Computation/Communication Offloading to FPGA with GPU

Taisuke Boku

Center for Computational Sciences

University of Tsukuba

based on research collaboration with Y. Yamaguchi, R. Kobayashi, N. Fujita, Y. Ohbata, M. Umemura and K. Yoshikawa

Outline

- FPGA for HPC as large scale parallel system
- AiS (Accelerator in Switch) concept
- PACS-X Project and PPX
- FPGA for high performance interconnection and computation
- Summary

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 (SIMT) mechanism in a chip
    - high bandwidth memory (GDR5, HBM) and local memory
  - bad for
    - not enough parallelism
    - not regular computation (warp splitting)
    - 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 x 2~4
  - Precision control is possible
  - Relatively low power
- Problems
  - Programmability: OpenCL is not enough, not efficient yet
  - 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)

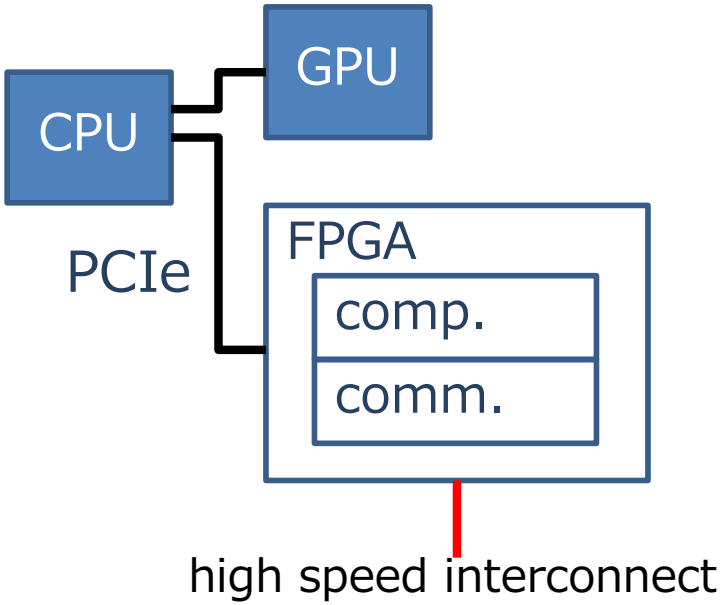
Simple pros/cons

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

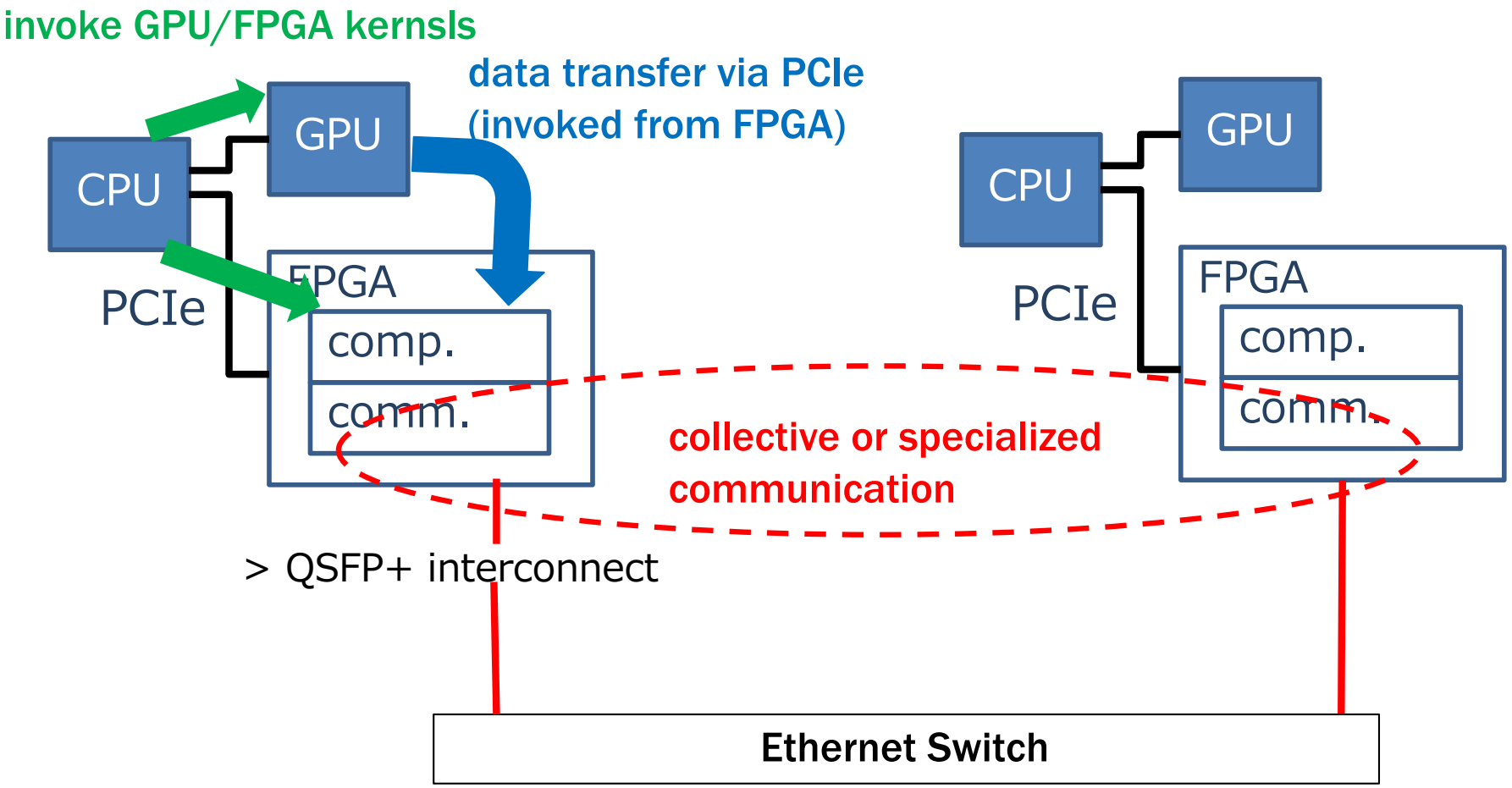
How to compensate with each other toward large degree of strong scaling ?

