TCA機構における GPU対応GASNetの実装

佐藤 賢太[†1], 藤田 典久[†2], 塙 敏博[†3] ,
朴 泰祐[†2, 1] , Khaled Ibrahim [†4]

†1筑波大学 大学院 システム情報工学研究科
†2筑波大学 計算科学研究センター
†3東京大学 情報基盤センター
†4Lawence Barkley National Laboratory

研究の背景

- GPGPUを利用したGPUクラスタの普及
  - 非常に高い演算性能を持つ
  - ノードを跨ぐGPU間の直接通信ができない

- TCA (Tightly Coupled Accelerators)
  - ノードを跨ぐ演算加速装置間での直接通信を実現
  - PEACH2 (PCI Express Adaptive Communication Hub ver.2)
    - TCAのプロトタイプ実装
  - 利用には独自のAPIを用いる必要がある
    - プログラミングコストが高い
    - 既存のアプリケーションの移植が困難

研究の背景 (Cont'd)

- GASNet
  - Lawrence Berkeley National Laboratory (LBNL)によって開発
  - PGAS言語向け通信ライブラリ
    - UPC, Co-array Fortran, XcalableMPなどが利用
  - ランタイムやコンパイラが生成したコードが利用することを想定
    - 機能は乏しいが軽量動作
  - 既存のGASNetはCPUメモリしか想定されていない

    ⇛LBNLでGPU対応のための拡張が開発中

研究の目的

- GPU対応GASNetをTCA/PEACH2で実装
  - LBNLで開発中のInfiniBand版と並行して実装を行う
  - GASNetやPGAS言語を介して各種ソフトウェアとの互換性が生じる
    - 将来的にXcalableACCのようなGPUに対応したPGAS言語で利用されることを想定

      ⇛TCA/PEACH2を広く利用できるようになる

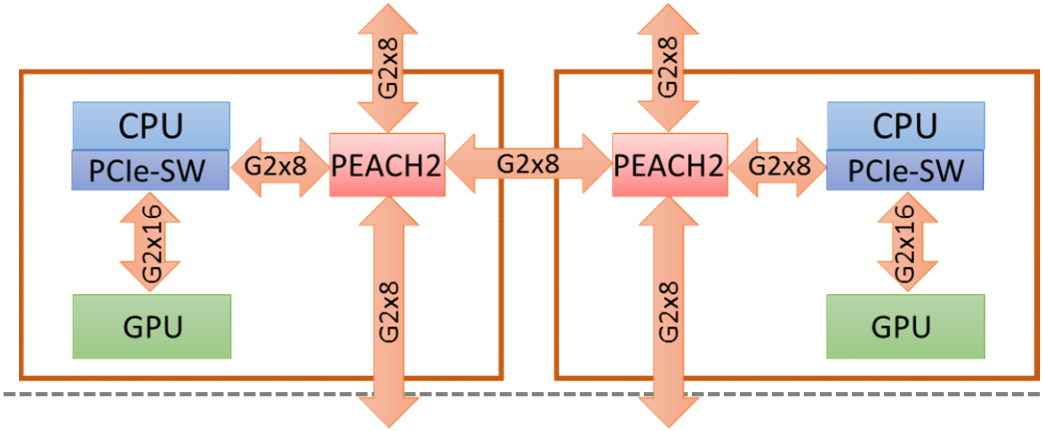
- 実装したGPU対応GASNetの性能評価
  - GASNet/IB, MVAPICH2-GDRとの性能比較
  - 姫野ベンチマークの性能評価

TCA/PEACH2

- FPGAによるTCAのプロトタイプ実装
- ノードを跨ぐCPU間およびGPU間での直接通信を実現

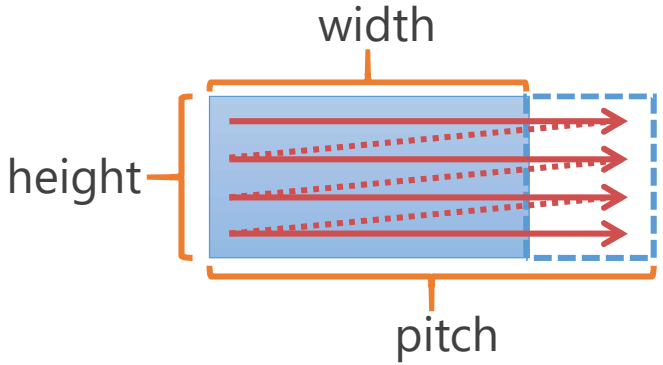
- ノード間の接続にPCIeを利用
  - PCIeパケットをルーティングすることでノードを跨ぐPCIeデバイス間で，PCIeプロトコルによる直接通信を実現
- 4つのPCIe Gen2 x8ポートを持つ
  - 1ポートはホストCPUと接続，残り3ポートは他ノードのPEACH2との接続
  - ノード内のGPUにはCPUに内蔵されたPCIe-SWを介して接続
- GPUのメモリにはGDR (GPU Direct RDMA)を利用してアクセス

- RDMA Writeのみ対応
  - アラインメントに関する制限がある
  - CPUメモリはTCAのAPI (*tcaMalloc()*)によって確保された領域のみ転送できる



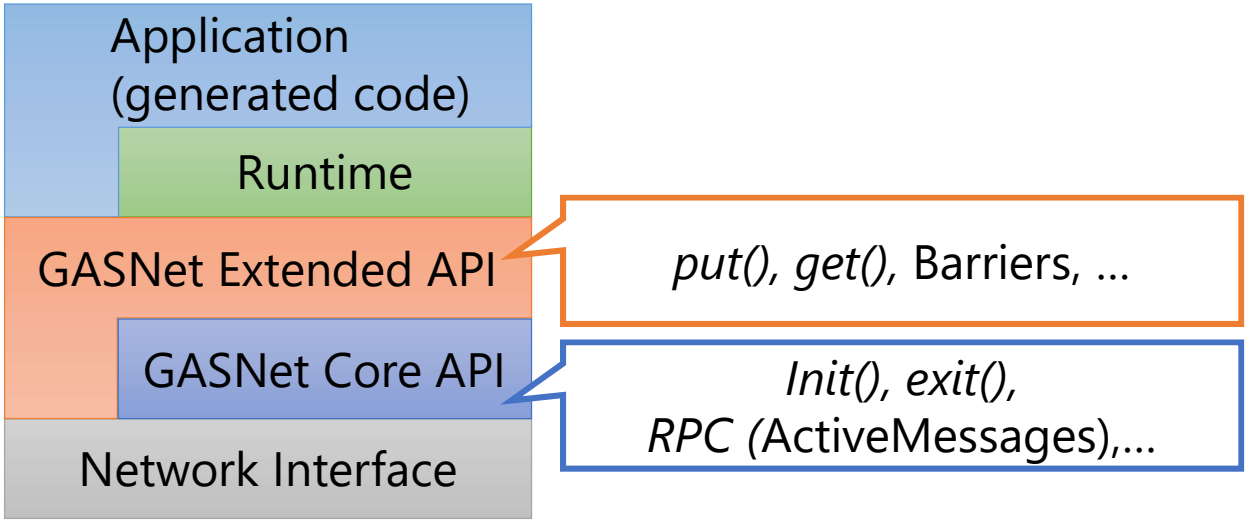
DMA通信

- DMAC (DMA Controller)を使った通信
  - PEACH2には4チャネル実装されている
- DMAディスクリプタで通信を指示
  - 転送元・転送先アドレス，転送サイズなど
  - ディスクリプタは再利用が前提
    - DMA起動が低コストで行える
- ブロックストライド転送



