Pre-PACS-X
計算科学研究センターの次期演算加速スパコンに向けて

朴　泰祐
筑波大学計算科学研究センター
HPC研究部門主任／次世代計算システム開発室長／副センター長
http://www.hpcs.cs.tsukuba.ac.jp/~taisuke/

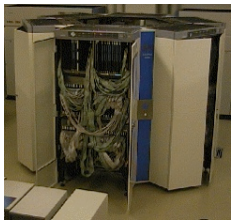
超並列計算機PAX (PACS)の開発の歴史

- 1977年に研究開始（星野・川合）
- 1978年に第一号機が完成
- 1996年のCP-PACSはTOP500第一位

1978
第1号機PACS-9

1980
第2号機PAXS-32

1989
第5号機QCDPAX



1996
世界最高速を達成した
第6号機CP-PACS

2006
バンド幅重視クラスタ
PACS-CS

2012~2013
GPU演算加速クラスタ
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

- 計算科学者＋計算機工学者の *co-design* による「実行性能重視スパコン」
- Application-drivenな開発
- 持続的な開発による経験の蓄積

この他：科研費特別推進研究によるハイブリッドクラスタ FIRST

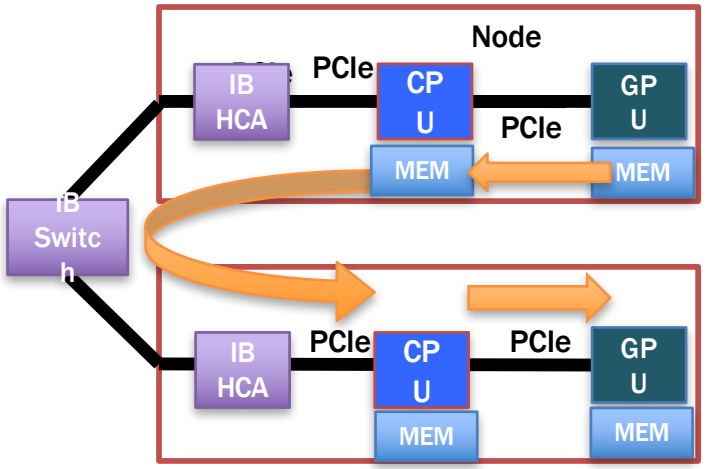
CCSにおいて現在運用中の2系列のスパコン

- HA-PACS (PACS-VIII)
  - GPU cluster、一部に独自開発のGPU間直接通信ネットワーク(TCA)を実装
  - PFLOPSクラスのGPUクラスタにより accelerated computing によるコード開発とプロダクトランを実施
  - novice user よりも professional user をターゲットとした先進的マシン

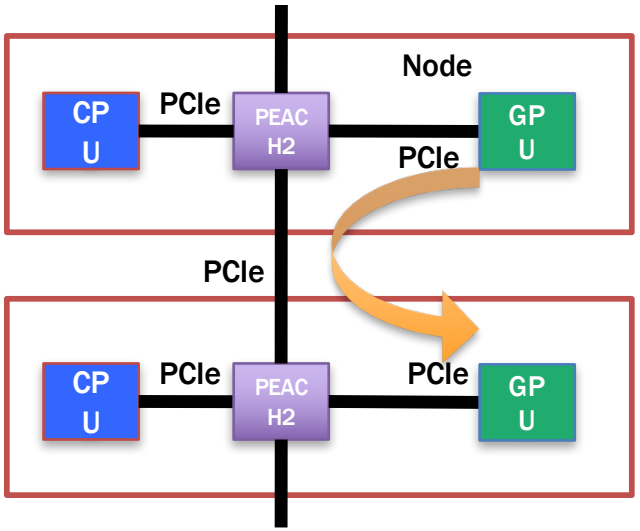
- COMA (PACS-IX)
  - Many Core cluster
  - PFLOPSクラスの many core architecture クラスタにより many core processor の性能特性を理解しつつ次世代の many core 向けコード開発とプロダクトランを実施
  - 筑波大・東大で共同設置した「最先端共同HPC基盤施設 (JCAHPC)」の新スパコン（OFP）における many core CPU利用の先行研究と各種テスト実施
  - 汎用並列処理向け、比較的広いユーザ層を対象
    ➟ 現在、MICは一種の演算加速器だがその本質は汎用プロセッサ

TCA: 独自開発の通信機構

- TCA (Tightly Coupled Accelerators)
  - PCIe によるAD間直接通信（ノード内・ノード間）を実現
  - PEACH2チップ（FPGAによるプロトタイプ）によるインテリジェントなPCIeスイッチ＋コントローラ
  - ホストCPU・メモリ・結合網に依存しないアクセラレータ間直接通信

