Vol.2014-ARC-213 No.32
Vol.2014-HPC-147 No.32
2014/12/10

HA-PACS/TCAにおける TCAおよびInfiniBandハイブリッド通信

小田嶋 哲哉[1,a)]　塙 敏博[3]　児玉 祐悦[1,2]　朴 泰祐[1,2]　村井 均[4]　中尾 昌広[4]　佐藤 三久[1,2]

概要：近年，HPC の分野では GPU を搭載した GPU クラスタが広く利用されている．しかし，ノードを跨ぐ GPU 間通信は，一度ホストメモリを経由して行うためレイテンシが増加し，アプリケーションの性能ボトルネックとなっている．筑波大学計算科学研究センターでは GPU クラスタである HA-PACS の拡張部として，ノード間および GPU 間を直接結合し，レイテンシ・バンド幅の改善を目的に密結合並列演算加速機構 TCA (Tightly Coupled Accelerators) を開発している．TCA は，ハードウェアの制限や実装効率の面で数十ノードを一組とするサブクラスタにとどまるが，今後，より大きな問題やサイズに対応するためにサブクラスタを跨いだ通信が必要とされている．本稿では，TCA と InfiniBand のハイブリッド通信により，サブクラスタ間通信を実現するだけでなく，TCA の低レイテンシ通信と InfiniBand の高バンド幅がそれぞれを効果的に使い分けることによって，より高速な GPU 間通信を実現する．その結果，3 次元配列の袖領域交換において，TCA のみのネットワークに対して約 1.2 倍，InfiniBand のみのネットワークに対して約 1.4 倍の高速化を実現することを示す．

1. はじめに

近年，GPU (Graphics Processing Unit) の持つ高い演算性能とメモリバンド幅に注目し，これを画像処理以外の汎用計算に用いる GPGPU (General-Purpose computation on GPU) が広く利用されている．TOP500[1] の上位には，GPU を搭載したいわゆる GPU クラスタが数多く出現するようになった．しかし，その高い演算性能とメモリバンド幅と比較して，GPU を接続する PCI Express (以降 “PCIe”) の通信性能は非常に低く，特に GPU 間でのデータの交換を行う際に大きなボトルネックになっている．これに加え，従来，ノード間を跨ぐ GPU 間のデータの交換には，ホストメモリを経由して行う必要があり，特にメッセージサイズが小さい時にはレイテンシが大きな問題となり，性能低下を引き起こす原因となっている．ステンシル計算の袖領域交換など，GPU 間のデータ交換が頻繁に必要なアプリケーションで強スケーリングを求める場合，並列度を上げると通信サイズが小さくなり，レイテンシがより性能に影響してくる．

そこで，筑波大学計算科学研究センターではノード間にまたがる GPU 間を直接結合し，レイテンシの改善を図るために密結合並列演算加速機構 TCA (Tightly Coupled Accelerators) を開発している．TCA を用いたアプリケーションでは，低レイテンシ通信により性能が向上している [2], [3], [4].

また，TCA の適用範囲は PCIe 等のハードウェア面の制約によりサブクラスタと呼ばれる数十ノードまでの直接結合にとどまる．今後，より大きな問題やサイズに対応するために，サブクラスタを跨ぐ通信を検討する必要がある．そこで，本研究では，TCA と InfiniBand によるハイブリッド通信 (以降これを単に “ハイブリッド通信” と呼ぶ) を実現し，より高いスケーラビリティを得ることを目的とする．ハイブリッド通信は，単に通信路が増えることだけではなく，TCA の低レイテンシ通信と，InfiniBand の高バンド幅通信を活かし，それぞれの利点が活かせるようにデータサイズや通信方法を選択することによって，TCA と InfiniBand だけでは得られない性能向上を期待している．

本稿では，2 次元，3 次元のステンシル計算における袖領域交換を想定し，TCA および InfiniBand+MPI それぞれの性能に基づき，ハイブリッド通信を行い，その有効性と性能について評価を行う．

[1] 筑波大学 大学院 システム情報工学研究科
Graduate School of System and Information Engineering, University of Tsukuba
[2] 筑波大学 計算科学研究センター
Center for Computational Sciences, University of Tsukuba
[3] 東京大学 情報基盤センター
Information Technology Center, The University of Tokyo
[4] 理化学研究所 計算科学研究機構
RIKEN Advanced Institute for Computational Science
[a)] odajima@hpcs.cs.tsukuba.ac.jp

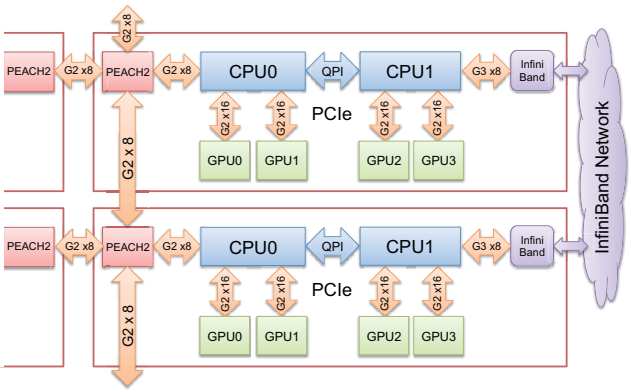

図 1 HA-PACS/TCA におけるノード構成 (文献 [2] より引用)

