演算通信融合機構による計算科学アクセラレーション

朴　泰祐
筑波大学・計算科学研究センター/システム情報工学研究科

共同研究者：
天野英晴（慶應義塾大学）、塙敏博（東京大学）
梅村雅之、山口佳樹、吉川耕司、小林諒平、藤田典久、大畠佑真、高山尋考（筑波大学）

ポストペタCREST：朴チームについて

- JST-CREST研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」、研究課題名「ポストペタスケール時代に向けた演算加速機構・通信機構統合環境の研究開発」（H24.10～H30.3、総額3.4億円）
- メンバー
  - 朴泰祐（代表、筑波大）
    天野英晴（慶大）、村井均（理研）、梅村雅之（筑波大）
- テーマ
  - 並列アクセラレータクラスタにおける通信ボトルネックの解消、特にstrong scalingにおいて重要なlatencyの削減をhardwareとsoftwareの協調で解消
    ⇒ TCA (Tightly Coupled Accelerators)
  - TCAのプロトタイプ実装としてPCIe intelligent switchであるPEACH2 (PCI Express Adaptive Communication Hub ver.2)をFPGAで実装
    ⇒ GPU間データ通信をCPUとそのネットワークを介在することなく実行
  - 研究期間後半ではFPGAをより積極的にアプリケーション加速「にも」利用
    ⇒ Accelerator in Switch
  - 高機能・高レベルのアクセラレータクラスタ向け並列言語の開発
    ⇒ XcalableACC


セッション構成

- 演算通信融合機構による計算科学アクセラレーション（朴泰祐）
  - TCAとAccelerator in Switch
- FPGAにおける計算科学アプリケーションの部分オフローディング（藤田典久）
  - OpenCL for FPGA
- TCA機構におけるGPU対応GASNetの実装（佐藤賢太）
  - TCA/PEACH2の汎用通信レイヤへの適用
- アクセラレータクラスタ向けPGAS言語XcalableACCの実装状況報告（田渕晶大）
  - XcalableACC実装


Agenda

- TCA and PEACH2
- Accelerator in Switch
  - First astrophysics problem – LET offloading
  - Next challenge on Radiation Transfer problem
- Challenges for Accelerator in Switch
- PACS-X & PPX
- Summary

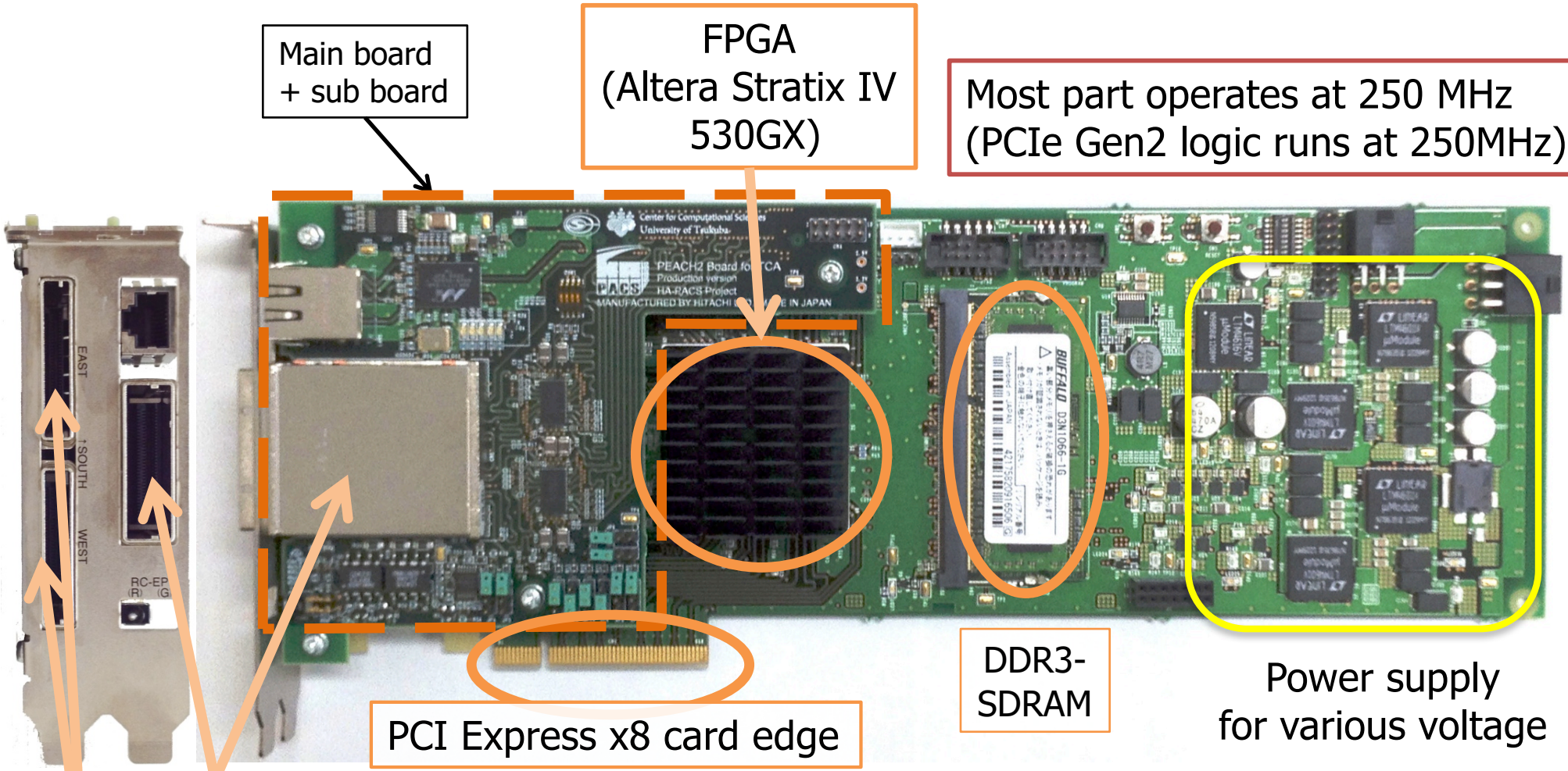

TCA (Tightly Coupled Accelerators)

Issues on accelerated (GPU) computing

- Trading-off: Power vs Dynamism (Flexibility)
  - fine grain individual cores consume much power, then ultra-wide SIMD feature to exploit maximum Flops is needed
- Interconnection is a serious issue
  - current accelerators are hosted by general CPU, and the system is not stand-alone
  - current accelerators are connected by some interface bus with CPU then interconnection
  - current accelerators are connected through network interface attached to the host CPU
- GPU is not perfect
  - highly regular (SIMD-like) computation to be supported by thousands of cores/threads
  - irregularity degrades the performance drastically
  - CPU is not enough powerful

- How to provide low-latency/high-bandwidth communication among accelerators
- How to fill the gap between GPU-performance and CPU-flexibility