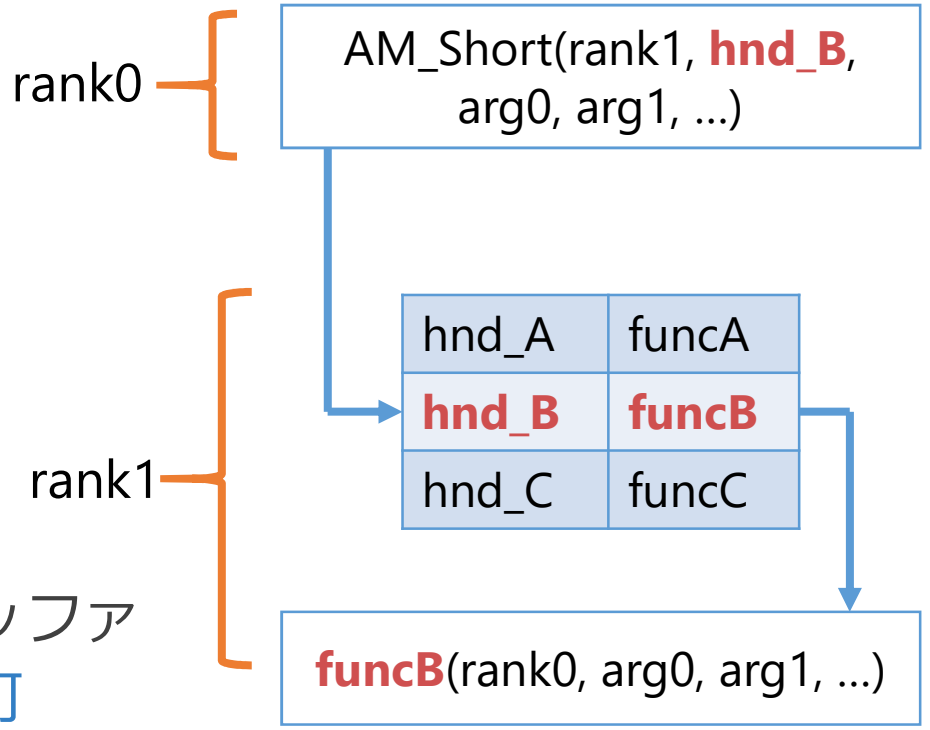
GASNet

- PGAS言語向け通信ライブラリ
  - ノード間(Point-to-Point)のデータ転送に特化
- Core APIとExtended APIで構成
  - Core API: 制御用のAPI
  - Extended API: データ転送のためのAPI
    - Core APIのみで実装されたリファレンス実装もある
    - ハードウェアが対応いない処理は実装しなくても良い



GASNet – Core API

- *init(), exit(), attach(), poll(), …*
- Active Message
  - RPCの一種
  - Short (*AM_Short()*)
    - 引数のみ
  - Medium (*AM_Medium()*)
    - 引数とデータ
    - 転送先はシステムのバッファ
    - 大きいデータは転送不可
  - Long (*AM_Long()*)
    - 引数とデータ
    - 転送先はユーザが指定した領域



GASNet – Extended API

- Memory to memory transfer
  - *put(), get(), memset()*
- Register to memory transfer
  - *put_val(), get_val()*
- Barrier
  - *barrier_notify(), barrier_wait(), barrier_try()*

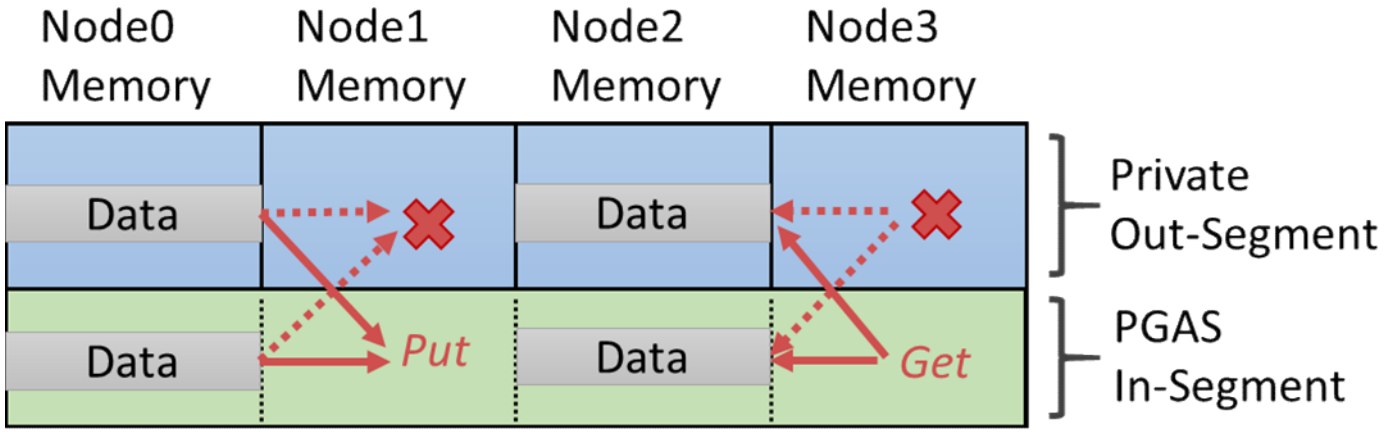
- Unofficial API
  - Non-contiguous data transfer
    - Vector: *putv(), getv()*
    - Indexed: *puti(), geti()*
    - Strided: *puts(), gets()*
  - Collective communication