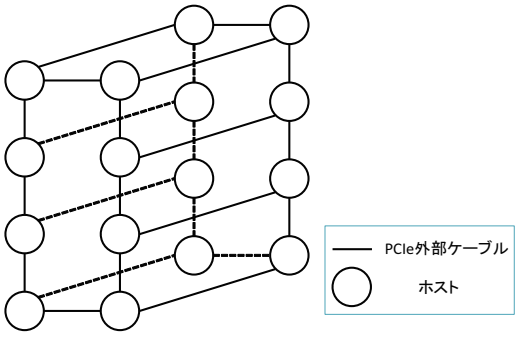

図 2 サブクラスタ：TCA ネットワーク構成

2. TCA アーキテクチャと PEACH2

TCA アーキテクチャおよび PEACH2 については文献 [2], [5], [6], [7] に詳しいが，ここではその概要を説明する．

2.1 TCA

筑波大学計算科学研究センターを中心に，ノード間のアクセラレータ (GPU) 同士を直接結合し，アクセラレータ間通信のレイテンシを改善することを目的とした，密結合並列演算加速機構：TCA (Tightly Coupled Accelerators) を開発している．現在の TCA 実装は PCIe 実装に基づいており，PCIe によって GPU などのアクセラレータが接続されるため，あらゆるアクセラレータを対象とできる．TCA は，ノード間の PCIe を接続することによってアクセラレータ間直接通信が可能になり，低レイテンシ通信を実現する．

同時に，TCA 用インターフェースボードとして PEACH2 (PCI Express Adaptive Communication Hub version 2) ボードが開発されている．PEACH2 ボード同士を接続することで TCA システムを構成する．TCA および PEACH2 の実験用クラスタとして，筑波大学計算科学研究センターの GPU クラスタ HA-PACS (Highly Accelerated Parallel Advanced system for Computational Sciences) [8] の拡張部 HA-PACS/TCA が現在運用されている．

図 1 に，HA-PACS/TCA のノード構成を示す．HA-PACS/TCA のノードは，2 ソケットの CPU Intel Xeon E5-2680v2 と 4 枚の GPU NVIDIA K20X (Kepler アーキテクチャ) を搭載している．CPU0 側には PEACH2 ボードが，CPU1 側には InfiniBand HCA が接続されている．InfiniBand は HA-PACS/TCA のすべてノードが Fat-tree で接続されている．一方，TCA のみで大規模なクラスタを構成することは，外部接続ケーブル長の限界や，性能面での制約により困難であるため，**図 2** のように 16 ノードを TCA で結合する．この組をサブクラスタと呼ぶ．HA-PACS/TCA では，64 ノードが 4 つのサブクラスタに分かれている．しかし，同時に，64 ノードすべてが InfiniBand によっても結合されている．このため，あるノードから見ると，TCA で直接通信可能なノードは 15 台あり，それらを含めたすべてのノードとは InfiniBand 経由で通信ができる．

2.2 PEACH2 チップ

PEACH2 チップは，PCIe パケットの中継処理や DMA 転送などを行うものであり，FPGA (Altera 社 Stratix IV GX[9]) で実装がされている．このチップは，4 つの PCIe Gen2 x8 ポートを持ち，1 つはホスト CPU と接続し，残り 3 つのポートを隣接ノードの PEACH2 ボードとの接続に使用する．これによって，サブクラスタは 2 重リングトポロジを構成する．PEACH2 には高度な DMA コントローラ (以降 “DMAC”) が搭載されており，高速な DMA や Chaining DMA などが可能である．パケットのバッファとして，FPGA 内蔵のメモリと外付けの DDR3 SDRAM を用いる．

2.3 DMA 通信

PEACH2 では，PIO と DMA の 2 つの通信方式があるが，ここでは DMA 通信のみについて言及する．

PEACH2 には DMAC が 4 つ搭載されており，それぞれ Chaining DMA 機能を持つ．ホスト上であらかじめ，読み込み元，書き込み先の PCIe アドレス，サイズを指定したディスクリプタを作成し，これらをアドレスポインタで連結する．DMA での通信を始めるときは，先頭のアドレスポインタを指定することで連続して DMA 処理を行うことが可能である．連続した領域に対する DMA だけではなく，バースト長やギャップ長を指定することでブロックストライド転送を行うことができる．ブロックストライド転送はステンシル計算などで必要となる隣接ノードとの袖領域交換において頻繁に用いられるが，従来の InfiniBand+MPI では，データを Pack/Unpack して送る必要がある．PEACH2 では Chaining DMA を使うことで，Pack/Unpack が必要無くなり，メッセージ長がある程度小さい場合において高いバンド幅が得られる．PEACH2 の

DMA 通信では，ディスクリプタの登録方法によって 3 つの通信モードを提供している．

レジスタモード

各 DMAC に対して最大 16 個までのディスクリプタが制御レジスタに直接登録できる．これによって，通信開始時にホストのメモリを読みに行くコストが削減でき，単一の DMA 通信または Chaining 数が少ない場合において性能が向上する．

ホストメモリモード

ホスト上に必要なサイズのディスクリプタを作成しておき，通信開始時に必要なディスクリプタをホストメモリから読み出し，通信を行う．必要分のディスクリプタを作成することができるため，Chaining が長い場合に有効である．