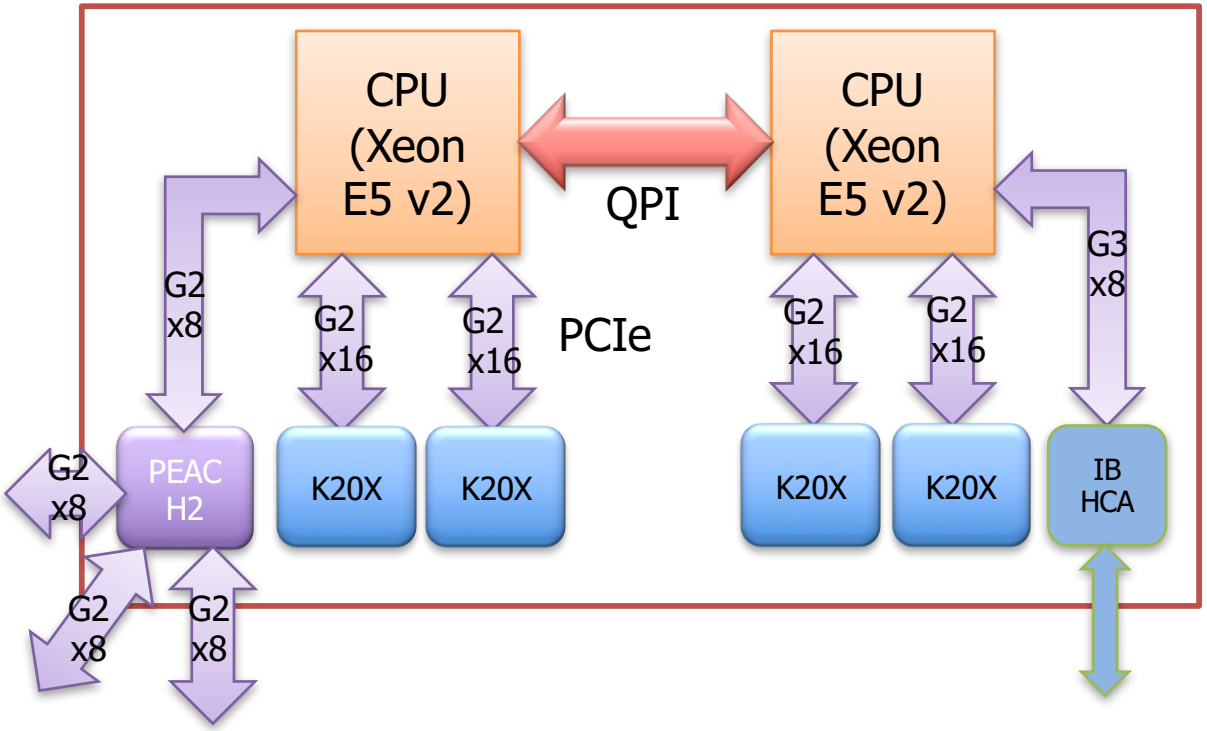
PEACH2 board – prototype implementation of TCA



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.

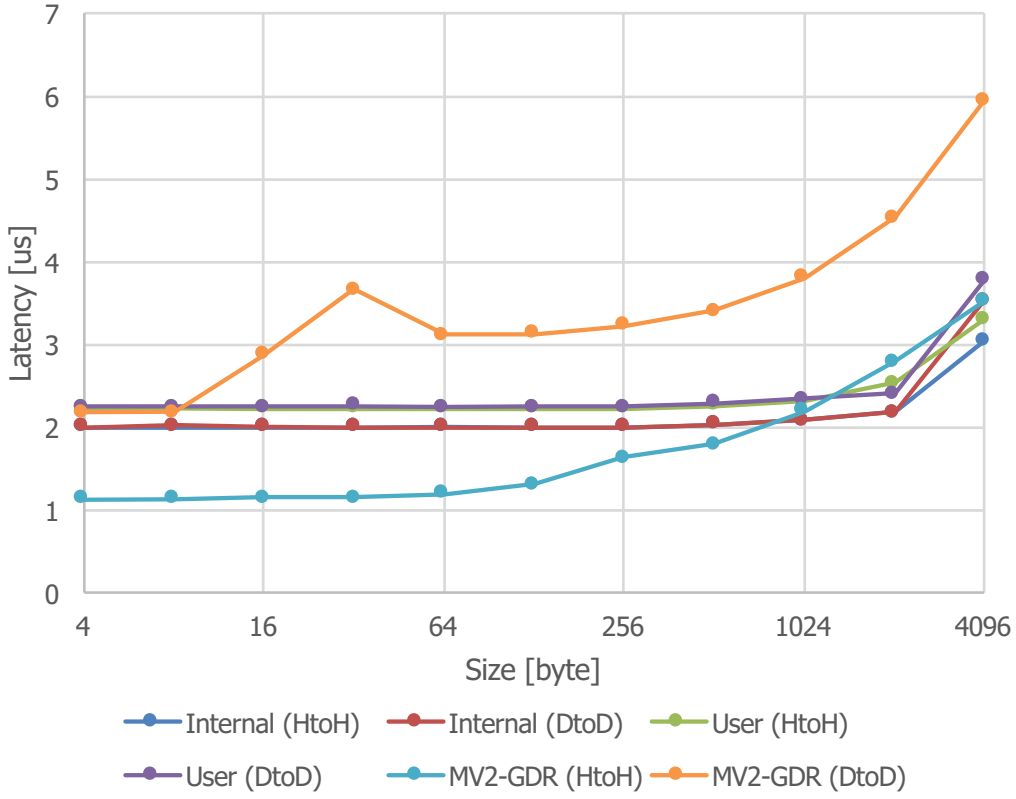


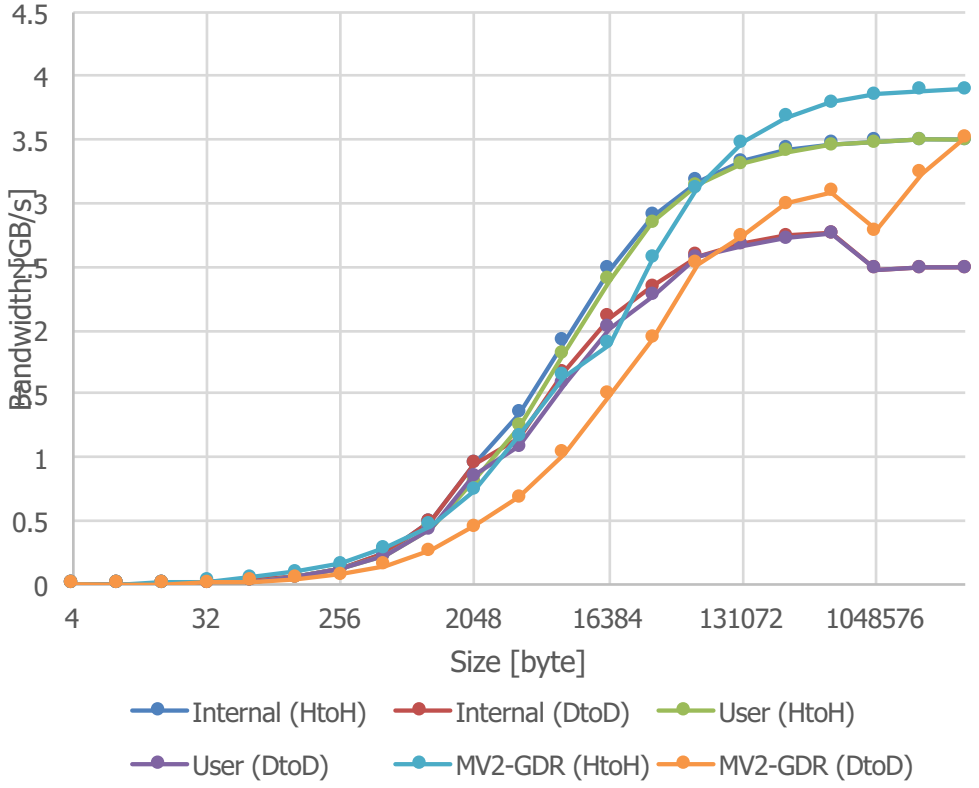
HA-PACS Base Cluster + TCA (TCA part starts operation on Nov. 1st 2013)

- 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

Ping-pong performance on TCA/PEACH2

Platform: HA-PACS/TCA
IB: Connect-X3 QDR (x 2rail)
GPU: K20X
(as in 2014)





Accelerator in Switch and Example on Astrophysics

Is GPU enough for everything ?

