Recent Activities in CCS

Taisuke Boku

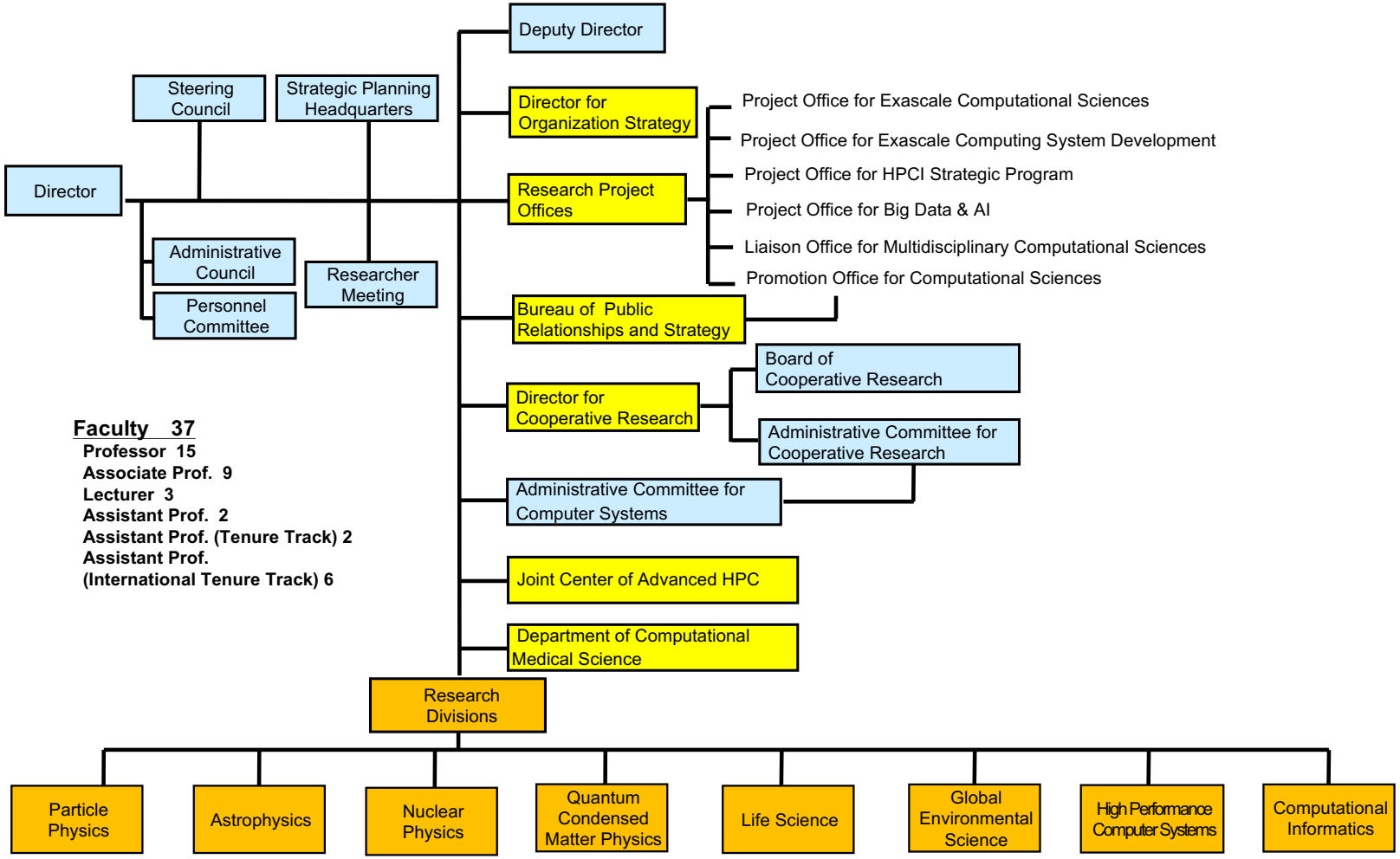
Director, Center for Computational Sciences

University of Tsukuba

Center for Computational Sciences, University of Tsukuba

- CCS = Center for Computational Sciences
- Established in 1992
  - 12 years as Center for Computational Physics
  - Reorganized as Center for Computational Sciences in 2004
- Daily collaborative researches with two kinds of researchers (40 in total)
  - Computational Scientists ~ 65%
    who have NEEDS (applications)
  - Computer Scientists ~ 35%
    who have SEEDS (system & solution)

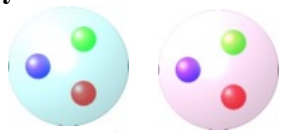
Organization of CCS



Multidisciplinary Computational Sciences

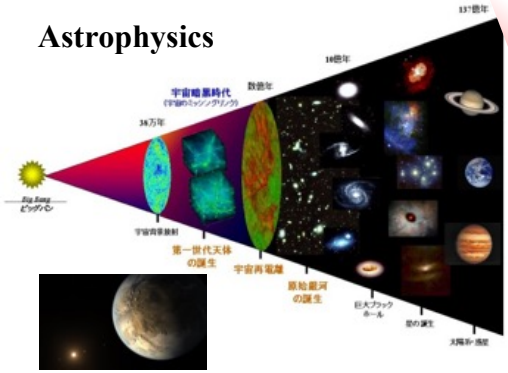
Cosmic History

Particle physics



Nuclear physics

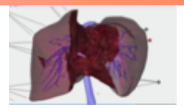
Astrophysics



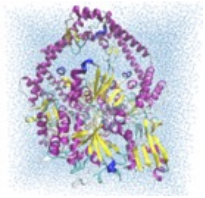
Exoplanets
Astrobiology

Origin of Life

Computational Medicine

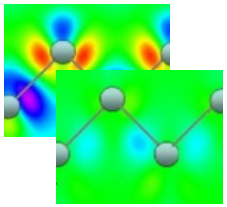


Codesign of Supercomputers



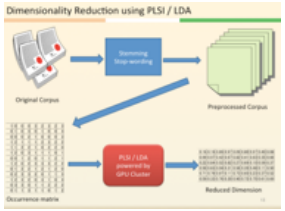
Life science

Materials sciences

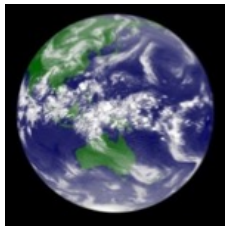


Big data・AI

Machine learning



Visualization



Urban climate
Global warming

Global Environments

MCRP (Multidisciplinary Cooperative Research Program)

- Supercomputer resource sharing
  - Domestic -> International
  - Multiple sizes and categories
- Travel support for Meetings
  - Financial Support for travel expenses to attend scientific meetings
- International Schools
  - International expansion of “Computational Science Literacy”
- Long-term visitors
  - Those who stay from a few weeks to a few months are encouraged to carry out MCRP


Oakforest-PACS: *Japan's Fastest General Purpose (non-accelerated) System*

- 25 PFLOPS peak
- 8208 KNL CPUs
- FBB Fat-Tree by OmniPath
- HPL 13.55 PFLOPS
  #1 in Japan
  #6 in World
- HPCG #3 in World

