Tightly Coupled Accelerators アーキテクチャに基づく GPU クラスタの構築

塙　敏博[†]　児玉祐悦[†]
朴　泰祐[†]　佐藤三久[†]

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．本稿では，TCA を実現する通信機構 PEACH2 とその予備評価について述べ，TCA を用いたアプリケーションの展望について論じる．

GPU Cluster Based on Tightly Coupled Accelerators Architecture

Toshihiro Hanawa,[†] Yuetsu Kodama,[†] Taisuke Boku[†]
and Mitsuhisa Sato [†]

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. Furthermore, the perspective of the application using TCA is discussed.

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) が盛んに行われている．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) について述べる．加えて基本転送性能評価によりその有効性を示す．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

† 筑波大学 計算科学研究センター
Center for Computational Sciences, University of Tsukuba

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 と密結合並列演算加速機構 TCA

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 (GPU) 同士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．

HA-PACS は，既に稼働しているベースクラスタ部に加え，2013 年秋に完成予定の TCA を用いた実験用クラスタである HA-PACS/TCA 部から構成される．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．全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 である[3]．また詳細は次節以降で述べるが，Kepler アーキテクチャでは GPUDirect Support for RDMA[4] 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. HA-PACS/TCA と 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),5)]．この PEACH2 ボード同士をつなぐことで 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

我々はこれまでに，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，ほぼ全ての 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 2.5GHz, 5GHz に加え，最新の Gen 3 規格では 8GHz までサポートされる (それぞれ実効レートは，2Gbps, 4Gbps, 7.88Gbps 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 側に当たる Root Complex (RC)，デバイス側に当たる複数の 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過ぎない．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)]．

隣接ノード同士を PCIe で接続するためには，一方は RC，もう一方は EP がペアの関係になる必要がある．しかし，ノード CPU は必ず RC であるため，ホストの PCIe インタフェース同士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，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伴う．例えば，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 4 ポートのハード IP を内蔵した Altera 社 Stratix IV GX[10] を用いている．FPGA を使用することにより，動作速度やレイテンシの面では大きな制約を受ける．その一方で，回路を柔軟に変更することができ，機能の改善や，様々な機能追加が後から可能であり，TCA のような実験的なシステムに適している．

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 は従来通りホストに直接接続される．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 が RC，4 つの GPU, IB HCA, PEACH2 は全てその配下の EP になる．この場合にも全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Tesla 版 K20 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．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 の発表に伴い，PEACH2 でもこれを利用することにした．但し，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は，

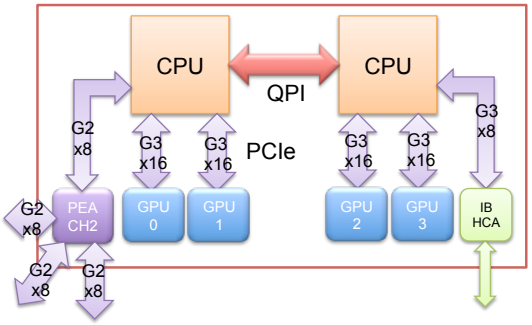

図 **1** HA-PACS/TCA におけるノード構成

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 共に他ノードの GPU0, GPU1 にアクセスできる．
(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 レーンの全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常に EP, E ポートは EP, W ポートは RC として，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，対向の PEACH2 と S ポート同士を接続する．

DMA コントローラには，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

☆ 実際には一部のブロック転送などは可能であり，問題ない性能が得られる場合もある．

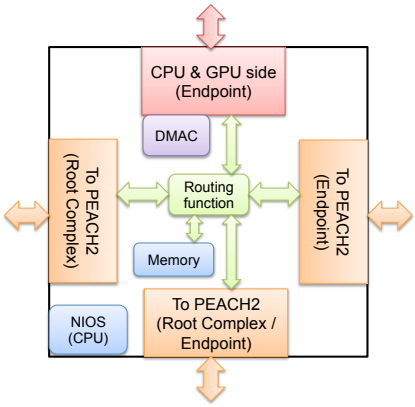

図 2　PEACH2 の構成

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全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☆．

3.4　DMA コントローラ

DMA コントローラとして，Altera 社 PCIe IP に含まれる DMA 機能を元に，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も設けている．各 DMA 要求にはブロックストライド転送を指定することもできる．

3.5　PCIe アドレスマッピングとルーティング

PEACH2 では，サブクラスタ内の全てのノードを PCIe 空間にマップし，それを元に宛先を決定してルーティングを行う[12]．PCIe アドレスは 64bit 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 512Gbyte) 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☆☆．このノード・デバイスの割り当てアドレス☆☆☆は全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，極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 同士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．

