Multi-Hybrid Accelerated Supercomputer: The new system of CCS

Taisuke Boku

HPC System Division, Center for Computational Sciences

University of Tsukuba

A part of slides are based on research collaboration with
N. Fujita, Y. Yamaguchi, R. Kobayashi, Y. Ohbata, M. Umemura and K. Yoshikawa

Outline

- Supercomputer history of CCS
- Next generation's accelerated computing system
- FPGA for HPC as large scale parallel system
- AiS (Accelerator in Switch) concept
- PACS-X Project and PPX
- Ongoing research on PPX
- Brief summary of new machine
- Summary

Our History and Current Machines

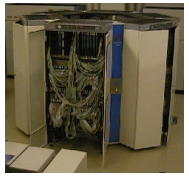
History of PAX (PACS) series at U. Tsukuba

- 1977: research started by T. Hoshino and T. Kawai
- 1978: PACS-9 (with 9 nodes) completed
- 1996: CP-PACS, the first vendor-made supercomputer at CCS, ranked as #1 in TOP500

1978
1st gen: PACS-9

1980
2nd gen. PACS-32

1989
5th gen, QCDPAX



1996
6th gen: CP-PACS
Ranked #1 in TOP500

2006
7th gen: PACS-CS

2012~2013
8th gen: GPU cluster HA-PACS

Year	Name	Performance
1978	PACS-9	7 KFLOPS
1980	PACS-32	500 KFLOPS
1983	PAX-128	4 MFLOPS
1984	PAX-32J	3 MFLOPS
1989	QCDPAX	14 GFLOPS
1996	CP-PACS	614 GFLOPS
2006	PACS-CS	14.3 TFLOPS
2012~13	HA-PACS	1.166 PFLOPS
2014	COMA (PACS-IX)	1.001 PFLOPS

- *co-design* by computer scientists and computational scientists toward “practically high speed computer”
- Application-driven development
- Sustainable development experience

Two streams of supercomputers in CCS (~ FY2017)

- Peak performance for advanced users: HA-PACS (PACS-VIII)
  - GPU cluster + originally developed inter-GPU communication system TCA (Tightly Coupled Accelerator)
  - Accelerated computing with PFLOPS class performance for code development and product run
  - Professional users rather than novice users
- Easy programming for general users: COMA (PACS-IX) (→ Oakforest-PACS)
  - Many core architecture for general programming (OpenMP + MPI) toward next generation's many-core (Intel Knights Corner)
  - Leading system toward Oakforest-PACS (by JCAHPC: joint organization with U. Tsukuba and U. Tokyo) which is equipped with Intel Knights Landing
  - Wide area and general users with ordinary programming
- HA-PACS retired on March 2018 and COMA will retire on March 2019

Next Generation's Accelerated Computing

Accelerators in HPC

- Traditionally...
  - Cell Broadband Engine, ClearSpeed, GRAPE. ...
  - then GPU (most popular)
- Is GPU perfect ?
  - good for many applications (replacing vector machines)
  - depending on very wide and regular computation
    - large scale SIMD (STMD) mechanism in a chip
    - high bandwidth memory (GDR5, HBM) and local memory
  - bad for
    - not enough parallelism
    - not regular computation (warp spliting)
    - frequent inter-node communication (kernel switch, go back to CPU)

FPGA in HPC

- Goodness of recent FPGA for HPC
  - True codesigning with applications (essential)
  - Programmability improvement: OpenCL, other high level languages
  - High performance interconnect: 40Gb~100Gb
  - Precision control is possible
  - Relatively low power
- Problems
  - Programmability: OpenCL is not enough, not efficient
  - Low standard FLOPS: still cannot catch up to GPU
    -> “never try what GPU works well on”
  - Memory bandwidth: 2-gen older than high end CPU/GPU
    -> be improved by HBM (Stratix10)

Simple pros/cons

	performance (FLOPS)	external communication (sec, B/s)	programming cost
CPU	△	○	◎
GPU	◎	△	○
FPGA	○	◎	×→△?

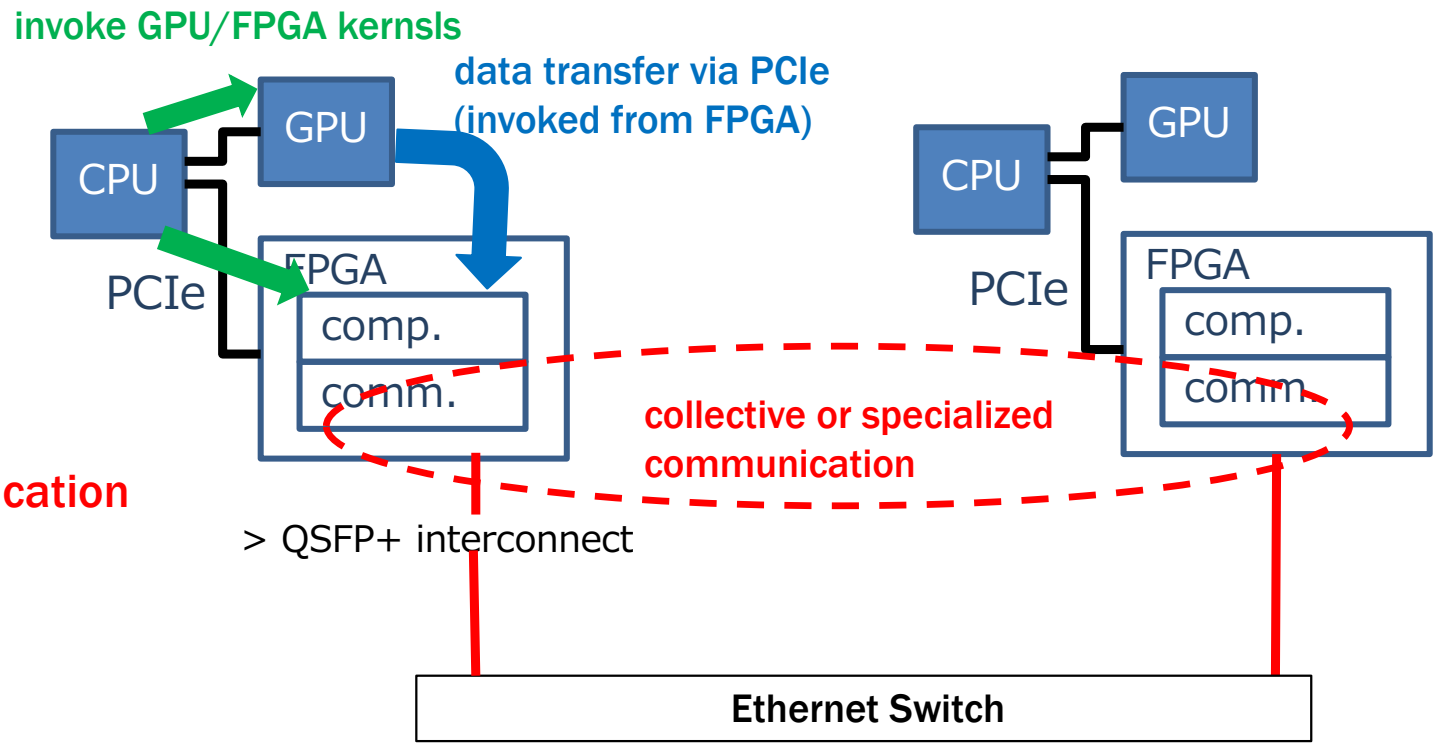
How to compensate with each other toward large degree of strong scaling ?