内蔵メモリモード

PEACH2 の FPGA に内蔵されているメモリにディスクリプタを登録する．ディスクリプタを読み出すコストがホストメモリモードよりも少ないため通信のレイテンシが短縮される．特にデータサイズが小さいときに優位な性能をもち，256Btye までの通信で約 2.0 μ sec の低レイテンシを実現する．しかし，内蔵メモリの容量には限りがあるため，最大で 1024 個のディスクリプタまでしか登録することができない．

本稿の評価には，ディスクリプタが 1024 個以下の場合は “内蔵メモリモード”，それ以上では “ホストメモリモード” を選択する．通信サイズが小さい場合，内蔵メモリモードはホストメモリモードに比べより低レイテンシであるが，ある程度通信サイズが大きくなるとその差はほとんど無くなる．

2.4 GPUDirect Support for RDMA

PEACH2 では GPU への直接通信を行うために, NVIDIA が提供する GPUDirect Support for RDMA 機能 [10] を用いる（以降 “GDR”）．GDR は，CUDA5.0 以降および Kepler アーキテクチャ世代以降の GPU を用いることで, GPU 上のデバイスメモリを PCIe アドレス空間にマッピングすることができる．同じ PCIe アドレス空間に属する GPU 同士では，ホストメモリを経由することなく直接 PCIe 上でデータの転送が可能になる．TCA では，ノード間にまたがって PCIe アドレス空間を共有することができるため，ノード間の GPU 間直接通信を実現することができる．

現在, 図 1 において, 各ノード内の GPU0,1 と GPU2,3 間の GDR を行う場合 Intel Xeon CPU 間の QPI (QuickPath Interconnect) を経由する必要がある．このときに通信性能が著しく低下してしまう問題 [5] があるため，PEACH2 がアクセスする GPU は CPU0 側の GPU0,1 のみに限定する．

表 1 評価環境（HA-PACS/TCA）

CPU	Intel Xeon E5-2680v2 2.8GHz x 2 sokets
GPU	NVIDIA Tesla K20X x 4
Main memory	DDR3 1866MHz 128GB
GPU memory	DDR5 6GB / GPU
Interconnection	InfiniBand : Mellanox Connect-X3 Dual-port QDR TCA : PEACH2 Board
OFED	Mellanox OFED-2.2-1.0.1
OS	CentOS release 6.4 kernel-2.6.32-358.el6.x86_64
MPI	MVAPICH2-GDR-2.0 MV2_USE_CUDA=1 MV2_NUM_PORTS=2
GPU compiler	CUDA 6.0 NVIDIA Driver 340.32

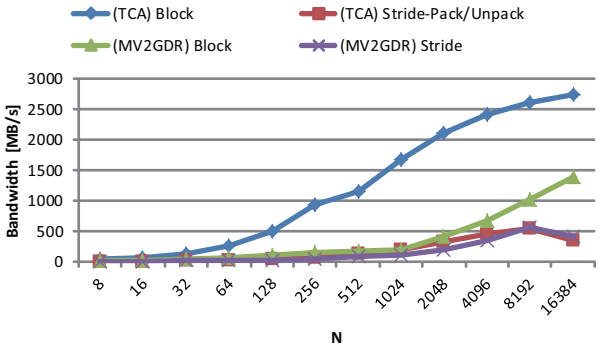

図 3 2D Ping-Pong によるバンド幅

3. GPU 間通信の予備評価

本節では，TCA と InfiniBand の通信性能について議論するために，隣接ノード間 GPU 通信を用いて予備評価を行う．評価環境として，**表 1** に示す筑波大学の HA-PACS/TCA の 2 ノードを用いる．TCA による通信で，Hop 数が増えないように図 2 の TCA ネットワーク上で隣接するようなプロセスを配置する．InfiniBand を利用するための MPI として，オハイオ州立大学が開発している MVAPICH2-GDR[11] (以降 “MV2GDR”) を用いる．MV2GDR は，TCA と同様に NVIDIA の GDR を用いて，InfiniBand を経由した GPU 間直接通信を実現している．2.4 節で触れたように，QPI を経由した GPU 間直接通信は性能が極端に低下してしまう問題があるため，TCA の測定には GPU0 同士，InfiniBand+MPI の測定には GPU2 同士の性能を測定する．

図 3 に，サイズ $N \times N$ の二次元配列に対して 1 行 (Block 転送)，1 列 (Stride 転送) の Ping-Pong の性能をそれぞれ示す．データサイズは 8 Byte の倍精度浮動小数点数である．図 3 より，Block 転送について TCA は MV2GDR に

