GPU・FPGA複合計算による次世代HPC/AI加速技術

朴泰祐
筑波大学・計算科学研究センター
taisuke@ccs.tsukuba.ac.jp

（共同研究：小林諒平・藤田典久・山口佳樹・梅村雅之・吉川耕司）

HPCにおけるアクセラレータ利用

- 近年のHPC (High Performance Computing) 分野ではアクセラレータの利用が増えている
  - Top500(2021/6)おける上位10マシン中7マシンが搭載している
  - 特にGPUの利用が主流
- 主流であるGPUの特徴
  - 高い並列演算性能とメモリバンド幅をもつ
  - しかし並列性が十分でないと性能がでない
    条件分岐，並列性の不足，ノード間通信
- そこでFPGAの利用
  - アプリケーションに特化した回路
    パイプライン並列
  - ハイエンドなFPGAは高速なノード間通信性能をもつ
  - 高位合成環境が発達

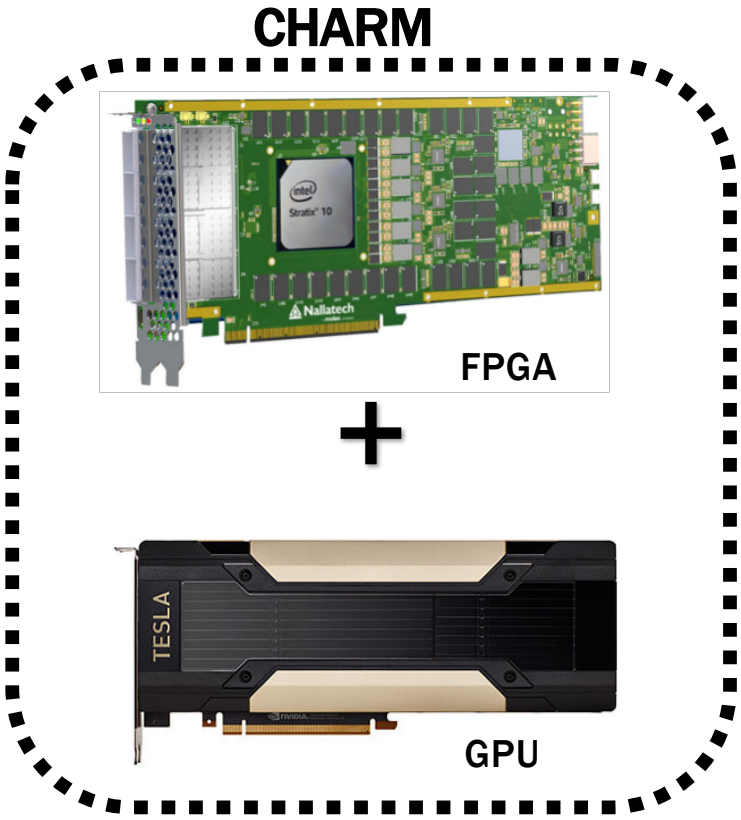
	GPU	FPGA
理論性能 (FLOPS)	◎	○
並列性	空間並列	パイプライン並列
ノード間通信性能	△	◎

GPU vs FPGA as HPC solutions

device	GPU	FPGA
parallelization	SIMD (x multi-group)	pipeline (x multi-group)
standard FLOPS	😀😀 (1000x cores)	😀 (~100x pipeline)
conditional branch	😢 (warp divergence)	😀 (both direction)
memory	😀😀 (HBM2e)	😢 (DDR)⟶😀 (HBM2)
interconnect	😢 (via host facility)	😀😀 (own optical links)
programming	😀 (CUDA, OpenACC, OpenMP)	😢 (HDL)⟶😉 (HLS)
self-controllability	😢 (slave device of host CPU)	😀 (autonomic)
HPC applications	😀 (various fields)	😢 (not much)

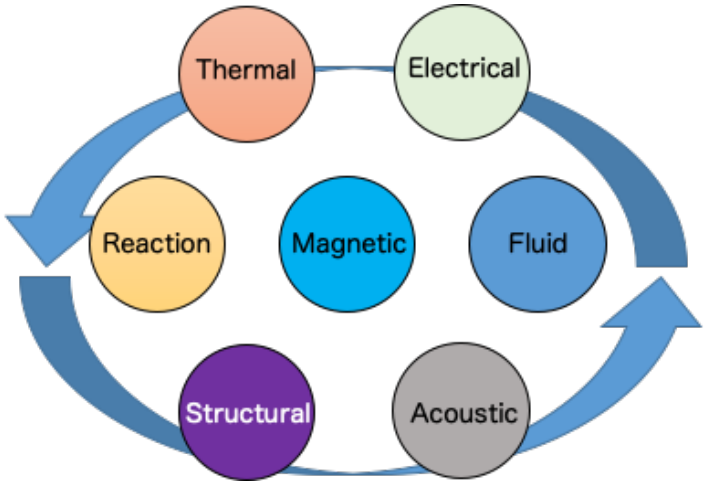
GPU・FPGAヘテロ演算加速プログラム

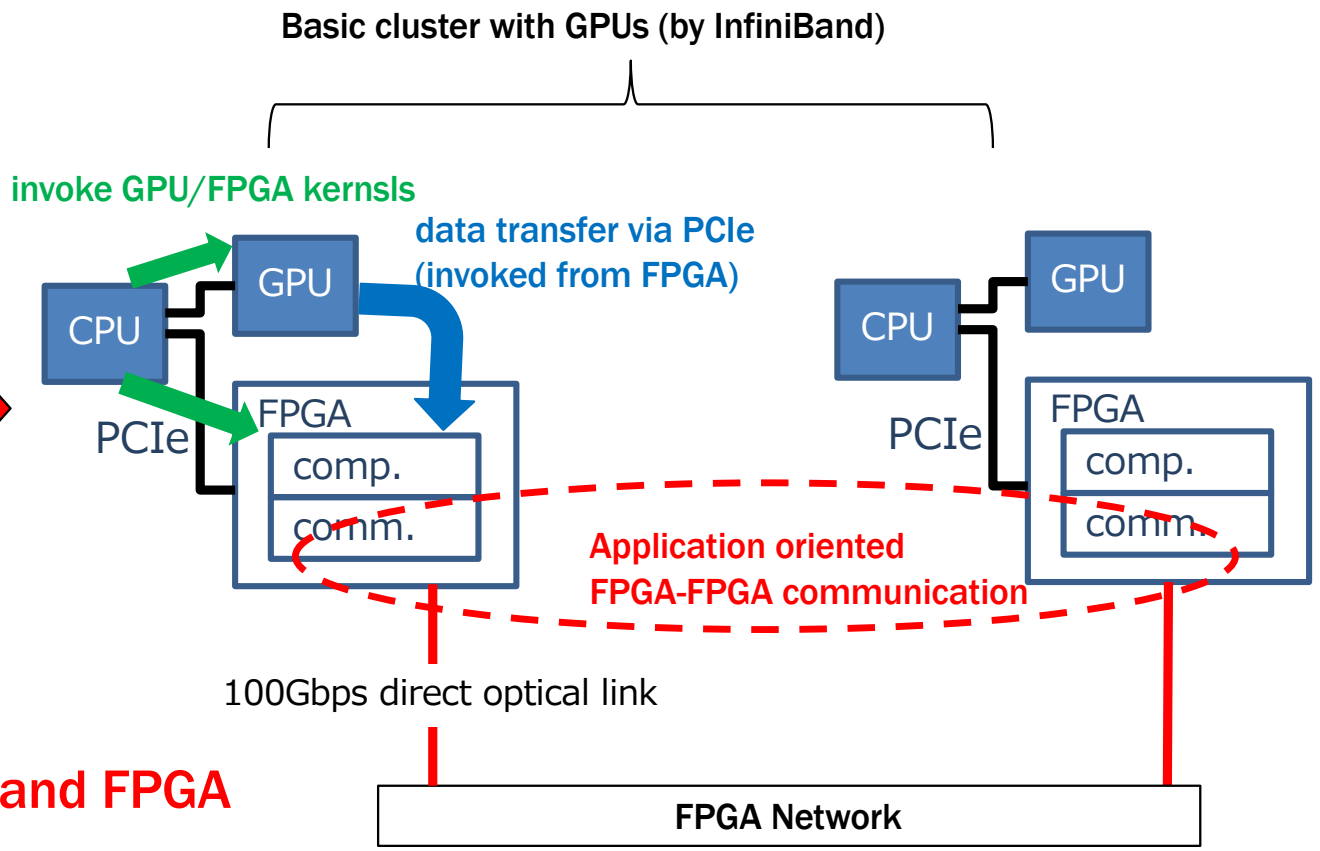
- GPUとFPGAを相補的に用いることで高速化をめざす
  - CHARM (Cooperative Heterogeneous Acceleration with Reconfigurable Multidevices)
  - それぞれの強みを最大限利用
    strong scalingにおけるボトルネックを解消
  - マルチフィジックスシミュレーション
    複数の物理現象の相互作用を考慮したシミュレーションであり，様々な特性の演算が出現
- プログラミング手法が大きな課題
  - 高レベルFPGA記述
  - GPUとFPGAの協調計算を自然に記述
  - FPGAコンパイラが鍵