AiS

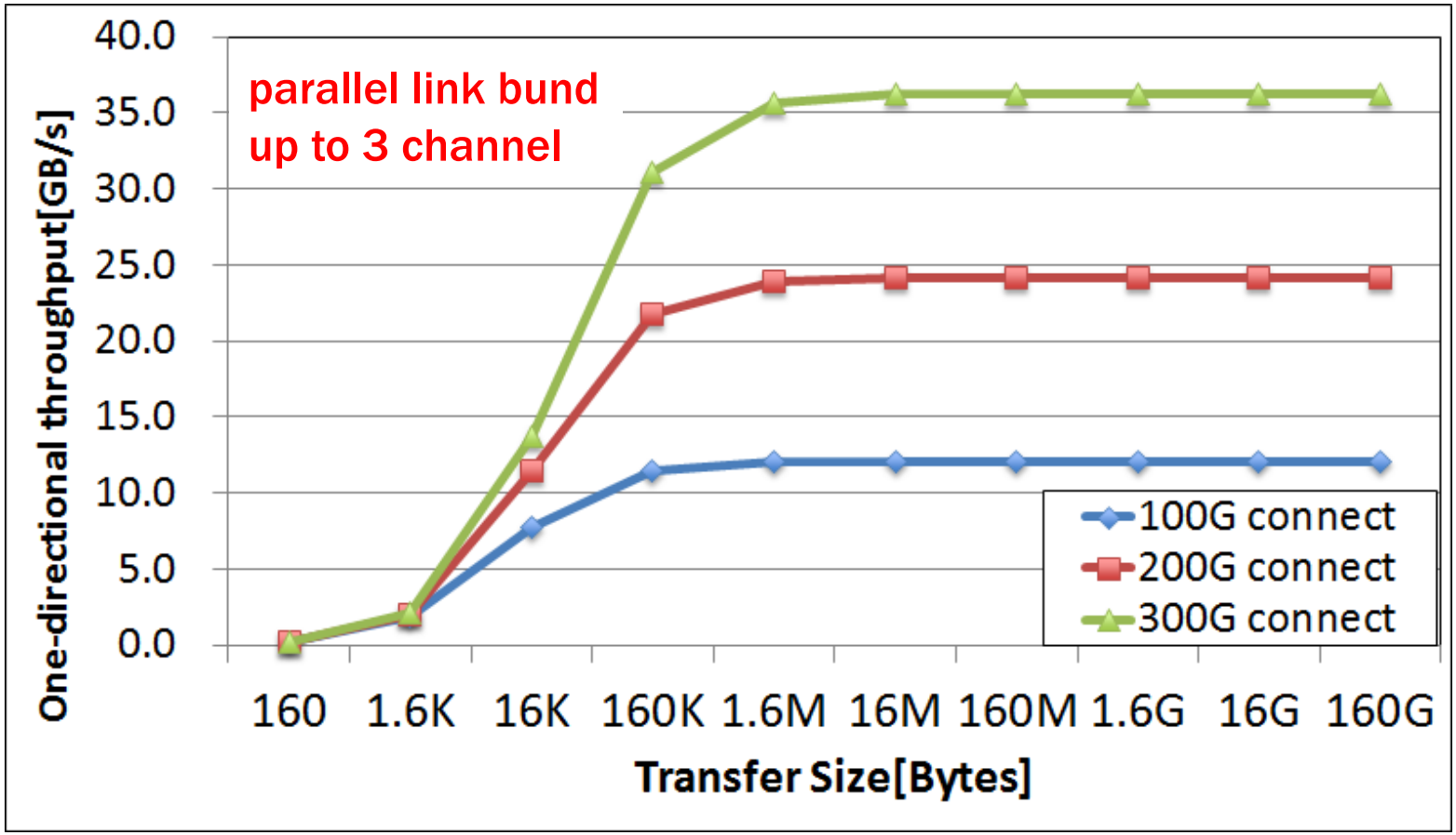
- AiS: Accelerator in Swtich
  - Using FPGA not only for computation offloading but also for communication
  - Combining computation offloading and communication among FPGAs for ultra-low latency on FPGA computing
  - Especially effective on communication-related small/medium computation (such as collective communication)
  - Covering GPU non-suited computation by FPGA
  - OpenCL-enable programming for application users



AiS computation model



How fast on communication ? (FPGA-FPGA link)



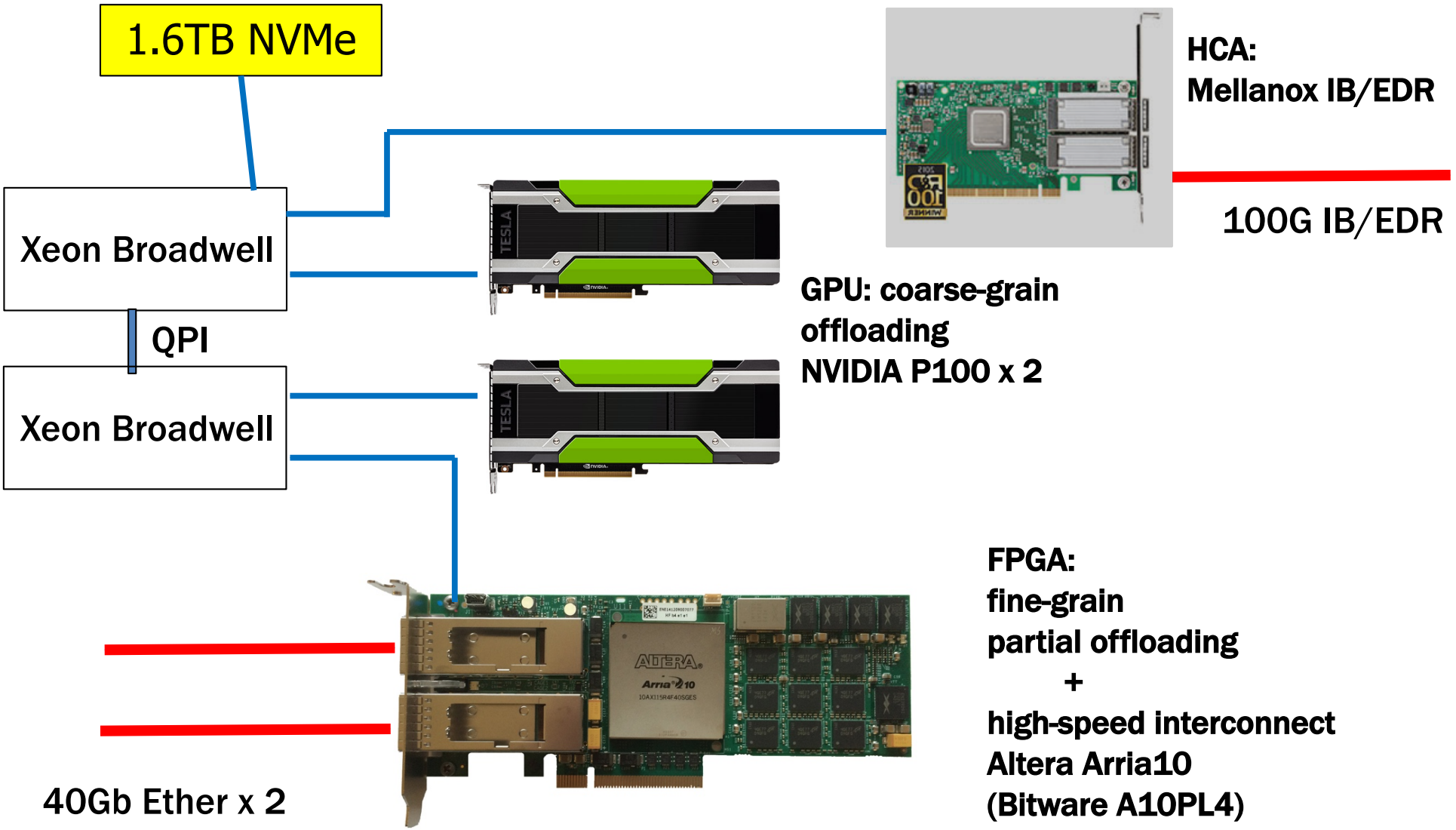
- Xilinx XC7VX1140T(Virtex7) with 100Gbps optical interconnect
- up to 96% of theoretical peak
- good scalability up to 3 channels aggregation