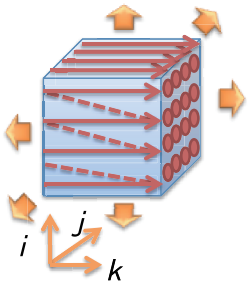

図 4 三次元配列の通信パターン

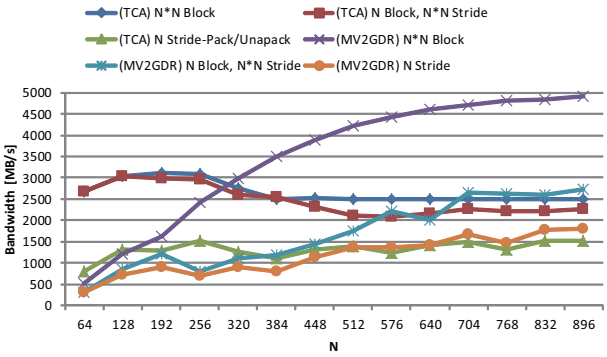

図 5 3D Ping-Pong によるバンド幅

対して非常に高いバンド幅を持っており，$N = 2048$ において 3 倍の性能差を示している．Stride 転送については，TCA では Chaining DMA を利用する事を考えられるが，あまりにも小さいサイズ (ここでは 8 Byte) では，N が増加するとオーバヘッドが見えてしまい，性能があまり上がらないことがわかっている [5]．そこで，Stride 転送では GPU メモリ上で Pack/Unpack を用いて，1 つの配列にまとめたあとに転送する方式を選択する．MV2GDR では，MPI_Type_vector 型を用いて同様に Pack/Unpack してデータ転送を行う．Stride 転送が Block 転送に比べて性能が低いのは，Pack/Unpack のコストが通信時間に対して大きいためである．そのため，Block 転送と比べ，Stride 転送のほうが TCA と MV2GDR の性能は近づき，$N = 4096$ 以上では，Stride 転送の性能が TCA と MV2GDR で逆転する．

図 4 は，3 次元配列の袖領域アクセスを示している．3 次元配列では，jk-平面 ($N \times N$ Block) はメモリアドレスが連続する Block 転送で，ik-平面 (N Block, $N \times N$ Stride) は N 要素の連続アドレスの次に $N \times N$ のギャップ長がある Block Stride 転送である．ij-平面は，1 要素の次に N 要素のギャップ長がある Stride 転送になる．TCA では，Block 転送および Block Stride 転送には Chaining DMA を用い，Stride 転送には 2 次元配列と同様に要素を Pack/Unpack する．MV2GDR では，Block 転送はそのまま Send/Recv をするが，Block Stride 転送および Stride 転送に関しては Pack/Unpack を行う．

表 2 TCA と InfiniBand+MPI 性能分岐点

	性能分岐点のサイズ
2D-Block 転送	16384 まで常に TCA が有利
2D-Stride 転送	8192 ～ 16384
3D-Block 転送	256 ～ 320
3D-Block Stride 転送	512 ～ 576
3D-Stride 転送	512 ～ 576

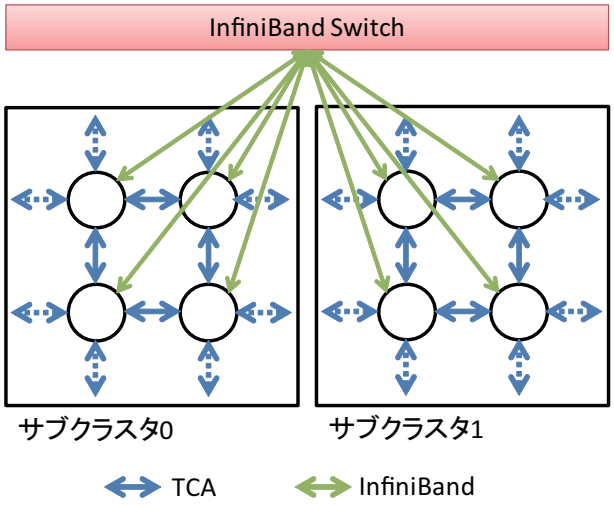

図 6 TCA と InfiniBand による階層ネットワーク

図 5 に，サイズ $N \times N \times N$ の三次元配列の各面のデータを隣接ノードかんで通信する場合の Ping-Pong 性能を示している．Block 転送について，TCA は Chaining DMA の効果により，サイズが小さい場合においても非常に高いバンド幅を示している．TCA と MV2GDR の性能は，$N = 256 \sim 320$ で逆転する．TCA の Block Stride 転送は，Block 転送とほぼ同等の通信性能がある．これより，PEACH2 の Chaining DMA は非常に有用であることがわかる．一方，MV2GDR は Pack/Unpack を行う必要があるため，Block 転送に比べ性能が低く，$N = 512 \sim 576$ と大きなサイズまで TCA の性能が優位である．Stride 転送では，ともに Pack/Unpack が必要であるため全体の性能差は他の方面よりも小さく，$N = 512$ で逆転する．

これらより，**表 2** に TCA と InfiniBand それぞれが優位に働く範囲をまとめ，これを元に次節ではハイブリッド通信について検討する．

4. TCA および InfiniBand ハイブリッド通信

本節では，TCA と InfiniBand ネットワークを併用しサブクラスタにまたがるハイブリッド通信を提案する．

4.1 TCA および InfiniBand ハイブリッド通信の概要

ハイブリッド通信は，ノードに 2 つの異なる Interconnection があることによって実現可能である．本研究では，単に 2 つの通信路を束ねて全体のバンド幅を向上するのではなく，各 Interconnection の特徴に応じた通信チャネ

