IXPUG HPC Asia 2020 @ Fukuoka, Japan
15:20-15:40, January 17th, 2020

筑波大学
University of Tsukuba

OpenCL-enabled GPU–FPGA Accelerated Computing with Inter-FPGA Communication

Ryohei Kobayashi[1,2)], Norihisa Fujita[1)],
Yoshiki Yamaguchi[2,1)], Ayumi Nakamichi[2)],
Taisuke Boku[1,2)]

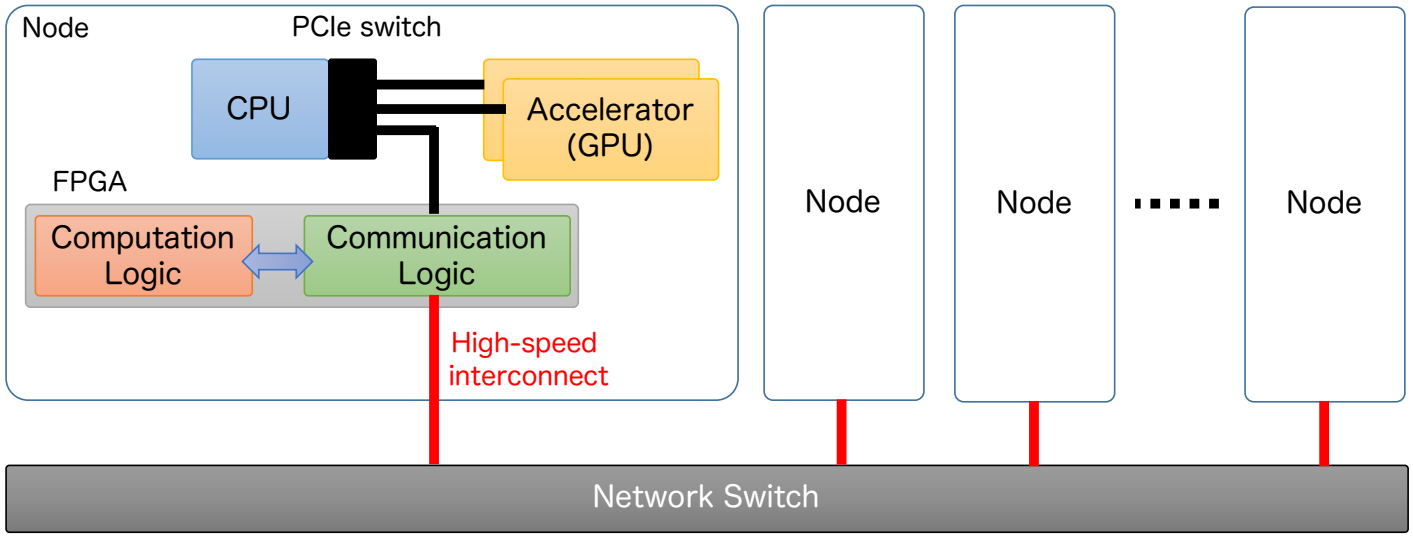
1: Center for Computational Sciences, University of Tsukuba
2: Graduate School of Systems and Information Engineering

Accelerators in HPC

- The most popular one: GPU
  - Strengths
    - ✓ large scale SIMD (SIMT) fabric in a chip
    - ✓ high bandwidth memory (GDDR5, HBM)
  - GPUs do not work well on applications that employ
    - ✓ partially poor parallelism
    - ✓ non-regular computation (warp divergence)
    - ✓ frequent inter-node communication
- FPGAs have been emerging in HPC
  - true co-designing with applications (**indispensable**)
  - OpenCL-based FPGA development toolchains are available
  - high bandwidth interconnect: ~100 Gbps x 4