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

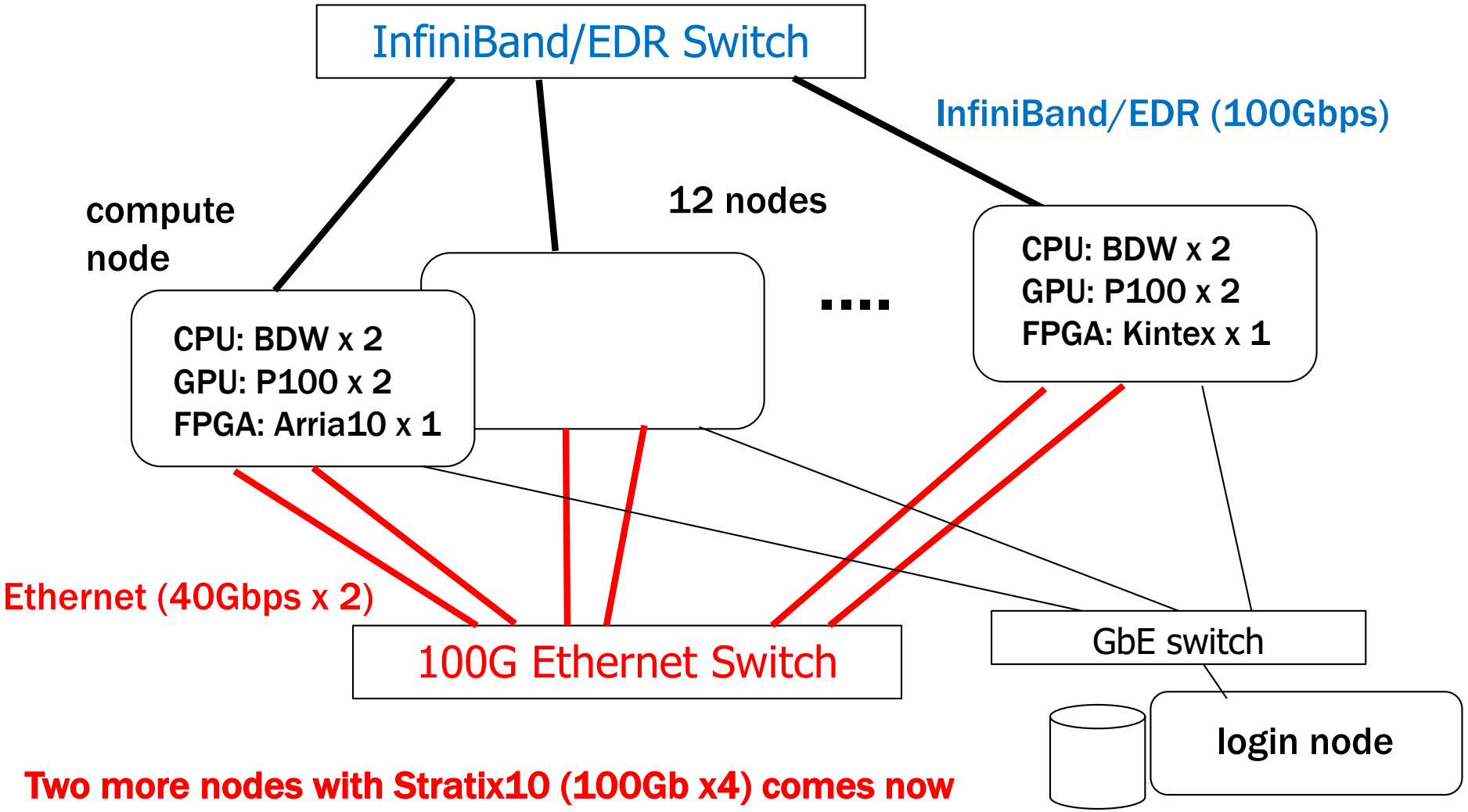
- PACS (Parallel Advanced system for Computational Sciences)
  - a series of co-design base parallel system development both on system and application at U. Tsukuba (1978~)
  - recent systems focus on accelerators
    - PACS-VIII: HA-PACS (GPU cluster, Fermi+Kepler, PEACH2, 1.1PFLOPS)
    - PACS-IX: COMA (MIC cluster, KNC, 1PFLOPS)
- Next generation of TCA implementation
  - PEACH2 with PCIe is old and with several limitation
  - new generation of GPU and FPGA with high speed interconnection
  - more tightly co-designing with applications
  - system deployment starts from 2018 (?)

➡ **PPX: Pre-PACS-X**

PPX: testbed under AiS concept (x6~12 nodes)



PPX mini-cluster system



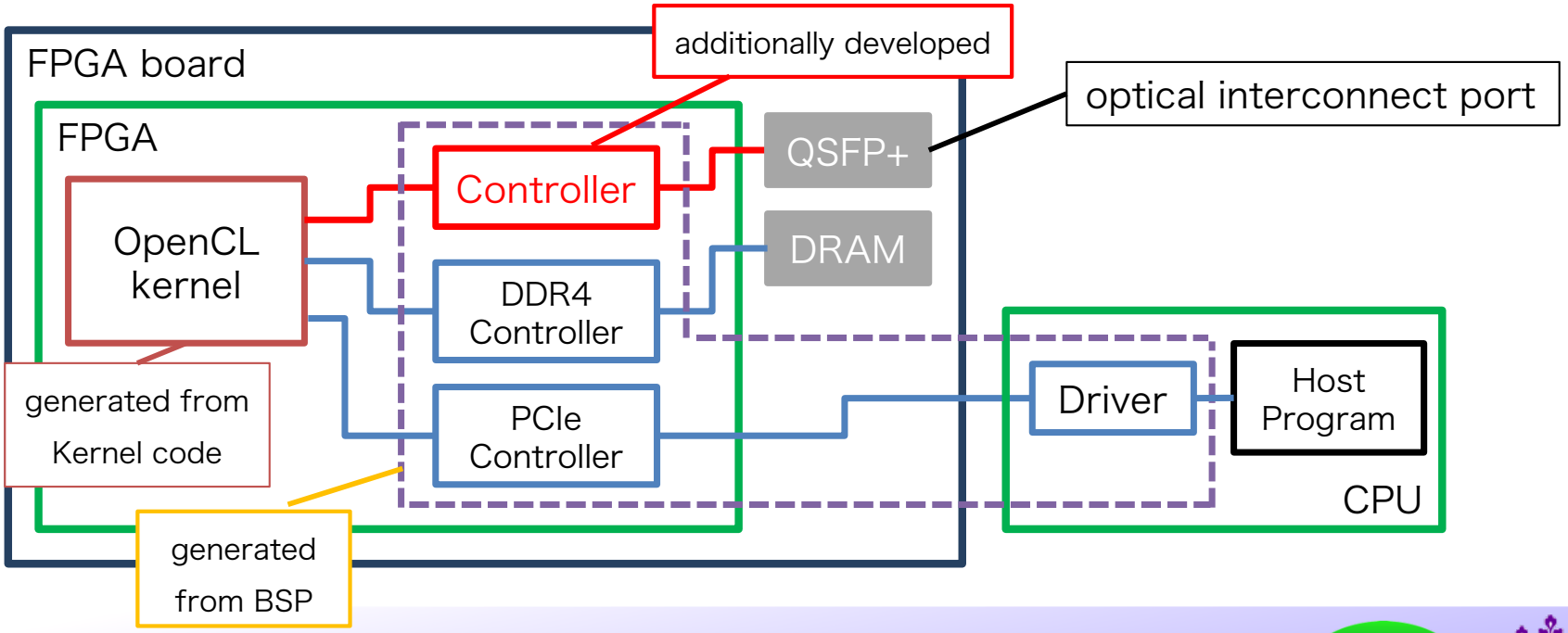
Two more nodes with Stratix10 (100Gb x4) comes now