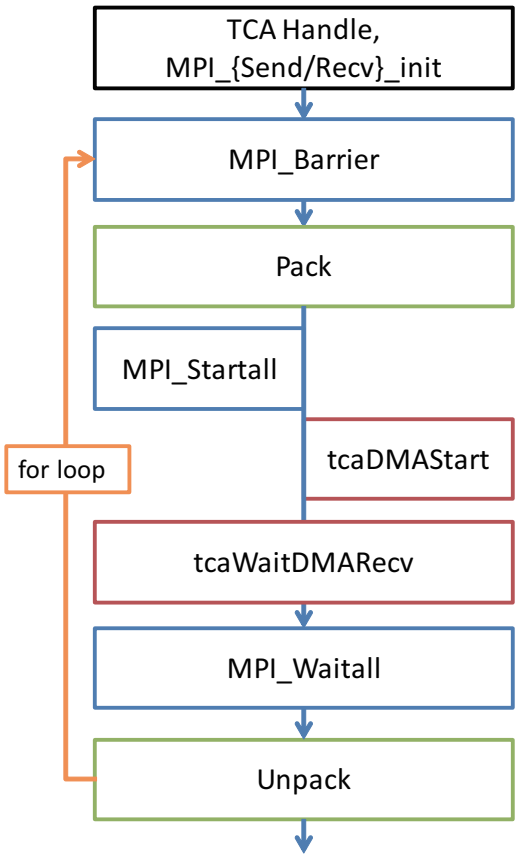

図 7 TCA/InfiniBand ハイブリッド通信の実行フロー

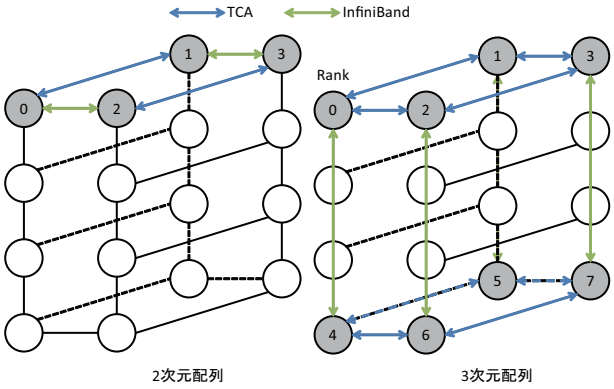

図 8 2 次元配列，3 次元配列における通信方向とノードへのマッピング

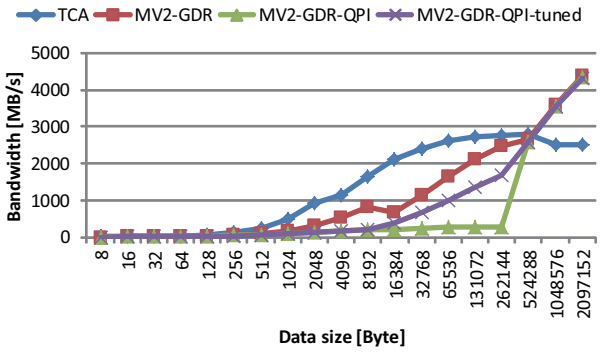

図 9 QPI を経由する事による MV2GDR の Ping-Pong 性能の変化．MV2GDR が CPU1 側の GPU 同士の通信，MV2GDR-QPI が CPU0 側の GPU 同士による通信，MV2GDR-Tuned が 8KB までを GDR による直接通信，それより大きいサイズではホスト経由の通信に切り替えた場合の QPI 越しの性能を示す．

ルを選択することで，通信性能を向上させることが目的である．また，ハイブリッド通信は，袖領域交換など TCA や InfiniBand の一部の通信をそれぞれが補完しあうことで全体の性能向上が得られるだけではなく，サブクラスタを跨いだ集団通信においても有効であると考えている．松本らは，TCA のサブクラスタ内における集団通信を実装し評価を行っている [12]．アプリケーションを強スケーリングさせる際には，集団通信の各メッセージサイズが小さくなるため，InfiniBand+MPI の環境では大きなボトルネックとなってきた．TCA では，強スケーリングした際のメッセージでも高い性能が得られる．そこで，TCA と InfiniBand によるサブクラスタ間通信を組み合わせることで，**図 6** のような階層的なネットワークを構成することができる．集団通信では，各サブクラスタ内では TCA の高速な通信を活かし，最終的にサブクラスタ間の InfiniBand を経由して交換するデータ量は最小限にすることで，全体として低レイテンシ通信を行うことが期待できる．このような階層的な低レイテンシの集団通信は，単に InfiniBand のバンド幅を増やすだけでは実現することはできず，TCA と InfiniBand のハイブリッド通信によって実現できる．

本稿では，まず典型的な袖領域交換についてハイブリッド通信の有効性を示し，階層的な集団通信については今後の課題としたい．

4.2 TCA および InfiniBand ハイブリッド通信による袖領域交換

3 節より，2 次元配列の袖領域交換では Block 転送は TCA が有効に働き，Stride 転送は TCA と InfiniBand で性能差はあまり大きくないということがわかった．そこで，Block 転送に TCA，Stride 転送に MV2GDR を用いることとする．3 次元配列では Block Stride 転送と Stride 転送は N が 512 以上で性能が逆転することがわかった．一方，Block 転送では比較的サイズの小さい部分で逆転が起こるため，Block 転送を MV2GDR に，Block Stride 転送および Stride 転送を TCA に割り当てる．

図 8 に 2 次元，3 次元の通信方向とサブクラスタへのマッピングを示す．本稿では，図 8 よりサブクラスタ内で仮想的にサブクラスタを跨ぐ通信を実現するため，2 次元配列では Rank0,1 と Rank2,3 の組がそれぞれ TCA ネットワークのみで接続され，その間を InfiniBand で結合する．3 次元配列では，Rank0~3 と Rank4~7 の組が TCA のみで接続され，同様にそれらの組み同士を InfiniBand で接続する．