Multi-Hybrid Accelerated Computing

- Combining goodness of different type of accelerators: GPU + FPGA
  - GPU is still an essential accelerator for simple and large degree of parallelism to provide ~10 TFLOPS peak performance
  - FPGA is a new type of accelerator for application-specific hardware with programmability and speeded up based on pipelining of calculation
  - FPGA is good for external communication between them with advanced high speed interconnection up to 100Gbps x4chan.
- Multi (two type of devices) – Hybrid (acceleration) Supercomputer
  - PC cluster with multiple acceleration devices: GPU + FPGA (+CPU)
  - Mixing their good feature toward high performance computational sciences especially for strong scaling covering various size of computation on complicated calculation such as multi-physical simulation
  - Using FPGA's external communication link for low latency and high bandwidth communication

AiS: conceptual model of Accelerator in Swtich

- FPGA can work both for computation and communication in unified manner
- GPU/CPU can request application-specific communication to FPGA



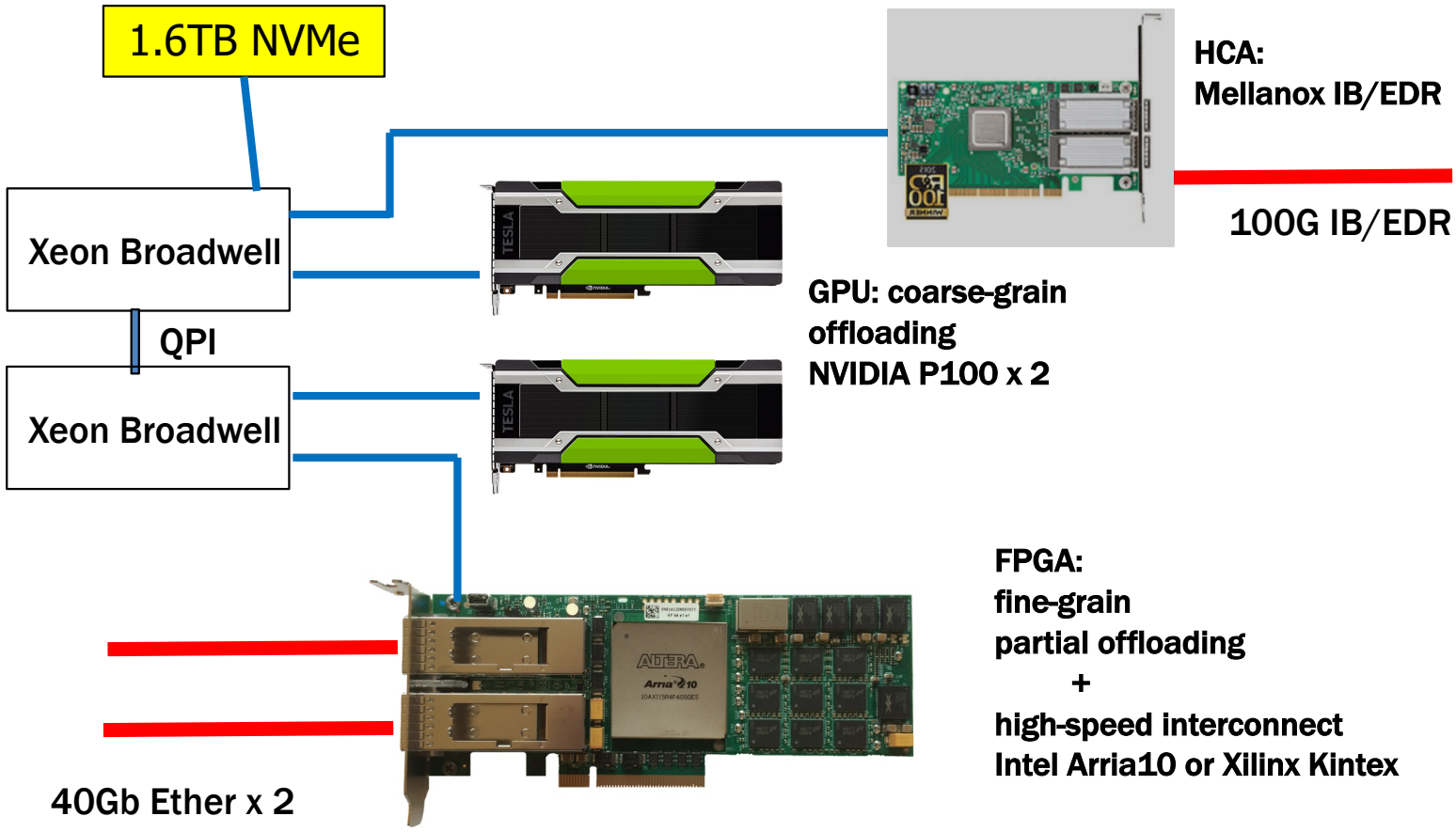
PACS-X Project
toward next generation's hybrid supercomputing

PACS-X (ten) Project at CCS, U. Tsukuba

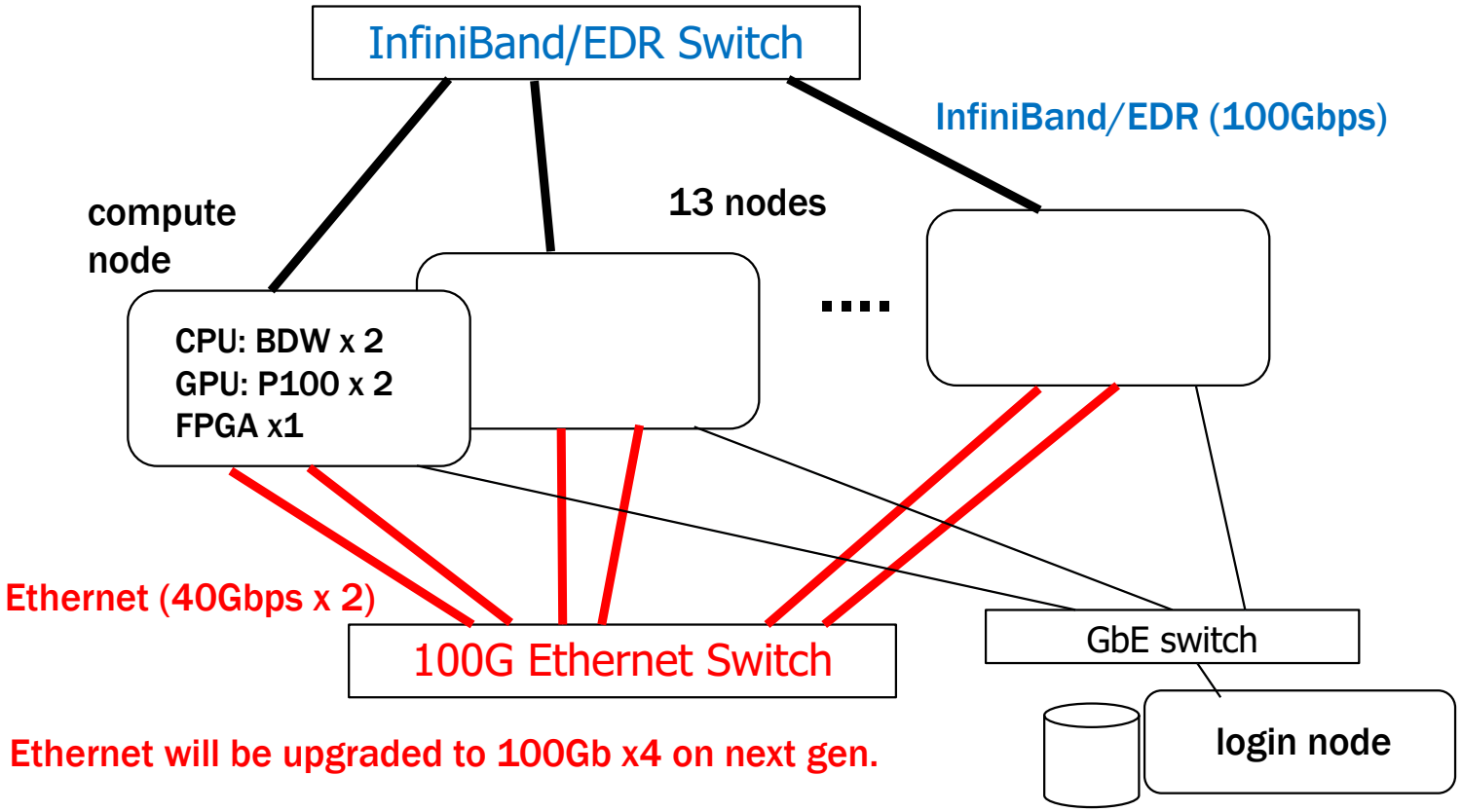
- History of supercomputers: PACS (PAX)
  - a series of co-design base parallel system development both on system and application at U. Tsukuba (1978~)
  - recent systems focus on accelerators
    - PACS-VIII: HA-PACS (GPU cluster, Fermi+Kepler, PEACH2, 1.1PFLOPS)
    - PACS-IX: COMA (MIC cluster, KNC, 1PFLOPS)
- Next generation of TCA implementation
  - new generation of GPU and FPGA with high speed interconnection
  - more tightly co-designing with applications
  - system deployment starts from 2017