OpenCL-enabled high speed network

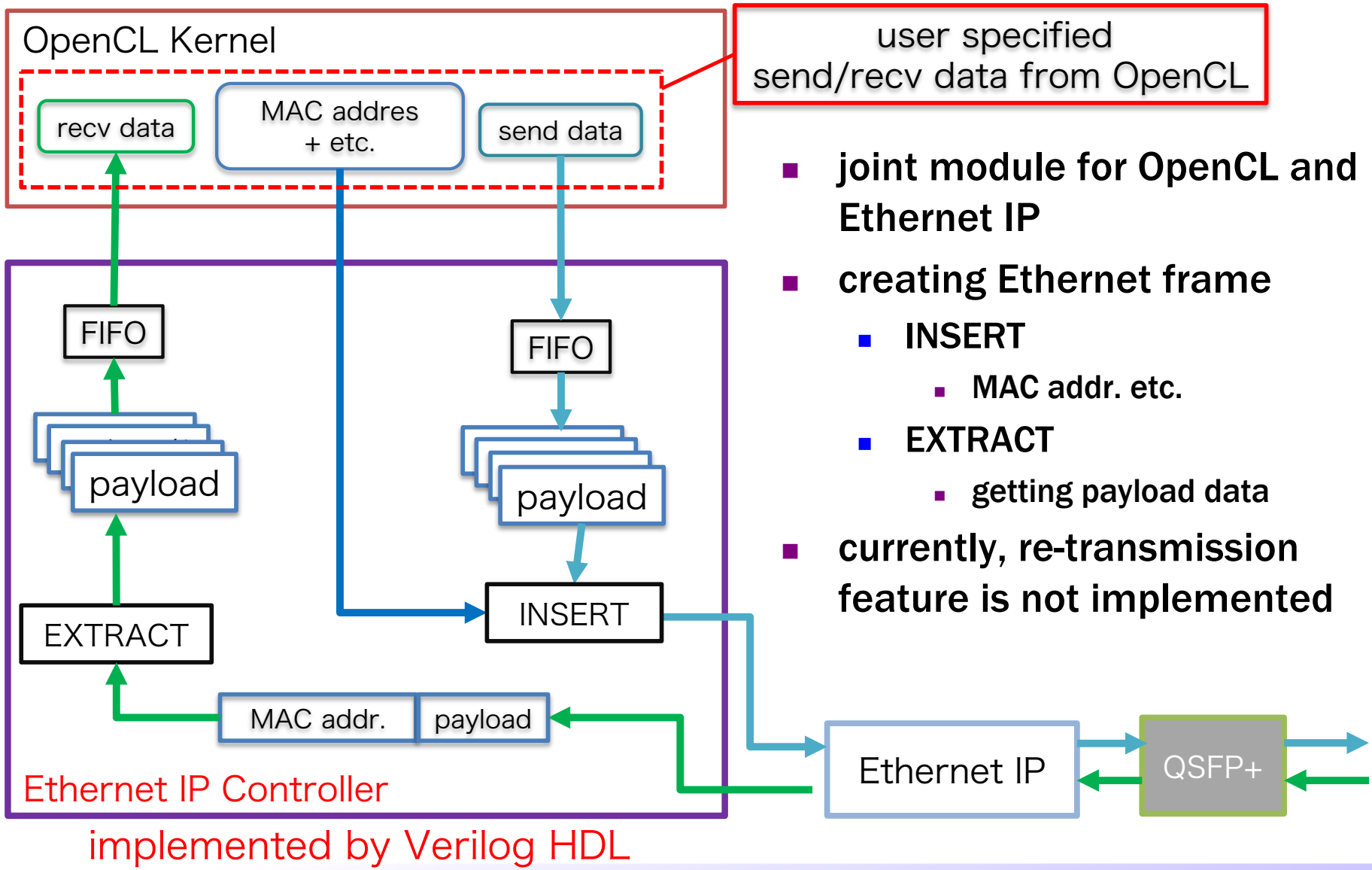
- OpenCL environment is available
  - ex) Intel FPGA SDK for OpenCL
  - basic computation can be written in OpenCL without Verilog HDL
- But, current FPGA board is not ready for OpenCL on interconnect access
  - BSP (Board Supporting Package) is not complete for interconnect
    → we developed for OpenCL 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

BSP (Board Support Package)

- description specifying FPGA chip and board peripherals configuration and access/control method
  - independent for each board with FPGA
  - a sort of virtualization to enable same kernel development on FPGA
- minimum interface is provided by board vendors
  - we need optical interconnection access method in BSP



Ethernet IP Controller



- joint module for OpenCL and Ethernet IP
- creating Ethernet frame
  - INSERT
    - MAC addr. etc.
  - EXTRACT
    - getting payload data
- currently, re-transmission feature is not implemented