  ⇛GASNet-EXでは公式にサポートされる予定

GASNet - Segment

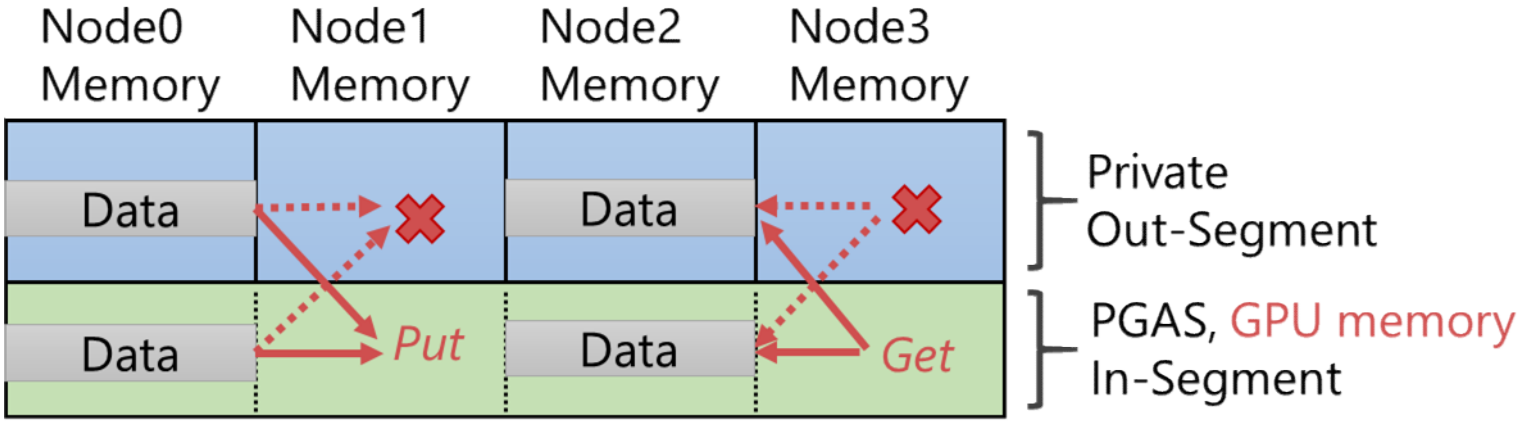
- 通信用に利用するメモリ領域を*Segment*と呼ぶ
  - 初期化時に単一連続領域を確保

- リモートアクセスする領域はSegment内のみ
  - *put()*と*AM_Long()*の転送先はSegment内
  - *get()*の転送元はSegment内



GASNet - GPU対応

- SegmentにGPUメモリを割り当てる
  - 既存のGASNetはSegmentにCPUメモリを割り当てていたため，仕様上，Segment外のGPUメモリにはアクセスできない
- Segment外の領域はどちらでも構わない



- リモートのCPUメモリにはアクセスできない
  - Segmentは単一の連続領域である必要があるため排他利用となる

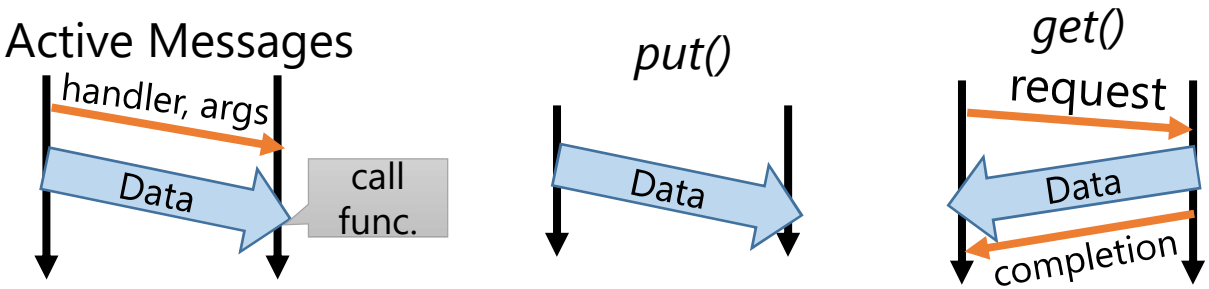
⇛GPUアプリケーションではGPU to GPUの通信が殆どのためそれほど大きな問題ではない

⇛GASNet-EXでは複数Segmentがサポートされる予定

実装

- 実装対象
  - Core API (≒Active Messages)
  - Memory to memory transfer: *put()*, *get()*
  - Non-contiguous data transfer (strided): *puts()*, *gets()*
    - PEACH2のブロックストライド転送との親和性が高い
    - ブロックサイズが小さい場合はPack/Unpack
- 上記以外のAPIはリファレンス実装を利用

- 以下の機能の組み合わせで実装
  - 制御メッセージの送受信
  - データ転送



制御メッセージの送受信

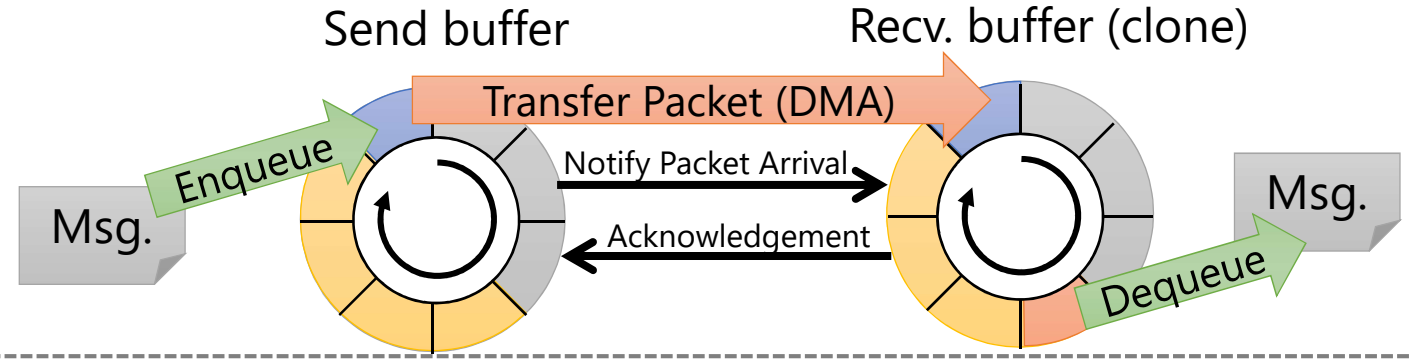
- PEACH2はSend/Recv.ができない

⇛RDMA Writeによるパケット通信を行なう

- 各ノードペア間にリングバッファを用意
  - 各バッファは*tcaMalloc()*で確保したCPUメモリ

1. 送信バッファにパケットを用意
2. 送信バッファから受信バッファにパケットを転送
3. パケット転送後，送信側はパケットの到着を通知
4. パケットの処理完了後，受信側はACKを返す
5. ACK受信後，送信バッファのエントリを解放（フロー制御）

RDMA Write



データ転送

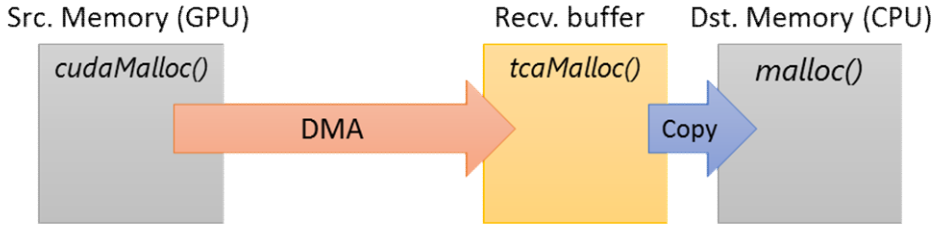
- CPUメモリは*tcaMalloc()*で確保した領域しか転送できない
  - *malloc()*などで確保された領域を転送するにはバッファを経由する必要がある
  - Segmentの制限によってAPI毎に実行可能な転送パターンは限定