➡ **PPX: Pre-PACS-X**

PPX: testbed under AiS concept (x 13 nodes)



PPX mini-cluster system

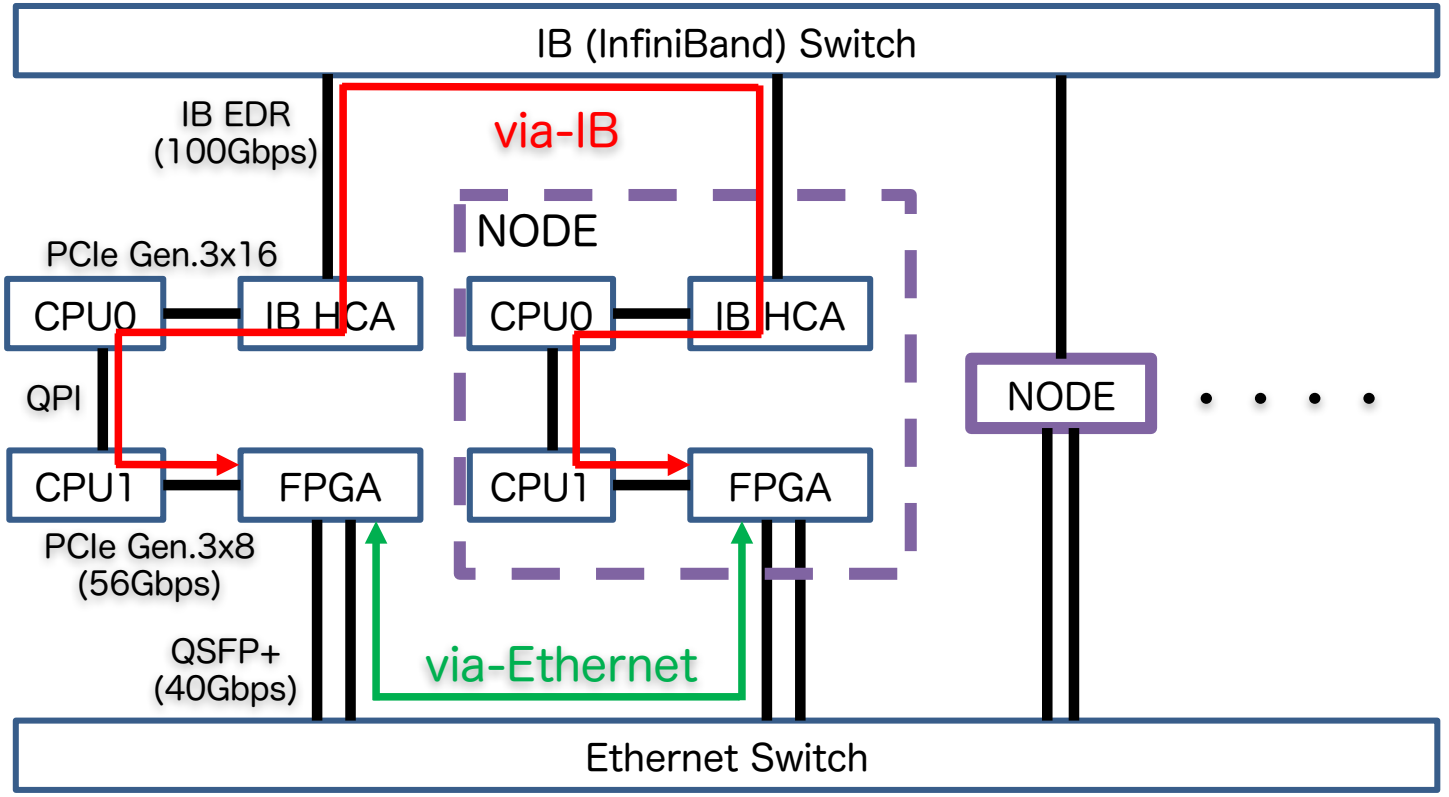


Current Research on PPX toward PACS-X Real System

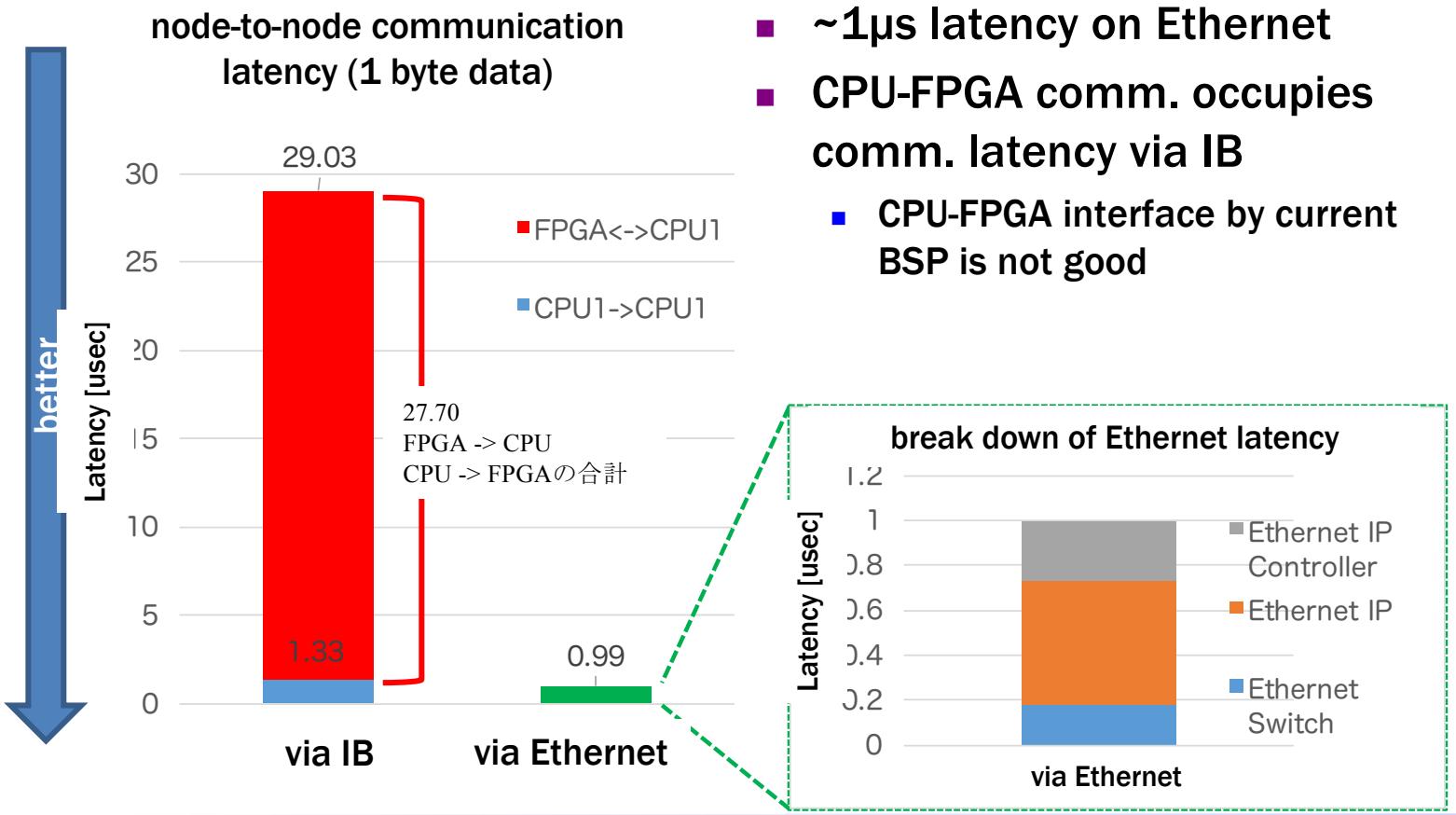
OpenCL-enabled high speed network

- OpenCL environment is available
  - ex) Intel FPGA SDK for OpenCL
  - basic computation can be written in OpenCL without Verilog HDL
- But, current FPGA board is not ready for OpenCL on interconnect access
  - BSP (Board Supporting Package) is not complete for interconnect
    → we developed for OpenCL access
- Our goal
  - enabling OpenCL description by users including inter-FPGA communication
  - providing basic set of HPC applications such as collective communication, basic linear library
  - providing 40G~100G Ethernet access with external switches for large scale systems

