Vol.2015-HPC-150 No.42
2015/8/6

密結合演算加速機構TCAにおけるVerbs実装によるMPI環境の実現

佐藤 賢太[1,a)]　藤田 典久[1]　塙 敏博[3]　朴 泰祐[1,2]

概要：近年，GPU のような演算加速装置を用いたクラスタが HPC 分野で多く用いられるようになってきている．筑波大学計算科学研究センターでは，ノードを跨ぐ演算加速装置間での直接通信を実現するために，密結合並列演算加速機構 TCA（Tightly Coupled Accelerators）を提唱しており，その実装として PEACH2（PCI Express Adaptive Communication Hub version 2）が開発されている．しかしながら，TCA/PEACH2 を利用するためには独自の API を用いる必要があり，プログラミングコストが高く既存のアプリケーションの移植も容易ではない．本稿では，TCA/PEACH2 によって InfiniBand の通信インタフェースである Verbs を実装することによって，MPI の動作を実現する．その結果，通信レイテンシの増大は避けられなかったが，MPI の利用が可能となり，最大バンド幅に関しては PEACH2 とほぼ等しい性能が得られた．また，Verbs はその他の InfiniBand 系対応の多くのソフトウェアに標準的に用いられており，本研究によって TCA/PEACH2 をそれらのシステムに容易に適用可能となると見込まれる．

1. はじめに

近年，GPU のような高い演算性能とメモリバンド幅を持つ演算加速装置を用いたクラスタが HPC 分野で広く用いられるようになってきている．Top500 List[1] においても，2014 年 11 月のリストで全体の約 10%のシステムが演算加速装置として GPU を利用している．一般に，GPU は PCIe（PCI Express）バスを介して CPU やノード内の他の GPU と接続されている．しかしながら，PCIe の転送バンド幅は GPU のメモリバンド幅よりも低く，GPU アプリケーションにおいてボトルネックとなることも多い．また，ノードを跨ぐ GPU 間の通信には IB（InfiniBand）などのネットワークを介する必要があり，PCIe と IB 間でのプロトコル変換などが必要となるため，通信レイテンシが増大し，強スケーリングの達成が困難となる．

そこで，筑波大学計算科学研究センターでは，ノードを跨ぐ演算加速装置間での直接通信を実現するために，密結合並列演算加速機構 TCA（Tightly Coupled Accelerators）を提唱しており，その実装として，PEACH2（PCI Express Adaptive Communication Hub version 2）を開発している [2]．TCA は演算加速装置間を直接的通信網で結合するというコンセプトであり，PEACH2 はこれを FPGA を用いて PCIe を対象として実装したプロトタイプである．以後，PEACH2 による TCA 実装を TCA/PEACH2 と記す．現在，TCA/PEACH2 を利用するためには独自の API を用いる必要があり，通信プリミティブとして RDMA Write しか提供されていないため，MPI と比べてプログラミングコストが高く，既存のアプリケーションの移植が容易ではないという問題がある．

高性能 PC クラスタでは，ノード間ネットワークとして IB が広く用いられており，対応した MPI 実装も数多く開発されている．IB における通信インタフェースとして Verbs が InfiniBand Trade Association[3] によって策定されている．そして，MPICH[4] や MVAPICH2[5]，OpenMPI[6]，IntelMPI[7] などの IB に対応した MPI 実装では，Verbs を用いて IB HCA を操作している．従って，TCA/PEACH2 上に Verbs インタフェースを実装することができれば，これらの MPI 環境において TCA/PEACH2 を容易に利用することが可能となる．さらに，Verbs に対応した IB 系システムソフトウェアの導入も同様に容易になると予想される．

以上の背景の下，本研究では，TCA/PEACH2 による Verbs の実装として TCA Verbs を開発する．これによって，TCA/PEACH2 で IB と同一のプログラムの動作が可能となり，IB 用に開発された MPI 実装を利用することが

[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
[a)] ksato@hpcs.cs.tsukuba.ac.jp

可能となる．その上で，Ping-Pong 通信および姫野ベンチマークによって TCA Verbs の通信性能の評価を行う．

2. TCA/PEACH2

本節では，本研究を理解するための TCA および PEACH2 について概要を説明する．これらの詳細については [2] を参照されたい．

2.1 TCA

筑波大学計算科学研究センターでは，ノードを跨ぐ演算加速装置間での直接通信を実現するために，密結合並列演算加速機構 TCA（Tightly Coupled Accelerators）を提唱している．TCA は，ノードを跨ぐ演算加速装置同士を密に結合することで，演算加速装置間での直接通信を可能とし，通信レイテンシを削減することで強スケーリングを達成するというものである．

2.2 PEACH2

TCA の実装として，PEACH2（PCI Express Adaptive Communication Hub version2）が開発されており，ノードを跨ぐ CPU 間および GPU 間での直接通信を行うことができる．現状では，NVIDIA 社製 GPU に対応している．PEACH2 では，ノード間の接続に PCIe を用いており，PEACH2 が PCIe パケットのルーティングを行うことでノードを跨ぐ PCIe デバイス間での直接通信を実現している．

PEACH2 は FPGA 上に実装されており，PCIe パケットのルーティングや DMA 転送などがハードワイヤードロジックによって処理される．FPGA を用いているため，ASIC と比べて動作周波数の面では不利だが，回路を柔軟に書き換えられるため，機能改善や不具合の修正・改良などが容易に行えるという利点がある．PEACH2 は 4 つの PCIe ポートを持っており，そのうち 1 ポートをホスト CPU との接続に使用し，残り 3 ポートを他のノードの PEACH2 との接続に使用する．PCIe インタフェース部は FPGA に内蔵されている 4 基のハード IP を使用しており，それぞれ PCIe Gen2 x8 で動作する．また，ノード内の PCIe デバイスには，何らかの PCIe スイッチで接続が可能であるが，Intel Sandy Bridge アーキテクチャ以降の CPU では内部に PCIe Gen3 対応スイッチが内蔵されているため，これを用いることを一般的実装と想定している．PEACH2 による GPU メモリへのアクセスには，NVIDIA Kepler アーキテクチャおよび CUDA[8] 5.0 からサポートされた GPUDirect RDMA（GDR）[9] を利用している．