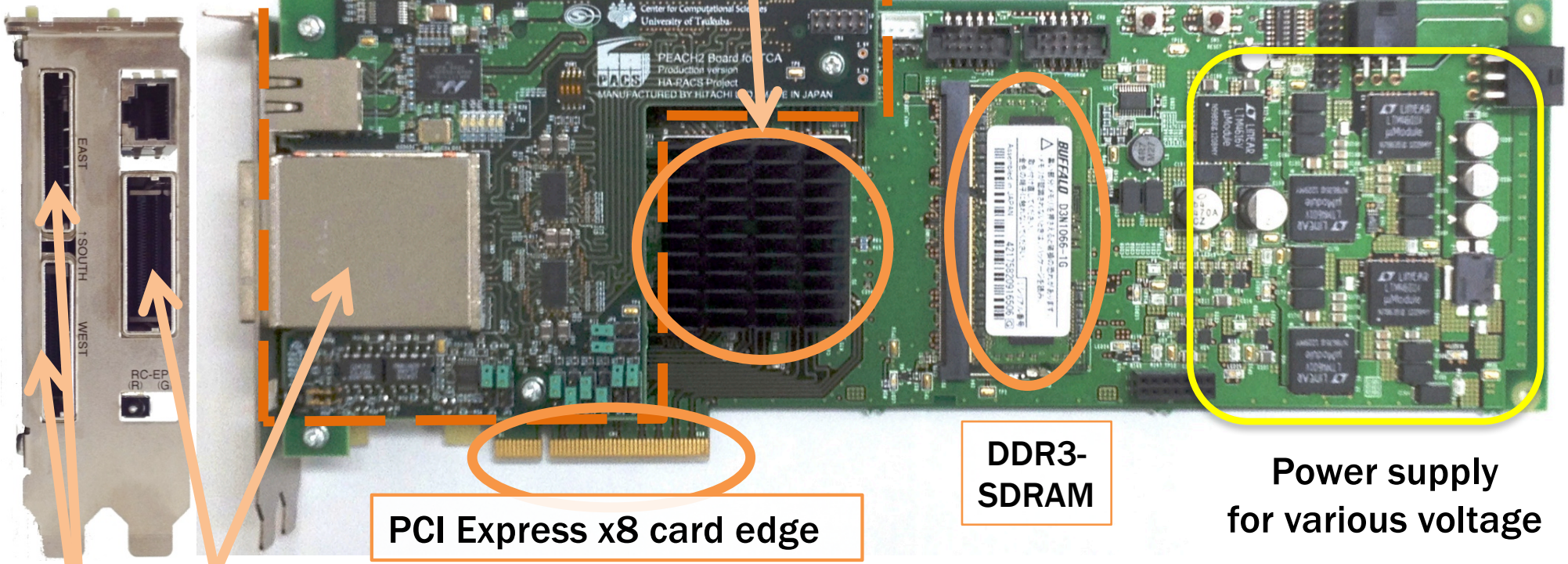


通常のノード間GPU間通信



TCAによるノード間GPU間通信

PEACH2 board

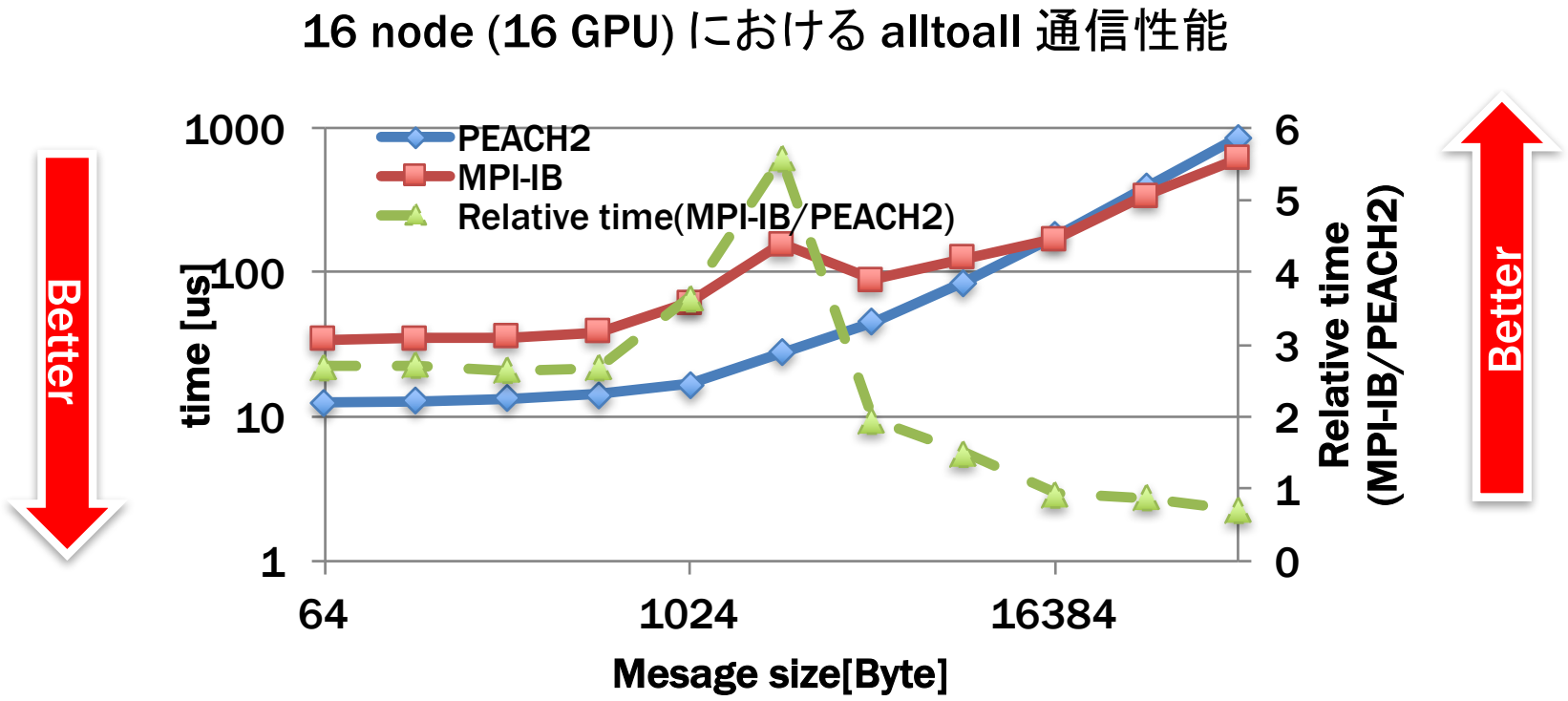


HA-PACS Base Cluster + TCA

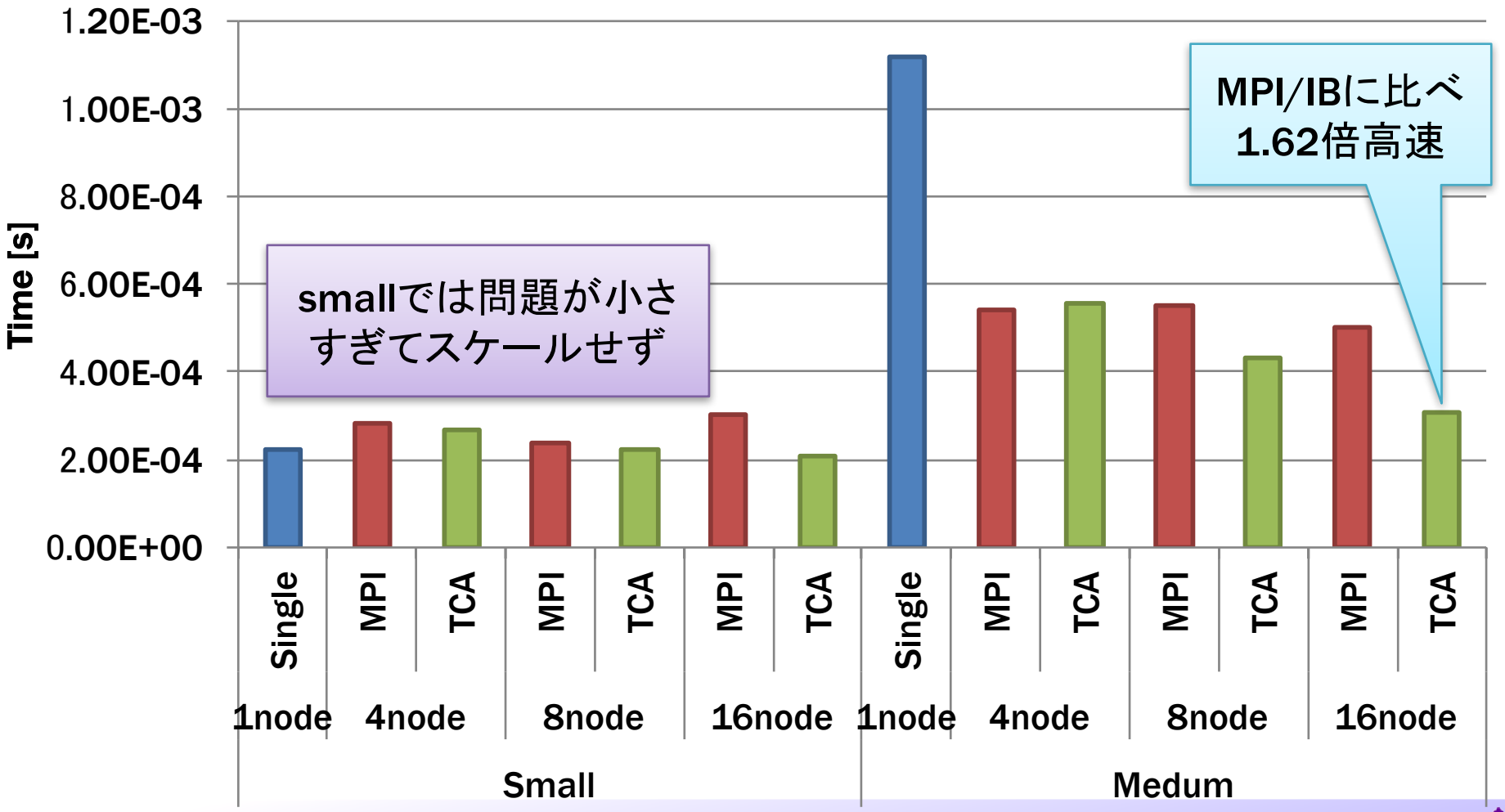
- HA-PACS Base Cluster = 2.99 TFlops x 268 node = 802 TFlops
- HA-PACS/TCA = 5.69 TFlops x 64 node = 364 TFlops (Green500 #3)
- TOTAL: 1.166 PFlops

FFTEベンチマーク：alltoall通信

- 高橋@筑波大のFFTE (6 step FFT)のPCクラスタ版をTCAに対応させたベンチマーク
- TCAによるalltoall通信をDMA chainingと同一GPU内のデータコピーにもcudaMemcpyAsync()ではなくPEACH2を使うことにより、GPU内データ移動をより高速化



FFTEの性能 (Small=2^{14}, Medium=2^{16})



今後の演算加速器系システム

- MICは過渡的に演算加速器であるが基本的にはx86互換の汎用プロセッサ
  - メニーコアであるが各コアは比較的非力であり、浮動小数点SIMD命令の強化でピーク性能を上げている
  - Programmability の点では優れているが、実効性能を上げるにはかなり工夫が必要
  - KNL (Knights Landing) からはメインプロセッサに⇒OFP
- 依然として演算加速器の本命はGPU
  - NVIDIAを中心としてGPUの高性能化が続いている
  - 電力は徐々に上がっているがFLOPS/Wは下がり続けている
  - これだけで十分か？