Full operation started Dec. 2016

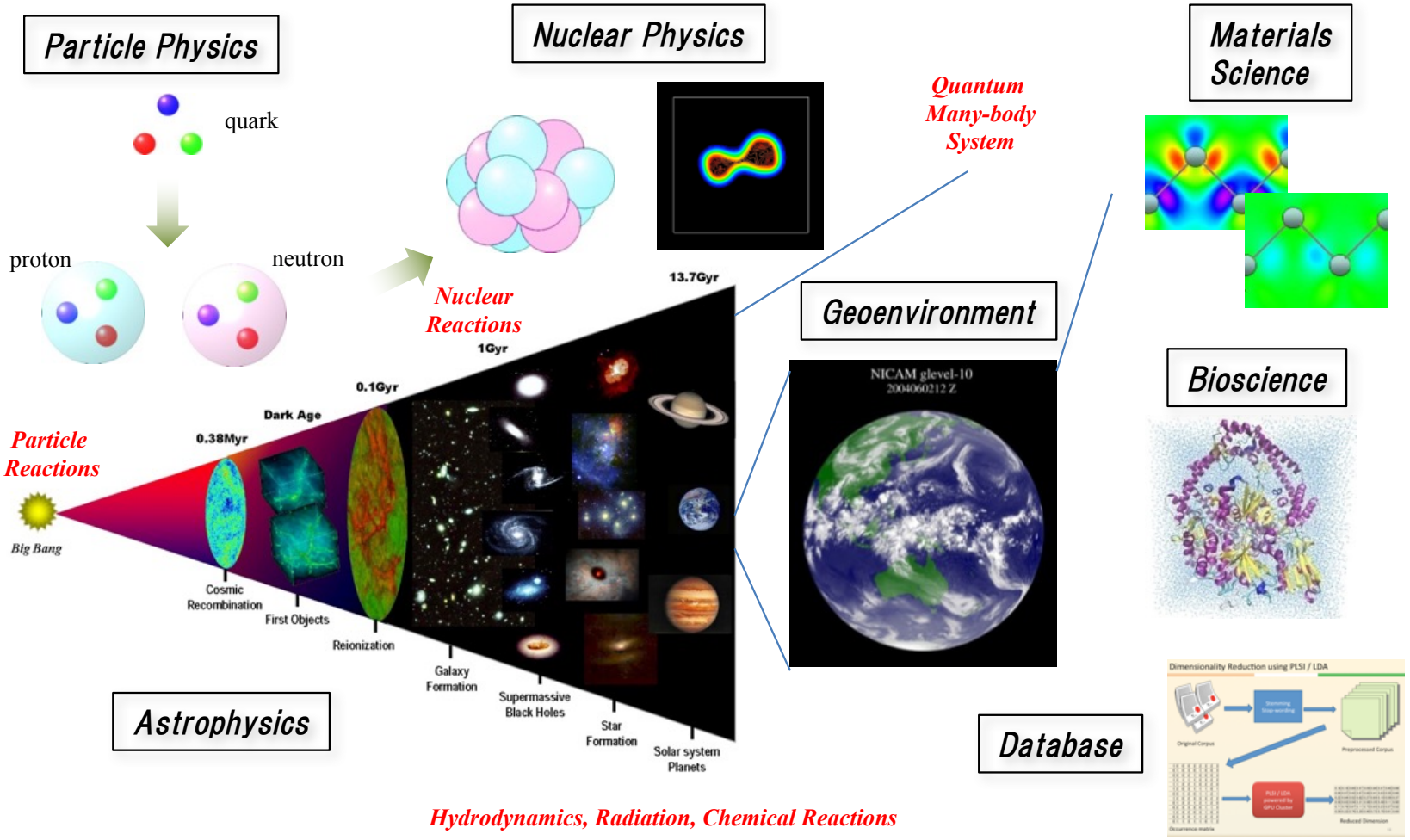
Supporting International Research Collaboration

- Proceeding tightly coupled international collaboration with people exchange with an appropriate duration

- Med-Term International Invitation to CCS
  - a couple weeks ~ months to stay in CCS
  - supporting travel fare, staying and some rewards

- Monthly Sabbatical
  - a couple of weeks ~ months to stay oversea research insititutes
  - covering university-wide program for sabbatical (3~12 months)

Computational Sciences

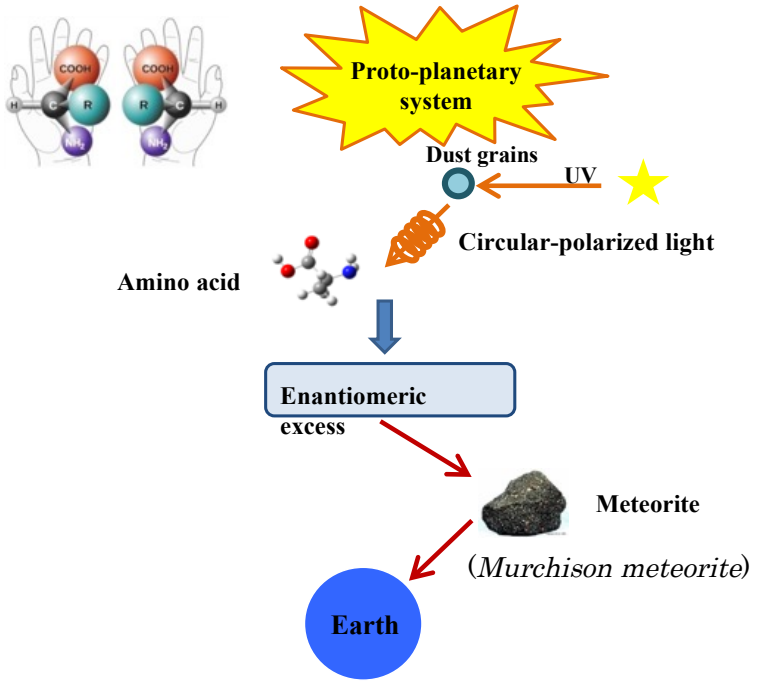


Computational Astrobiology

Collaboration of Astrophysics, Biophysics, and Planetary Science

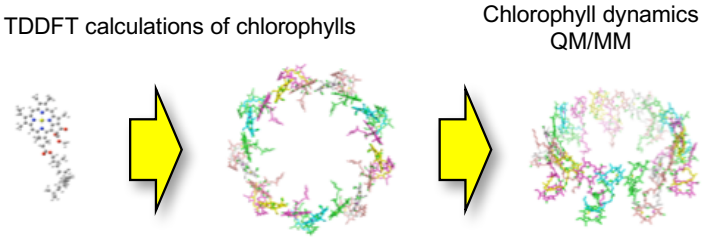
Cosmic origin of L-amino acid

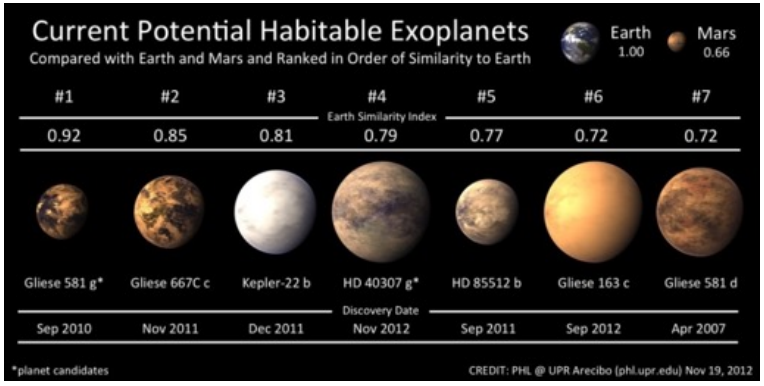
Enantiomeric excess of Amino Acid by interstellar circular-polarized light



Photosynthesis on extra-solar planets

Red edge as a biomerker

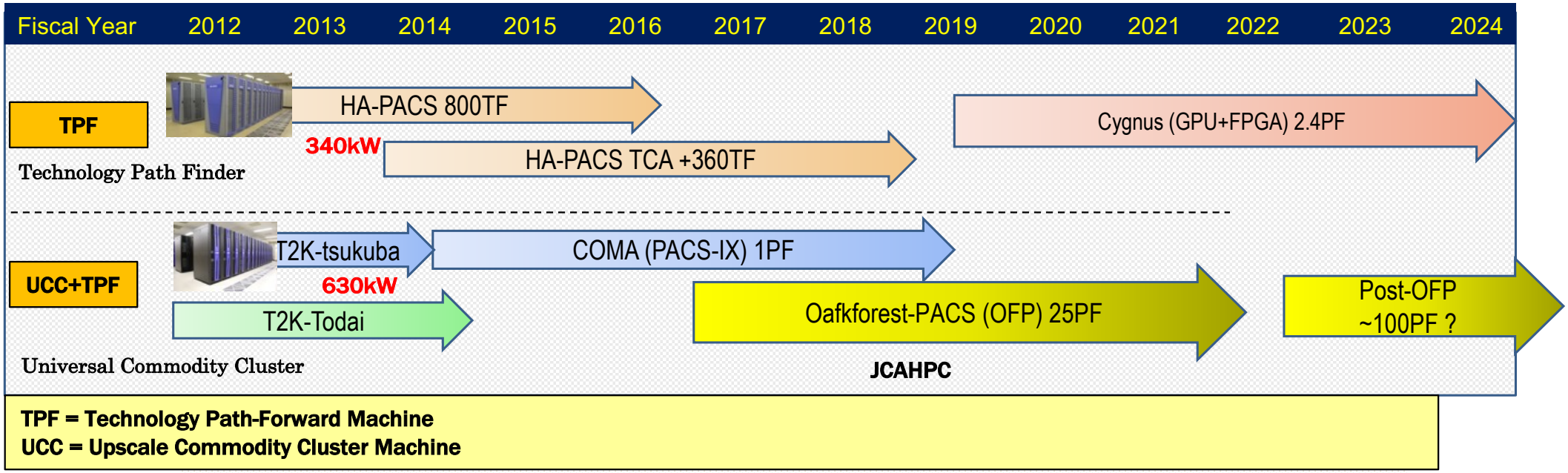




Collaboration with Post-K Project (FUGAKU)

- Post-K (so called Exascale Project in Japan) Project has been launched on Apr. 2014 at RIKEN AICS, Kobe (currently R-CCS)
- Collaboration with Post-K Project
  - Application:
    ~ Post-K Prioritized Application Field 9
    "Analysis of Fundamentals on Universe and Development"
    (PI: Shinya Aoki)
  - System: Codesign on system development of Post-K Computer
    "Parallel Programming Environment and Interconnection Network Research on Post-K"
    (PI: Taisuke Boku)
  - Sub-topics of applications: quantum material science in Field 7 (by Kazuhiro Yabana)
- Using JCAHPC resource OFP as an application and system software development platform for Post-K