一般的な CUDA プログラミングでは，MPI による通信の指示はホスト CPU で行われるが，PEACH2 を用いるプログラミングにおいても同様に，通信の設定や開始指示はホスト CPU で行う．そして，通信の開始を指示した後は，PEACH2 によって直接データの転送が行われる．

2.3 DMA 転送

PEACH2 には，DMAC（DMA Controller）が 4 チャネル実装されている．DMA 転送は，ホスト上であらかじめ転送元アドレスや転送先アドレス，転送サイズなどを指定したディスクリプタを作成しておき，使用する DMAC にこれを登録することで行われる．また，PEACH2 の DMAC は Chaining 機能を持っており，複数のディスクリプタを連結することで，連続した DMA 転送を 1 回の DMA 要求で開始することができる．ただし，ディスクリプタの作成には時間が掛かるため，できる限り事前に作成しておき，以降は事前に作成しておいたディスクリプタを DMAC に登録するだけに留めるのが望ましい．また，PEACH2 では通常の連続アドレス上のデータのブロック転送の他，ブロックストライド転送も実行可能である．よって，定型的なブロックストライド転送は DMA Chaining を用いなくても実行可能であり，さらに DMA Chaining との組み合わせも可能である．

ディスクリプタを格納する場所として，FPGA に内蔵されたメモリ（内蔵メモリモード）と，ホスト CPU のメモリ（ホストメモリモード）がある．内蔵メモリモードでは，ディスクリプタを読み出すコストが低いため，ホストメモリモードよりも通信レイテンシが低くなる．しかしながら，メモリ容量には限りがあるため，最大で 1024 個のディスクリプタしか格納できない．また，ディスクリプタを作成する度に，PEACH2 にアクセスする必要があるため，ディスクリプタ作成に掛かる時間が大きい．一方，ホストメモリモードは通信レイテンシが若干増えるものの，ホストのメモリ容量の許す限りディスクリプタを格納することができ，またディスクリプタの作成はホスト内で完結するため，作成に掛かる時間が小さい．このように，各モードで特性が異なるため，通信パターンなどに応じて適切に選択することで最適化が可能である．

PEACH2 の DMAC は DMA 転送として，RDMA Write のみ対応している．そのため，RDMA Read が必要な場合は，ソフトウェアによるプロキシなどを用意した上で Proxy Write を行う必要がある．この点はアプリケーションおよびシステムソフトウェアの実装および性能上，大きな影響を及ぼすので注意が必要である．

2.4 PEACH2 の制限事項

PEACH2 にはハードウェアなどの制約から以下のような制限事項が存在する．

- 転送元と転送先のアドレスは 4byte 境界に揃える必要がある．
- 転送元と転送先のアドレスは下位 4bit が一致している必要がある．

- CPU メモリを通信対象とする場合，そのメモリは専用の API（*tcaMalloc()*）で確保されたものに限定される．

2.5 HA-PACS/TCA

筑波大学計算科学研究センターでは，TCA/PEACH2 の実験用システムとして，HA-PACS/TCA（Highly Accelerated Parallel Advanced System for Computional Sciences/TCA）が運用されている [10]．表 1 および図 1 にノード構成を示す．本稿における今後の性能評価は，このクラスタを用いて行う．

表 1　HA-PACS/TCA のノード構成

ハードウェア	
CPU	Intel Xeon-E5 2680v2 2.8GHz × 2 sockets
Memory	DDR3 1866MHz × 4ch，128GB
GPU	NVIDIA Tesla K20X × 2
GPU Memory	GDDR5 2600MHz，6GB/GPU
InfiniBand	Mellanox ConnectX-3 QDR 4X Dual-port
PEACH2	Altera Stratix IV GX，EP4SGX530
ソフトウェア	
OS	CentOS 6.4
CUDA	CUDA 6.5
OFED	MLNX_OFED_LINUX-2.2-1.0.1 libibverbs-1.1.7mlnx1

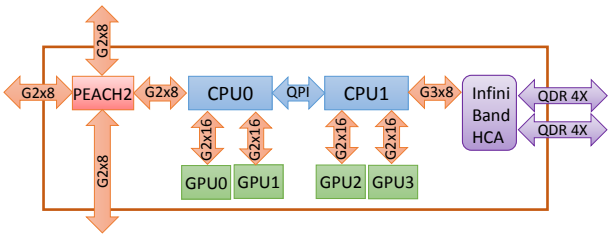

図 1　HA-PACS/TCA におけるノード構成

2.6 PEACH2 の基本性能

今後の性能評価の基準となる，PEACH2 の基本的な通信性能を図 2 および図 3 に示す．図中の凡例における “Internal” および “Host” は，DMA ディスクリプタの配置の違いで，前者は内蔵メモリモード，後者はホストメモリモードをそれぞれ示している．また，括弧内は送信元と送信先を示している．例えば，“Internal (GPU)” は，GPU メモリから別ノードの GPU メモリへのデータ転送を DMA ディスクリプタを PEACH2 内蔵メモリに置いた場合を示している．

図 2 より，PEACH2 の最小レイテンシはメモリの種別に関わらず，内蔵メモリモードで 2.0μs，ホストメモリモードで 2.2μs である．また，図 3 より，最大バンド幅はモードに関わらず，CPU メモリの場合で 3.5GB/s，GPU メモリの場合で 2.8GB/s である．