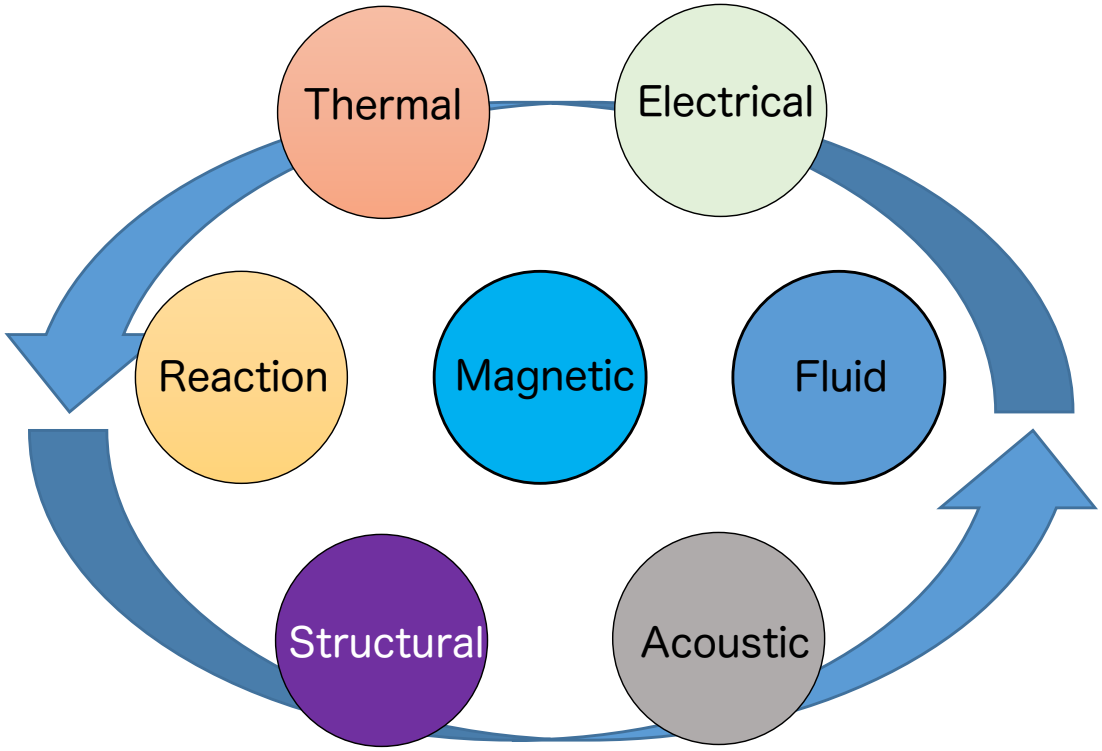
Accelerator in Switch (AiS) concept

- What's this?
  - using FPGA for not only computation offloading but also communication
  - covering GPU non-suited computation by FPGA
  - combining computation offloading and ultra-low latency communication among FPGAs
  - especially effective on communication-related small/medium computation (such as collective communication)
  - OpenCL-enable programming for application users



Why GPU-FPGA coupling is needed?

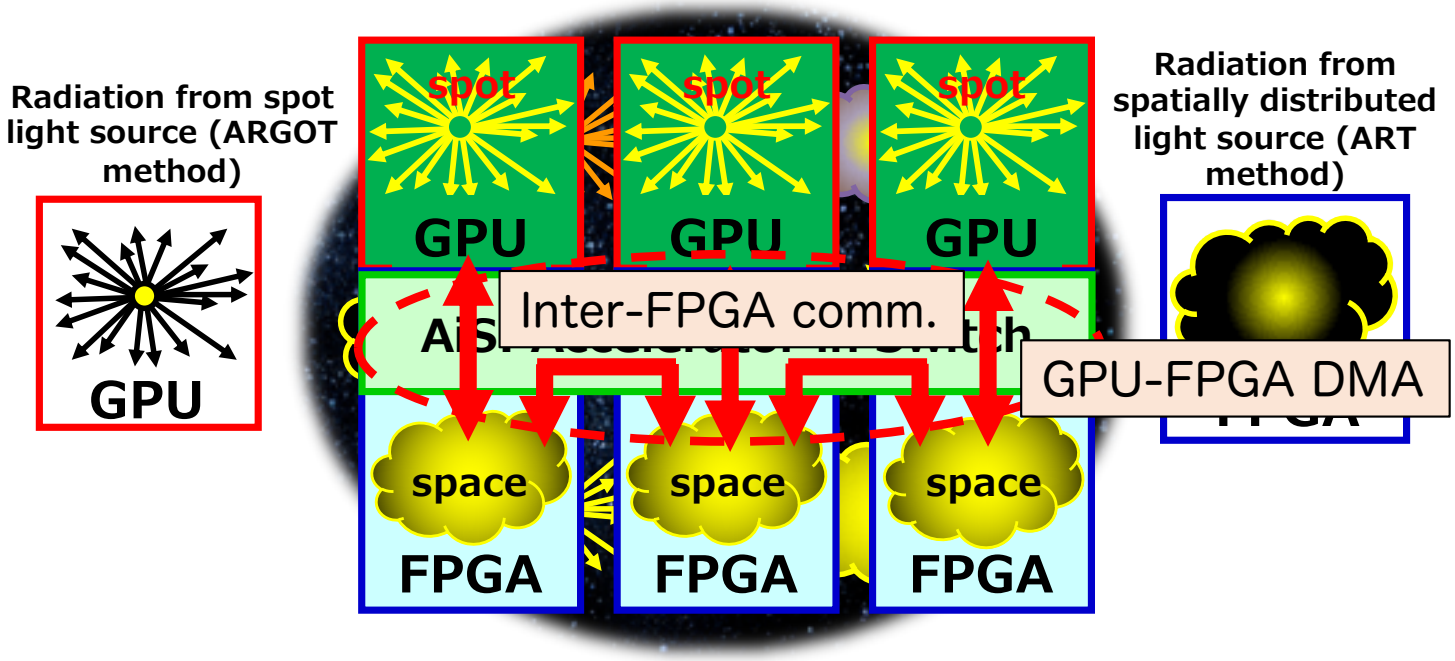
- Keyword: Multiphysics
  - Simulations with multiple interacting physical properties
  - Various computations are included within a simulation
    → Hard to accelerate simulation speed by GPU only