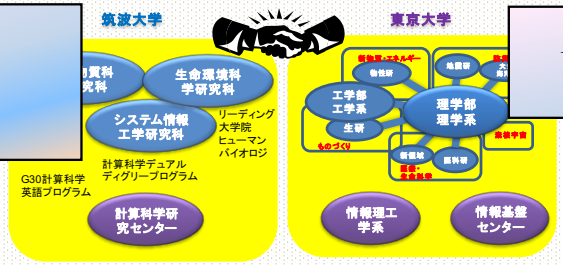
Future Plans of Computing System Development



TPF = Technology Path-Forward Machine
UCC = Upscale Commodity Cluster Machine

◆ Joint Center for Advanced High Performance Computing (JCAHPC)

Center for Computational Sciences
U. Tsukuba



Information Technology Center
U. Tokyo

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

2014
9th gen: COMA

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
2019	Cygnus (PACS-X)	2.5 PFLOPS

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

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 parallelism
    - large scale SIMD (STMD) mechanism in a chip
    - high bandwidth memory (HBM, HBM2) and local memory
  - insufficient for cases with...
    - not enough parallelism
    - not regular computation (warp splitting)
    - frequent inter-node communication (kernel switch, go back to CPU)

NVIDIA Tesla V100 (Volta)
with PCIe interafce

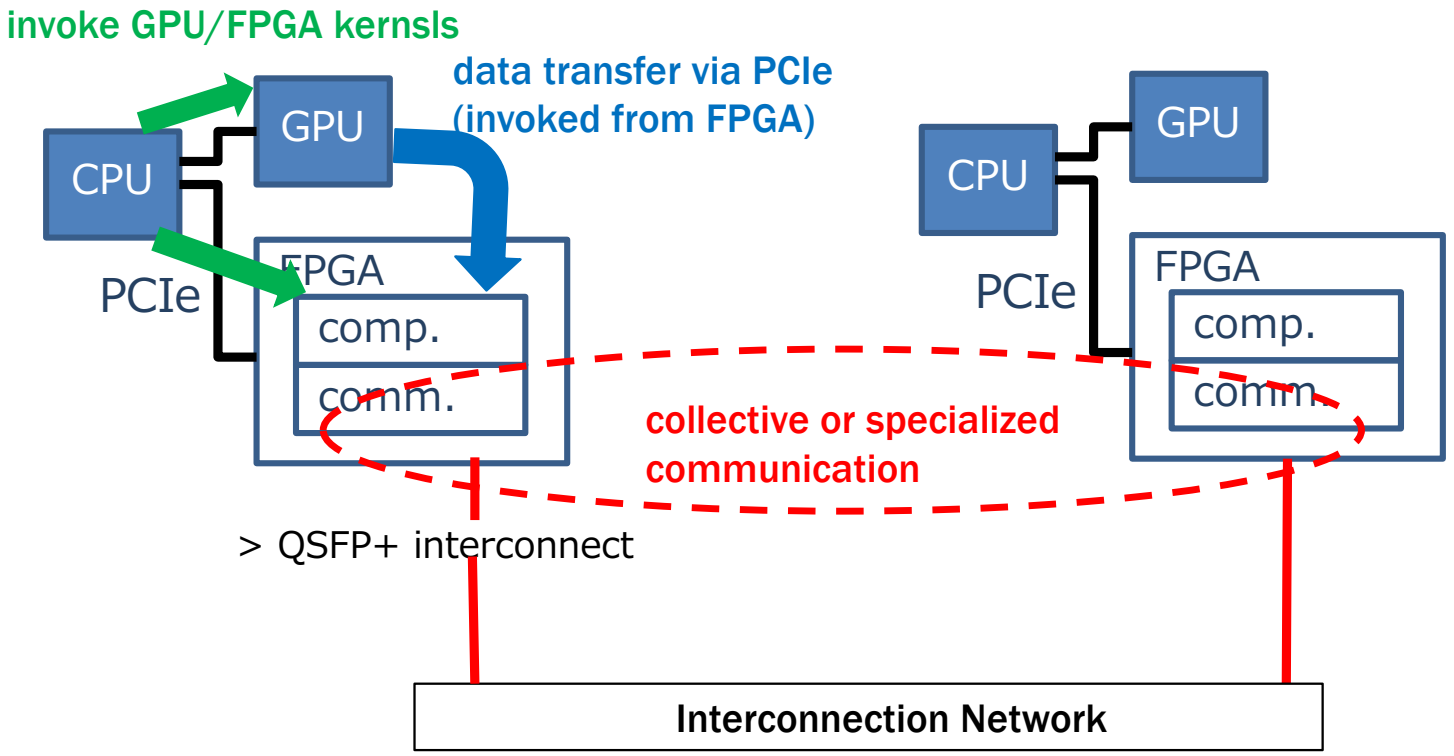
FPGA (Field Programmable Gate Array)

- Goodness of recent FPGA for HPC
  - True codesigning with applications (essential)
  - Programmability improvement: OpenCL, other high level languages
  - High performance interconnect: 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: 1-gen older than high end CPU/GPU
    -> be improved by HBM (Stratix10)

Nallatech 520N with Intel Stratix10 FPGA equipped with 4x 100Gbps optical interconnection interfaces

AiS: conceptual model of Accelerator in Switch

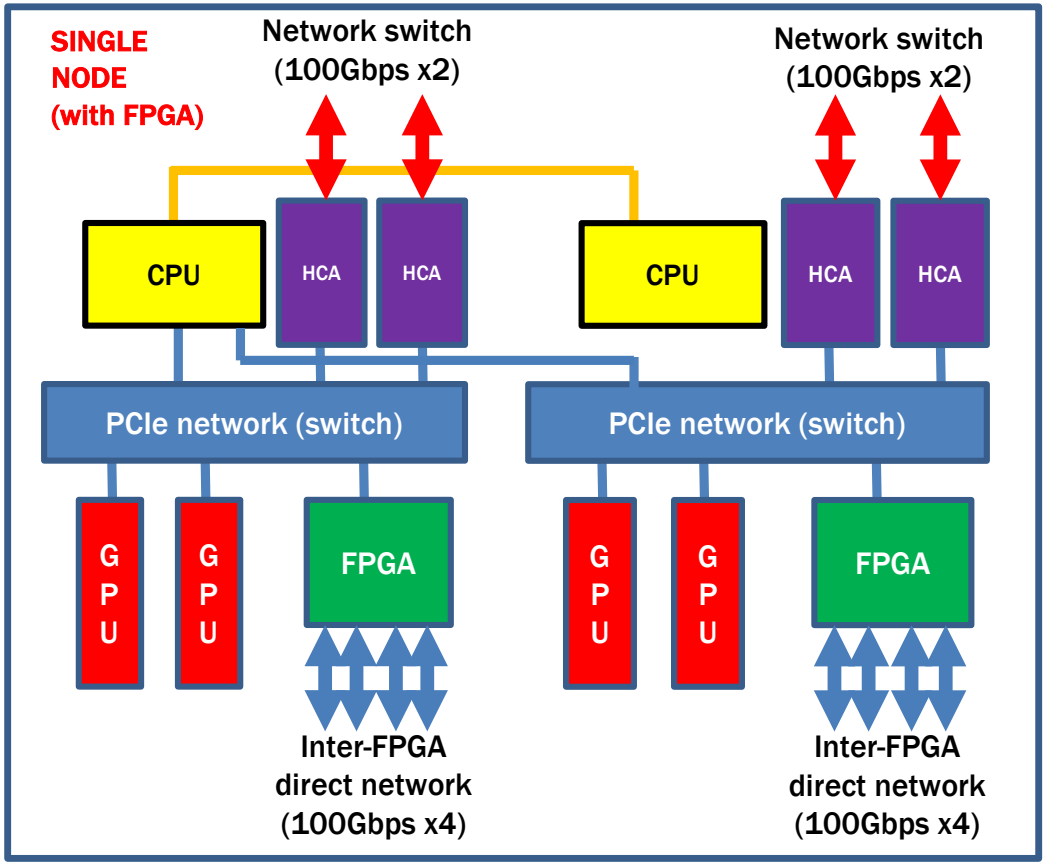
- FPGA can work both for computation and communication in unified manner
- GPU/CPU can request application-specific communication to FPGA



Cygnus Overlook (CCS, U. Tsukuba, April 2019)

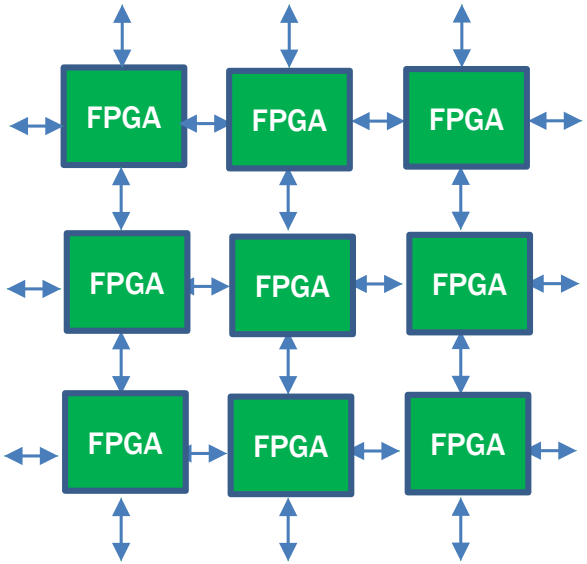
Single node configuration (Albireo) of Cygnus cluster

