C2050		
Host Memory	16 GB		
OS	CentOS 6.0		
Host Compiler	gcc4.4		
CUDA	Toolkit 4.2		
MPI	×	OpenMPI 1.6	×
Network	×	10GbE	10GbE x2
Switch	×	DELL Force10 S2410P	

4.2 GPU-BOX

Before evaluation, the target prototype system called GPU-BOX is introduced. Although the proposed system enables to treat a lot of GPUs located in the distant places, as the first step of research, here, multiple GPUs are implemented in a single cabinet called GPU-BOX. As shown in Figure 4, GPU-BOX provides multiple PCIe slots with ExpEther NIC and power supply. The target GPU-BOX in this paper has eight slots. Through the ExpEther NIC implemented with an FPGA, two 10Gb Ethernet ports (EFP+) are provided for each slot, thus, 20Gbps bandwidth is available in total. The power supply of the target GPU-BOX is up to 3000W which is enough to connect eight GPUs.

5. EVALUATION

5.1 Experimental Environment

GPU-BOX is evaluated and compared with traditional multi-GPU systems in which each node provides a GPU (GPU cluster). Also, a single node multi-GPU system (PCIe Direct) connecting multiple GPUs with PCIe is evaluated for measuring the overhead of ExpEther. Table 1 shows the detail specification of multi-GPU systems used in the evaluation. Of course, the same GPUs, host CPU, OS and programming environment are used for fair evaluation.

5.2 Communication Performance

First, the communication performance between hosts and devices of the proposed system is evaluated by measuring small kernels described in CUDA. Transfer time from a host

Table 2: Evaluation target in microbenchmark

target	called function
HtoD Sync	cudaMemcpy
HtoD Async	cudaMemcpyAsync
DtoH Sync	cudaMemcpy
DtoH Async	cudaMemcpyAsync
DtoD Sync	cudaMemcpyPeer
DtoD Async	cudaMemcpyPeerAsync
Kernel Sync	kernel + cudaThreadSynchronize
Kernel Async	kernel

Table 3: Minimum latency

	PCIe Direct (μs)	Proposed (μs)
HtoD Sync	24.9	225
HtoD Async	2.79	14.2
DtoH Sync	26.8	223
DtoH Async	2.79	19.1
DtoD Sync	30.0	39.4
DtoD Async	23.3	57.8
Kernel Sync	22.8	184
Kernel Async	1.34	6.19

to a device (HtoD), that from a device to a host (DtoH), and that from a device to another device (DtoD) were evaluated. Also, by executing a tiny kernel function, the latency overhead for calling kernel function was evaluated. Two cases: synchronous calls(Sync) and asynchronous calls(Async) were tried and compared. For asynchronous kernel call, calling kernel function was substituted by `cudaThreadSynchronize()`, so that the kernel was asynchronously launched. Evaluated times are average of a large number of iterative executions.

5.2.1 Latency

Figure 5 shows the execution time of the data transfer in GPU-BOX with ExpEther. The execution time is kept constant until a certain size of data transfer, and this time is considered as the minimum latency required for set up the data transfer. It depends on the type of data transfer as shown Table 3.

In GPU-BOX, synchronous data transfer between a host and a device requires about ten times minimum latency compared with the asynchronous transfer. Also, kernel execution time of the host-device communication is largely increased because of the synchronization. It is considered that synchronizing a host and a device needs relatively long time, about 200μs.

Table 3 also compares PCIe direct and the proposed system. Because of the overhead using ExpEther, the minimum latency is increased from 1.3 times to 9 times. The difference is large for the communication between host and devices, while the difference in DtoD Async and DtoD Sync communication is 1.3 and 2.48, respectively. This evaluation results suggest that in multi-GPU system with ExpEther, the communication between GPUs must be done directly as possible.

5.2.2 Maximum Throughput

Figure 5 shows that more than 1MB sized data, the latency of all methods is almost the same depending on the maximum throughput in the system. The measured values of maximum throughput on each type of data transfer are shown in Table 4.

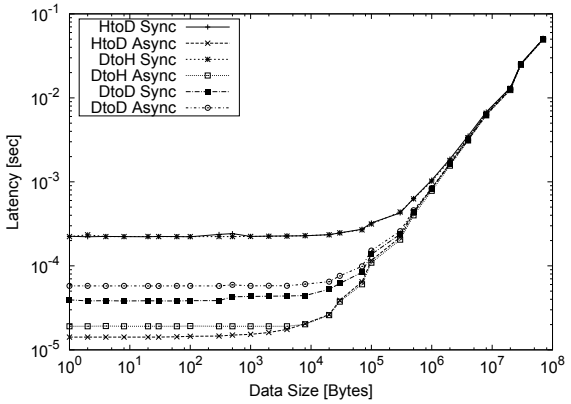


Figure 5: Latency of data transfer in the proposed system

Table 4: Maximum throughput

	PCIe Direct (Gbps)	Proposed (Gbps)
HtoD Sync	48.8	10.6
HtoD Async	48.4	10.7
DtoH Sync	52.1	10.8
DtoH Async	52.6	10.9
DtoD Sync	47.1	10.7
DtoD Async	47.1	10.7

The throughput of PCIe direct is, of cause, larger than the proposed system, by about 5 times. However, considering that the maximum throughput of GPU-BOX is 20Gbps (10Gbps $\times$ 2) at maximum, the proposed system could make well use of the bandwidth of Ethernet.