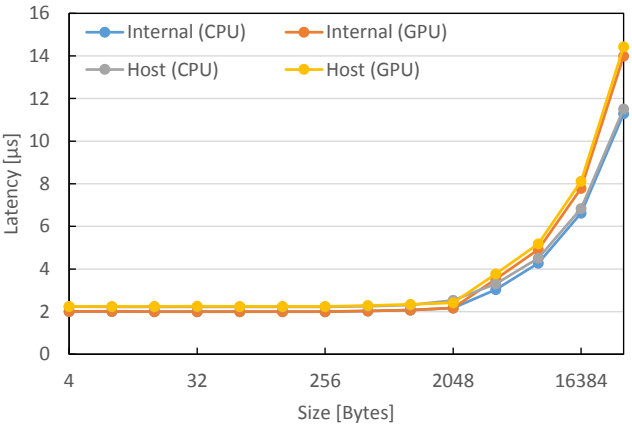

図 2　PEACH2 のレイテンシ

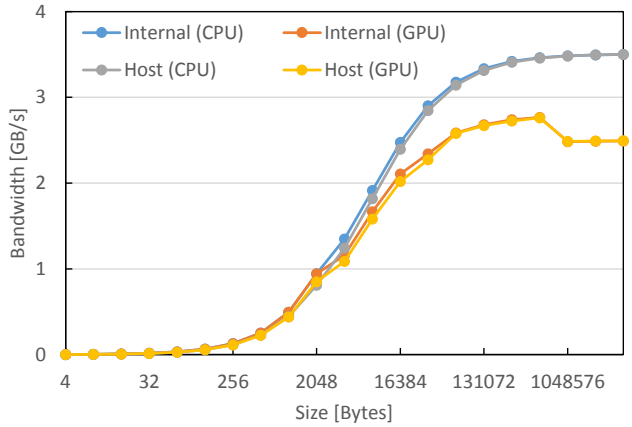

図 3　PEACH2 のバンド幅

3. Verbs

3.1 基本概念

Verbs は HCA（Host Channel Adapter）と OS の間のインタフェースを定義したもので，InfiniBand Trade Association[3] によって策定されている．Verbs には OS のカーネル空間から利用するものと，ユーザ空間から利用するものがある．ユーザ空間から利用する場合は，使用できない機能が一部存在し，HCA の初期化などではカーネル経由で HCA を操作する必要がある．しかしながら，基本的な通信操作はカーネルバイパスによってユーザ空間から直接 HCA を操作することができるため，通信性能に関してはカーネル空間から利用する場合と変わらない．

Verbs の基本的な通信モデルを図 4 に示す．Verbs では，すべての通信は HCA によって非同期に行われる．そして，プログラムから HCA へのエンドポイントとして，QP（Queue Pair）と CQ（Completion Queue）を用いる．QP は SQ（Send Queue）と RQ（Receive Queue）で構成されており，SQ に SWR（Send Work Request）を投入するこ

とで送信を指示し，RQ に RWR（Receive Work Request）を投入することで受信を指示する．QP に投入された WR（Work Request）は WQE（Work Queue Element）として管理され，投入順に HCA が処理する．そして，処理が完了すると CQ に CQE（Completion Queue Element）が投入される．プログラムは CQ をポーリングすることで，通信の完了を検出することができる．

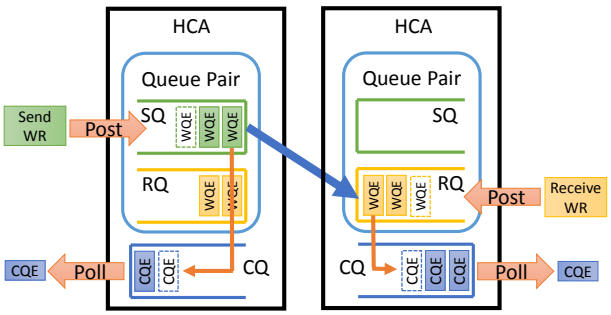

図 4　Verbs の通信モデル

3.2　通信オペレーション

Verbs では，以下にのような通信オペレーションをサポートしている．

Send/Receive

SWR で指定されたデータを，RWR で指定された受信領域に対して送信する．受信側はあらかじめ RWR を RQ に投入しておく必要があるが，送信側では送信したデータがどこに格納されるのかを意識する必要はない．派生オペレーションとして，Send with Immediate があり，通常の Send に加えて 4byte の即値データを送信することができる．この際，受信した即値データは RWR に直接書き込まれる．

RDMA Write

SWR で指定されたデータを，受信側の HCA が直接メモリに書き込む．受信側では RWR を必要とせず，CQ に CQE が投入されることもないため，通信が行われたことを検出することはできない．派生オペレーションとして，RDMA Write with Immediate があり，Send with Immediate と同様に 4byte の即値データを書き込むことができる．この場合は，受信側では RWR が消費され，CQ に CQE が投入されるため，通信の完了を検出することができる．

RDMA Read

SWR で指定したリモートノードのメモリからデータを読み込み，読み込んだデータを SWR で指定した受信領域に書き込む．リモートノードでは，RWR を必要としないため，通信が行われたことを検出することはできない．

Atomic

SWR で指定したリモートノードのメモリに対して，SWR で指定された Atomic オペレーションに実行し，その結果を SWR で指定した受信領域に書き込む．Atomic オペレーションとして，CAS（Compare and Swap）と FAA（Fetch and Add）に対応している．リモートノードでは，RWR を必要としない．

3.3　Verbs の GDR 対応

Verbs は GDR に対応しており，GDR に対応した HCA を利用することで，ノードを跨ぐ GPU 間での直接通信を行うことができる．この場合，Verbs の各 API で GPU メモリを CPU メモリと同様に扱うことができる．GDR に対応した HCA として，Mellanox 社の ConnectX-3 および Connect-IB がある [11]．

4.　TCA Verbs

本研究では，TCA/PEACH2 による Verbs 実装として TCA Verbs を提案・開発する．本節では，TCA Verbs の構造や動作，Verbs の各通信オペレーションの性能評価結果について述べる．