- 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



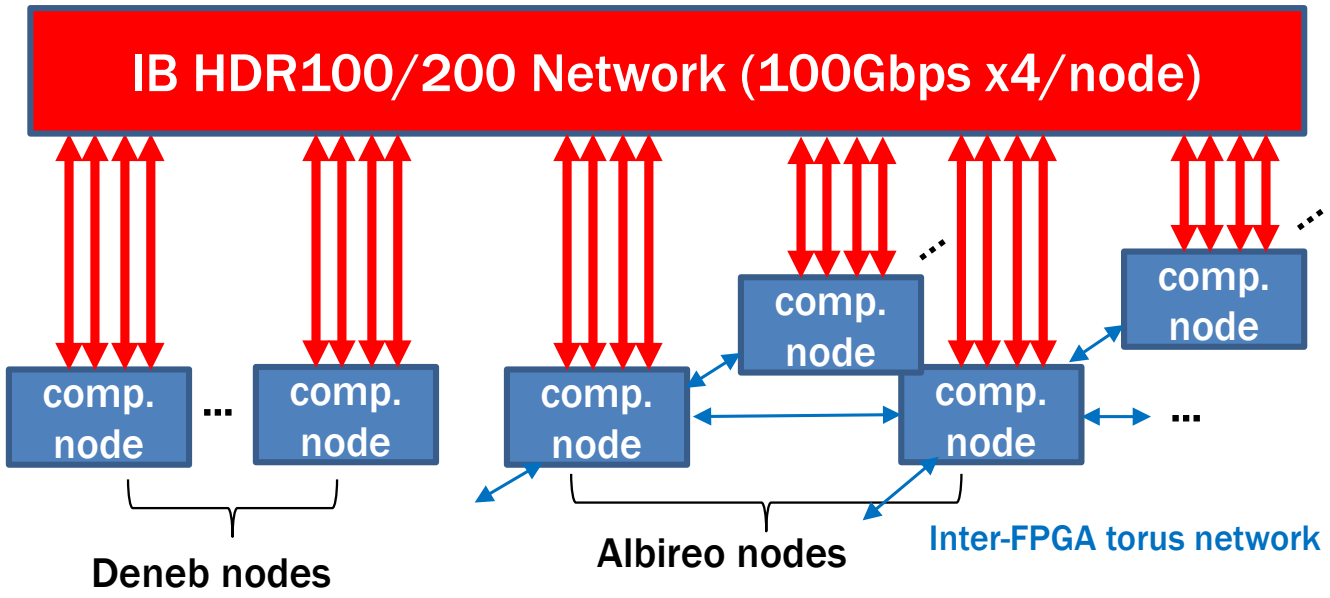
Two types of interconnection network

Inter-FPGA direct network
(only for Albireo nodes)

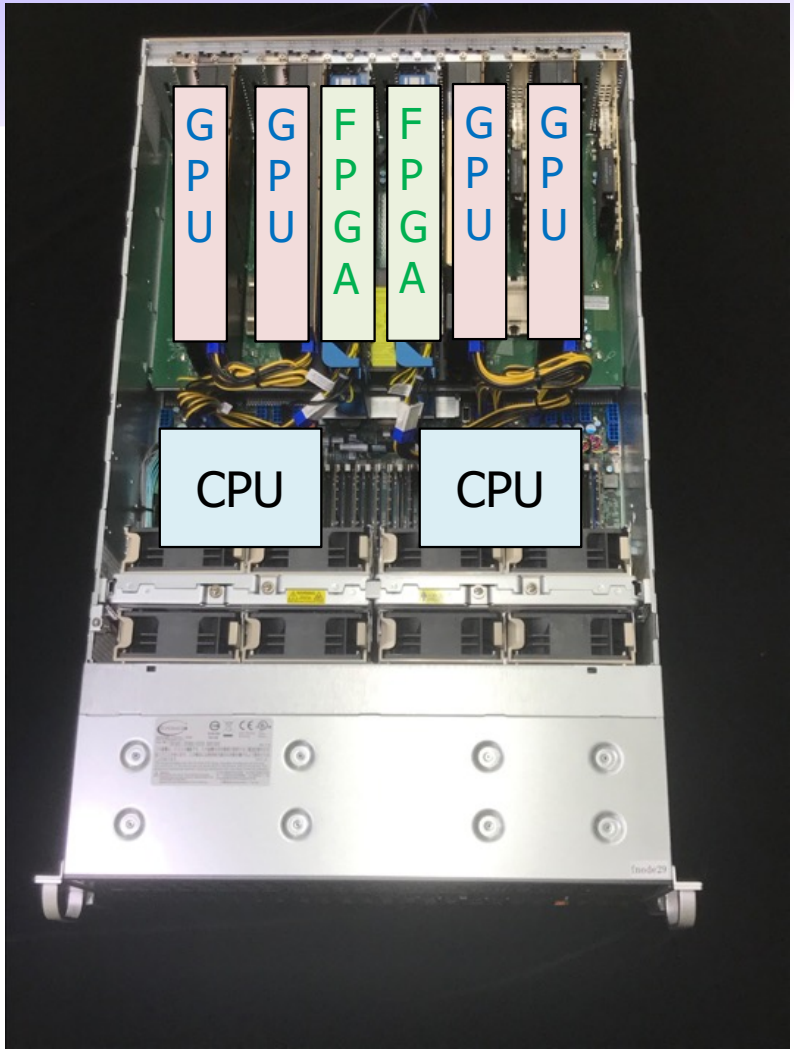


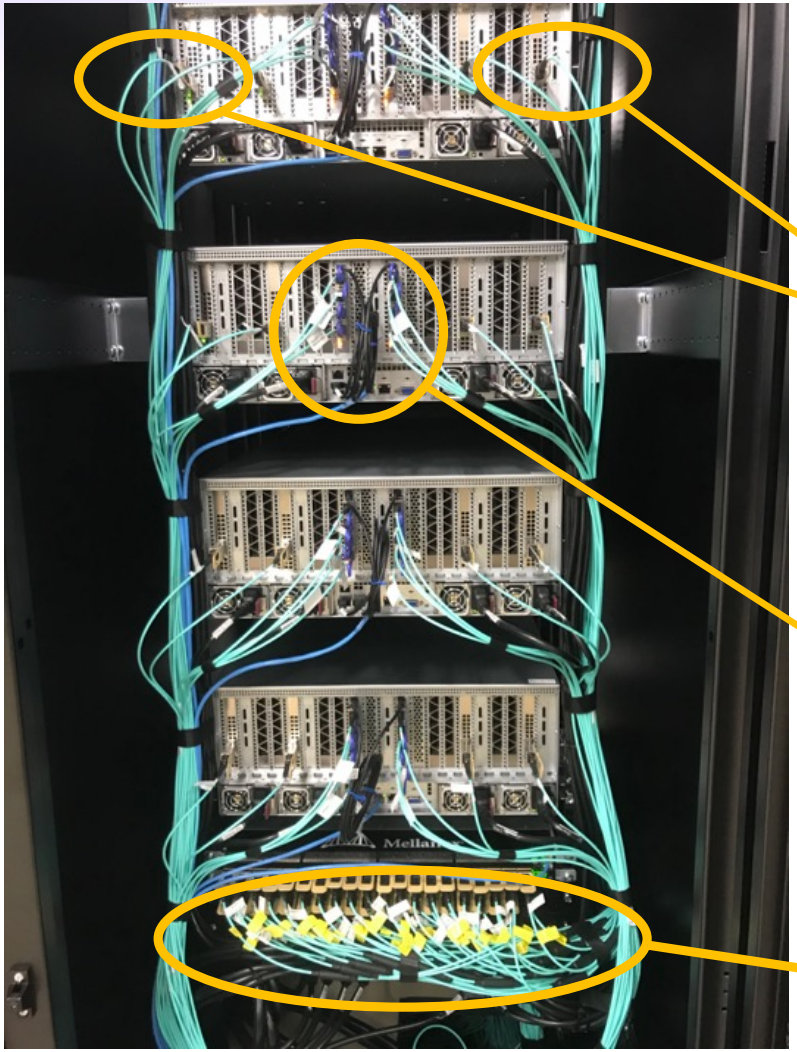
64 of FPGAs on Albireo nodes (2 FPGAS/node) are connected by 8x8 2D torus network without switch

InfiniBand HDR100/200 network for parallel processing communication and shared file system access from all nodes



For all computation nodes (Albireo and Deneb) are connected by full-bisection Fat Tree network with 4 channels of InfiniBand HDR100 (combined to HDR200 switch) for parallel processing communication such as MPI, and also used to access to Lustre shared file system.