OpenCL code example for pingpong

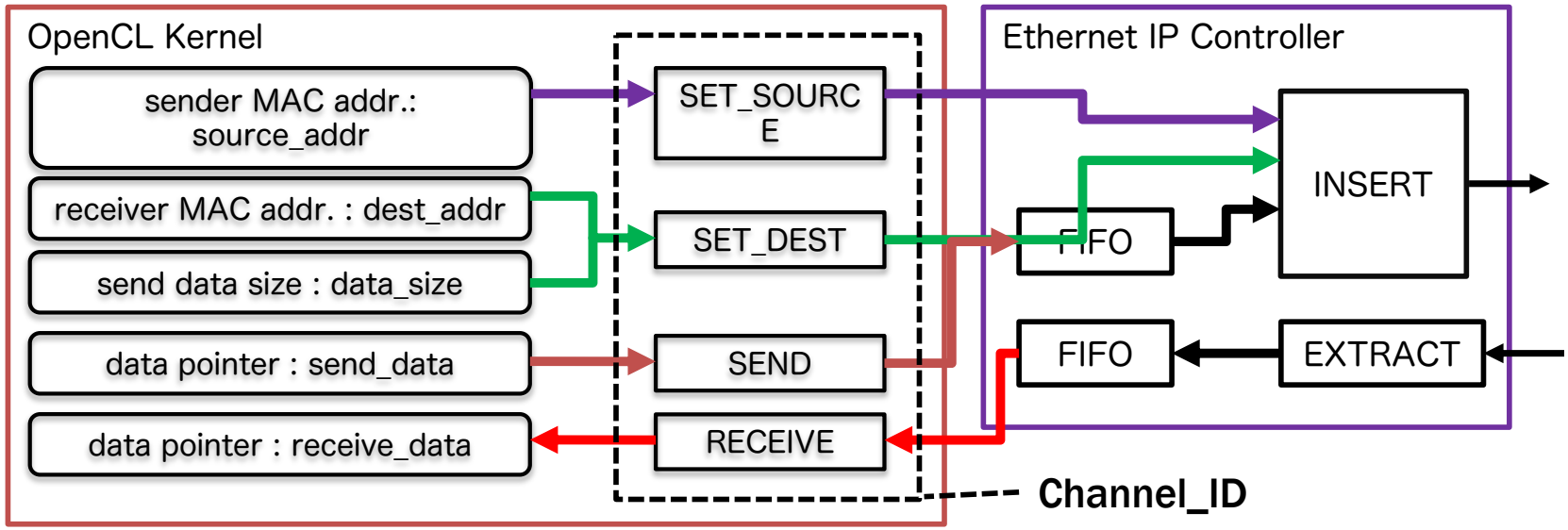
```
write_channel_intel (SET_SOURCE , source_addr);
write_channel_intel (SET_DEST , (int2)(data_size , dest_addr));

for( i = 0 ; i < data_size ; i ++)
      write_channel_intel (SEND , send_data[i] );
```

sender

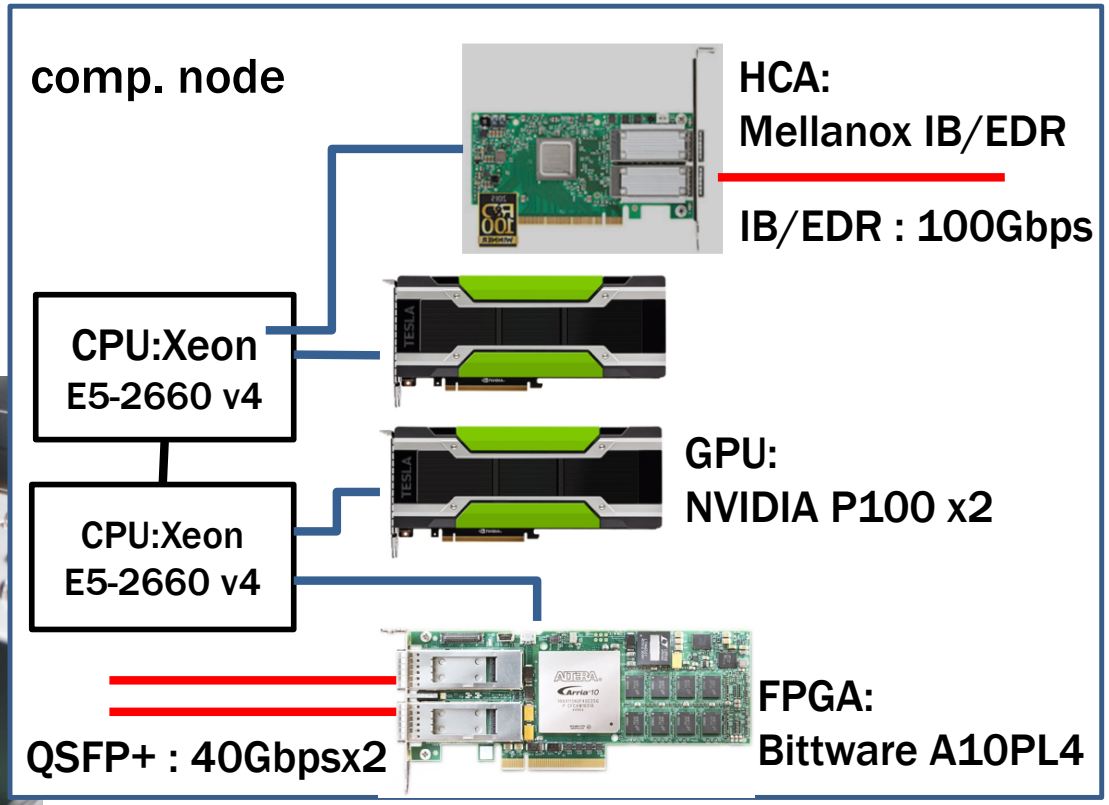
```
for( i = 0 ; i < data_size ; i ++)
      receive_data[i] = read_channel_intel (RECEIVE);
```