CHARM: Cooperative Heterogeneous Acceleration with Reconfigurable Multi-devices

multi-physics/multi-scale complicated problem

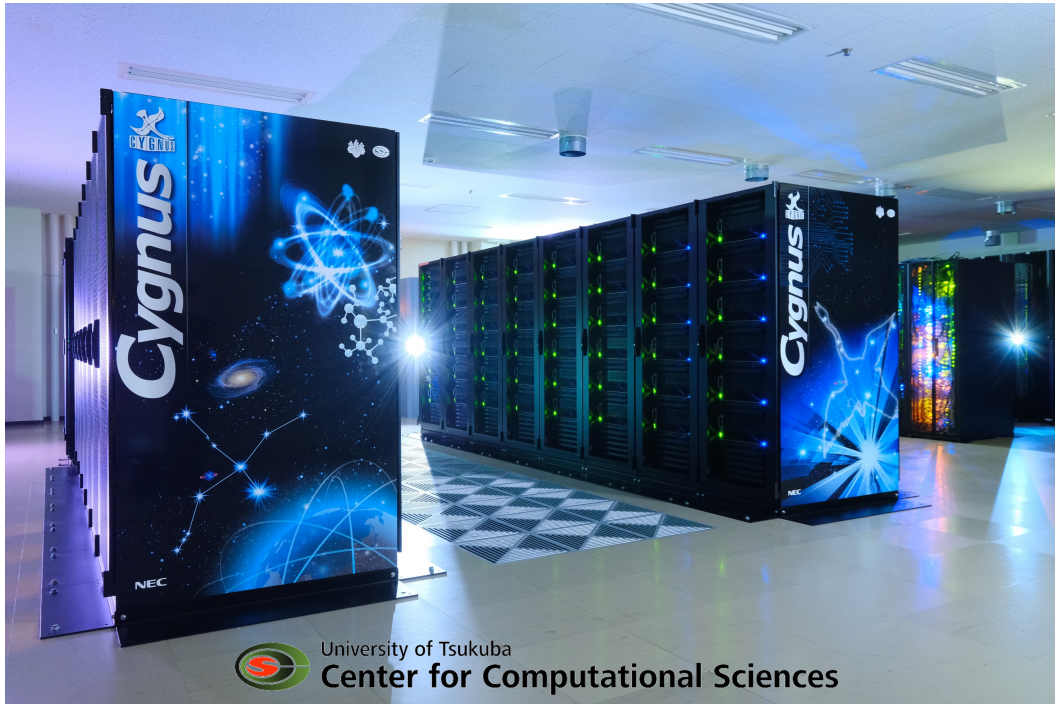




Cooperative computing with GPU and FPGA

Cygnus: world first multi-hybrid cluster with GPU+FPGA

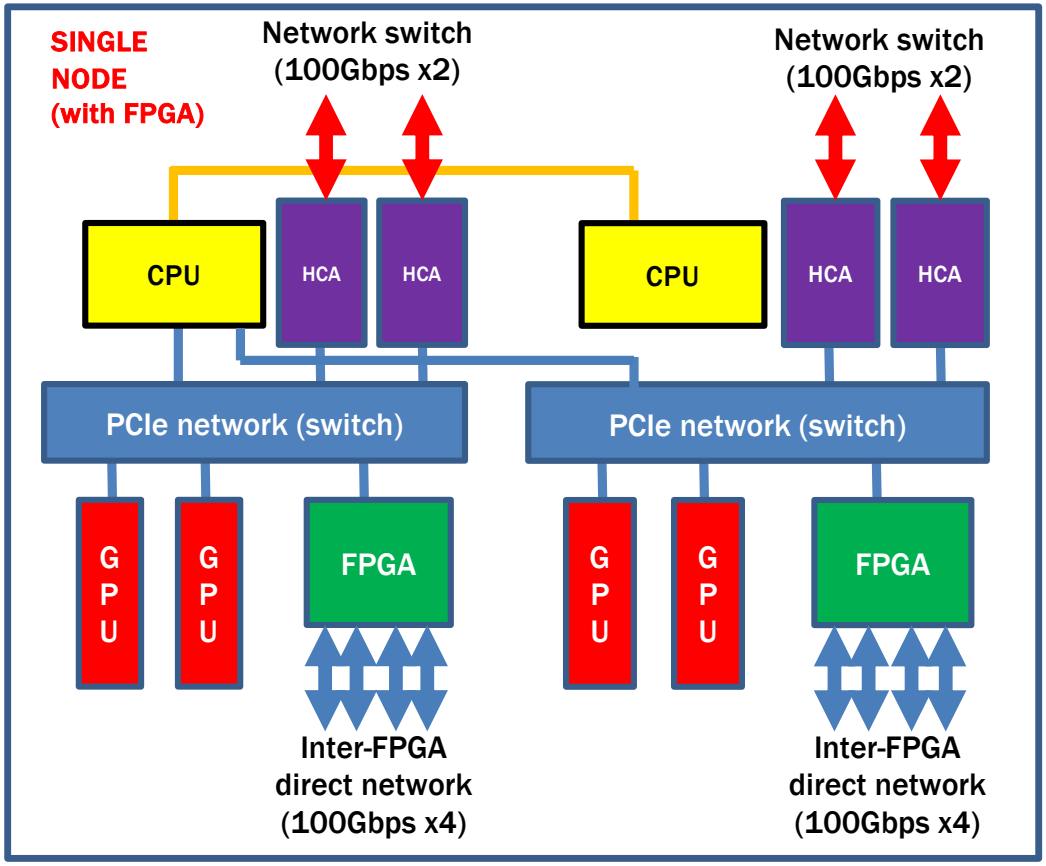
@ CCS, Univ. of Tsukuba (deployed by NEC)



Cygnus supercomputer at Center for Computational Sciences, Univ. of Tsukuba (Apr. 2019~)
85 nodes in total including 32 “Albireo” nodes with GPU+FPGA (other “Deneb” nodes have GPU only)

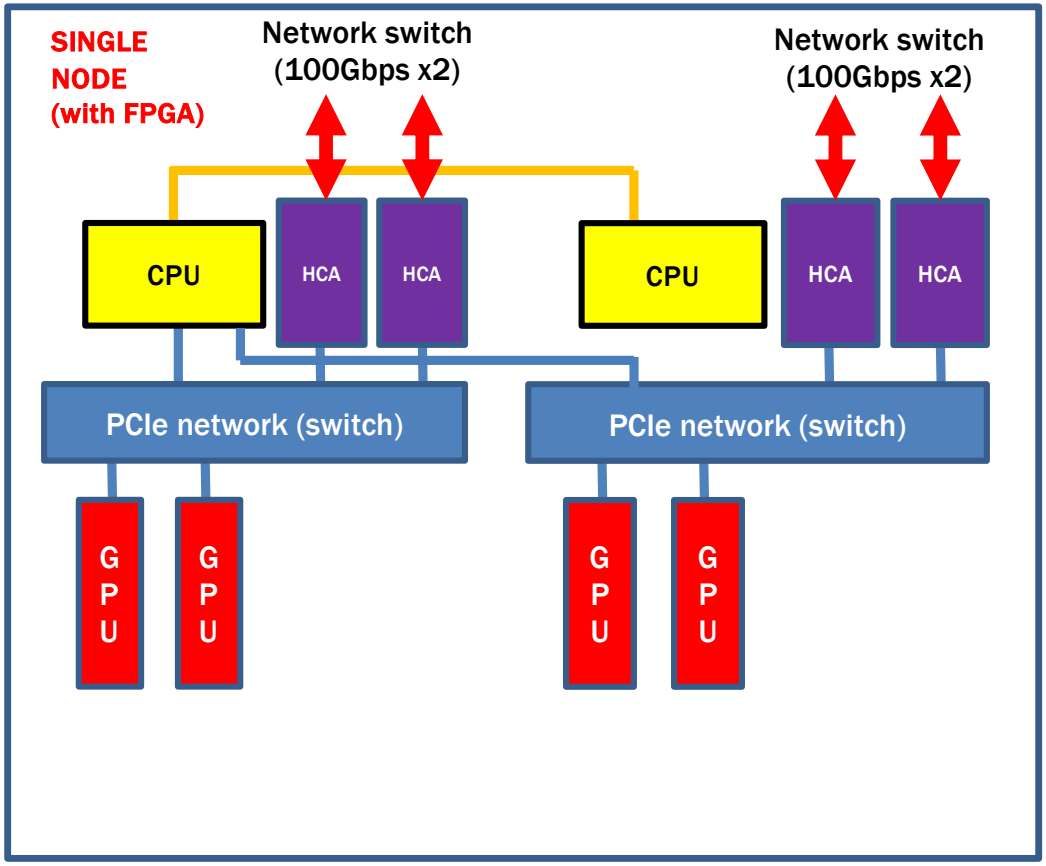
Single node configuration (Albireo)

- 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



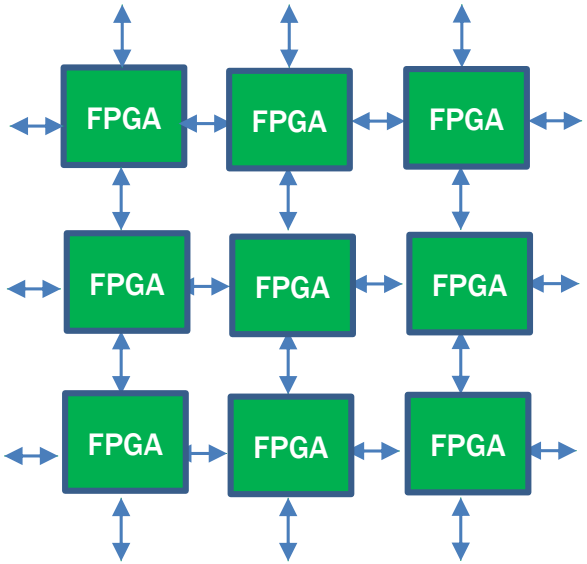
Single node configuration (Deneb)

