HA-PACS/TCA: Low Latency Communication among Accelerators and Its Experimental Platform

Taisuke Boku

Deputy Director

Center for Computational Sciences

University of Tsukuba

taisuke@cs.tsukuba.ac.jp

Outline of talk

- HA-PACS Project overview
- HA-PACS Base Cluster
- TCA (Tightly Coupled Accelerators)
- Preliminary Performance Result
- Summary

HA-PACS Project

Toward Exascale

- Accelerated computing technology
  - GPU, many-core, or any other style of accelerating devices
  - Application characteristics is limited and not all-mighty for general purpose
    - fixed pattern of computation with large degree of parallelism in a chip
    - performance gap between computation and communication
  - We need algorithm-level development for large scale accelerated applications
- Computation and Communication
  - Bandwidth is (almost) catching up with computing performance
  - Latency is the big issue
  - Multiple layers of hardware/software stack is serious problem
- Weak scaling to Strong scaling
  - time to solution
  - memory size

Project plan of HA-PACS

- **HA-PACS (Highly Accelerated Parallel Advanced system for Computational Sciences)**
- Accelerating critical problems on various scientific fields in Center for Computational Sciences, University of Tsukuba
  - The target application fields will be partially limited
  - Current target: QCD, Astro, QM/MM (quantum mechanics / molecular mechanics, for life science)
- Two parts
  - HA-PACS *base cluster*:
    - for development of GPU-accelerated code for target fields, and performing product-run of them
  - HA-PACS/*TCA*: (TCA = Tightly Coupled Accelerators)
    - for elementary research on new technology for accelerated computing
    - Our original communication system based on PCI-Express named “PEARL”, and a prototype communication chip named “PEACH2”

GPU Computing: current trend of HPC

- GPU-base systems in TOP500 on Jun. 2013
  - 2nd Titan (Rpeak=27.1 PFLOPS)
  - 10th Tienha-1A (Rpeak=4.70 PFLOPS)
  - 16th Nebulae (Rpeak=2.98 PFLOPS)
  - 21st TSUBAME2.0 (Rpeak=2.29 PFLOPS)
  - (1st Tianhe-2 Rpeak=54.9 PFLOPS)
- Features
  - high peak performance / cost ratio
  - high peak performance / power ratio
  - large scale applications with GPU acceleration are rare to run in daily-productive level

  ⇒ **Our First target is to develop large scale applications accelerated by GPU in real computational sciences**

Issues of GPU Cluster

- Problems of GPGPU for HPC
  - Data I/O performance limitation
    - Ex) GPGPU: PCIe gen2 x16
    - Peak Performance: 8GB/s (I/O) ⇔ 665 GFLOPS (NVIDIA M2090)
  - Memory size limitation
    - Ex) M2090: 6GByte vs CPU: 4 – 128 GByte
  - Communication between accelerators: no direct path (external)
    ⇒ communication latency via CPU becomes large
    - Ex) GPGPU:
      GPU mem ⇒ CPU mem ⇒ (MPI) ⇒ CPU mem ⇒ GPU mem
- Researches for direct communication between GPUs are required

Our another target is developing a direct communication system between external GPUs for a feasibility study for future accelerated computing

Project Formation

- HA-PACS (Highly Accelerated Parallel Advanced system for Computational Sciences)
  - Apr. 2011 – Mar. 2014, 3-year project (the system will be maintain until Mar. 2016), lead by Prof. M. Sato
    "Advanced research and education on computational sciences driven by exascale computing technology", $4.5M supported by MEXT
  - Project Office for Exascale Computing System Development（Leader: Prof. T. Boku）
    - Develop two types of GPU cluster systems: 15 members
  - Project Office for Exascale Computational Sciences (Leader: Prof. M. Umemura）
    - Develop large scale GPU applications : 14 members

    Elementary Particle Physics, Astrophysics, Bioscience, Nuclear Physics, Quantum Matter Physics, Global Environmental Science, Computational Informatics, High Performance Computing Systems

HA-PACS Base Cluster

HA-PACS base cluster (Feb. 2012)

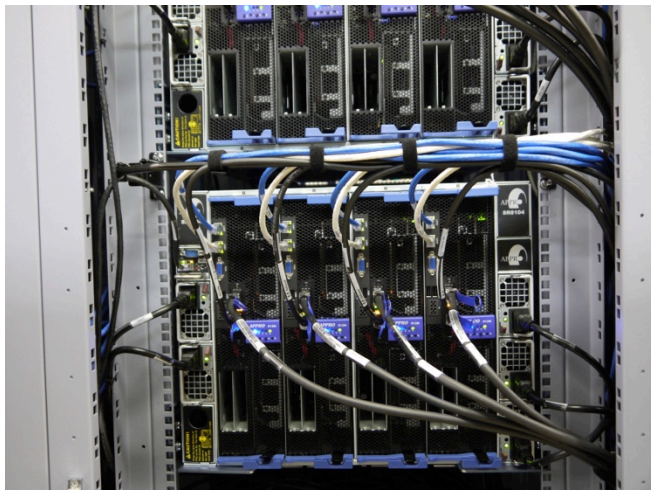
System based on Appro's GreenBlade solution