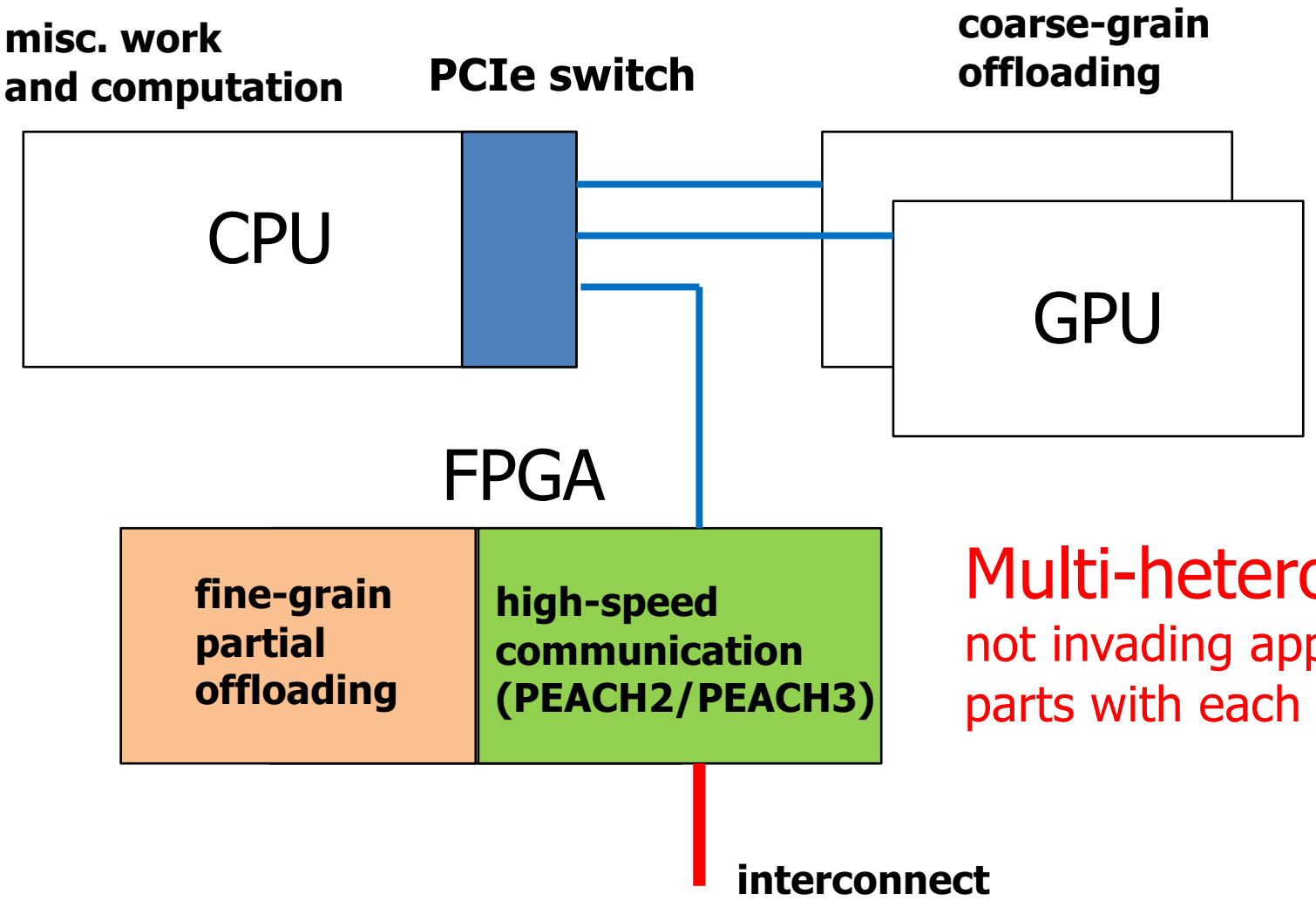
- GPU is good for:
  - coarse grained parallel applications
  - regular pattern of parallel computation without exception
  - applications relying on high memory bandwidth
- In recent HPC applications:
  - various *precision* of data is introduced (double, single, half...)
  - some computation is strongly related to communication
    ➟ fine ~ mid grained computation not suitable for GPU and too slow by CPU
  - complicated algorithm with exception, non-SIMD, partially poor parallelism


PEACH2/PEACH3 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 LE 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)