- 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



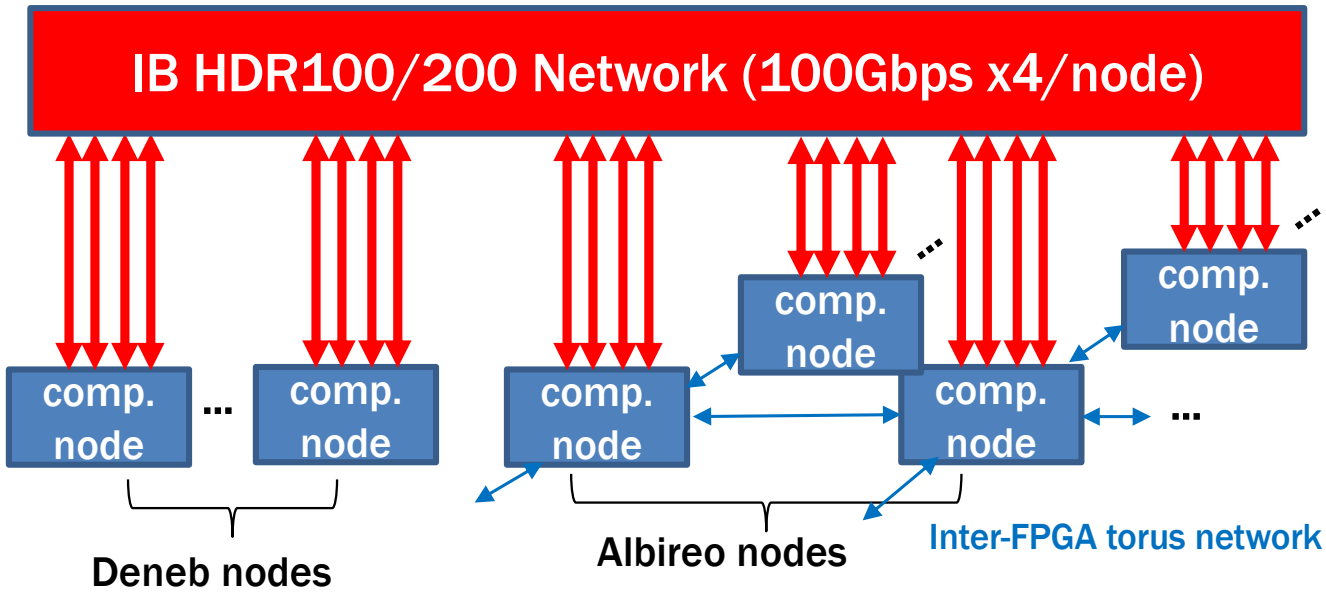
Two types of interconnection network

Inter-FPGA direct network (only for Albireo nodes)

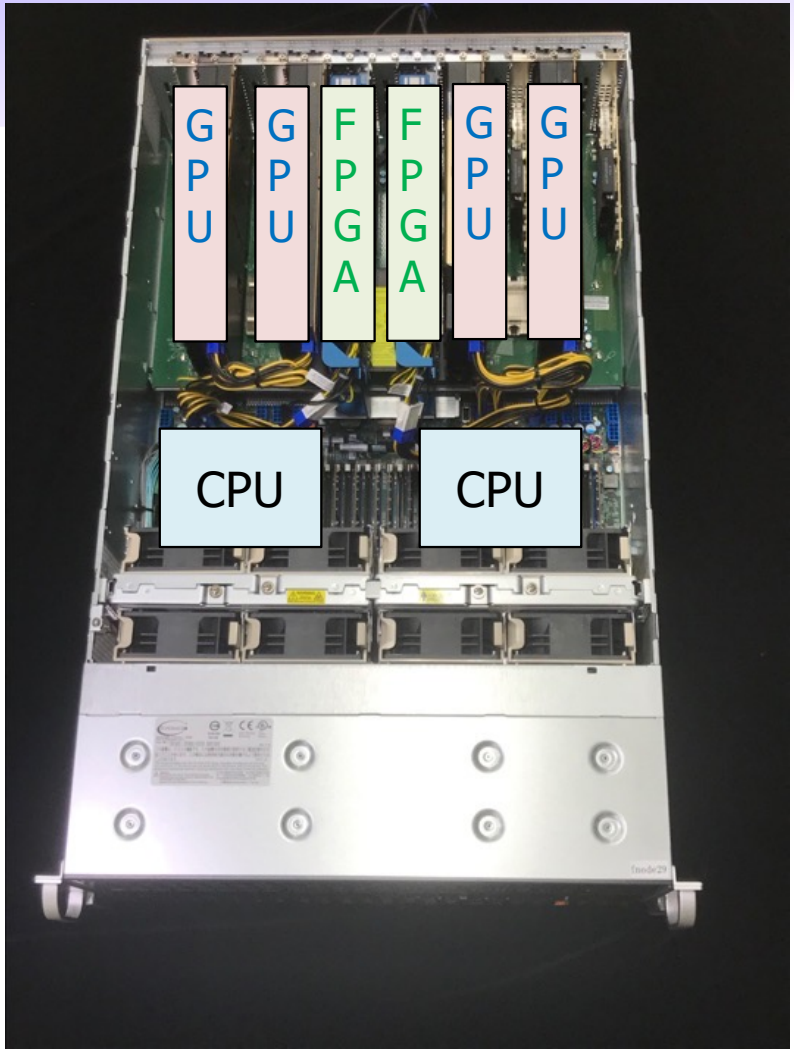


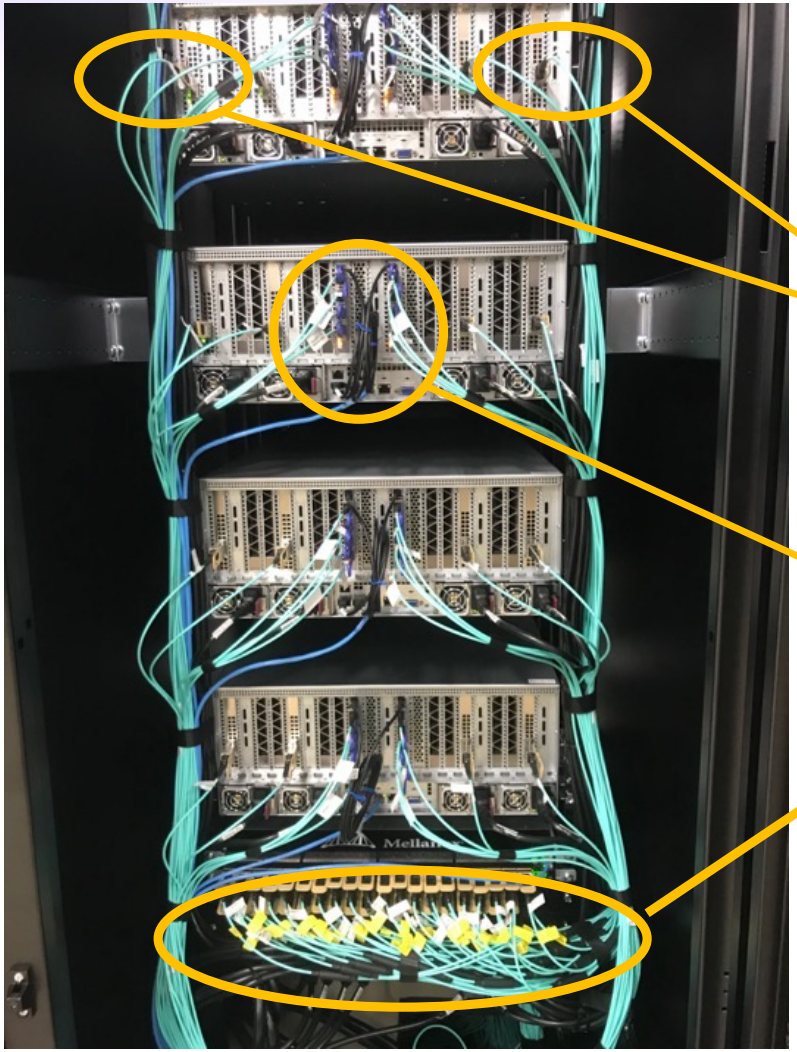
64 of FPGAs on Albireo nodes (2FPGAS/node) are connected by 8x8 2D torus network without switch