HA-PACS base cluster

Front view

Side view

HA-PACS base cluster



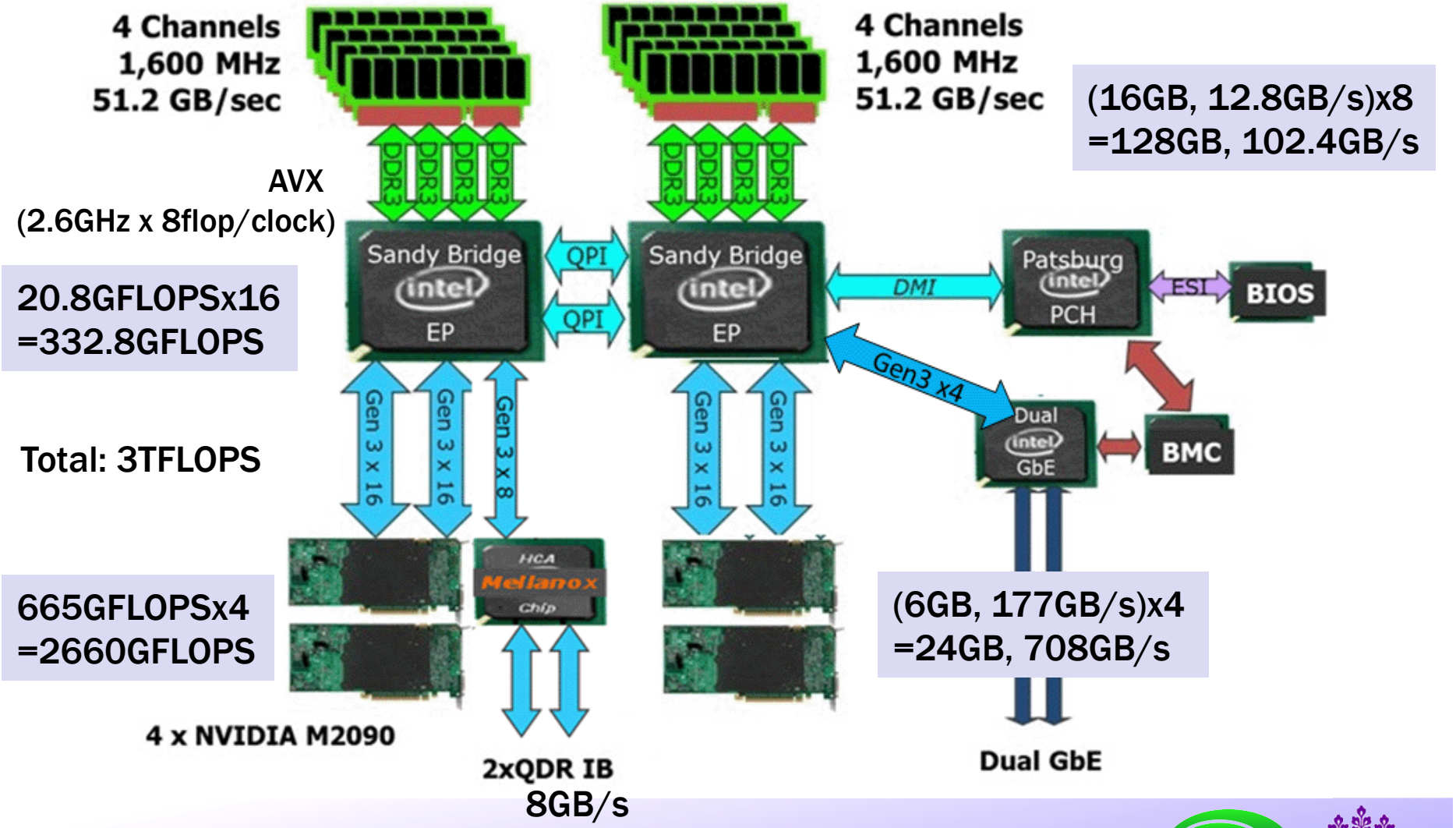
Rear view of one blade chassis with 4 blades

Front view of 3 blade chassis

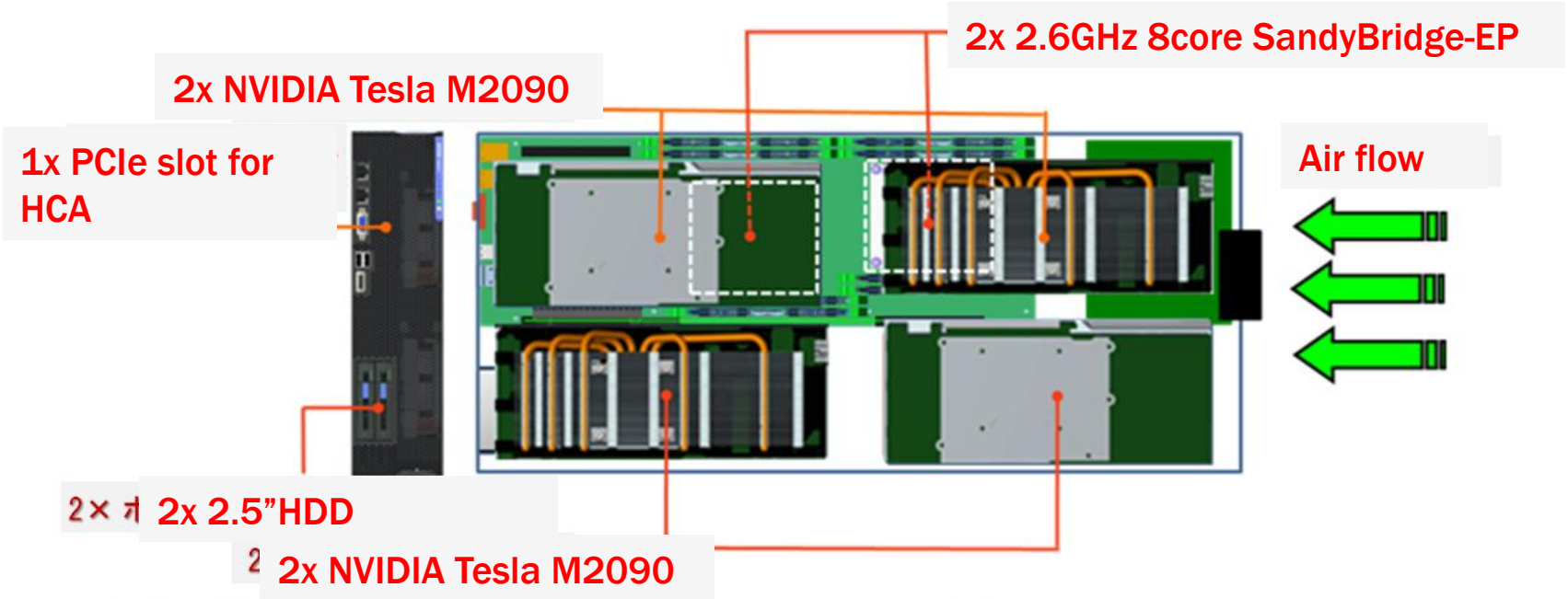
Rear view of Infiniband switch and cables
(yellow=fibre, black=copper)

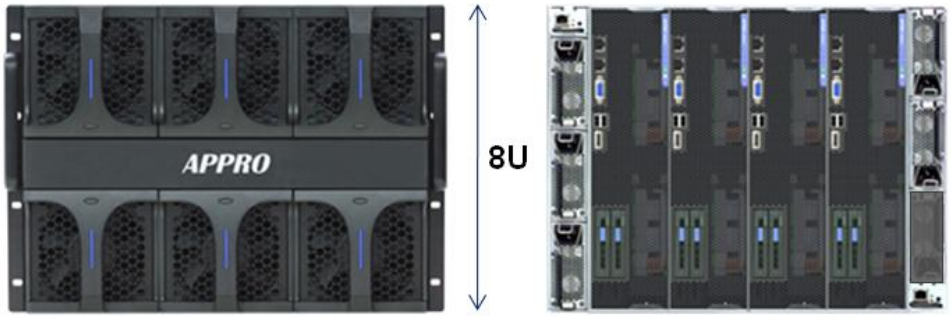
HA-PACS base cluster node

HA-PACS: base cluster (computation node)



HA-PACS: base cluster unit（blade node）





Front view

Rear view

Power Supply Unit and Fan
- 8U enclosure
- 4 nodes
- 3 PSU(Hot Swappable)
- 6 Fans(Hot Swappable)

HA-PACS: base cluster unit（CPU）

- **Intel Xeon E5 (SandyBridge-EP) x 2**
  - 8 cores/socket (16 cores/node) with 2.6 GHz
  - AVX (256-bit SIMD) on each core
    ⇒ peak perf./socket = 2.6 x 4 x 2 = 166.4 GFLOPS
    ⇒ pek perf./node = 332.8 GFLOPS
  - Each socket supports up to 40 lanes of PCIe gen3
    ⇒ great performance to connect multiple GPUs without I/O performance bottleneck
    ⇒ current NVIDIA M2090 supports just PCIe gen2, but next generation (Kepler) will support PCIe gen3
  - M2090 x4 can be connected to 2 SandyBridge-EP still remaining PCIe gen3 x8 x2
    ⇒ Infiniband QDR x 2

