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　14–25 (Oct. 2013)

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

塙 敏博[1,a)]　児玉 祐悦[1]　朴 泰祐[1]　佐藤 三久[1]

受付日 2013年4月9日, 採録日 2013年5月13日

概要：GPUなどの演算加速装置を用いたクラスタがHPCシステム向けに広く使われている．しかしこのようなクラスタでは，ノード間をまたがる演算加速装置間の通信において，CPUを介した複数回のメモリコピーが必要であった．このレイテンシ増加はアプリケーション性能を著しく低下させる．そこで，筑波大学計算科学研究センターでは，大規模GPUクラスタであるHA-PACSとしてコモディティ技術による大規模GPUクラスタ部分に加え，ノード間接続およびGPU間接続に，レイテンシとバンド幅の改善を目指した独自開発の密結合並列演算加速機構TCA（Tightly Coupled Accelerators）の開発を行っている．本論文では，TCAを実現する通信機構PEACH2とその基本転送性能の評価について述べる．さらにTCAを用いたアプリケーションの予備評価として，隣接2ノード間のping-pong通信における性能を測定し，従来の通信機構と比較した．その結果，ノードをまたぐCPUメモリ間転送では，最小レイテンシは0.9 μsを実現し，最大バンド幅は3.5 GB/sと，理論ピークの96%の性能が得られた．ノードをまたぐGPUメモリ間転送においては，最小で2.3 μsのレイテンシを実現し，短いメッセージ長ではCUDAによるノード内GPU間転送を超える性能を示した．

キーワード：GPGPU，PCI Express，相互結合網，CUDA，GPU Direct，RDMA

Development and Preliminary Evaluation of GPU Cluster Based on Tightly Coupled Accelerators Architecture

Toshihiro Hanawa[1,a)]　Yuetsu Kodama[1]　Taisuke Boku[1]　Mitsuhisa Sato[1]

Received: April 9, 2013, Accepted: May 13, 2013

Abstract: In recent years, heterogenious clusters using accelerators are widely used for high performance computing system. In such clusters, the inter-node communication among accelerators requires several memory copies via CPU memory, and the communication latency causes severe performance degradation. To address this problem, we propose Tightly Coupled Accelerators (TCA) architecture to reduce the communication latency between accelerators over different nodes. In addition, we promote the HA-PACS project in Center for Computational Sciences, University of Tsukuba not only in order to build up HA-PACS base cluster system, as the commodity GPU cluster, but also in order to develop the experimental system based on TCA architecture, as the proprietary interconnection network connecting among accelerators beyond the nodes. In the present paper, we describe TCA architecture, and the design and implementation of PEACH2 to realize TCA architecture. We also evaluate the basic performance of PEACH2 chip, and the performance of ping-pong communication compared with the conventional communication method for among GPUs. The results demonstrate that the PEACH2 chip has a latency between adjacent nodes of 0.9 μsec in minimum, and sufficient maximum performance with 95% of the theoretical peak performance in the case of inter-CPU communication. In the case of inter-GPU communication between adjacent nodes, we achieve the better performance with 2.3 μsec latency in minimum than the inter-GPU comminucation latency within a node using CUDA.

Keywords: GPGPU, PCI Express, interconnect, CUDA, GPU Direct, RDMA

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着目し，様々な HPC アプリケーションへ適用する GPGPU（General Purpose GPU）がさかんに行われている．GPU を搭載する高性能計算サーバをノードとする GPU クラスタも増加する一方であり，TOP500 リスト [1] には，年々多くの GPU クラスタが名を連ねてきており，2012 年 11 月のリストでは全体の 10%以上のシステムが GPU を搭載している．しかしながら，GPU クラスタにおいては，複数ノードをまたがる GPU 間の通信が必要であるが，従来は，CPU メモリを介してノード間を転送するため数回のデータコピーが必要となっていた．特にサイズの小さいデータ転送の場合にレイテンシの増加が性能低下を引き起こしていた．

そこで，筑波大学計算科学研究センターでは，HA-PACS としてコモディティ技術による大規模 GPU クラスタに加え，ノード間接続および GPU 間接続に，レイテンシとバンド幅の改善を目指した独自開発の密結合並列演算加速機構 TCA（Tightly Coupled Accelerators）の開発を行っている [2]．

本論文では，TCA アーキテクチャ，および TCA アーキテクチャに基づいた通信機構の実装として PEACH2（PCI Express Adaptive Communication Hub version 2）について述べ，基本転送性能の評価を行う．さらに，TCA を用いたアプリケーションの予備評価として，隣接 2 ノード間の ping-pong 通信における性能を測定し，従来の GPU 間通信機構と性能比較を行う．2 章では，HA-PACS と密結合並列演算加速機構 TCA について説明する．3 章では，HA-PACS/TCA および PEACH2 の構成，機能やボード実装について述べる．4 章では，PEACH2 ボードと 2 ノードを用いた性能予備評価について述べる．5 章で関連研究について説明した後，6 章で結論を述べる．

2. HA-PACS の概要

HA-PACS（Highly Accelerated Parallel Advanced system for Computational Sciences）は，筑波大学計算科学研究センターにおける大規模並列システム PACS/PAX シリーズの 8 代目にあたるシステムである [3]．HA-PACS では，対象とするアプリケーションとして，素粒子物理学，宇宙物理学，ライフサイエンスを中心に据え，限られた電力とスペースでこれらのアプリケーションを効率良く実行するため，アクセラレータ技術に基づく大規模クラスタを構築している．

HA-PACS はベースクラスタ部と TCA 部から構成される．

表 1　HA-PACS ベースクラスタ部の構成
Table 1　Specification of HA-PACS base cluster.

ノード構成	
CPU	Intel Xeon-E5 2670 2.6 GHz × 2 ソケット (8 コア + 20 M cache) / ソケット
メモリ	DDR3 1600 MHz × 4 チャネル，128 GB
ピーク性能	332.8 GFlops
GPU	NVIDIA Tesla M2090 1.3 GHz × 4 台
メモリ	GDDR5 6 GB / GPU
ピーク性能	2660 GFlops
InfiniBand	Mellanox Connect-X3 Dual-port QDR
全体構成	
ノード数	268
ストレージ	Lustre ファイルシステム 504 TB
相互接続網	InfiniBand QDR 288 ポートスイッチ × 2 台
ピーク性能	802 TFlops
ラック数	26
最大消費電力	408 kW

2.1 HA-PACS ベースクラスタ

HA-PACS ベースクラスタは，アクセラレータ技術を用いたアプリケーションの開発やプロダクトランを行うために導入された，コモディティ GPU クラスタである [2], [3]．

各計算ノードは 2 ソケットの Xeon E5 2600（SandyBridge-EP）CPU と 4 枚の NVIDIA M2090 GPU を搭載する．各 CPU ソケットはそれぞれ x40 レーンの PCI Express（PCIe）Gen3 を備えており，各 GPU はそれぞれ x16 レーンを用いて直接 CPU ソケットに接続されている（ただし，M2090 は Gen2 の速度しかサポートしない）．さらに残りの x8 レーンずつを使って，相互結合網用の NIC やその他のデバイスを接続することができる．

ノード間相互結合網には，Gen3 x8 レーンから 8 GB/s のバンド幅を想定して，バランスのとれた InfiniBand QDR デュアルポートを用いている．全体では，268 ノードをフルバイセクションバンド幅を持つ 2 系統の Fat-tree で接続している．