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 x2

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

1.2Tbps/node

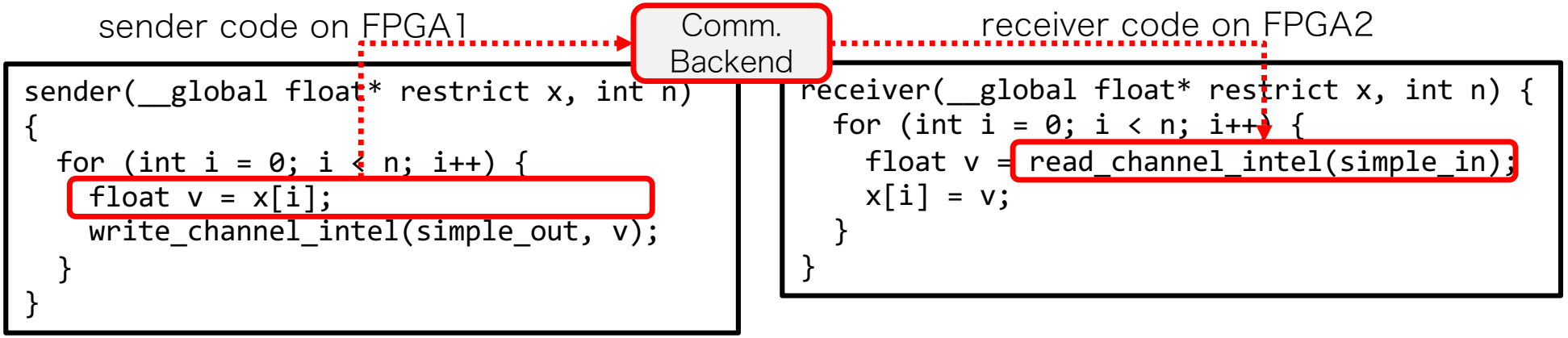
Albireo node

これまでの研究

- FPGA-network: CIRCUS (Communication Integrated Reconfigurable CompUting System)
  - FPGAボード間を光リンク（最大4本、100Gbps）で結合するネットワークを構築、ルータ機能をOpenCLから利用可能とする
  - 演算と通信を細粒度でパイプライン処理 ⇨ FPGAの特性を並列処理に最大限に活かす
- GPU-FPGA DMA: FPGAから起動しCPUの助けを借りない
  - PCIeのプロトコルを利用したDMAでGPUメモリとFPGAメモリ（global memory）間で高速データ転送
- Programming:
  - MHOAT (Multi-Hetero OpenACC Translator)
    ⇒ OpenACCによる単一コード記述でGPUとFPGAのオフローディングを簡潔に記述
  - Intel oneAPI
    ⇒ taskベースでGPUとFPGAのカーネル起動を簡潔に記述、DPC++を基本とするがOpenCLやCUDAも吸収可能
- Appllication: 宇宙物理学コードARGOT
  - 2種類の異なる動作と特性を持つ計算部分をGPUとFPGAに分散

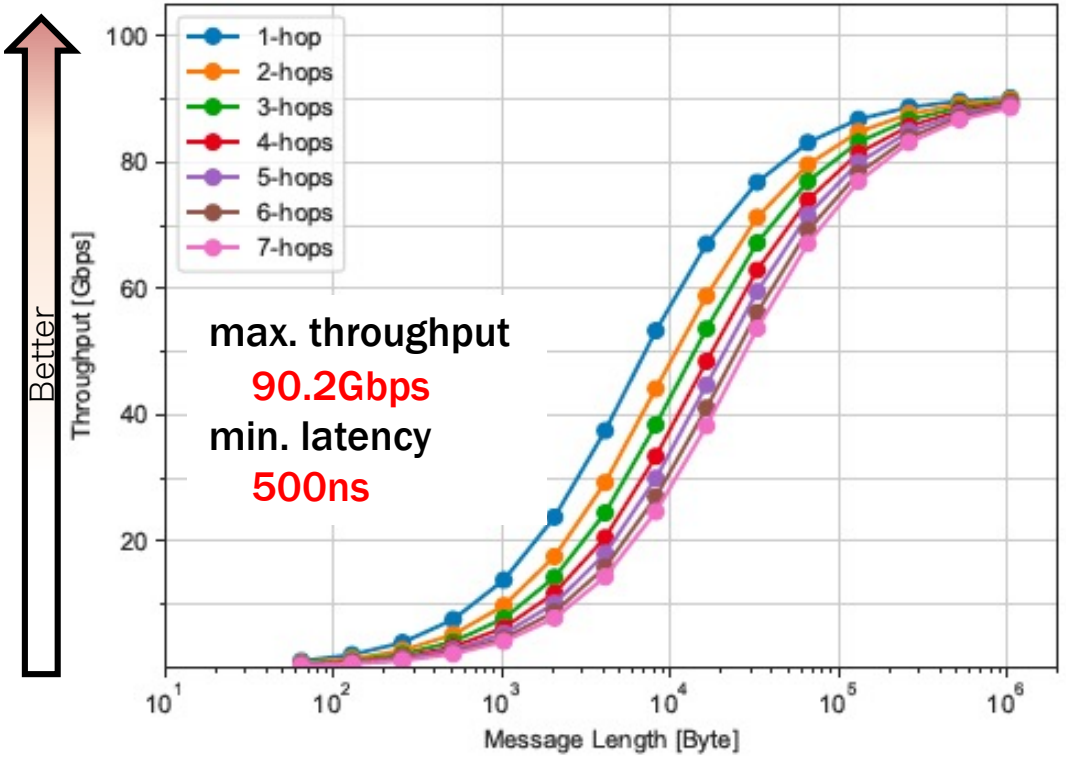
CIRCUS

- Intel FPGA SDK for OpenCL
  - FPGA hardware の起動を OpenCL APIで可能に
- 問題：光リンクはあるがそのルーティングを全部OpenCLで記述するのか？
  - 高速・低レイテンシの通信には Verilog HDL などのHDLによる記述が必要
  - MPI等は memory-to-memory 通信モデルでFPGAにはなじまない
  - FPGAの持つパイプライン演算特性を通信に延長
- →CIRCUS: Communication Integrated Reconfigurable CompUting System
  - 演算と通信をシームレスに結合し、細粒度処理を実現

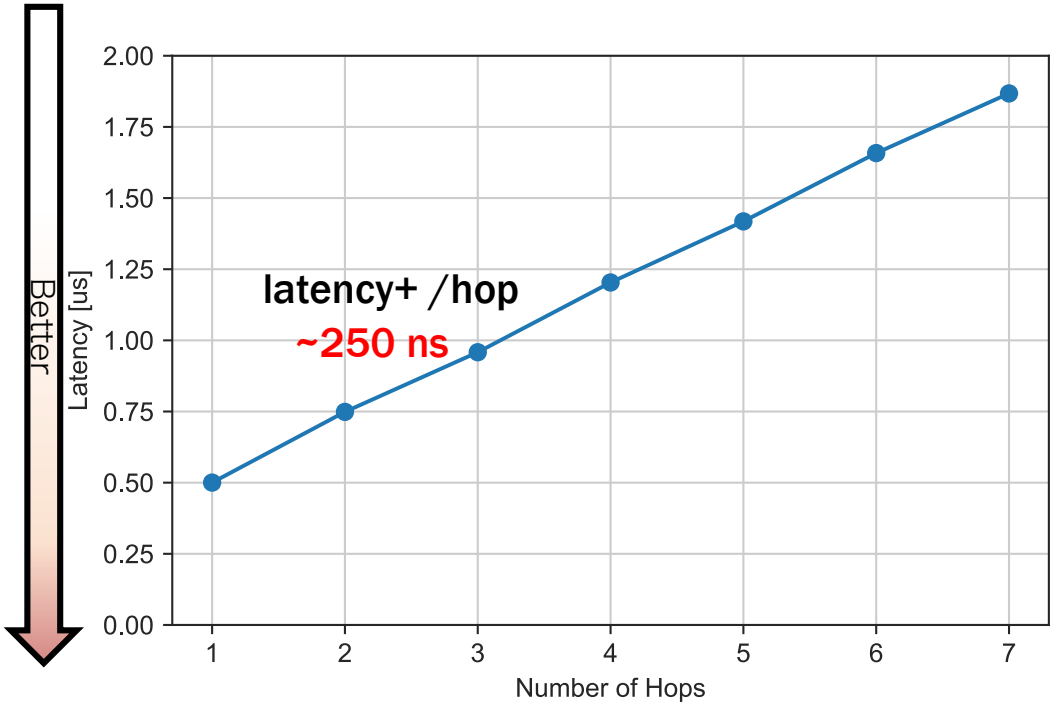


* N. Fujita, et al., “Performance Evaluation of Pipelined Communication Combined with Computation in OpenCL Programming on FPGA”, AsHES2020.

CIRCUS 通信性能（Cygnus）



Throughput（1hop~7hops）

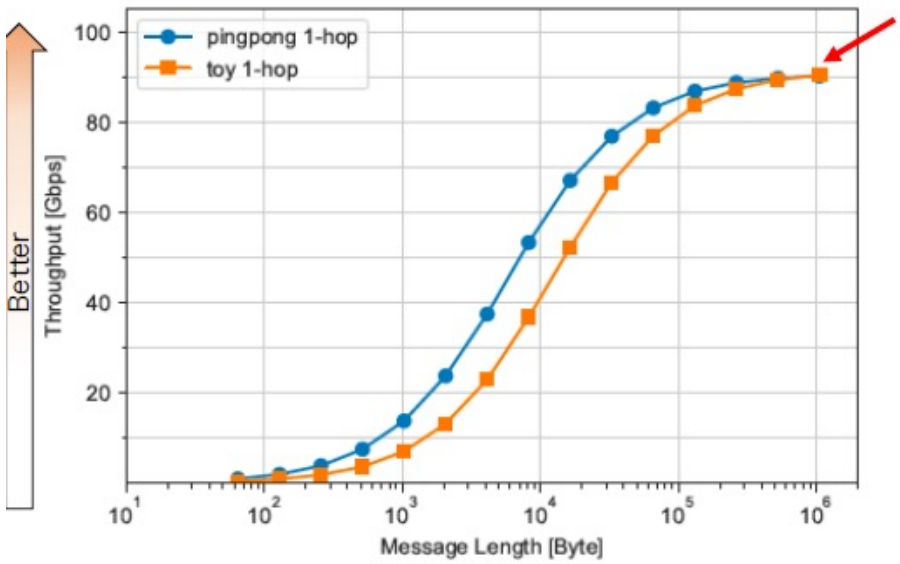


Latency（1hop~7hops）

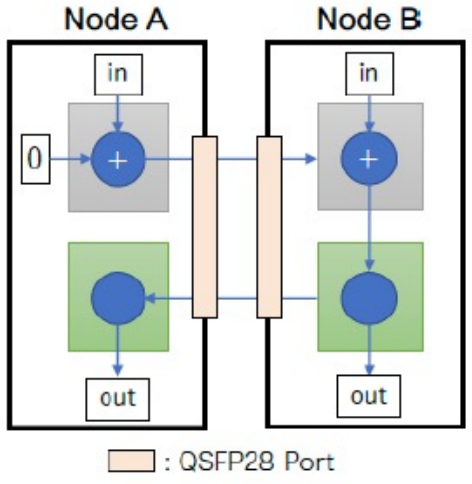
Evaluated on up to 8 Bittware 520N FPGA boards in Cygnus supercomputer at CCS, University of Tsukuba
N. Fujita, et al., “Performance Evaluation of Pipelined Communication Combined with Computation in OpenCL Programming on FPGA”, AsHES2020.