HA-PACS base cluster (System)

- Number of nodes = 268
- Performance
  - CPU: 332.8 GFLOPS x 268 = 89 TFLOPS
  - GPU: 2660 GFLOPS x 268 = 713 TFLOPS
  - TOTAL: 802 TFLOPS
- File server
  - Lustre file system over RAID-6, direct access from all nodes
  - 500 TByte
- Power consumption
  - Peak: 408 kW (1.4kW/node)
- HPL: 421.6 TFLOPS (#41 in TOP500 at first appearance)

Basic performance data

- MPI pingpong
  - 6.4 GB/s ($N_{1/2}$= 8KB)
  - with dual rail Infiniband QDR (Mellanox ConnectX-3)
  - actually FDR for HCA and QDR for switch
- PCIe benchmark (Device -> Host memory copy), aggregated perf. for 4 GPUs simultaneously
  - 24 GB/s ($N_{1/2}$= 20KB)
  - PCIe gen2 x16 x4, theoretical peak = 8 GB/s x4 = 32 GB/s
- Stream (memory)
  - 74.6 GB/s
  - theoretical peak = 102.4 GB/s

HA-PACS/TCA

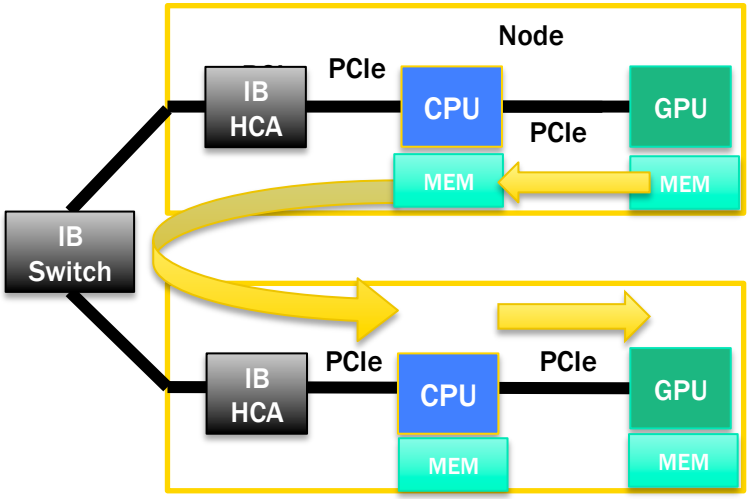
HA-PACS: TCA (Tightly Coupled Accelerator)

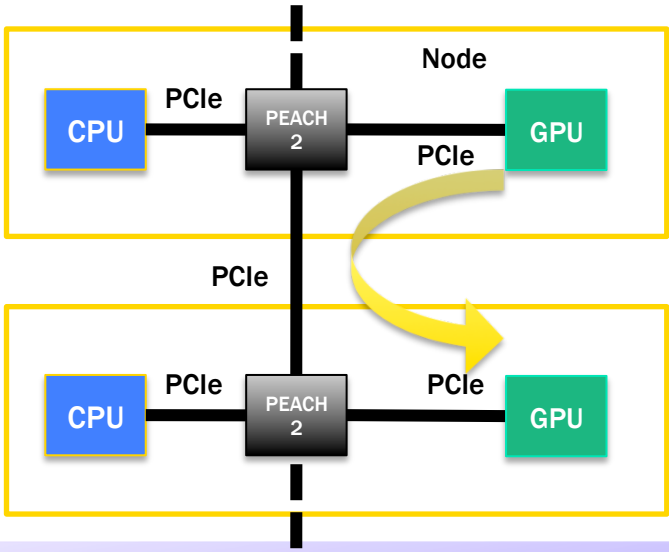
- **TCA: Tightly Coupled Accelerator**
  - Direct connection between accelerators (GPUs)
  - Using PCIe as a communication device between accelerator
    - Most acceleration device and other I/O device are connected by PCIe as PCIe end-point (slave device)
    - An intelligent PCIe device logically enables an end-point device to directly communicate with other end-point devices
- **PEARL: PCI Express Adaptive and Reliable Link**
  - We already developed such PCIe device (PEACH, PCI Express Adaptive Communication Hub) on JST-CREST project "low power and dependable network for embedded system"
  - It enables direct connection between nodes by PCIe Gen2 x4 link

⇒ **Improving PEACH for HPC to realize TCA**

HA-PACS/TCA (Tightly Coupled Accelerator)

- **True GPU-direct**
  - current GPU clusters require 3-hop communication (3-5 times memory copy)
  - For strong scaling, inter-GPU direct communication protocol is needed for lower latency and higher throughput
- Enhanced version of PEACH
  ⇒ ***PEACH2***
  - x4 lanes -> x8 lanes
  - hardwired on main data path and PCIe interface fabric