One of Multiphysics applications

- ARGOT code: Radiation transfer simulation
  - The radiation transfer from spot light and spatially distributed light st… of first object in spa…
  - Th… radiation a… computation on GPU and … separately
    → needs to combine both devices for high speed simulation

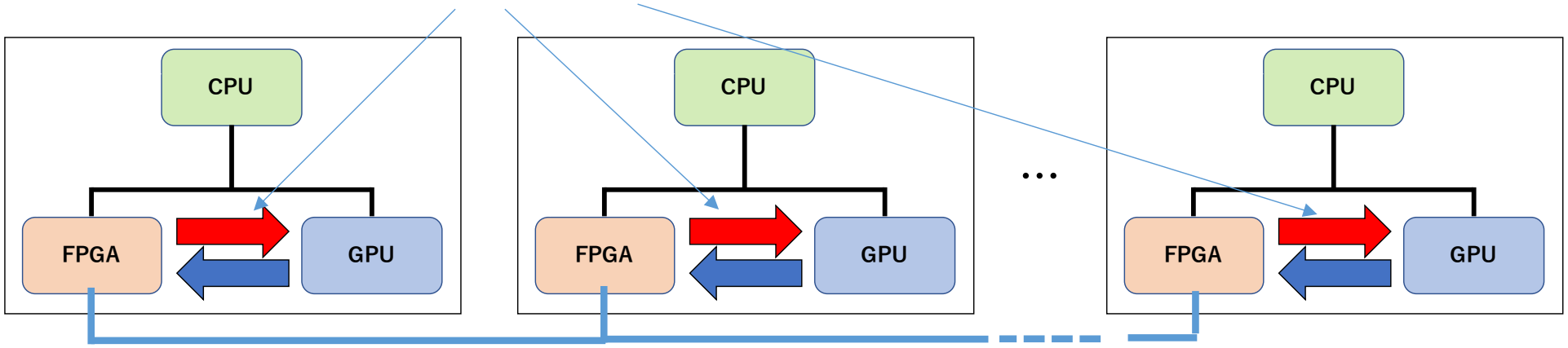
How to use them simultaneously?