  ⇛転送パターンに応じて最適な転送モードを選択し，メモリコピーを最小化

Faster

転送パターン	API	転送モード	メモリコピー	説明
GPU to GPU	*put(), get()*	Memory to Memory	0回	DMAによる直接転送
CPU to GPU	*put()*	Buffer to Memory	1回	送信バッファを経由
GPU to CPU	*get()*	Memory to Buffer	1回	受信バッファを経由
（原則）アラインメント調整のみ		Buffer to Buffer	2回	送信・受信バッファを経由 （パケット通信）

- Ex.) *get()* (GPU to CPU) ⇛ Memory to Buffer mode



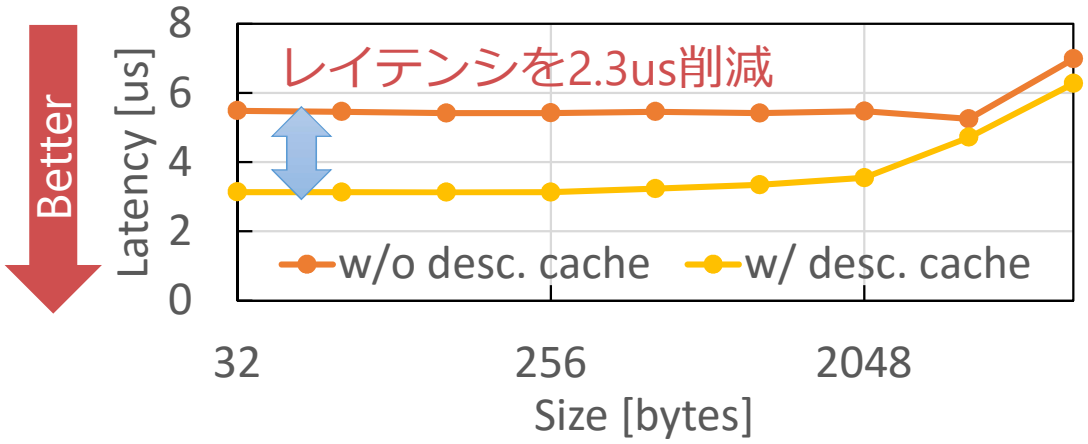
最適化1：ディスクリプタキャッシュ

- TCA/PEACH2は固定パターンの通信を繰り返すアプリケーションを想定
  - ディスクリプタの再利用によるDMAC起動コスト削減
  - ユーザが起動時にディスクリプタを作成し使い回す

- GASNet/TCAではディスクリプタ作成・利用はライブラリ内部で行われる
  - ディスクリプタを毎回作成するとオーバヘッドが大きい
    - ディスクリプタの再利用が必要
  - 任意の通信を想定する必要がある
    - 固定パターンの通信だけが実行されるとは限らない
    - ディスクリプタのエントリ数には限りがある

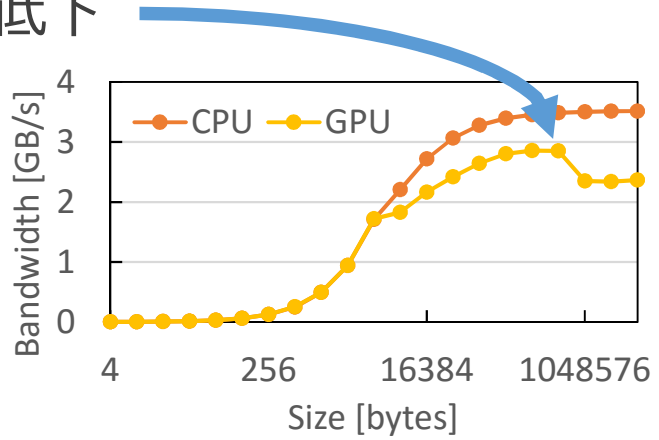
⇛通信パターン毎に実行回数，利用頻度などを記録して再利用性を判定

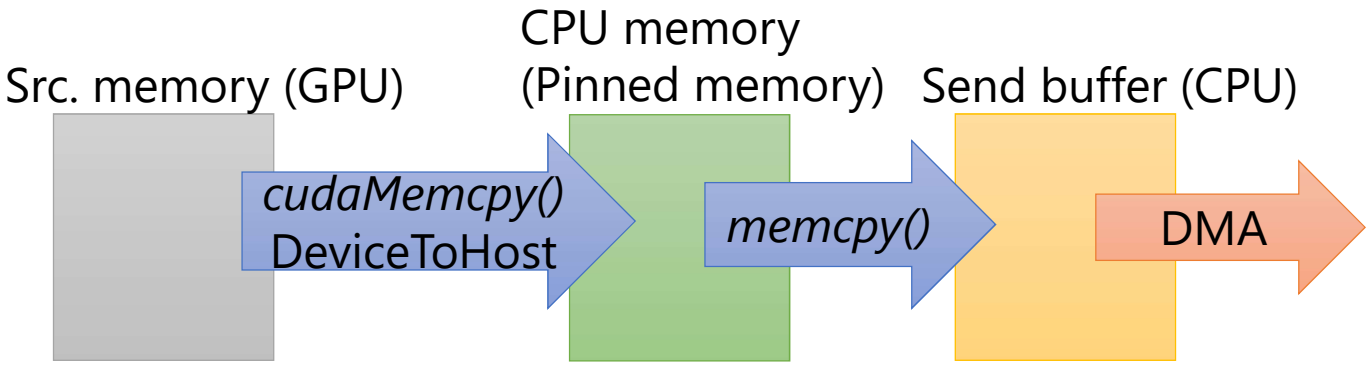
- 同じアプリケーション（通信パターンが同じ）であれば，
  TCA/PEACH2を直接利用する場合と同様の再利用が自動的に行われる



最適化2：CPUメモリを経由した転送

- GDR (GPU Direct RDMA)による転送はバンド幅が低い
  - 512KBをピークとしてバンド幅が低下
  - PEACH2のピーク性能
    - CPU: 3.5 GB/s
    - GPU: 2.9 GB/s
- *cudaMemcpy()*はGDRより高いバンド幅が得られる
- ⇛CPUメモリを経由して転送

