University of Tsukuba's Accelerated Computing

Taisuke Boku

Deputy Director, Center for Computational Sciences

University of Tsukuba

under collaboration with JST-CREST and CCS PACS-X Projects

Agenda

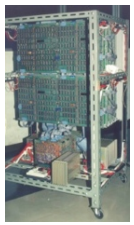
- CCS, U. Tsukuba
- TCA & Accelerator in Switch
- Challenge on FPGA for HPC
- PACS-X Project
- Example in Astrophysics
- Summary


History of PACS (PAX) Systems in U. Tsukuba

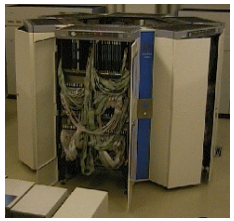
- 1977: PAX research started (by Hoshio & Kawai)
- 1978: 1st PAX (PAX-9) built
- 1996: CP-PACS ranked #1 in TOP500

1978
#1 PAX-9

1980
#2 PAXS-32



1989
#5 QCDPAX



1996
#6 CP-PACS
World fastest

2006
#7 bandwidth-aware
PACS-CS

2012~2013
GPU cluster
HA-PACS

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

- *co-design by computer scientists and computational scientists for "performance aware system"*
- Application-driven development
- Continuous R&D

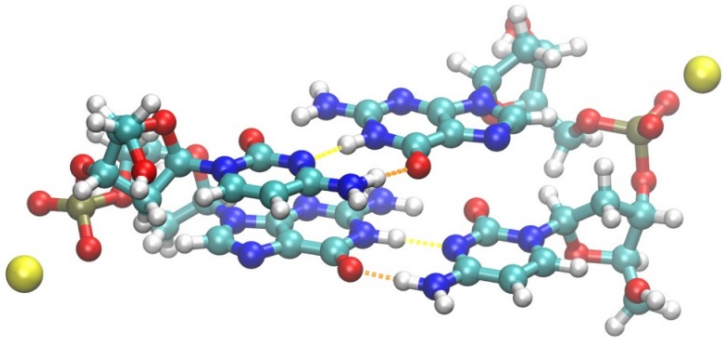
HA-PACS/TCA

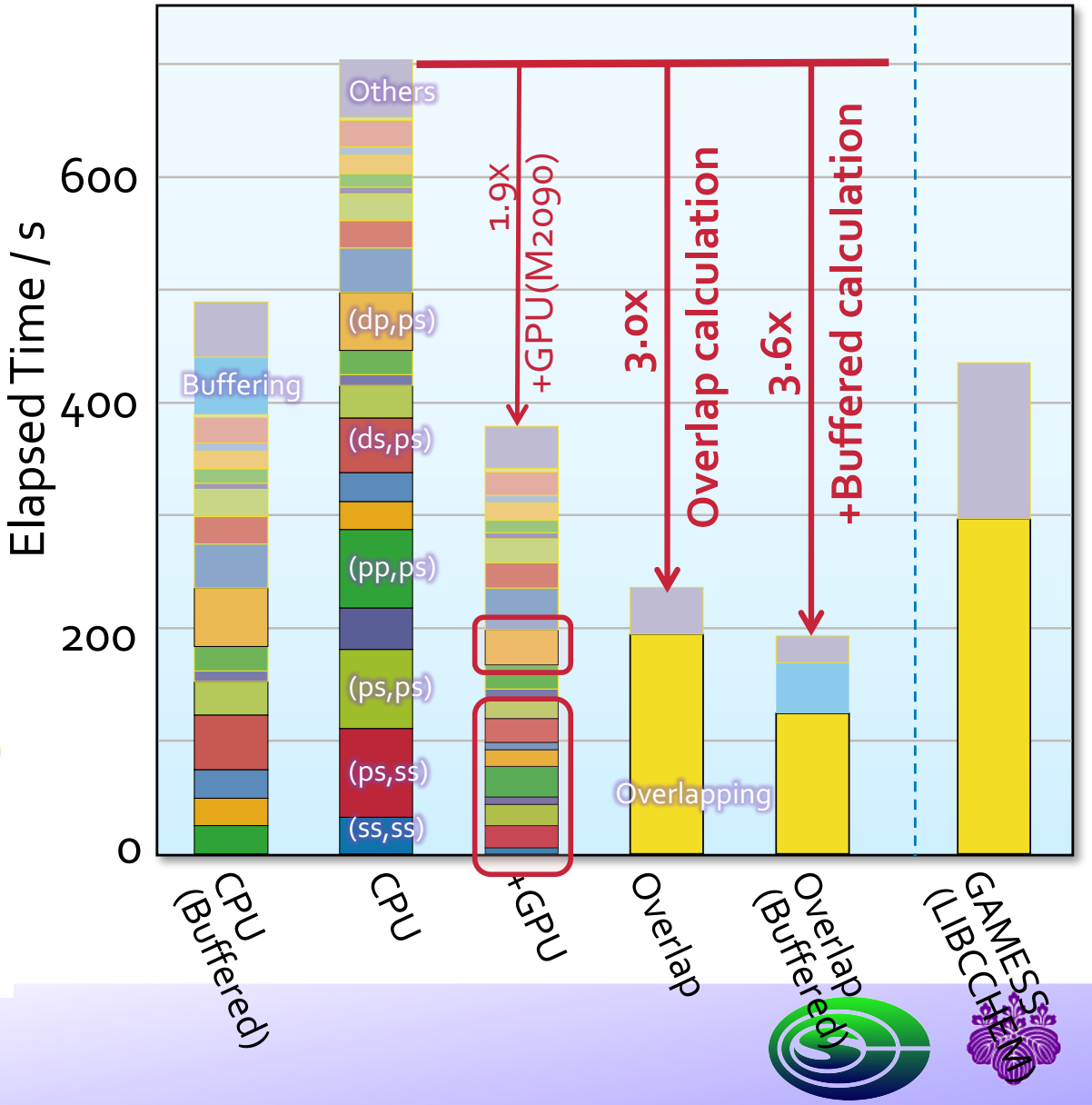
- Practical test-bed for TCA architecture with advanced GPU cluster computation node with PEACH2 board and its network
- HA-PACS (Highly Accelerated Parallel Advanced System for Computational Sciences) project
  - Three year project for 2011-2013
  - Base cluster with commodity GPU cluster technology
  - TCA part for advanced experiment on TCA and PEACH2
- Base cluster part with 268 nodes
  - Intel SandyBridge CPU x 2 + NVIDIA M2090 (Fermi) x 4
  - dual rail InfiniBand QDR
- TCA part with 64 nodes
  - Intel IvyBridge CPU x 2 + NVIDIA K20X (Kepler) x 4
  - PEACH2 board is installed to all nodes and connected by its network (additionally to original InfiniBand QDR x 2)

HA-PACS/TCA computation node inside

HA-PACS result: Hartree Fock matrix calc.

- Model DNA (CG)2
  - HF/6-31G(d)
  - 126 atom, 1,208 AO
  - 14 SCF iterations
- HA-PACS 1node
  - 16 CPU cores
    - Intel SandyBridge-E5, 2.6GHz
  - 4 GPU(NVIDIA M2090)
- Software
  - OpenFMO
  - GAMESS
    - Version: 1 MAY 2013 (R1)
    - GPU support (LIBCCHEM)





TCA & Accelerator in Switch

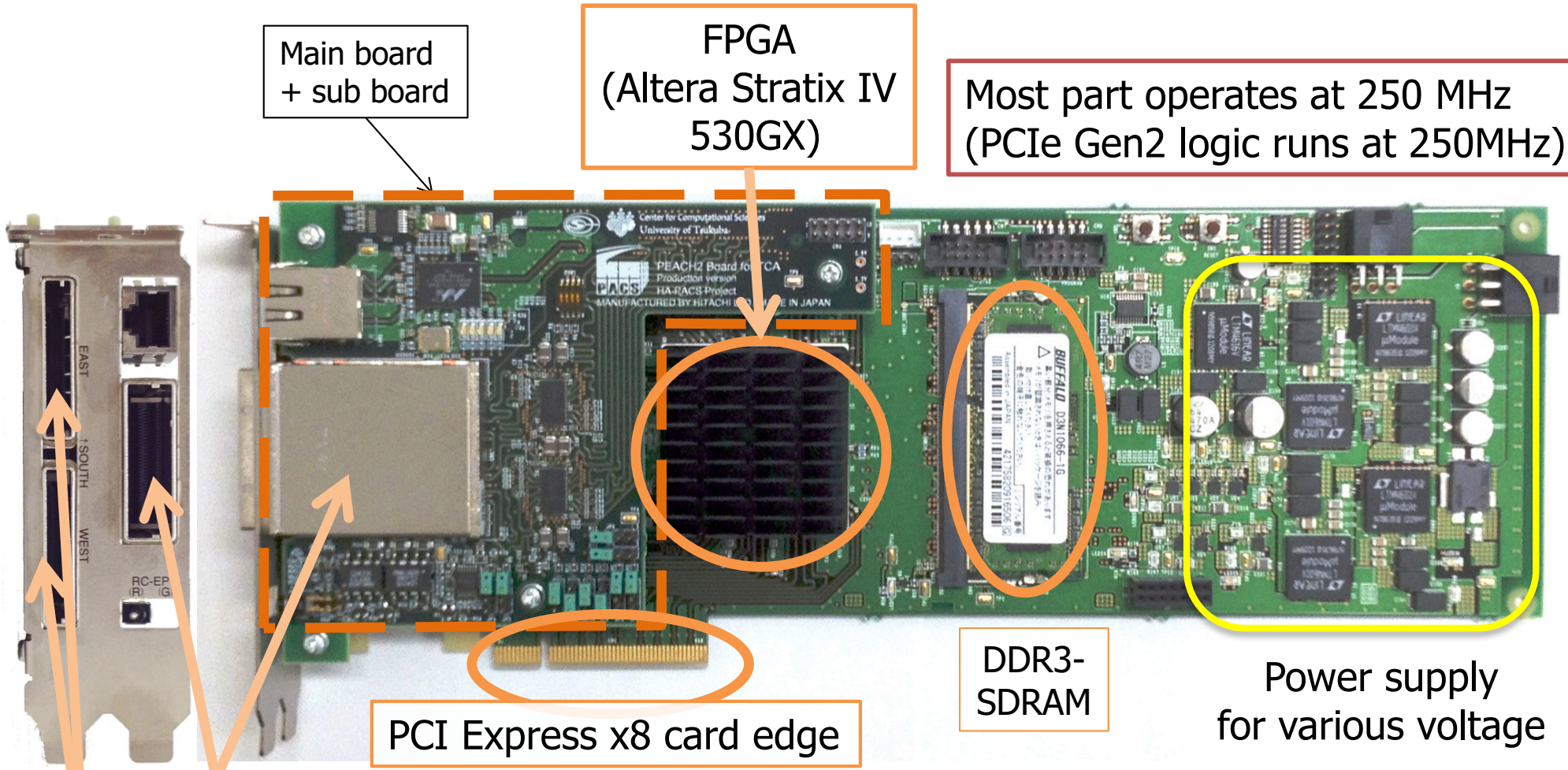
AC-CREST: Acceleration & Communication

- JST-CREST research projects
  - research area "Development of System Software Technologies for post-Peta Scale High Performance Computing" (RS: Dr. M. Sato, RIKEN)
  - Research theme: "Research and Development on Unified Environment of Accelerated Computing and Interconnection for Post-Petascale Era" (PI: T. Boku, U. Tsukuba), Oct. 2012 - Mar. 2018, US$3.4M+ (total)
- Topics
  - How to make a low latency & high bandwidth of inter-accelerator (GPU) communication system, over nodes for efficient parallel processing
  - How to port/code the applications relying on accelerators with complicated algorithm