図 3　PEACH2 試作ボードの写真 (左：パネル面，右：基板面)

3.6　PEACH2 ボード

図 3 に現在実装中の PEACH2 ボードを示す．このボードは，PCIe ボード規格[13] のフルサイズ (106.7cm×312mm) であり，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 GX，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のみから給電され，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 250MHz 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

☆ 図には示していないが，他に，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．

☆☆ 但し，512Gbyte 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．

☆☆☆ 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．

表 1　テスト環境

ハードウェア	
CPU	Xeon E5 2670 2.6GHz (SandyBridge) × 2
メモリ	DDR3 1600MHz x 4ch, 128GB
マザーボード	(a) SuperMicro X9DRG-QF (b) Intel S2600IP
GPU	NVIDIA K20 (Kepler アーキテクチャ)
GPU メモリ	GDDR5 2600MHz, 5GB
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, 290
PEACH2 論理	20130121 バージョン
ソフトウェア	
OS	Linux, CentOS 6.3 kernel-2.6.32-279.19.1.el6.x86_64
GPU ドライバ	NVIDIA-Linux-x86_64-310.32
プログラミング環境	CUDA 5.0

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☆．これを拡張し，ノードを跨ぐ 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，TCA のサブクラスタ内についても同様なブロック転送，ブロックストライド転送など，chaining DMA ハードウェアに適した API 関数を用意する．

4. 性能予備評価

本節では，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．テスト環境は表 **1** に示す通りである．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．

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．

☆ これも GPUDirect の 1 機能で，GPUDirect Peer-To-Peer Transfers and Memory Access と呼ばれる．

(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．以前の実装では，DMAC として Altera 社の chaining DMA を使用しており，片方には必ず FPGA 内部メモリを指定する必要があった．今回実装した DMAC では，読み出し・書き込み共に PCIe アドレスを指定できる．

以降の測定は，全てドライバ内で，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．各項目は 5 回測定したうちで最良の値を示した．

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．

旧実装 (Altera) の場合の DMA write(PEACH2 から CPU), DMA read(CPU から PEACH2) をそれぞれ 255 回繰り返した場合，本実装において CPU から CPU への DMA を 128 回繰り返した結果を図 **4** に示す．旧実装では，ディスクリプタサイズが 16byte, 4KB の表に最大 255 個登録することができた．本実装では，ディスクリプタの登録数に制限はないが，同じ 4KB に合わせた．ディスクリプタは 32byte であるため 128 個のディスクリプタを置くことができる．また，旧実装は転送終了割込みにより測定しているが，本実装ではポーリングにより転送終了を判定して測定した．なお，割込みを用いる場合に比べて 1μ 秒程度高速であった．

この結果より，本実装では，1 回の DMA サイズ 4Kbyte で約 3.3 Gbyte/sec の結果を得られた．但し，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．

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 4Gbyte/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 256byte となっているため，パケットごとにトランザクション層ヘッダが 16byte，データリンク層のシーケンス番号 2byte, LCRC 4byte, 物理層のスタートフレーム 1byte, エンドフレーム 1byte がつくため，

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

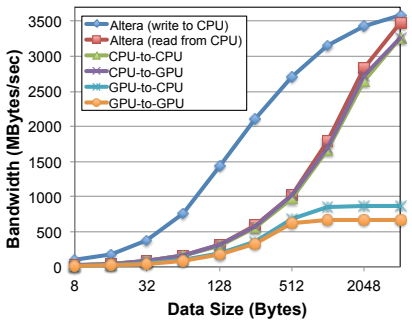

図 4　ノード内 PEACH2 DMA 性能（旧: 255 回，新: 128 回連続）

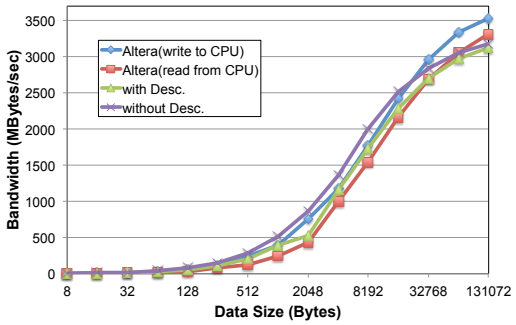

図 5　ノード内 PEACH2 DMA 性能 (1 回のみ)