receiver



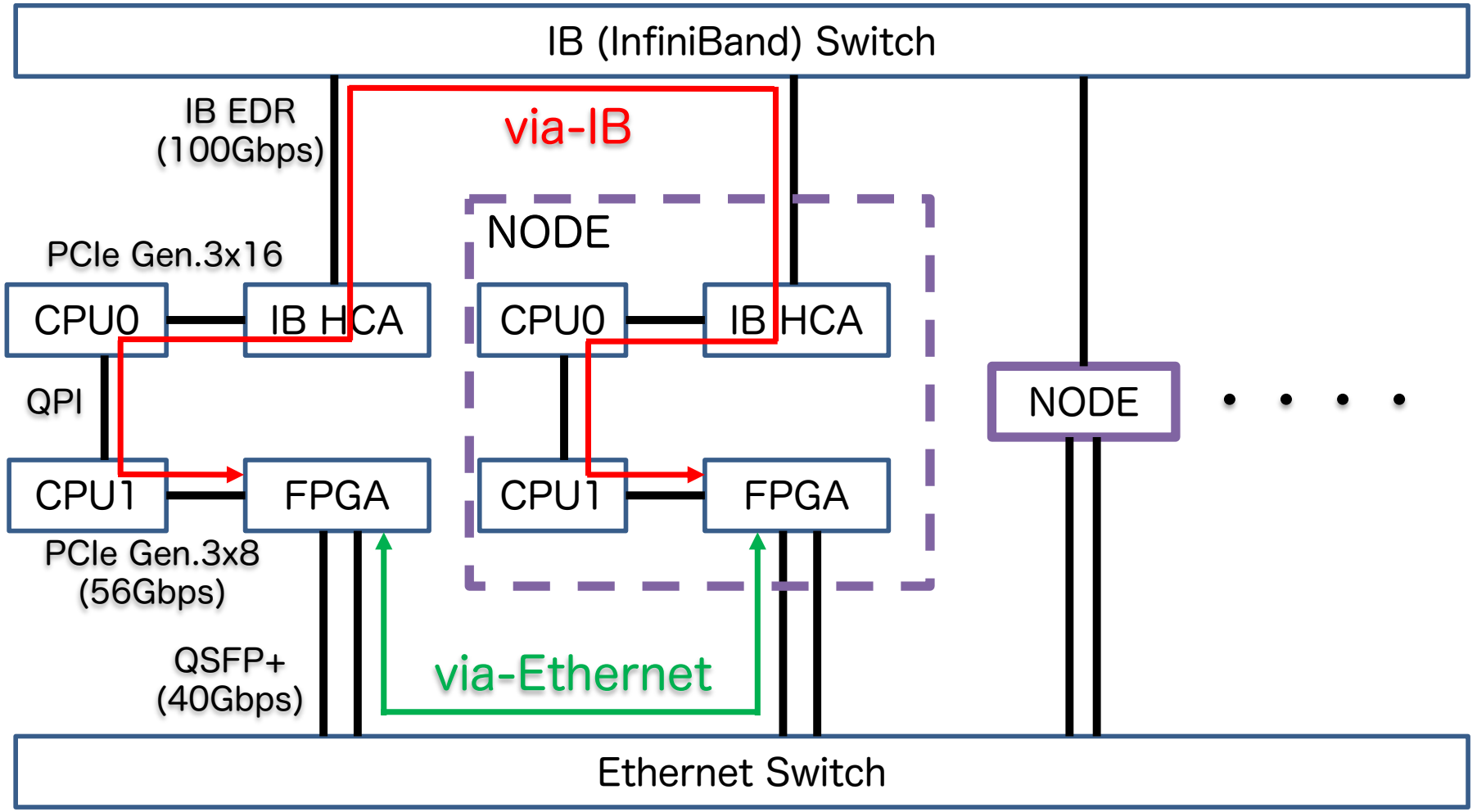
Evaluation test-bed

- Pre-PACS-X (PPX)
  - CCS, U. Tsukuba
  - PACS-X prototype

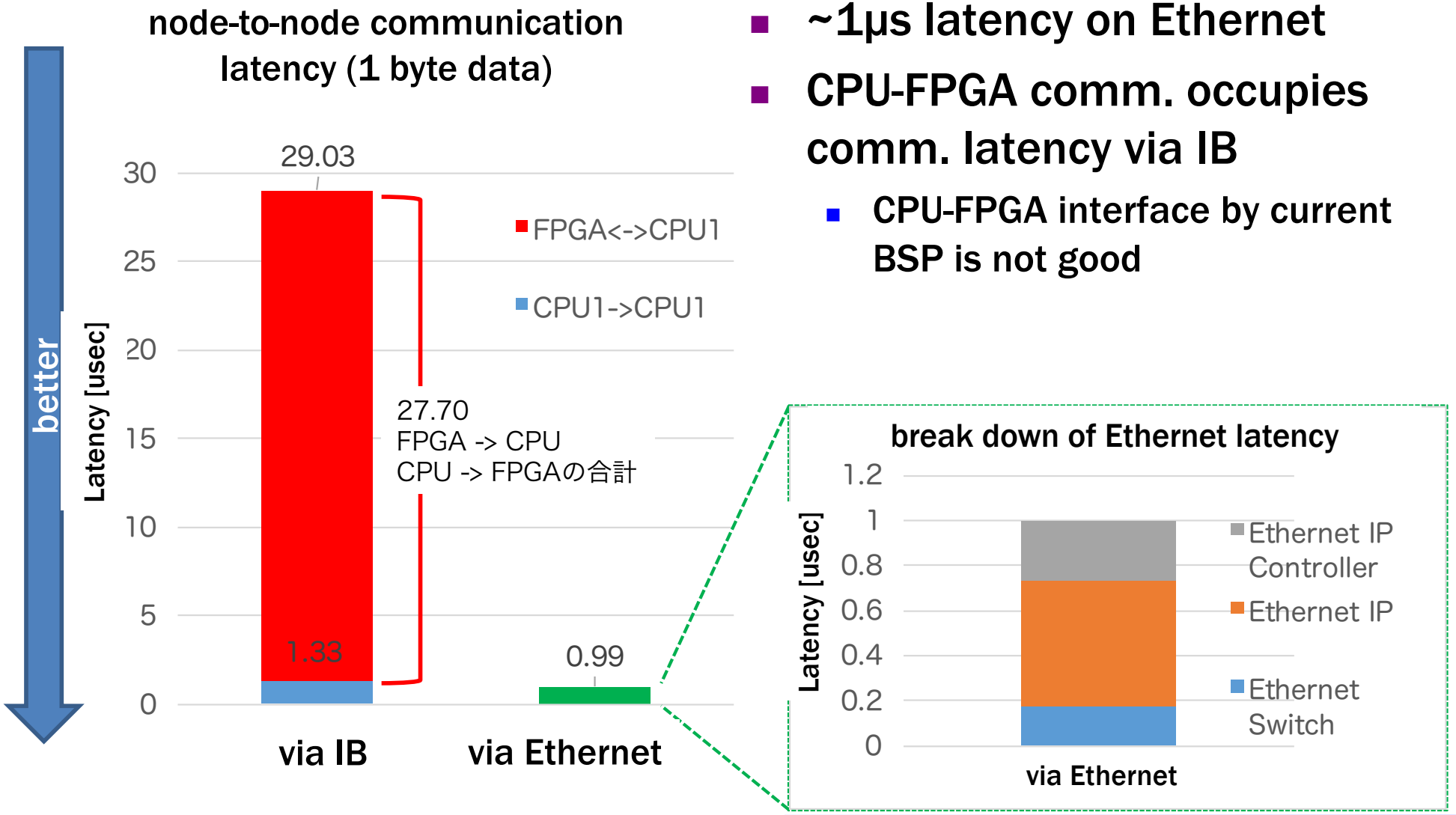


Host OS	CentOS 7.3
Host Compiler	gcc 4.8.5
FPGA Compiler	Intel FPGA SDK for OpenCL, Intel Quartus Prime Pro Version 17.0.0 Build 289

