Tightly Coupled Accelerators for Minimizing Communication Latency among Accelerators

Toshihiro Hanawa, Yuetsu Kodama, Taisuke Boku, Mitsuhisa Sato

Center for Computational Sciences

University of Tsukuba, Japan
Agenda

- Background
- HA-PACS Project
  - Introduction of HA-PACS / TCA
- Organization of TCA
- PEACH2 Board Designed for TCA
- Preliminary Performance Evaluation
- Summary
GPU Computing and Problems of GPU Cluster

- GPU is widely used for HPC systems thanks to high peak performance / cost ratio and high peak performance / power ratio.

Problems

- Data I/O performance limitation
  - Ex) GPGPU: PCIe Gen2 x16
  - Peak Performance: 8GB/s (I/O) ⇔ 1.31 TFLOPS (NVIDIA K20X)

- Memory size limitation
  - Ex) K20X: 6GByte vs CPU: 128GByte

- Communication between accelerators: no direct path (external)
  Several memory copies ⇒ communication latency via CPU becomes large
  - Ex) GPGPU:
    GPU mem ⇒ CPU mem ⇒ (MPI) ⇒ CPU mem ⇒ GPU mem

Ultra-low latency is important for strong-scaling problems in Exa-scale era.
⇒ Researches for direct communication between GPUs are required.

Our target is developing a direct communication system between external GPUs for a feasibility study for future accelerated computing
HA-PACS Project

- **HA-PACS** (Highly Accelerated Parallel Advanced system for Computational Sciences)
  - 8th generation of PAX/PACS series supercomputer
- Promotion of computational science applications in key areas in our Center
  - Target field: QCD, astrophysics, QM/MM (quantum mechanics / molecular mechanics, bioscience)

HA-PACS is not only a “commodity GPU-accelerated PC cluster” but also experiment platform for direct communication among accelerators.

- Two parts
- HA-PACS *base cluster*
  - for development of GPU-accelerated code for target fields, and performing product-run of them
  - Now in operation since Feb. 2012
- HA-PACS/TCA (TCA = Tightly Coupled Accelerators)
  - for elementary research on new technology for accelerated computing
  - Our original communication chip named “PEACH2”
HA-PACS/TCA (Tightly Coupled Accelerators)

**TCA:** Tightly Coupled Accelerators
- Direct connection between accelerators (GPUs) over the nodes
- Using PCIe as a communication device between accelerator
  - Most accelerator devices and other I/O devices are connected by PCIe as PCIe end-point (slave device)
  - An intelligent PCIe device logically enables an end-point device to directly communicate with other end-point devices

**PEACH2:** PCI Express Adaptive Communication Hub ver. 2
- In order to configure TCA, each node is connected to other nodes through PEACH2 chip.
Design policy of PEACH2

- Sufficient communication bandwidth
  - PCI Express **Gen2 x8, 4 ports**
    - Limitation by chip I/O and board design
    - Each port has the same bandwidth as InfiniBand QDR 4x
  - Sophisticated DMA controller
    - Chaining DMA

- Latency reduction
  - Hardwired logic
  - Efficient mapping of PCIe address area

- **Implement by FPGA with four PCIe Gen.2 IPs**
  - Altera FPGA (Stratix IV GX) is used
TCA node structure

- CPU can uniformly access to GPUs.
- PEACH2 can access every GPUs
  - Kepler architecture + CUDA 5.0 “GPUDirect Support for RDMA”
  - Performance over QPI is quite bad. => support only for GPU0, GPU1
- Connect among 3 nodes

- This configuration is similar to HA-PACS base cluster except PEACH2.
  - All the PCIe lanes (80 lanes) embedded in CPUs are used.

![Diagram of TCA node structure]
Overview of PEACH2 chip

- Root and EndPoint must be paired according to PCIe spec.
- Port N: connected to the host
- Port E and W: form the ring topology
- Port S: connected to the other ring
  - Selectable between Root and Endpoint
- Write only except Port N
  - Instead, “Proxy write” on remote node realizes pseudo-read.
Communication by PEACH2

- PIO
  - CPU can store the data to remote node directly using mmap.

- DMA
  - Chaining DMA function
    - DMA requests are described as the DMA descriptors in the descriptor table.
    - DMA transactions are operated automatically according to the DMA descriptors by hardware.
  - Implementation of DMAC is partially used on the IP contained in the PCIe reference design by Altera.
    - In this design, the internal memory of PEACH2 must be specified as the source or destination address, and two phase operations are required via the internal memory of PEACH2.

=> New implementation has been developed after this work.
PEACH2 board (Production version for HA-PACS/TCA) **New!**

- PCI Express Gen2 x8 peripheral board
  - Compatible with PCIe Spec.