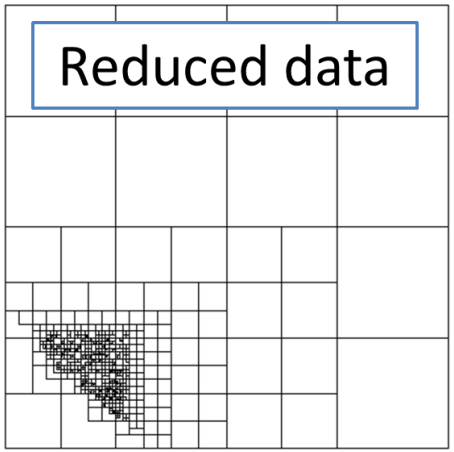
mask[id] array

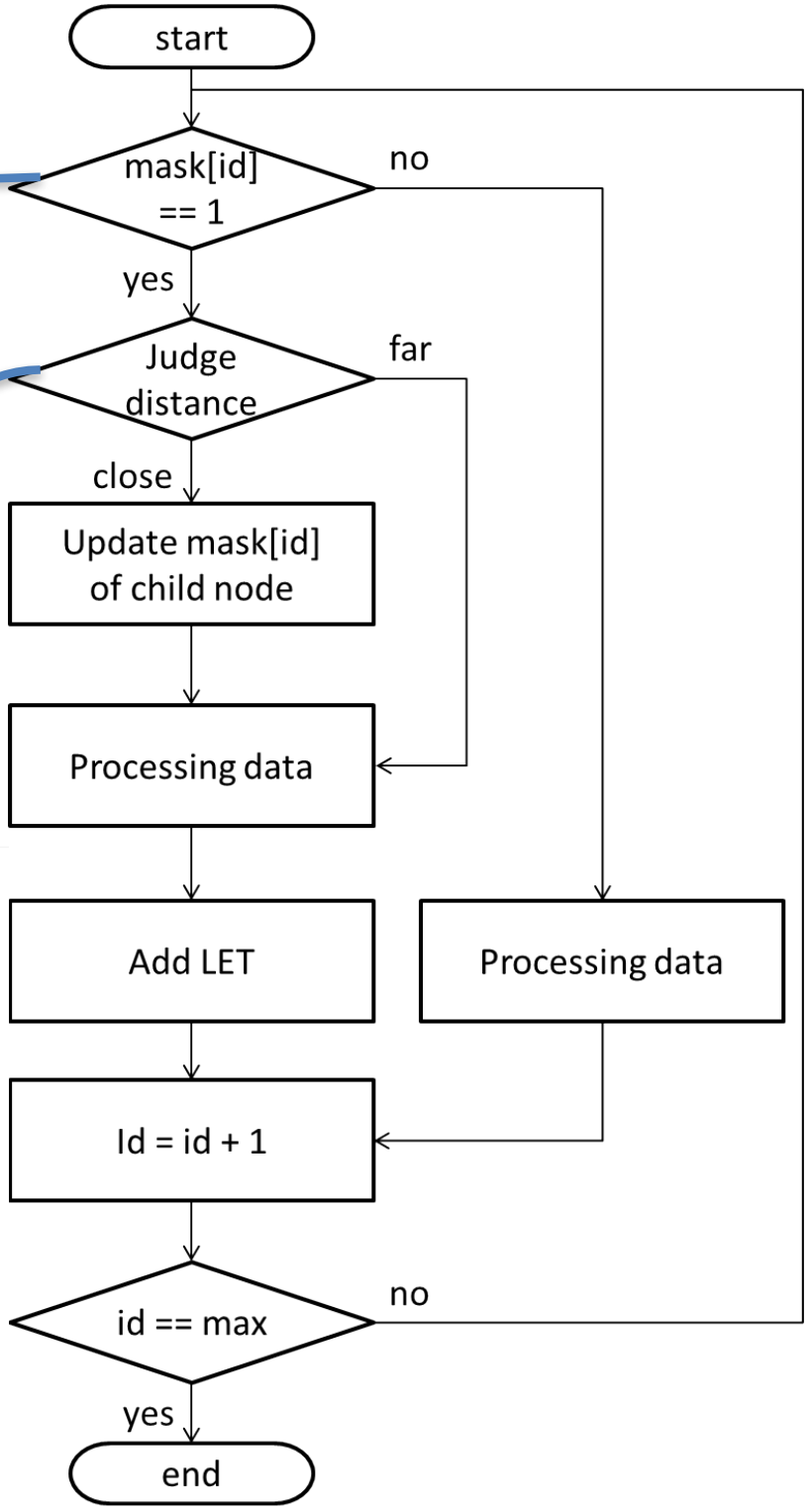
mask[id] == 0 skip

mask[id] == 1 add to LET

distance judgment

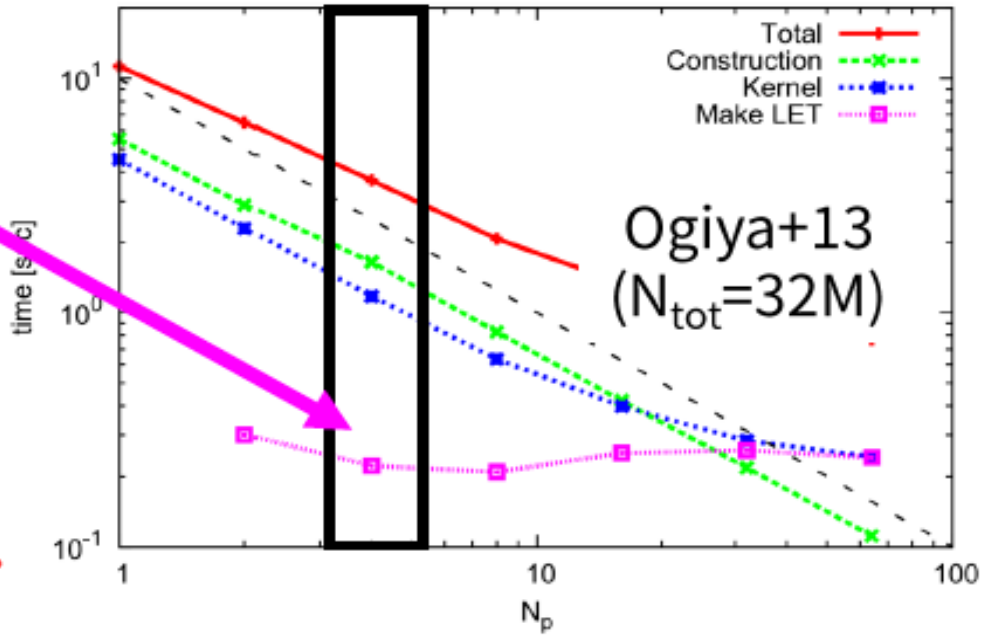
partial regional data in receiver side and each cell in sender side

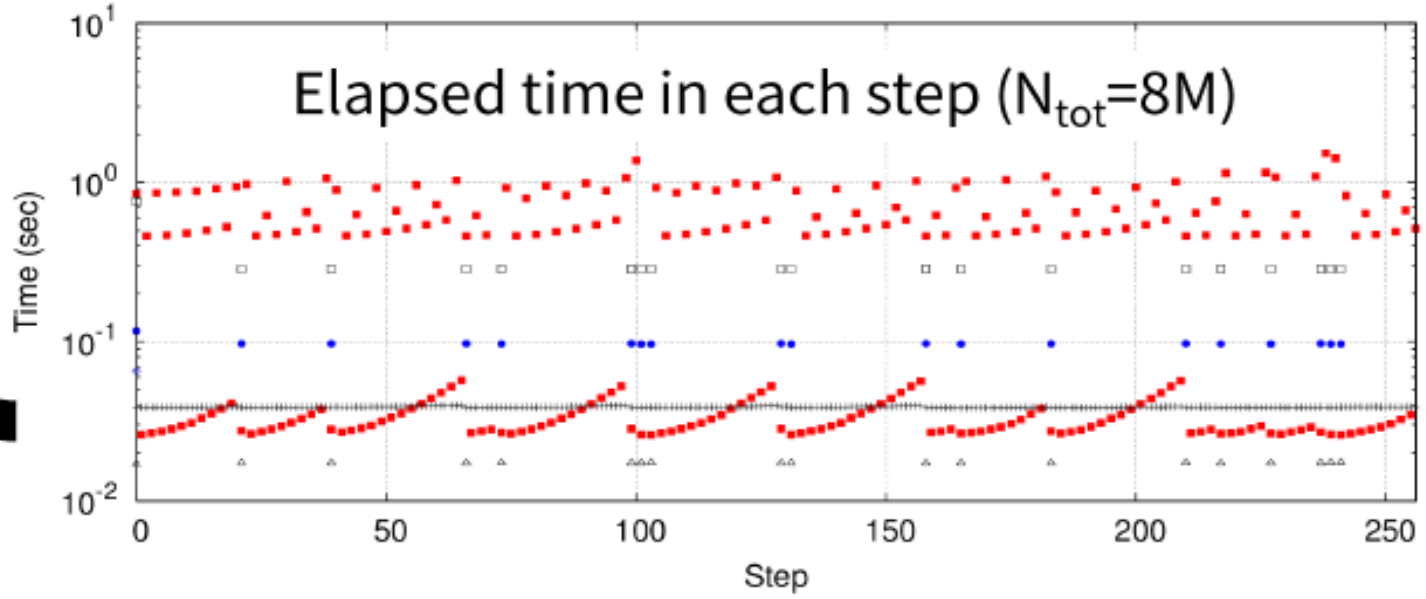




Cost of using LET

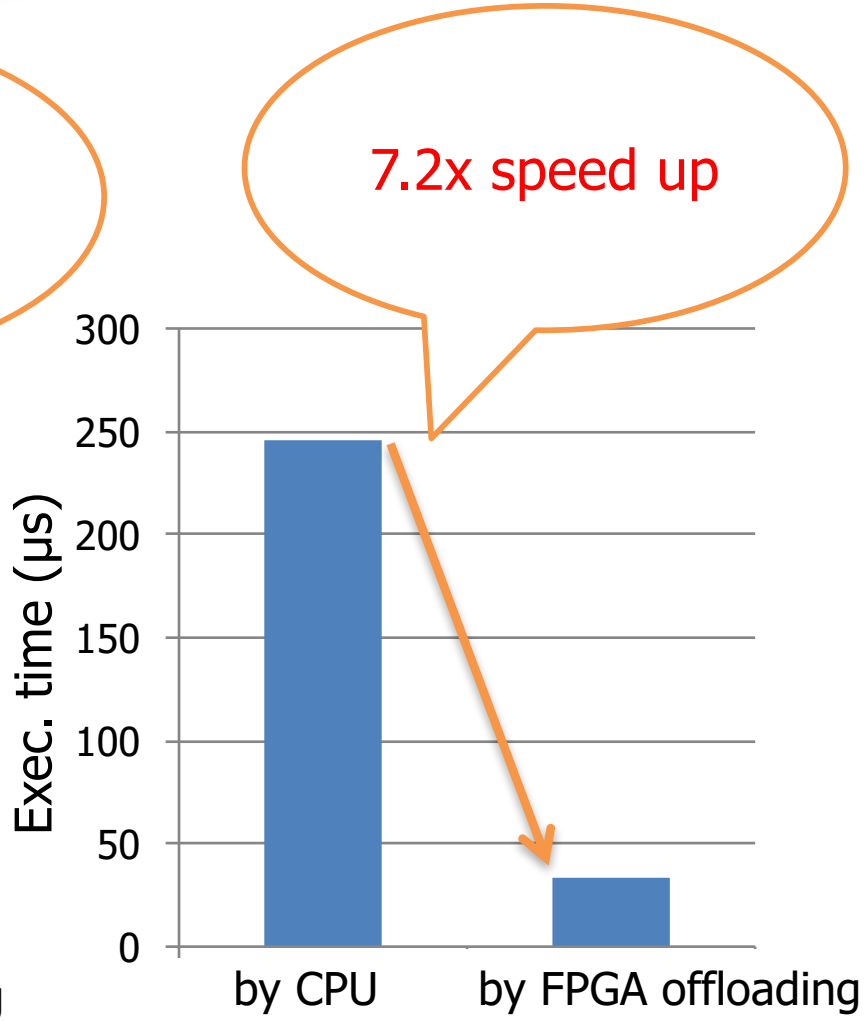
- ～0.2 sec. (Ogiya+13)
- Must be smaller than ～0.04 sec. to scale.
  - At least ～5 times acceleration is required.