4.1　概要

2.3 節で述べた通り，PEACH2 の DMAC は RDMA Write しか行うことができず，Send/Receive や RDMA Read，Atomic といった通信オペレーションが不足している．また，RDMA Write に関しても，Verbs と TCA では API が異なり，単純に置き換えることはできない．そして，PEACH2 には 2.4 節で述べたような制限があるが，Verbs を用いる通常のプログラムはこのような制限を考慮しないため，制限に抵触した場合でも通信を行えるようにする必要がある．

PEACH2 の制限に抵触するデータであっても，一旦別の領域にコピーし，その領域を DMAC の対象とすることで PEACH2 によって転送することができる．従って，*tcaMalloc()* などで確保したメモリを送信・受信バッファとして用いることで，任意のデータを対象とした通信機構を構築することができる．そして，この通信機構を用いて，制御情報のやり取りを行うことで，Verbs の各通信オペレーションを実装することができる．また，通常 RDMA Write ではリモートノードの書き込み先アドレスなどを事前に取得しておく必要があり，そのための通信を行う必要がある．しかし，送信・受信バッファにリングバッファを用いて，各バッファを固定長パケット単位で利用することで，最初にリングバッファの先頭アドレスを取得しておけば，それ以降は直接パケットを転送することが可能となる．また，PEACH2 の制限を満たしている場合は，この機構の利用は制御情報のやり取りに留め，データに関しては直接転送を行うことで，高い通信性能を得ることができる．

4.2 パケット通信機構

図 5 に，TCA Verbs におけるパケット通信機構の構造を示す．この機構では，送信バッファからリモートノードの受信バッファに対して，PEACH2 の RDMA Write を行うことでパケットの転送を行う．各リングバッファは，*tcaMalloc()* によって確保した CPU メモリを使用し，パケットサイズを 4 の倍数にすることで，PEACH2 の制限を回避して任意のデータを転送することができる．

各リングバッファ内のパケットは PSN（Packet Sequence Number）によって管理し，Send PSN はパケットを送信する度に，Receive PSN はパケットの受信処理が完了する度にインクリメントする．そして，Send PSN と Receive PSN の組を PSN Pair として，DMA Chaining によってパケットの転送に続けて転送する．これによって，送信側ではリモートノードの受信バッファの空き容量や受信処理の進捗を把握することができ，受信側ではパケットの到着を検出することができる．

パケット通信において，高いバンド幅を得るためには，パケットサイズを大きくするか，一度に大量のパケットをまとめて転送できるようにする必要がある．しかし，パケットサイズを大きくすると，パケットの転送に掛かる時間が大きくなる．そして，一度に大量のパケットを転送する場合は，転送パターンの組み合わせが莫大になるため，ディスクリプタを事前に用意することができず，通信の際にディスクリプタを作成することになるため，レイテンシが大きくなってしまう．そこで，比較的短い固定長データを扱う「低レイテンシ通信用バッファ」と，可変長データを扱う「高バンド幅通信用バッファ」を，それぞれの送信・受信バッファとして用意し，データ長に応じて使い分けることで，低レイテンシと高バンド幅を両立することができる．

低レイテンシ通信用バッファでは，パケットサイズが小さく，1 回の RDMA Write では 1 パケットの転送に限定する．この場合，必要なディスクリプタはバッファブロック数分で済むため，レイテンシの低い内蔵メモリモードを利用でき，プログラム起動時にディスクリプタを作成しておくことで，DMAC にディスクリプタを登録するだけで通信を開始することができる．そして，パケットサイズが小さく，転送に掛かる時間も小さいため，低いレイテンシで通信を行うことができる．

高バンド幅通信用バッファでは，1 回の RDMA Write で大量のパケットを転送することができる．この際，リングバッファの先頭を跨ぐ場合は，メモリ領域が不連続となるため，DMA Chaining を用いて連続して転送する．そして，ディスクリプタは事前に用意せず，通信を行う際にホストメモリモードで作成する．ホストメモリモードはディスクリプタ作成に掛かる時間が小さいため，レイテンシへの影響をある程度に抑えつつ，高いバンド幅で通信を行うことができる．図 5 では簡略化のため，低レイテンシ通信用バッファを用いた場合の通信のみを示しているが，高バンド幅通信用バッファを用いる場合は，連続した複数のパケットが 1 回の RDMA Write で転送される．

4.3 GPU メモリの転送

通常，パケット通信機構によって GPU メモリの転送を行う場合，CPU メモリである送信・受信バッファと GPU メモリ間でのメモリコピーが必要となる．一般に，GPU メモリのコピーには CUDA の API である *cudaMemcpy()* を使用するが，これを用いたメモリコピーはレイテンシが大きいという問題がある．そのため，TCA Verbs ではパケット通信機構によって GPU メモリの転送を行う際に，gdrcopy[12] による GPU メモリのコピーと，送信バッファへのメモリコピーの回避を併用することで GPU メモリの高速な転送を実現する．

gdrcopy は NVIDIA によって提供されている GDR 機能を用いて，CPU メモリと GPU メモリ間のメモリコピーを行うためのライブラリである．gdrcopy はレイテンシが極めて小さく，CPU メモリから GPU メモリへのメモリコピーのバンド幅が非常に高いという特徴がある．これによって，受信バッファから GPU メモリへのコピーは高速に行うことができる．しかし，gdrcopy は GPU メモリから CPU メモリへのメモリコピーのバンド幅は非常に低い．そのため，gdrcopy による GPU メモリから送信バッファへのコピーは，データ長が小さい場合や PEACH2 の制限に抵触する場合に限定する．そして，ディスクリプタをその場で作成し，GPU メモリからリモートノードの受信バッファに直接データの転送を行うことで，GPU メモリから送信バッファへのコピーを回避する．この際，ディスクリプタはホストメモリモードで作成することで，レイテンシへの影響をある程度に抑えつつ，高いバンド幅で転送することができる．