Side View

Top View
PEACH2 board (Production version for HA-PACS/TCA) **New!**

- Main board + sub board
- **FPGA (Altera Stratix IV 530GX)**
- Most part operates at 250 MHz (PCIe Gen2 logic runs at 250MHz)
- **Power supply for various voltage**
- **PCI Express x8 card edge**
- **PCIe x16 cable connector**
- **PCIe x8 cable connector**
- **DDR3-SDRAM**
Preliminary Performance Evaluation

Environment

- CPU: Intel Xeon-E5 (SandyBridge EP) 2.6GHz x2socket
- MB: SuperMicro X9DRG-QF, Intel S2600IP
- Memory: DDR3 128GB
- OS: CentOS 6.3 (kernel 2.6.32-279.x.1.el6.x86_64)
- GPU: NVIDIA K20, GDDR5 5GB
- CUDA: 5.0, NVIDIA-Linux-x86_64-304.xx
- PEACH2 prototype board
Evaluation items

- DMA basic performance within a node
  - PCIe performance on the FPGA
  - Performance of the DMA controller (DMAC)

- Performance among different nodes
  - Latency by PIO
  - Bandwidth by DMA

- Ping-pong performance using new DMAC  New!
  - Evaluation as application
  - Comparison with CUDA and MVAPICH2 for GPU-GPU communication

- In order to access GPU memory by the other device, “GPU Direct support for RDMA” in CUDA5 API is used.
  - Special driver named “TCA p2p driver” to enable memory mapping is developed.

- “PEACH2 driver” to control the board is also developed.
DMA basic performance (within a node)

- The results from 255 times transfer with the same address
- 93% of theoretical peak performance

The results from a fixed data size of 4 Kbytes for various burst counts
DMA basic performance (within a node)

- The results from 255 times transfer with the same address
- 93% of theoretical peak performance
- The results from a fixed data size of 4 Kbytes for various burst counts

Max Payload Size = 256byte
Theoretical peak \(3.66\) Gbyte/s = 4Gbyte/sec \(\times\) 256 / (256 + 16 + 2 + 4 + 1 + 1)

DMA read performance from GPU is saturated up to 870Mbyte/s.
⇒ PCIe switch embedded in SandyBridge doesn’t have enough resources.
In IvyBridge, the performance may be improved ...
Programming model for TCA cluster

Basic Idea:

- **cudaMemcpyPeer()** under Unified Virtual Memory environment is expanded to the direct communication among GPUs over the node on TCA cluster.
  - User can write the program for TCA more easily than for MPI
  - Ex) `tcaMemcpy(nodeid, dest, source, size, flag);`

- **Suitable for Stencil computation**
  - Good performance at nearest neighbor communication due to direct network
  - Ex) Under 3-D stencil computation
  - Chaining DMA can bundle many DMA requests, such as data transfers for every “Halo” planes composed of block-stride arrays, to a single DMA operation on the host.
  - In each iteration, DMA descriptors can be reused and only a DMA kick operation is needed
Related Work

- **Non Transparent Bridge (NTB)**
  - NTB appends the bridge function to a downstream port of the PCI-E switch.
  - Inflexible, the host must recognize during the BIOS scan
  - It is not defined in the standard of PCI-E and is incompatible with the vendors.

- **APEnet+ (Italy)**
  - GPU direct copy using Fermi GPU, different protocol from TCA is used.
  - Latency between GPUs is around $5us$?
  - Original 3-D Torus network, QSFP+ cable

- **MVAPICH2 + GPUDirect**
  - CUDA5, Kepler
  - Latency between GPUs is reported as $6.5us$.
  - Currently not released (latter half of 2013)
Current status of PEACH2 board

- PEACH2 boards are ready for TCA.
- Evaluation using 8 node cluster is ongoing.
  - NVIDIA Tesla K20 is used.
  - Communications among nodes are confirmed.
- Development and Improvement
  - PEACH2 driver
  - NVIDIA P2P driver
  - TCA Library for programmers
Current status and future schedule of HA-PACS/TCA

- Procurement of HA-PACS/TCA as an expansion of HA-PACS system will be over next week!
- 64 nodes or more, 200TFlops or more (TBD)
  - The performance is expected to achieve beyond 1PFlops with base cluster.
  - HA-PACS/TCA will be installed on the end of Oct. 2013.
Summary

- **HA-PACS: Next Gen. GPU Cluster in Univ. of Tsukuba**
  - Base Cluster: Large scale application development using GPU acceleration
  - HA-PACS/TCA: Development of direct communication among accelerators as an element technology becomes a basic technology for next gen’s accelerated computing in exa-scale era.

- **PEACH2 board: Implementation for realizing TCA using PCIe technology**
  - Bandwidth: max. 3.3~3.5Gbyte/sec (over 90~95% of theoretical peak)
  - Latency: min. 0.8~0.9us
  - GPU-GPU communication over the nodes can be demonstrated.
  - By the ping-pong program, PEACH2 can achieve lower latency than existing technology, such as CUDA and MVAPICH2 in small data size.
  - Under development of driver and library

- **HA-PACS/TCA will be installed on the end of Oct. 2013.**
  - Actual proof system of TCA technology with over 64 nodes
  - HA-PACS will become over 1PFlops system combined with HA-PACS base cluster.
  - Development of the application using TCA, and production-run
    - In particular, performance comparison between TCA and InfiniBand