Today's topic

- What we've done for this presentation is...
  - to combine GPU-FPGA DMA[1] and inter-FPGA comm.[2]
    - ✓ Both of data communication are invoked from OpenCL kernel

Data communication with GPU-FPGA DMA transfer



FPGA-to-FPGA data communication

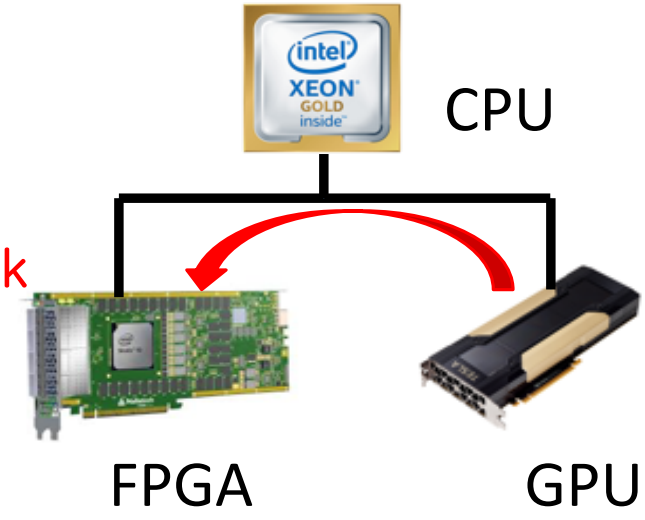
[1] Ryohei Kobayashi et al., "GPU-FPGA Heterogeneous Computing with OpenCL-enabled Direct Memory Access", pp.489-498, IPDPSW2019 (AsHES), May 20th, 2019.
[2] Norihisa Fujita et al., "Parallel Processing on FPGA Combining Computation and Communication in OpenCL Programming", pp.479-488 , IPDPSW2019 (AsHES), May 20th, 2019.

Data communication with GPU-FPGA DMA transfer

- Example: GPU-to-FPGA data transfer

```
__kernel void fpga_dma(__global float *restrict fpga_mem,
                       const ulong gpu_memadr,
                       const uint id_and_len)
{
  cldesc_t desc;
  // DMA transfer GPU -> FPGA
  desc.src = gpu_memadr;
  desc.dst = (ulong)(&fpga_mem[0]);
  desc.id_and_len = id_and_len;
  write_channel_intel(fpga_dma, desc);
  ulong status = read_channel_intel(dma_stat);
}
```