TCA (Tightly Coupled Accelerators)

direct communication channel among GPUs with minimum load by CPU

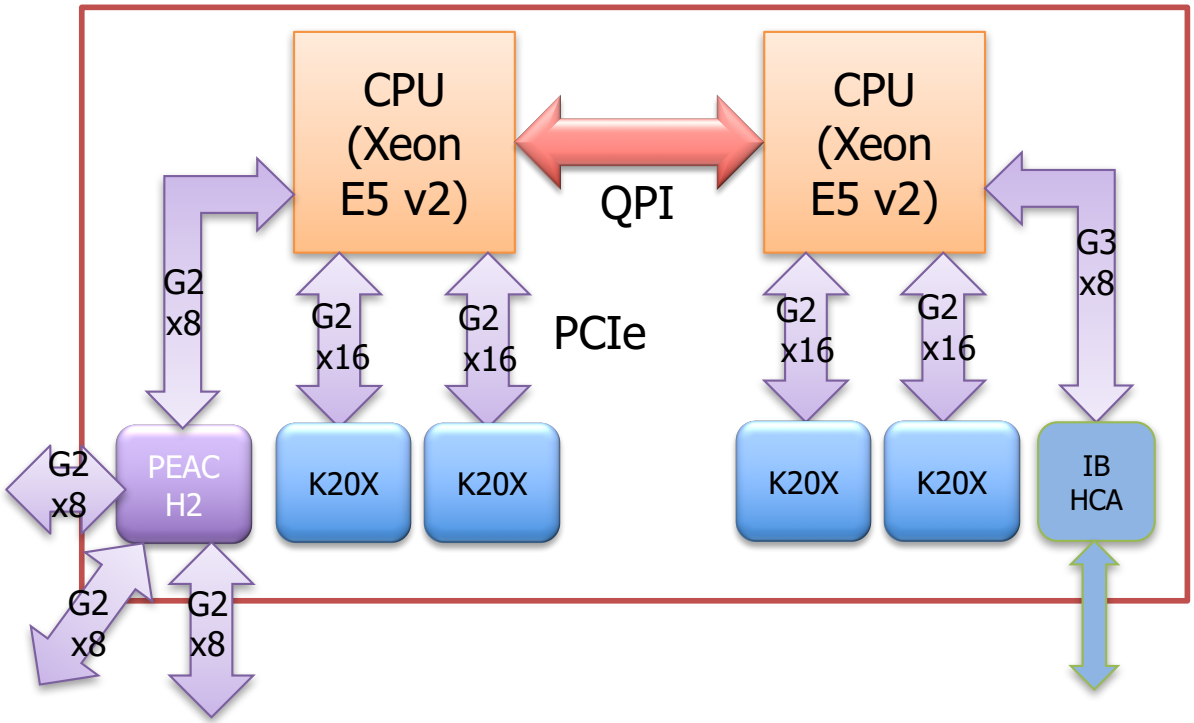
PEACH2 board – prototype implementation of TCA



HA-PACS/TCA test-bed 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.

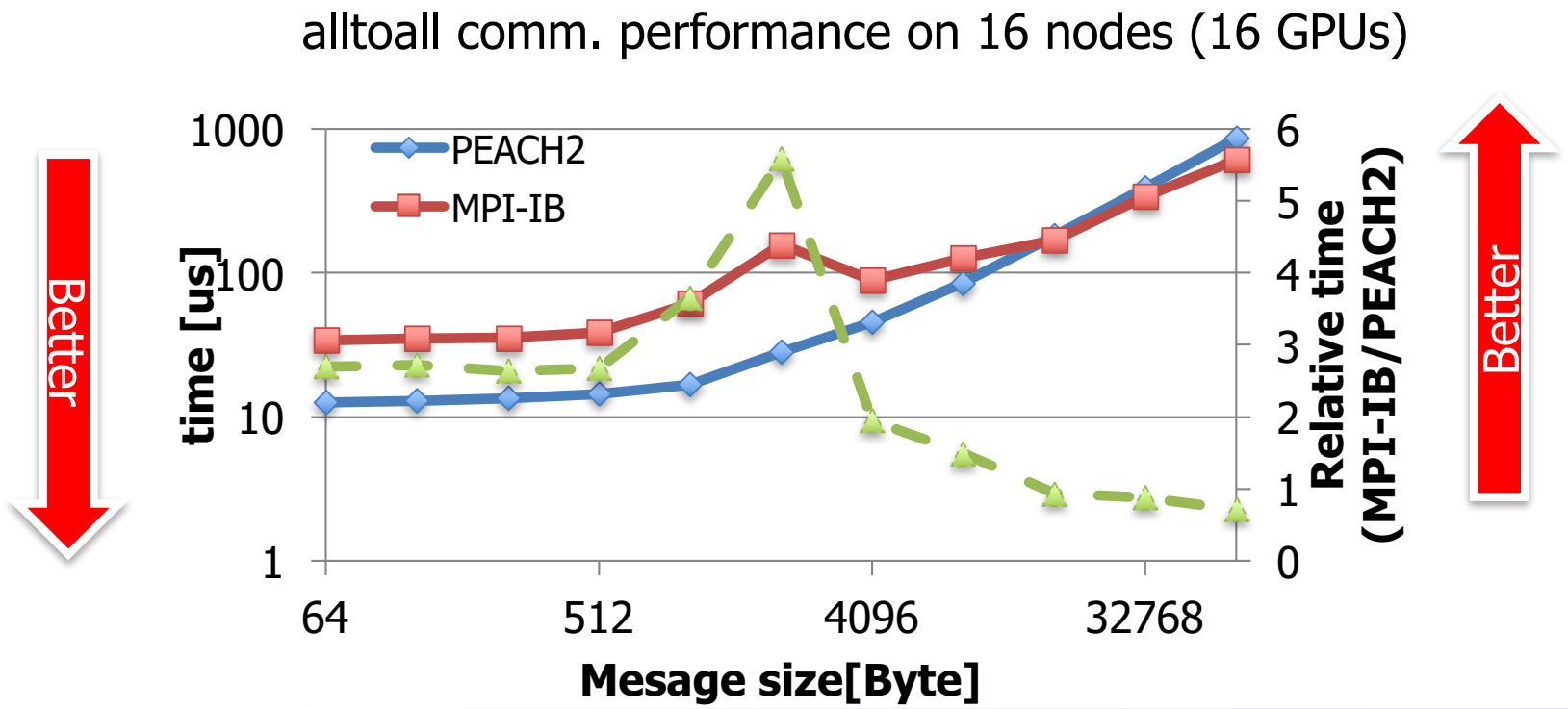


HA-PACS Base Cluster + TCA (TCA part starts operation on Nov. 1st 2013)

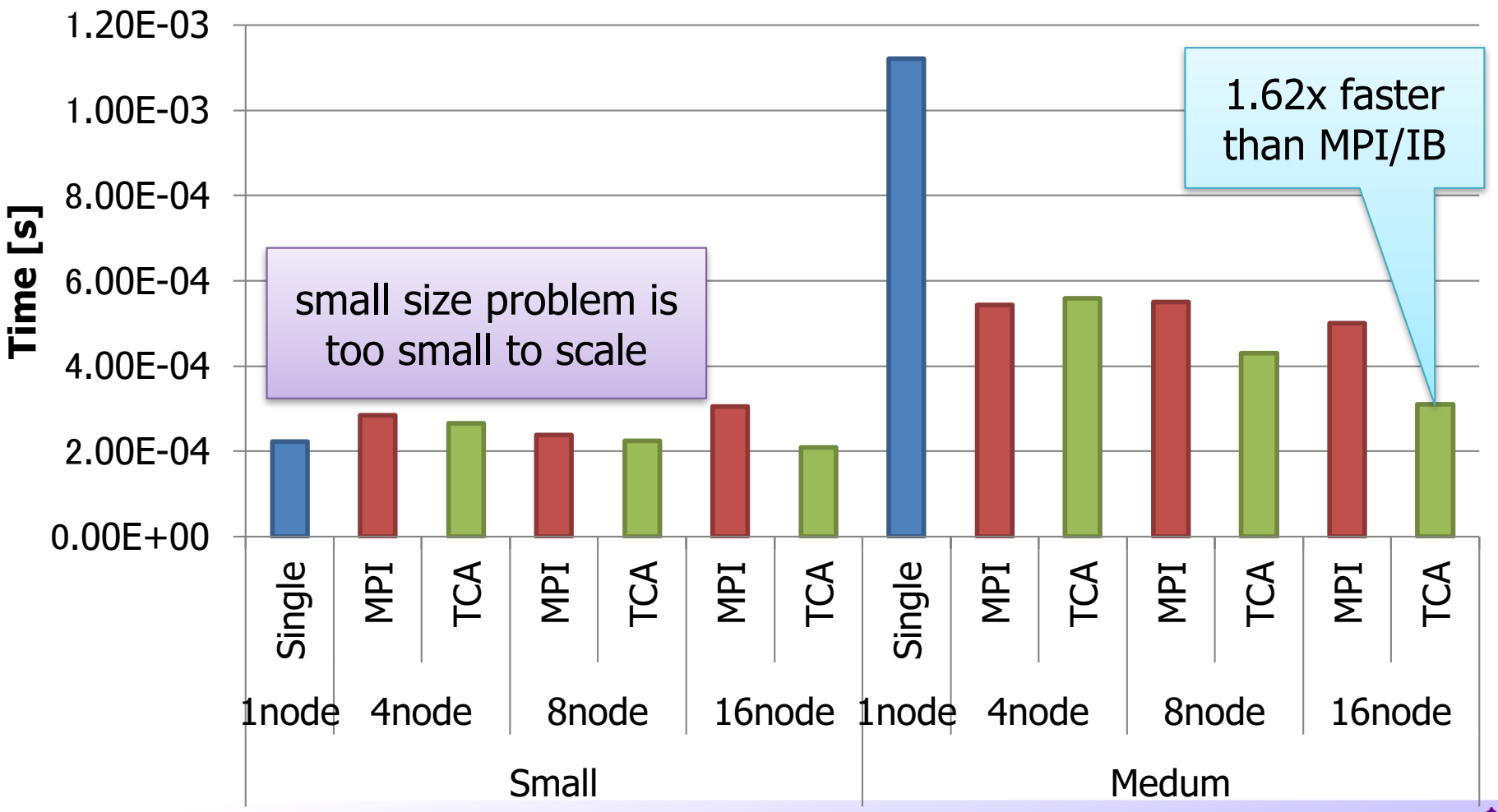
- HA-PACS Base Cluster = 2.99 TFlops x 268 node (GPU) = 802 TFlops
- HA-PACS/TCA = 5.69 TFlops x 64 node (with GPU+FPGA) = 364 TFlops
- TOTAL: 1.166 PFlops
- TCA part (individually) ranked as #3 in Green500, Nov. 2013

FFTE Benchmark on TCA (PEACH2)

- FFTE (6 step FFT) by D. Takahashi, CCS
- Using TCA communication by DMA chaining for alltoall on inter-node and intra-node GPU-GPU data copy

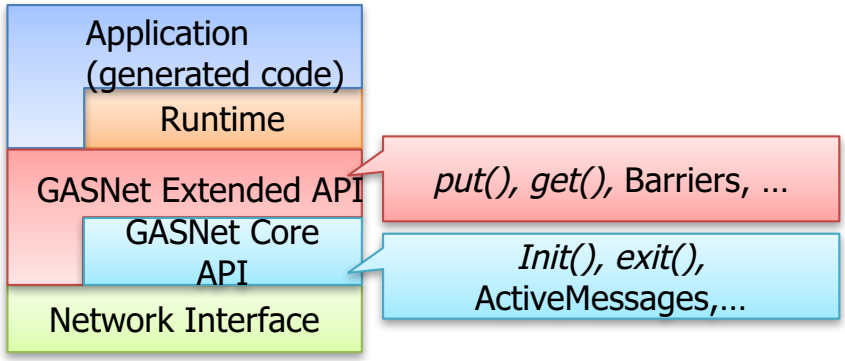


Performance of FFTE (Small=2^{14}, Medium=2^{16})