ハイブリッド通信は，**図 7** の流れで実行される．袖領域交換の実装では，動的に袖領域が変わるプログラムでない限り，通信に必要なディスクリプタの登録や Pack/Unpack 用の配列などの確保を予め済ませておく．イテレーションのはじめに MPI を使ってバリア同期を行う．通信方向で 1 要素の Stride 通信がある場合，cudaMemcpy2D 関数を使い事前に用意しておいた通信用のバッファにコピーを行う．その後，MPI_Startall で事前に登録していた MPI 通信を開始し，次に PEACH2 の DMAC に登録しておいたディスクリプタを tcaDMAStart で読み出すことで TCA 通信が開始される．TCA 通信では，各ディスクリプタからリモートへの書き込みが完了した ACK が返ってくるので，それを登録した通信方向の数だけ tcaWaitDMARecv で待つ．同時に MPI 通信も MPI_Waitall で待ち受ける．TCA 通信が完了するまでに MPI 通信が完了してしまった場合，MPI_Waitall でブロックされずに通過するため，1 プロセスによって 2 つの通信の待ち受けを行ってもオーバヘッドにはならないと考えている．そして，各方向からデータを受け取ったあと，必要に応じてデータを Unpack する．ここまでが 1 イテレーションとなる．

5. 性能評価

ハイブリッド通信では，TCA の性能を有効に利用するため，図 1 における GPU0 側の GPU0，1 を対象とする．しかし，MV2GDR によって InfiniBand 経由で GPU 間通信を行う場合，QPI を通る必要がある．そのため，QPI を経由しない MV2GDR よりも性能が低下してしまう．

図 9 に CPU0 側，CPU1 側の GPU 同士で Ping-Pong を行った場合の性能低下を示す．これより，QPI を経由する MV2GDR の通信では 256KB までのバンド幅が最高でも 280MB/s 程度しか出ていない．デフォルトの MV2GDR は，以下のようにデータサイズによって通信の方法を切り替えている．

～ 8KB
GDR による GPU 間直接通信．

8KB ～ 512KB
一度ホストにデータをコピーし，その後 GDR を用いてリモートの GPU へデータ書き込む．

512KB ～
一度ホストへのデータ転送，ホスト間のメッセージパッシングし，リモートノードでのホストからデバイスへのデータ転送をパイプライン処理する．これによって同じデータサイズにおいて GDR を使うよりも性能が向上している．

そのため，本来 QPI を経由する通信は性能が低下してしまうが，512KB からはホストを経由するため “MV2GDR-QPI” の性能は “MV2GDR” と同様になっていることがわかる．詳しくは，MV2GDR の README を参照されたい

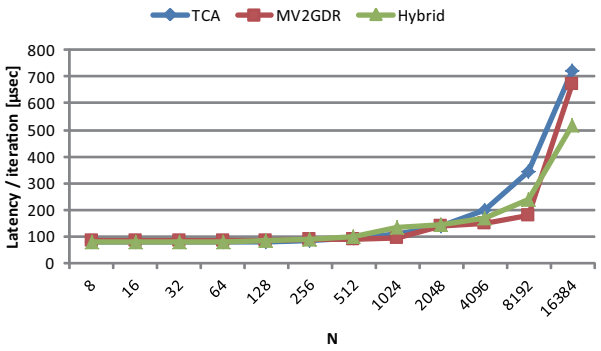

図 10 2 次元配列の袖領域交換．各方向の通信サイズは $N \times sizeof(double)$ となる．

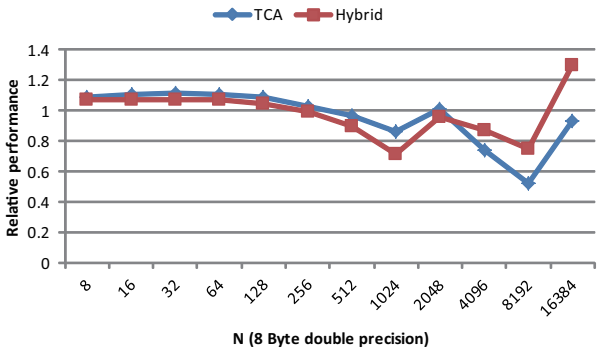

図 11 2 次元配列の袖領域交換：InfiniBand+MPI に対する TCA とハイブリドーマの相対性能．1 より大きいと InfiniBand+MPI よりも高速であることを示す．

が，実行時の環境変数によってホスト経由のコピーを行うようにするデータサイズを切り替えることができる．本稿における測定では，その値を 8KB とするのが最適である事がわかっており，“MV2GDR-Tuned” がこれに該当する．つまり，8KB までは GDR による GPU 間直接通信，それ以上ではホスト経由の通信に切り替わるという選択が MVAPICH2-GDR の中で自動的に行われる．ただし，今後，ハイブリッド通信測定では MV2GDR の性能は InfiniBand+MPI 環境よりも若干劣ることを考慮し，この辺の切り替えポイントはある程度再調整する必要がある．本稿ではこの値のまま測定を行うこととする．

今回の測定では，2 次元と 3 次元配列の袖領域交換を TCA のみ，InfiniBand+MPI (MV2GDR) のみ，ハイブリッド通信それぞれで測定し，1 イテレーションの実行時間を比較する．ノードのマッピングは図 8 に示すとおりである．また，周期境界条件となっており，例えば，2 次元配列の Rank0 は Block 転送を上下方向 (ともに Rank2) に，Stride 転送を左右方向 (ともに Rank1) とデータ交換を行う．