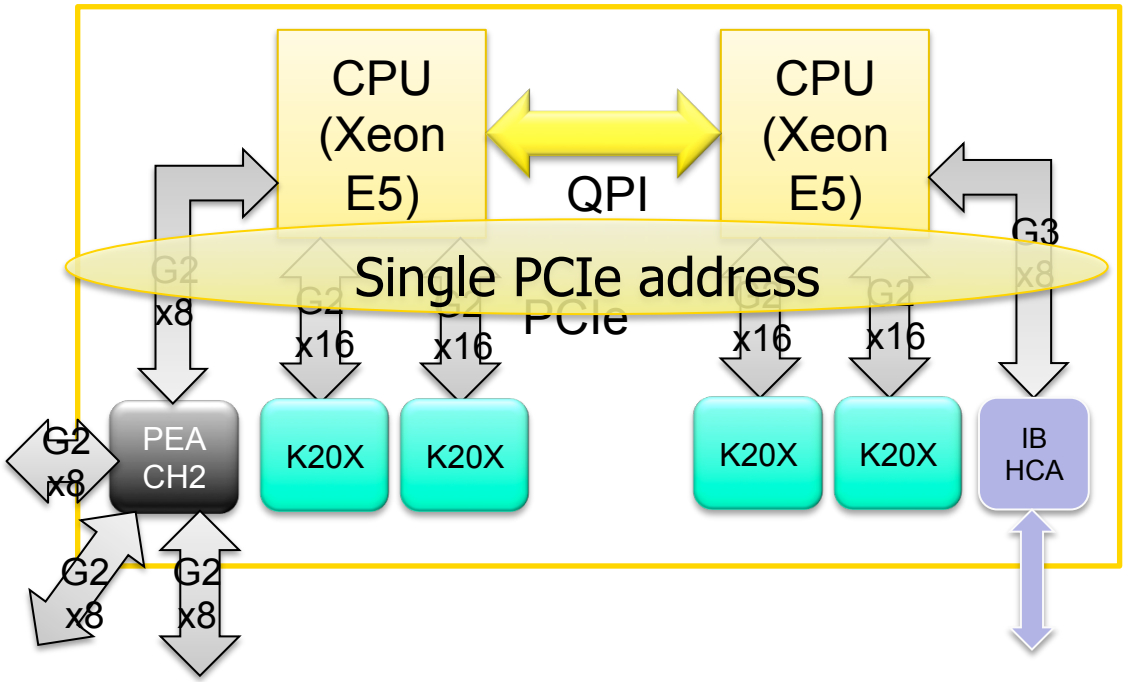




TCA node structure

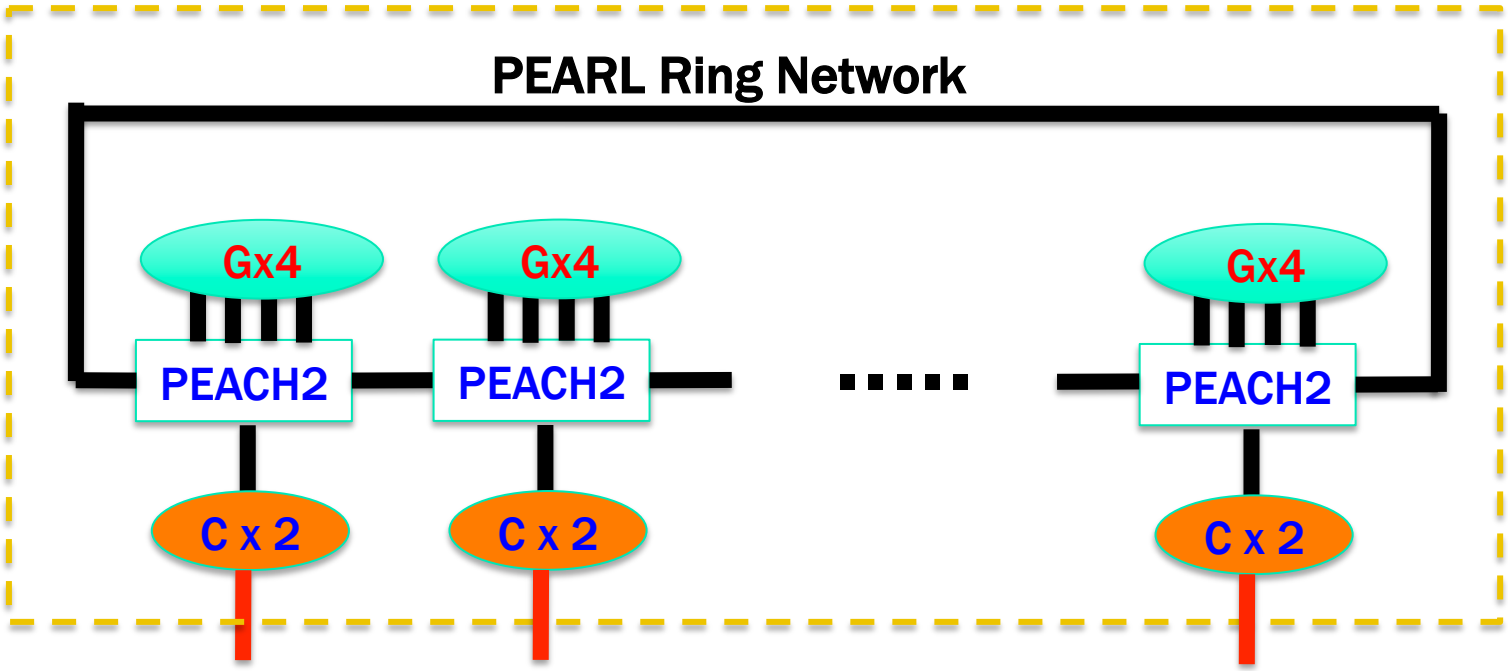
- CPU can uniformly access to GPUs.
- PEACH2 can access every GPUs
  - Kepler architecture + CUDA 5.0 “GPUDirect Support for RDMA”
  - Performance over QPI is quite bad.
    => support only for two GPUs on the same socket
- Connect among 3 nodes

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



HA-PACS/TCA

Node Cluster = NC

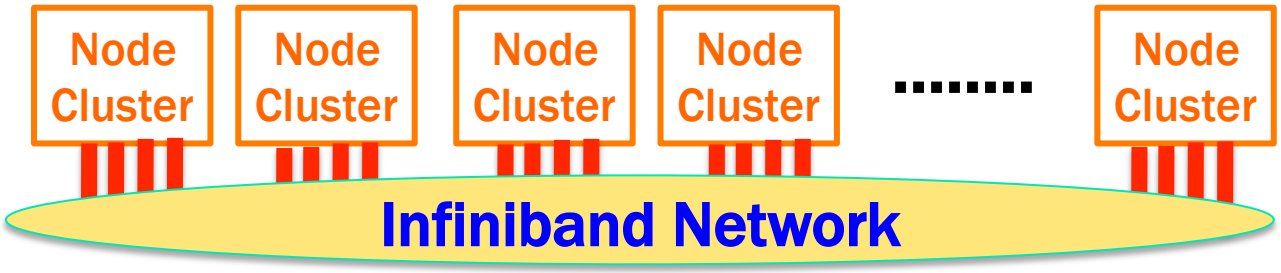


Node Cluster with 16 nodes
- GPUx64 (G)
- CPUx32 (C)
- GPU comm with PCIe
- IB link / node
- CPU: Xeon E5
- GPU: Kepler

Infiniband Link

- High speed GPU-GPU comm. by PEACH within NC (PCI-E gen2x8 = 5GB/s/link)
- Infiniband QDR (x2) for NC-NC comm. (4GB/s/link)

4 NC with 16 nodes, or 8 NC with 8 nodes = 360 TFLOPS extension to base cluster



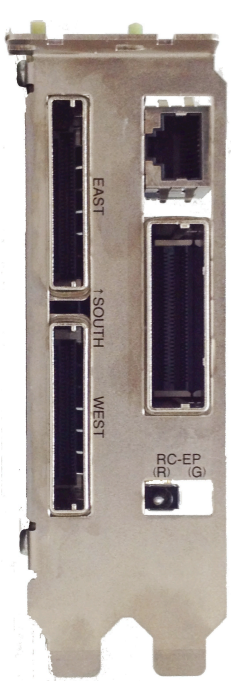
Communication by PEACH2

- 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.