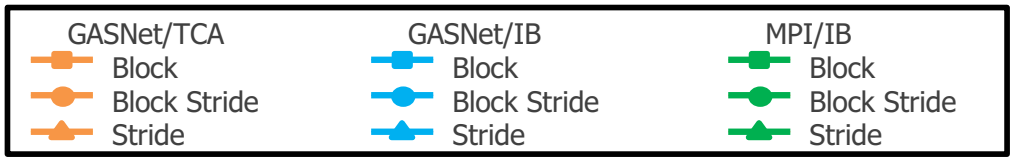


GASNet/TCA

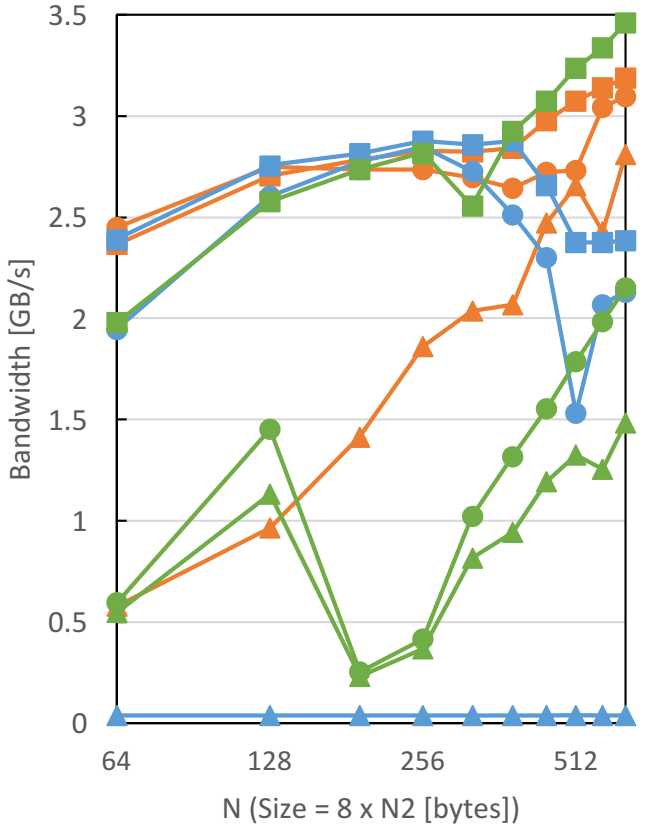
- collaboration with LBNL: new GASNet/GPU is implemented on TCA
- special tuning for Block-Stride communication for extended API of GASNet/GPU
- many optimization such as descriptor caching



Implementation layer of GASNet/GPU/TCA



- Block-stride communication is supported by hardware on PEACH2 and achieves good performance on real application



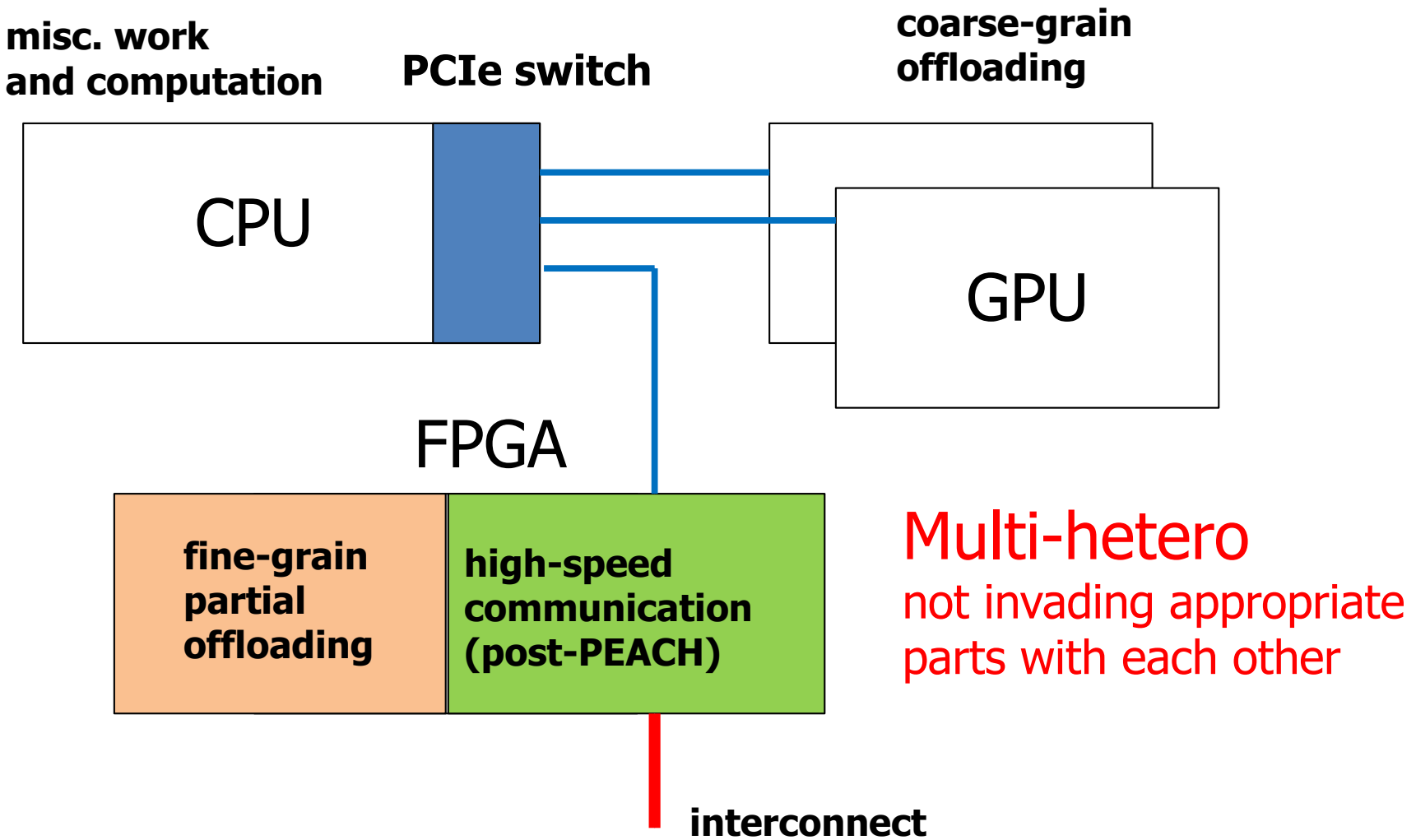
Halo exchange on 3-D stencil

PEACH2 is based on FPGA

- FPGA for parallel platform for HPC
  - in general and regular computation, GPU is better
  - for something "weird/special" type of computation
  - (relatively) non bandwidth-aware computation
- PEACH solution on FPGA provides communication and computation on a chip
  - PEACH2/PEACH3 consumes less than half of logic elements on FPGA
  - "partial offloading" of computation in parallel processing can be implemented on rest of FPGA

➟ Accelerator in Switch (Network)


Schematic of Accelerator in Switch



Multi-hetero
not invading appropriate parts with each other

Example of Accelerator in Switch

- Astrophysics
  - Gravity calculation in domain decomposition
  - Tree search is efficient
  - LET (Locally Essential Tree) is introduced to reduce the search space in tree structure
    ➟ too complicated to handle in GPU
  - CPU is too slow
    ➟ implementing the function on FPGA and combining with PEACH3 communication part


LET (Locally Essential Tree)