GPU-to-FPGA DMA kick



FPGA-to-FPGA data communication

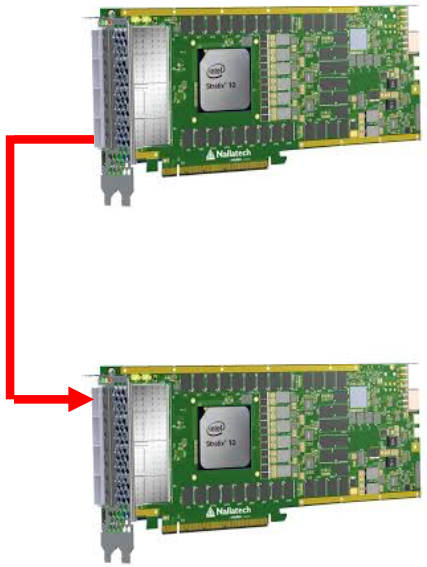
- Enabling user level programming

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(network_out, v);
    }
}
```

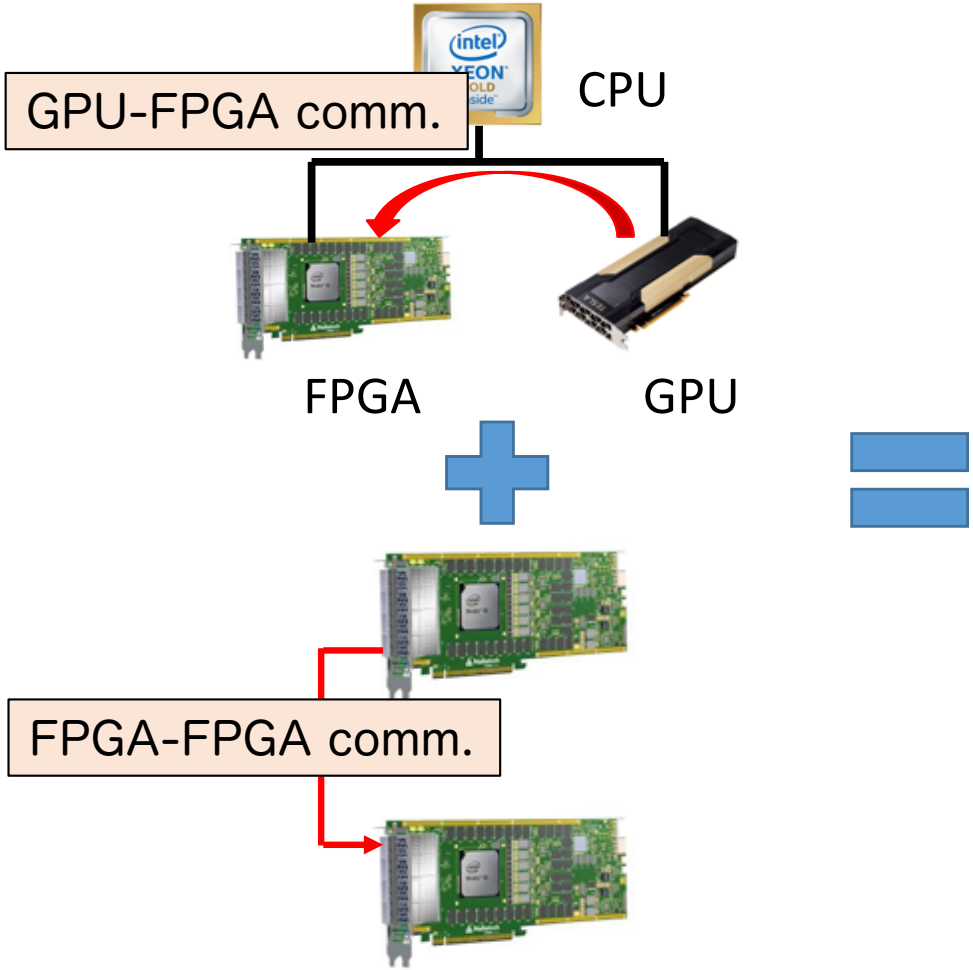
receiver code on FPGA2

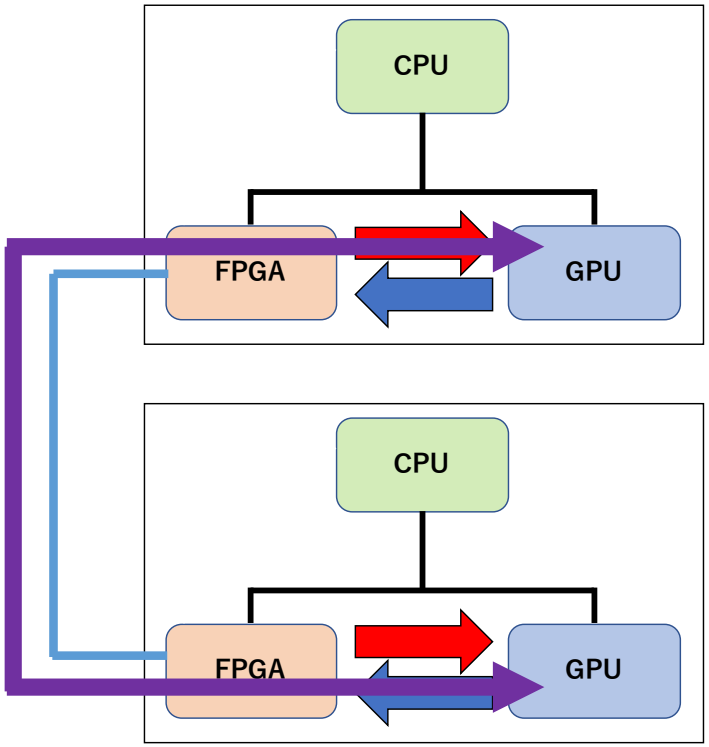
```
__kernel void receiver(__global float* restrict x, int n) {
    for (int i = 0; i < n; i++) {