Preliminary Evaluation

Exec. time for making LET



from LET making to GPU data transfer

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

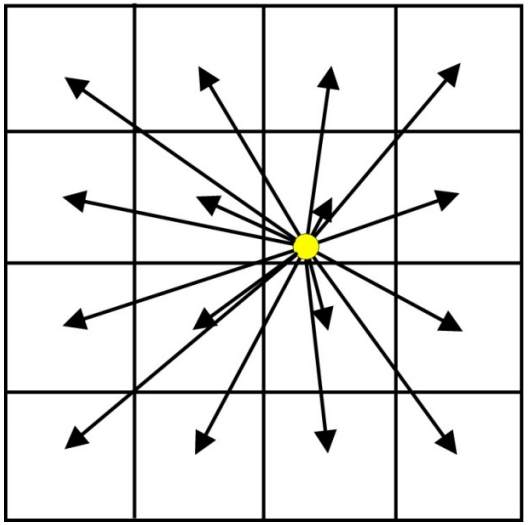
$$\frac{dI(\nu)}{d\tau(\nu)} = -I(\nu)$$

v : vibration

I(v) : strength of radiation

τ(v) : optical thickness

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



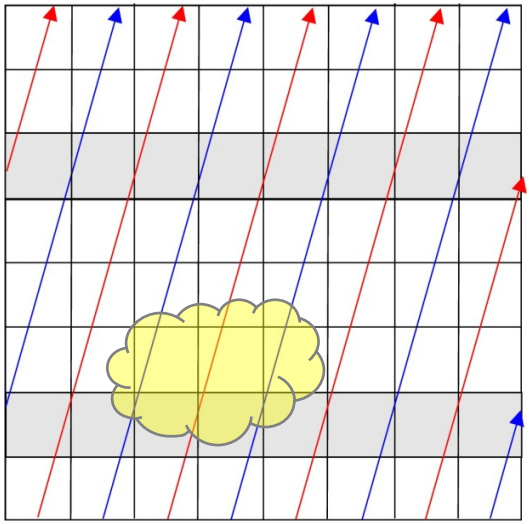
- Diffuse radiation transfer from spread light sources in the space

a number of parallel lights from the boundary

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

S(v) : source function

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



ARGOT: Radiation Transfer Simulation code

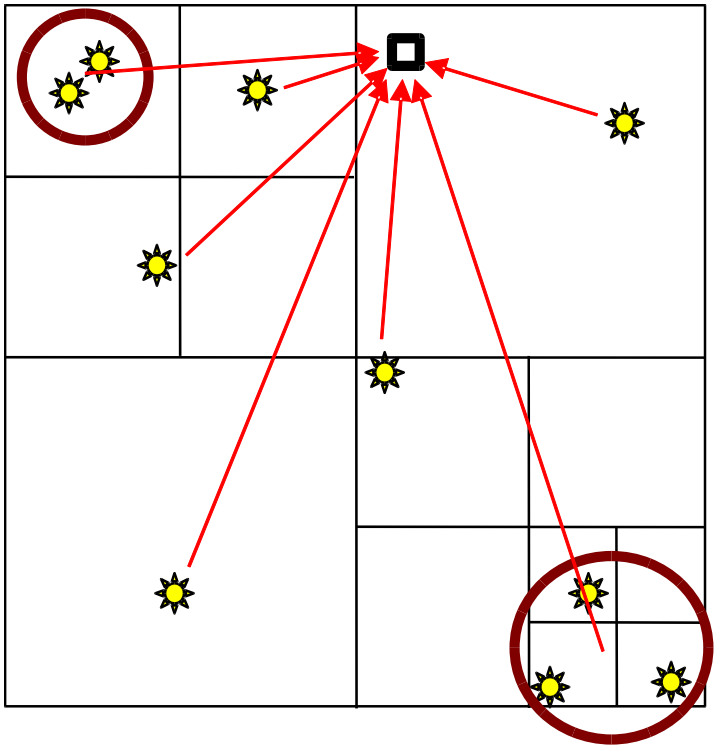
- Simultaneous simulation on radiation transfer and fluid dynamics
- Applied to single light source
- For multiple light sources, using Tree structure to merge the effect

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

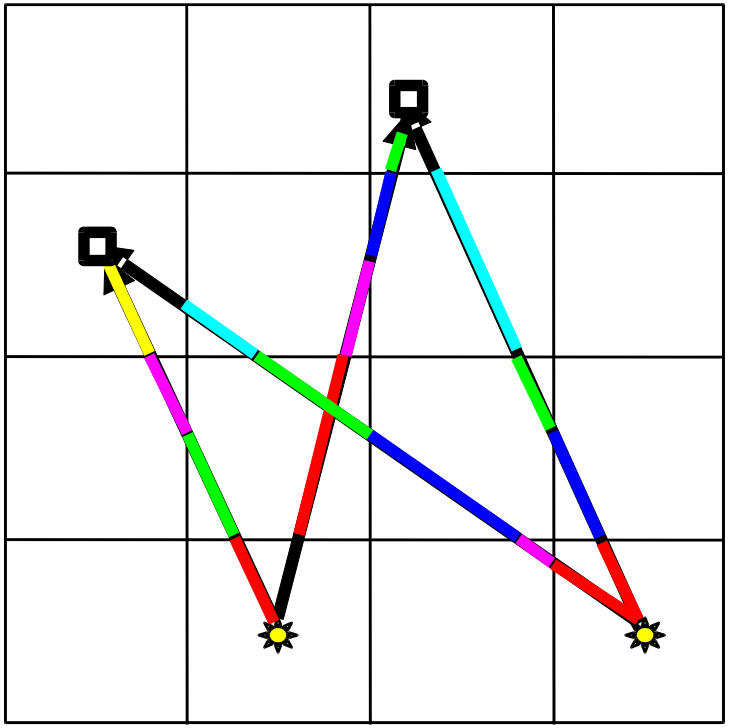
- Currently written by CUDA (GPU) +OpenMP (CPU)
- Scalable with MPI
- https://bitbucket.org/kohji/argot