IB HDR100 x4
⇨ HDR200 x2

100Gbps x4
FPGA optical
network

IB HDR200
switch (for
full-bisection
Fat-Tree)

Specification of Cygnus

Item	Specification
Peak performance	2.4 PFLOPS DP (GPU: 2.24 PFLOPS, CPU: 0.16 PFLOPS + FPGA: 0.64 SP FLOPS)
# of nodes	80 (32 Albireo nodes, 48 Deneb nodes) => 320x V100 + 64x Stratix10
CPU / node	Intel Xeon Gold x2 sockets
GPU / node	NVIDIA Tesla V100 x4 (PCIe)
FPGA / node	Nallatech 520N with Intel Stratix10 x2 (each with 100Gbps x4 links)
NVMe	Intel NVMe 1.6TB, driven by NVMe-oF Target Offload
Global File System	DDN Lustre, RAID6, 2.5 PB
Interconnection Network	Mellanox InfiniBand HDR100 x4 = 400Gbps/node (SW=HDR200)
Total Network B/W	4 TB/s
Programming Language	CPU: C, C++, Fortran, OpenMP GPU: OpenACC, CUDA FPGA: OpenCL, Verilog HDL
System Integrator	NEC

How to open such a complicated system to application users ?

- OpenCL environment is available
  - ex) Intel FPGA SDK for OpenCL
  - basic computation can be written in OpenCL without Verilog HDL
- 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
  - GPU/FPGA communication is very slow via CPU memory
- 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

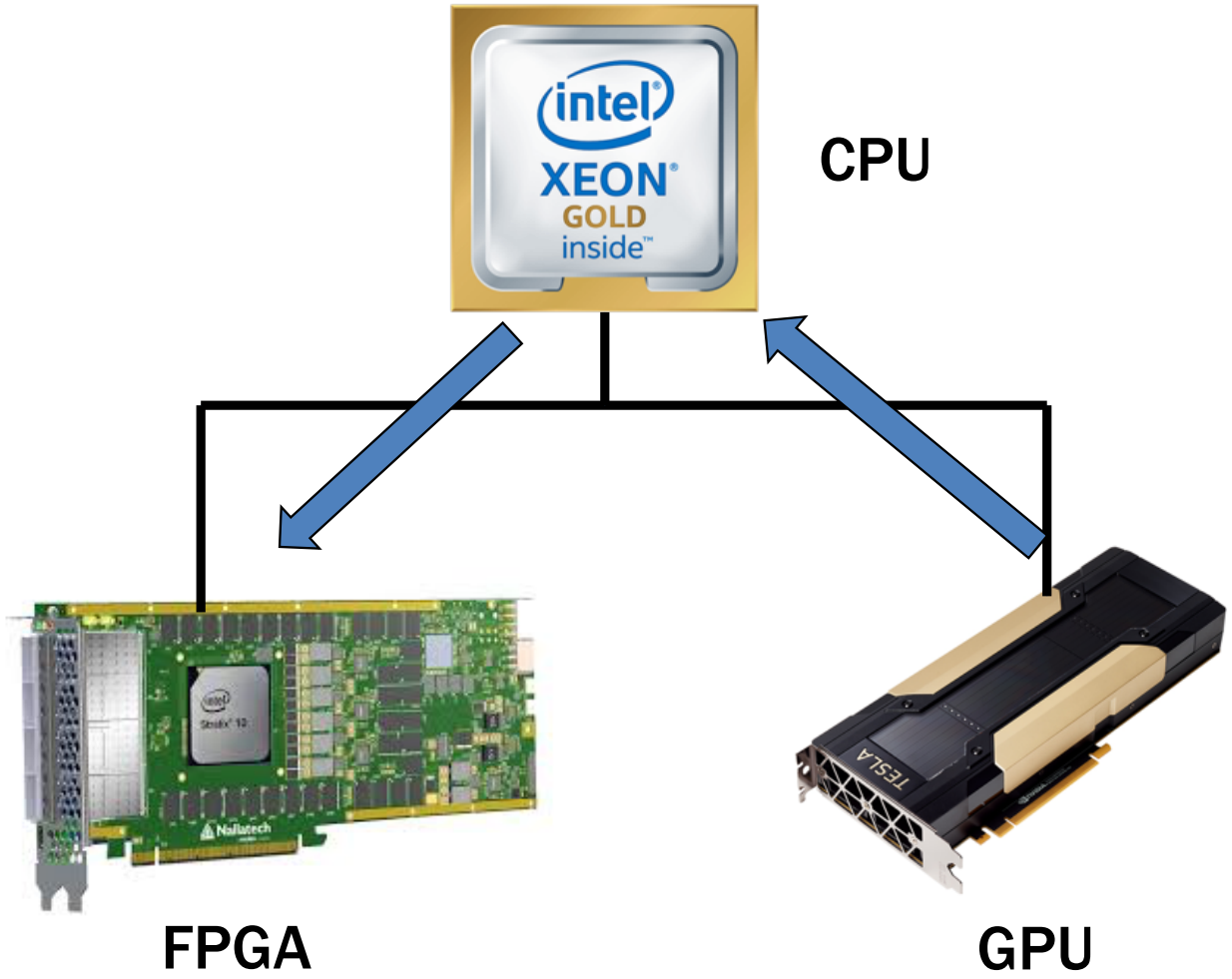
CIRCUS

- FPGA is possible to combine computation and communication in a single framework of pipelined data stream
  - loop computation is pipelined according to the index
  - all the computation part is implemented on logic elements except buffering on memory
  - possible to access IP by chip provides (ex. Intel) for optical link driving
- making all to be programmable on OpenCL
  - scientific users never write Verilog HDL -> perhaps OK with OpenCL
  - key issue for practical HPC cluster: OpenCL-enabled features such ash
    - FPGA communication link
    - GPU/FPGA DMA

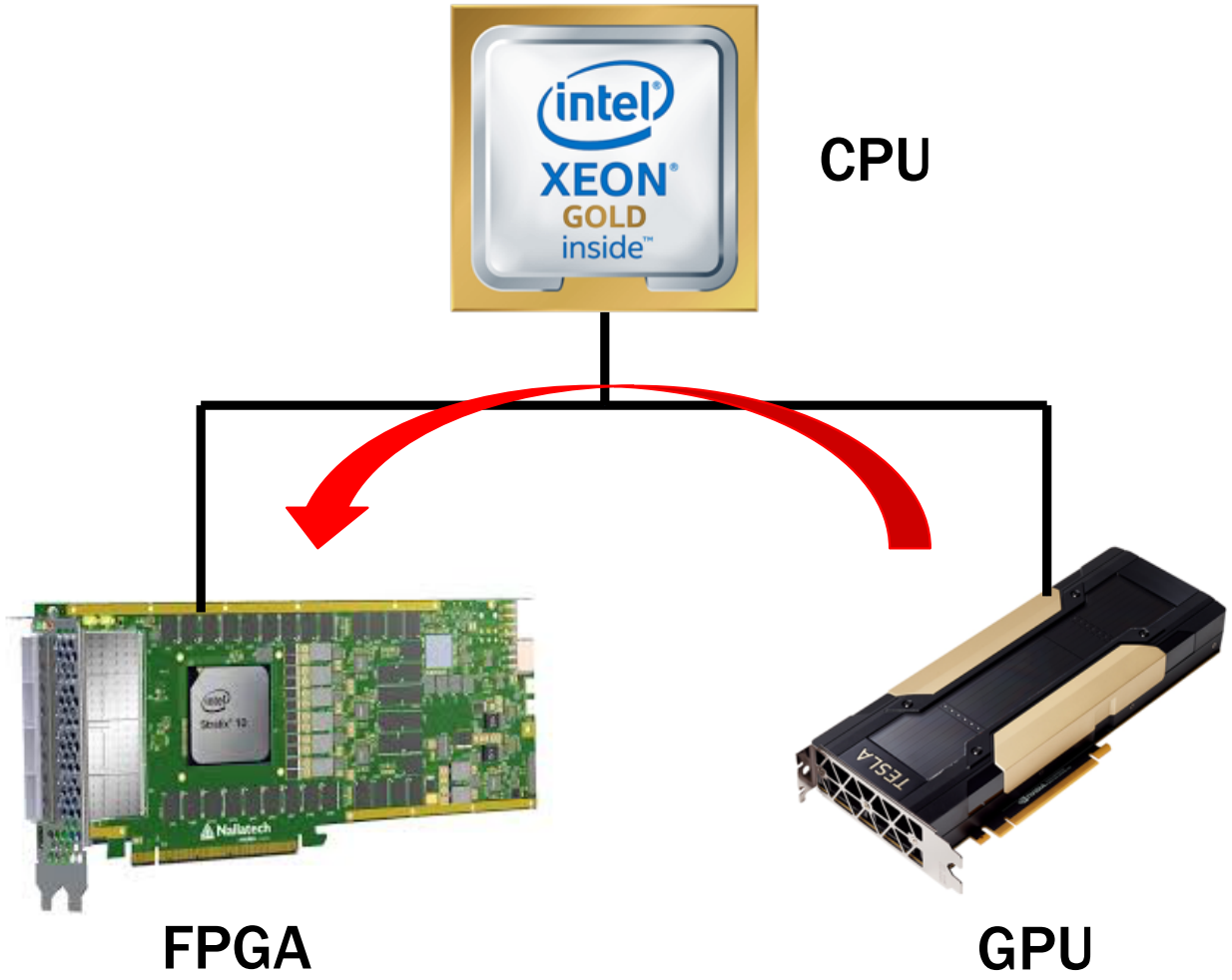
CIRCUS: Communication Integrated Reconfigurable CompUting System