表 1 に HA-PACS ベースクラスタの仕様を示す．ピーク性能は 802 TFlops であり，2012 年 6 月の Top500 リストでは 41 位にランクインしている．性能電力比に優れたシステムで，1.04 GFlops/W の優れた電力効率を示している．HA-PACS ベースクラスタは，2012 年 2 月に運用を開始し，2012 年 10 月からは学際共同利用プログラムに基づく公募により広く利用されている．

2.2 HA-PACS/TCA：密結合並列演算加速機構

我々は，コモディティ技術の集合として GPU クラスタを実現するだけでなく，次世代の HPC におけるアクセラレータ技術の要素技術として，ノード間のアクセラレータ

[1] 筑波大学計算科学研究センター
University of Tsukuba, Tsukuba, Ibaraki 305–8577, Japan
[a)] hanawa@ccs.tsukuba.ac.jp

（GPU）どうしを直接結合することにより，レイテンシ，バンド幅の改善を目指している．このため HA-PACS では，ベースクラスタ部に加え，アクセラレータ間の直接結合を実現する通信機構を新規に研究開発している．これにより，ノード内 GPU 間だけでなくノード間にまたがる GPU 間の直接通信を実現する．この機構を「密結合並列演算加速機構：TCA（Tightly Coupled Accelerators）」と呼ぶ．

TCA を用いた実験用クラスタ HA-PACS/TCA は，2.1 節で述べたベースクラスタを拡張する形で導入され，HA-PACS システムとして一体で運用される予定である．

TCA は基本的なハードウェア技術としては PCIe を応用したものである．TCA のみで数百ノードのクラスタを構成することはケーブル長の限界から物理的に困難であり，性能上での利点も失われるため，8 から 16 台程度のノードを TCA で結合する．このノードグループをサブクラスタと呼ぶ．すべてのノードはベースクラスタと同様 InfiniBand でも結合され，TCA と InfiniBand とで階層ネットワークを構成する．特に大規模並列 GPU アプリケーションでは，高速な局所通信と大規模な一般通信とを組み合わせた最適化が可能になると考えられる．

現時点で TCA が対象にするモデルは，NVIDIA 社の Kepler アーキテクチャ Tesla 版である Tesla K20 ファミリである [4]．また詳細は次章以降で述べるが，Kepler アーキテクチャでは GPUDirect Support for RDMA [5] が利用可能になり，PCIe デバイスとの間で直接 GPU メモリの読み書きが可能になる．

3. TCA のための通信機構 PEACH2

3.1 PEACH2 ボードの概要

PEACH2 ボードは，我々が HA-PACS/TCA 向けに開発を進めている，TCA 用インタフェースボードである [2], [3]．この PEACH2 ボードどうしをつなぐことで TCA のシステムが構成される．

3.1.1 PCI Express による通信リンク PEARL

我々はこれまで，PCIe リンクを直接ノード間通信に用いる PEARL（PCI Express Adaptive and Reliable Link）を提案してきた [6]．

PCIe は，PC にデバイスを接続するためのシリアルインタフェース規格 [7] であり，今日では Ethernet，InfiniBand などのネットワークインタフェース，GPU など，ほぼすべての I/O デバイスが PCIe インタフェース経由で接続されている．各レーンの転送レートは，現在主流の Gen 2 規格における 2.5 GHz，5 GHz に加え，最新の Gen 3 規格では 8 GHz までサポートされる（それぞれ実効レートは，2 Gbps，4 Gbps，7.88 Gbps となる）．さらに，複数のレーンを束ねてバンド幅を拡張することができる（レーン数は “x4” のように表記する）．

PCIe における操作は，主に CPU とデバイスとの間でのメモリリード・ライトであるが，実際は，CPU 側にあたる Root Complex（RC），デバイス側にあたる複数の EndPoint（EP），それぞれの間で双方向のパケット通信を行っているにすぎない．そこで我々は，この PCIe リンク上の高速パケット通信をノード間接続に拡張した PEARL を提案し，PEARL を実現するための一種のルータとして，PCIe ポートを複数持ちルーティングを行う PEACH（PCI Express Adaptive Communication Hub）チップを開発した [8]．

PCIe で隣接ノードどうしを直接接続するためには，一方が RC，もう一方が EP で一対の関係になる必要がある．しかし，通常のノード CPU は RC であるため，ホストの PCIe インタフェースどうしを直接接続することはできない．そこで，各ノードには PEACH チップを搭載した PCIe 準拠のボードを装着し，その間を PCIe 外部接続ケーブル [9] を用いて接続する．このとき，前述のように，ケーブルの両端のポートは，RC と EP の対にする必要がある．

3.1.2 PEARL によるアクセラレータ直接結合

通常の GPU クラスタにおいて，複数ノード上にある GPU 間で通信を行うには，少なくとも 3 ステップのデータコピーをともなう．たとえば，ノード A 上の GPU A からノード B 上の GPU B に通信を行うには，以下のデータコピーが必要となる．

(1) GPU A のメモリから PCI Express 経由でノード A のメモリにコピー
(2) ノード A のメモリからネットワーク経由でノード B の CPU メモリにコピー
(3) ノード B のメモリから PCI Express 経由で GPU B のメモリにコピー

ここで，ネットワークを PEARL に置き換えることで，PCI Express のプロトコルのまま，ホストメモリにデータをコピーすることなくノード A 上の GPU A からノード B 上の GPU B へ通信することが可能になる．

3.1.3 PEACH2 チップ

現在，我々は HA-PACS/TCA に向けて FPGA を用いた PEACH2 を開発している．PCIe パケットの中継処理，高度な DMA 転送などをハードワイヤード処理で行う．FPGA には，PCIe Gen2 x8 レーンのハード IP を 4 ポート分内蔵した，Altera 社 Stratix IV GX [10] を用いている．FPGA を使用することで，動作速度やレイテンシの面では大きな制約を受けるが，その一方で，回路を柔軟に変更することができる．TCA のような実験的なシステムでは，機能の改善や，様々な機能追加が後から可能な FPGA が適しており，最大限性能が得られるように注意して実装を行っている．

PEACH2 は，PCIe Gen 2 x8 レーンを 4 ポート搭載し，1 ポートはホスト CPU と接続する．

PCIe ポートにおける RC と EP の切替えについては，当面は個別に FPGA のコンフィグレーションデータを用意

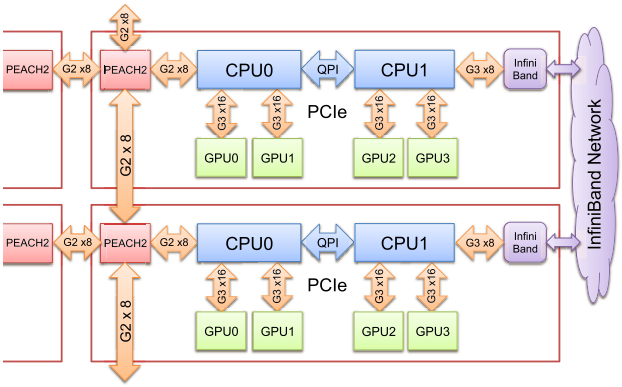

図 1　HA-PACS/TCA におけるノード構成
Fig. 1　Block diagram of computation node in HA-PACS/TCA.

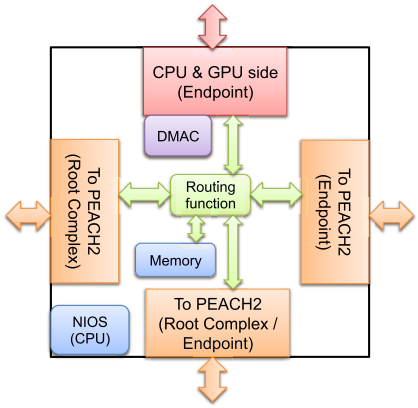

図 2　PEACH2 の構成
Fig. 2　Configuration of PEACH2.