演算・通信パイプライン融合による collective 通信

- **CIRCUS**の機能を使って隣接通信と加算を**word**単位でパイプライン処理

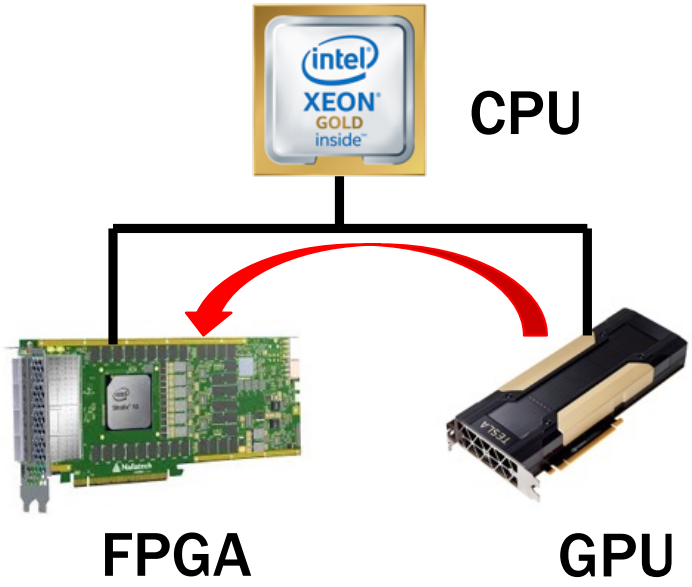


Performance comparison
pingpong benchmark vs. allreduce benchmark



Pipelining test code of Allreduce(+)