        float v = read_channel_intel(network_in);
        x[i] = v;
    }
}
```



GPU-FPGA comm. + FPGA-FPGA comm.

- Using of both features simultaneously
  - E.g. pingpong benchmark program with 2 comp. nodes



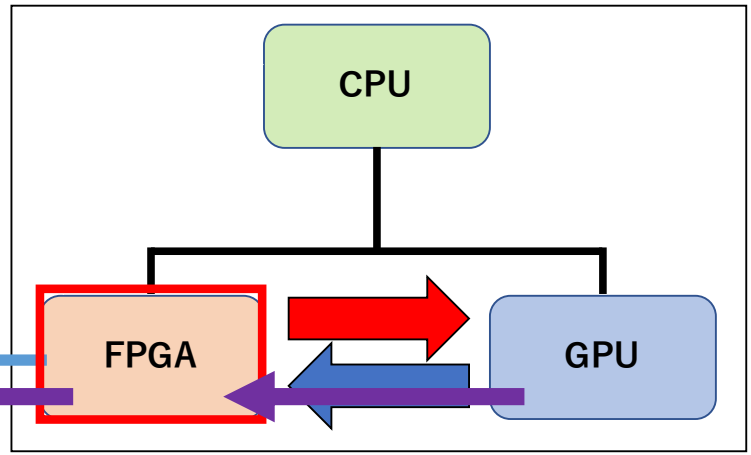


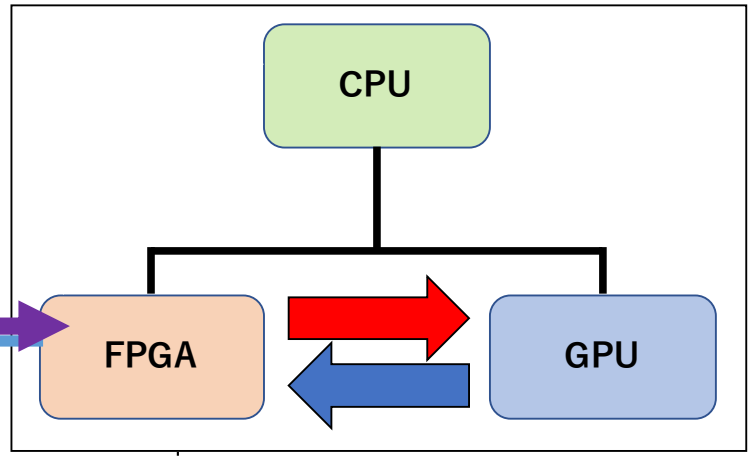
Pingpong benchmark
between two GPUs over different node