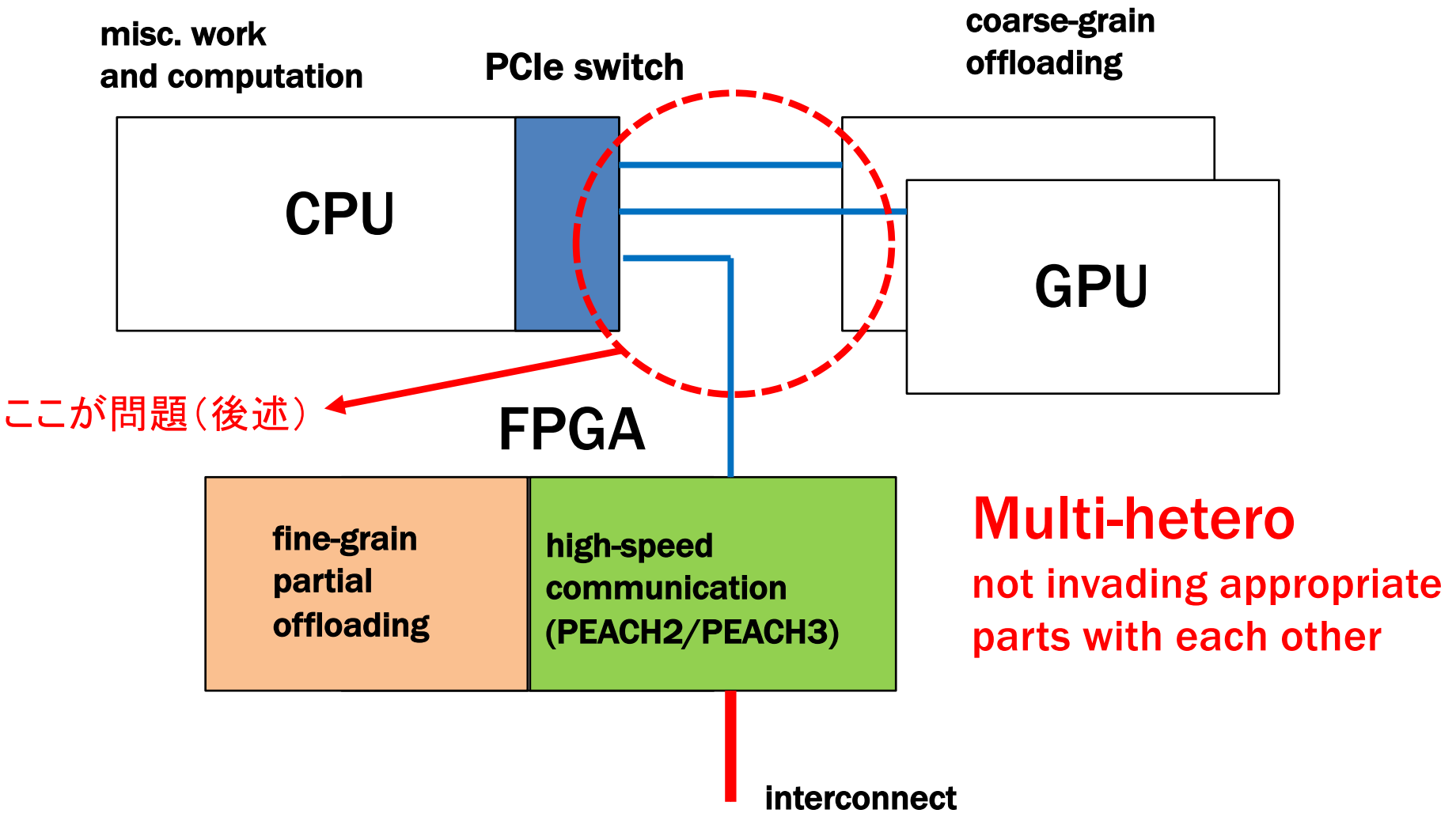
Accelerator in Switch (Network)

- GPUクラスタだけでよいか？
  - GPUが持つ高速性と電力性能比は重要
  - GPUは完璧ではない
    - 高SIMD・高並列・単純なバルク並列処理に適している
- GPUの非柔軟性・CPUの低速性を埋める
  - GPUの高速性とCPUの柔軟性のブリッジが必要
  - 加えてstrong-scalingに要求される低レイテンシ通信
  - アプリケーション/アルゴリズムに依存する様々な精度
  - FPGAによる解決が有効（TCAの経験）

➡ **Accelerator in Switch (Network)**

(Post-Peta CREST: “ポストペタ時代に向けた演算加速機構・通信機構統合システムの研究開発”,
代表：朴泰祐で実施中）

Accelerator in Switch のノード概念図



Multi-hetero

not invading appropriate parts with each other

アプリケーションFPGAオフローディング（宇宙物理部門＋慶應義塾大学との共同研究）

PEACH2によるオフローディング：宇宙物理学の重力計算において、LET(Locally Essential Tree)を on-the-flyで作成

- 幅優先探索によるTree code on GPU を実装
  - 扇谷コード，中里コードよりも高速
  - まだ高速化の余地がある
- 現在実装中の内容
  - GPU 版 tree make
  - 近傍粒子探索＋PH-key jump の動的決定
- 並列版はもう少し調整が必要
  - 計算時間の比を用いて各GPUに割り当てる粒子数を設定しているが，MPIの実行時間だけをうまく省かないといけない