することで対応する．将来的には partial reconfiguration 機能により，RC と EP を切り替えることを検討している．

3.2　PEACH2 と GPU 間の接続

HA-PACS/TCA では図 1 に示す構成を用いる．GPU は従来どおりホスト上の CPU に直接接続される．一方，PEACH2 ボードはホスト上の別の PCIe スロットに接続し，CPU 内蔵の PCIe スイッチを介して GPU にアクセスする．この構成では，CPU のみが RC になり，残る 4 つの GPU，IB HCA，PEACH2 は，すべてその配下の EP になる．この場合にもすべてのデバイスは単一の PCIe アドレス空間に配置されるため，GPU と PEACH の間は直接 PCIe プロトコルで通信可能である．

GPU メモリに他の PCIe デバイスがアクセスするために，Kepler アーキテクチャおよび CUDA 5.0 から，GPUDirect Support for RDMA と呼ばれる機能が提供されている [11]．これは，GPU 内のメモリを PCIe 空間上にマップして，同一 PCIe バス上にある他のデバイスが直接読み書きできるようにするものである．我々は，PEACH2 の開発当初からこのような機能が必要だと考えており，NVIDIA 社との NDA に基づき開発を行ってきたが，GPUDirect Support for RDMA の発表にともない，PEACH2 でもこれを利用することにした．ただし，GPU0 または GPU1 と，GPU2 または GPU3 の間の QPI をまたぐ直接アクセスには，性能の問題があり無効にされている．これと同様に，PEACH2 からのアクセスも GPU0，GPU1 のみに限定することを想定している．

図 1 の構成には，以下の利点がある．

(1) PEACH2 を使用しない場合，HA-PACS と同一の構成になるため，PEACH2 を使用した効果について正確に比較することが可能になる．
(2) GPU0，GPU1 ともに PEACH2 を利用して他ノードの GPU0，GPU1 に対してアクセスできる．これにより，16 ノードを TCA サブクラスタとして結合する場合には，32 GPU を 1 グループとして扱うことができる．
(3) 他の 3 ノードと接続が可能になる．サブクラスタのノード数を増やしてもホップ数を抑えられる．8 ノードであれば，最大 3 ホップで到達できる．

一方で，Xeon E5 CPU 2 個が提供する，PCIe 80 レーンのすべてが利用可能なプラットフォームを選択する必要がある．80 レーンを持たないシステムを PCIe スイッチを使って拡張する方法も考えられるが，ここでは省略する．

3.3　PEACH2 チップの構成

PEACH2 チップの構成を図 2 に示す．前節でも述べたように，4 つの PCIe Gen2 x8 ポートを持つ．便宜上それぞれ N(orth)，E(ast)，W(est)，S(outh) ポートと呼ぶことにする．N ポートはホストとの接続に用いるためつねに EP である．E ポートは EP，W ポートは RC に固定して，隣接ノードの PEACH2 との間でリングトポロジを構成するために使う．S ポートは RC と EP を選択可能にし，2 つのリング間で，対向の PEACH2 と S ポートどうしを接続する．

DMA コントローラには，後述するように，chaining DMA 機能を備えたものを搭載し，高速な DMA を可能にしている．パケットバッファとして，FPGA 内蔵のエンベデッドメモリ，および外付けの DDR3 SDRAM を用いる．

また，PEACH2 には経路表書き換えなどの操作のために管理用プロセッサを搭載する．PEACH2 ではパケット転送に関わる処理はすべてハードウェアによって処理を行うため，プロセッサ性能はあまり必要なく，FPGA 内蔵向けの小規模なプロセッサとして Altera 社の Nios II を用いる[*1]．

3.4　PCIe アドレスマッピングとルーティング

PEACH2 では，サブクラスタ内のすべてのノードを PCIe

[*1] 図には示していないが，ほかに，Nios II-ホスト間通信用，デバッグ用として，Gigabit Ethernet，RS-232C，液晶モジュールの各インタフェースを備える．

空間にマップし，それをもとに宛先を決定してルーティングを行う [12]．PCIe アドレスは 64 bit 空間を持つが，自ノードのホストやデバイスはそのうちごく一部しか使用しない．そこで，PEACH2 デバイスとしてホストから比較的大きなメモリ領域（現実装では 512 Gbyte）を割り当てることにし，その内部のアドレス空間をクラスタを構成するノードに割り当て，さらにその内部をホストメモリ領域や GPU にそれぞれ割り当てる[*2]．このノード・デバイスの割当てアドレス[*3]はすべてのクラスタ構成ノードが共有する．これらの PCIe アドレスはそれぞれアラインされたアドレスなので，アドレス上位ビットを比較するだけで済み，アドレス変換表を用いるのに比べて，きわめて低コストで出力先ポートを決定することができる．

実際に転送を行うためには，あるアドレスへのアクセスを，どのポートに出力するかの設定が必要である．ルーティング機構にはアドレスマスクと，範囲を指定するレジスタを用意し，マスク結果の範囲を判定することにより静的に経路を決定する．

PEACH2 が受信したパケットを N ポートからホスト側に送る場合には，PEACH2 どうしで共有しているアドレス空間から，ホスト内の固有のアドレス空間に対して変換が必要になる．各ノードでは PEACH2 に与えられているオフセットを記憶しておき，到着したパケットヘッダのアドレスを減算してからホストに送出する．

3.5 PEACH2 による通信

PEACH2 において，リモートノードに対するアクセスは，基本的に，PCIe における Memory Write 要求に限定しており，すなわち RDMA put プロトコルのみをサポートする．これは，Memory Read 要求に対応する場合，応答パケット（PCIe では Completion with Data）の制御が困難であり，仮に実装したとしても性能面で利点が少なく，他の性能に悪影響を与えると判断したためである．ただし，N ポートについては，ローカルノードの CPU メモリや GPU メモリからの読み出しのため Memory Read 要求に対応している．

リモートノードに対する読み出し要求については，直接ハードウェアで対応はしないが，間接的に対応することは可能である．PEACH2 には割込み制御レジスタが備えられており，これには他ノードからも書き込むことができる．そこで，リモートノードに対して，あらかじめ読み出し要求を登録しておき，割込みを発生させて，ドライバ内で要求に基づいてデータを要求元ノードに書き込む．これにより，読み出しと同等の操作を実現することができる．

[*2] ただし，512 Gbyte の領域を割り当てられる BIOS が必要であり，ごく限られたマザーボードだけが対応している．
[*3] マップされるアドレス自体は，BIOS の認識順などで異なる可能性がある．

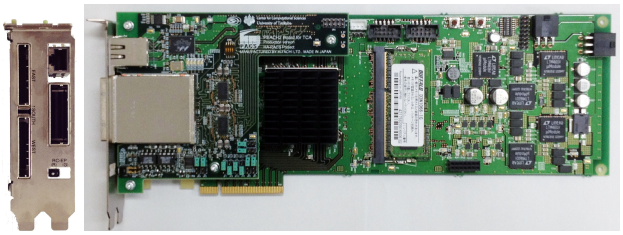

図 3　PEACH2 ボードの写真（左：パネル面，右：基板面，中央左のヒートシンク下に FPGA が隠れている）

Fig. 3　Photograph of PEACH2 board (left: side view, right: top view; FPGA is hidden under a heatsink on the center left of the board).