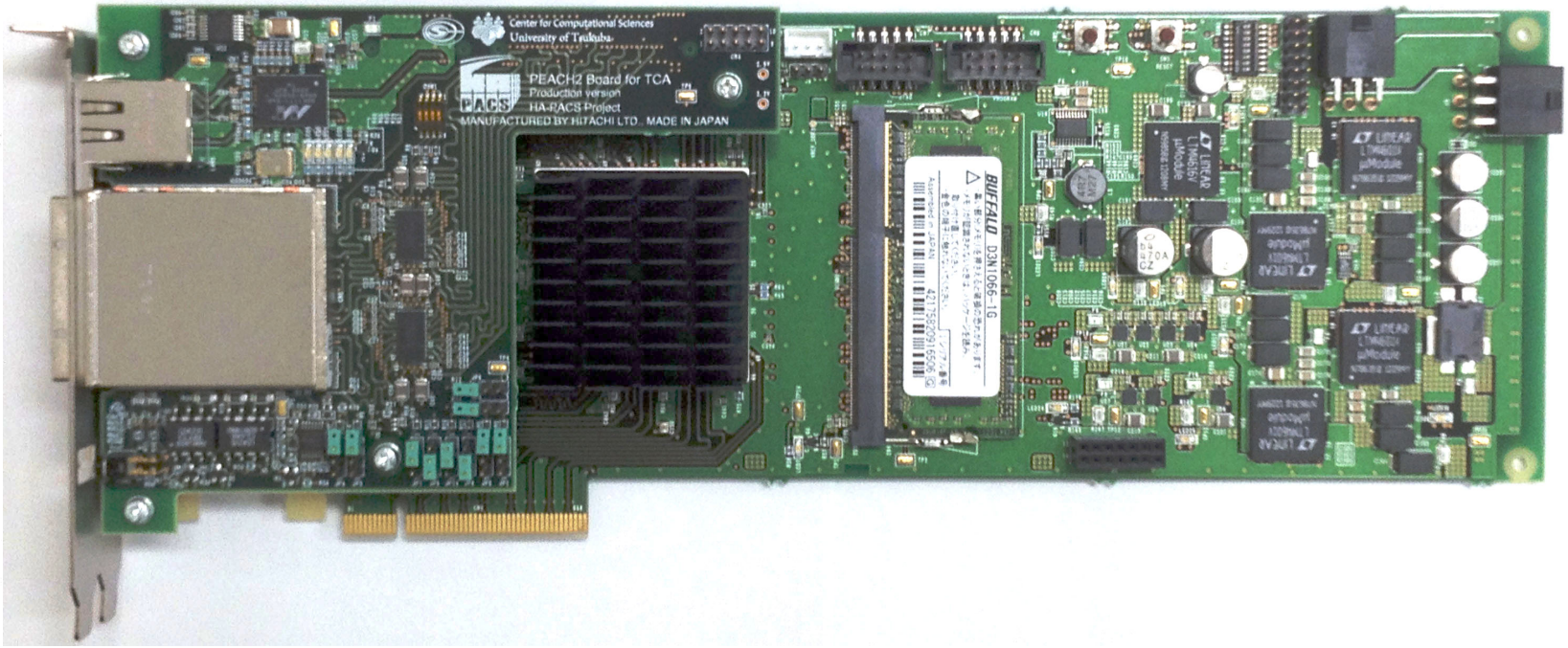
    => New implementation has been developed after this work.