Communication paths



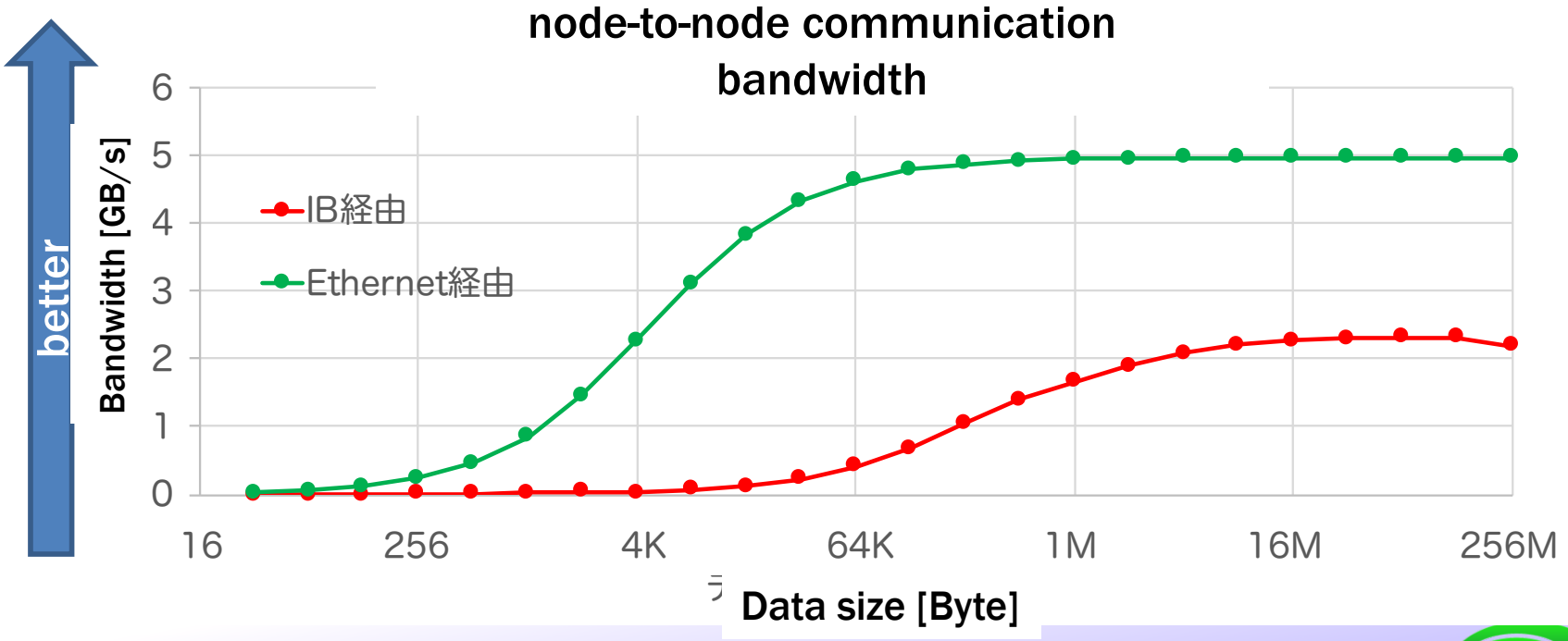
Communication latency



- ~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

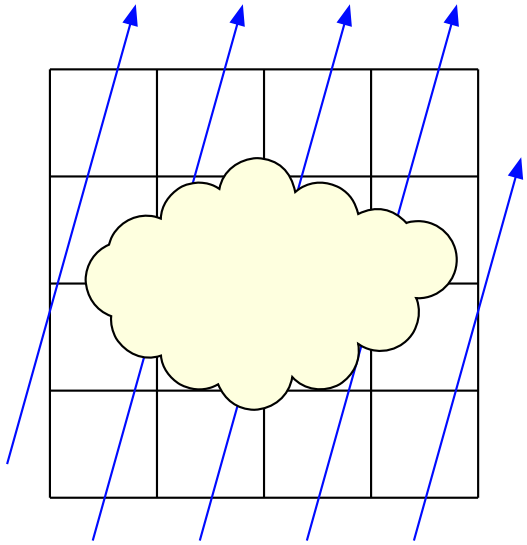


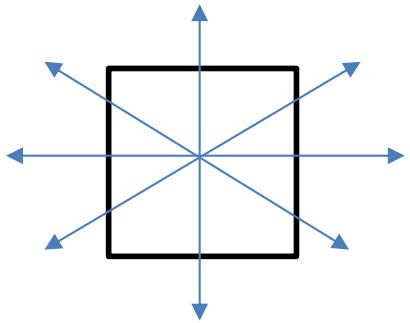
AiS application example: ARGOT

- ARGOT (Accelerated Radiative transfer on grids using Oct-Tree)
  - Radiative transfer simulation code developed in CCS
  - Two basic computing methods for radiation transfer
    - ARGOT method
      - from a light source
    - ART
      - from spatially spread light sources
- CPU version and GPU version with MPI
- ART method occupies >90% of computation even on GPU, and we need more speedup
  → making FPGA offloading in AiS concept
- Other computation is till on GPU, so GPU-FPGA connectivity is also important

ART method

- radiative transfer computing on spatially spread light sources
- ray-tracing on 3-D space with grid decomposed partitions
  - rays are in parallel
  - different input angles
  - no reflection nor refraction (different from 3-D graphics ray-tracing)
  - HEALPix algorithm for ray generation
- large scale for parallel processing
  - mesh size: 100^3~1000^3
  - ray angles: 768~1000s





Performance (single FPGA) on ART method

Device	Perf. [M mesh/sec]	vs CPU
CPU	117.49	-
FPGA@228.57MHz (w/o autorun)	593.11	5.05
FPGA@236.11MHz (w/ autorun)	**1714.97**	**14.60**

- up to 14.6x faster than CPU, and 5.1x faster than GPU
- 93% of computation time of ARGOT is dominated by ART method
  → 7.48x speedup on entire code is expected