PEACH2 では，PIO と DMA の 2 つの通信方式を想定している．

3.5.1 PIO 通信

PIO 通信は，CPU の store 操作によってリモートノードに書き込みを行うことであり，小サイズのデータに向いている．PEACH2 チップに割り当てられたアドレス空間を，デバイスドライバを通してユーザ空間に mmap することで，ユーザプログラムから直接リモートノードの目的アドレスに向けて送信が可能になる．

3.5.2 DMA 通信

PEACH2 チップには chaining 機能を持つ，高性能な DMA コントローラを 4 チャネル搭載している．

Altera 社 PCIe IP に含まれる DMA 機能をもとに，chaining DMA 機能を備える DMA コントローラを開発した．ホスト上であらかじめ読み込み元，書き込み先 PCIe アドレス，サイズを指定したディスクリプタを，アドレスポインタで連結する．先頭アドレスを登録することにより連続して DMA 処理することが可能で，特に小パケットの連続通信の効率を高める．一方，ディスクリプタを転送するオーバヘッドが存在するため，制御レジスタを設定することで数個の DMA を軽量に発行できる機能（レジスタモード）も設けている．各 DMA 要求には，バースト長と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もできる．

3.6 PEACH2 ボード

図 3 に完成した PEACH2 ボードを示す．このボードは，PCIe ボード規格 [13] に準拠しており，メイン基板は 12 層，サブ基板は 8 層からなる．Gen2 x8 のエッジコネクタと，左側面には PCIe ケーブルポートが合計 3 個（E, W, S ポート）配置されている．E, W ポートは x8, S ポートには x16 のコネクタを使用するが信号は 8 レーンしか使用しない．中央部には Altera 社の FPGA Stratix IV 530GX, DDR3 SO-DIMM 1 枚が搭載されている．電源は，右上の PCIe ペリフェラル用コネクタから給電され，ホストとは独立して動作する可能性もあるため，PCIe スロットの電

源は使用しない．ボードの右部分には FPGA が使用する各電源電圧を生成するレギュレータ類がある．PEACH2 チップは，PCIe Gen2 IP の動作周波数に合わせて，主要な機能は 250 MHz で動作する．

3.7 TCA におけるプログラミング環境

TCA におけるプログラミング環境は，NVIDIA から提供されている CUDA 開発環境 [14] を基本とする．CUDA 4.0 以降では，同一 PCI バス内の GPU 間での直接転送，同一ノード内の複数 GPU およびホストとの間でアドレス空間を共有する UVA（Unified Virtual Addressing）が提供されている[*4]．これを拡張し，ノードをまたぐ GPU との間での直接転送を可能にすることを検討している．

サブクラスタ内では，あらかじめノード番号およびノード数を決めておき，それぞれのノードにある GPU は，ノード番号と GPU 番号で識別する．同一 PCI バス内の場合の GPU 間の直接転送は，CUDA では cudaMemcpyPeer() 関数，あるいは UVA を使った cudaMemcpy() などで指定できるが，それと同様に，TCA サブクラスタ内においてもメモリコピー機能を中心にした API 関数を用意する．

PEACH2 には前述のとおり，chaining 機構やブロックストライド転送が可能な DMA コントローラが備えられている．たとえば，ステンシル計算においては，袖領域の通信が各反復ごとに発生するが，毎回，同じメモリ領域を同じ宛先に転送することになる．そこで，あらかじめ計算を始める前に，すべての袖領域通信パターンの DMA ディスクリプタをチェインしたテーブルに登録しておく．3 次元配列における袖領域についても，バースト長とギャップ長を指定することで，ブロックストライド転送により効率良く転送を行うことができる．各反復においては，DMA ディスクリプタテーブルの先頭アドレスを DMA コントローラに登録するだけで自動的に転送が行われ，その間に CPU は内側領域の計算に専念することができる．

4. 性能予備評価

本章では，PEACH2 ボードおよび現時点での FPGA 論理を用いて，性能予備評価を行う．PEACH2 ボードは，試作版と，HA-PACS/TCA に搭載する予定の量産版とがある[*5]が，FPGA のピン配置は共通であり，FPGA 論理も同じものが使用できる．今回の評価では，試作ボードを用いた．

テスト環境は**表 2** に示すとおりである．これはマザーボード，GPU，OS を除いてほぼ HA-PACS ベースクラスタと同じ構成である．PEACH2 用のドライバに加えて，GPUDirect Support for RDMA に対応するためのドライバも開発した．

表 2　テスト環境
Table 2　Test environment for evaluation.

ハードウェア	
CPU	Xeon E5 2670 2.6 GHz × 2
メモリ	DDR3 1600 MHz x 4ch, 128 GB
マザーボード	(a) SuperMicro X9DRG-QF (b) Intel S2600IP
PCIe 設定	最大ペイロード長 256 byte, Max Read Request Size 4096 byte
GPU	NVIDIA K20（Kepler アーキテクチャ）
GPU メモリ	GDDR5 2600 MHz, 5 GB
PEACH2	PCIe 規格フルサイズ,
試作ボード	16 層+8 層（サブ）
FPGA	Altera Stratix IV GX 530
PEACH2 論理	20130222 バージョン
InfiniBand	Mellanox Connect-X3 FDR
	（FDR10 4x で使用，ノード間直結）
ソフトウェア	
OS	Linux, CentOS6.3
	kernel-2.6.32-279.22.1.el6.x86_64
GPU ドライバ	NVIDIA-Linux-x86_64-310.44
プログラミング環境	CUDA 5.0
MPI 環境	MVAPICH2 1.9b（enable-cuda 付き）

4.1 DMA 基本転送性能

PEACH2 ボードによる DMA 転送の性能を測定した．これにより，

(1) FPGA の PCIe 性能
(2) Chaining DMA コントローラ（DMAC）の性能

を確認することができる．

以降の測定は，ドライバ内において，chaining DMA のディスクリプタを設定した後，DMA をキックする直前に時間測定を開始し，FPGA からの転送終了をポーリングによりチェックし時間測定を終了する．測定には，CPU 内蔵のクロックカウンタ（TSC）を用いた．5 回測定したうちでそれぞれの項目で最も良い値を示した．

4.1.1 ローカルノード内 CPU 間

まず，ローカルノード内において，CPU メモリのみの DMA 性能を測定した．PEACH2 ドライバの内部に DMA 用バッファを確保し，PEACH2 からバッファの内容を DMA 転送することにより性能を測定した．

DMA write，DMA read をそれぞれ 128 回連続して行った結果得られたバンド幅を**図 4** 中の “CPU-CPU” に示す．本実装では，ディスクリプタの登録数に制限はないが，1 ディスクリプタのサイズが 32 byte であることから，CPU のページサイズ 4 KB に収まる 128 個を単位として測定した．

この結果より，本実装では，1 回の DMA サイズ 4 Kbyte

[*4] これも GPUDirect の 1 機能で，GPUDirect Peer-To-Peer Transfers and Memory Access と呼ばれる．
[*5] 図 3 に示したのは量産版の写真である．

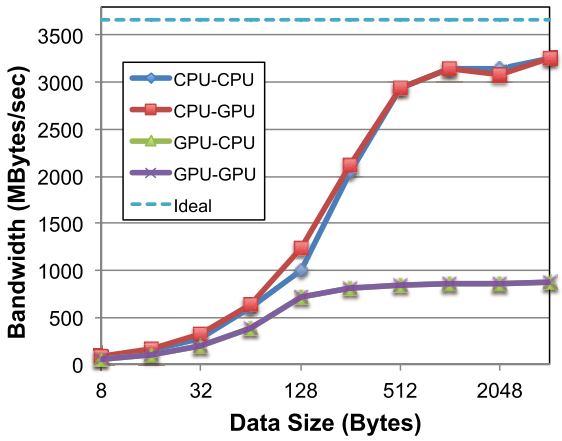

図 4　データサイズを変化させた場合のノード内 DMA 転送バンド幅（128 回連続）

Fig. 4　DMA bandwidth within a node for various data size (continuous transfer of 128 times).