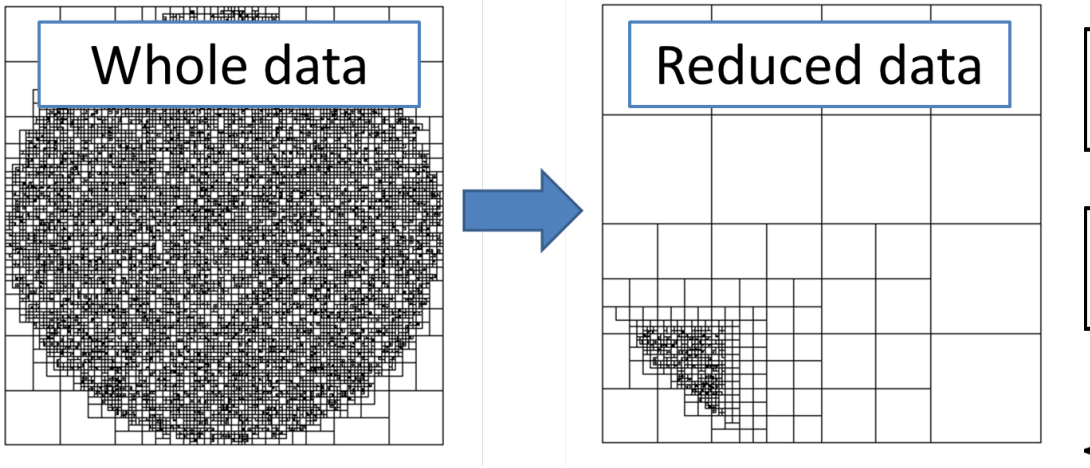
mask[id] array

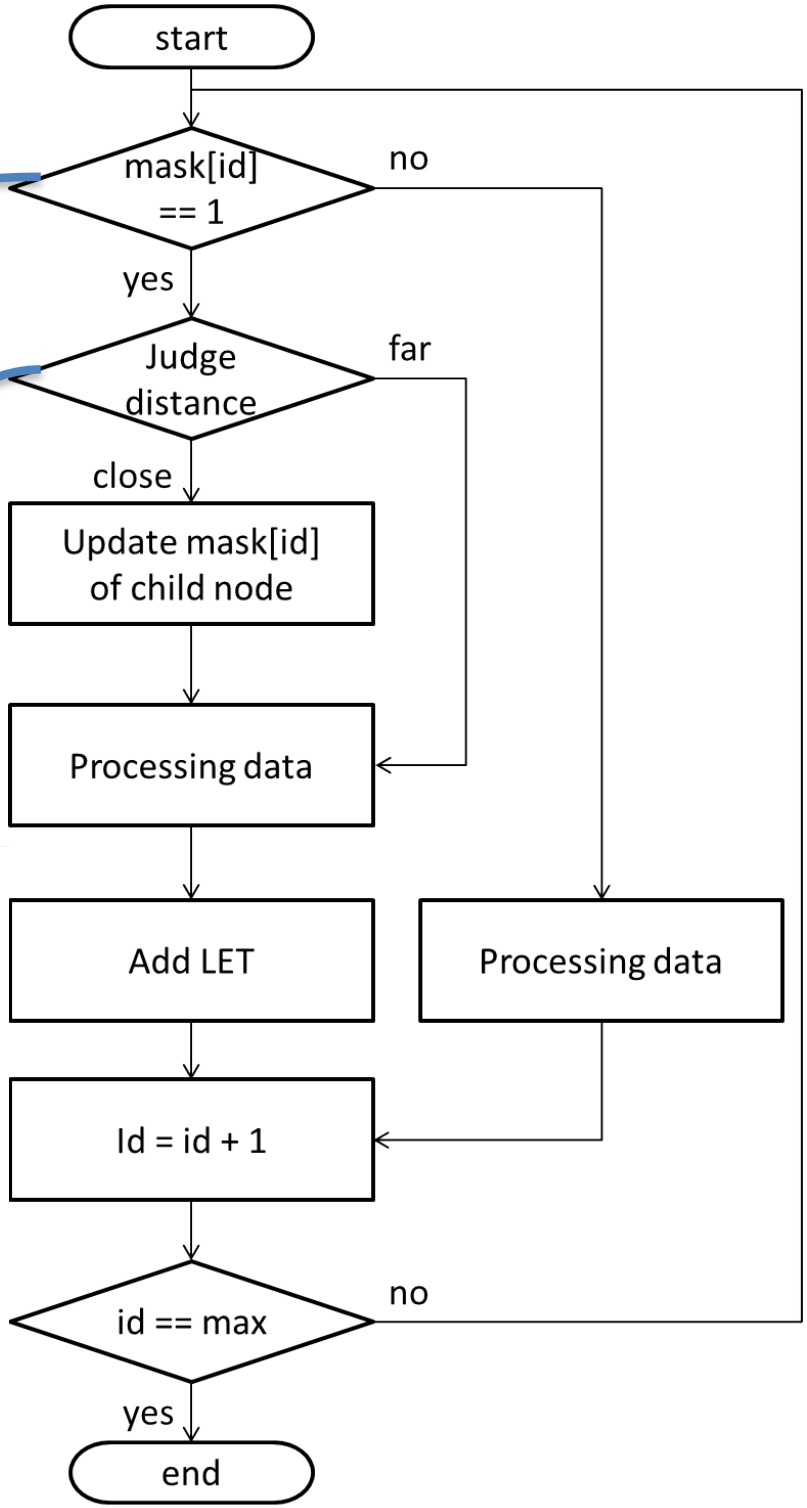
mask[id] == 0 skip

mask[id] == 1 add to LET

distance judgment

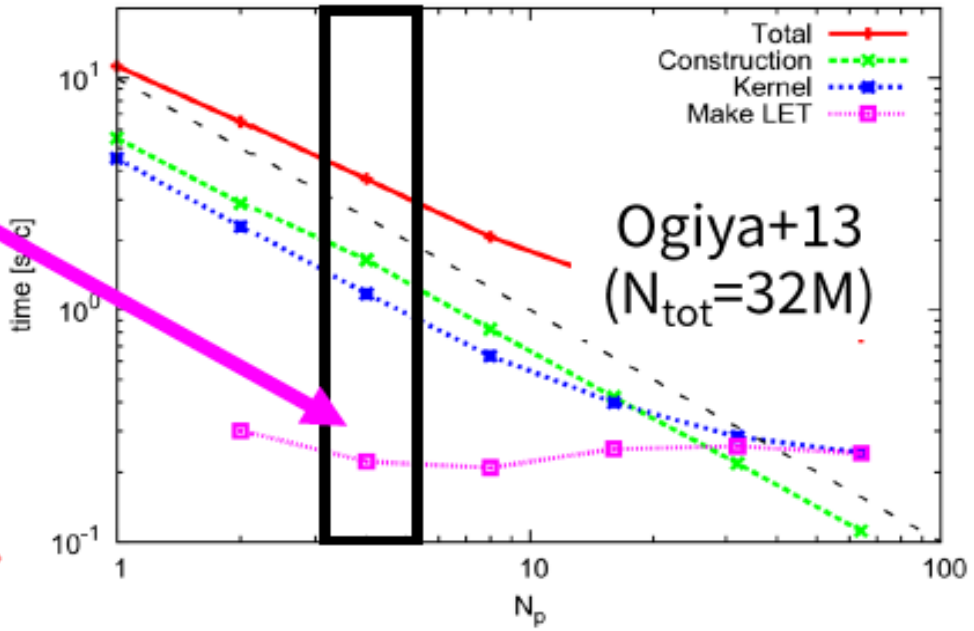
partial regional data in receiver side and each cell in sender side

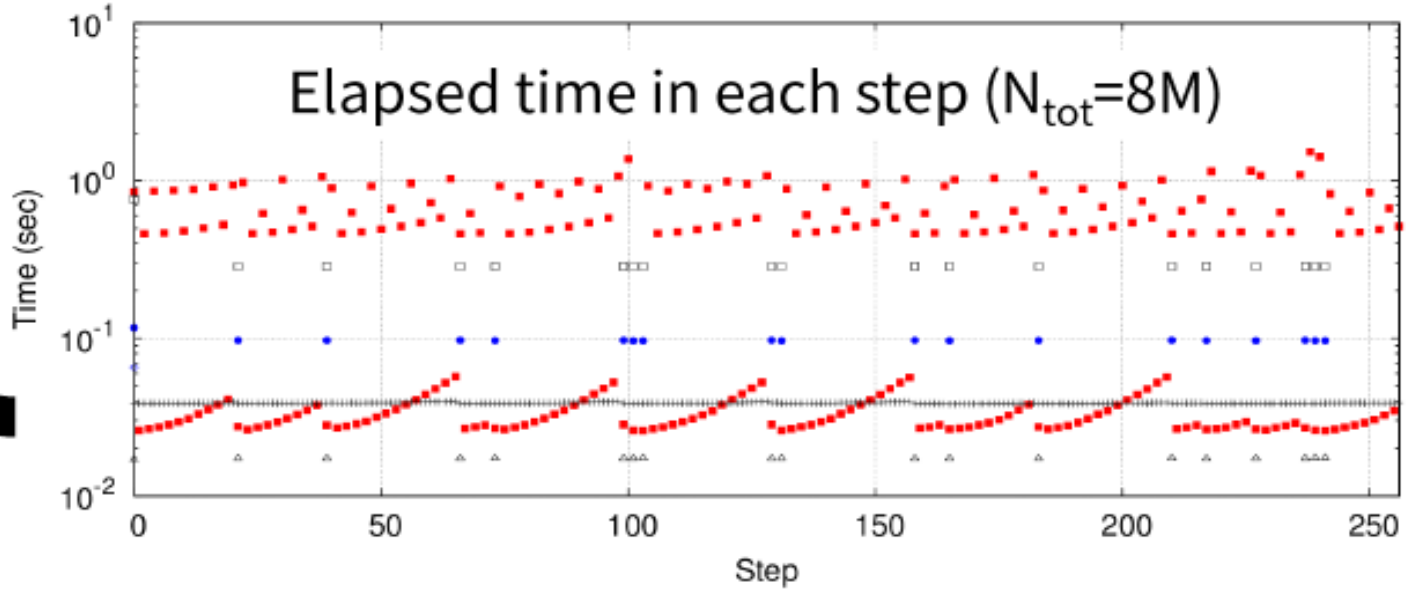




Cost of using LET

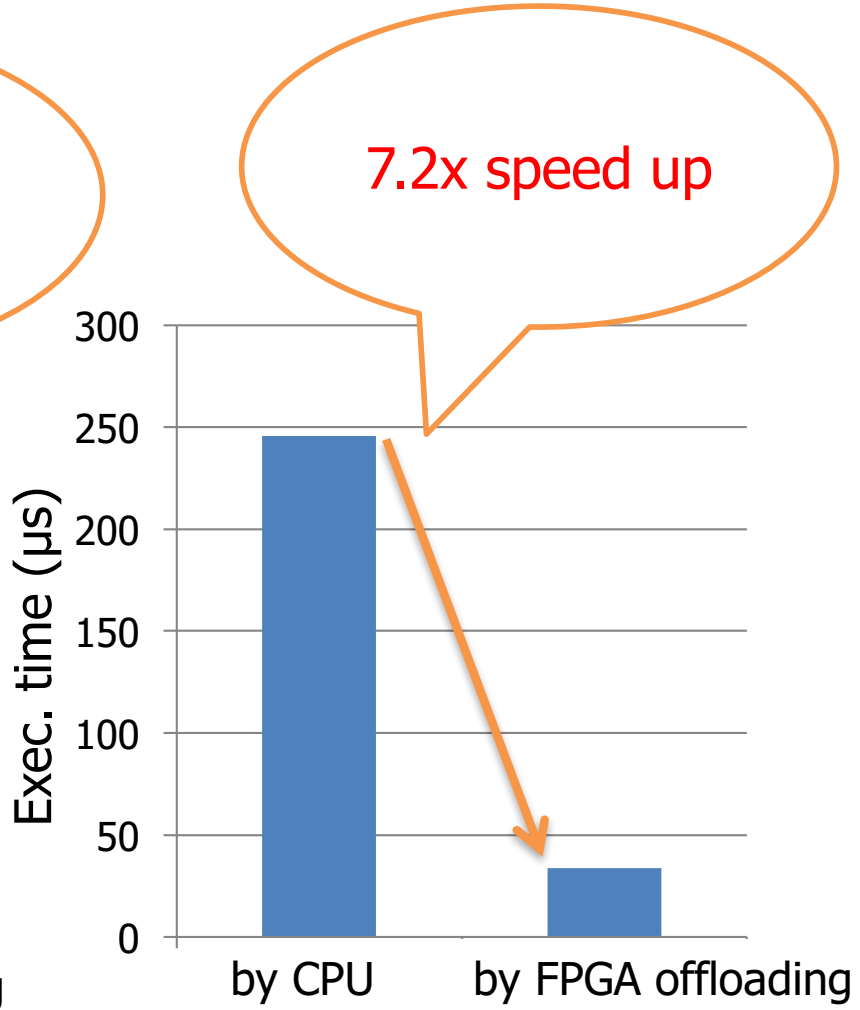
- ～0.2 sec. (Ogiya+13)
- Must be smaller than ～0.04 sec. to scale.
  - At least ～5 times acceleration is required.





Preliminary Evaluation

Exec. time for making LET



from LET making to GPU data transfer

Challenge on FPGA for HPC

FPGA in HPC

- Goodness of recent FPGA for HPC
  - True codesigning with applications (essential)
  - Programmability improvement: OpenCL, other high level languages
  - High performance external I/O (not to CPU): 40Gb~100Gb
  - Precision Control is possible
  - New semiconductor technology applied (traditionally)
  - Relatively low power
- Problems
  - Programmability: OpenCL is not enough, not efficient
  - Absolute 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)