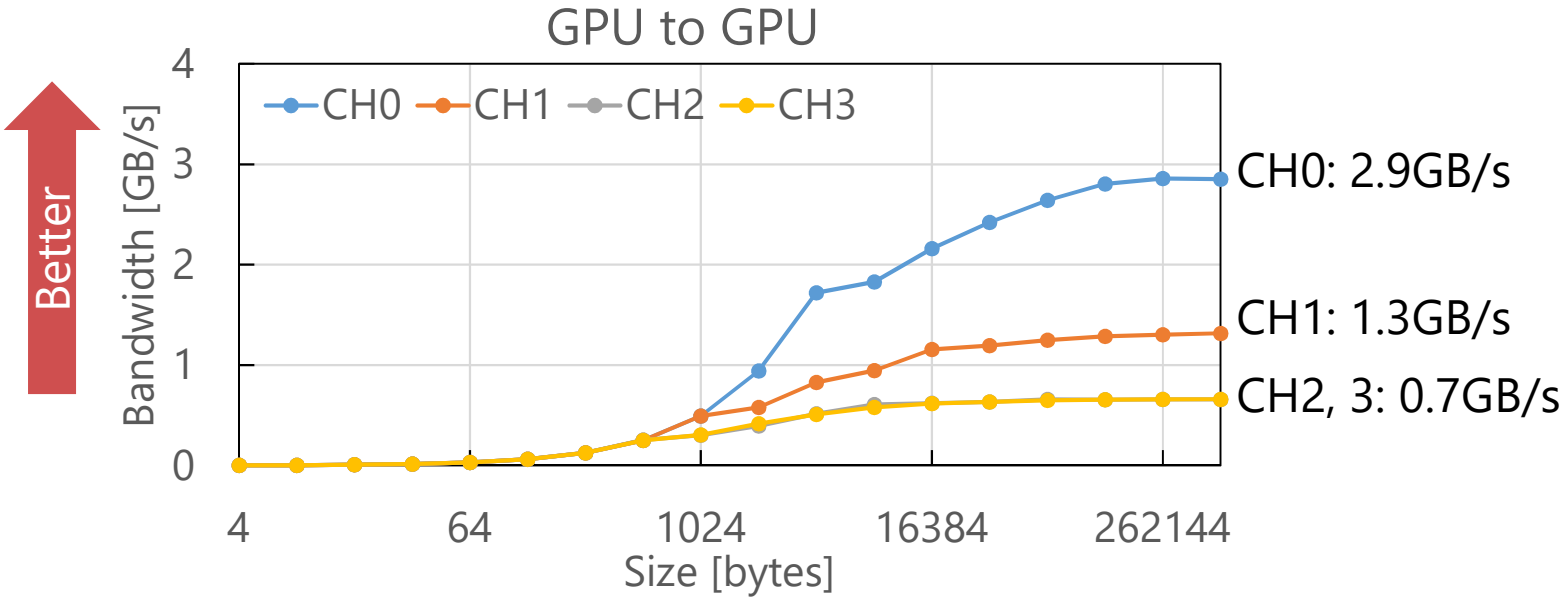




最適化3：複数DMACの利用

- PEACH2にはDMACが4チャネル搭載されている

⇛複数チャネルを束ねることで性能向上



- 各DMACの転送性能：CH0 > CH1 > CH2 = CH3

⇛DMACの性能に応じた負荷分散

最適化3：複数DMACの利用 (Cont'd)

- 通信リクエストを複数の小さい転送に分割
  - DMACの性能に応じて分割サイズの調整が必要

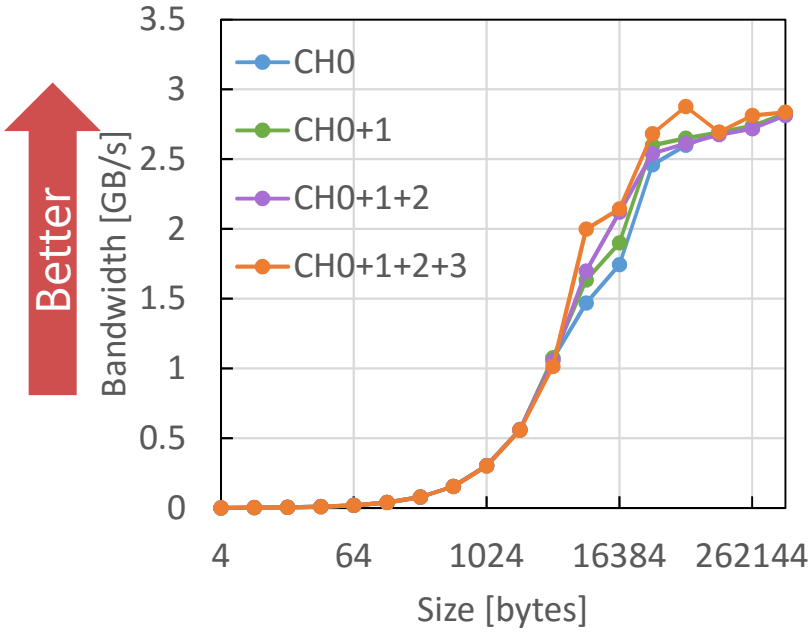
- 分割サイズの調整
  - ① 分割サイズと転送時間から各DMACのバンド幅を計算
  - ② 各DMACのバンド幅に比例するように再分配