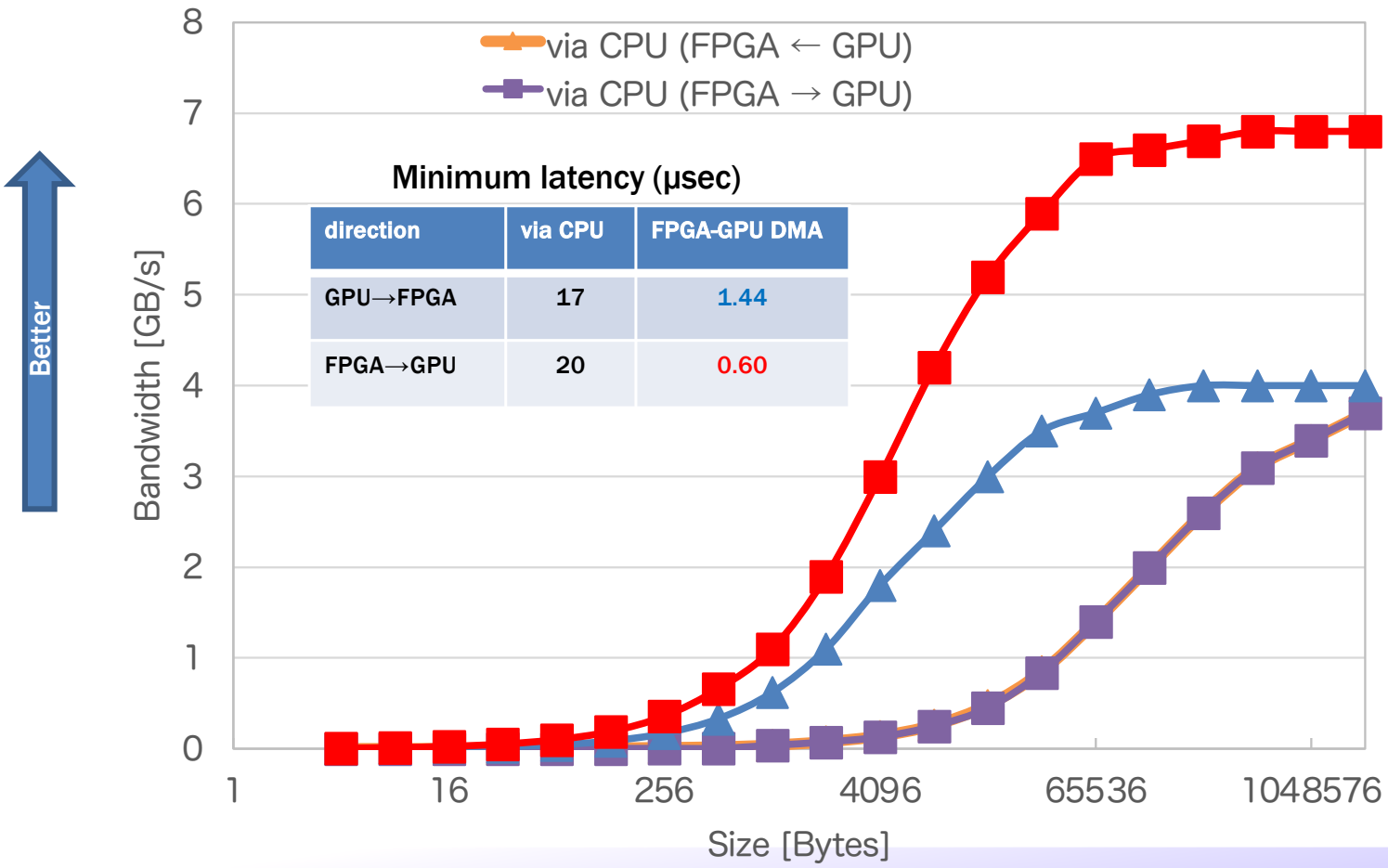
GPU-FPGA DMA




```
__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

FPGA-GPU DMA（Intel A10 + NVIDIA V100）



Minimum latency (μsec)

direction	via CPU	FPGA-GPU DMA
GPU→FPGA	17	1.44
FPGA→GPU	20	0.60

[Reference]

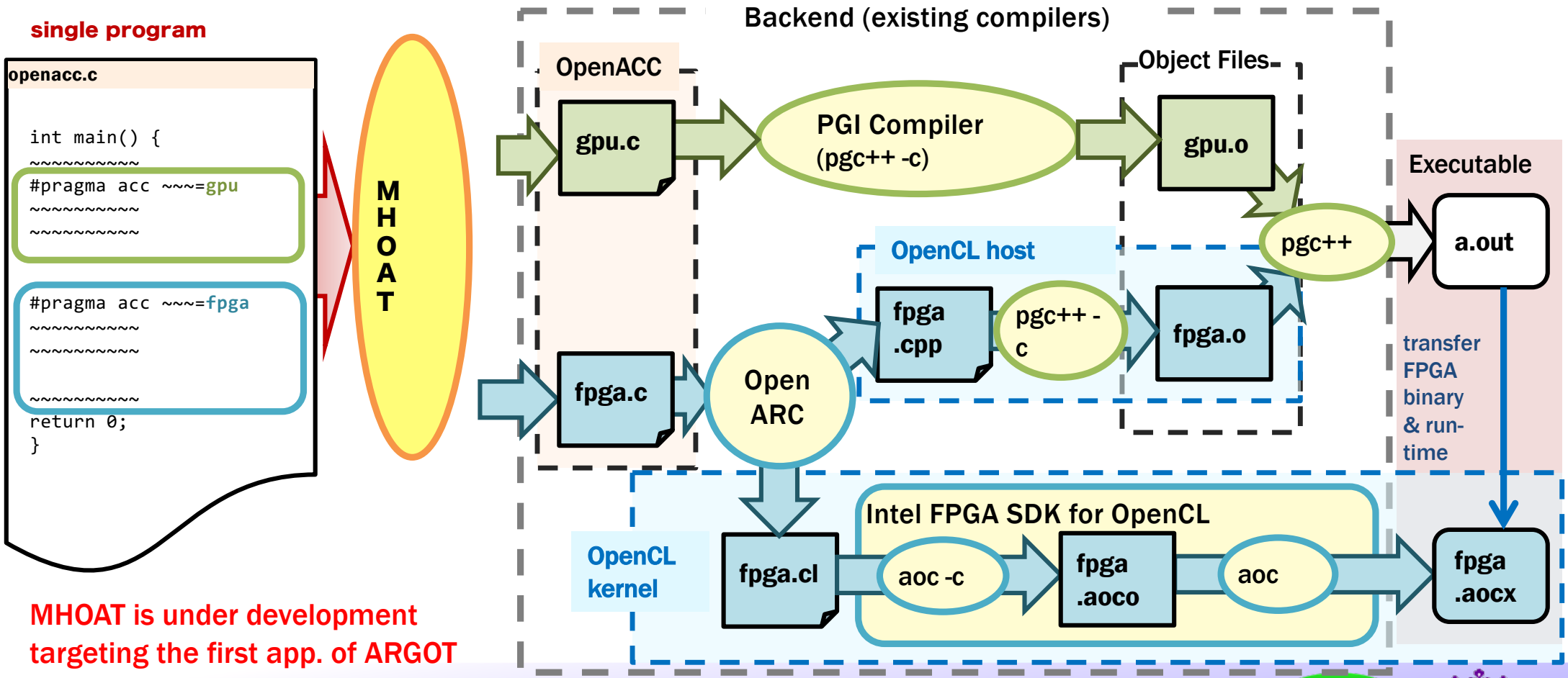
- Ryohei Kobayashi, Norihisa Fujita, Yoshiki Yamaguchi, Ayumi Nakamichi, Taisuke Boku, "GPU-FPGA Heterogeneous Computing with OpenCL-enabled Direct Memory Access", Proc. of Int. Workshop on Accelerators and Hybrid Exascale Systems (AsHES2019) in IPDPS2019 (to be published), May 20th, 2019.

OpenACCによる統一的記述を実現するメタコンパイラ

- OpenACC はNVIDIA GPUで使えるだけでなくFPGAでも利用可能（研究ベース）
  - 筑波大学CCS、ORNL、理研R-CCSでメタコンパイラを共同開発
  - GPU – PGI OpenACC コンパイラ
  - FPGA– OpenARC for FPGA on OpenACC: OpenACCコードをIntel FPGA向けのOpenCLに変換
- ホストコード上のランタイムシステムとの細かいコンフリクトを解消し、両コンパイラを機能結合するトランスレータ

  ⇨ MHOAT (Multi-Hybrid OpenACC Translater)
- 問題
  - GPUとFPGAの並列処理特性の違い
  - GPU – SIMD的な水平型データ並列処理
  - FPGA – クロックレベルでのパイプライン処理（＋マルチインスタンス）
  - メモリモデルの違い

Compilation flow of MHOAT



MHOAT is under development
targeting the first app. of ARGOT

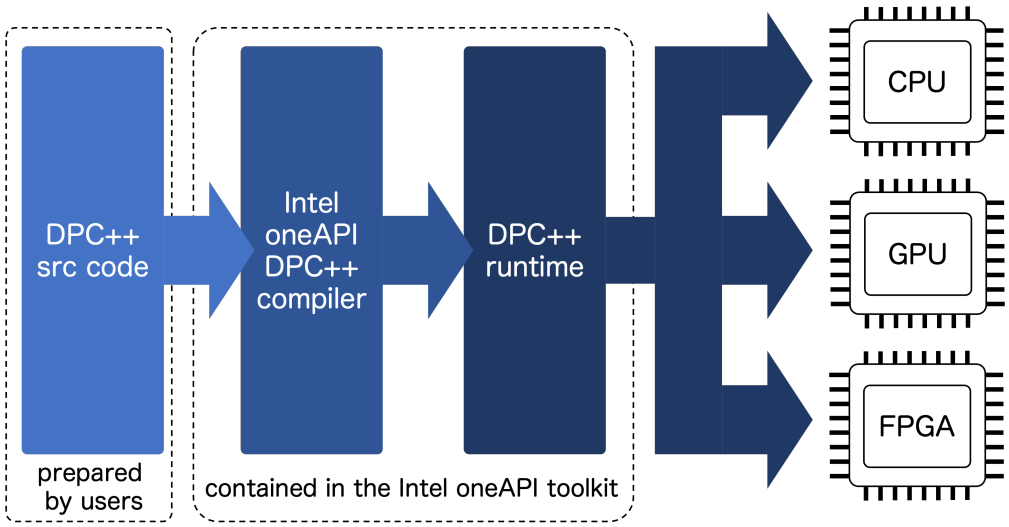
MHOATの現状

- 簡単なコードは実行可能
- 実アプリケーション（**ARGOT**）のコンパイルでは**OpenARC**コンパイラの特異性の解消が難しく細かいソースコード書き換えが必要
- **ARGOT**コードのコンパイルはできるが性能、演算結果に問題がある

⇨ 開発を継続中

もう一つのアプローチ：Intel oneAPI

- oneAPI is a cross-architecture programming framework
  - simplify the development across different architectures
- Data Pallalell C++ (DPC++)
  - DPC++ = C++ + SYCL + extensions
  - enables unified description across different architectures



こちらについては2022年4月のPCクラスタコンソーシアムOSSワークショップで紹介済み

アプリケーション例

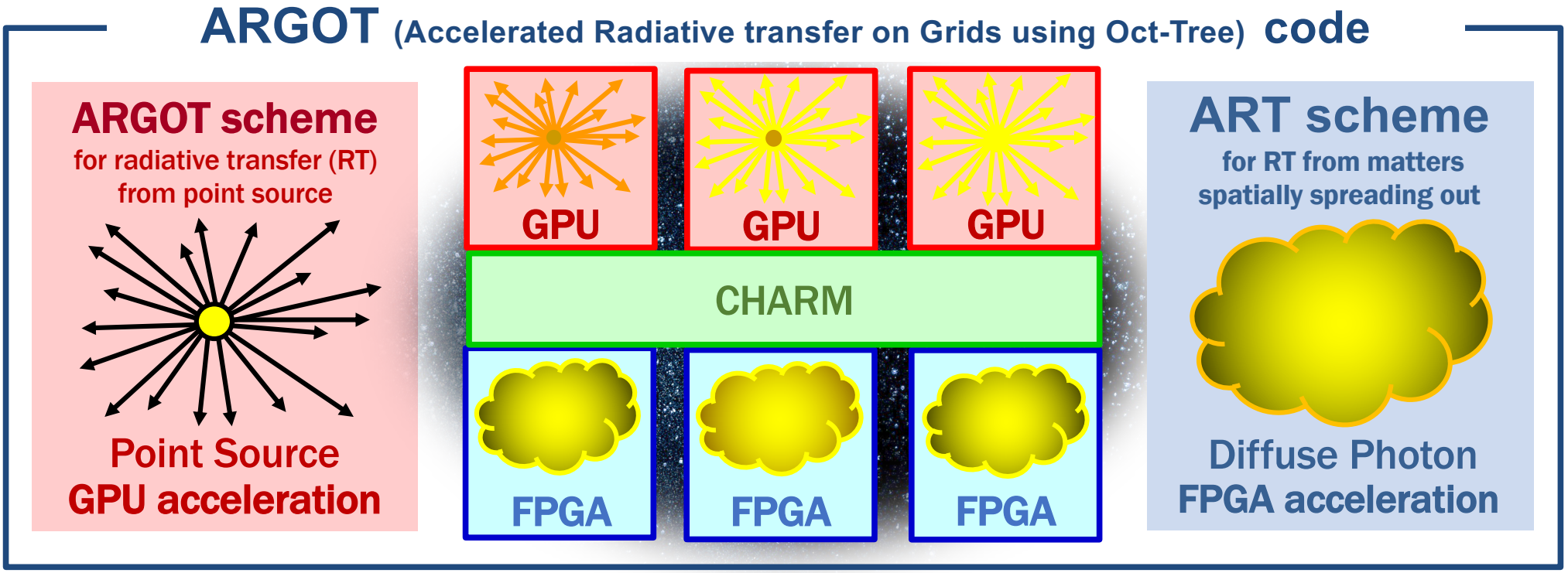