Communication paths



Communication latency

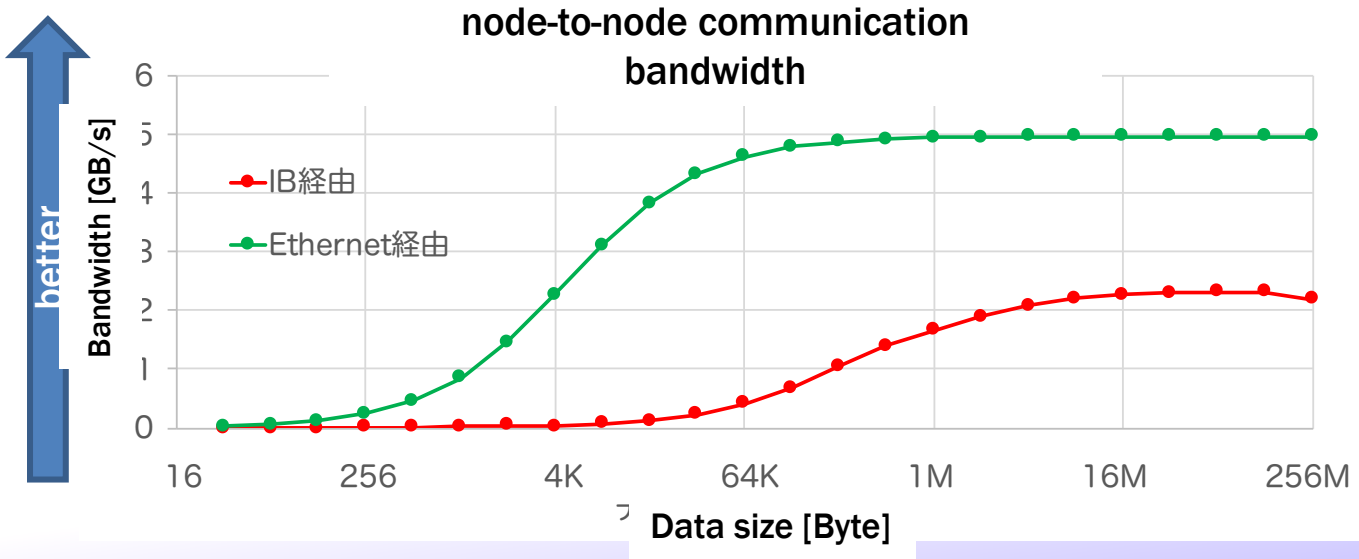


- ~1μs latency on Ethernet
- CPU-FPGA comm. occupies comm. latency via IB
  - CPU-FPGA interface by current BSP is not good

Communication bandwidth

- 40Gbps Ethernet achieves 4.97GB/s
  - 99.8 % of theoretical peak (w/o error handling)
  - small $N_{1/2}$ by short latency
- via-IB achieves 2.32GB/s
  - non-pipelined
  - no special feature (such as GPUDirect) on FPGA-HCA

➡ To be upgraded with 100Gbps x4chan. on next generation Stratix10



Channel over Ethernet (CoE): new implementation

- Channel
  - Exchange data between kernels in an FPGA
  - One of Intel FPGA's extensions to the OpenCL language
- Channel over Ethernet
  - enables us transfer data in channel over the network
  - Using same APIs as Channel from the user code
  - Ethernet as a protocol on the network

sender code on FPGA1

```
__kernel void sender(__global float* restrict x, int n) {
    for (int i = 0; i < n; i++) {
        float v = x[i];
        write_channel_intel(simple_out, v);
    }
}
```

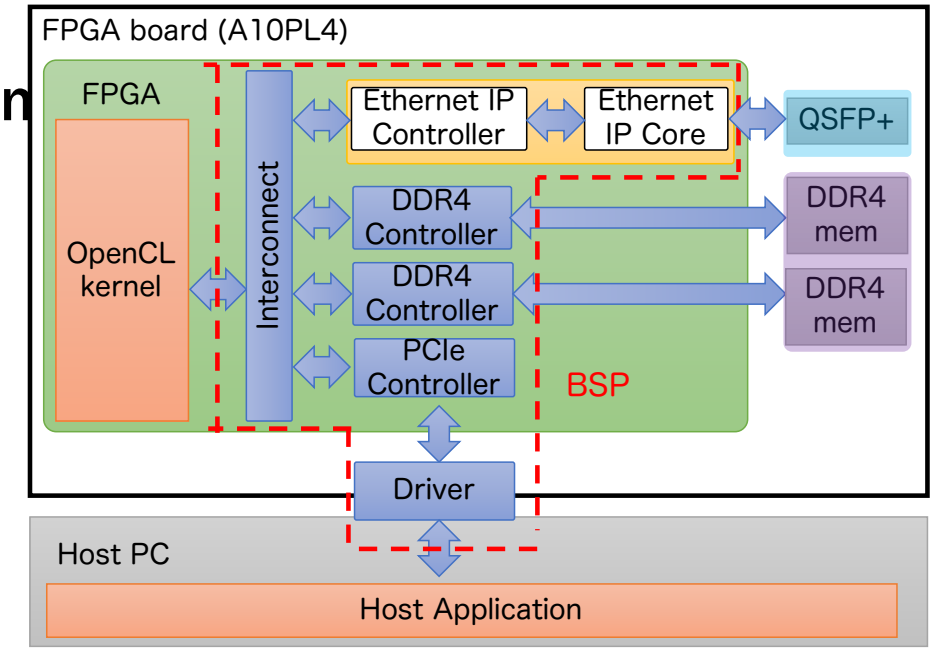
Eth.

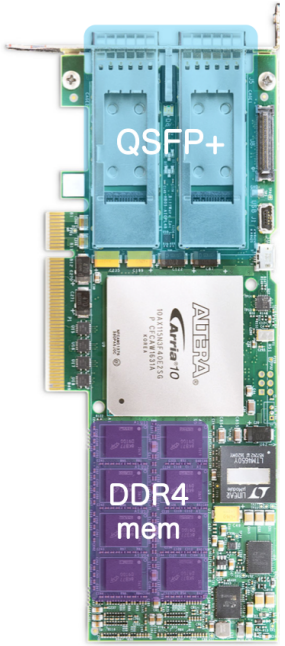
receiver code on FPGA2

```
__kernel void receiver(__global float* restrict x, int n)
    for (int i = 0; i < n; i++) {
        float v = read_channel_intel(simple_in);
        x[i] = v;
    }
}
```