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

ARGOT method

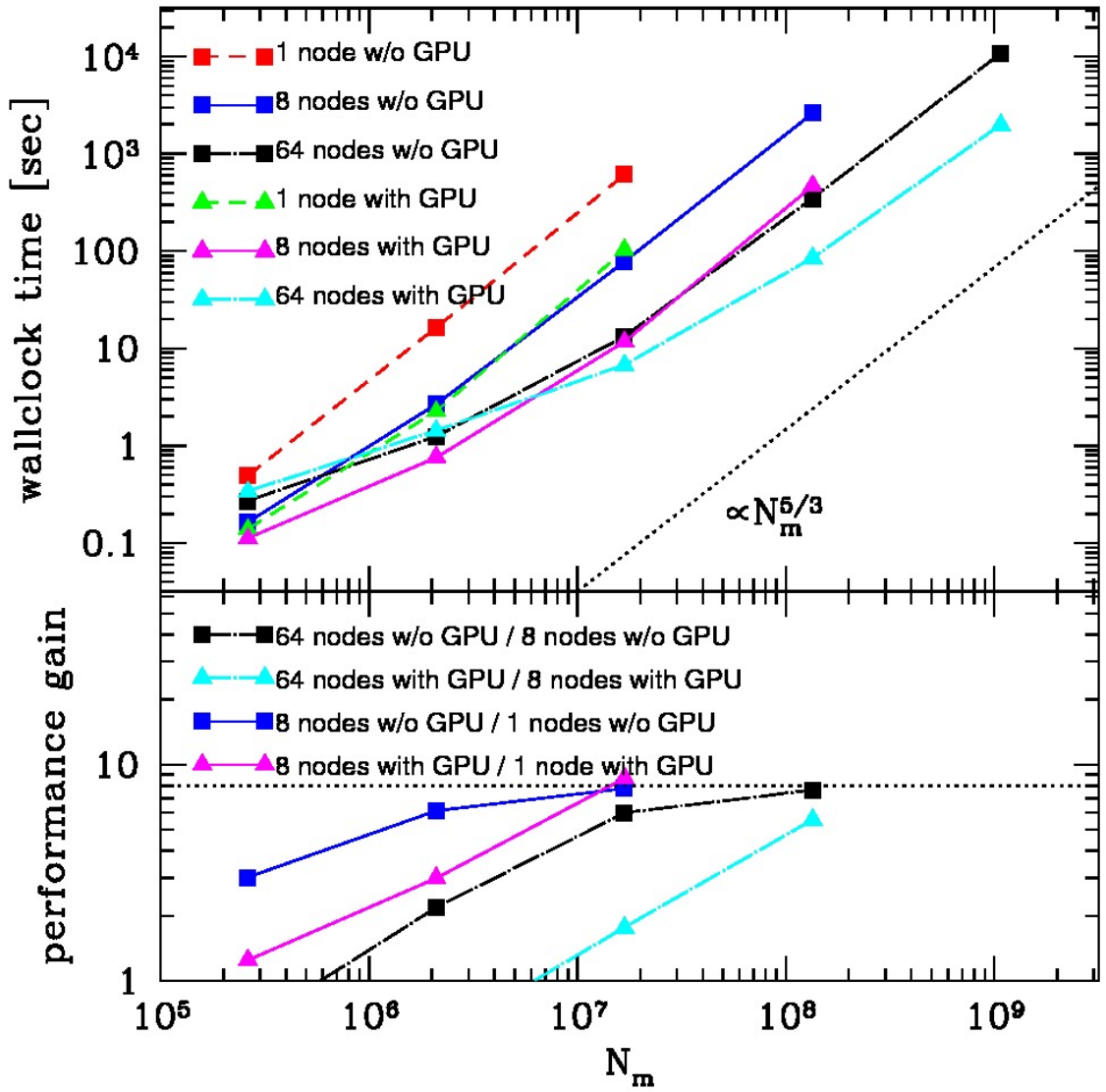


Parallelized ARGOT method



- **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**

Communication bottleneck on ARGOT



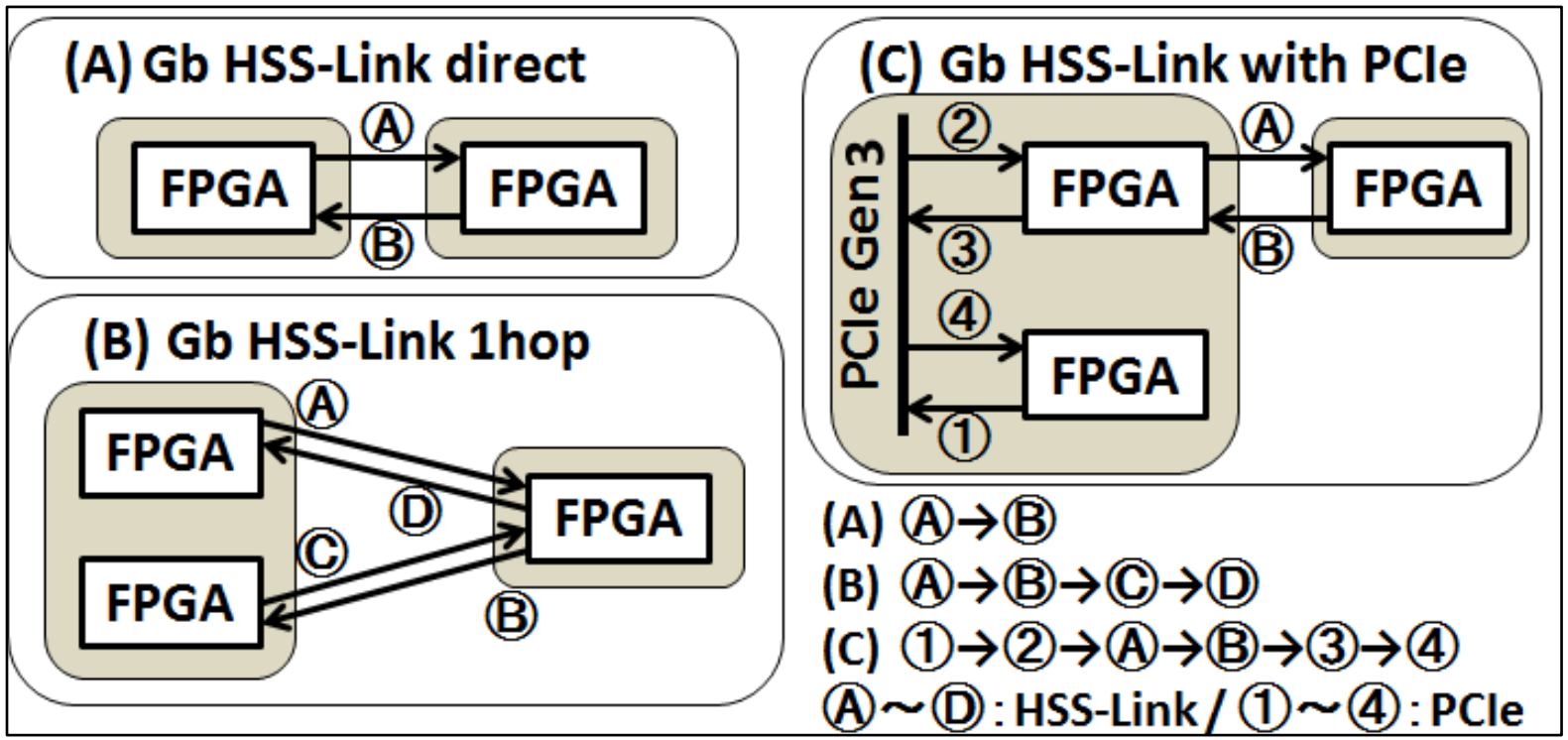
- Scaling with > 64^3mesh/node
- On GPU CUDA code, scaling with > 128^3mesh /node
- Problem: inter-node communication by MPI when the light crosses on the cell border
- Offloading to FPGA
  - **Fine tuning on precision and algorithm**
  - **High speed communication without PCIe bottleneck**
  - **With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

Issues on Accelerator in Switch

Challenge (1): Communication Performance

- 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



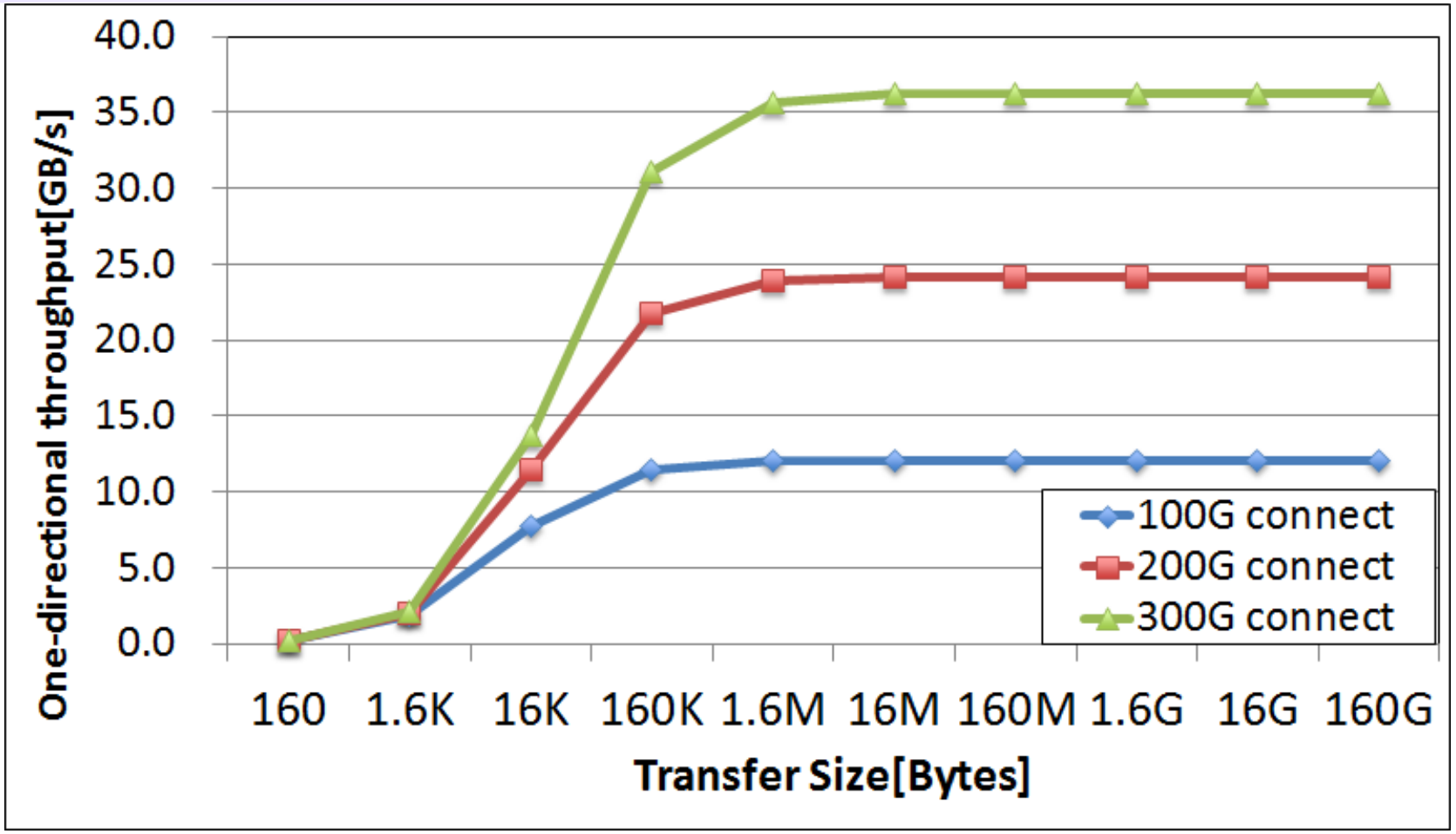
Xilinx XC7VX1140T(Virtex7)