LET (Locally Essential Tree) 作成

mask[id]配列を用意

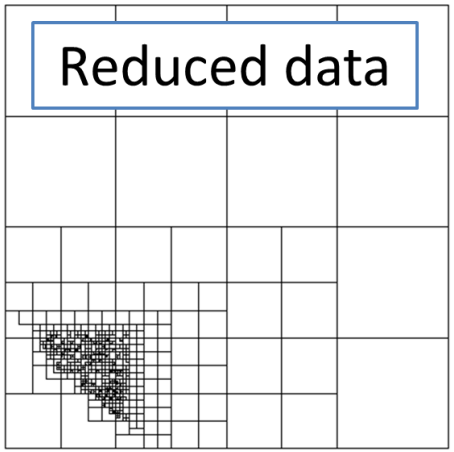
mask[id] == 0間引く

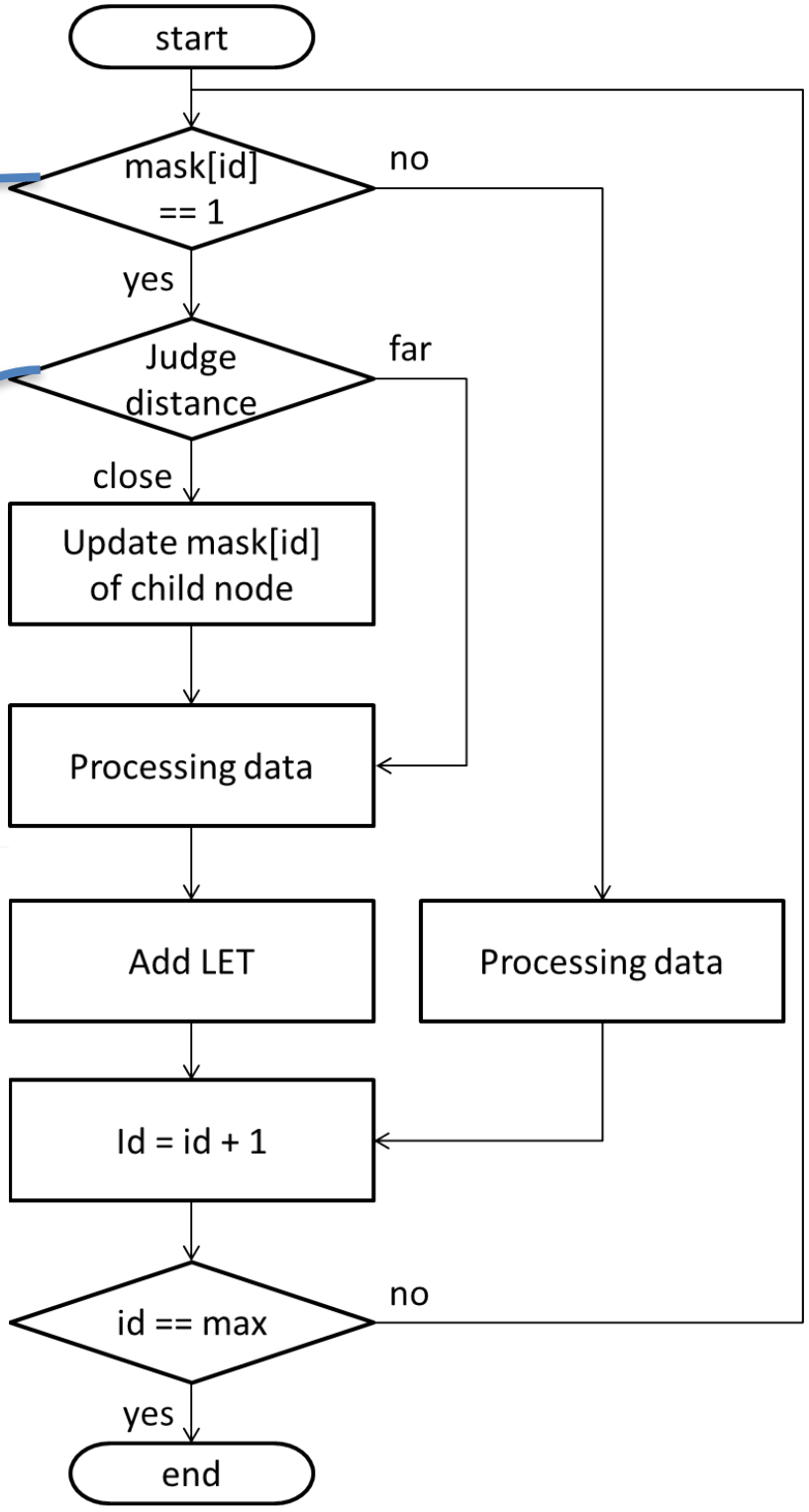
mask[id] == 1LETに追加

距離判定

受信側の領域データの一部と

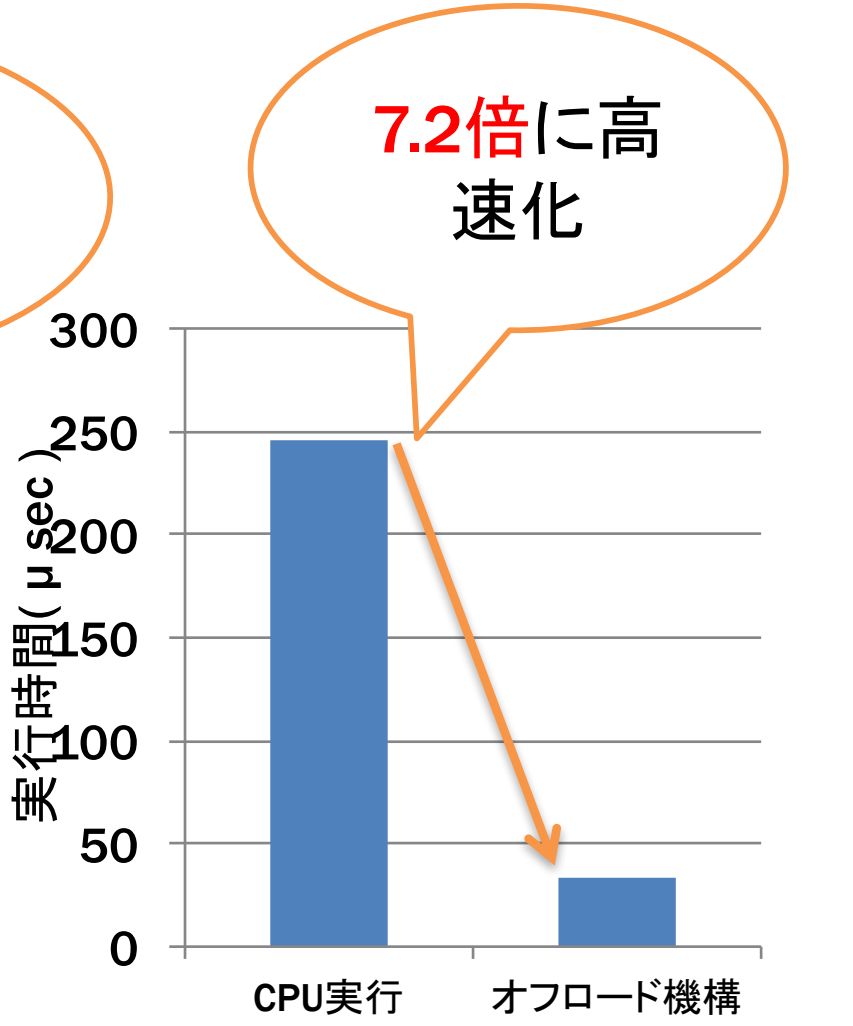
送信側の各セルにて距離判定





性能評価

LET作成部分の実行時間

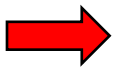


LET作成からGPUへ
送信までの実行時間 (参考値)

次世代の演算加速システムを目指して

- 新しい演算加速システム
  - GPU + FPGA + short latency network
  - JST-CRESTにおける研究等に基づきFPGA partial-offloadingを検討中

    ⇒ Accelerator in Switch
  - PEACH2は PCIe 技術に基づいていた
    - GPU, network の全てを１つのデータ移動・通信系で統合できるよいアイデアであった
    - しかし技術的制約（アドレス空間・デバイス数上限、デバイスの世代交代）により優位性が低下
  - FPGAに統合可能な新しい通信技術と演算処理の統合が必要



PACS-X (10th generation of PACS)

PACS-X [ten] 計画

- 第10世代のPACS
  - TCAの概念を発展させ演算・通信融合を加速させる
  - 最先端のGPUとFPGAを導入
  - PEACH2/PEACH3によるPCIe依存の統合システムから発展し、さらに高速・汎用・拡張性のある通信系を採用
- PACS-X
  - 2016年度に調査研究費が文科省より認められる（2700万円）
  - 2018年度を目処に本格システムの実装を目指す
  - 目標：10PFLOPS程度
- Pre-PACS-X (PPX)：予備実験機
  - PACS-X調査研究費に基づき予備実験装置を導入
  - 2017年3月より稼働開始予定