Enabling Networking from OpenCL

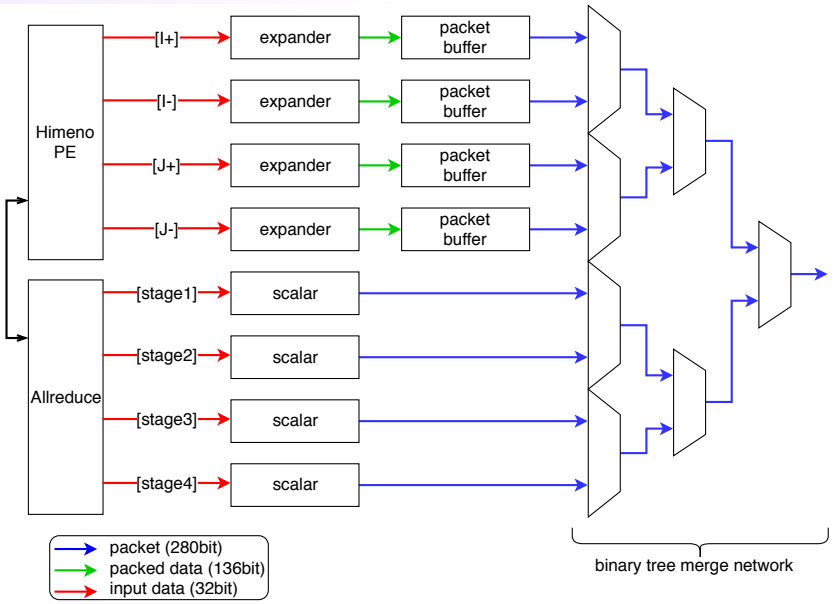
- Board Support Package (BSP) is a hardware component describing a board specification
- Adding network controller into BSP
  - 40Gb Ethernet IP
  - Bridge logic between OpenCL kernels and the IP
- OpenCL kernels can use it through I/O Channels.

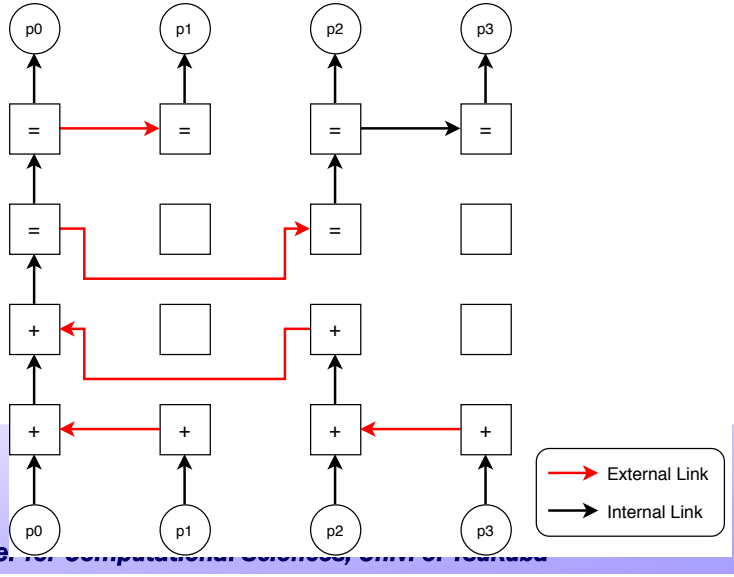




Implementation

- Packet handler logics are written in OpenCL
  - Buffering and binary merge tree for sending
  - Decoding packets for receiving
  - Tree algorithm for allreduce
- We decide to use this design to reduce modifying cost of BSP
  - Interface to the BSP is fixed
    - can be used for a variety of applications
  - But resource usage may be increased





Preliminary Result

- Himeno Benchmark [1]
  - 3D-Poisson Equation Solver, 19-point stencil computation
    - Performance is limited by memory bandwidth
  - Pipeline implementation using OpenCL:
    direct data transfer over the network to/from the pipeline
  - Size XS (33x33x65) on 1, 2 and 4 FPGAs (Strong Scaling)

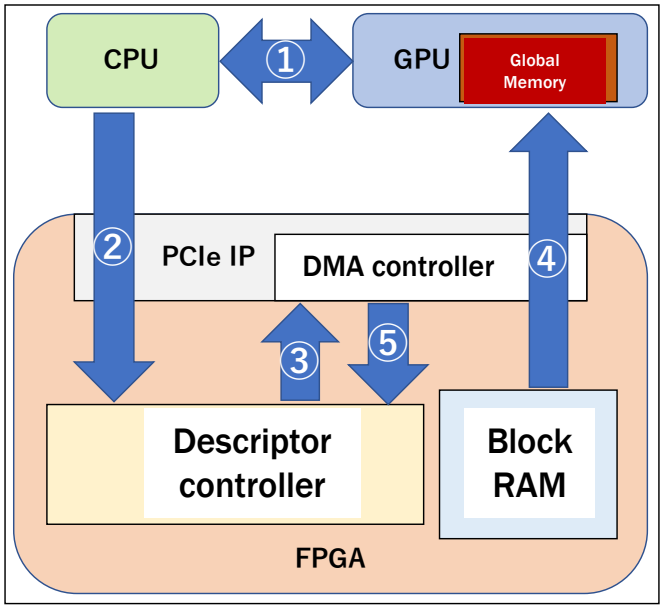
XS 1 node @210MHz	XS 2 nodes @225MHz	XS 4 nodes @208MHz
5019 MFLOPS	9762 MFLOPS	15225 MFLOPS
x1.00	x1.95	x3.03

- Scaling seems to be struggled by the problem size
  - XS has relatively larger halo size than other sizes.

[1] http://accc.riken.jp/en/supercom/himenobmt/

GPU-FPGA direct communication

- On PCIe (gen3 x16 lanes), FPGA and GPU can directly communication with DMA controller on FPGA to enhance the performance both on latency and bandwidth
- Ordinary way: GPU->CPU, CPU->FPGA (and reversed way)
- FPGA invokes DMA data transfer from/to GPU actively during computation without support by CPU (initial parameters should be preset by CPU)



* Hanawa, T et al., Interconnection Network for Tightly Coupled Accelerators, 2013 IEEE 21st Annual Symposium on High-Performance Interconnects, pp 79-82

```
void* d_m; cudaMalloc(&d_m, numdata*sizeof(uint32_t));

unsigned long long paddr;
tcaCreateHandleGPU(&paddr, d_m, numdata*sizeof(uint32_t));

if (m == MODE_READ) {
  for (int n = 1; n <= maxnumdata; n *= 2) {
    if (n == maxnumdata) n = numdata;
    uint64_t count_sum = 0;

    for (int i = 0; i < n; ++i) h_m[i] = n + i;
    cudaMemcpy(d_m, h_m, n*sizeof(uint32_t), cudaMemcpyHostToDevice);

    for (int t = 0; t < numtry; ++t) {
      int id = random_id();
      rd_ast_rx(ctrl, paddr, ONCHIP_MEMORY_OFFSET, n*sizeof(uint32_t), id);

      uint32_t status = rd_ast_tx(ctrl);
      if (status == unsigned(0x0100) + id) {
        count_sum += count_result_rd(ctrl);
      } else {
        std::cerr << "incomplete status = " << std::hex << status <<
std::endl;
        exit(1);
      }
    }
```

PCIe address mapping

descriptor transfer

watch termination signal from CPU

Code for FPGA ← GPU data transfer

GPU-FPGA data transfer latency

- Data size: 4 Bytes
  - Minimum size of data to be transferred
- FPGA ← GPU
  - 5.5x faster than ordinary method