Vivado 2016.1

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

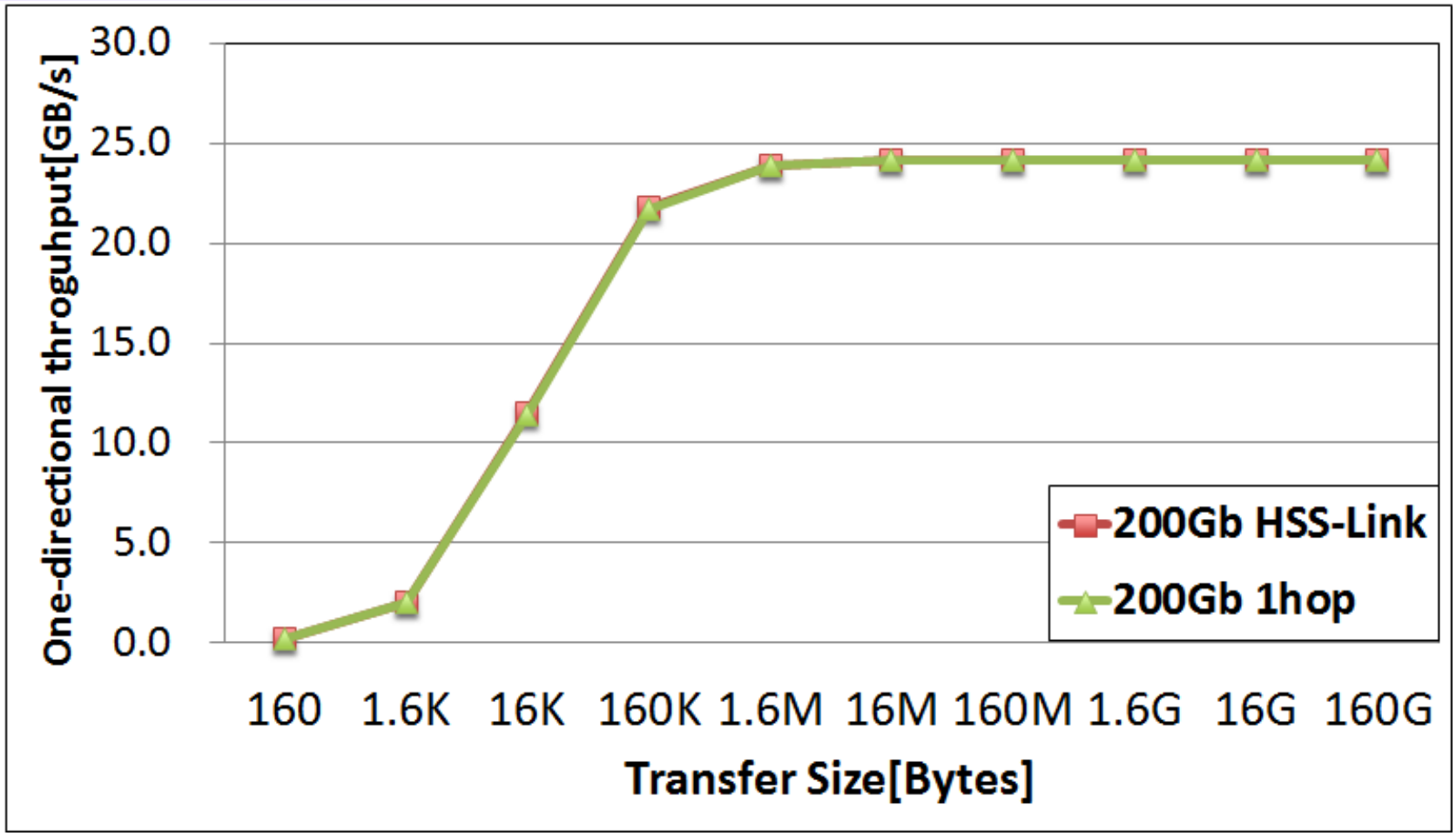
(Aurora 64B/66B v11.1)

Case-A: peer-to-peer



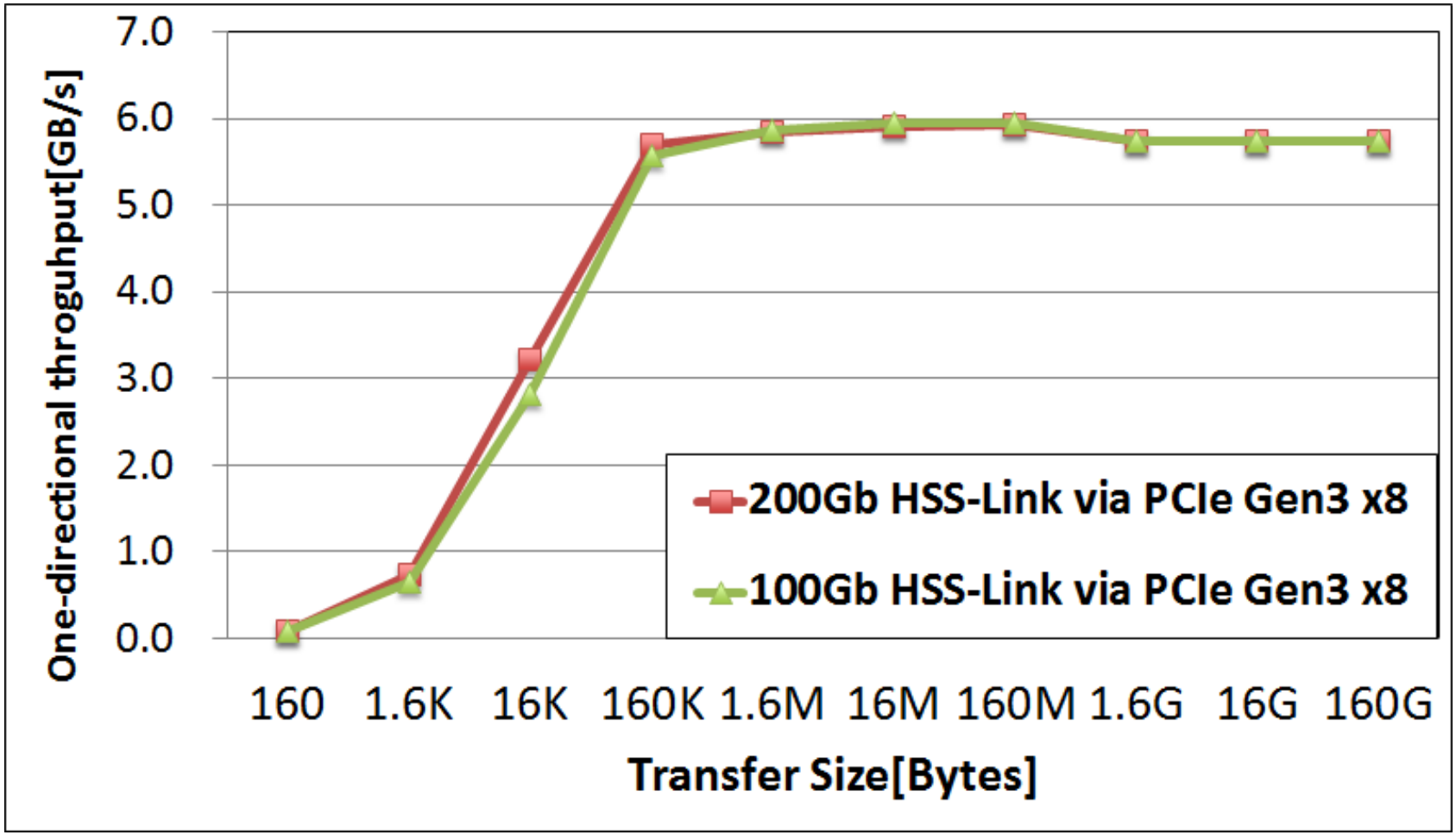
- up to 96% of theoretical peak
- good scalability up to 3 channels