4.4 通信オペレーションの実装および性能評価

4.4.1 Send/Receive

Send/Receive では，受信領域を RWR で指定するため，データを直接，受信領域に転送することはできない．そのため，基本的にパケット通信を用いてデータの転送を行う．ただし，データ長が大きい場合には，リモートノードに対して，受信領域を問い合わせ，データを直接転送する．

Send/Receive による Ping-Pong 通信のレイテンシを図 6 に，バンド幅を図 7 に示す．これらの図より，TCA Verbs の最小レイテンシは CPU メモリの場合で 3.5μs，GPU メモリの場合で 4.2μs であることがわかる．この性能を 2.6 節の PEACH2 の性能と比較すると，TCA Verbs によるオーバーヘッドは，CPU メモリの場合で，1.5μs，GPU メモリの場合で 2.2μs となる．GPU メモリのほうがオーバー

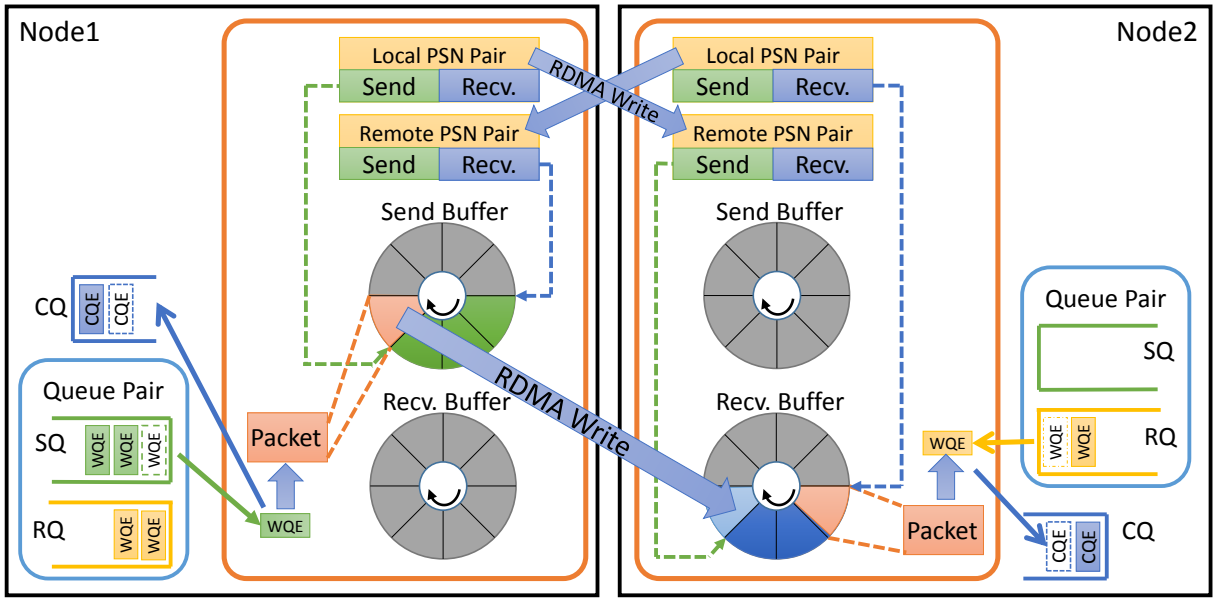

図 5　TCA Verbs におけるパケット通信機構

ヘッドが大きいのは，4.3 節で述べたような処理を行っているためである．また，IB と比べると，CPU メモリの場合でも非常にレイテンシが大きいことがわかる．これは，Verbs が元々 IB のために設計されたものであり，IB ではほとんどの処理が HCA のハードウェアで行われているが，TCA Verbs ではソフトウェアによって様々な処理を行う必要があるためである．一方，最大バンド幅に関しては，CPU メモリの場合で 3.5GB/s，GPU メモリの場合で 2.7GB/s となっており，ピーク性能だけを見ると PEACH2 の性能をほぼ出しきれていることがわかる．また，最大バンド幅が IB よりも低いのは，PEACH2 が PCIe Gen2 x8 で CPU に接続されているのに対し，IB HCA は PCIe Gen3 x8 で接続されており，PCIe バスでの最大バンド幅が IB のほうが高いためである．

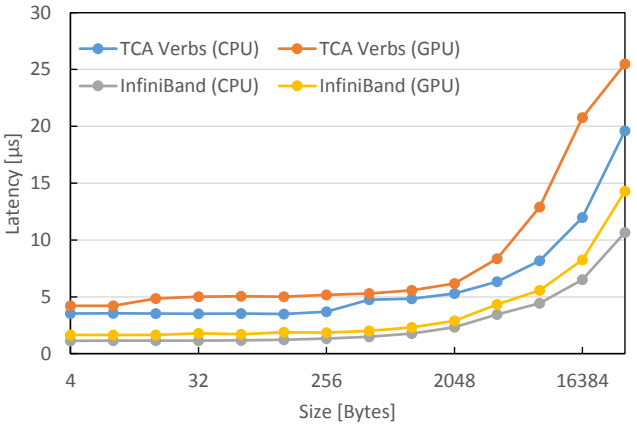

図 6　Send/Receive のレイテンシ

4.4.2　RDMA Write

RDMA Write では，受信領域が SWR で指定されているため，データを直接，受信領域に対して転送することができる．ただし，制御情報や RDMA Write with Immediate の即値データの転送のためにパケット通信も併用する．また，データ長が小さい場合は，ディスクリプタ作成に掛かる時間を節約するために，データに関してもパケット通信で転送する．