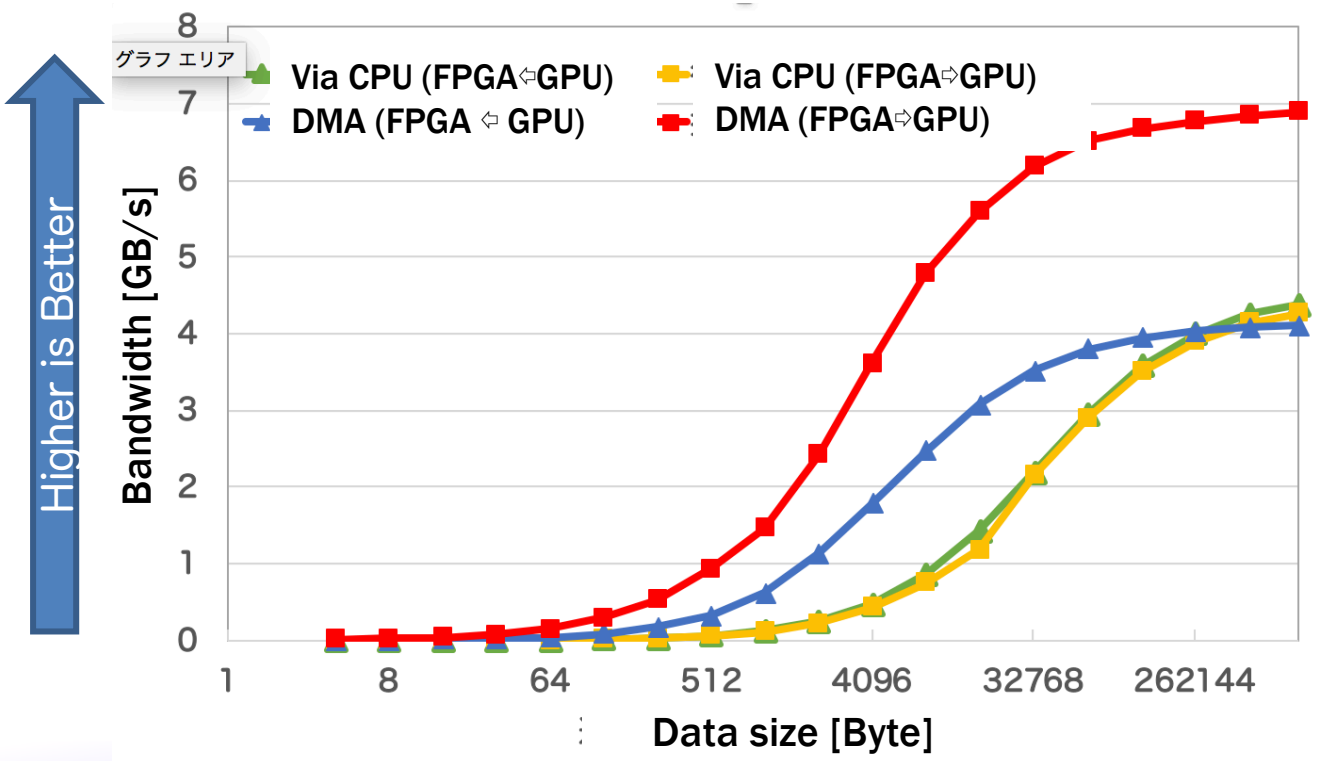
Method	Min. latency
Via CPU	8.04 μsec
DMA	1.45 μsec

- FPGA → GPU
  - 19x faster than ordinary method

Method	Min. latency
Via CPU	8.18 μsec
DMA	0.43 μsec

GPU-FPGA data transfer bandwidth

- Data size: 4 ~ 1M Bytes
  - Current bandwidth is bound by PCIe for A10PL4 (Arria10) of PCIe gen.3 x8lanes

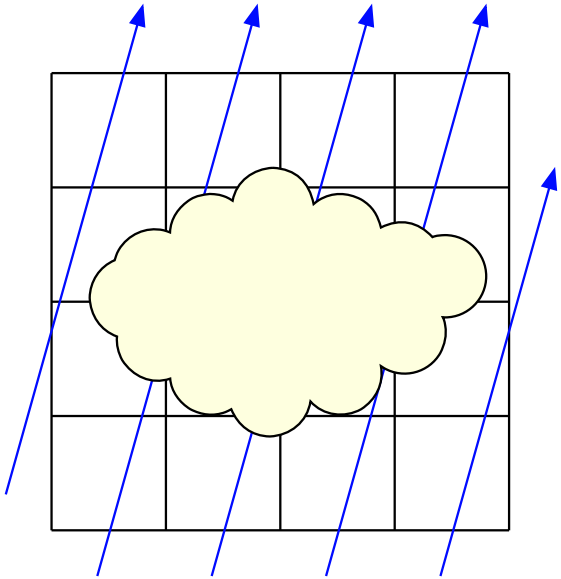


ARGOT

- ARGOT (Accelerated Radiative transfer on grids using Oct-Tree)
  - Radiative transfer code developed in Center for Computational Sciences (CCS), University of Tsukuba
  - CPU (OpenMP) and GPU (CUDA) implementations are available
  - Inter-node parallelisms is also supported using MPI
- ART (Authentic Radiation Transfer) method
  - It solves radiative transfer from light source spreading out in the space
  - Dominant computation part (90%~) of the ARGOT program
- In this research, we accelerate the ART method on an FPGA using Intel FPGA SDK for OpenCL as an HLS environment

ART Method

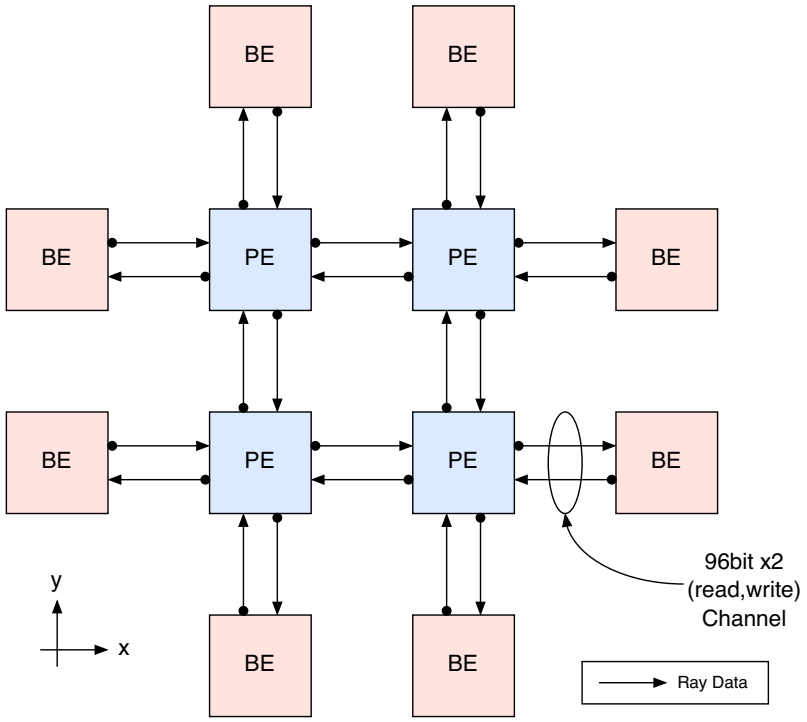
- ART method is based on ray-tracing method
  - 3D target space split into 3D meshes
  - Rays come from boundaries and move in straight in parallel with each other
  - Directions (angles) are given by HEALPix algorithm
- ART method computes radiative intensity on each mesh as shows as formula (1)
  - Bottleneck of this kernel is the exponential function (expf)
  - There is one expf call per frequency (v). Number of frequency is from 1 to 6 at maximum, depending on the target problem
  - All computation uses single precision computations
- Memory access pattern for mesh data is varies depending on ray's direction
  - Not suitable for SIMD style architecture
  - FPGAs can optimize it using custom memory access logics.



$$I_\nu^{out}(\hat{\boldsymbol{n}}) = I_\nu^{in}(\hat{\boldsymbol{n}})e^{-\Delta\tau_\nu} + S_\nu(1 - e^{-\Delta\tau_\nu}) \qquad (1)$$