PPXの計算ノード

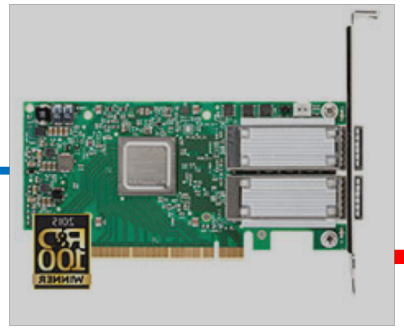
448 GFLOPS
64GiByte
76.8GB/s

Xeon
E5-2660 v4

QPI

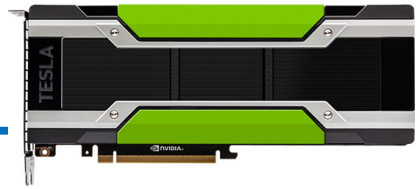
Xeon
E5-2660 v4

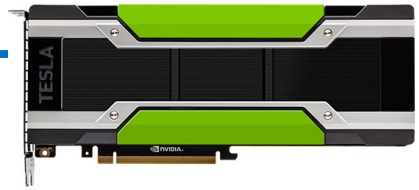
448 GFLOPS
64GiByte
76.8GB/s



HCA:
Mellanox IB/EDR

100Gbps IB/EDR



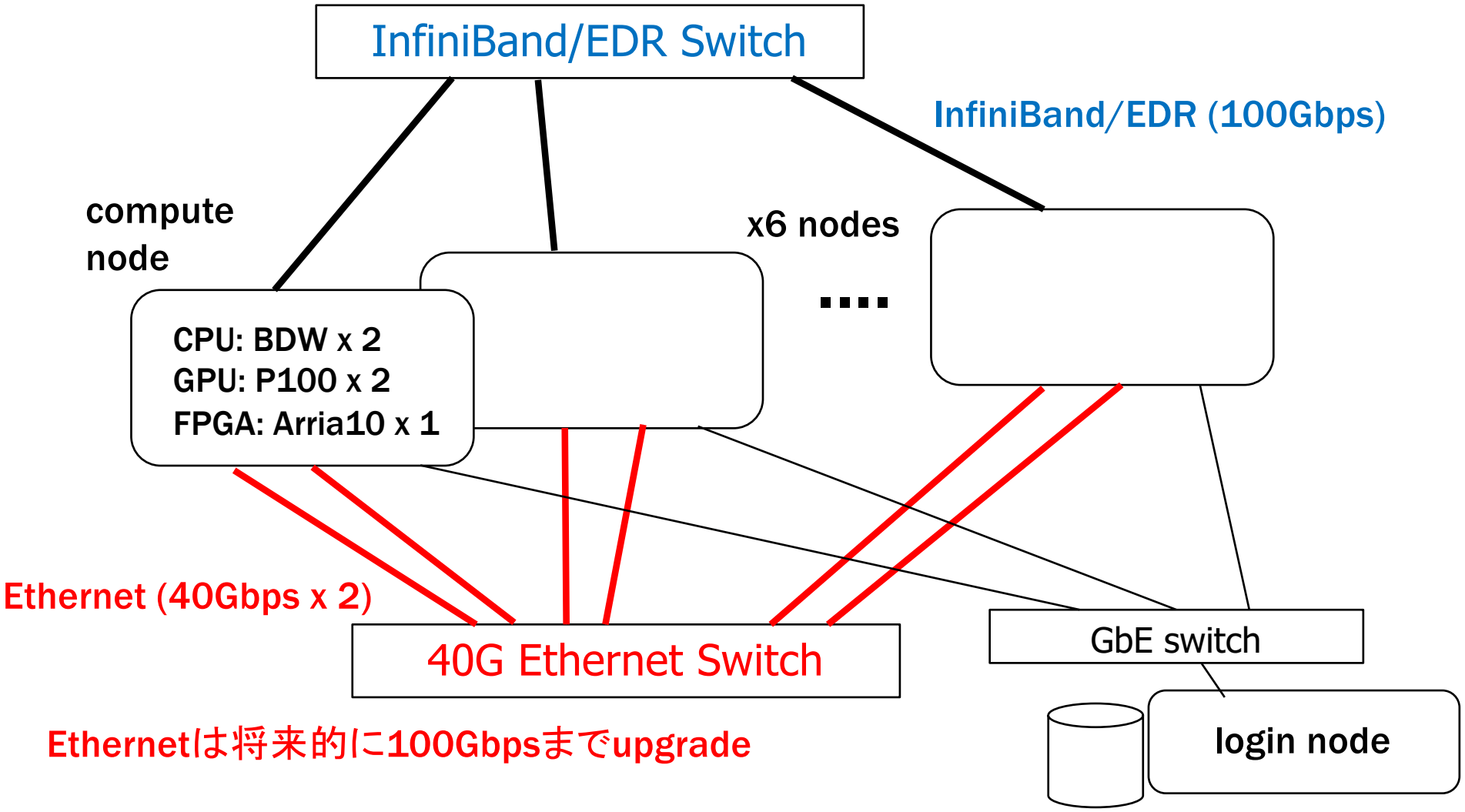


GPU: coarse-grain
offloading
NVIDIA P100 x 2
(4.7TFLOPS x2
16GiByte x 2
720GB/s x 2)

FPGA:
fine-grain
partial offloading
+
high-speed interconnect
Bitware: A10PL4
(Altera Arria10)

40Gb Ether x 2

PPXミニクラスタシステム



ターゲットアプリケーション例：輻射輸送

- 天体から放射される光（輻射）と物質の相互作用

  - 原子の電離
  - 分子の光解離
  - ガスの光加熱
  - 輻射圧によるガスの運動の駆動

様々な天体形成にとって重要な物理過程

- 恒星の形成
- 銀河から電離光子による宇宙の再電離

- ray tracing法による数値シミュレーション
  - 最もversatileな手法
  - 様々な状況で正確な結果を与えるが、計算コストが膨大
  - これまでは近似的な代替手法(拡散近似)による計算がほとんど

（提供：
吉川耕司講師）

輻射輸送

- 点光源からの輻射輸送

光源から各メッシュへ光線

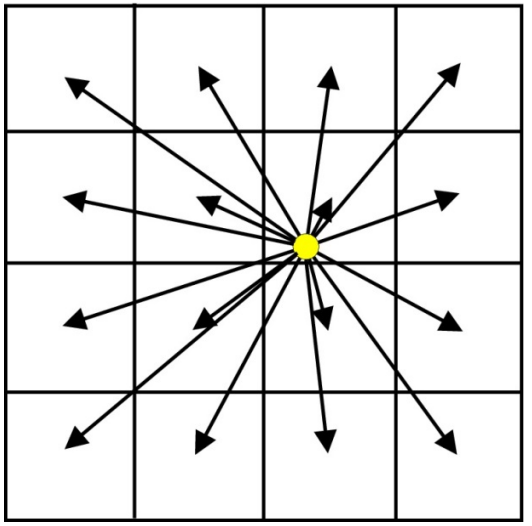
$$\frac{dI(\nu)}{d\tau(\nu)} = -I(\nu)$$

ν：振動数

$I(\nu)$：輻射強度

$\tau(\nu)$：光学的厚み

計算コスト: $\propto N_{\mathrm{m}} N_{\mathrm{s}}$



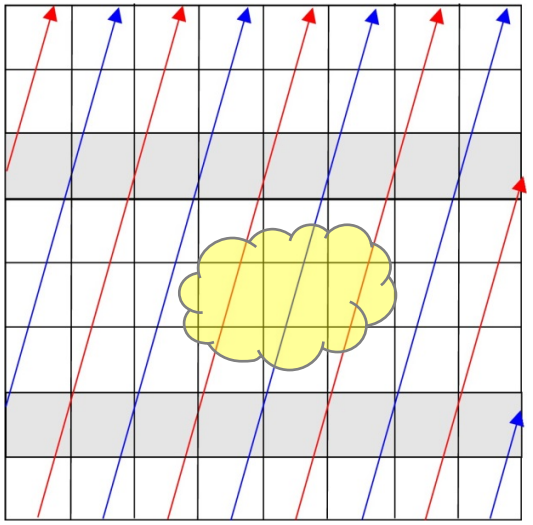
- 空間的に広がった光源からの輻射(diffuse radiation)の輸送

境界から多数の平行な光線

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

$S(\nu)$：source function

計算コスト: $\propto N_{\mathrm{m}}^{5/3}$



（提供：
吉川耕司講師）

輻射流体シミュレーションコード・ARGOT

（Tはサイレント）

- 輻射輸送と流体力学を同時に解くコード
- 点光源と広がった光源からの輻射輸送をどちらも扱うことが出来る。

  点光源からの輻射輸送：ARGOT法

  広がった光源からの輻射輸送：ART法
- ARGOT法：多数の点光源がある場合にTree構造を使って光源を束ねて計算。

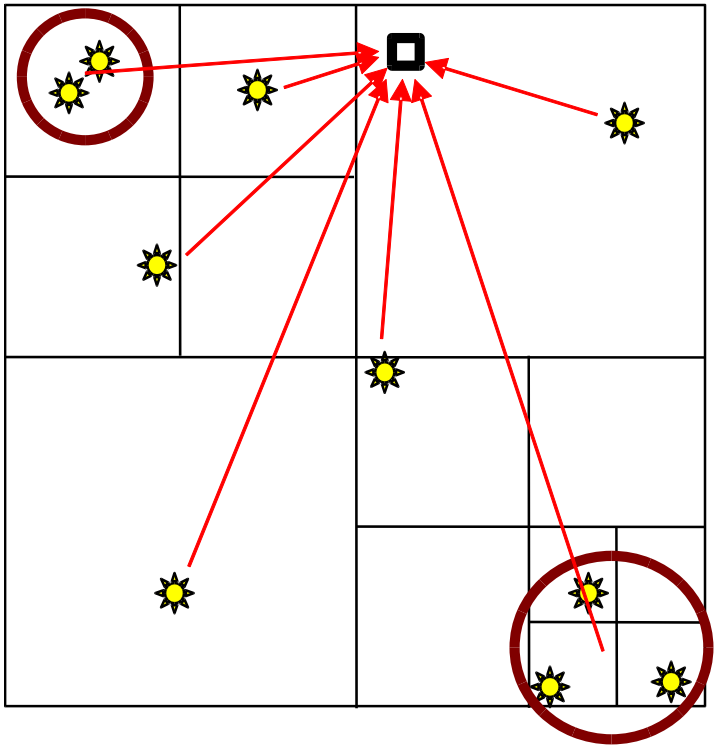
計算コスト $\propto N_{\mathrm{m}} N_{\mathrm{s}} \Longrightarrow \propto N_{\mathrm{m}} \log N_{\mathrm{s}}$