5.3 Application Performance

For performance evaluation, we implemented two application programs, the simulation of particles motion and the calculation of advection term. The former has no communication between devices, while the latter includes a considerable amount of communication between devices. They are mainly consisting of following three parts; (1) computing in GPUs, (2) data transfer between host and GPU, and (3) data exchange between GPUs. The former, the calculation of particle has only two of them, while the latter, the calculation of advection term includes all of them. In the programs for GPU-BOX, the communication between GPUs are described using memory copy APIs shown in Table 2 in GPU-BOX. Synchronization of multiple GPUs are done using cudaThreadSynchronize API in CUDA. In contrast, in the programs for GPU cluster, MPI functions (MPI_Send, MPI_Recv, MPI_Isend, and MPI_Irecv) are used for data communication as well as a barrier synchronization function (MPI_Barrier).

5.3.1 Target Application

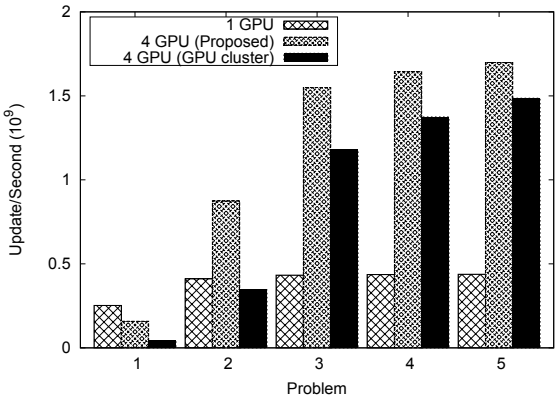


Figure 6: Performance of calculating particle motion

5.3.1.1 Particle Motion Simulation 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. Since the motion of each particle is independent from the other, it is possible to perform every particle motion computation in parallel, and there is no communication between GPUs in multi-GPU environment. The computational complexity is O(N) where N is the number of particles.

5.3.1.2 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 grids 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 multiple GPUs, we have to exchange data around memory-boundary between GPUs. The computational complexity is O(M×N) where M and N are the number of grids for x and y direction, respectively. The communication amout becomes O(M).

5.3.2 Evaluation of execution time

5.3.2.1 Simulation of Particle Motion.

Figure 6 shows the performance on simulating particle motion in three systems, a single GPU system, proposed system with four GPUs, and a GPU cluster with four node in which each node has a GPU. Table 5 shows the size of each problem used in evaluation.

The performance was improved with multiple GPUs for enough large size problem. Compared with a single GPU, proposed system with four GPUs achieved 3.88 times performance improvement.

Table 5: Problem Size of Particle Motion

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

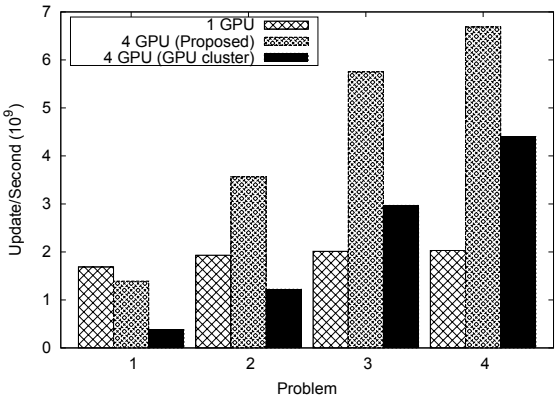


Figure 7: Performance of calculating advection terms

On the other hand, computing the small size problem (number 1 and 2), both proposed system and GPU cluster could not achieve the performance proportional to the number of GPUs. In the case of small problems, the parallelism is not enough for the number of core, and the performance is influenced strongly by host-device data transfer.

However, the proposed system achieves better performance than GPU cluster especially for small sized problem. In term of execution time, the difference is almost a constant corresponding to the initialization of communication. Even in a large sized problem, the performance of the proposed system is about 7% better than that of GPU cluster.

5.3.3 Calculation of Advection Term

Figure 7 shows the performance on computing advection term in three systems. Table 6 shows the size of each problem used in evaluation.

The proposed system achieved 3.29 times performance at most compared with a single GPU system, for enough size of problem.

Similar to the case of particle motion simulation, the performance improvement by multiple GPUs could not be obtained for computing small problems. In calculating advection term, the communication overhead between GPUs also causes the performance degradation as well as communication overhead between host and GPUs.