PEACH2 board

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

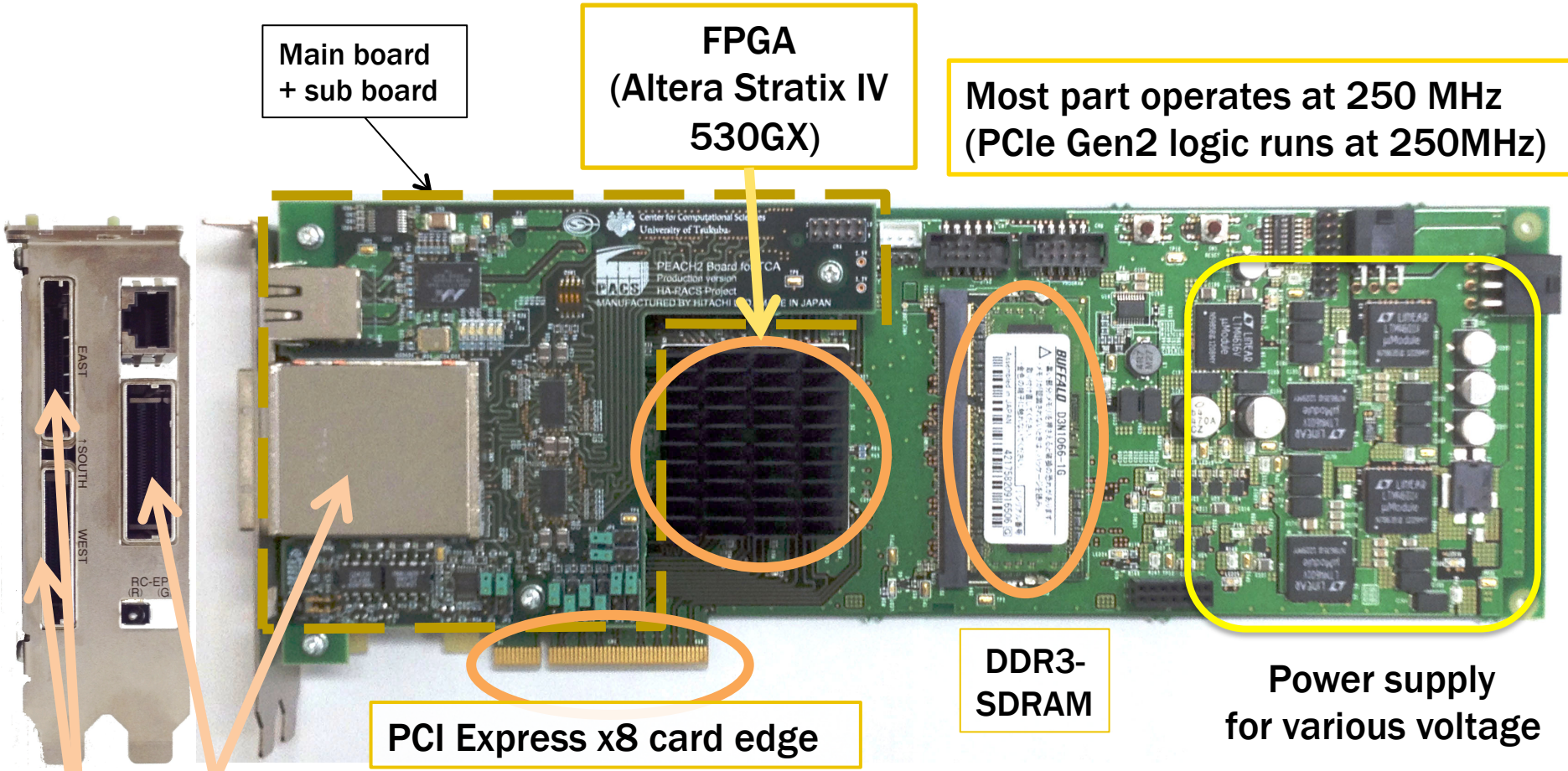


Side View

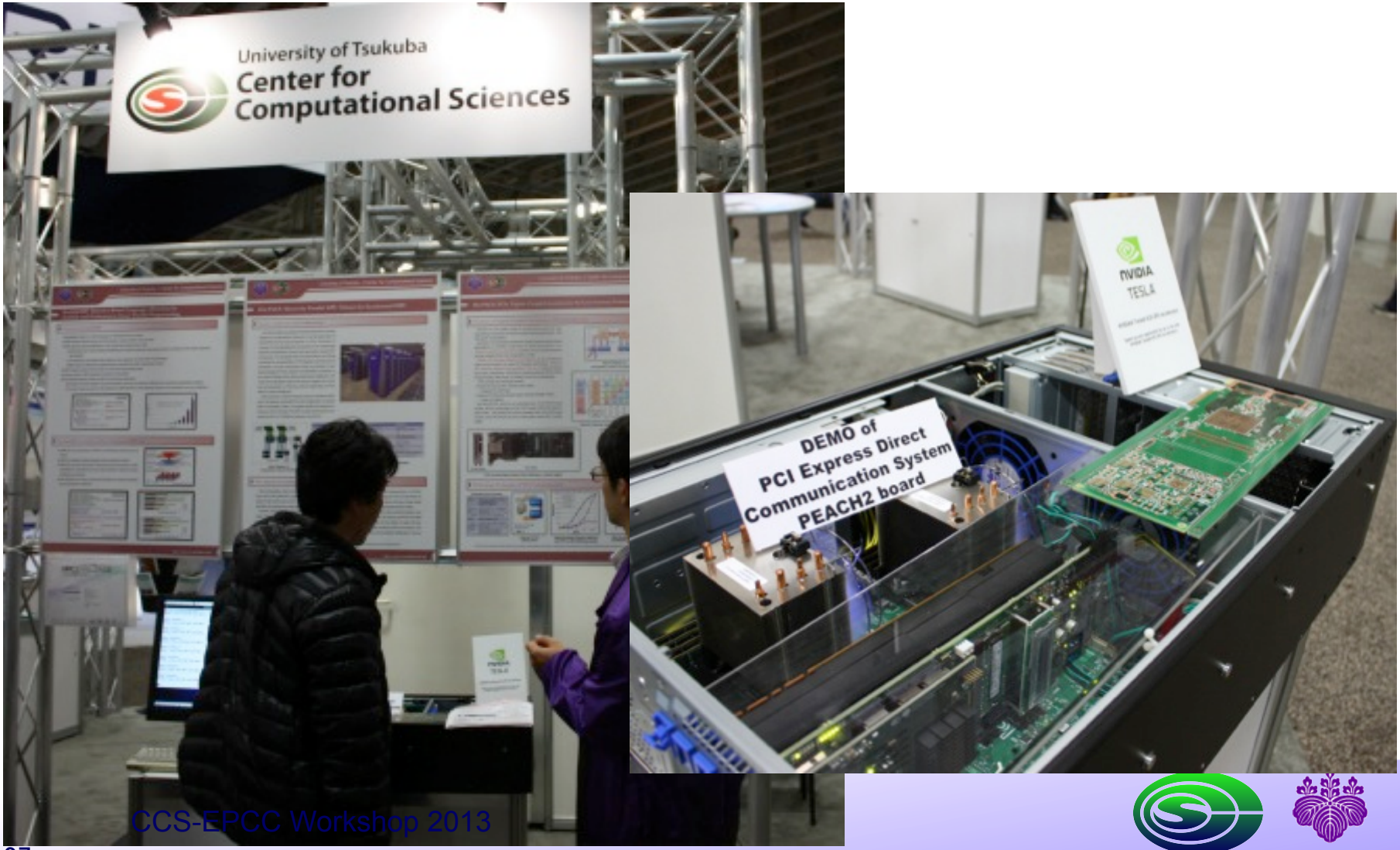


Top View

PEACH2 board



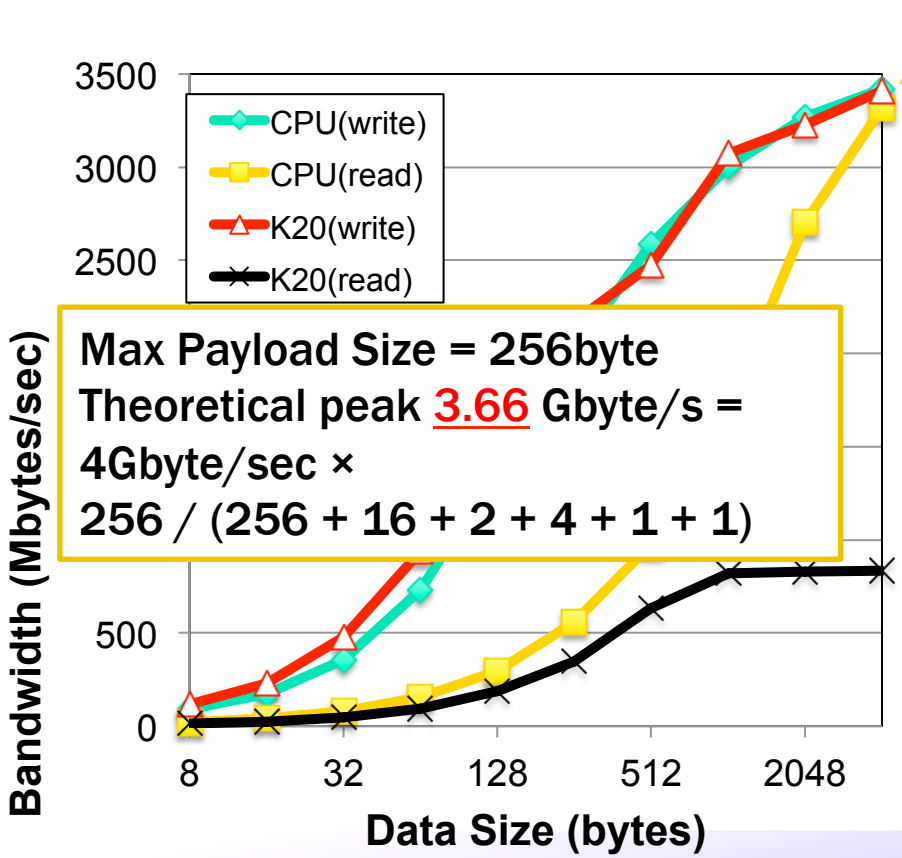
Demonstration at SC12



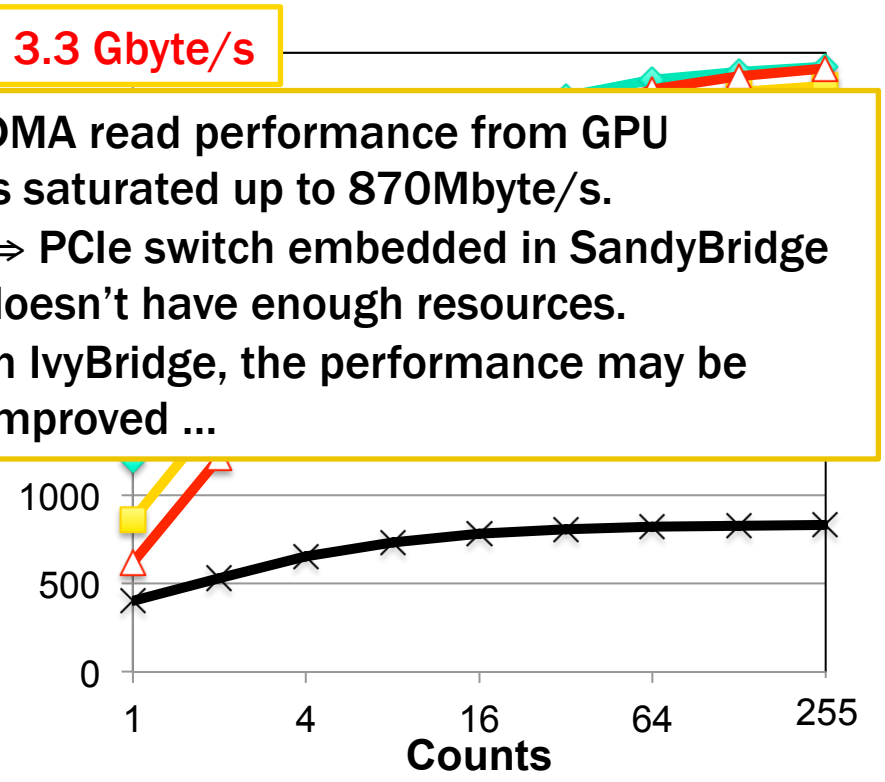
DMA basic performance（within a node）

- 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 ×
256 / (256 + 16 + 2 + 4 + 1 + 1)