で約 3.3 Gbyte/sec の結果を得られた．ただし，本測定では DMA 送信完了のみを確認しているため，指定先のアドレスに書き込まれる前に終了と判定される可能性があるが，128 個の要求を連続して送っているため実質的な差は少ないと考えられる．

ここで，PCIe パケットヘッダを含めた，より厳密な理論ピーク性能を計算してみる．PCIe Gen2 x8 レーンで 4 Gbyte/sec のバンド幅であるが，テスト環境では PEACH2-ホスト間のペイロードサイズが 256 byte であるため，パケットごとにトランザクション層ヘッダが 16 byte，データリンク層のシーケンス番号 2 byte，LCRC 4 byte，物理層のスタートフレーム 1 byte，エンドフレーム 1 byte がつき，

$$4\mathrm{GB/sec} \times \frac{256}{256+16+2+4+1+1} = 3.66\mathrm{GB/sec}$$

になる（図中に Ideal で示した）．したがって，ピーク性能の 90%の結果が得られており，PEACH2 の PCIe IP および DMAC が十分な性能を持つことが分かる．

次に，chaining DMA のオーバヘッドを見るため，転送回数を 1 回にして測定した場合のバンド幅を**図 5** 中の “Chaining” に示す．ディスクリプタ転送の影響がほとんど無視できた 128 回の場合の図 4 と比べて，オーバヘッドが大きく性能が低下していることが分かる．そこで，ディスクリプタのオーバヘッドを考慮し，ディスクリプタを使わずレジスタ設定による DMA 機構についても測定した．得られたバンド幅を図 5 の “Register” に示す．その結果，ディスクリプタ使用の場合に比べて 2 Kbyte 以下において 30～40%の性能向上が得られており，データサイズが小さく少量の DMA 転送において特に効果が高いことが分かった．

図 6 中の “CPU-CPU” に，サイズを 4 Kbyte に固定し，連続転送回数を変化させた場合の結果得られたバンド幅を示す．4 回の転送でも，最大性能の 70%の性能を示している．また，2 回以降の結果は図 5 の 8 Kbyte 以降の結果とほぼ一致しており，全体の転送量が同じであれば，ディスクリプタの個数にはあまり影響がないことが分かった．

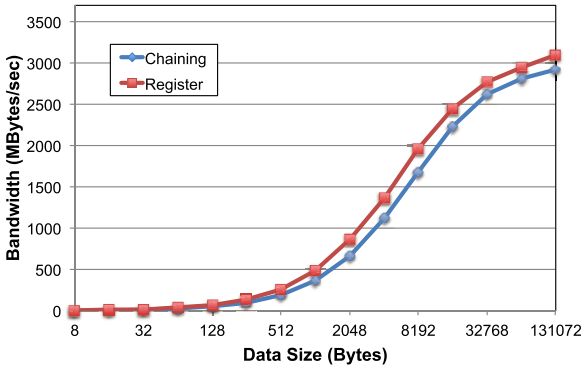

図 5　ディスクリプタモードとレジスタモードによる DMA 転送バンド幅の比較（1 回のみ転送）

Fig. 5　Comparison of DMA bandwidth between descriptor mode and register mode (single transfer).

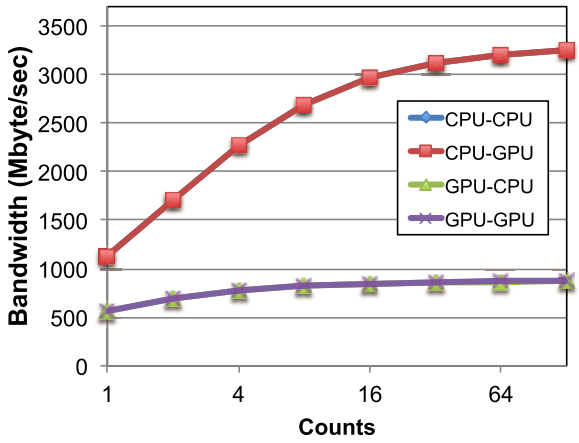

図 6　データサイズ 4096 byte 固定，ディスクリプタ数を変化させた場合のノード内 DMA 転送バンド幅

Fig. 6　DMA bandwidth within a node for various number of descriptors with fixed data size of 4096 bytes.

4.1.2　GPU（K20）との間の DMA 転送

次に，GPU 側に確保したメモリ領域に対し，以下の手順で FPGA から chaining DMA を用いて書き込みを行った．

(1) *cuMemAlloc()* または *cudaMalloc()* により GPU メモリを確保．
(2) 確保したメモリのアドレスを *cuPointerGetAttribute()* に渡し，トークンを得る．
(3) トークンを使って GPU メモリを PCIe 空間にピンダウンする．
(4) PEACH2 からアクセス可能になる．

ここで (3) を実現するためにはカーネル API が必要であり，デバイスドライバによって実現されている．ピンダウン操作は測定前にあらかじめ行っておく．したがって，指定先が CPU メモリの場合との操作の違いは，DMA ディスクリプタ内に指定する PCIe アドレスとして，(3) のピ

ンダウンにより得られたマップ済みのアドレスを指定することだけである．

CPU の場合と同様に，図 4 と図 6 に，それぞれ，データサイズを変えながら 128 回連続転送した場合，4 Kbyte に固定し連続転送回数を変えた場合のバンド幅を示す．“CPU-GPU” は，CPU メモリから読み出し，ノード内の GPU メモリに書き込んだ場合，“GPU-GPU” は，GPU メモリから読み出し，異なる GPU デバイスのメモリに書き込んだ場合を示す．その結果，CPU から GPU への DMA 性能は，CPU 間での DMA とほぼ同等であり，GPU に対する直接書き込みは，CPU を経由して転送し直す場合に比べて，高い性能が得られることが分かる．

一方，GPU メモリから DMA 読み出しが必要な場合，性能は最大 870 MB/s 程度で，CPU の場合に比べて性能が出ていない．これは，CPU に内蔵されている PCIe スイッチ機能の転送能力が不足していることが原因と考えられ，次世代の IvyBridge プロセッサにおいて改善されることが期待される．また，GPU 内部でも PCIe アドレスにマップするためアドレス変換が必要であり，オーバヘッドが存在すると考えられるが，詳細は今後調査する予定である．

4.2 PEACH2 ボードによる隣接ノード間 ping-pong 通信

次に，PEACH2 ボードを用いた ping-pong 通信をユーザプログラムとして記述し，PCIe ケーブルで PEACH2 間を直結した 2 ノード間で，レイテンシを測定しバンド幅を求めた．いずれも，片方のノードにおいて 100 回の ping-pong 通信にかかる往復時間を，計 100 回ずつ測定し，それぞれの片道時間のうちで最も良いものを採用した．