⇛同じパターンの通信が繰り返されるため有効

- 分割サイズが毎回変わる
  - ディスクリプタキャッシュとの相性が悪い

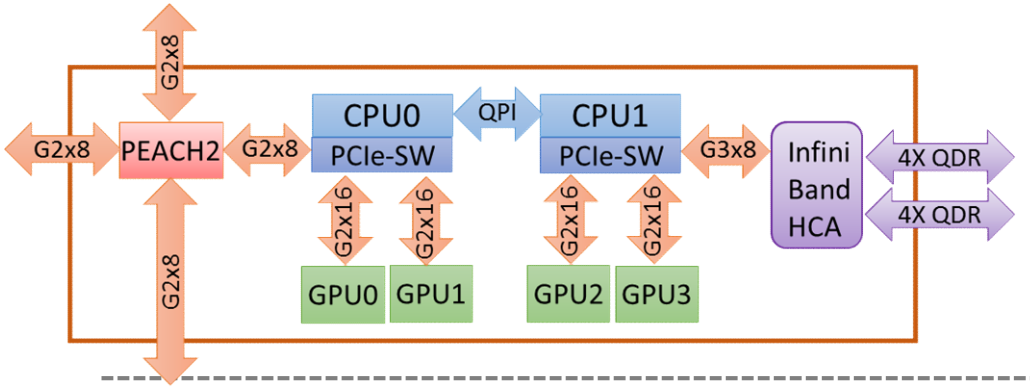
  ⇛分割サイズの調整を一定回数に制限

- 有効範囲は限定的 (8KB~64KB)
- １チャネルと比べて最大1.4倍のバンド幅



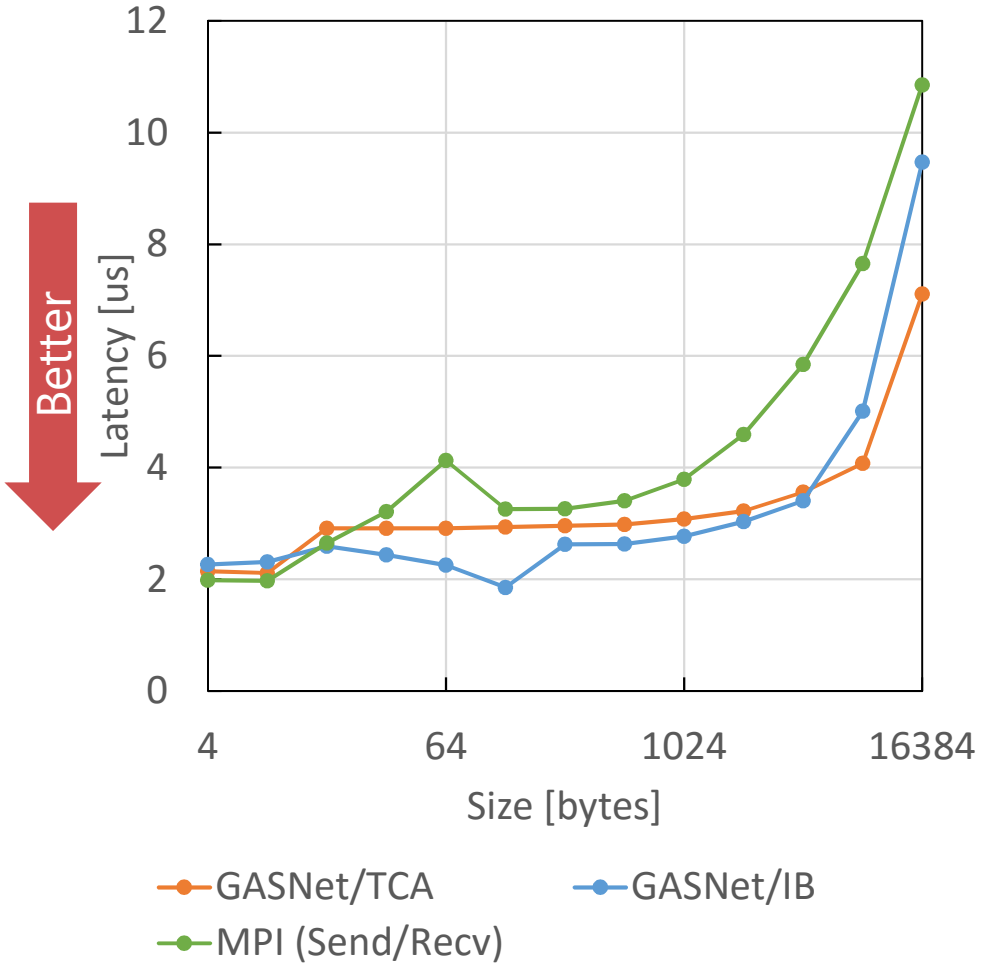
評価環境: HA-PACS/TCA

- HA-PACS/TCA
  - TCA/PEACH2の実験用システムとして，筑波大学計算科学研究センターで運用中
- QPIを跨がない構成で実行
  - PEACH2：CPU0＋GPU0
  - InfiniBand：CPU1＋GPU2
- IB HCAは1ポートのみ使用
  - PCIe Gen2 x8 ≒ QDR 1port



Hardware	
CPU	Intel Xeon-E5 2680v2 2.8GHz×2 (IvyBridge)
Memory	DDR3 1866MHz 128GB
GPU	NVIDIA Tesla K20X ×4
InfiniBand	Mellanox ConnectX-3 4X QDR Dual-port (PCIe Gen3 x8: 8GB/s)
PEACH2	Altera Stratix IV GX (PCIe Gen2 x8: 4GB/s)
Software	
OS	CentOS 6.4
CUDA	CUDA 6.5
MPI	MVAPICH2-GDR 2.1a

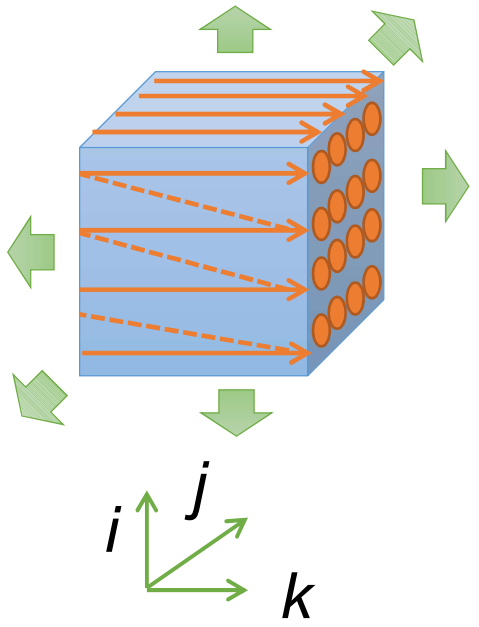
GPU対応GASNetの基本通信性能



- *AM_Long()*によるPing-Pong通信（GPU to GPU）
- 最小レイテンシ
  - GASNet/TCA: 2.1us
  - GASNet/IB: 1.9us
  - MPI: 2.0us
- 32byte以上ではMPIよりも低い
- 8KB以上ではGASNet/IBよりも低い
  - 複数DMAC利用による性能向上