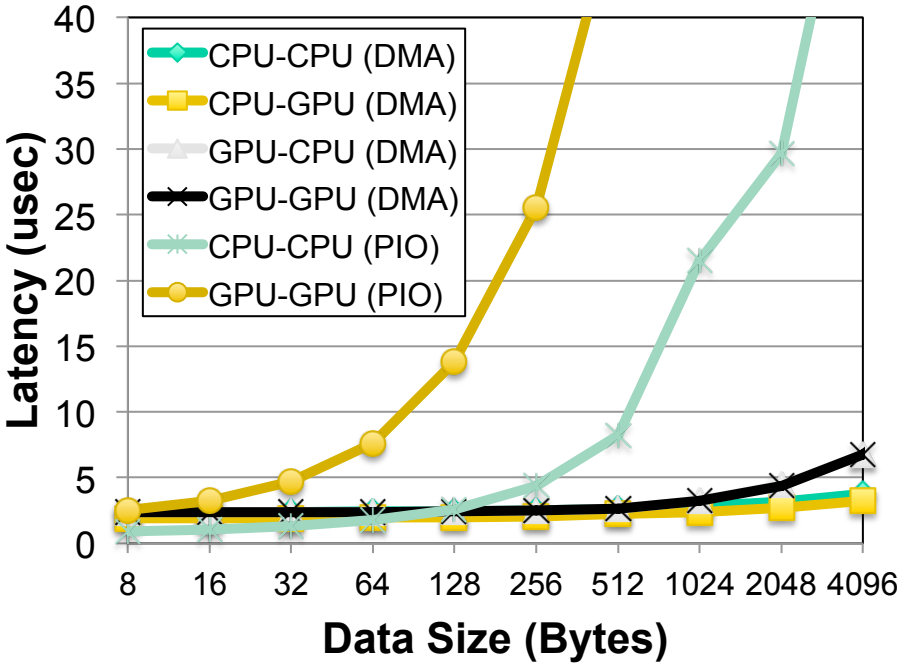


DMA read performance from GPU
is saturated up to 870Mbyte/s.
⇒ PCIe switch embedded in SandyBridge doesn't have enough resources.
In IvyBridge, the performance may be improved ...

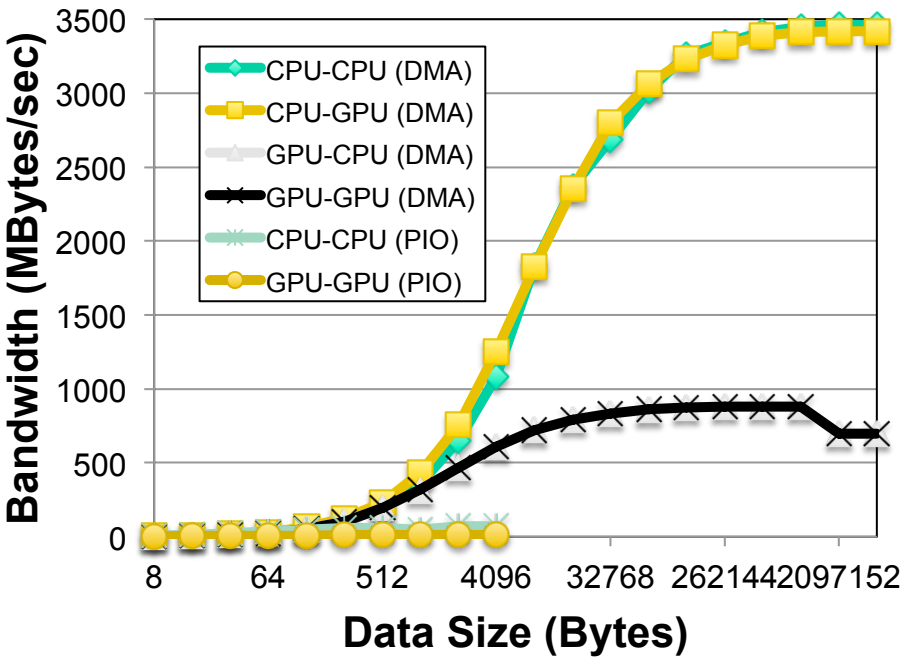
Ping-pong communication (user space)

- **Latency**

  DMA: CPU-to-*: 1.9us –
  GPU-to-*: 2.4us –
  PIO: CPU-to-CPU: 0.9us –
  GPU-to-GPU: 2.5us –



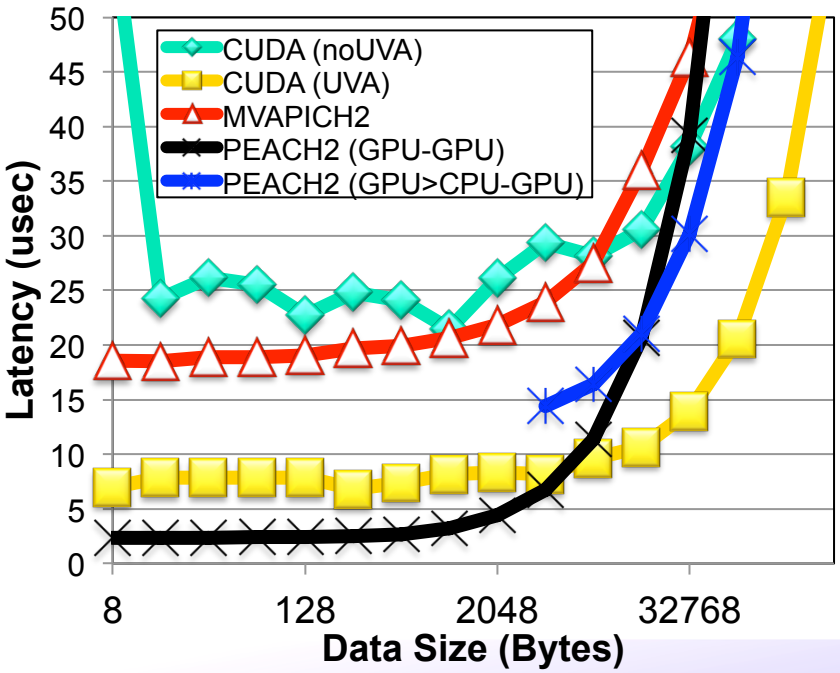
- **Bandwidth**
  - CPU-to-CPU: Max. 3.48GB/s => 95% of peak performance
  - GPU-to-GPU: Max. 880MB/s