4.2.1 PIO 通信

まず，CPU において PIO 通信を用いた場合の ping-pong 通信プログラムは以下のとおりである．

(1) あらかじめ，PEACH2 デバイスドライバ内に受信バッファを用意しておく．
(2) 次に，相手ノードの受信バッファの PCIe アドレスに対して，PEACH2 チップに割り当てられた PCIe アドレス空間内に対応するアドレスオフセットを計算する．3.5.1 項で述べたように，これをデバイスドライバを通して mmap し，ユーザプログラムから直接このアドレスに対して書き込みをすることで，リモートノードの受信バッファに値を書き込むことができる．
(3) 同様に，ドライバ内の受信バッファを mmap して，ユーザプログラムで直接，到着したかどうかをポーリングする．

ノードのうち一方は，(2) により相手側に送信を行った後 (3) により受信待ちをし，もう一方では，逆の操作をする．

GPU 間の ping-pong 通信プログラムでは，GPU メモリの受信待ちを行う必要がある一方，送信は CPU から書き込みを行う必要がある．ここでは，cudaMalloc() により確保した GPU メモリを PCIe 領域にピンダウンした後，さらにそれを mmap することにより，CPU 上のユーザプログラムからローカル GPU メモリを直接ポーリングして受信待ちを行う．送信については，相手ノード中のピンダウンされた GPU メモリの PCIe アドレスに対して，自分の PEACH2 空間内のアドレスを計算し，そこに対して mmap することで，ユーザプログラムから相手の GPU メモリに送信可能にしている．

4.2.2 DMA 通信

続いて，DMA 通信を用いる場合の ping-pong 通信プログラムは以下のとおりである．

DMA 通信には，レジスタモードを使用する．あらかじめ DMA 要求のソースに送信バッファの PCIe アドレス，ディスティネーションに相手の受信バッファの PCIe アドレスを指定しておく．受信確認用には，別途バッファを用意し，DMA 完了時に相手ノードの受信確認用バッファに自動的にステータスが書き込まれるようにセットしておく．

送信側では，DMA コントローラの PCIe レジスタ空間を mmap しておき，制御レジスタに書き込むことで DMA 開始を指示するだけである．

受信側では，受信確認用バッファを mmap してユーザプログラム内でポーリングすることにより受信待ちを行う．

4.2.3 測定結果

図 7 に，各通信ペアにおける小データ時のレイテンシの測定結果を，**図 8** に，レイテンシから求めたバンド幅の結果を示す．

その結果，2 ノードの CPU メモリ間においては，PIO を用いた場合に最小 0.9 μs であることを確認した．これは，現在広く用いられている InfiniBand FDR におけるノード間転送レイテンシに匹敵する値 [15] である．InfiniBand FDR が Gen3 x8 であるのに対して PEACH2 は Gen2 x8 であることを考えると，きわめて低いレイテンシを実現しているといえる．

一方，DMA を用いた場合には，CPU メモリが読み出し側の場合，最小 2.4 μs で転送が可能であることを確認した．転送サイズが 8 KB の場合にもレイテンシは 5 μs 以下であった．転送サイズを大きくしていくと，4.1.1 項で測定したローカルノード・ドライバ内での測定結果よりも性能が向上し，最大で 3.5 GB/s のバンド幅性能を得た．これは，理論ピーク性能の約 96%である．この場合，読み出しと書き込みのノードが異なるため，メモリアクセス競合がなくなったからだと考えられる．また，相手側が CPU メモリでも GPU メモリでも性能には大きな差は認められなかった．

GPU メモリを読み出し側に指定した場合には，PIO，DMA ともに 2.3〜2.5 μs のレイテンシとなった．PIO 転送

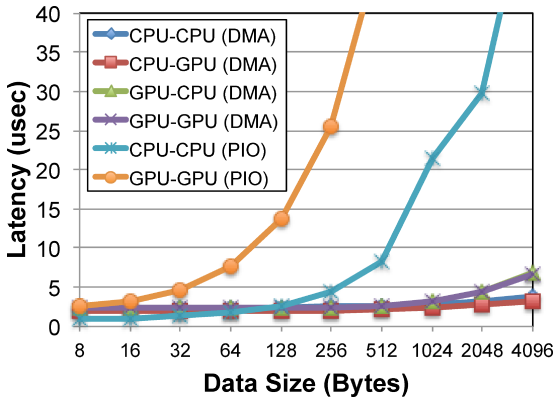

図 7　Ping-pong 転送におけるレイテンシ
Fig. 7　Latency of ping-pong transfer.

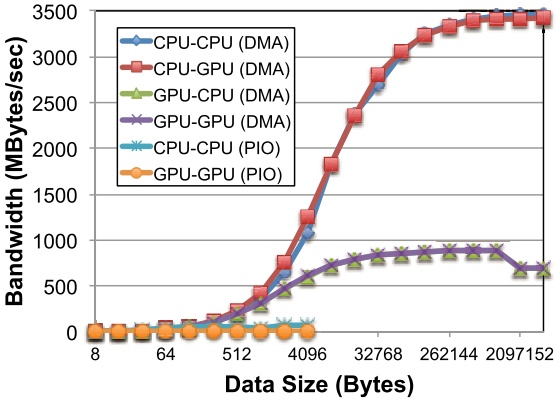

図 8　Ping-pong 転送におけるバンド幅
Fig. 8　Bandwidth of ping-pong transfer.

については，CUDA カーネルを利用するなど，改善の余地があると思われる．DMA については，転送サイズが 1 KB 程度までは，CPU 間転送に匹敵するバンド幅が得られており，小データにおいては十分実用的であると考えられる．転送サイズを大きくしていくと，4.1.2 項で述べたのと同様，性能が 880 MB/s 程度で飽和している．

以上の結果より，TCA において，転送の合計サイズが数 KB 程度の小規模なデータにおいても，理論ピーク値の半分に達するほど，高速な転送が可能であることが分かる．一方，数 byte 程度の極少量のデータの場合には PIO による転送，数 KB 未満の少量データではディスクリプタを使わない，レジスタ設定による DMA と使い分けることで，データ量に応じて最適な低レイテンシ通信が実現できる．GPU を用いた場合については，読み出し性能がボトルネックとなり，現状では期待されたほどの性能が得られていない．

4.3　TCA と既存通信機構との比較

最後に，PEACH2 を用いる TCA アーキテクチャと既存手法とで，GPU メモリ間での ping-pong の性能比較を行う．

ここでは，以下の 3 つを比較対象として示す．評価環境

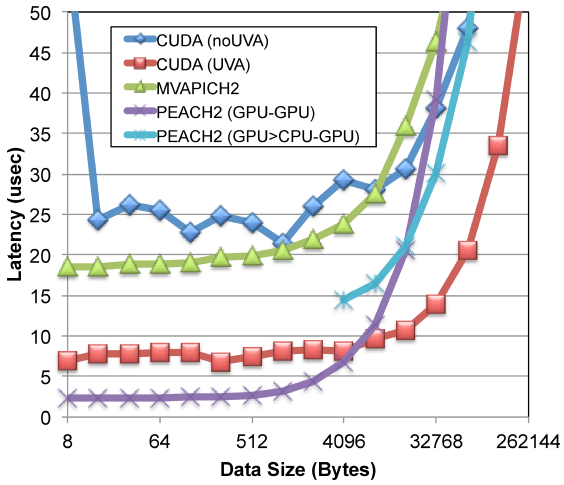

図 9　GPU 間 Ping-pong 転送におけるレイテンシの比較
Fig. 9　Comparison of latency on ping-pong transfer between GPUs.

