Tightly Coupled Accelerators Architecture for Low-latency Inter-node Communication between Accelerators

Toshihiro Hanawa
Information Technology Center, The University of Tokyo, Japan

Yuetsu Kodama Taisuke Boku Mitsuhisa Sato
Center for Computational Sciences, University of Tsukuba, Japan

I. Introduction

In recent years, heterogenious clusters using accelerators are widely used for high performance computing system. In such clusters, the inter-node communication among accelerators requires several memory copies via CPU memory, and the communication latency causes severe performance degradation. To address this problem, we propose Tightly Coupled Accelerators (TCA) architecture, which is capable of reducing the communication latency between accelerators over different nodes. The TCA architecture communicates directly via the PCIe protocol, which allows it to eliminate protocol overhead, such as that associated with InfiniBand and MPI, as well as the memory copy overhead.

II. HA-PACS System

HA-PACS (Highly Accelerated Parallel Advanced system for Computational Sciences) is the 8th generation of PACS/PAX series supercomputer in Center for Computational Sciences, University of Tsukuba. HA-PACS system consists of two parts as follows:

- **HA-PACS base cluster** is operated for the development and product-run on advanced scientific computations since Feb. 2012 with 802 TFlops of peak performance. Each node is equipped with two Intel Xeon E5 (SandyBridge-EP) CPUs, four NVIDIA M2090 GPUs, and dual port of InfiniBand QDR.
- **HA-PACS/TCA** is an extension of the base cluster, and operation has started on Oct. 2013 with 364 TFlops of peak performance. Each node is equipped with two Intel Xeon E5 v2 (IvyBridge-EP) CPUs, four NVIDIA K20X GPUs, and each one has not only InfiniBand HCA but also TCA communication board (PEACH2 board) as the proprietary interconnect for GPU in order to realize GPU-to-GPU direct communication over the nodes.

The TCA architecture is based on the concept of using the PCIe link as the communication network link between GPUs on communication nodes rather than simply using the PCIe link and intra-node I/O interface.

Figure 1 shows a block diagram of the communication in HA-PACS/TCA. This configuration is similar to that of the HA-PACS base cluster, with the exception of the PEACH2 board. Whereas accessing the GPU memory from the other devices is normally prohibited, a technology called GPUDirect Support for RDMA [1] allows direct memory access through the PCIe address space under a CUDA 5.0 [2] or above environment with a Kepler-class GPU [3], and we realize TCA communication using this mechanism. In practice, since the direct access over QPI between sockets degrades the performance significantly on Intel E5 CPUs, we assume that PEACH2 only accesses GPU0 and GPU1 in Figure 1. PEACH2 can transfer not only GPU memory but also host memory seamlessly since PEACH2 relies on the PCIe protocol.

III. PEACH2 Chip and Board

HA-PACS/TCA is constructed by using the interface board with PCIe, which employs an FPGA chip referred to as the PCI Express Adaptive Communication Hub version 2 (PEACH2) chip. The PEACH2 chip has four PCIe Gen2 x8 ports. One is dedicated to the connection to the host CPU in order to be treated as the ordinary PCIe device. Another two ports are used to configure the ring topology. A remaining port is used to combine two rings by connecting to the other PEACH2 chips on the neighboring nodes.

PEACH2 provides two types of communication: PIO and DMA. PIO communication is useful for short message transfers, and the CPU can only perform the store operation to remote nodes with minimum low latency. A DMA controller with four channels is also embedded. The DMA controller provides a chaining DMA function, which allows to transfer multiple data segments automatically by hardwired logic according to the chained DMA descriptors. The DMA controller also permits a block-stride transfer which can be specified with a single descriptor.

IV. Basic Performance Evaluation

We evaluate the basic performance of TCA communication using HA-PACS/TCA. We develop two device drivers: the PEACH2 driver for controlling the PEACH2 board and the P2P driver for enabling GPUDirect Support for RDMA. We also evaluate the performance using a CUDA-aware MPI, MVAPICH2-GDR 2.0b for comparison. The detailed environment is summarized in the poster draft, and we use up to 16 nodes as a TCA sub-cluster for this evaluation.