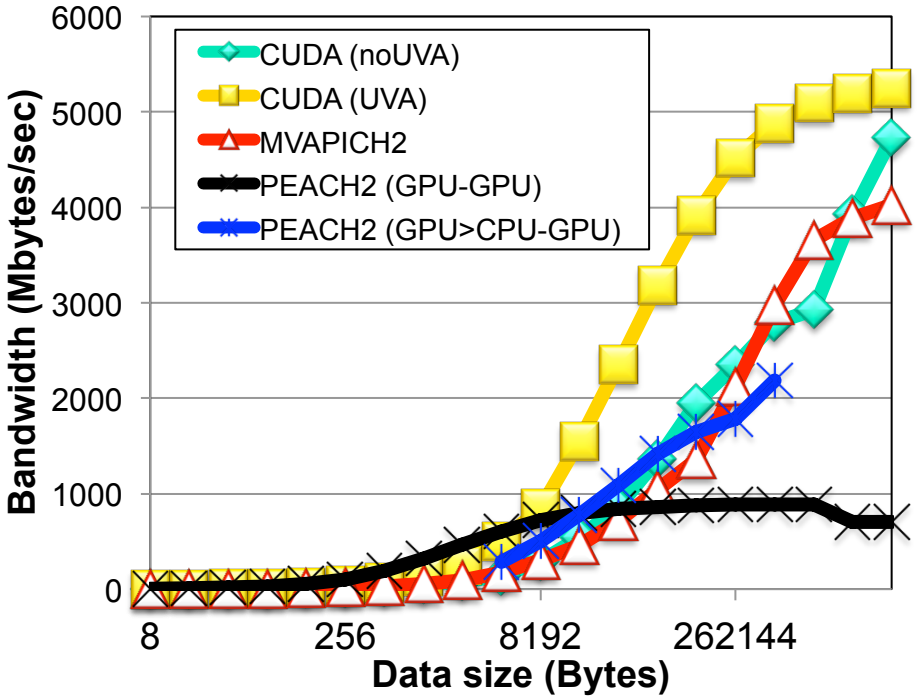
Performance Comparison between TCA and Conventional Technologies

Ping-pong between GPU device memories

- **CUDA: cudaMemcpy() between GPUs within a node**
  - No-UVA: without Unified Virtual Address
  - UVA: with Unified Virtual Address
- **MVAPICH2: 1.9b, MV2_USE_CUDA=1**
  - **InfiniBand FDR10 (Mellanox Connect-X3 directly connected by cable)**



- **The performance of PEACH2 is better than that of CUDA, MVAPICH2.**
  - Up to 4Kbytes
  - **Better than the performance within a node by CUDA**



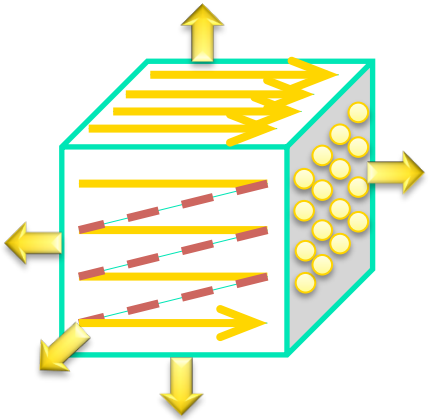
Programming model for TCA cluster

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

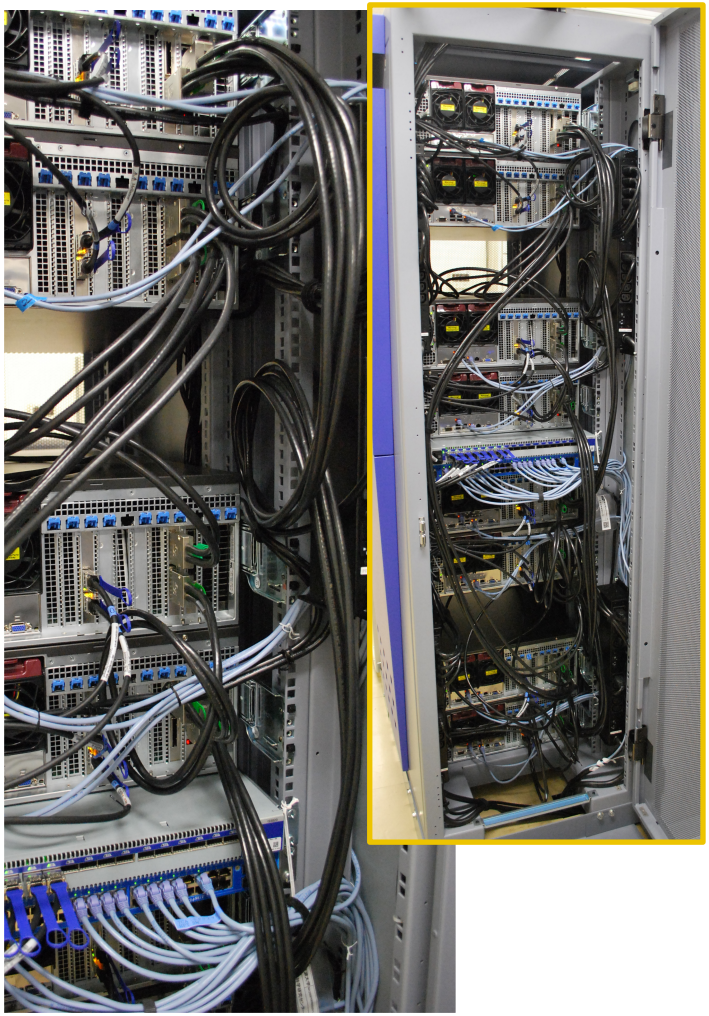
  Ex) Under 3-D stencil computation
  - Chaining DMA can bundle many DMA requests, such as data transfers for every “Halo” planes composed of block-stride arrays, to a single DMA operation on the host.
  - In each iteration, DMA descriptors can be reused and only a DMA kick operation is needed



Bundle to 1 DMA

Current status of PEACH2 board

- PEACH2 boards are ready for TCA.
- Evaluation using 8 node cluster is ongoing.
  - NVIDIA Tesla K20
  - Communications among nodes are confirmed
- 64 node additional cluster with 2x IvyBrid and 4x K20X per node will be installed on Oct. 2013 as HA-PACS/TCA
  ⇒ attached to Base Cluster
- Development and Improvement
  - PEACH2 driver
  - NVIDIA P2P driver
  - TCA Library for programmers



Summary

- Toward exascale computing, computation and communication must be tightly coupled and cooperated especially for strong scaling computation
- Effort on algorithm-level development on applications and very low latency communication system development is required
- HA-PACS focuses on both issues based on large scale commodity GPU cluster and R&D of TCA for “active communication” on accelerators including GPU, MIC, etc.
- FPGA implementation of PEACH2 is undergoing toward HA-PACS/TCA installation on Oct. 2013 with 362 TFLOPS added to HA-PACS/Base which makes total performance with 1.16 PFLOPS
- Currently GTC-P in XMP(-dev) version is coded in Tsukuba