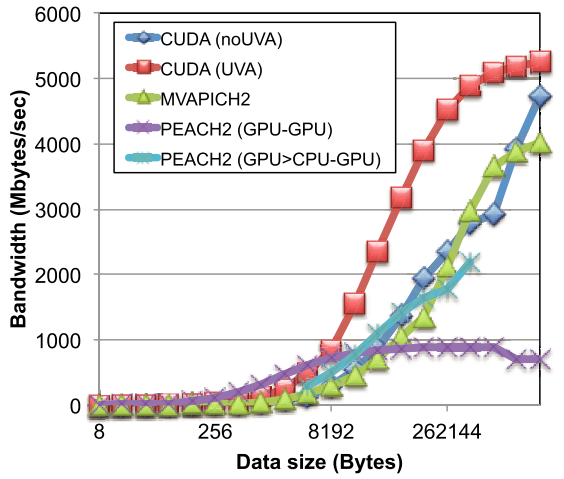

図 10　GPU 間 Ping-pong 転送におけるバンド幅の比較
Fig. 10　Comparison of bandwidth on ping-pong transfer between GPUs.

は表 2 に示すとおりである．いずれも，100 回の ping-pong にかかる往復時間を，計 5 回測定し，最も良いものを採用した．

(1) CUDA によって提供される，同一ノード内の GPU 間における cudaMemcpyPeer()，Unified Virtual Address(UVA) なし [CUDA (noUVA)]
(2) (1) の UVA あり [CUDA (UVA)]
(3) enable-cuda オプションをつけた MVAPICH2 を用いて，InfiniBand で接続されたノード間で GPU メモリを MPI_Send()，MPI_Recv() [MVAPICH2]

これに対し，PEACH2 では，GPU 間の DMA 転送（前述の GPU-GPU(DMA) に対応）[PEACH2(GPU-GPU)] に加えて，これでは 880 MB/s に律速されることから，GPU メモリからいったん CPU メモリに cudaMemcpy() して，それを相手 GPU に DMA 転送するプログラム [PEACH2(GPU>CPU-GPU)] も作成した．

図 9, **図 10** に，それぞれレイテンシの測定結果，レイテンシから求めたバンド幅の結果を示す．PEACH2 はいずれの既存手法よりもレイテンシが低く，ノード内 GPU 間コピーの CUDA（UVA）よりも高速である．これは，CUDA で使用しているプロトコルが PEACH2 とは異なるためだと考えられるが，ノード内においても PEACH2 の DMA コントローラを使用した方が高速ということになる．

MVAPICH2 については，現在 GPUDirect support for RDMA を用いた実装が進められており，最小で 6.54 μs の遅延時間を実現したとの報告もある [16] が，それでも依然として PEACH2 のレイテンシの方が低い．これは，PEACH2 には MPI のプロトコルスタックのオーバヘッドがないこと，DMA ディスクリプタをあらかじめ設定してあることがあげられるが，InfiniBand FDR のホストアダプタは Gen3 x8 であるのに対し，PEACH2 は Gen2 x8 を使用しており，PCIe 転送性能は本来 PEACH2 の方が約 1/2 である．

バンド幅を改良するために作成した，CPU メモリに cudaMemcpy を行うプログラムについては，CPU メモリのバッファ領域をカーネル空間にマップする必要があり，ドライバにそれを実現する機能を追加した．しかし現状では，実装上の問題があり，4 KB〜256 KB の範囲でしか動作していない．測定の結果，32 KB〜128 KB の範囲では MVAPICH2 よりも高いバンド幅を示したが，まだ改善の余地がある．

以上の結果から，PEACH2 による TCA 通信機構は，特にサイズの小さいデータの GPU 間転送において既存手法に比べて低レイテンシ，高いバンド幅性能を示し，既存のマルチ GPU アプリケーションの性能向上に大きく寄与できると考えられる．

5. 関連研究

PCIe スイッチに，Non transparent bridge（NTB）という特殊な機能を持たせることにより，ホスト間の通信を可能にする技術も存在している [17]．これをもとに製品化も行われている．NTB では，PCIe スイッチの下流ポート部分にブリッジ機能を追加し，その下流ポートからのアクセスと，そのポート以外からのアクセスとで異なった別の EP として振る舞う．この 2 つの Endpoint 間で相互にアドレス変換を行うことにより，異なる 2 つの RC との間，つまりプロセッサ間での通信を実現する．しかし，NTB は PCIe の仕様ではなく，PCIe 関連チップのベンダがそれぞれに独自の実装を行っているため，互換性はない．さらに，あらかじめ BIOS スキャン時に他ホストに対応する EP を認識させておく必要があり，ホストとの接続が切れた場合などには再スキャンが必要になる．PEACH2 においては，PCIe バスは各ポートで独立しており，他ホストとの接続状態が変化しても，ホスト-PEACH2 間にはまったく影響を与えない．

APEnet+ [18], [19] は，FPGA による独自の 3D Torus ネットワークを開発している．この中で我々と同様 Fermi アーキテクチャの GPU を用いて GPUDirect 機能も実現している．しかし，APEnet+はケーブルとして QSFP+を用いた独自のネットワークを用いており，PCIe をそのまま使うわけではない．また，GPU との通信プロトコルとして，GPUDirect Support for RDMA とは異なる P2P プロトコルを念頭に設計が行われており，我々のアプローチとは異なっている．

NVIDIA 社 GPU 向けのプログラム開発環境 CUDA 5 では，Kepler アーキテクチャの GPU と組み合わせることで，GPUDirect Support for RDMA と呼ばれる GPU メモリへの RDMA 機構が利用可能になる [5]．これを利用すると，InfiniBand でも GPU メモリの内容を直接ゼロコピーで送受信できるようになる [20]．PEACH2 においても，GPUDirect Support for RDMA を利用しているが，InfiniBand のようにプロトコルを変換する必要がなく，直接 PCIe パケットとして転送できるため，さらにレイテンシを小さく抑えることが期待できる．また，TCA においては MPI を用いる必要がなく，プロトコルスタックのオーバヘッドが削減できる．しかしながら，4.3 節でも述べたとおり，MVAPICH2 においては FDR を用いても GPU メモリ間の ping-pong 通信のレイテンシに 6.54 μs かかっており [16]，TCA に優位性がある．

6. おわりに

本論文では，HA-PACS プロジェクトにおいて開発を行っている，密結合並列演算加速機構 TCA について述べ，その通信機構の実装として PEACH2 について述べた．まず，基本転送の性能評価によって，十分な性能を持つことを示した．また，ping-pong 通信における性能を測定し，さらに既存手法である，CUDA による GPU 間メモリコピー，MVAPICH2 による GPU 間メモリコピーと性能を比較し，PEACH2 による TCA アーキテクチャの有効性を示した．PEACH2 を用いて 2 ノード間の通信性能を測定した結果，CPU 間転送では，PIO 転送により最小 0.9 μs の低レイテンシである一方，高性能な DMA コントローラにより理論ピークバンド幅の 96%の性能が得られた．一方，GPU 間転送では，最小 2.3 μs のレイテンシを実現し，既存手法よりも優れた性能を示した．

HA-PACS/TCA については，PEACH2 ボードの量産が完了し，現在は 8 ノードの実験用クラスタにおいて実証実験を始めている．2013 年秋には大規模実験クラスタが稼働する予定で，既存のベースクラスタと合わせて 1 PFlops 以上のシステムが完成する．HA-PACS ベースクラスタでは，すでに大規模 GPU アプリケーションのコーディングや性能評価が行われており，これらの知見を活かして TCA

用 API の整備，TCA を用いた本格的なアプリケーションの開発を進めてゆく予定である．演算加速器用インタコネクトを持つ，初の大規模実証クラスタとして，TCA の有効性を示すことができると考えている．