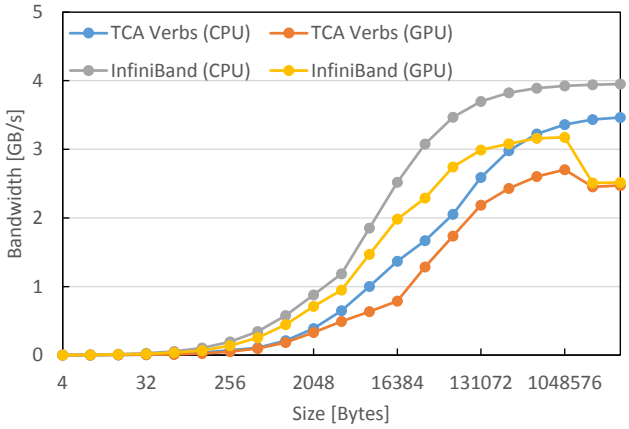

図 7　Send/Receive のバンド幅

RDMA Write with Immediate による Ping-Pong 通信のレイテンシを図 8 に，バンド幅を図 9 に示す．これらの図より，TCA Verbs の最小レイテンシは CPU メモリの場合で 3.5μs，GPU メモリの場合で 4.2μs であることがわかる．また，最大バンド幅は CPU メモリの場合で 3.5GB/s，GPU メモリの場合で 2.8GB/s であることがわかる．Send/Receive の性能と比べると，最小レイテンシや最大バンド幅はほぼ同じだが，RDMA Write のほうがバンド幅の立ち上がりが早いことがわかる．これは，Send/Receive は基本的にパケット通信機構を用いてデータの転送を行うため複数回のメモリコピーを必要とするが，RDMA Write ではリモートノードの受信領域に直接データを転送するためだと考えら

れる．

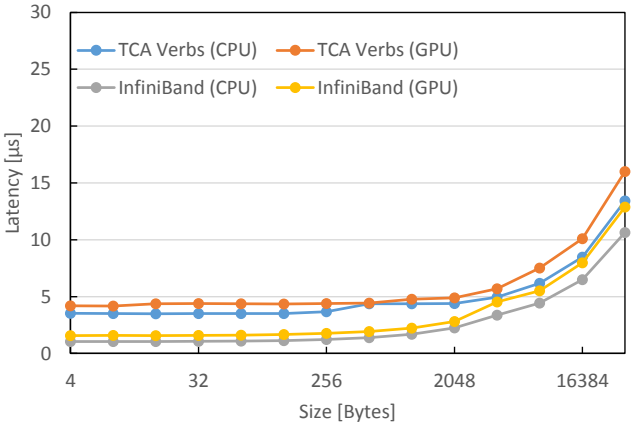

図 8　RDMA Write のレイテンシ

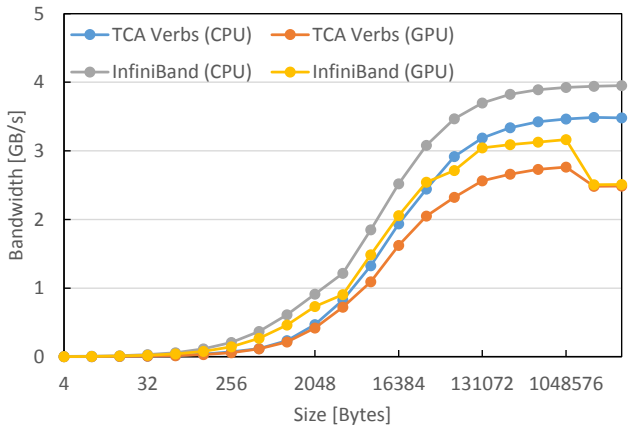

図 9　RDMA Write のバンド幅

4.4.3　RDMA Read

RDMA Read は，Read の対象となるメモリアドレスや受信領域，サイズなどの情報をパケット通信によって転送し，リクエストを受け取ったノードがデータを送り返す．送り返す際のデータ転送は，RDMA Write と同様である．

RDMA Read のレイテンシを図 10 に，バンド幅を図 11 に示す．これらの図より，TCA Verbs の最小レイテンシは CPU メモリの場合で 6.5μs，GPU メモリの場合で 7.2μs であることがわかる．これは，Send/Receive や RDMA Write の性能と比べて非常に大きいが，リクエストを送信し，その後，送り返すことを考慮すると，妥当なものだと考えられる．また，最大バンド幅は CPU メモリで 3.5GB/s，GPU メモリで 2.7GB/s となっており，RDMA Write とほぼ同じ性能が得られていることがわかる．

4.4.4　Atomic

Atomic は，対象となるメモリアドレスと Atomic オペレーションの種類，パラメータなどの情報をパケット通信によって転送し，リクエストを受け取ったノードが指定された処理を行い，結果をパケット通信で送り返す．

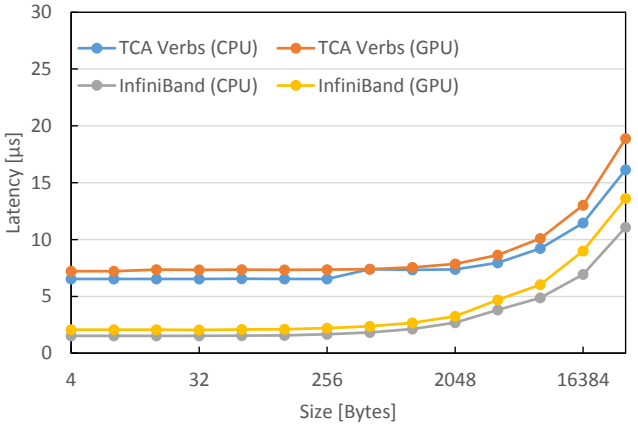

図 10　RDMA Read のレイテンシ

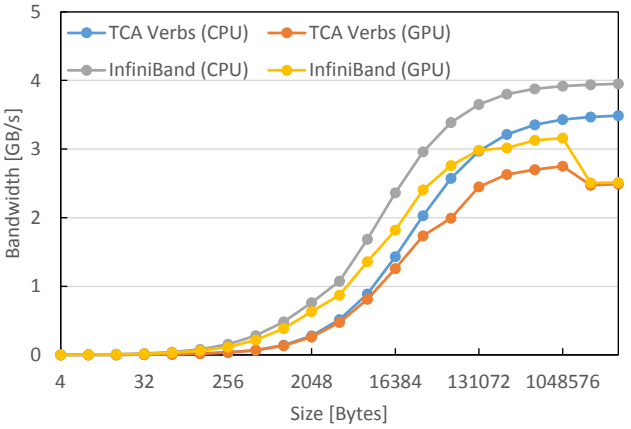

図 11　RDMA Read のバンド幅

Atomic に掛かった時間を表 2 に示す．この表より，Atomic の処理時間は種類に依らず 6.5μs であることがわかる．この結果は，RDMA Read と同様に，リクエストを送り，その後，結果を送り返すことを考慮すると妥当なものだと考えられる．