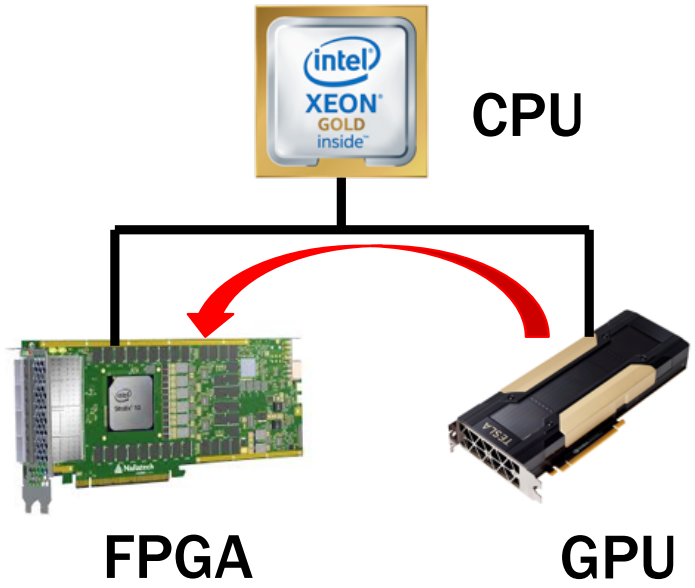
=> detail talk by Prof. Norihisa Fujita

GPU-FPGA communication (via CPU memory)



GPU-FPGA communication (DMA)