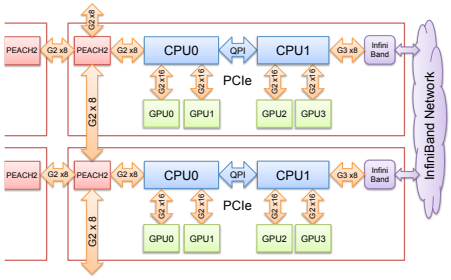

Fig. 1. Block Diagram of the Computation Node in HA-PACS/TCA

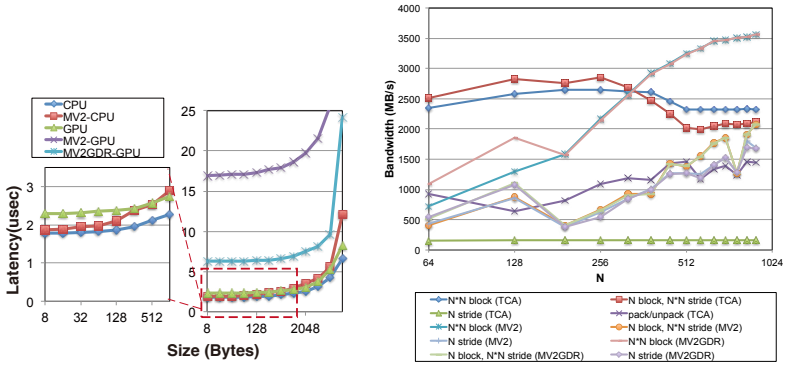

Fig. 2. Performance Comparison among PEACH2 and MPI+InfiniBand; (a) Ping-pong Latency (left), (b) Bandwidth on Halo exchange of N^3 cube (right) [CPU: CPU-to-CPU neighbor comm., GPU: GPU-to-GPU neighbor comm., MV2: MVAPICH2, MV2GDR: MVAPICH2-GDR 2.0a]

Figure 2(a) shows the ping-pong latency in the case of CPU-CPU and GPU-GPU using DMA, under small data size conditions. Using TCA, the minimum latency in the case of CPU-CPU is 1.9 μsec, and that in the case of GPU-GPU is 2.3 μsec. The overhead of GPU communication is 0.4 μsec, and it is caused by read operation from GPU memory. On the other hand, the minimum latency in the case of MVAPICH2 is 17 μsec. TCA result is 7.4 times better than MVAPICH2 in terms of the minimum latency on GPU-to-GPU communication. MVAPICH2-GDR is an improved version for latency using GPUDirect for RDMA on GPU-to-GPU communication. However, the latency is 6 μsec, and TCA performance is still 2.6 times better than MVAPICH2-GDR.

Figure 2(b) shows the performance of the halo exchange pattern in the stencil computation. When the size of cubic array is $N \times N \times N$ with a contiguous memory for k direction, the memory transfer on each halo plane can be described as follows:

- jk-plane: a block transfer with $N \times N$ of block size
- ik-plane: N times of block-stride transfers with N of block size and $N \times N$ of stride size
- ij-plane: $N \times N$ times of stride transfers with a single data block and N of stride size

MPI are measured using `MPI_Type_vector` on the ik- and ij-plane. The maximum bandwidth is 2.7 Gbyte/sec at $N = 192$, and we could obtain better performance than MVAPICH2-GDR at message lengths up to $N = 320$ with 800 Kbytes. The effect of DMA chaining function shows a great performance gain in the case of block stride with N block and N stride (jk-plane) where the performance of TCA is 3x to 6x better than both MVAPICH2 and MVAPICH2-GDR.

V. Himeno Benchmark Results

Lastly, we evaluate Himeno benchmark using TCA communication compared with the MPI version. Himeno benchmark solves the 3-D Poisson's equation using Jacobi iterative method [4], and we newly describe the MPI+CUDA program based on [5] which permits weak-scaling execution only, in order to enable the strong-scaling speedup same as [4].

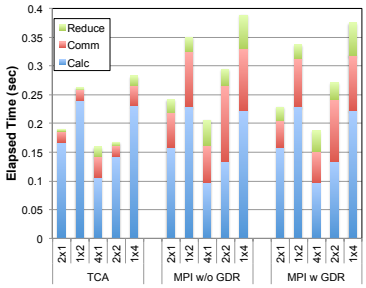

Fig. 3. Result of Himeno benchmark with small size ($i \times j \times k$: $128 \times 64 \times 64$)

Figure 3 shows the results of Himeno benchmarks. $i \times j$ in x-axis denotes the number of the 2-D decompositions. In the case of 2×2, TCA improves the elapsed time with 38% and the performance with 62%. Even in the small size, since the latency impacts the total application performance rather than the bandwidth, TCA can obtain good scaling in proportion of the division number.

Acknowledgment

The present study was supported in part by the JST/CREST program entitled "Research and Development on Unified Environment of Accelerated Computing and Interconnection for Post-Petascale Era."

References

[1] *Developing A Linux Kernel Module Using RDMA For GPUDirect*, NVIDIA Corp. [Online]. Available: http://developer.download.nvidia.com/compute/cuda/5_0/rc/docs/GPUDirect_RDMA.pdf

[2] *NVIDIA CUDA: Compute Unified Device Architecture*, NVIDIA Corp. [Online]. Available: http://developer.nvidia.com/category/zone/cuda-zone

[3] *NVIDIA Tesla Kepler GPU Computing Accelerators*, NVIDIA Corp. [Online]. Available: http://www.nvidia.com/content/tesla/pdf/Tesla-KSeries-Overview-LR.pdf

[4] R. Himeno. The Riken Himeno CFD benchmark. [Online]. Available: http://accc.riken.jp/hpc_e/

[5] E. Phillips and M. Fatica, "Implementing the himeno benchmark with cuda on gpu clusters," in *Proc. of 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS 2010)*, Apr. 2010, pp. 1–10.