```
__kernel void ping(
    __global float* restrict data,
    int n
) {
  uchar request = 1;  // GPU-to-FPGA mode
  write_channel_intel(dma_req, request);
  for (int i = 0; i < n; i++) {
    float v = read_channel_intel(ping_data);
    write_channel_intel(send, v);
  }
}
```

Inter-FPGA comm.

Send DMA request






```
__kernel void gpu_fpga_dma(
          __global volatile float* restrict recv_data,
          int n,
          const ulong gpu_memadr
          ) {
  uchar request = read_channel_intel(dma_req);
  if (request == 0) {  // FPGA-to-GPU mode

      ~ FPGA-to-GPU DMA procedure ~

  } else if (request == 1) {  // GPU-to-FPGA mode
    ulong ret  = dma_gpu_to_fpga(gpu_memadr, recv_data, n*sizeof(float));
    for (int i = 0; i < n; i++) {
      write_channel_intel(ping_data, recv_data[i]);
    }
  }
}
```

Return ping-data

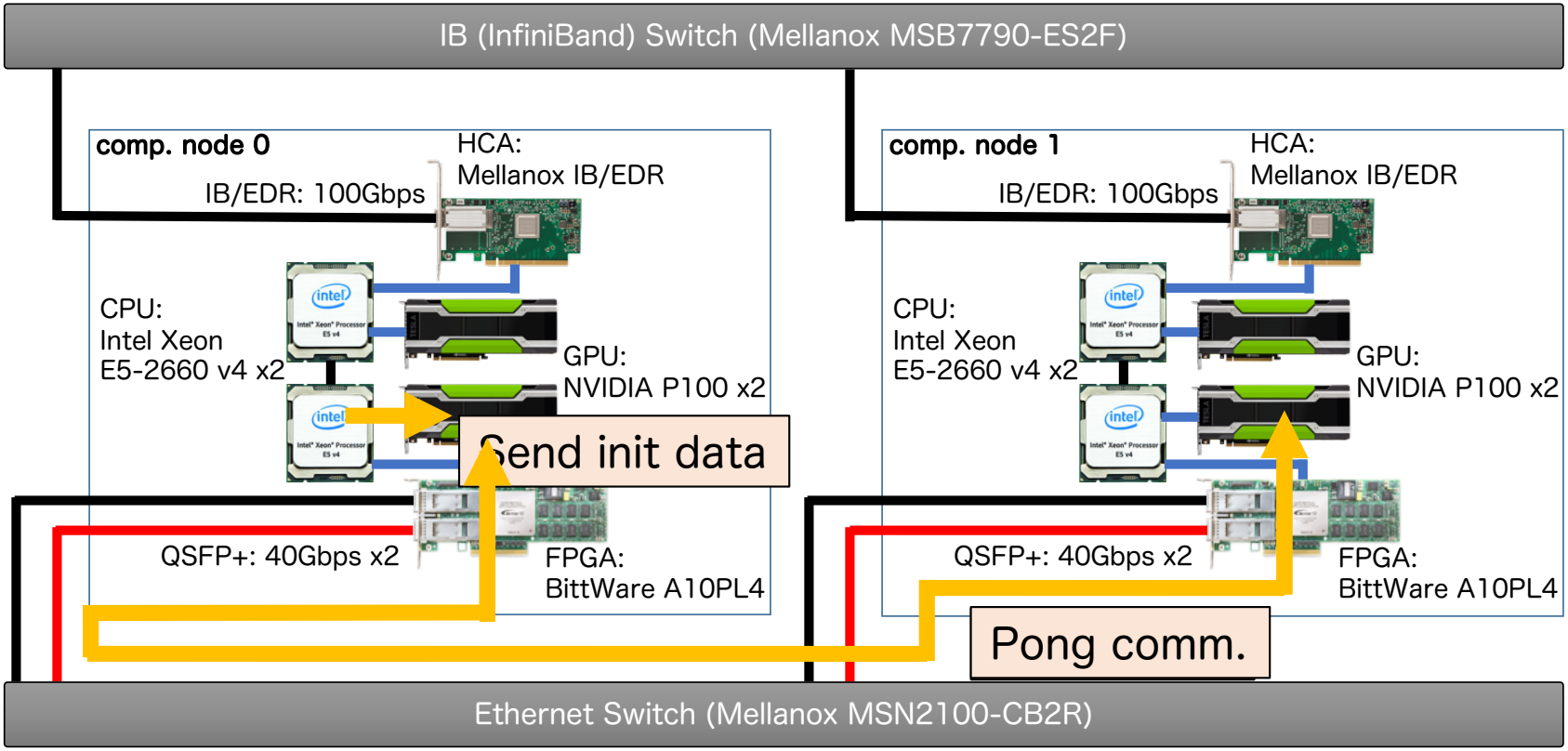
GPU-to-FPGA DMA kick

Procedure of Ping comm.

Experimental Settings

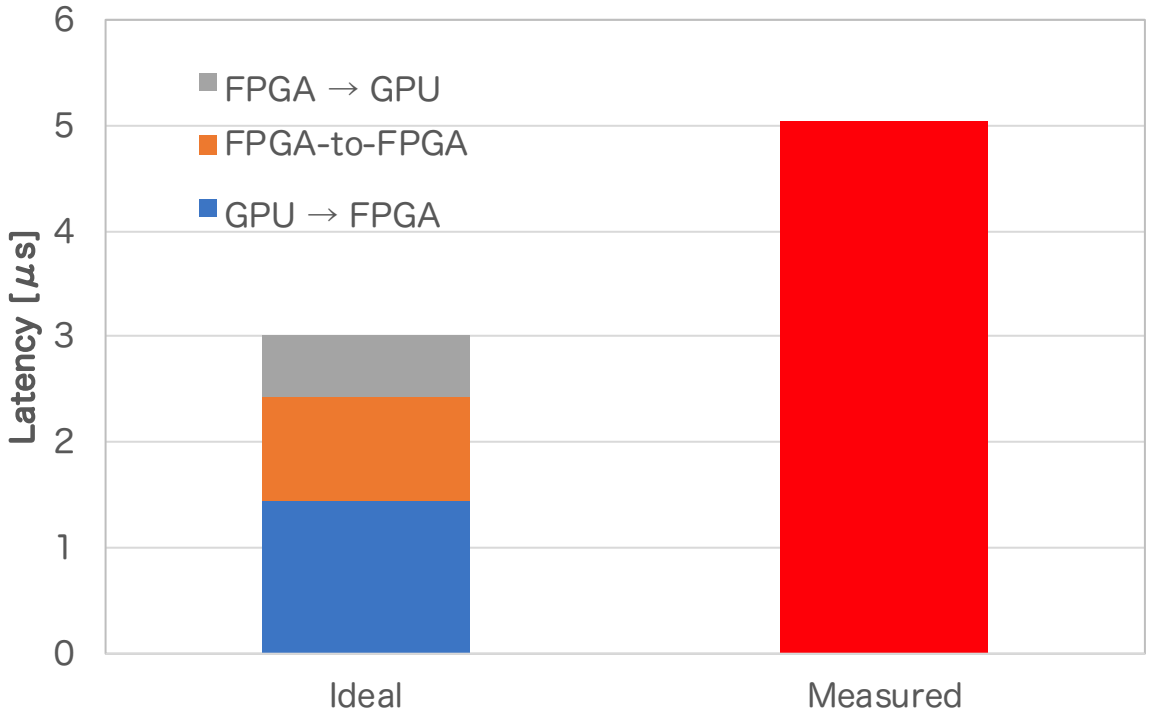
Hardware specification	
CPU	Intel Xeon E5-2660 v4 x2
Host Memory	DDR4-2400 16GB x4
GPU	NVIDIA Tesla P100 x2 (PCIe Gen3 x16)
FPGA	Intel Arria 10 GX (BittWare A10PL4) (PCIe Gen3 x8)

Software specification	
OS	CentOS 7.3
Host Compiler	gcc 4.8.5, g++ 4.8.5
MPI	OpenMPI 3.0.1
GPU Compiler	CUDA 9.1.85
FPGA Compiler	Intel Quartus Prime Pro 17.1.2.304



Communication latency result between GPUs over different nodes

- Communication latency for 4-byte transferring
  - The “ideal” latency is based on [1, 2]
  - Performance degradation is because of non-optimized impl.



[1] Ryohei Kobayashi et al., “GPU-FPGA Heterogeneous Computing with OpenCL-enabled Direct Memory Access”, pp.489-498, IPDPSW2019 (AsHES), May 20th, 2019.
[2] Norihisa Fujita et al., “Parallel Processing on FPGA Combining Computation and Communication in OpenCL Programming”, pp.479-488 , IPDPSW2019 (AsHES), May 20th, 2019.

The most important finding in this evaluation is that...