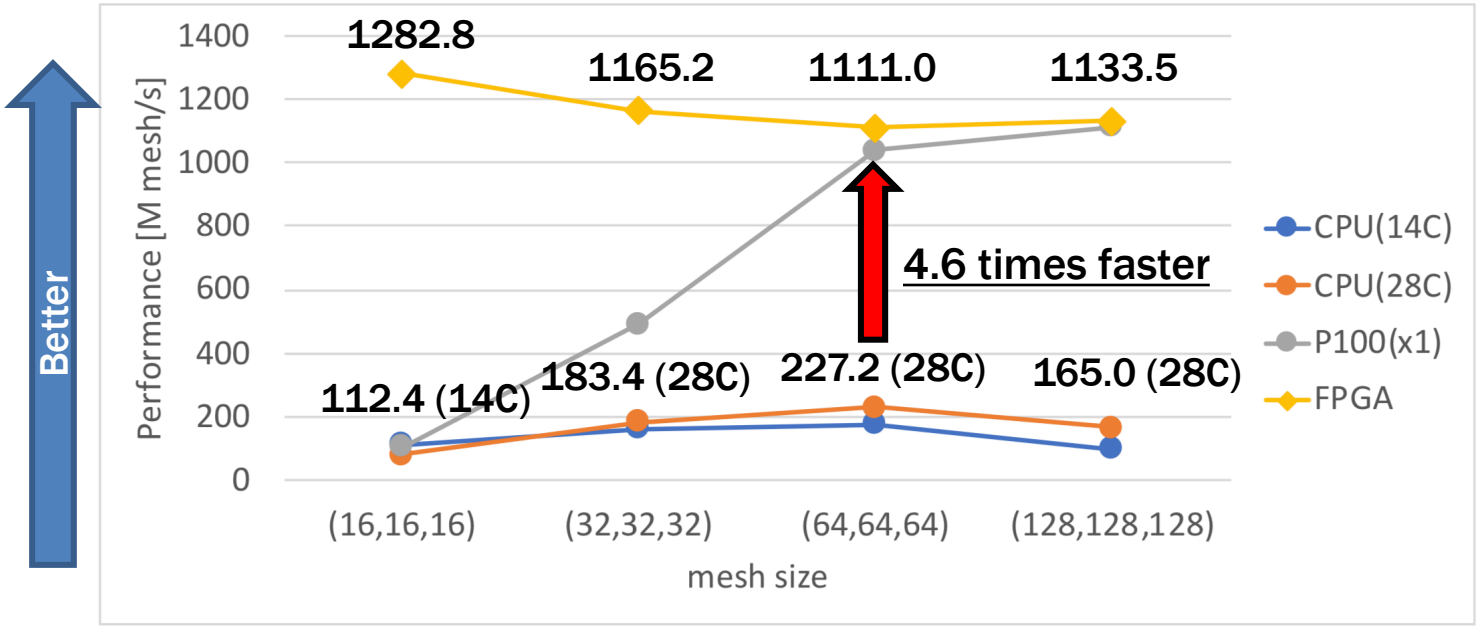
Parallelizing ART method using channels

- PE (Processing Element)
  - Computation kernel of ART method
  - Each PE has 8^3 size memory of mesh
- BE (Boundary Element)
  - BE handles ray I/O including boundary processing
  - R/W access of boundary, or R/W access of ray buffer
- Channels are used for communication
  - Transferring ray data to/from neighbor PE or BE
  - Two channels are used for each connection because a channel is one-sided



Performance (FPGA vs. CPU)

14C: 14 CPU cores=1 socket
28C: 28 CPU cores=2 sockets



- FPGA is faster than CPU on all problem sizes
- FPGA is 4.6 times faster on 64^3 and 6.9 times faster on 128^3
  - Decreasing on CPU from 64^3 to 128^3 is caused by cache miss
  - Size of mesh data cannot fit into the cache

Cygnus
New Supercomputer at CCS

Objective and time line

- Since COMA system will retire in the end of March 2019, we need a new machine as a follow-up system from April 2019
- We decided to deploy a new system with CCS budget with partial support from MEXT
- The new system is designed based on AiS concept to realize Multi-Hybrid Accelerated Computing with GPU and FPGA
- Time Line
  - RFI: Feb. 21st 2018
  - RFC: Jun. 8th 2018
  - RFP: Aug. 9th 2018
  - Proposal submission: Sep. 25th 2018 ⇨ Final system specification fixing (not yet today!!)
  - Bid open: Oct. 25th 2018
  - Delivery: Mar. 15th 2019
  - Test operation: Apr. 1st 2019
  - Real program starts: May 1st 2019

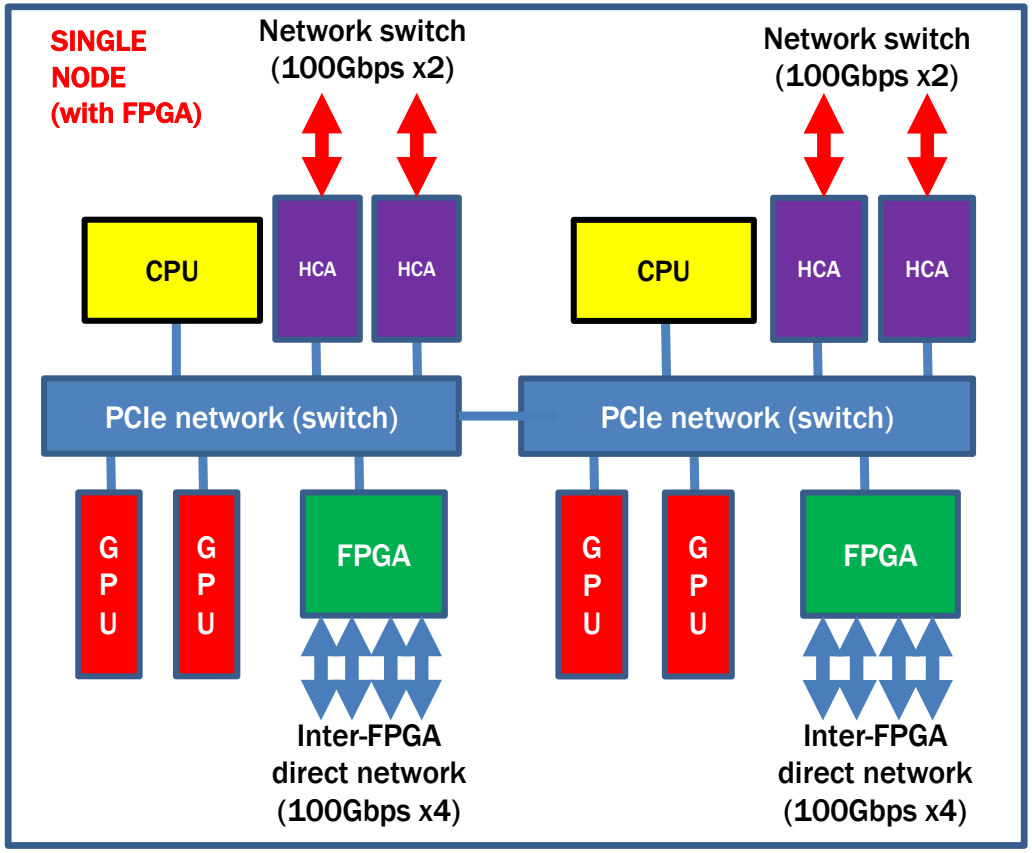
Specification data shown today is subject to be changed

System name and construction

- System name: “Cygnus”
  - based on “Cygnus A” galaxy with two accelerated radiation streams
- Consisting of two parts
  - GPU-only part: equipped with CPU and GPU to be used as ordinary GPU cluster ⇨ Deneb?
  - GPU+FPGA part: equipped with GPU and FPGA connected via PCIe and CPU to be used as AiS system, as well as ordinary GPU cluster (not using FPGA) ⇨ Albireo?
  - CPU and GPU of either part are identical as well as PCIe switch configuration
- Two parts are combined with InfiniBand high performance interconnection
- When FPGA-ready job is not running, GPU+FPGA part is also available as GPU-only part for high utilization ratio of the system
- Each computation node of either part is with “fat” configuration:
  very high performance both on computation and communication