- 輻射輸送計算はCUDA、OpenMPでスレッド並列化
- MPIでノード並列化
- https://bitbucket.org/kohji/argot

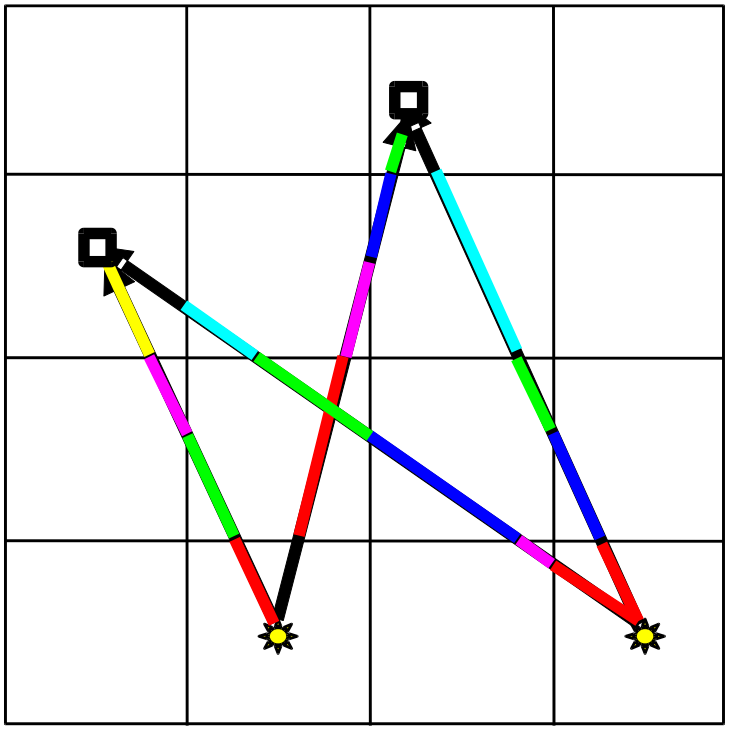
（提供：
吉川耕司講師）

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

ARGOT法



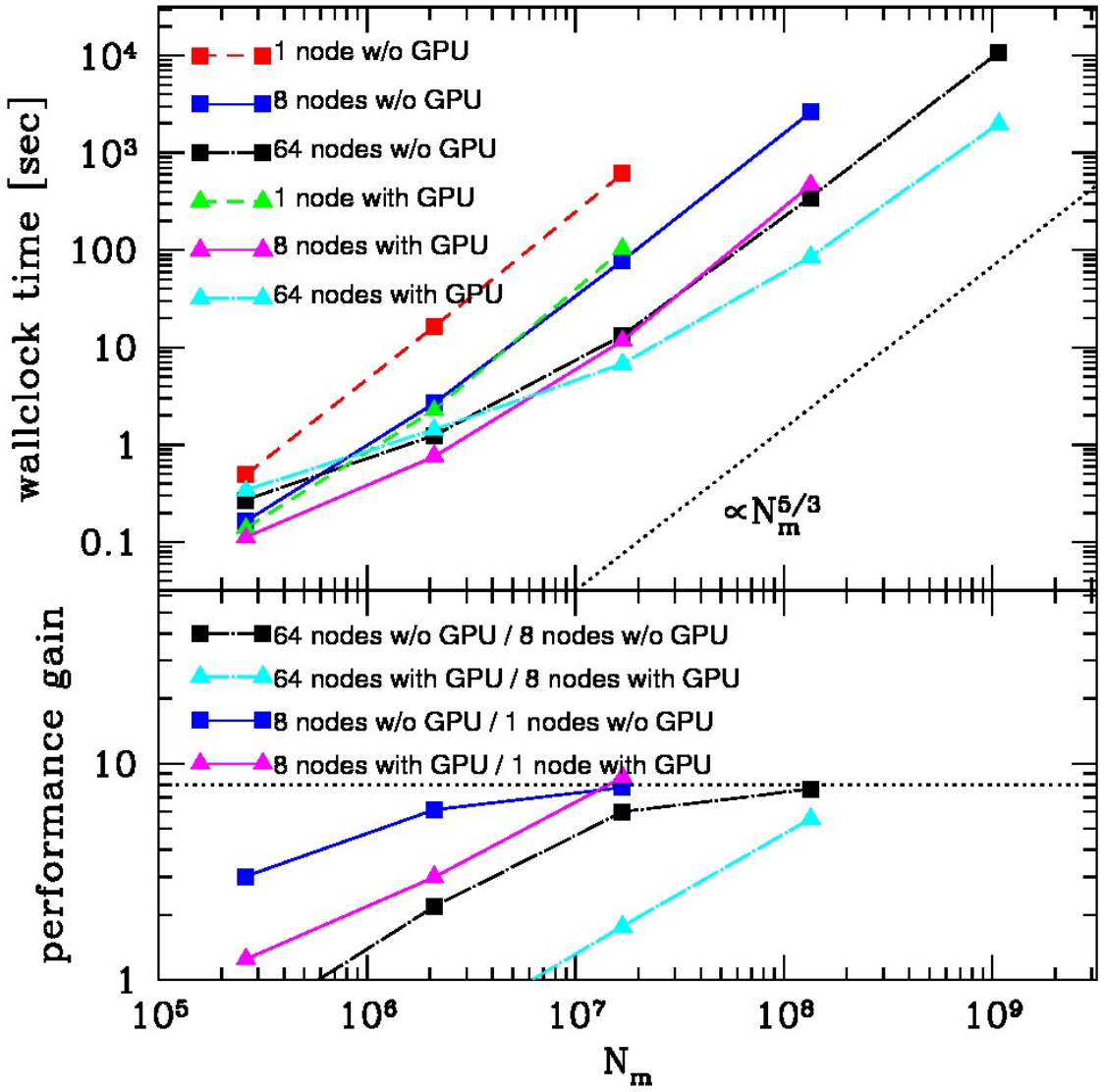
ARGOT法の並列化



LETの実装で行ったツリー法計算とノード間通信と基本的に同じ構造

（提供：
吉川耕司講師）

マルチノードでの並列化



- 64^3メッシュ/node 以上でスケールする
- GPUを使うと128^3メッシュ /node 以上でしかスケールしなくなる。
- 光線がノード境界をまたぐときのノード間通信が原因

Accelerator in Switch で解決

（提供：
吉川耕司講師）

PPXの技術的課題

- ノード内の各要素の結合
  - CPU, GPU, FPGAを結合しているのは依然としてPCIe
  - FPGAは gen3x8 lane で一番貧弱
  - FPGAに offloading される計算は 40GEx2 によって高速通信可能
    ⇒ GPUは IB/EDR とし、これらをどう連携させるか
- プログラミング
  - GPUにはOpenACC or CUDAが有効
  - FPGA: HDL + OpenCL ?
    - OpenCLは性能、記述性ともにまだ不完全
    - 特に高速化が要求されるモジュールは Verilog HDL 等で記述し、OpenCLとの合成を
    - デバッグのTATを短縮させるための partial reconfiguration
  - 全体を統合するシステムソフトウェアが必要