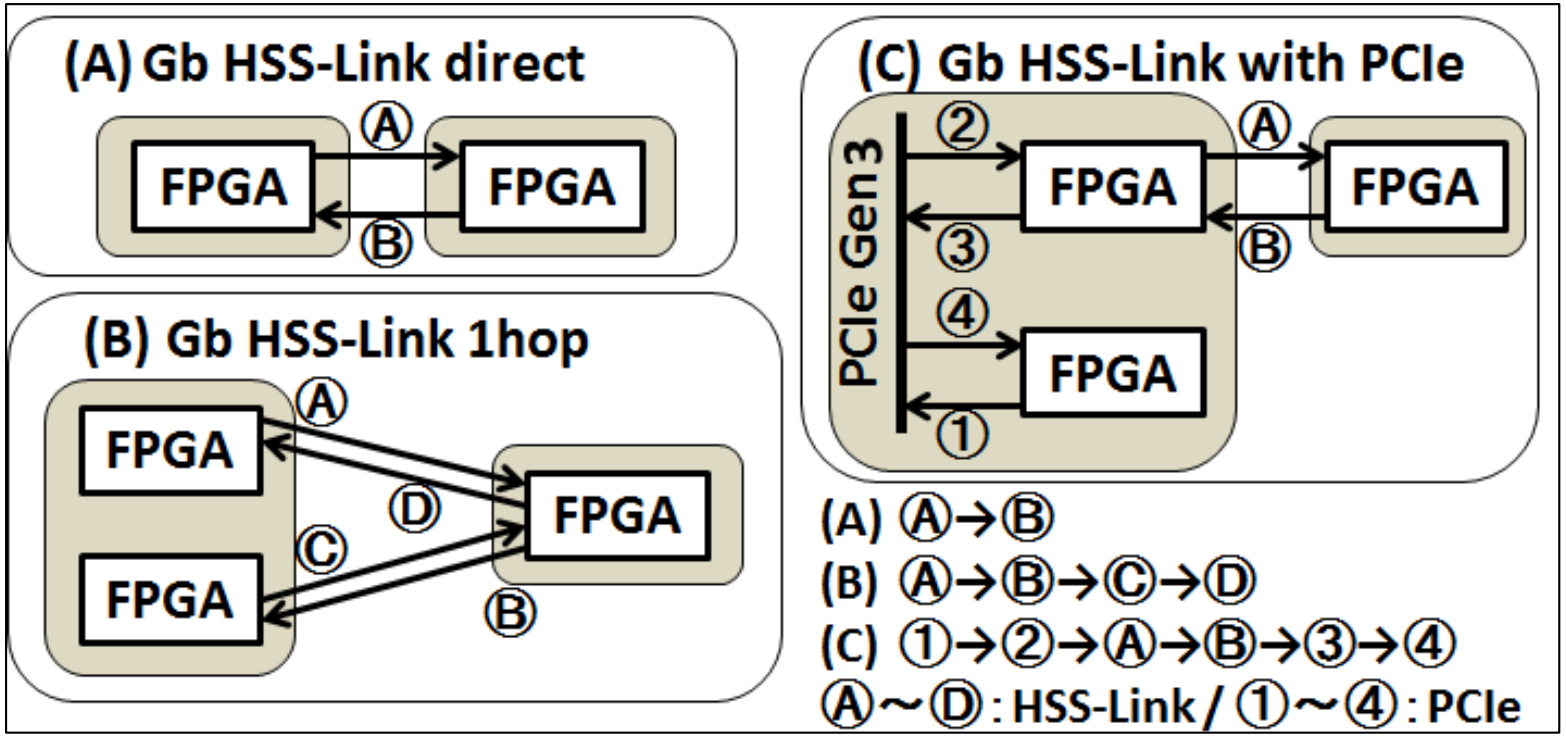

Complexity vs Flexibility & Efficiency

- Our goal
  - Why not using both GPU and FPGA ?
    - Complicated program/algorithm such as in multi-physics
  - It's too much complicated
    - system hardware
    - connection among devices
    - programming
    - total management (higher level structure and programming)
  - Easy use for end users (application researchers)
    - global scope of programming system
    - Make "library" for FPGA
    - OpenCL at most
  - Trading off between complexity and efficiency


Challenge (1): external communication

- PEACH2/PEACH3 I/O bottleneck
  - depending on PCIe everywhere, to connect CPU and GPU, and also for external link
  - PCIe is a bottleneck on today's advanced interconnect
- High performance interconnection between FPGA
  - Optical interconnect interface is ready
  - up to 100Gb speed
  - provided as IP for users
- FPGA-FPGA communication without intra-node communication bottleneck
  - on-the-fly computation & communication


100GbE inter-node communication experiment



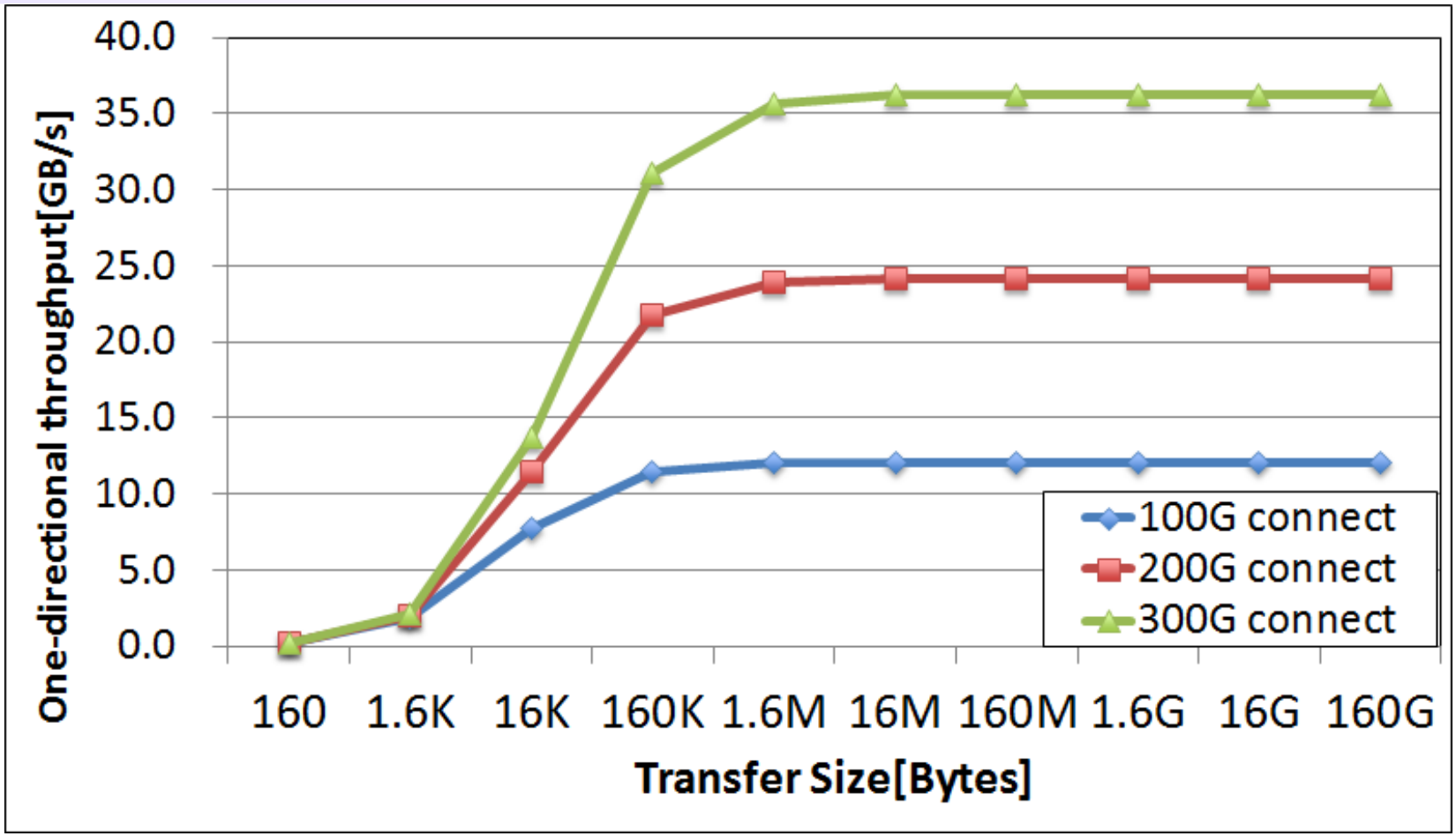
Xilinx XC7VX1140T(Virtex7)

Vivado 2016.1

Virtex-7 FPGA Gen3
Integrated Block for PCI Express v4.1

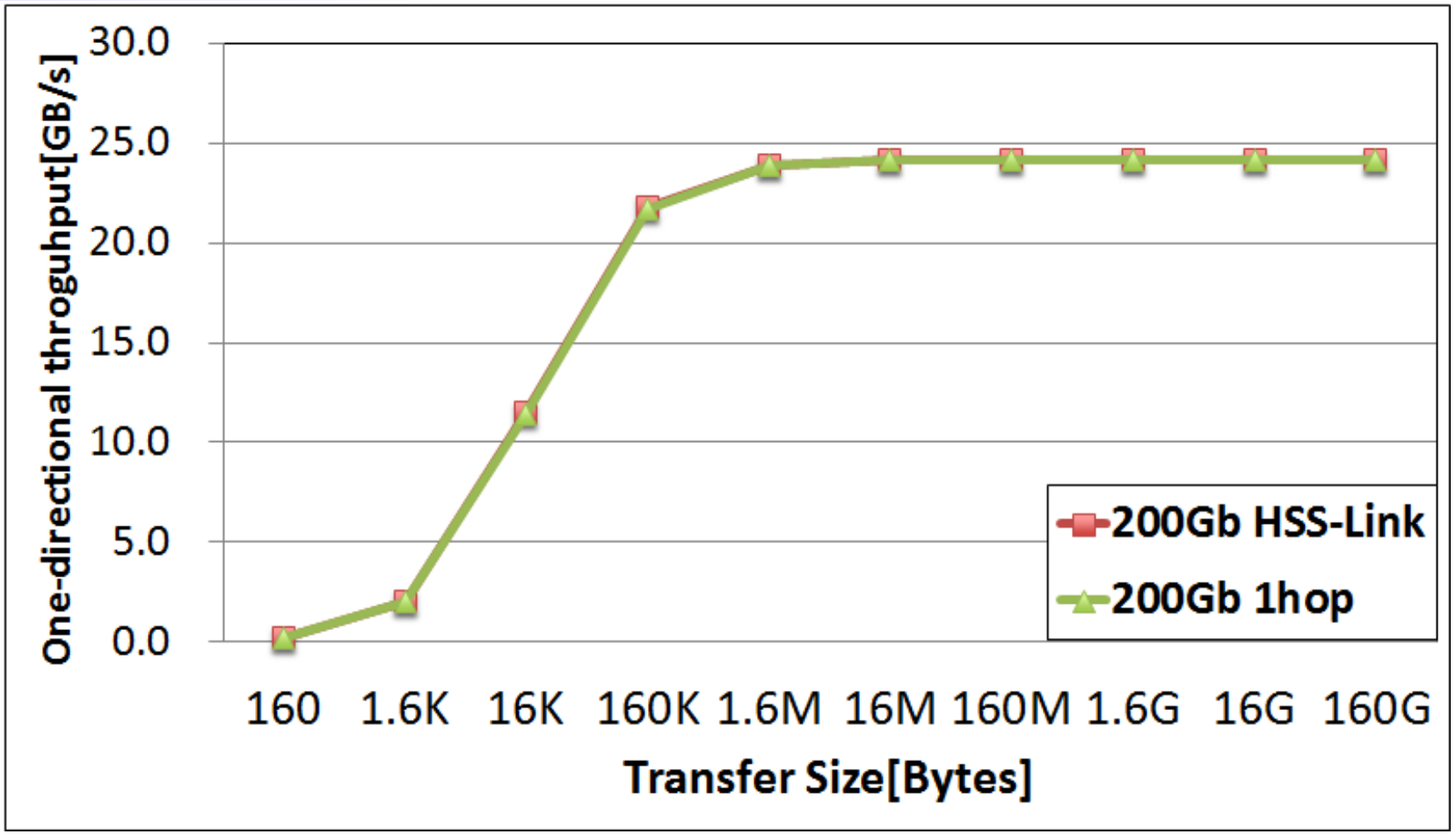
(Aurora 64B/66B v11.1)