- ARGOT (Accelerated Radiative transfer on Grids using Oct-Tree)
  - Simulator for early stage universe where the first stars and galaxies were born
  - Radiative transfer code developed in Center for Computational Sciences (CCS), University of Tsukuba
  - CPU (OpenMP) and GPU (CUDA) implementations 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
- We accelerate the ART method on an FPGA using Intel FPGA SDK for OpenCL as an HLS environment (with oneAPI)

ARGOTコードの2つの計算要素：ARGOT法とART法

- ARGOT method: Point Source processing
- ART method (Authentic Radiation Transfer): Diffused Photon processing

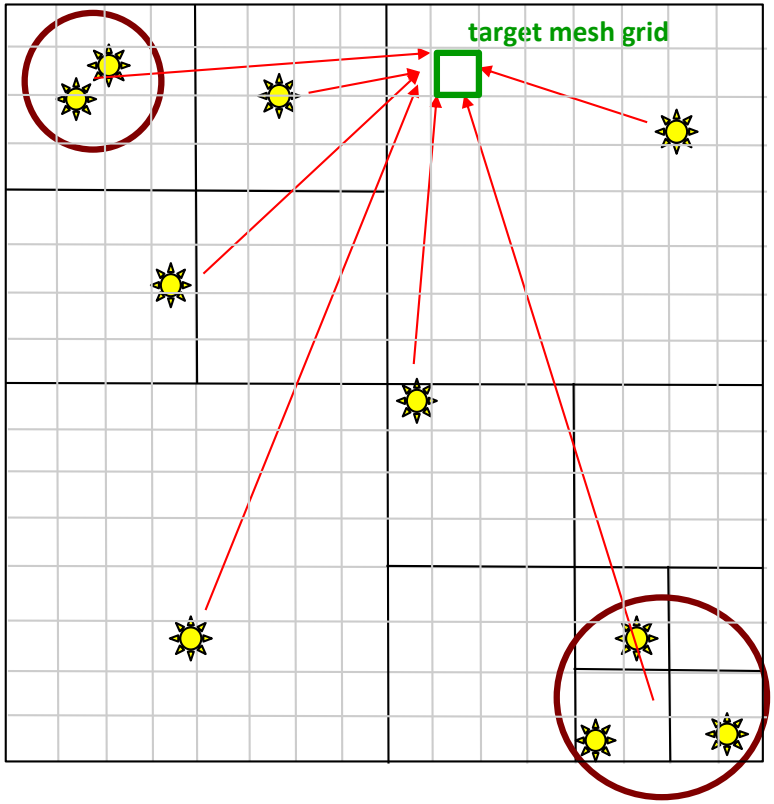
ARGOTコードの2つの計算要素：ARGOT法とART法

- ARGOT method: Point Source processing
- ART method (Authentic Radiation Transfer): Diffused Photon processing



ARGOT法　(CUDAによる実装)

- ARGOT法は点光源の輻射輸送を計算
- 八分木を用いて3次元空間に分散する点光源を表す
  - 遠距離にある点光源の集合を単一の光源とみなせる
    計算量: O(N^2) → O(NlogN)
- 重力計算における Tree-Code に似た手法
  - GPU実装に適している

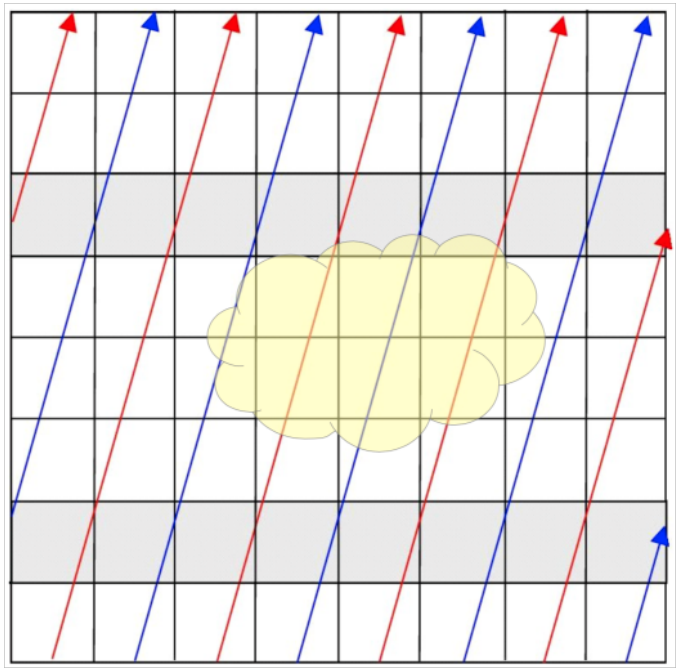


ART法(OpenCLによる実装)

- ART法は空間に広がる光源からの輻射輸送を計算
  - 問題空間の端からレイが並行に直進
- ART法はGPUに不向きな計算
  - レイの進行方向によってメッシュデータのメモリアクセスパターンが変化
    ランダムアクセスに近いアクセスパターン
  - Atomic演算が必要
  - 計算中に含まれる演算数が不十分

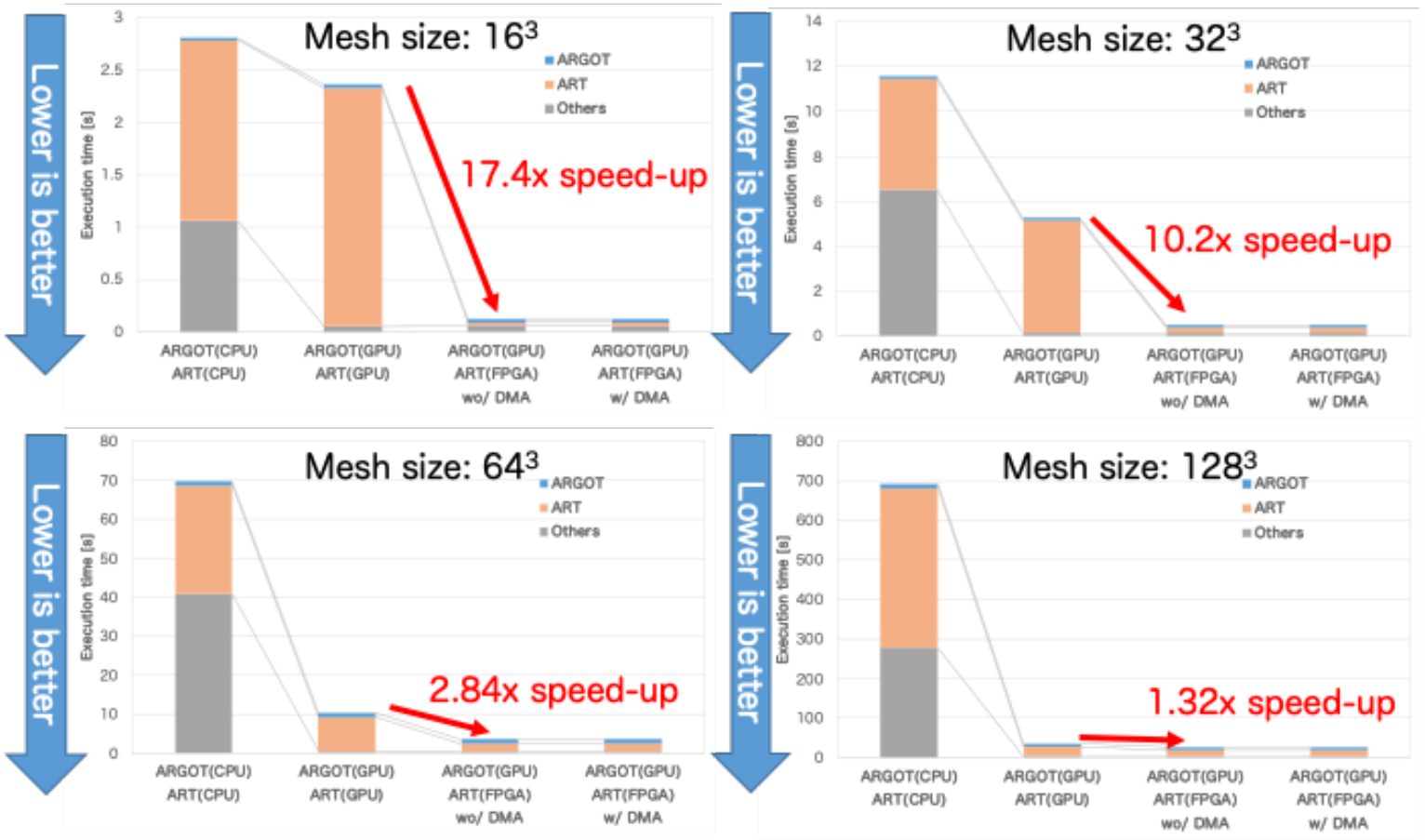
- FPGAを用いて高速化
  - FPGAのon-chipメモリを活用
  - 一定サイズのベクトル処理が独立に大量に必要
    ⇒ GPUの大規模SIMDが有効活用できない

ART法の実行時間は全体の90%を占める



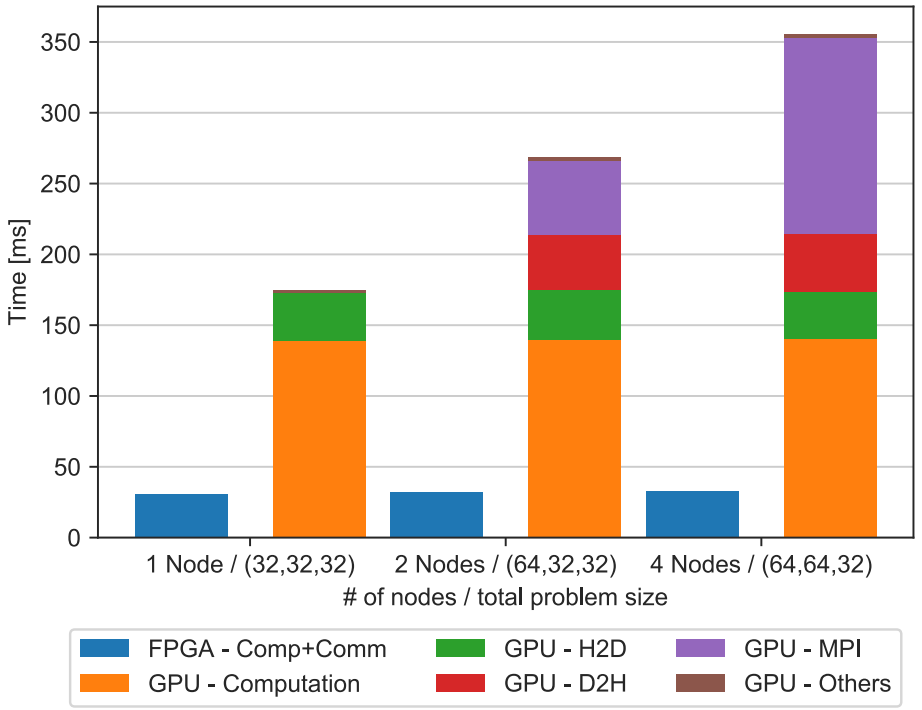
詳細：**Norihisa Fujita, et.al., “Accelerating Space Radiative Transfer on FPGA using OpenCL”, HEART2018**

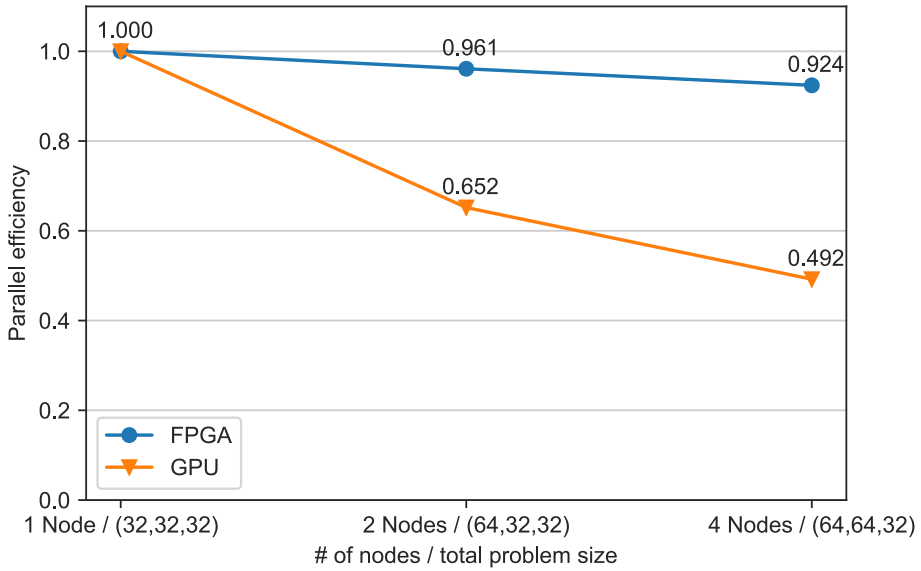
GPU-only vs GPU-FPGA 協調計算（ARGOT全体）



R. Kobayashi, et. al., “Accelerating Radiative Transfer Simulation with GPU-FPGA Cooperative Computation”, ASAP2020, Jul. 2020.