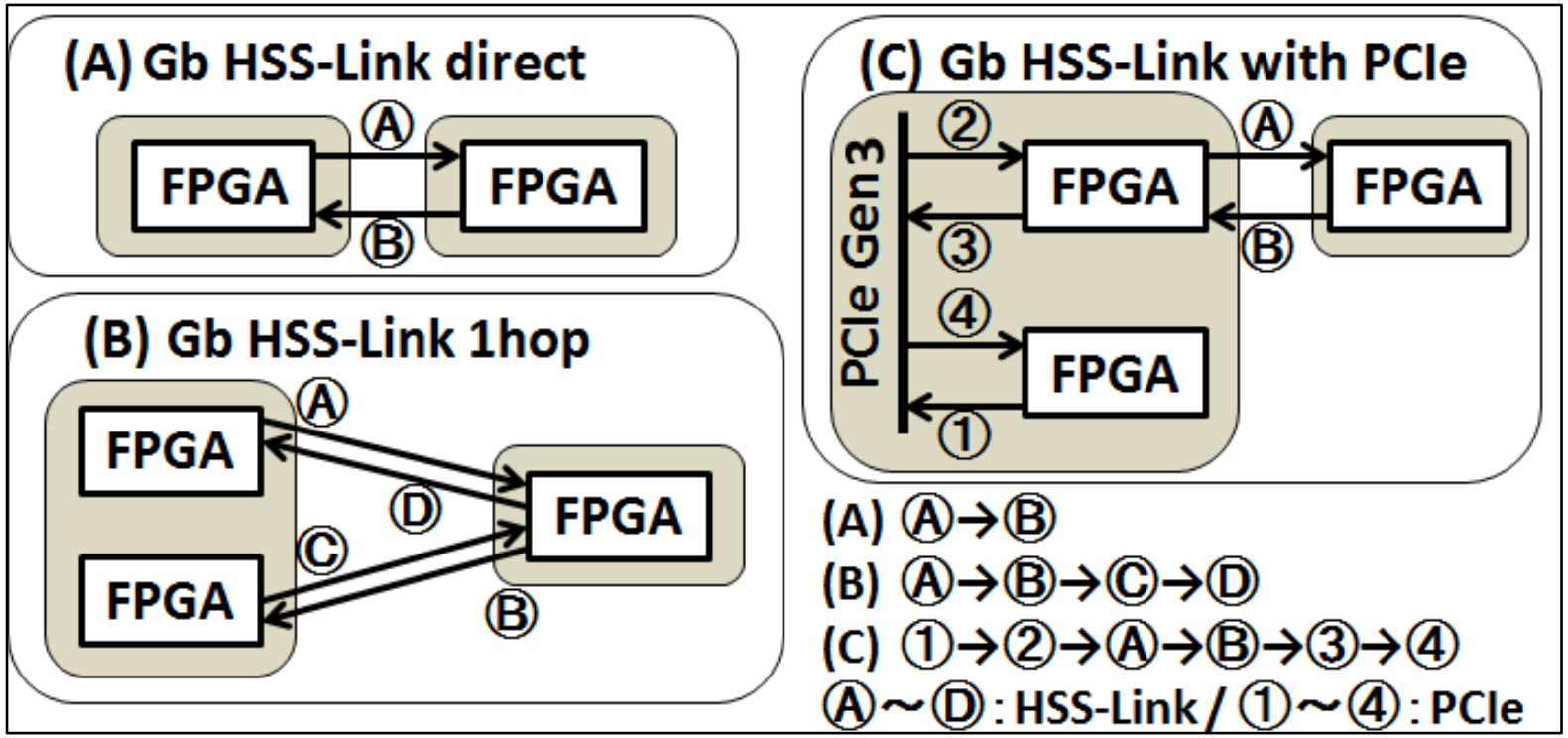
ノード内のCPU・GPU・FPGA間の結合の課題

- 方法１：PCIe による結合
  - 今後、gen3からgen4への移行が進む予定、PCIe gen4 x16 lanesでは ~30GB/s
  - GPU、FPGAがちゃんと着いてくるか？
- 方法２： IntelによるXeon+FPGA
  - QPIによってFPGAをCPUのようにcoherent busでXeon CPUと接続
  - MCMで一体化されるがFPGAのI/Oが全くない⇒GPUはどうする？
- 方法３： NVLINK2による結合
  - OpenPOWERに基づき、POWER9とVoltaではNVLINK2で結合される予定、共有メモリ、キャッシュコヒーレンシをサポート
  - FPGAへの直接接続はどうなる？
    ⇒ NVIDIAがNVLINK2 IP for FPGAを提供すれば...
- 方法４： OpenCAPI in OpenPOWER
  - POWER CPUにおけるQPIのようなもの
  - FPGA上のIPを提供

FPGA間接続ネットワーク

- 40G Ethernetは既に利用可能
- 100Gbpsの光インターコネクトがXilinxより提供されている
- PEACH2/PEACH3のPCIe接続からFPGA光インターコネクトへの切り替え
- 100Gbpsインターコネクトの実験を開始
  （共同研究員：山口佳樹准教授との共同研究）

100Gbps光インターコネクトの実験



Xilinx XC7VX1140T(Virtex7)

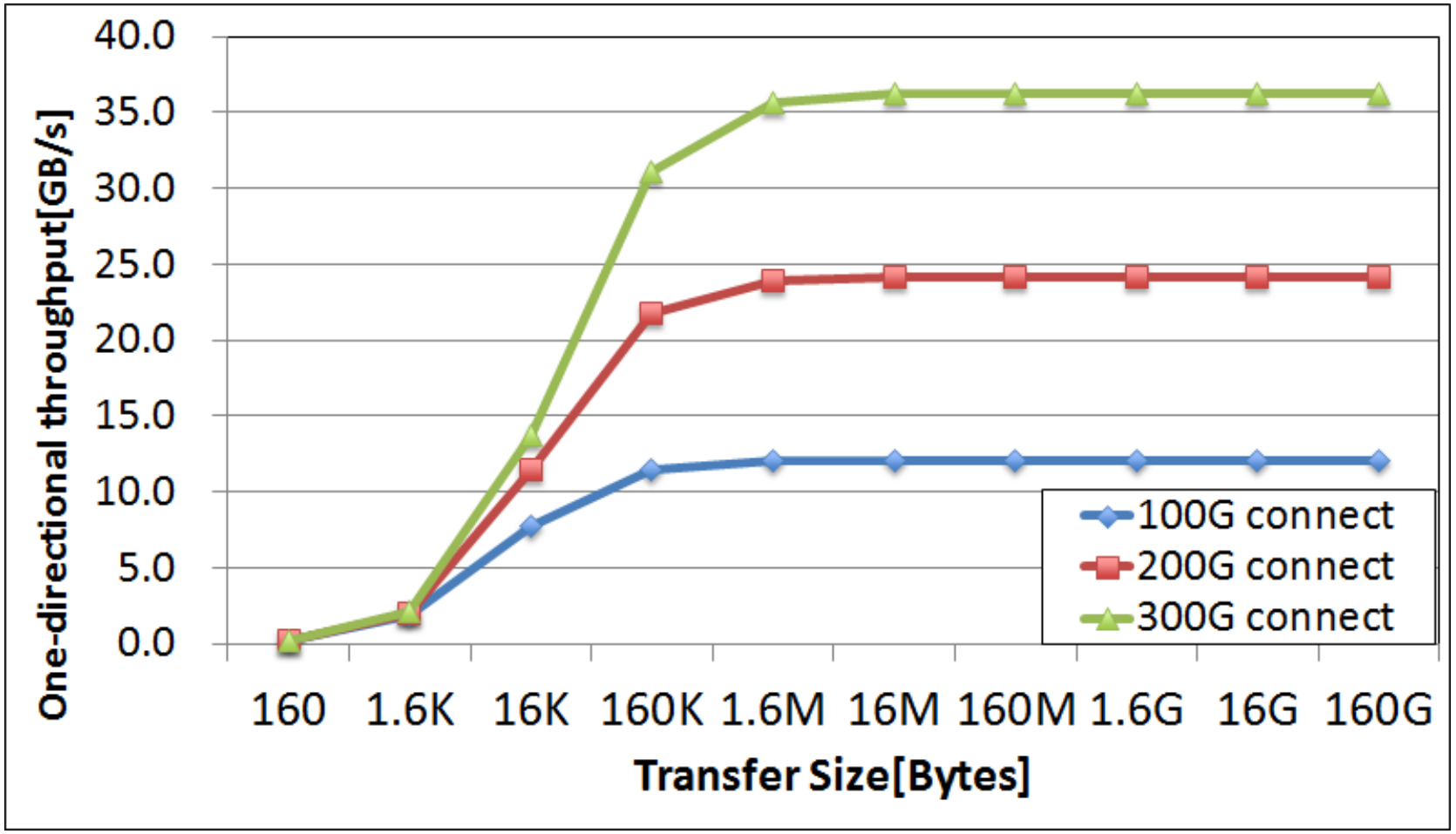
Vivado 2016.1

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

(Aurora 64B/66B v11.1)

(高山・山口・朴 FIT2016)