表 2　Atomic の測定結果

Operation	TCA Verbs	InfiniBand
CAS	6.5 μs	1.5 μs
FAA	6.5 μs	1.5 μs

5.　MPI における性能評価

本節では，TCA Verbs を用いて MPI を動作させた際の通信性能を，Ping-Pong 通信および姫野ベンチマーク [13] によって評価する．また，HA-PACS/TCA に搭載されている IB HCA は Dual-port なので，MPI では束ねて利用することもできるが，Verbs のレイヤでの通信性能と比較するため，本稿では 1 ポートのみ使用する．

5.1　Ping-Pong 通信の性能

MPI による Ping-Pong 通信のレイテンシを図 12 に，バンド幅を図 13 に示す．これらの図より，TCA Verbs の最

小レイテンシは CPU メモリの場合で 3.6μs，GPU メモリの場合で 4.6μs であることがわかる．IB の結果と比べると，レイテンシが大きく，Verbs のレイヤでの性能差がそのまま現れていることがわかる．また，この結果を Send/Receive の最小レイテンシと比較すると，MPI のオーバーヘッドは CPU メモリの場合で 0.1μs，GPU メモリの場合で 0.4μs となる．GPU メモリのほうがオーバーヘッドが大きいのは，MPI 内部でも CPU メモリと GPU メモリ間でのメモリコピーなどを行っているためだと考えられる．最大バンド幅に関してはは CPU メモリで 3.5GB/s，GPU メモリで 2.6GB/s となっており，ピーク性能だけ見ると，PEACH2 の性能をほぼ出しきれている．

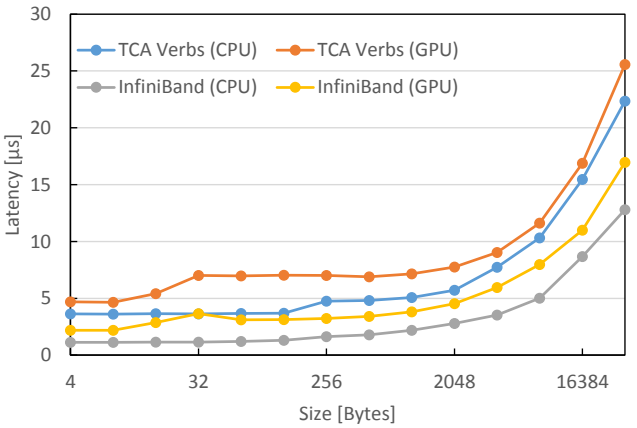

図 12 MPI のレイテンシ

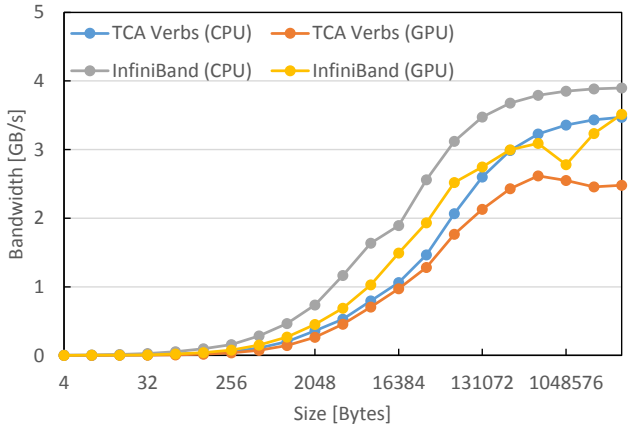

図 13 MPI のバンド幅

5.2 姫野ベンチマークの性能

本稿では，MPI 版の姫野ベンチマークを CUDA に移植したものを，さらに，NVIDIA Kepler アーキテクチャ向けに最適化し，i，j 次元方向で分割できるようにしたものを利用した [14], [15]．また，このプログラムでは通信性能の評価を行うため，通信のオーバラップは行われていない．問題サイズは，表 3 に示すものを使用し，ヤコビ法の反復回数は 1000 回に固定している．

HA-PACS/TCA の PEACH2 によるネットワークトポロジは図 14 のようになっており，直接接続されていないノード間で通信を行う場合には，他のノードの PEACH2 を経由する必要がある．この際，同時に複数の通信が同じ経路を使用すると，PCIe リンクを共有するため通信性能が劣化してしまう．姫野ベンチマークにおける通信は，袖領域の交換と収束判定のための Allreduce である．Allreduce は，データ長が 4 バイトと小さいため，PCIe リンクを共有することによる影響は小さいが，袖領域の交換はデータ長が大きいため影響は大きいと考えられる．そのため性能測定は，袖領域の交換で PCIe リンクの共有が生じないように各ノードに MPI ランクを割り当てた上で行っている．ただし，4×4 で分割する場合は，PCIe リンクの共有が生じないような割り当ては不可能のため，一部で PCIe リンクの共有が生じている．