- We have confirmed our proposed method enables GPUs and FPGAs to work together over different nodes
  - The latency value has no meaning
    - ✓ For JUST performing GPU-to-GPU comm., GPUDirect is better than Ideal latency of our proposed method
  - We would like to make FPGA not only communication but also computation, and this experiment shows that it is possible
  - Accelerator-centric (GPU+FPGA) application can be implemented without requiring deep FPGA knowledge and significant engineering effort because there is no HDL

Conclusion

- Proposal
  - An approach for GPU–FPGA accelerated computing with the OpenCL programming framework
- What we did is...
  - to confirm that our proposed method makes it possible for GPUs and FPGAs in different nodes to work together
    - ✓ Data communication with GPU-FPGA DMA transfer
    - ✓ FPGA-to-FPGA data communication
- Contribution
  - Application developers can implement CPU-non-intervention application code running on multiple GPUs and FPGAs without writing any HDL code

Future work

- How FPGA knows GPU computation completion?
  - A sophisticated synchronization mechanism is needed
- We do not want to write multiple code!! (CUDA, OpenCL, etc)
  - needs a comprehensive programming framework
    - ✓ Currently, we are developing such framework with OpenACC
- Targeting real applications
    - ✓ Currently, we are focusing on astrophysics application

Cygnus:
A Next-gen supercomputer at CCS

- An AiS proof-of-concept system
  - Operation started in April 2019
  - 2x Intel Xeon CPUs, 4x NVIDIA V100 GPUs, 2x Intel Stratix10 FPGAs
  - Deneb: 48 nodes
    - ✓ CPU + GPU
  - Albireo: 32 nodes
    - ✓ CPU + GPU + FPGA
    - ✓ 2D torus network for FPGAs
      - 100Gbps per link

https://www.nallatech.com/store/fpga-accelerated-computing/pcie-accelerator-cards/nallatech-520-compute-acceleration-card-stratix-10-fpga/

Promotion video is available on Youtube!!
https://www.youtube.com/watch?v=Vrlq5Q-3f0o&t=190s