Case-A: peer-to-peer



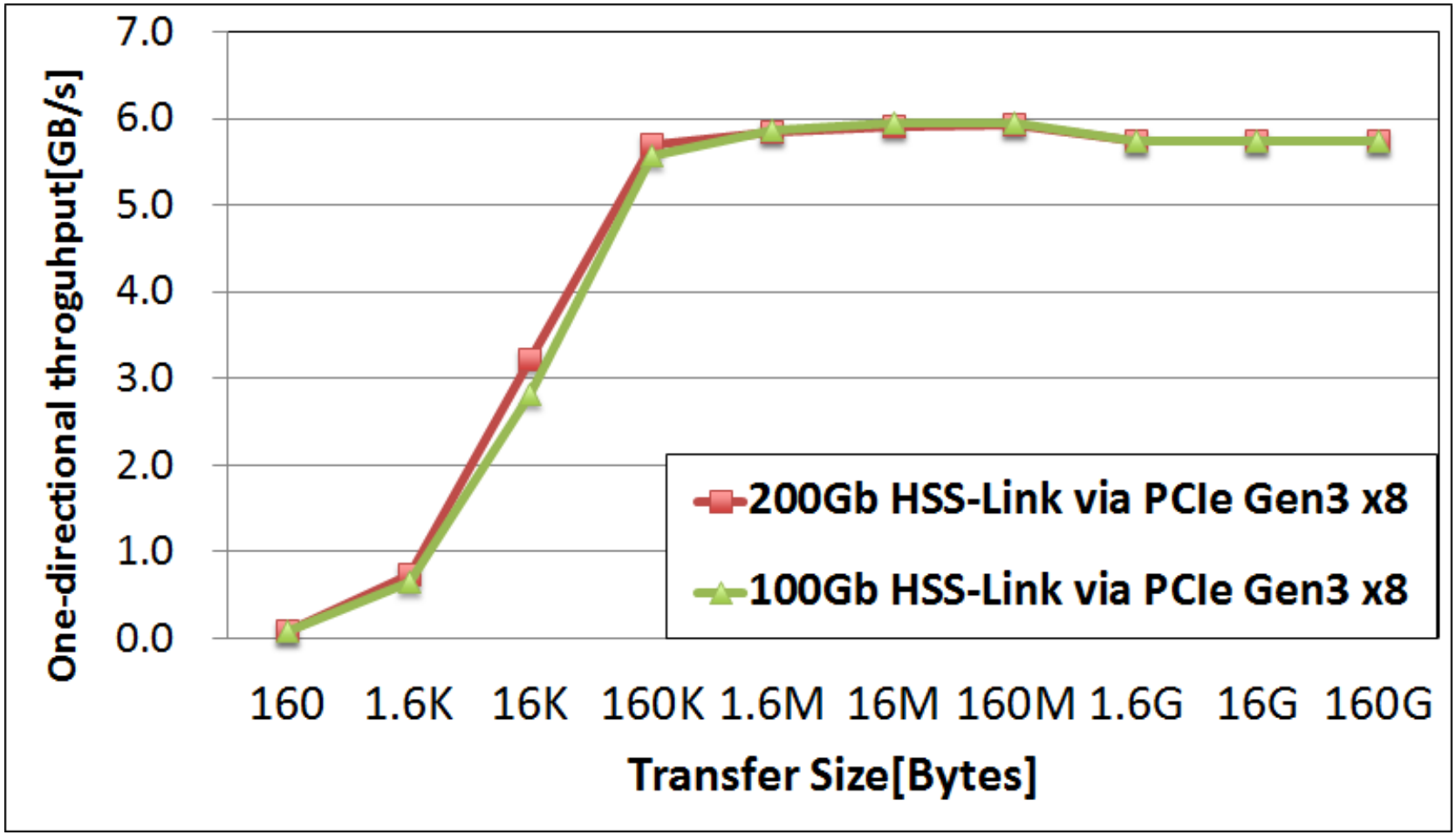
- up to 96% of theoretical peak
- good scalability up to 3 channels

Case-B: 1-hop routing via FPGA



- FPGA hop latency is just 20ns

Case-C: PCIe intra-communication

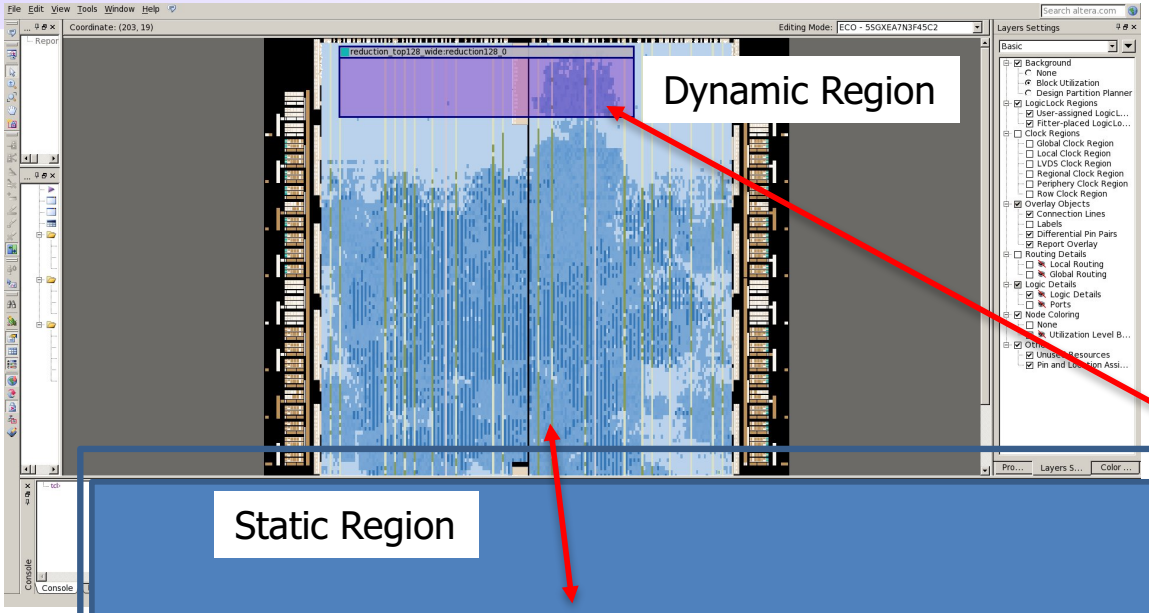


- bottlenecked by PCIe bandwidth
- >90% of theoretical peak performance

Challenge (2): Programming

- OpenCL is not perfect but best today
  - Much smaller number of lines than Verilog
  - Easy to understand even for application users
  - Very long compilation time to cause serious TAT for development
  - Not perfect to use all functions of FPGA
- We need "glue" to support end-user programming
  - Similar to the relation between "C and assembler"
    -> "OpenCL and Verilog"
  - Making Verilog-written low level code as "library" for OpenCL
  - Potentially possible (by Altera document), but hard
  - Challenge: Partial Reconfiguration
- Open source for applications & examples
  - Combination of OpenCL app. + Verilog modules
  - On commodity platform (⇔ PEACH2: special hardware)


Partial Reconfiguration & Partial Loading on FPGA



Persona1: Reduction Comp. for Particles
Persona2: LET Accelerator

Static Region

PEACH3 Switch

Mega Function for Controller

Controller

ROM

Reconfiguration Control Circuit

Persona1

Wrapper

Freeze control circuit

Persona2

Wrapper

It is possible to switch Personas even running Static Region logic is running

Challenge (3): system configuration

- Intra-node connection issue
  - PCIe is the only solution today
  - Intel HARP: QPI (UPI) for Xeon + FPGA (in MCM)
  - IBM OpenCAPI: POWER + FPGA (IP provided)
- How to make the system compact
  - HARP solution is very good to save footprint, but no I/O allowed for FPGA
  - OpenCAPI has flexibility on FPGA external link
  - NVLINK2 access via OpenCAPI ?