袖領域通信の性能評価

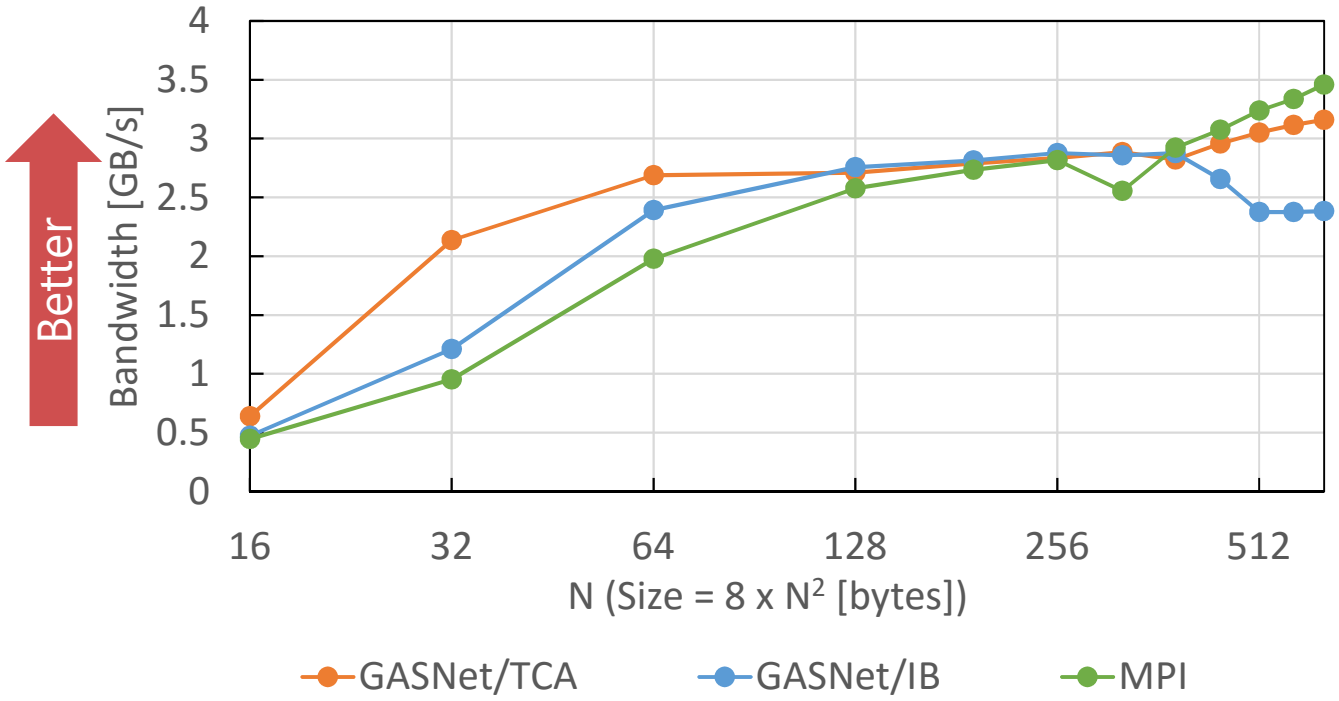
- N×N×Nの３次元配列の各面を転送
- 転送パターンはGPU to GPU



- j-k平面：ブロック転送
  - N×N要素の転送を1回
- i-k平面：ブロックストライド転送
  - N要素の転送をN回
- i-j平面：ストライド転送
  - 1要素の転送をN×N回

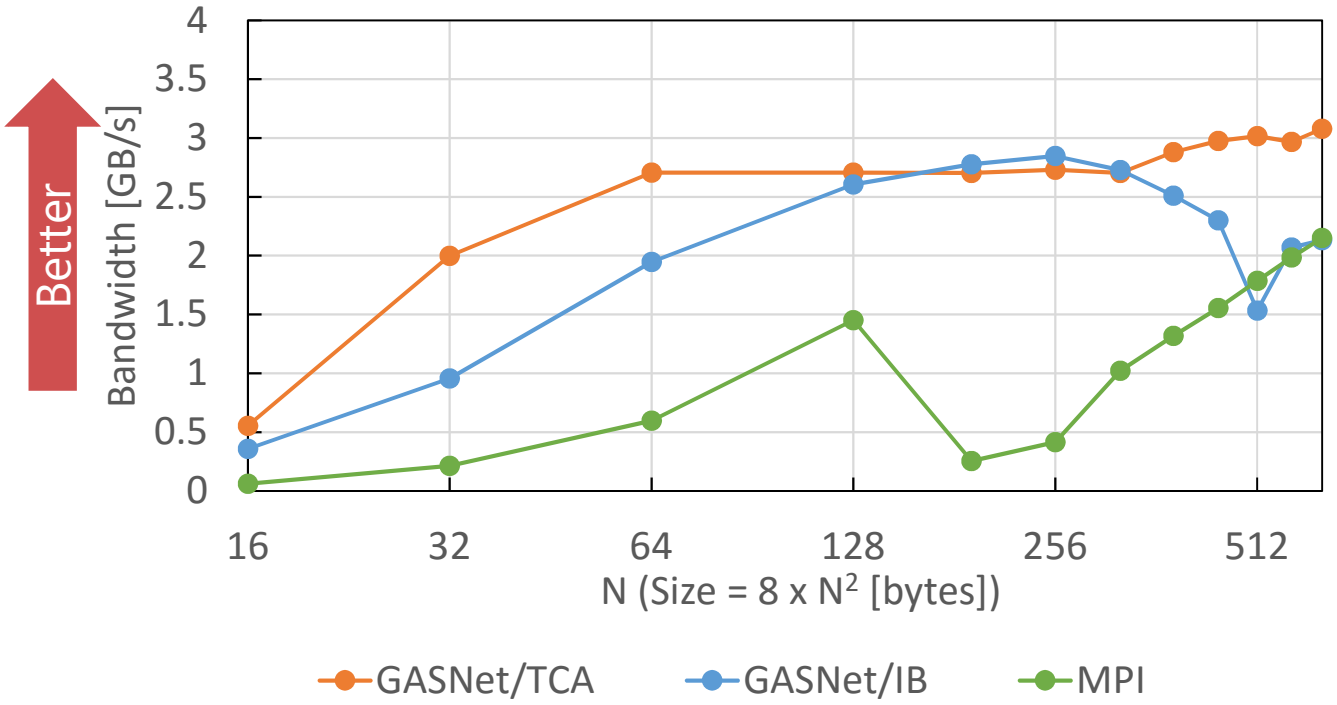
- MPI: *MPI_Type_vector()* + Send/Recv
- GASNet: *put() / puts()* + *AM_Short()*

袖領域通信の性能 (ブロック転送)



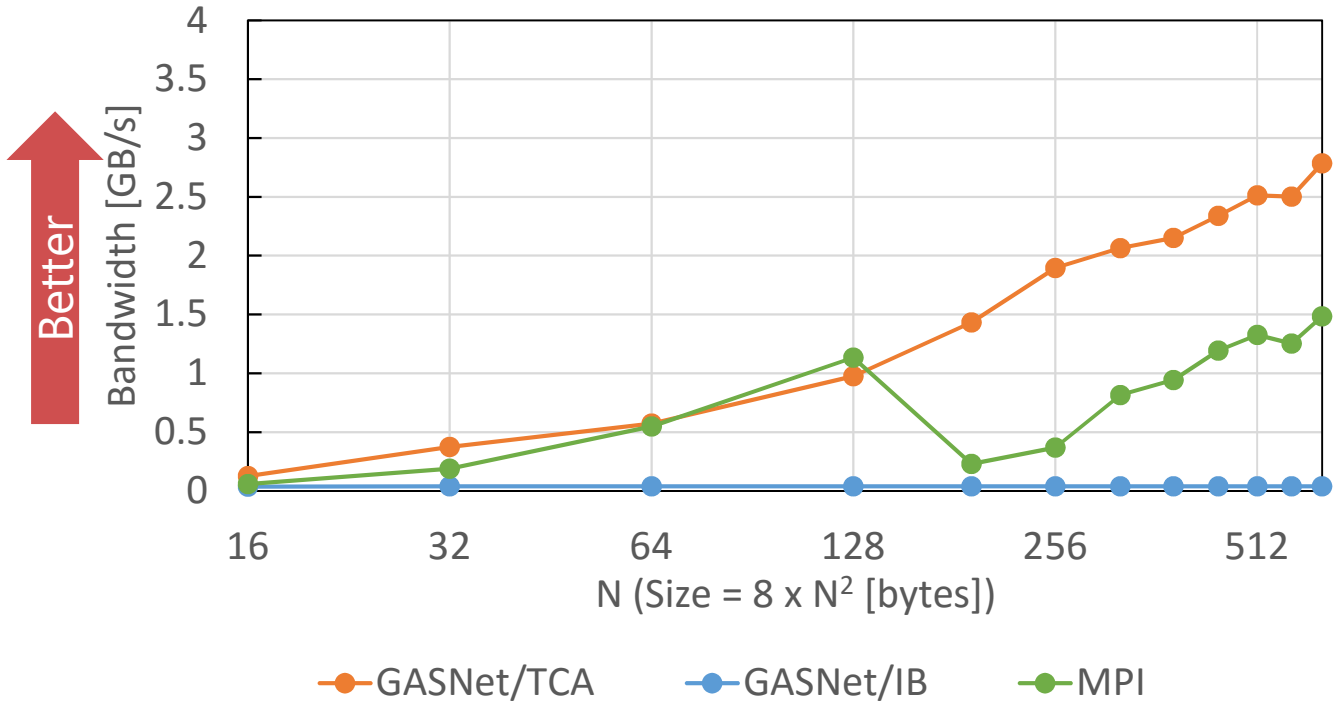
- 最大バンド幅
  - GASNet/TCA: 3.2GB/s, GASNet/IB: 2.8GB/s MPI: 3.5GB/s
- TCAのAPIを直接利用する場合 (2.9GB/s)よりも高い
  - CPUメモリ経由の転送による性能改善
  - GASNet/IBはN=384~でバンド幅が低下
- 複数DMACの利用によって立ち上がりが早い (N=32~64)