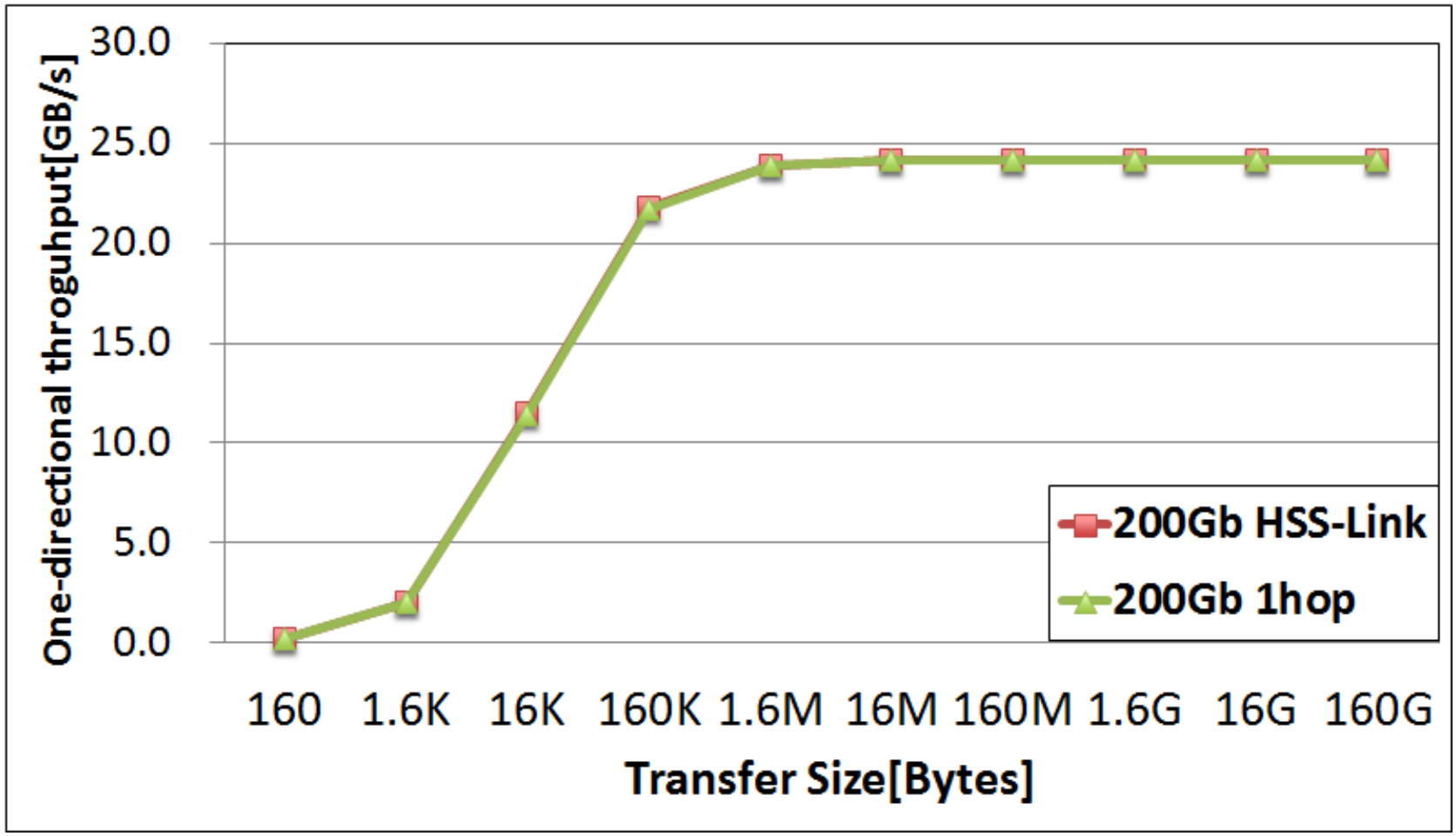
Case-A: peer-to-peer



- 理論ピーク性能の96%まで出ている
- 3 channelでも良いスケーラビリティ

(高山・山口・朴 FIT2016)

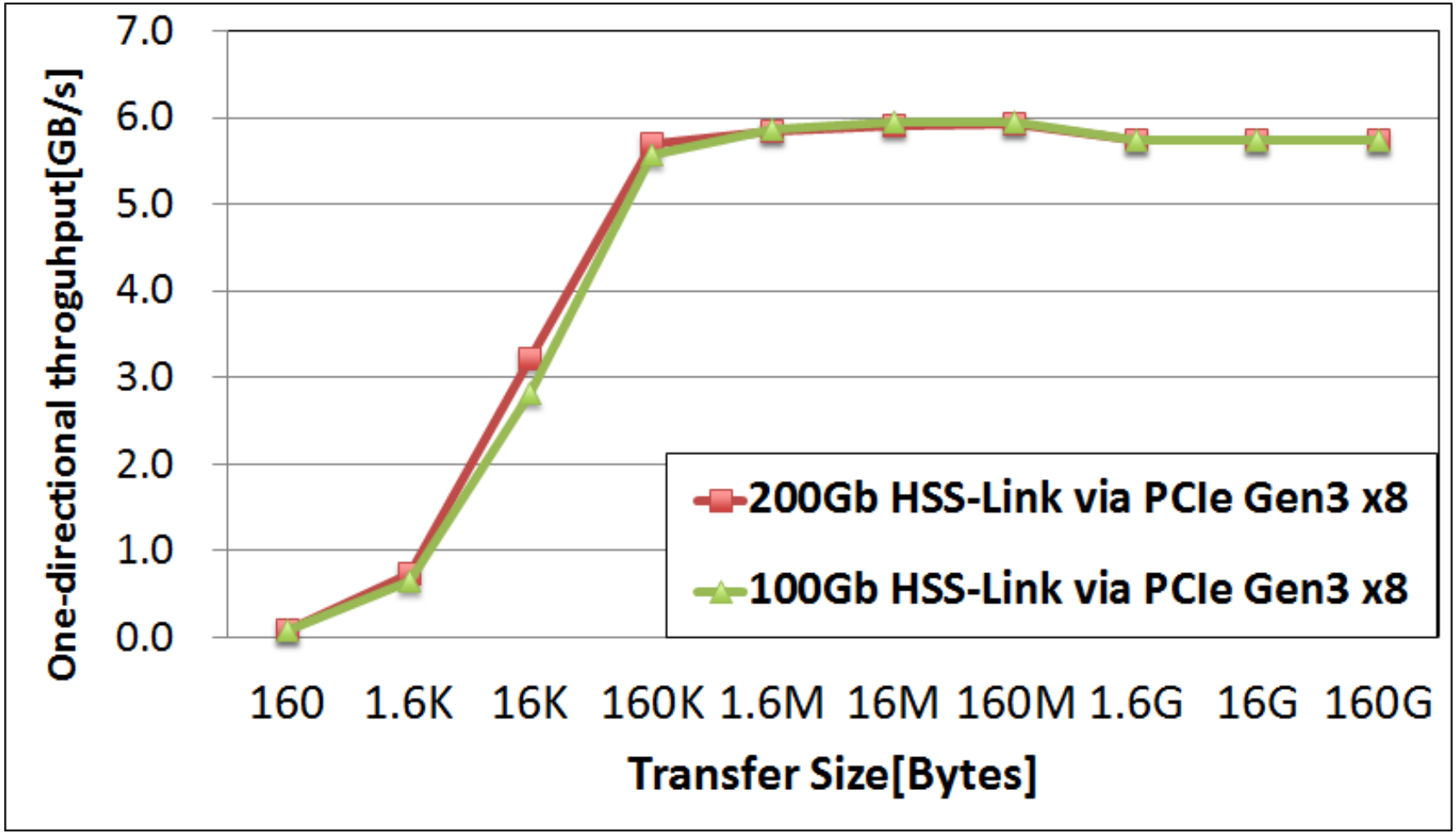
Case-B: 1-hop routing via FPGA



- FPGAのhop latencyは20ns程度

(高山・山口・朴 FIT2016)

Case-C: PCIe intra-communication



- PCIeのバンド幅がボトルネック
- 理論ピークの90%程度の性能は出ているが...

(高山・山口・朴 FIT2016)

FPGA外部リンク

- PPXではBitware A10PL4 (Arria10)を搭載するが、PCIeボードであるため別のFPGAボードに差し替え可能
- 他のArria10ボードやXilinxボードも対象とした実験を行う
- 外部リンクのIPのOpenCL等からの使いやすさも検討事項
- 課題
  - FPGA partial-offloadingする場合の性能とバンド幅が外部リンクのバンド幅と釣り合うか？
  - GPU間通信にも使えるか？
  - 有効利用するためのアプリケーションとアルゴリズム
  - 40G以上のスイッチは高額（Ethernet）
  - マルチホップ、n-D torusの可能性は？

まとめ

- 第10世代のPACSに向けて研究を始動
- Multi-hetero環境を想定した次世代の演算加速並列システムの骨格を作る
- アプリケーション開発はFPGAを含めるため困難
  ⇒ OpenCLなどの高位言語の利用とその使いやすさを検証
  ⇒ アプリケーションに対応するモジュールの実装
- PPX: Pre-PACS-Xによる各種実験を予定
- 2018年度を目処に実システムの導入を目指す