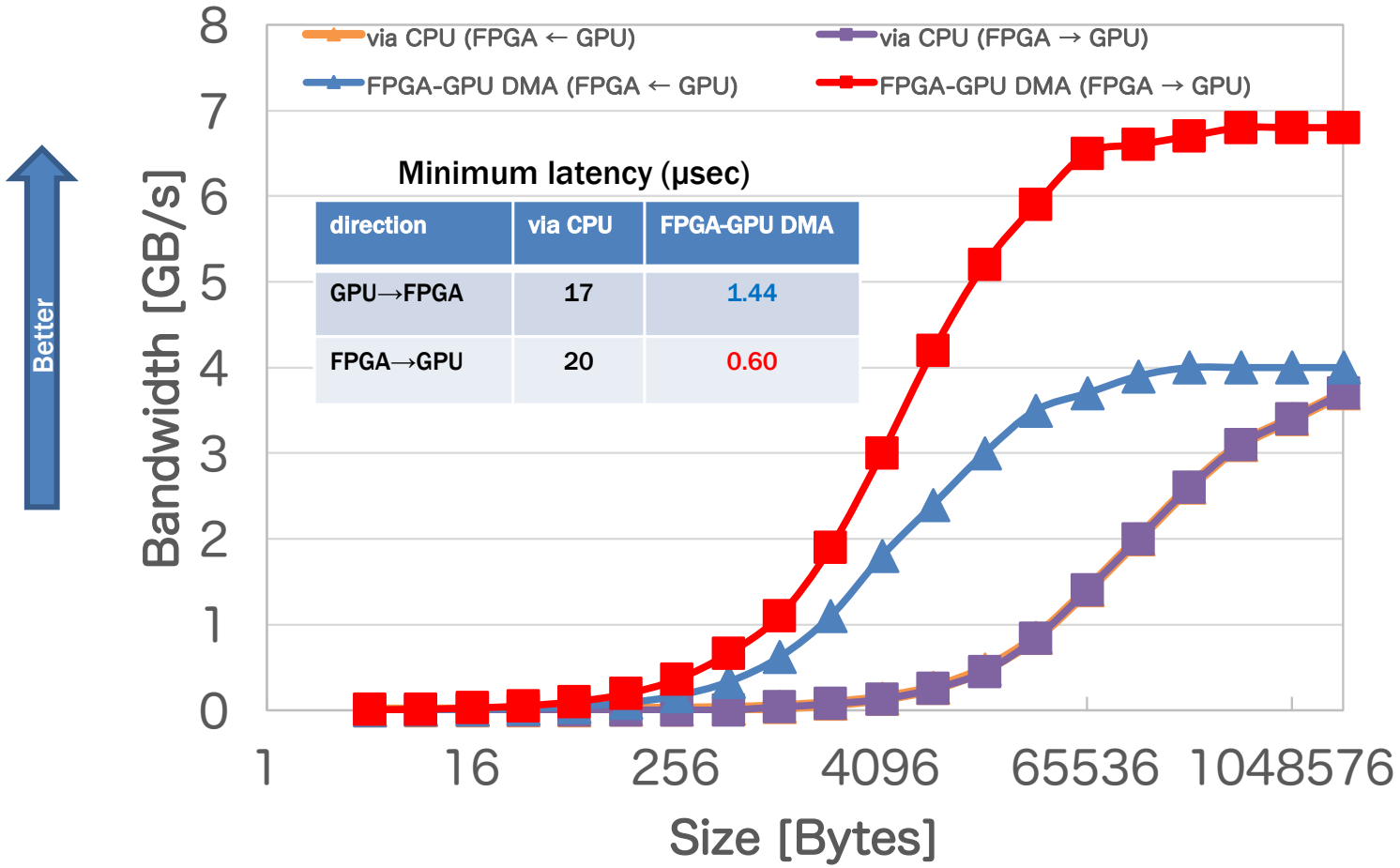





```
__kernel void fpga_dma(__global float *restrict fpga_mem,
                       const ulong gpu_memadr,
                       const uint id_and_len)
{
  cldesc_t desc;
  // DMA transfer GPU -> FPGA
  desc.src = gpu_memadr;
  desc.dst = (ulong)(&fpga_mem[0]);
  desc.id_and_len = id_and_len;
  write_channel_intel(fpga_dma,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

GPU-to-FPGA DMA kick

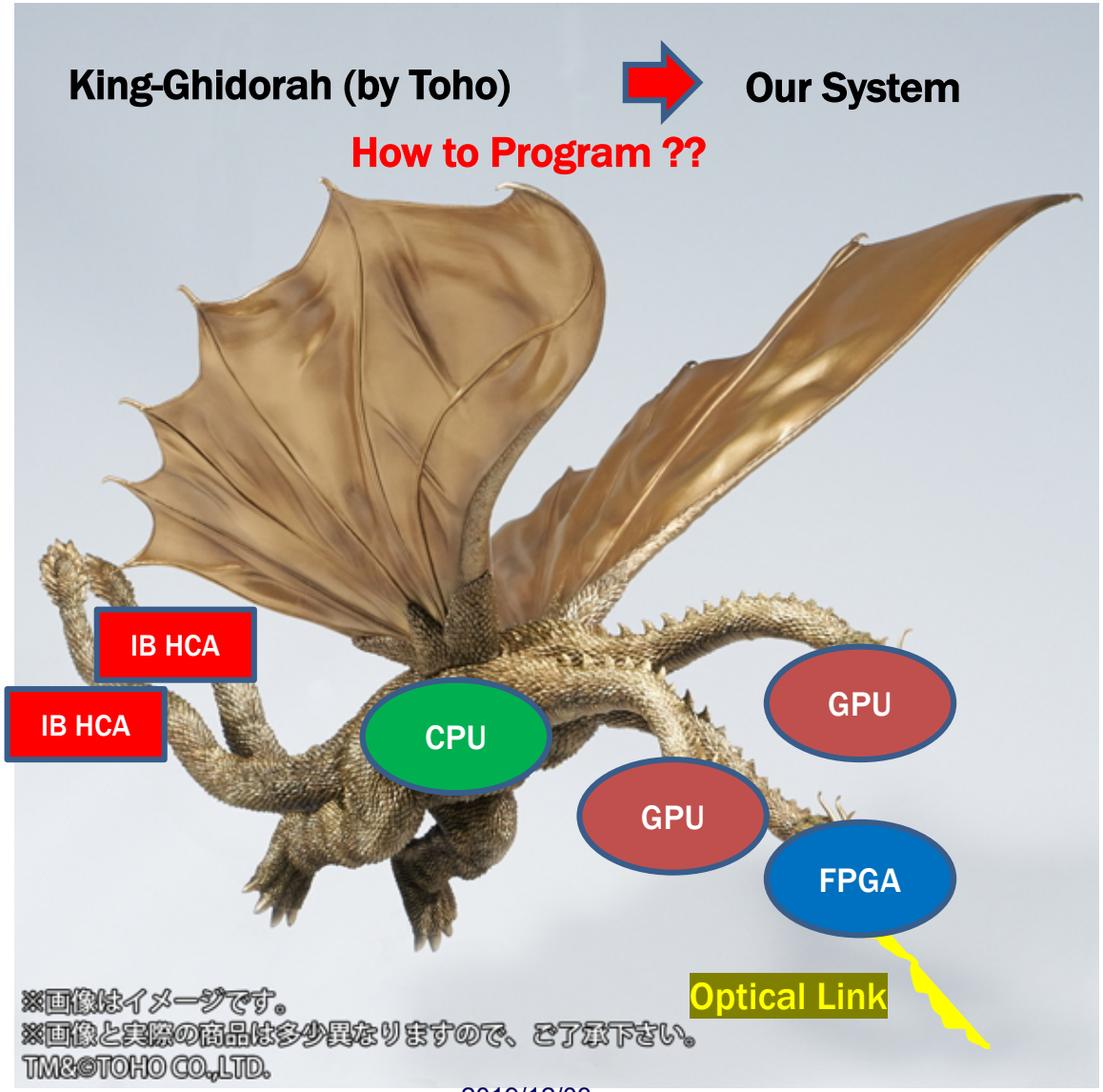
Communication Bandwidth (on Arria10 – V100)



[Reference]

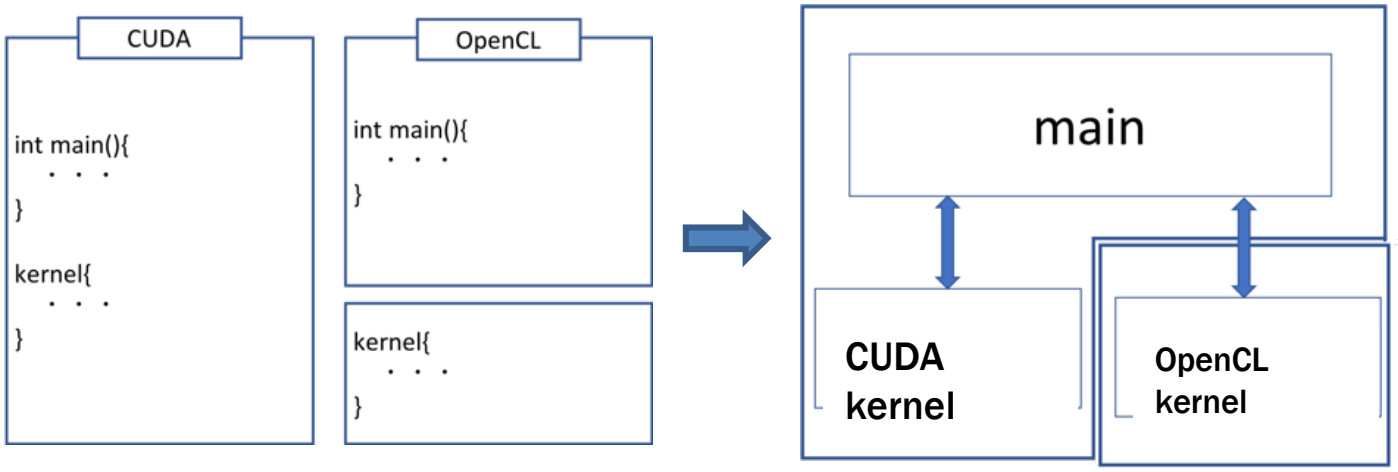
- Ryohei Kobayashi, Norihisa Fujita, Yoshiki Yamaguchi, Ayumi Nakamichi, Taisuke Boku, "GPU-FPGA Heterogeneous Computing with OpenCL-enabled Direct Memory Access", Proc. of Int. Workshop on Accelerators and Hybrid Exascale Systems (AsHES2019) in IPDPS2019 (to be published), May 20th, 2019.


King-Ghidorah (by Toho)
※画像はイメージです。
※画像と実際の商品は多少異なりますので、ご了承下さい。
TM&©TOHO CO.,LTD.


King-Ghidorah (by Toho)
Our System
How to Program ??
IB HCA
IB HCA
CPU
GPU
GPU
FPGA
Optical Link
※画像はイメージです。
※画像と実際の商品は多少異なりますので、ご了承下さい。
TM&©TOHO CO.,LTD.

CUDA (GPU) + OpenCL (FPGA)

- **Calling two device Kernels written in CUDA (for GPU) and OpenCL (for FPGA)**
  - CUDA compiler (NVIDIA/PGI) and OpenCL compiler (Intel)
    → Two ”host” program exist
    Behavior of Host Program differs on two systems, but can be combined
    → One Host Program calls different system kernels
- We found the library to be resolved for each compiler and confirmed that hey don’t conflict
  → Linking everything



OpenACC high level coding

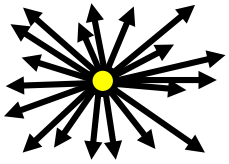
- OpenACC is available both for GPU and FPGA
  - under development in collaboration between CCS-Tsukuba and ORNL
  - GPU compilation – PGI OpenACC compiler
  - FPGA compilation – OpenARC for FPGA on OpenACC, developed at FTG ORNL
    -> collaboration between CCS and ORNL
- Solving some conflict on host code environment and run-time
  - Basic running is confirmed and testing example codes
- Issues
  - Performance enhancement on GPU and FPGA totally differs
    - GPU – horizontal (data) parallelism in SIMD manner
    - FPGA – clock level pipelining
    - Memory model difference: HBM2 vs BRAM