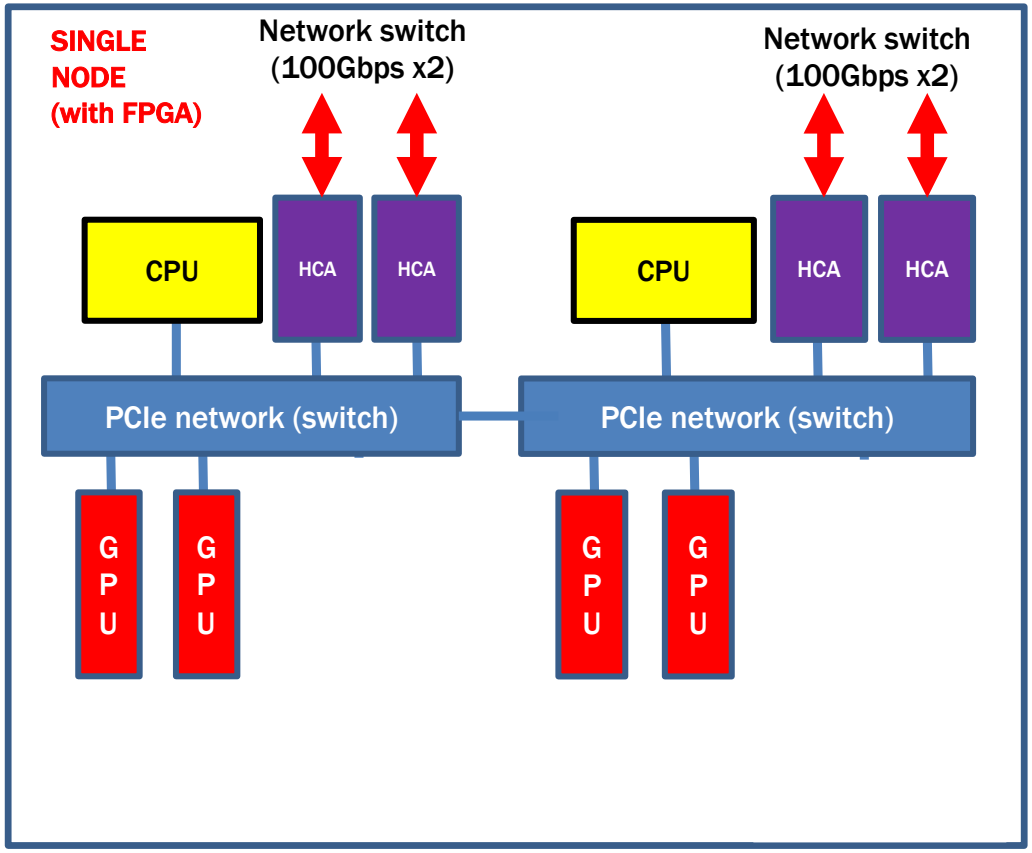
Single node configuration (with FPGA): “fat node”

- Each node is equipped with both IB EDR and FPGA-direct network
- Some nodes are equipped with both FPGAs and GPUs, and other nodes are with GPUs only



Single node configuration (without FPGA): “fat node”

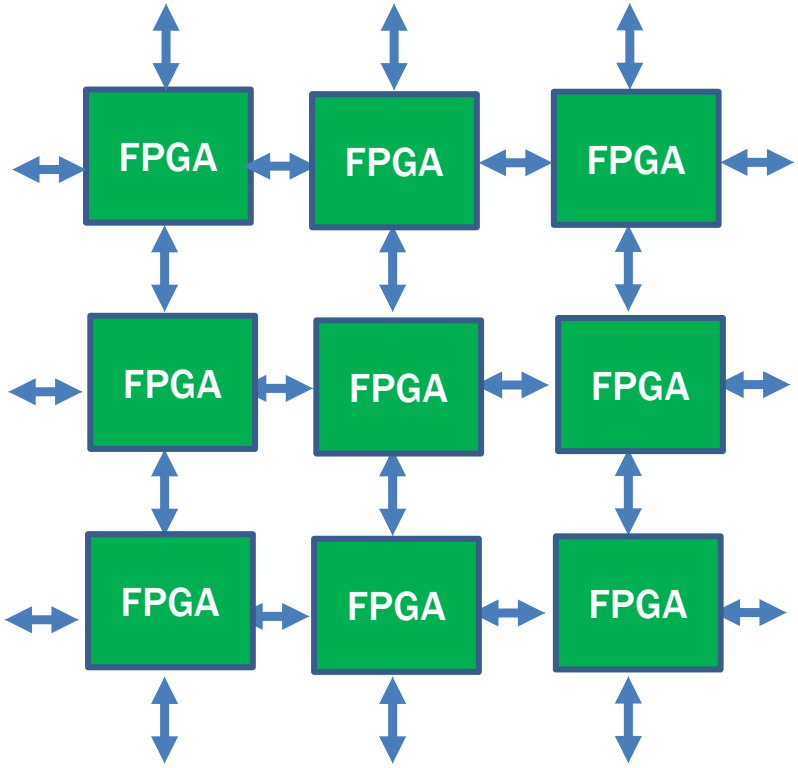
- Each node is equipped with both IB EDR and FPGA-direct network
- Some nodes are equipped with both FPGAs and GPUs, and other nodes are with GPUs only



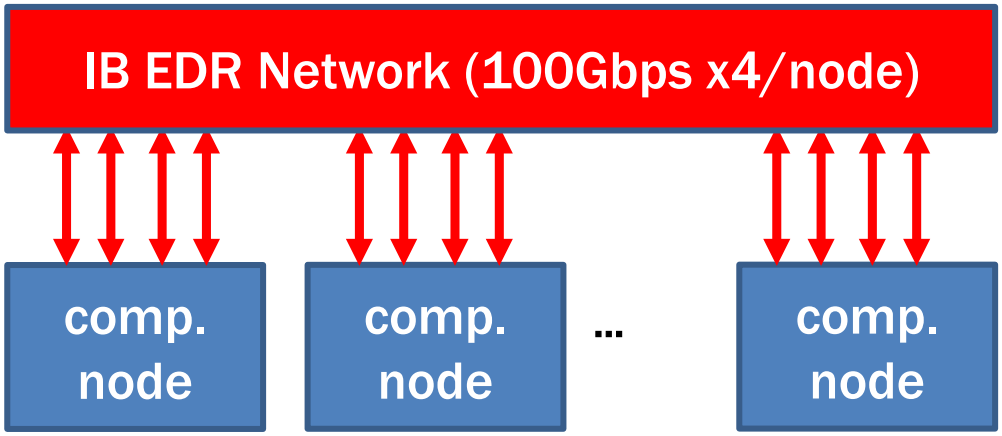
Approximate specification of Cygnus

Item	Specification
Peak performance	~3.0 PFLOPS DP (GPU: 2.3 PFLOPS, CPU: 0.12 PFLOPS, FPGA: 0.6 PFLOPS) ⇨ enhanced by mixed precision and variable precision on FPGA
# of nodes	~80 (~32 GPU+FPGA nodes, ~48 GPU-only nodes)
CPU / node	Intel Xeon Gold x2 sockets
GPU / node	NVIDIA V100 x4 (PCIe)
FPGA / node	Intel Stratix10 x2 (each with 100Gbps x4 links)
Global File System	Lustre, RAID6, ~3.0 PB
Interconnection Network	Mellanox InfiniBand EDR (HDR100) x4
Total Network B/W	~4 TB/s
Programming Language	CPU: C, C++, Fortran, OpenMP GPU: OpenACC, CUDA FPGA: OpenCL, Verilog HDL

Two types of inter-node network



(only for GPU+FPGA node)



Ordinary inter-node communication channel for CPU and GPU, but they can request it to FPGA

High Level Programming Paradigm

- XcalableACC
  - under development in collaboration between CCS-Tsukuba and RIKEN-AICS
  - PGAS language XcalableMP is enabled to imply OpenACC for sophisticated coding of distributed memory parallelization with accelerator
  - inter-node communication among FPGA can be implemented by FPGA-Ethernet direct link
  - Data movement between GPU and FPGA
- OpenACC for FPGA
  - (plan) research collaboration with ORNL FTG
  - OpenACC -> OpenCL -> FPGA compilation by OpenARC project is under development
  - final goal: XcalableACC with OpenARC compiler and FPGA-Ethernet link

Summary

- CCS will introduce a new supercomputer based on AiS concept as the next generation‘s Multi-Hybrid Accelerated Supercomputer
- New machine Cygnus will be equipped with very high performance GPU and FPGA (partially) to make “strong scaling ready” accelerated system for applications where GPU-only solutions is weak, as well as all kind of GPU-ready applications
- FPGA for HPC is a new concept toward next generation’s flexible and low power solution beyond GPU computing
- Cygnus system will be delivered on next March and official program start on May 2019