袖領域通信の性能 (ブロックストライド転送)



- GASNet/TCAではブロック転送と同程度の性能
  - PEACH2のブロックストライド転送機能の効果
- N=32では
  - GASNet/IBの2.1倍の性能
  - MPI/IBの9.5倍の性能

袖領域通信の性能 (ストライド転送)



- ブロックストライド転送と比べると性能が悪い
  - ブロックストライド転送機能を使わずにPack/Unpackによって転送している
  - GASNet/IBやMPIよりは性能が高い

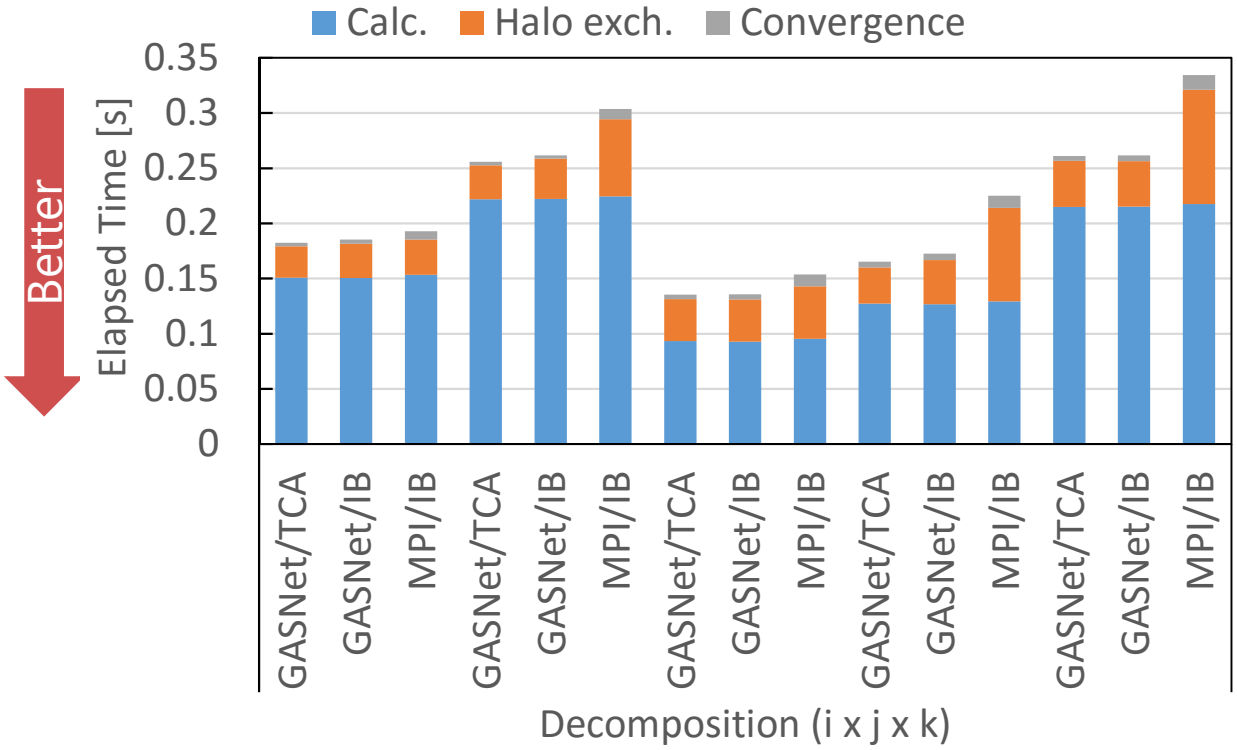
姫野ベンチマーク

- CUDA版姫野ベンチマークを実装
  - 現時点でGPU対応GASNetに対応したPGAS言語処理系は存在しないため，GASNetのAPIで直接実装を行う
- MPI版の通信部分をGASNetのAPIに置換
  - 袖領域通信は*put(), puts()*を使用
  - 収束判定のAllreduceは*AM_Medium()*を用いて実装

- 問題サイズはSmall (64x64x128)
- *i*次元と*j*次元で分割
- 反復回数を固定して処理時間を測定

姫野ベンチマークの結果

- 全ての分割パターンでMPIよりも高性能
- GASNet/IBと比べるとほぼ同性能
  - PEACH2の不具合に起因するオーバヘッドが原因（後述）



姫野ベンチマークについての考察

- DMACの連続起動に問題がある
  - DMA通信の完了通知が不完全
    - 通知を受け取ったタイミングでDMACが完全に停止していない
  - 現在はPEACH2の制御レジスタ(Perf. Counter)をポーリングして停止を検出している
    - PCIe経由のためオーバヘッドが大きい
    - 各DMAC毎に行うため複数DMACを利用する場合に影響が大きくなる

⇒PEACH2はFPGA実装のため，今後修正予定

まとめ

- LBNLで開発中のGPU対応GASNetをTCA/PEACH2によって実装し，性能評価を行った

- 各種最適化やPEACH2のハードウェアの有効活用
  - GPU to GPUの転送ではTCAのAPIを直接利用場合よりも高いバンド幅が得られた
  - ブロックストライド転送ではIB版の実装と比べて最大2.1倍，MPIと比べて最大9.5倍の性能が得られた
  - 姫野ベンチマーク
    - MPIと比べると全ての分割パターンで高性能
    - GASNet/IBと比べると同性能
      - PEACH2の不具合に起因するオーバヘッドが原因

- 今後の課題
  - FPGAの問題を修正
  - FPGAをさらに活用
    - Collective通信のハードウェアオフローディング