ART法の並列FPGA性能



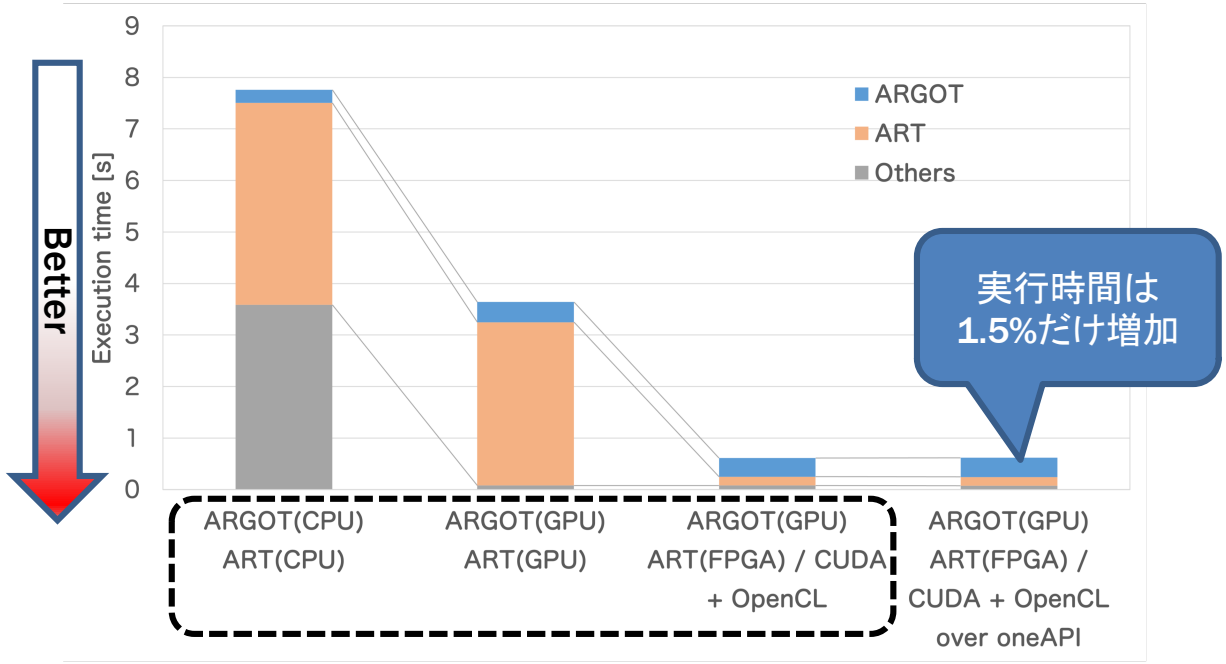


Weak scalingによる4ノードまでのART法処理時間（左）と並列化効率（右）

*Norihisa Fujita, et.al., "OpenCL-enabled Parallel Raytracing for Astrophysical Application on Multiple FPGAs with Optical Links", 2020 IEEE/ACM International Workshop on Heterogeneous High-performance Reconfigurable Computing (H2RC) in conjunction with SC20, Nov. 2020.

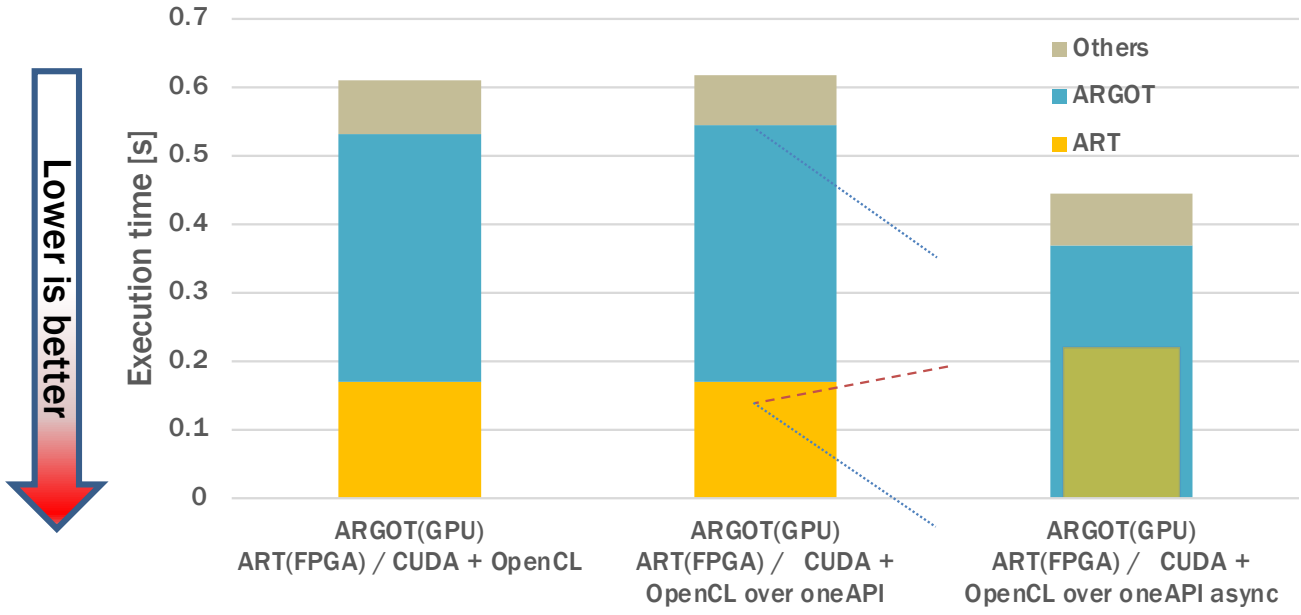
oneAPI (single node) 性能評価

- 問題サイズ: 32^3
- Others
  - ベクトル並列化が可能
    GPUによる高速化が効果的
- ART
  - ART法のGPU実装はCPUと同程度の性能
  - FPGAにより高速化
- GPU:CUDA+FPGA:OpenCL
  - GPUのみの場合と比べて10xの性能向上
- oneAPI vs CUDA+OpenCL
  - oneAPI版の実行時間が1.5%増加
    →oneAPI処理のオーバヘッドは無視できる



ARGOT法およびART法の非同期実行

- 本来ARGOT法とART法は2種類の物理現象を独立に計算し、最後に結果を足し合わせている
- 各Queueは非同期に実行可能
  - OpenMPを利用して，各Queueにactionを非同期に投入
  - Queueに投入されたactionは非同期実行される
- ART法の実行時間は ARGOT法の実行時間に隠蔽される
  - 全体として1.38xの高速化



*柏野隆太他, “oneAPIを用いたGPU・FPGA混載ノードにおける宇宙物理シミュレーションコードARGOTの実装”, 第183回HPC研究会

まとめと今後の展望

- FPGA単体でのHPC利用はなかなか難しい
  - 絶対性能（FLOPS）
  - メモリテクノロジ
- GPUとFPGAを相補的に利用することでmulti-physics simulationに柔軟に対応
  - バルク並列処理
  - 複雑な処理、条件分岐
  - ノード間通信
- 筑波大学CCSではCHARM (Cooperative Heterogeneous Acceleration with Reconfigurable Muti-devices)コンセプトの下、GPUとFPGAを適材適所で用いることで全方位的な新しいHPC solutionを開拓している
- Big Data/AI処理のベースは現状ではDeep LearningでGPUがそれに特化されつつあるが、柔軟な部分処理にはFPGAが適用できる（例：sort engine、staging処理等）
- HPC/AIのエンドユーザ向けのプログラミングが最大の問題
- 今後の展望
  - CygnusにおけるoneAPIの本格利用
  - ORNL-OpenARCとは独立に理研R-CCSと筑波大の共同研究で Omni OpenACC-OpenCL translater (for FPGA)を開発中
  - MHOATの実装とARGOTのような実コードの高速化
  - 富岳/ESSPERにARGOTを移植中

研究協力

- 筑波大学・計算科学研究センター/筑波大学・システム情報系


小林涼平


藤田典久


山口佳樹


梅村雅之


吉川耕司