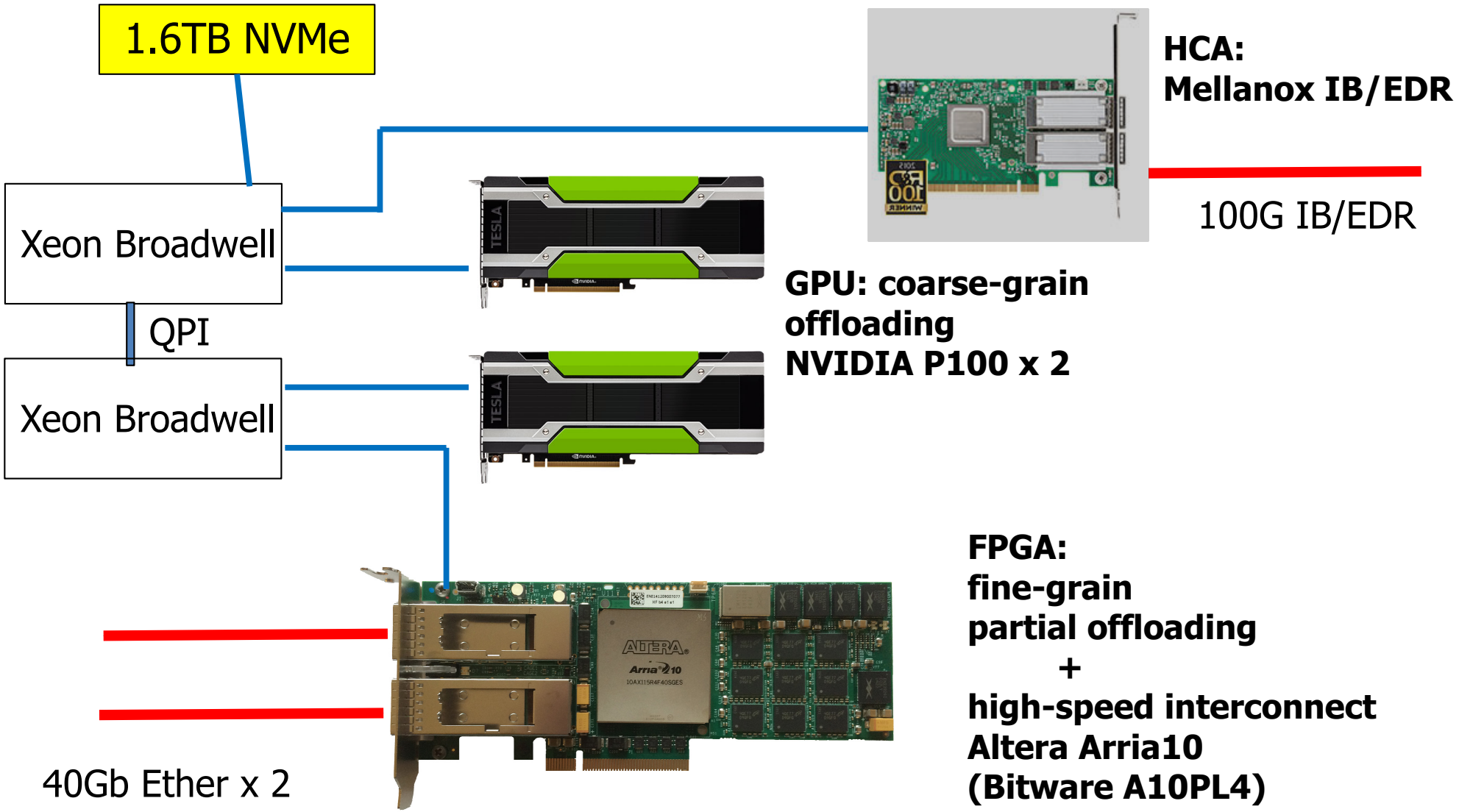

CCS's Next Accelerated Computing "PACS-X" Project


PACS-X (ten) Project

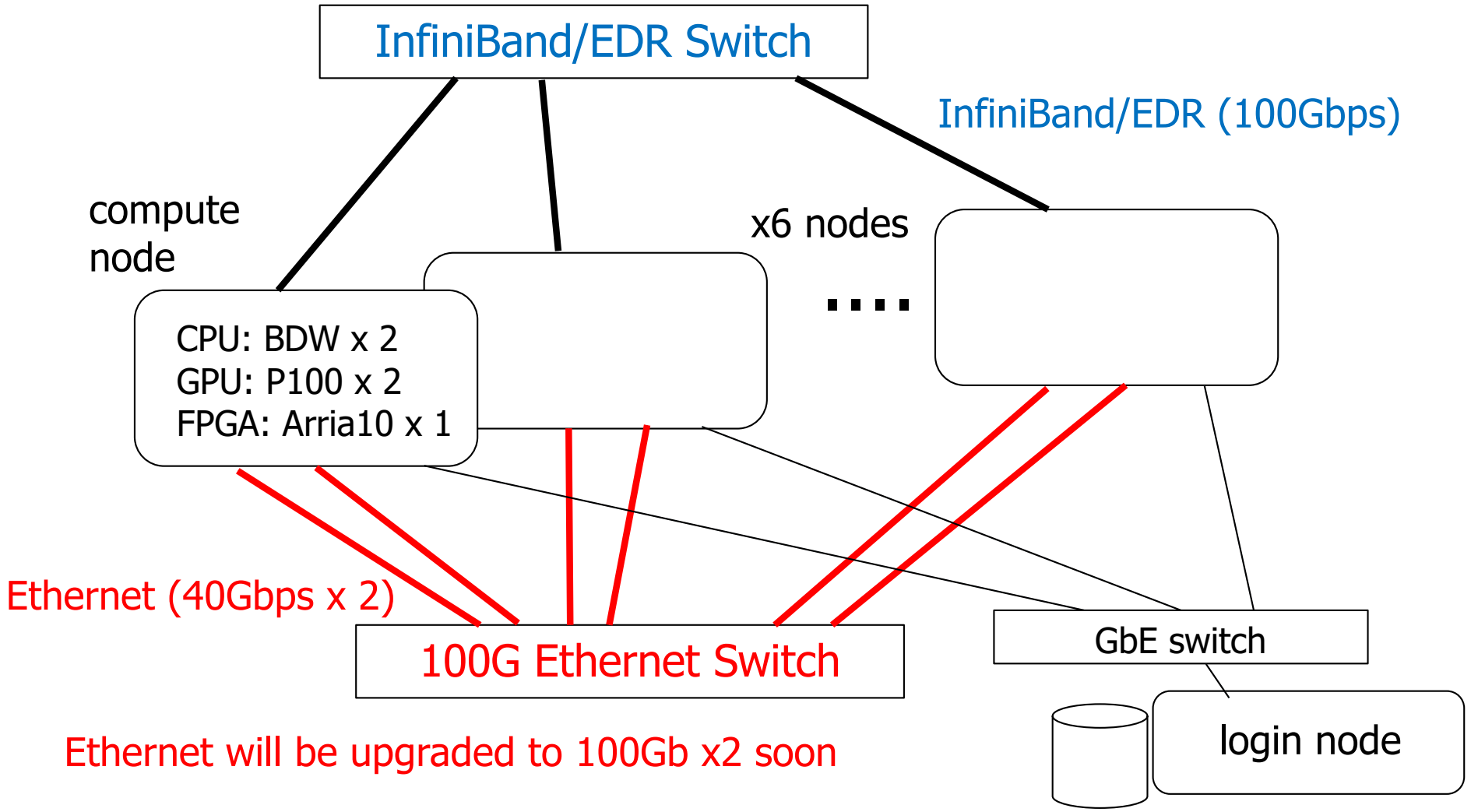
- PACS (Parallel Advanced system for Computational Sciences)
  - a series of co-design base parallel system development both on system and application at U. Tsukuba (1978~)
  - recent systems focus on accelerators
    - PACS-VIII: HA-PACS (GPU cluster, Fermi+Kepler, PEACH2, 1.1PFLOPS)
    - PACS-IX: 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 (also used for CREST)


PPX: latest multi-hetero platform (x6 nodes)



PPX mini-cluster system



Radiation Transfer in early Universe

- Interaction among Light (Radiation) from Objects and Material

  - Reionization of atoms
  - Molecules split by Light
  - Heating of gas by Light
  - Mechanical dynamics by gas pressure

They are so important for early universe generation

- generation of stars
- reionization of universe by photons from galaxy

- Research so far: by ray tracing
  - Very costly computation
  - Approximation is applied so far

Radiation Transfer Calculation

- Radiation transfer from single light source

light to each mesh from the source

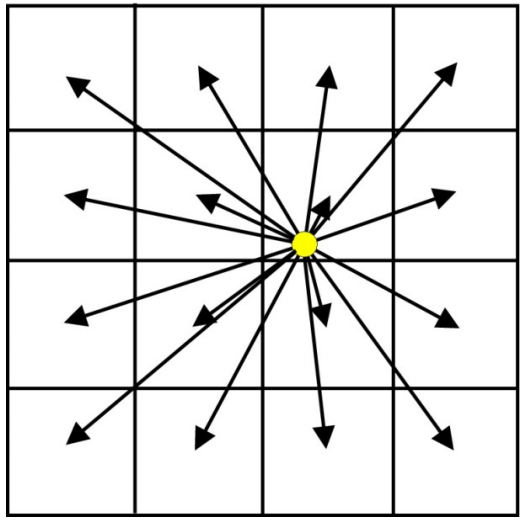
$$\frac{dI(\nu)}{d\tau(\nu)} = -I(\nu)$$

ν : vibration

$I(\nu)$: strength of radiation

$\tau(\nu)$: optical thickness

computation cost $\propto N_{\mathrm{m}} N_{\mathrm{s}}$



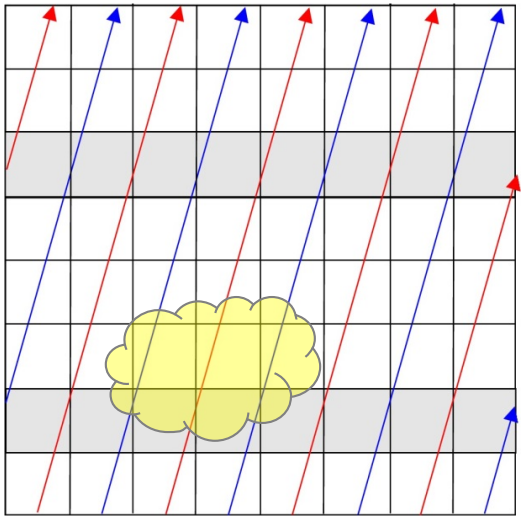
- Diffuse radiation transfer from spread light sources in the space

a number of parallel lights from the boundary

$$\frac{dI(\nu)}{d\tau(\nu)} = -I(\nu) + S(\nu)$$

$S(\nu)$: source function

computation cost $\propto N_{\mathrm{m}}^{5/3}$



ARGOT: Radiation Transfer Simulation code

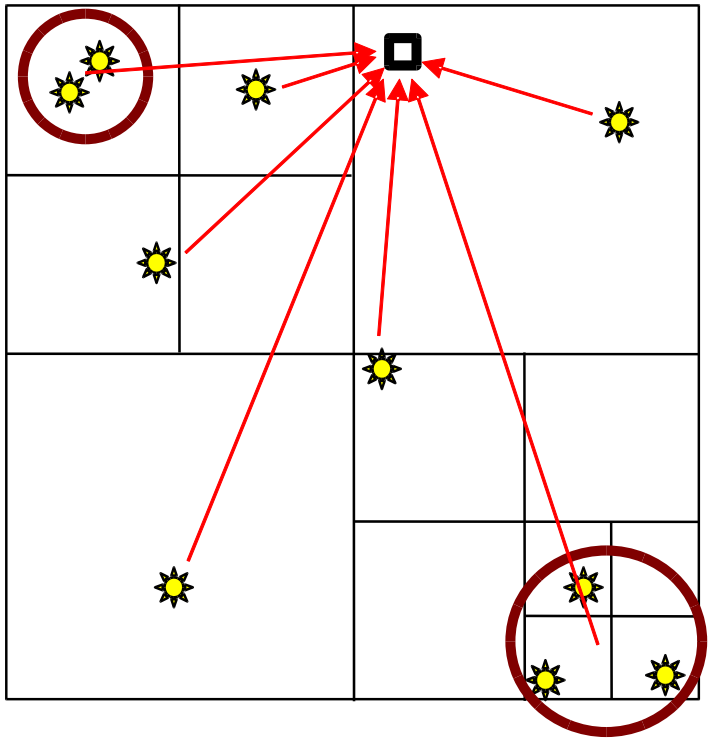
- Simultaneous simulation on radiation transfer and fluid dynamics
- Applied to single light source
- For multiple light sources, using Tree structure to merge the effect

  Computation Cost $\propto N_{\mathrm{m}} N_{\mathrm{s}} \Longrightarrow \propto N_{\mathrm{m}} \log N_{\mathrm{s}}$

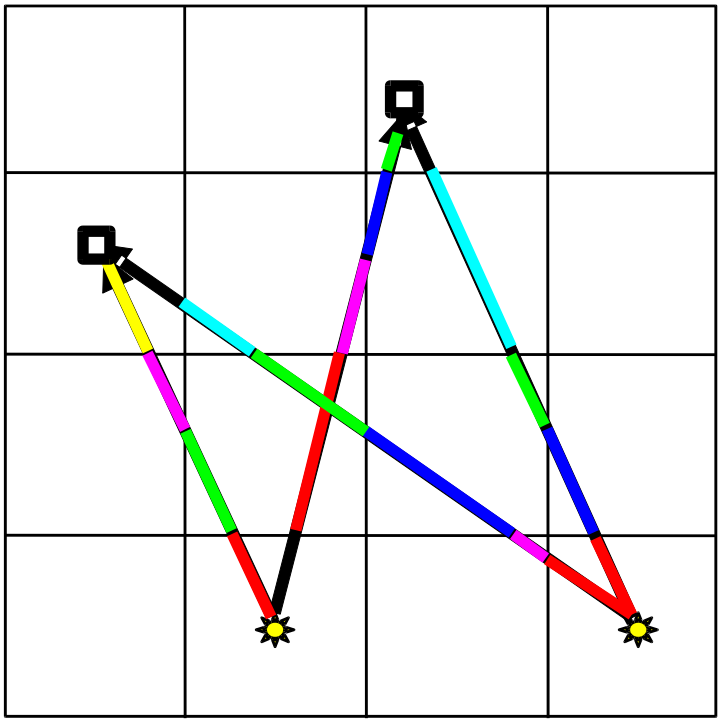
- Currently written by CUDA (GPU) +OpenMP (CPU)
- Scalable with MPI
- https://bitbucket.org/kohji/argot


ARGOT(Accelerated Ray-Tracing on Grids with Oct-Tree)