謝辞　本研究に際しご協力，ご助言をいただいた Dale Southard，Joel Scherpelz をはじめとする NVIDIA 社，および NVIDIA JAPAN 諸氏に深く感謝する．日頃よりご議論いただいている筑波大学計算科学研究センター次世代計算システム開発室および先端計算科学推進室の各メンバに感謝する．本研究の一部は文部科学省特別経費「エクサスケール計算技術開拓による先端学際計算科学教育研究拠点の充実」事業，および 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] Dongarra, J., Meuer, H., Stromaier, E. and Simon, H.: TOP500 List, available from ⟨http://www.top500.org/⟩.
[2] 塙　敏博，児玉祐悦，朴　泰祐，佐藤三久：Tightly Coupled Accelerators アーキテクチャのための通信機構，情報処理学会研究報告（計算機アーキテクチャ），Vol.2012-ARC-201, No.26, pp.1–8 (2012).
[3] 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：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-HPC-130, No.21, pp.1–7 (2011).
[4] NVIDIA Corp.: NVIDIA Tesla Kepler GPU Computing Accelerators, available from ⟨http://www.nvidia.co.jp/content/tesla/pdf/Tesla-KSeries-Overview-LR.pdf⟩.
[5] NVIDIA Corp.: NVIDIA GPUDirect, available from ⟨http://developer.nvidia.com/gpudirect⟩.
[6] Hanawa, T., Boku, T., Miura, S., Okamoto, T., Sato, M. and Arimoto, K.: Low-Power and High-Performance Communication Mechanism for Dependable Embedded Systems, *Proc. 2008 International Workshop on Innovative Architecture for Future Generation Processors and Systems*, pp.67–73 (2008).
[7] PCI-SIG: *PCI Express Base Specification, Rev. 3.0* (2010).
[8] Otani, S., Kondo, H., Nonomura, I., Uemura, M., Hayakawa, Y., Oshita, T., Kaneko, S., Asahina, K., Arimoto, K., Miura, S., Hanawa, T., Boku, T. and Sato, M.: An 80Gbps Dependable Communication SoC with PCI Express I/F and 8 CPUs, *2011 IEEE International Solid-State Circuits Conference*, pp.266–267 (2011).
[9] PCI-SIG: *PCI Express External Cabling Specification, Rev. 1.0* (2007).
[10] Altera Corp.: Stratix IV Device Handbook, available from ⟨http://www.altera.co.jp/literature/lit-stratix-iv.jsp⟩.
[11] NVIDIA Corp.: Developing A Linux Kernel Module Using RDMA For GPUDirect, available from ⟨http://developer.download.nvidia.com/compute/cuda/5_0/rc/docs/GPUDirect_RDMA.pdf⟩.
[12] 塙　敏博, 児玉祐悦, 朴　泰祐, 佐藤三久：Tightly Coupled Accelerators アーキテクチャ向け通信機構の予備評価，情報処理学会研究報告（HOKKE-20），Vol.2012-ARC-202 / 2012-HPC-137, No.13, pp.1–8 (2012).
[13] PCI-SIG: *PCI Express Card Electromechanical* (*CEM*) *Specification, Rev. 2.0* (2007).
[14] NVIDIA Corp.: NVIDIA CUDA: Compute Unified Device Architecture, available from ⟨http://developer.nvidia.com/category/zone/cuda-zone⟩.
[15] Mellanox Technologies, Ltd.: *ConnectX-3 VPI Product Brief*.
[16] Panda, D.: MVAPICH2: A High Performance MPI Library for NVIDIA GPU Clusters with InfiniBand, GPU Technology Conference (GTC2013) Presentation (2013), available from ⟨http://on-demand.gputechconf.com/gtc/2013/presentations/S3316-MVAPICH2-High-Performance-MPI-Library.pdf⟩.
[17] Gudmundson, J.: Enabling Multi-Host System Designs with PCI Express Technology (2004), available from ⟨http://www.plxtech.com/products/expresslane/techinfo⟩.
[18] Ammendola, R. et al.: APEnet+: High bandwidth 3D torus direct network for petaflops scale commodity clusters, *Journal of Physics*, Conference Series, Vol.331, Part 5, No.5 (2011).
[19] Rosetti, D. et al.: Leveraging NVIDIA GPUDirect on APEnet+ 3D Torus Cluster Interconnect (2012). available from ⟨http://developer.download.nvidia.com/GTC/PDF/GTC2012/PresentationPDF/S0282-GTC2012-GPU-Torus-Cluster.pdf⟩.
[20] Mellanox Technologies: Mellanox OFED GPUDirect, available from ⟨http://www.mellanox.com/content/pages.php?pg=products_dyn&product_family=116&menu_section=34⟩.

塙 敏博　（正会員）

平成 10 年慶應義塾大学大学院理工学研究科計算機科学専攻博士課程修了．博士（工学）．同年東京工科大学工学部情報工学科講師，平成 15 年東京工科大学コンピュータサイエンス学部講師，平成 19 年筑波大学計算科学研究センター研究員を経て，平成 20 年より筑波大学システム情報工学研究科准教授．計算機アーキテクチャ，高性能相互結合網，ハイパフォーマンスコンピューティング，GPU コンピューティング等に関する研究に従事．平成 7 年度情報処理学会論文賞．IEEE-CS 会員．

児玉 祐悦　（正会員）

昭和 63 年東京大学大学院情報工学専門課程修士課程修了．同年通商産業省電子技術総合研究所入所．平成 13 年独立行政法人産業技術総合研究所に改組．平成 23 年より筑波大学大学院システム情報工学研究科教授．データ駆動やマルチスレッド等の並列計算機システムの研究に従事．特にプロセッサアーキテクチャ，並列性制御，FPGA 応用，ネットワーク制御，アクセラレータ等に興味あり．博士（工学）．情報処理学会奨励賞，情報処理学会論文賞（平成 2 年度），市村学術賞（平成 7 年）等受賞．電子情報通信学会，IEEE-CS 各会員．

朴 泰祐　（正会員）

昭和 35 年生．昭和 59 年慶應義塾大学工学部電気工学科卒業．平成 2 年同大学大学院理工学研究科電気工学専攻後期博士課程修了．工学博士．昭和 63 年慶應義塾大学理工学部物理学科助手．平成 4 年筑波大学電子・情報工学系講師，平成 7 年同助教授，平成 16 年同大学大学院システム情報工学研究科助教授，平成 17 年同教授，現在に至る．超並列計算機アーキテクチャ，ハイパフォーマンスコンピューティング，クラスタコンピューティング，GPU コンピューティングに関する研究に従事．筑波大学計算科学研究センターにおいて，超並列計算機 CP-PACS，PACS-CS，HA-PACS 等の研究開発を行う．平成 14 年および平成 15 年情報処理学会論文賞，平成 23 年 ACM ゴードンベル賞，平成 24 年情報処理学会山下記念研究賞，各受賞．IEEE CS，ACM 各会員．

佐藤 三久　（正会員）

昭和 34 年生．昭和 57 年東京大学理学部情報科学科卒業．昭和 61 年同大学大学院理学系研究科博士課程中退．同年新技術事業団後藤磁束量子情報プロジェクトに参加．平成 3 年通産省電子技術総合研究所入所．平成 8 年新情報処理開発機構並列分散システムパフォーマンス研究室室長．平成 13 年より筑波大学システム情報工学研究科教授．平成 19 年度より平成 24 年度まで同大学計算科学研究センターセンタ長．理学博士．並列処理アーキテクチャ，言語およびコンパイラ，計算機性能評価技術，グリッドコンピューティング等の研究に従事．IEEE，日本応用数理学会会員．