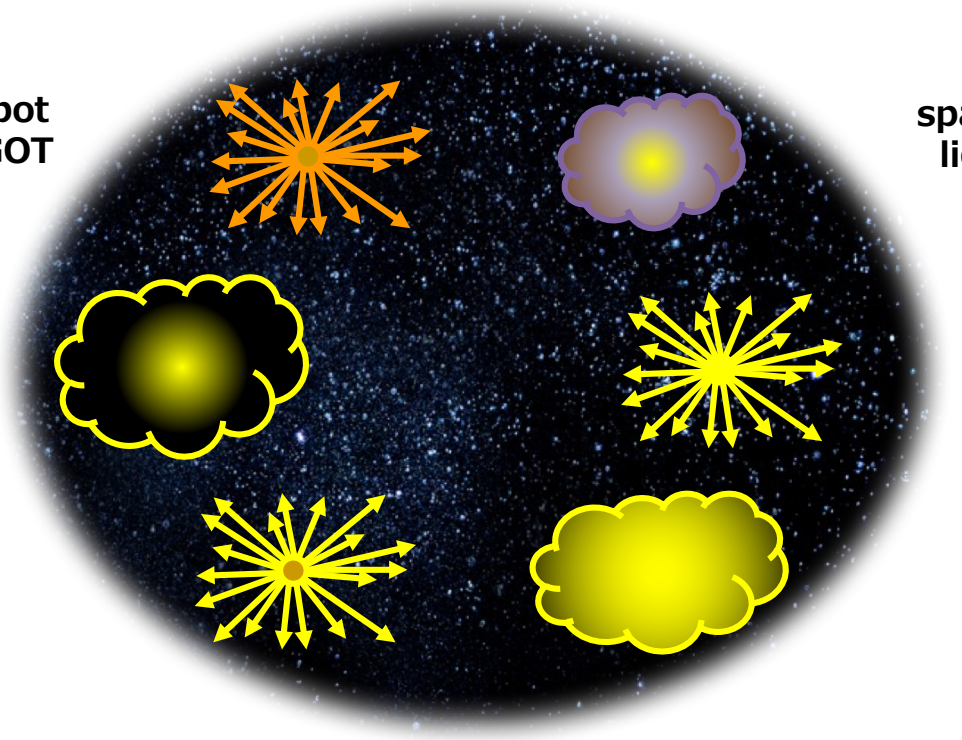
Application Example – ARGOT (collab. with M. Umemura et. al.)

- ARGOT (Accelerated Radiative transfer on Grids using Oct-Tree)
  - Simulator for early stage universe where the first stars and galaxies were born
  - 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

ARGOT code: radiation transfer simulation

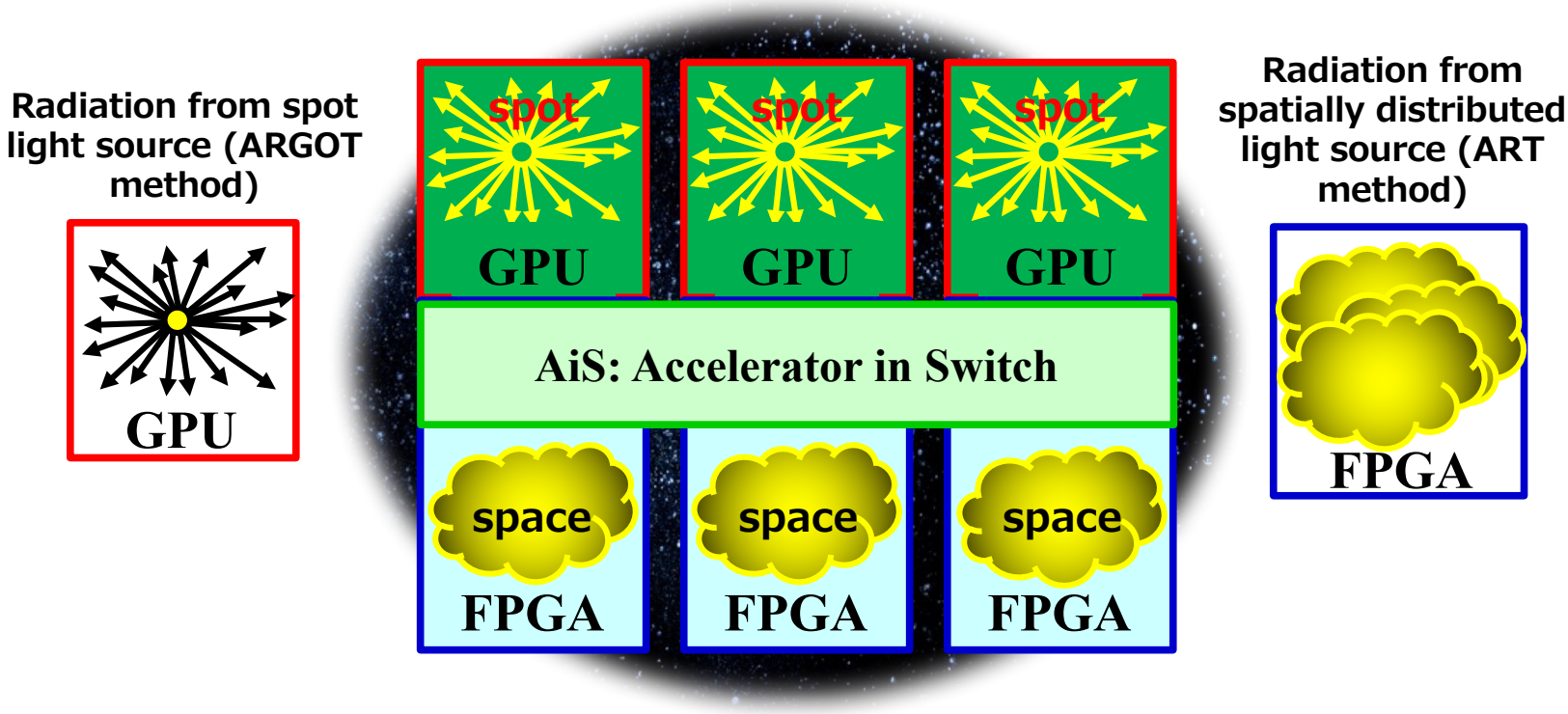
Radiation from spot light source (ARGOT method)





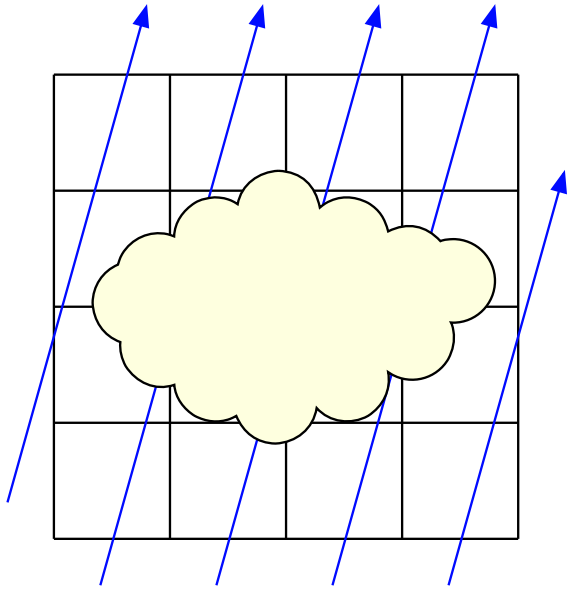
Radiation from spatially distributed light source (ART method)

ARGOT code: radiation transfer simulation



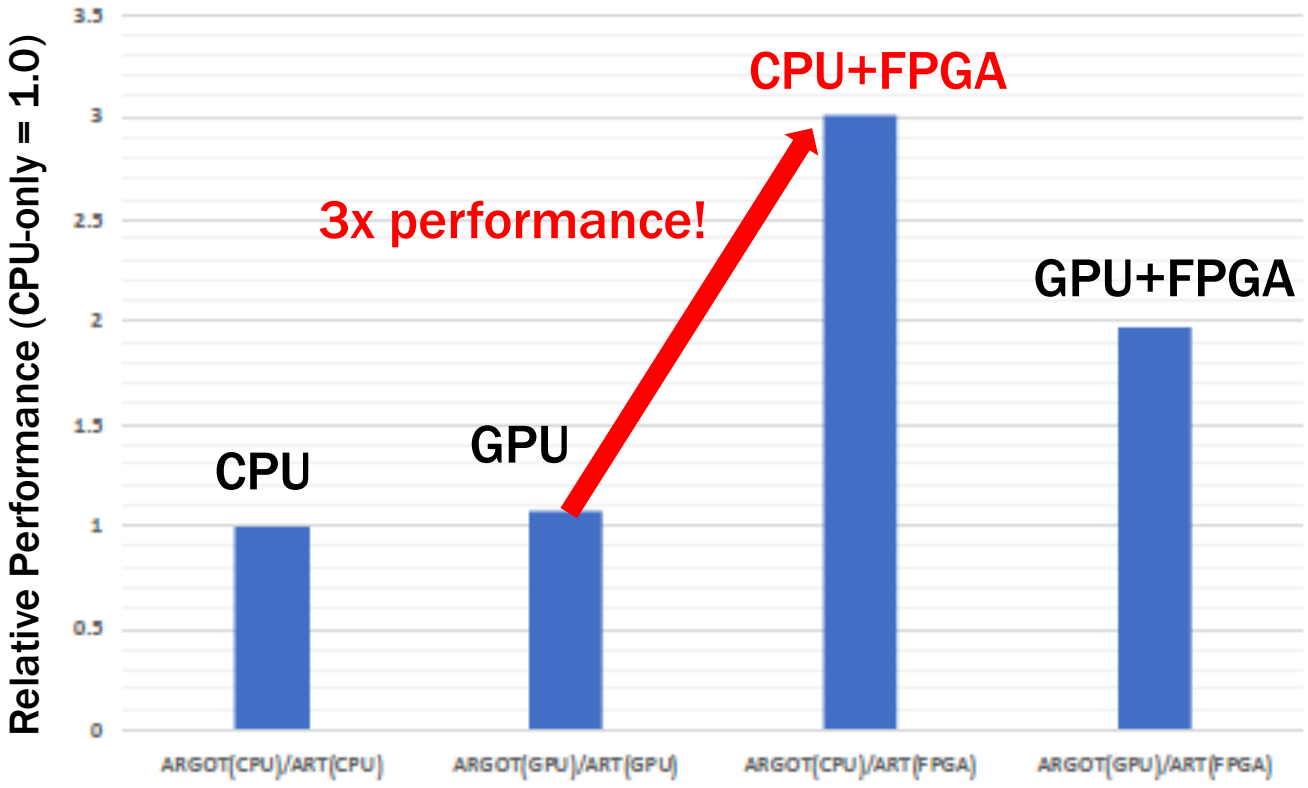
ART Method

- 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.



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

Overall performance improvement on ARGOT (14x2 cores Xeon, P100, Arria10)



- Relative performance of entire ARGOT code with ARGOT method and ART method
- GPU cannot gain the performance
- CPU (for ARGOT method) + FPGA (for ART method) achieves 3x performance with single P100 and single A10 compared with 28-core CPU
- No GPU/FPGA DMA

Other target application candidates and topics

- Quantum Bioscience
  - non-local potential calculation in RSDFT, not good at GPU
- City-LES (local climate)
  - surface simulation has weak parallelism and conditional branches
- Data Science
  - using FPGA for data search and file system control
  - Graph
- Programming
  - OpenACC-only approach for AiS programming
  - collaboration with FTG (PI: Jeff Vetter) at ORNL to apply their research compiler OpenARC
  - high level PGAS approach with OpenACC (XcalableACC) under collaboration with Programming Environment Group (PI: Mitsuhisa Sato) at R-CCS RIKEN

Summary

- CCS has continues “codesigning” from the origin of center to provide an ideal field for collaboration between application scientists and computer scientists
- Collaboration with Medical field through Computational Medical Science activities
- New machine Cygnus is 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-only computing
- Multi-physics simulation is the first stage target of Cygnus and will be expanded to variety of applications where GPU-only solution has some bottleneck
- New applications are under development