Compared with GPU cluster, the proposed system also achieved higher performance. Even for a large size problem, the performance of the proposed system is about 30the case of particle motion simulation, since advection term requires a lot of GPU-GPU communications. As shown in the

Table 6: Problem Size of Advection Term

Number	X	Y	Steps
1	1024	1024	1024
2	2048	2048	1024
3	4096	4096	1024
4	8192	8192	1024

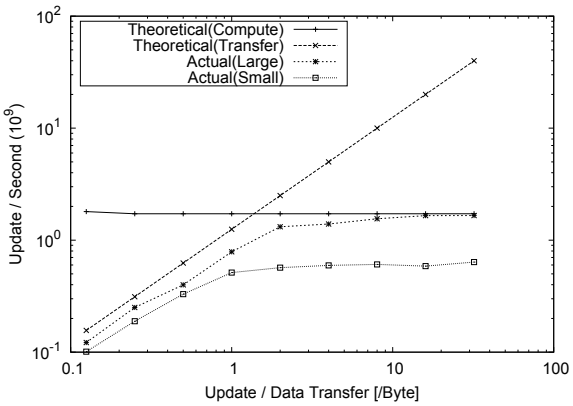


Figure 8: Performance versus Data transfer

previous evaluation results, the minimum latency of DtoD communications is small in the proposed method. Since the ExpEther supports an original low latency protocol, the latency of communication is shorter than that of GPU cluster which requires the communication through the host.

5.4 Performance versus arithmetic intensity

Figure 8 shows the performance versus arithmetic intensity measured by executing simulation of particle motion.

In the simulation, GPUs send host locations of particles represented with 8 byte data in every fixed number of steps. The arithmetic intensity can be computed as follows.

$$(number of steps/(8 \times number of data transfer)$$

Theoretical performance and data transfer lines are also shown in the graph. The actual performance is evaluated with computing large size (1024x1024 particles) problem and small size (32x1024 particles) problem.

Computing the large size problem, the proposed system achieves performance close to two theoretical performance. Data transfer becomes bottleneck in the case of low computation ratio and computation becomes bottleneck in the case of high computation ratio. Figure 8 shows that the bottleneck of the system can be estimated based on a single GPU performance, data transfer throughput and the ratio of computation and data transfer in the application.

However, computing the small size problem, performance in the system is lower than theoretical, especially at high computation ratio. It occurs because of lack of the parallelism which cannot be estimated in the graph.

6. CONCLUSION

A multi-GPU system with ExpEther is proposed and evaluated. It allows to interconnect a single host PC and multiple GPU devices with ExpEther which extends PCIe bus to Ethernet.

Through the evaluation with micro-benchmarks, host-device communication of the proposed system requires up to 9 times minimum latency compared with the case only PCIe bus is used. On the other hand, the minimum latency for communication between devices did not increase greatly. In every data transfer, the maximum throughput is about 11 Gbps showing that it makes well use of 20 Gbps Ethernet bandwidth.

Evaluation with two application programs with and without GPU-GPU communication appeared that the proposed system with four GPUs achieved 3.88 and 3.29 times performance improvement respectively compared with a single GPU system. Compared with GPU cluster system in which each node provides a GPU, the proposed system achieved about 7% and 30% performance improvement, respectively. Now, the system with evaluation provides only four GPUs. Evaluating larger size system is our fugure work.

7. REFERENCES

[1] 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 *Proc. of the 2nd ICNC, 2011*, pages 141–147. IEEE, 2011.

[2] GSIC. Tsubame computing services. http://tsubame.gsic.titech.ac.jp/en.

[3] T. Hamada. Degima: The greenest accelerator-based supercomputer in the top500 list. http://www.cs.tsukuba.ac.jp/˜yoshiki/heart/HEART2012/keynote/HEART2012-Hamada.pdf, June 2012.

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

[5] Khronos. The opencl specification version: 2.0, November 2013.

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

[7] NVIDIA. CUDA Toolkit Documentation. http://docs.nvidia.com/cuda/index.html.

[8] PCI-SIG. Pci express. http://www.pcisig.com/specifications/pciexpress/.

[9] R.Aoki, S.Oikawa, T.Nakamura, and S.Miki. Hybrid opencl: Enhancing opencl for distributed processing. In *Proc. of the 9th ISPA, 2011*, pages 149–154. IEEE, 2011.

[10] J. Suzuki, Y. Hidaka, J. Higuchi, T. Yoshikawa, and A. Iwata. Expressether-ethernet-based virtualization technology for reconfigurable hardware platform. In *High-Performance Interconnects, 14th IEEE Symposium on*, pages 45–51. IEEE, 2006.

[11] T.Miyoshi, H.Irie, K.Shima, H.Honda, M.Kondo, and T.Yoshinaga. Flat: a gpu programming framework to provide embedded mpi. In *Proc. of the 5th Annual Workshop on General Purpose Processing with Graphics Processing Units*, pages 20–29. ACM, 2012.