まず，2 次元配列の袖領域交換の実行時間を**図 10** に，InfiniBand+MPI に対する性能向上を**図 11** に示す．これ

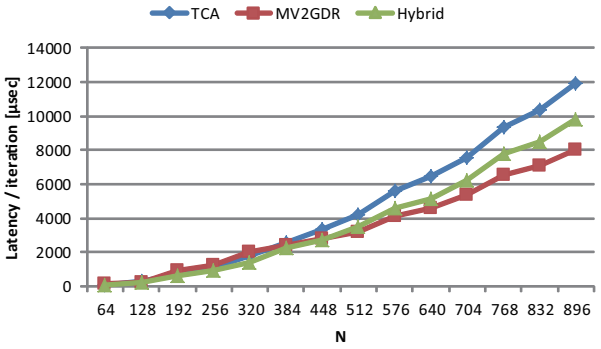

図 12 3 次元配列の袖領域交換．各方向の通信サイズは $N \times N \times sizeof(double)$ となる．

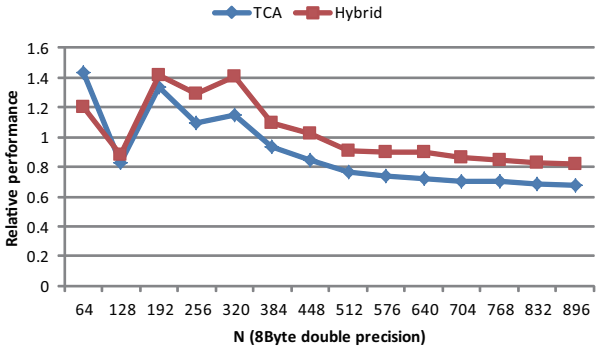

図 13 3 次元配列の袖領域交換：InfiniBand+MPI に対する TCA とハイブリドーマの相対性能．1 より大きいと InfiniBand+MPI よりも高速であることを示す．

より，1 方向に送る要素数が $N = 256$ までは TCA のみで袖領域交換を行うのが高速で，$N = 2048 \sim 8192$ までは MV2GDR が最も高速である．$N = 16384$ では，ハイブリッド通信が最も高速になっている．図 3 の 2 次元配列の Ping-Pong 通信では，Block 転送に関して TCA は MV2GDR に対して大きなアドバンテージがあり，Stride 通信はほぼ同等の性能が得られていた．そのため，TCA が常に優位であると想定されたが，図 10 より，TCA と MV2GDR の性能差が Ping-Pong に比べ小さい事がわかる．この原因として，Ping-Pong 通信では PEACH2 が GPU からデータを読み出すアクセスと受けとったデータを書き込むアクセスが別々に発行されていたが，袖領域交換では PEACH2 が GPU からデータを読み出すのと受け取ったデータを書き込むことが同時同時に発生するため，アクセスがシリアライズされてしまい，理想的な実行時間にならなかった可能性がある．このようなオーバヘッドがかさみ，今回の実行時間に至ったと考えられる．ハイブリッド通信では，データを TCA と InfiniBand 経由で送るため，このような TCA 内を経由する総データ量が削減されたため，オーバヘッドが軽減し，結果的に全体の性能向上につながったと考えられる．しかし，現段階では詳細な検証ができていないため，詳しい原因については今後の検討課題とする．

次に，3 次元配列の袖領域交換の実行時間を**図 12** に，InfiniBand+MPI に対する性能向上を**図 13** に示す．3 次元配列の袖領域交換では，図 5 より，特に TCA の Block Stride 転送が非常に高速であることがわかる．一方，3 次元配列の 6 面を転送する必要があり，1 イテレーション当たりの総データサイズが大きくなる．そのため，図 12 では TCA が MV2GDR と比較して大きな差が見られない．$N = 128$ を超えた時からあたりから TCA を利用することによって，高い性能が得られている．特に，ハイブリッド通信においては，TCA だけの通信よりも高い性能であり，TCA の高速な Block Stride 転送とデータ量が多い時に有効な MV2GDR を組み合わせることで，従来では得られなかった速度向上を得ることを示した．これは，PEACH2 が処理を行っていたデータの一部が MV2GDR で処理されたため，TCA のみで通信するときに発生していた PEACH2 の読み出しや書き出しのオーバヘッドが軽減されたことが要因であると考える．MV2GDR に対しては $N = 192 \sim 448$ まではハイブリッド通信が有効である事がわかる．特に，$N = 192$ および $N = 320$ では，同サイズにおけるハイブリッド通信が 40%も高速になっている．このサイズでは，TCA の Chaining DMA による高速なデータ転送が有効に働いていること挙げられる．

6. 関連研究

GPUDirect では，コモディティネットワークの InfiniBand と NVIDIA の Kepler アーキテクチャを搭載した GPU により，GPU 間直接通信を実現している [13]．これを MPI のインタフェースに適用したものとして，オハイオ州立大学の MVAPICH2-GDR がある [10]．PEACH2 でも同様に GDR を用いた通信をするが，InfiniBand の通信には HCA 間の通信には PCIe とは異なるプロトコルを用いている．一方，PEACH2 は PCIe のパケットをそのまま利用できるのでプロトコル変換のオーバヘッドがなく，InfiniBand の GDR に比べて低いレイテンシで処理が可能である．ハイブリッド通信でも InfiniBand を用いるが，すべての通信が必ずしも InfiniBand を経由することはないので，オーバヘッドは低減される．