ARGOT method

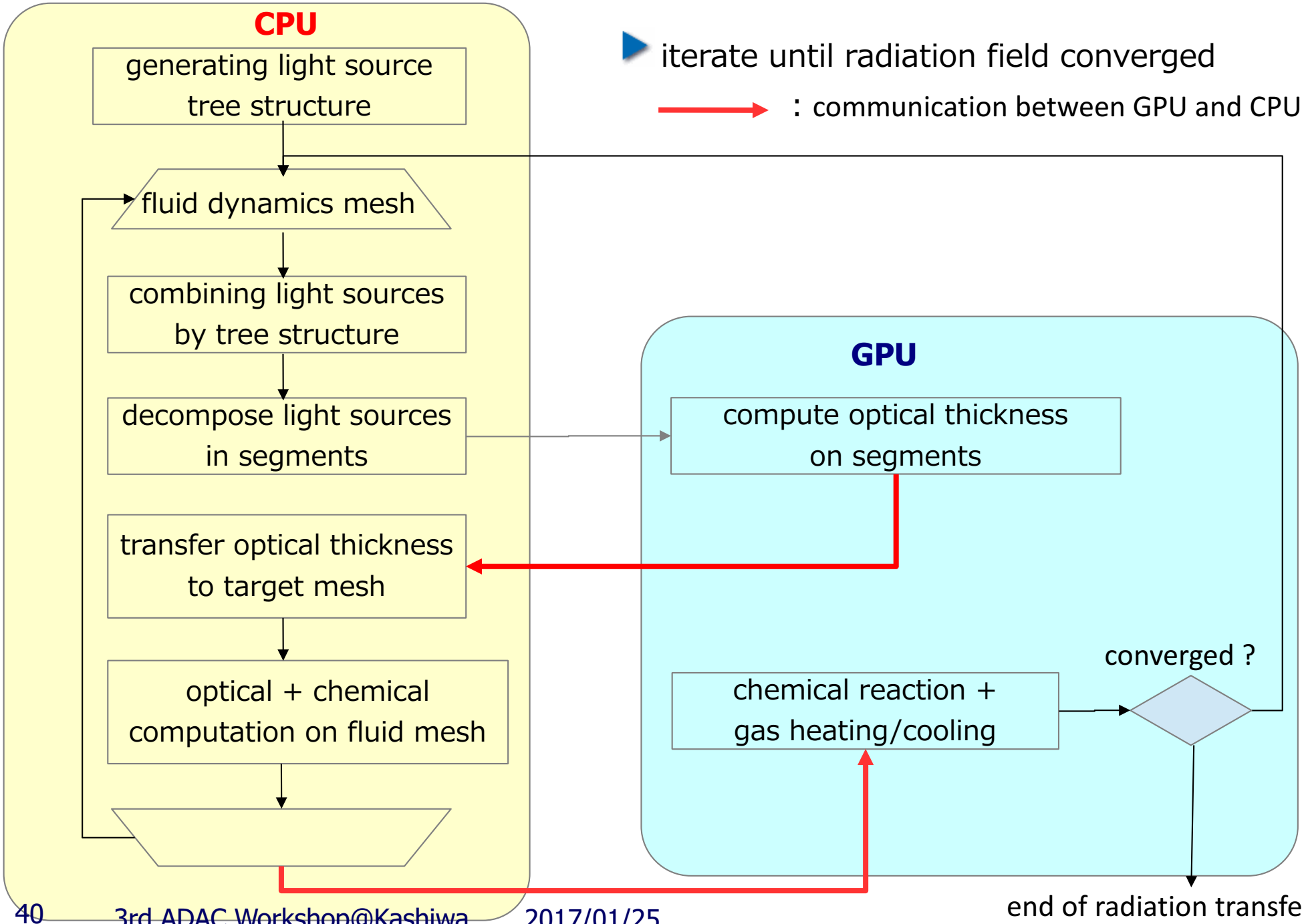


Parallelized ARGOT method



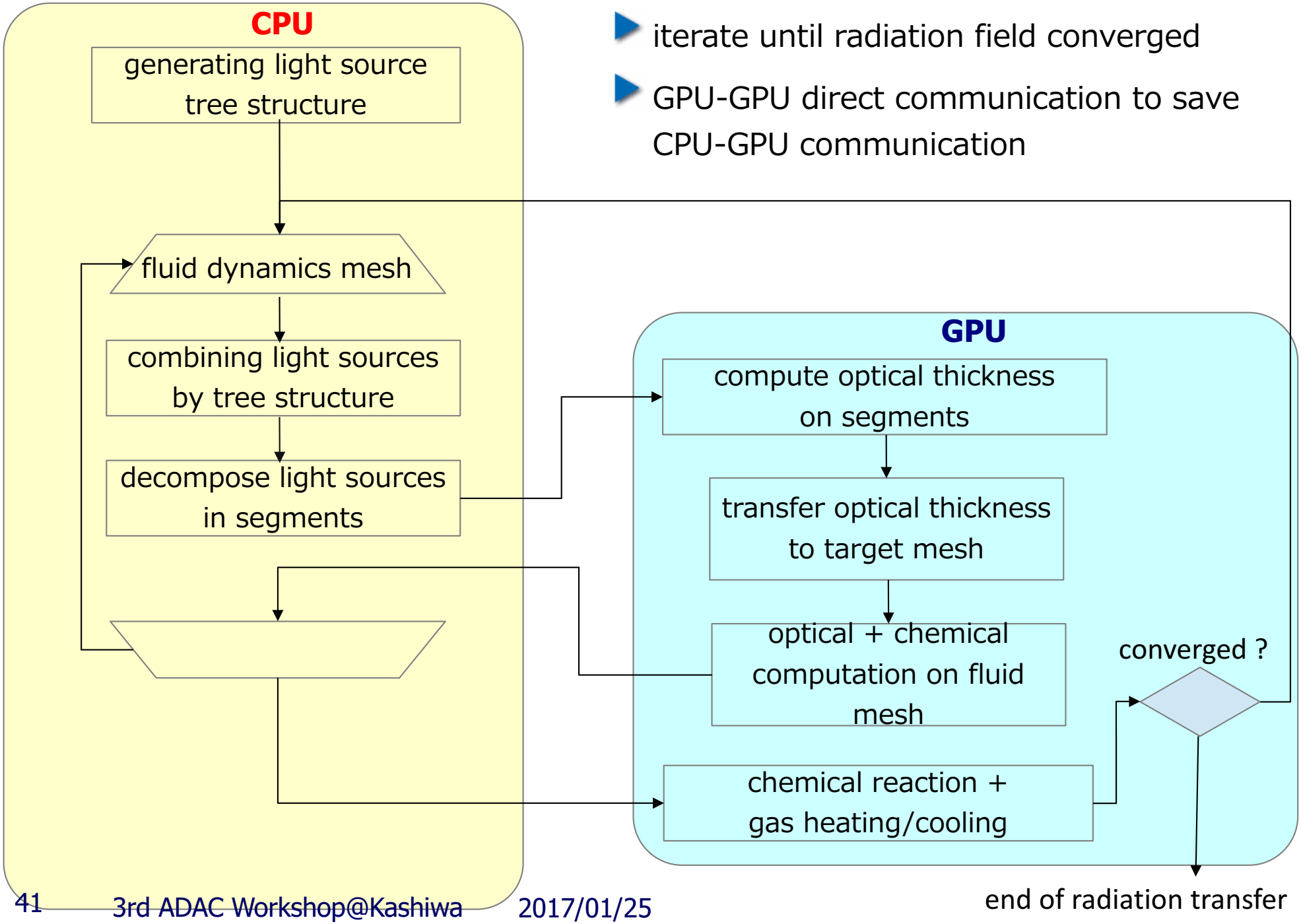
- Basic structure and method are similar to LET on gravity
  ⇒ also possible to implement on FPGA
- Computation accuracy (precision) is low, FP16 is too much

CPU+GPU processing

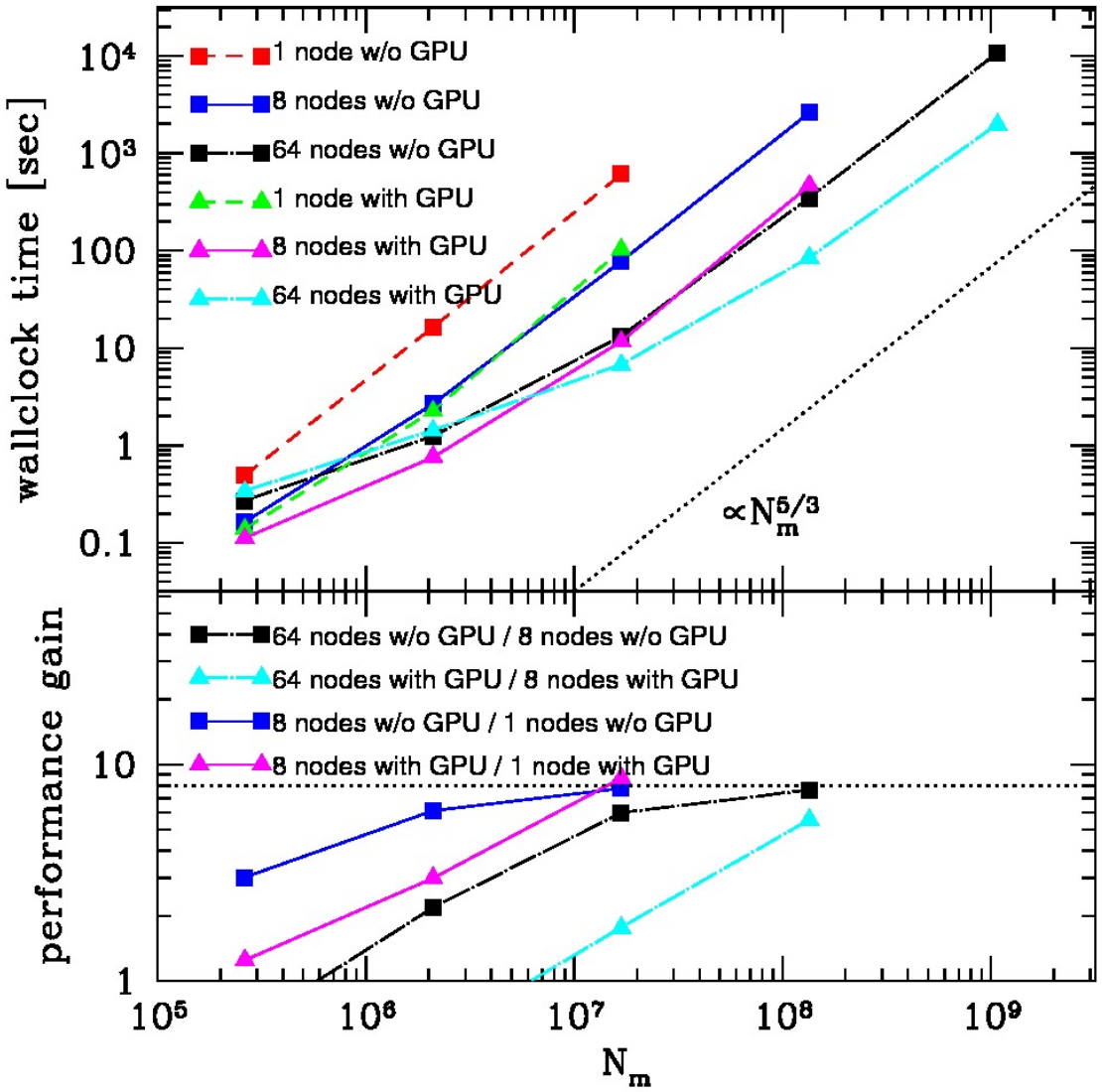


CPU+GPU+TCA processing

- iterate until radiation field converged
- GPU-GPU direct communication to save CPU-GPU communication



Parallelized by MPI



- Scaling with > 64^3mesh/node
- On GPU CUDA code, scaling with > 128^3mesh /node
- Problem: inter-node communication by MPI when the light crosses on the cell border
- Offloading to FPGA
  - **Fine tuning on precision and algorithm**
  - **High speed communication without PCIe bottleneck**
  - **With High Level Language ?**

Programming framework (on going)

- on FPGA, OpenCL + Verilog HDL combination
  - OpenCL for high level algorithm description
  - Verilog HDL for low level functions on hardware depended features (interconnection, peripherals) and highly accelerated library (BLAS, gravity, etc.)
- on CPU+GPU, OpenACC + FPGA offloading
  - GPU: OpenACC
  - FPGA: OpenCL (with Verilog HDL)
  - not data parallel decomposition
  - function mapping feature


Summary

- Multi-hetero environment for multi-level complexity, multi-physics simulation
- Issues
  - programming
  - external/internal interconnect
  - space saving
  - overall view of system/programming
- We started PACS-X project
  - multi-hetero platform experiment
  - toward FPGA for HPC solution
  - supportive hardware platform for our last year work on CREST
- Open source distribution of OpenCL & Verilog codes