表 3 姫野ベンチマークにおける問題サイズ（$i \times j \times k$）

Small	Middle	Large
64 × 64 × 128	128 × 128 × 256	256 × 256 × 512

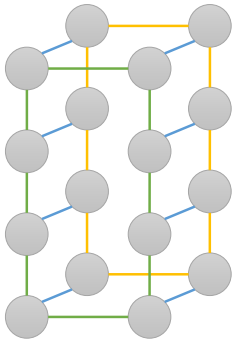

図 14 PEACH2 におけるネットワークトポロジ

測定結果を図 15 に示す．図中の凡例における “Calc.” は計算に掛かった時間，“Halo exch.” は袖領域の交換に掛かった時間，“Allreduce” は Allreduce に掛かった時間をそれぞれ示している．図より，Small においては IB との処理時間の差が大きく，通信レイテンシの影響を受けていることがわかる．しかし，Middle ではいくつかの分割においては処理時間の差が大きいものの，それ以外ではそれほど差がないことがわかる．そして，Large ではどの分割においても処理時間が IB とほぼ等しいことがわかる．これは，問題サイズが大きくなることで全体の処理時間に占める通信時間の割合が低くなったことと，通信サイズが大きくなりレイテンシよりもバンド幅が重要になったためだと考えられる．また，Large の 4×4 では若干差が大きいが，これは前述した通り PCIe リンクを共有していることで，通信性能の劣化が生じたためだと考えられる．

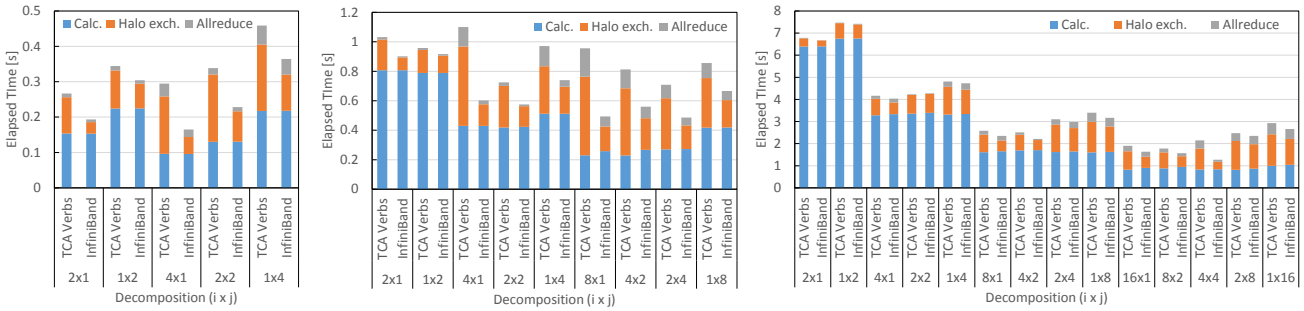

図 15　TCA 姫野ベンチマークの測定結果：左から順に Small，Middle，Large

6. おわりに

本稿では，IB の通信インタフェースである Verbs の TCA/PEACH2 による実装として TCA Verbs を提案・開発し，Ping-Pong 通信や姫野ベンチマークを用いて通信性能の評価を行った．その結果，MPI を動作させるという目標は達成でき，最大バンド幅に関しては PEACH2 とほぼ等しい性能が得られたが，レイテンシが増大してしまった．また，姫野ベンチマークにおいては，レイテンシが重要となる Small では IB よりも性能が劣ってしまったが，バンド幅が重要となる Large では IB とほぼ等しい性能が得られた．

TCA の目的は通信レイテンシの削減による強スケーリングの達成であるため，今後の課題としてはレイテンシの削減が最重要である．TCA Verbs によるレイテンシの増大は，Verbs を実装するためにソフトウェアで様々な処理を行っていることが原因だと考えられる．PEACH2 は FPGA 上に実装されているため，機能の拡張も比較的容易に行える．そこで，現在ソフトウェアで行っている処理を PEACH2 のハードウェアで行うことで，レイテンシの削減が可能だと考えられる．

本研究では TCA/PEACH2 を容易に利用するために Verbs を実装して MPI 環境を実現したが，必ずしもこの構成に限定する必要はない．Verbs は，Send/Receive や RDMA Read，Atomic などを実装する必要があり，PEACH2 との親和性はそれほど高くない．また，MPI は機能が多岐にわたるため，ライブラリの規模が大きくなり通信性能劣化の原因となりがちである．そのため，今後は GASNet[16] などの軽量な通信ライブラリの実装についても検討する予定である．

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．また，本研究における HA-PACS/TCA の利用は，筑波大学計算科学研究センター学際共同利用プログラム・平成 27 年度課題「密結合演算加速機構アーキテクチャに向けたアプリケーションの開発と性能評価」による．

参考文献

[1] Top500 Supercomputer Sites. http://www.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（ACS），pp.14-25，2013.
[3] InfiniBand Trade Association. http://www.infinibandta.org/.
[4] MPICH — High-Performance Portable MPI. http://www.mpich.org/.
[5] MVAPICH. http://mvapich.cse.ohio-state.edu/.
[6] Open MPI: Open Source High Performance Computing. http://www.open-mpi.org/.
[7] Intel MPI Library. https://software.intel.com/en-us/intel-mpi-library
[8] CUDA Toolkit. https://developer.nvidia.com/cuda-toolkit.
[9] NVIDIA GPUDirect. https://developer.nvidia.com/gpudirect.
[10] HA-PACS プロジェクト http://www.ccs.tsukuba.ac.jp/research/research_promotion/project/ha-pacs.
[11] Mellanox GPUDirect RDMA User Manual. http://www.mellanox.com/related-docs/prod_software/Mellanox_GPUDirect_User_Manual_v1.0.pdf.
[12] NVIDIA gdrcopy. https://github.com/NVIDIA/gdrcopy.
[13] 姫野ベンチマーク http://accc.riken.jp/2145.htm.
[14] E. Phillips and M.fatica. Implementing the Himeno benchmark with CUDA on GPU clusters Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium，pp.1-10, Apr. 2010.
[15] Toshihiro Hanawa, Yuetsu Kodama, Taisuke Boku, Mitsuhisa Sato. Tightly Coupled Accelerators Architecture for Low-latency Inter-Node Communication Between Accelerators SC14 poster, Nov. 2014.
[16] GASNet Communication System. http://gasnet.lbl.gov/.