7. おわりに

本稿では，TCA と InfiniBand によるハイブリッド通信を提案し，それぞれの通信方式が有効に働く範囲を選択することで，2 次元および 3 次元配列の袖領域交換の評価を行った．ハイブリッド通信による袖領域交換は，特に TCA の Block Stride 転送が有効に働いていることがわかり，TCA，InfiniBand+MPI の環境と比較してそれぞれに対して最大で 1.2 倍，1.4 倍の性能向上が得られた．

今後の課題として，データサイズや通信手法を変化させることでより最適化が必要である．そのために，それぞれの通信性能を元にモデル化を考えている．今回は袖領域交換のみに焦点を当てていたが，集団通信を含めたハイブリッド通信についても検討し，有効性を示していきたい．

また，現在，村井，中尾らによってアクセラレータ向けのプログラミング言語 XcalableACC が開発されている [14]．そのなかで，TCA が GPU 間のデータ転送に用いられているが，データサイズによって通信路を TCA，GDR，ホスト経由による通信と切り替えるような実装となっている．我々は，この XcalableACC のフレームワークの中に TCA および InfiniBand ハイブリッド通信を組み込むことによって，プログラミングコストの減少と通信性能の向上を目的として開発を行っていく予定である．

謝辞

本研究の一部は，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「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」による．

参考文献

[1] Top500 Supercomputer Sites. http://top500.org/.
[2] 塙敏博, 児玉祐悦, 朴泰祐, 佐藤三久. Tightly Coupled Accelerators アーキテクチャに基づく GPU クラスタの構築と性能予備評価. 情報処理学会論文誌. コンピューティングシステム, Vol. 6, No. 4, pp. 14–25, Oct. 2013.
[3] 藤井久史, 藤田典久, 塙敏博, 児玉祐悦, 朴泰祐, 佐藤三久, 藏増嘉伸, MikeClark. GPU 向け QCD ライブラリ QUDA の TCA アーキテクチャ実装の性能評価. 情報処理学会研究報告 (ハイパフォーマンスコンピューティング), Vol. 2014, No. 43, pp. 1–9, Jul. 2014.
[4] N. Fujita, H. Fujii, T. Hanawa, Y. Kodama, T. Boku, Y. Kuramashi, and M. Clark. QCD Library for GPU Cluster with Proprietary Interconnect for GPU Direct Communication. In *12th The International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms (HeteroPar2014) in conjunction with Euro-Par2014*, Aug. 2014.
[5] 塙敏博, 児玉祐悦, 藤井久史, 朴泰祐, 佐藤三久. HA-PACS/TCA システムにおけるマルチノード GPU 間通信性能評価. 情報処理学会研究報告 (計算機アーキテクチャ), Vol. 2014, No. 20, pp. 1–8, Jan. 2014.
[6] T. Hanawa, Y. Kodama, T. Boku, and M. Sato. Interconnection Network for Tightly Coupled Accelerators Architecture. In *IEEE 21st Annual Symposium on High-Performance Interconnects (HOT Interconnects 21)*, pp. 79–82, Aug. 2013.
[7] T. Hanawa, Y. Kodama, T. Boku, and M. Sato. Tightly Coupled Accelerators Architecture for Minimizing Communication Latency among Accelerators. In *The Third International Workshop on Accelerators and Hybrid Exascale Systems (AsHES) in conjunction with IEEE 27th International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1030–1039, May 2013.
[8] 朴泰祐, 佐藤三久, 塙敏博, 児玉祐悦, 高橋大介, 建部修見, 多田野寛人, 藏増嘉伸, 吉川耕司, 庄司光男. 演算加速装置に基づく超並列クラスタ HA-PACS による大規模計算科学. 情報処理学会研究報告 (ハイパフォーマンスコンピューティング), Vol. 2011, No. 21, pp. 1–7, Jul. 2011.
[9] Altera Corporation. Stratix IV Device Handbook. http://www.altera.co.jp/literature/lit-stratix-iv.jsp.
[10] NVIDIA Corporation. NVIDA GPUDirect. https://developer.nvidia.com/gpudirect.
[11] MVAPICH2: A High Performance MPI Library for NVIDIA GPU Clusters with InfiniBand. http://on-demand.gputechconf.com/gtc/2013/presentations/S3316-MVAPICH2-High-Performance-MPI-Library.pdf.
[12] 松本和也, 塙敏博, 児玉祐悦, 藤井久史, 朴泰祐. 密結合並列演算加速機構 TCA を用いた GPU 間直接通信による CG 法の実装と予備評価. 情報処理学会研究報告 (ハイパフォーマンスコンピューティング), Vol. 2014, No. 12, pp. 1–9, May 2014.
[13] Mellanox Technologies: Mellanox OFED GPUDirect. http://www.mellanox.com/page/products_dyn?product_family=116.
[14] 中尾昌広, 村井均, 下坂健則, 田渕晶大, 塙敏博, 児玉祐悦, 朴泰祐, 佐藤三久. XcalableACC:OpenACC を用いたアクセラレータクラスタのための PGAS 言語 XcalableMP の拡張. 情報処理学会研究報告 (ハイパフォーマンスコンピューティング), Vol. 2014, No. 7, pp. 1–11, Sep. 2014.