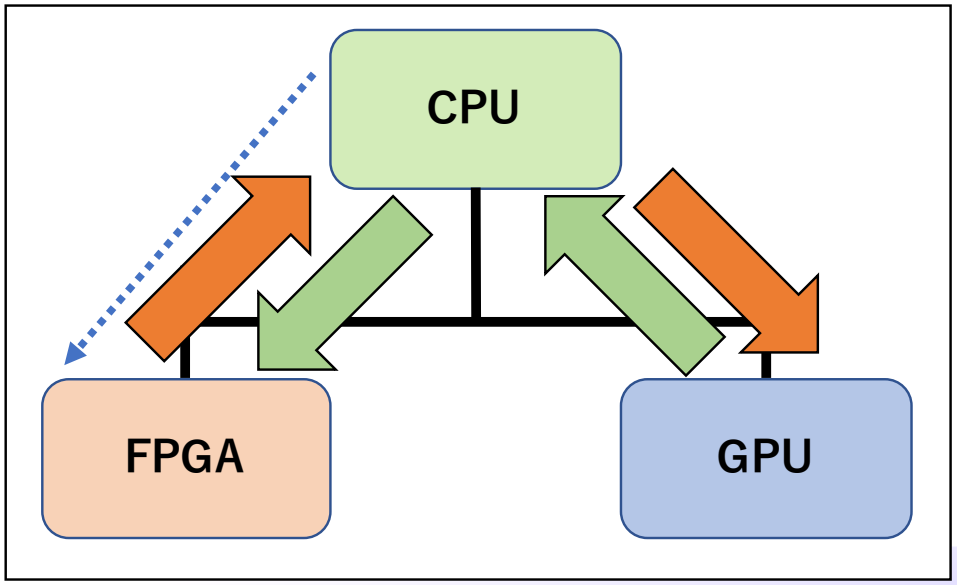
Circuit resource utilization

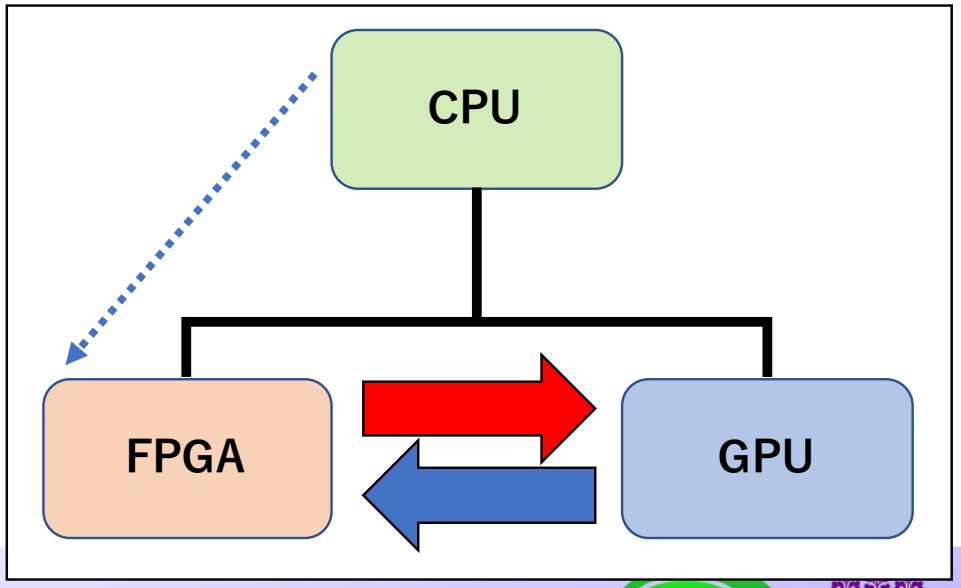
	ALMs	Registers	M20K	MLAB	MLAB size	DSP	Freq.
w/o autorun	228,610 (54%)	473,747 (55%)	1,839 (68%)	4,330	47,968 bits	536 (35%)	228.57 MHz
w/ autorun	228,835 (54%)	467,225 (55%)	1,716 (63%)	7,350	138,288 bits	536 (35%)	236.11 MHz
difference	+225	-6,255	-123	+3,020	+90,320	0	+7.54

- largest resource use is on M20K (63%)
  - actually 53.3% (without BSP use)
- DSP utilization is only 53%
- We can achieve up to 2x more speed

GPU-FPGA communication

- FPGA is only ready for CPU-FPGA data exchange through PCIe
- We developed FPGA-GPU DMA module to be activated by FPGA
  - Currently, still needs CPU help → FPGA standalone





Performance comparision

Latency (minimum)

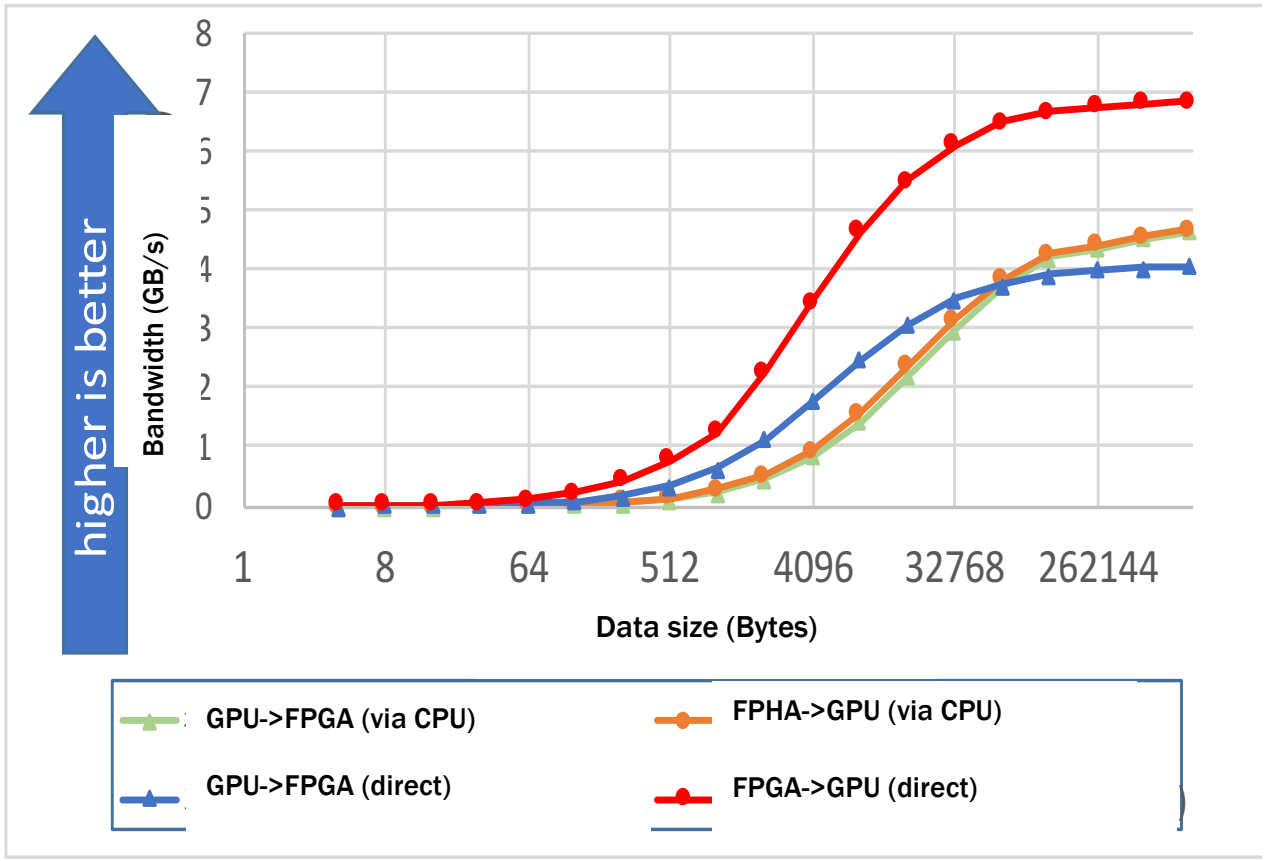
GPU -> FPGA DMA

Communication	Min. Latency
GPU-CPU-FPGA	3.92 µsec
GPU-FPGA	1.45 µsec

FPGA -> GPU DMA

Communication	Min. Latency
FPGA-CPU-GPU	3.63 µsec
FPGA=GPU	0.43 µsec

Bandwidth



Next Step

- Precision controlling
  - for ART and ARGOT, SP is too much, HP is not balanced
  - finding best (e, m, s) combination
    e=exponent m=mantissa s=(exponent digit shift)
- Combining communication and computation
  - OpenCL computing kernels binding with OpenCL Ethernet communication layer kernels with OpenCL Channel (by Intel SDK)
- Combining GPU and FPGA
  - Basic system is done, but not yet combined for full-control by FPGA
- Final work: How to Program ???
  - FPGA-parallel programming framework with high level parallel language XcalableACC
  - Will be started as a new theme for DOE-MEXT collaboration with FTG of ORNL (Jeff Vetter) to welcome OpenARC compiler for OpenACC->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
  - one goal: XcalableACC with OpenACC-FPGA compiler and FPGA-Ethernet link
  - Collaboration with FTG, ORNL

Summary

- FPGA for HPC is very attractive theme for next generation of accelerated platform
- FPGA is usable not only for computing but also for communication
- 360-degree system to cover highly parallel SIMT computing by GPU and flexible processing on FPGA with communication feature
- OpenCL-enabled programming including communication for application users
- CCS, U. Tsukuba is moving forward to realize AiS concept on next generation multi-hetero supercomputing