Case-B: 1-hop routing via FPGA



- FPGA hop latency is just 20ns

Case-C: PCIe intra-communication

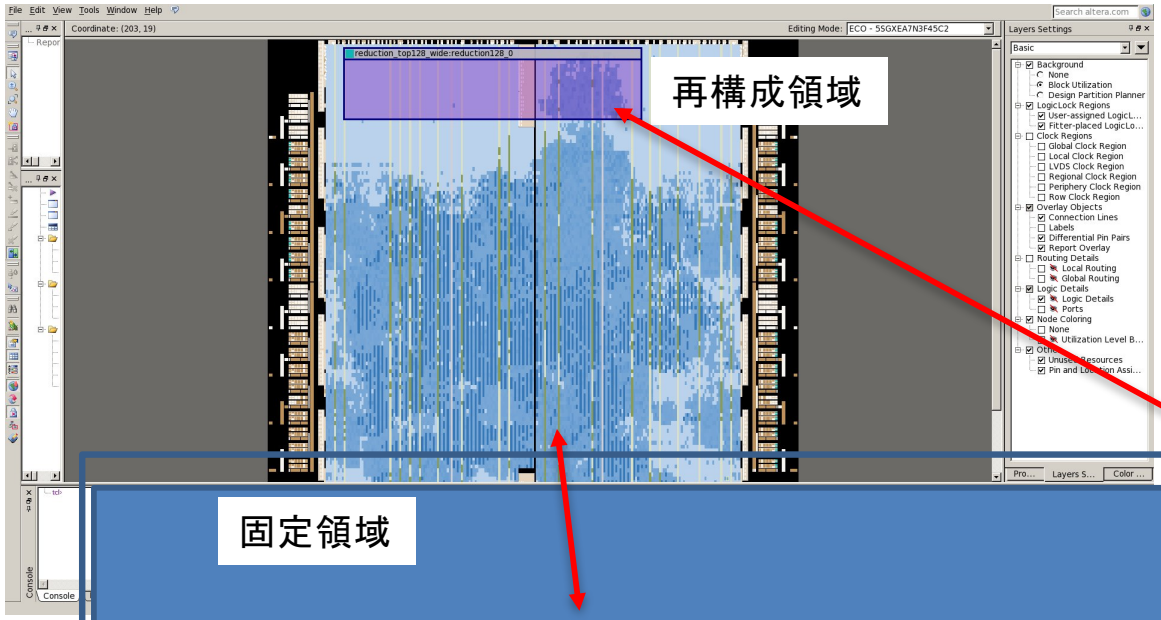


- bottlenecked by PCIe bandwidth
- >90% of theoretical peak performance

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
- Overall programming system


Accelerator in Switch のための機能切替



固定領域

固定領域モジュール
PEACH3のスイッチ

コントローラ用
Mega Function

コントローラ

PR用ROM

再構成制御回路

Persona1

ラッパー

Freeze制御回路

Persona2

ラッパー

固定部のスイッチを動作させたままで
Personaを切り替える

Persona1:天体計算の
リダクション演算
Persona2: LETのアクセラレータ

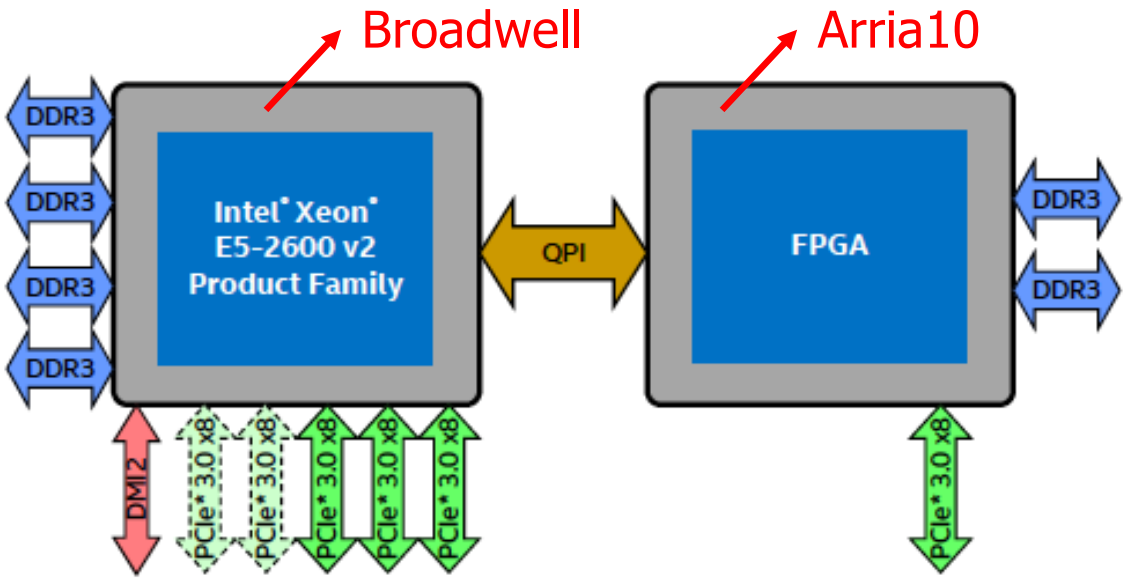
Challenge (3): System Configuration

- 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 I/O on FPGA is limited
  - OpenCAPI has flexibility on FPGA external link
  - NVLINK2 access via OpenCAPI ?


Intel+Altera heterogeneous architecture

IVB+FPGA Software Development Platform

Software Development for Accelerating Workloads using Xeon and coherently attached FPGA in-socket



Processor	Intel® Xeon® E5-26xx v2 Processor
FPGA Module	Altera Stratix V
QPI Speed	6.4 GT/s full width (target 8.0 GT/s at full width)
Memory to FPGA Module	2 channels of DDR3 (up to 64 GB)
Expansion connector to FPGA Module	PCIe 3.0 x8 lanes - maybe used for direct I/O e.g. Ethernet
Features	Configuration Agent, Caching Agent,, (optional) Memory Controller
Software	Accelerator Abstraction Layer (AAL) runtime, drivers, sample applications

Heterogeneous architecture with homogenous platform support

intel | 14

from PK Gupta, "Xeon+FPGA Platform for the Data Center", ISCA/CARL 2015
http://www.ece.cmu.edu/~calcm/carl/lib/exe/fetch.php?media=carl15-gupta.pdf

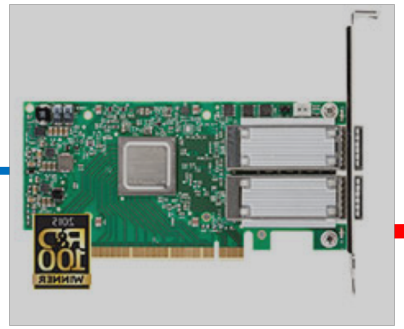
PACS-X and PPX

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


PPX: latest multi-hetero platform (x6 nodes)



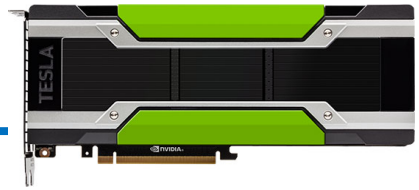
HCA:
Mellanox IB/EDR

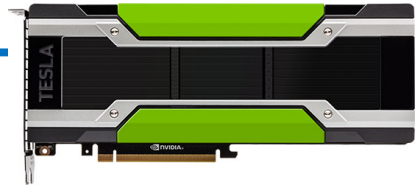
100G IB/EDR

Xeon Broadwell

QPI

Xeon Broadwell





GPU: coarse-grain offloading
NVIDIA P100 x 2

40Gb Ether x 2

FPGA:
fine-grain
partial offloading
+
high-speed interconnect

Summary and Issues

- Multi-hetero environment for multi-level complexity, multi-physics simulation
- Issues
  - How to configure the system ? What interface ?
    - PCIe (gen3 -> gen4): performance bottleneck
    - Intel HARP (Xeon CPU + Altera FPGA): no extension for external link
    - OpenCAPI by OpenPower -> NVLINK2: CPU & GPU is limited
  - How to program ?
    - OpenCL for higher level (application) code + HDL for lower level (fundamental modules), but how to coordinate ?
    - Partial reconfiguration technology to save the debugging TAT