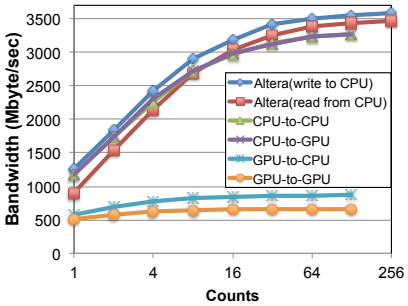

図 6　ノード内 PEACH2 DMA 性能 (4096byte 固定)

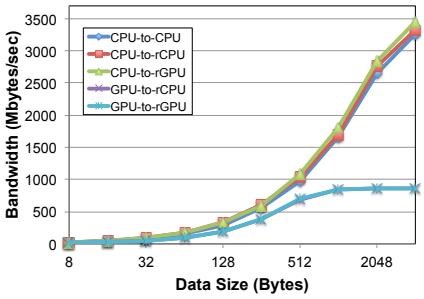

図 7　隣接ノード間 PEACH2 DMA 性能 (128 回連続)

になる．従って、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．また，旧実装と比較すると，DMA read 時の性能に近いことが分かる．これは，DMA により読み出したデータをそのまま書き込むため，DMA read の性能で律速されるためである．

次に，chaining DMA のオーバヘッドを見るため，転送回数を 1 回として測定した結果を図 **5** に示す．ディスクリプタテーブル転送の影響がほとんど無視できる 128 回の場合と比べて，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．ディスクリプタのオーバヘッドを考慮した，ディスクリプタを使わずレジスタ設定による DMA 機構についても同時に測定した．その結果，ディスクリプタ使用の場合に比べて最大で 2Kbyte のとき 63%の性能向上を得られ，単発の DMA 転送において特に効果が高いことが分かった．

図 **6** には，サイズを 4Kbyte に固定し，連続転送回数を変化させた場合の結果を示す．4 回程度の転送でも，最大性能の 75 から 80%の性能を示している．また，2 回以降の結果は図 5 の 8Kbyte 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．

4.1.2　GPU(K20) との 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()* 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．いずれも CUDA driver API を用いている．また，(3) を実現するためにデバイスドライバが必要である．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 から図 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，転送 1 回のみ，4Kbyte に固定し連続転送回数を変えた場合の転送性能をそれぞれ示す．その結果，CPU から GPU への DMA 性能は，CPU に対する DMA 書き込みとほぼ同等であり，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 に対する DMA 読み出し性能は最大 870MB/s 程度で，CPU の場合に比べて性能が出ていない．これは，現バージョンの SandyBridge プロセッサに内蔵されている 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．さらに，読み書き共に GPU を指定した場合にはより性能が低下しているが，これは同一デバイスに対して read/write を同時に指定したためであると考えられる．

4.2 PEACH2 ボードによる隣接ノード間通信

まず，PEACH2 ボード間を転送した場合の遅延時間の評価を行った．ここでは DMA は用いず，PIO(つまり CPU からの store 命令) によって測定した．

測定は，(1) 1 台のマシンのマザーボードに，PEACH2 ボードを 2 枚 (A, B とする) 装着し，A, B 間を PCIe 接続ケーブルで接続 (2) PEACH2 ドライバ中でクロックカウンタを読み出し (3) ボード A の PCIe 空間中，ボード B に相当するバッファアドレスに対して 4byte をストア (4) ボード A から B に自動的にパケット転送，ボード B からバッファに書き込まれる (5) ボード A はバッファの値が変化するまでポーリング，クロックカウンタを読み出し，(2) との差を測定，の手順である．1 台の PC で測定しているのは，書き込み開始から完了までの時間測定を厳密に行うためである．その結果，**782ns** で転送が行えることを確認した．現時点の論理には最適化の余地があるため，さらに改善できる可能性がある．

続いて，PEACH2 の DMAC を用いて，隣接ノードへの DMA 性能を測定した．図 **7** に，128 回連続転送した場合の結果を示す．図中の，CPU-to-rGPU は，ローカルノード (転送元) が CPU メモリ，隣接ノード (転送先) が GPU メモリであることを示す．比較のために，ローカルノード内の CPU メモリ間転送 (CPU-to-CPU) も示した．PEACH2 間の転送遅延は，DMAC のパイプライン処理により隠蔽され，ほとんどスループットの低下は見られないことが分かった．これにより，PEACH2 チップにおける PCIe IP，および DMAC が十分な性能を発揮していることが分かる．

以上の結果より，PEACH2 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．DMAC にはブロックストライド機能が備わっており、特に 3 次元のステンシル計算において，袖領域の通信に有効な機能である．これらを chaining 機能で連結することにより、1 度の DMA 発行でハードウェアのみで複数の袖領域を連続して転送することができる．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．

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[15]．これを元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 間には全く影響を与えない．

APEnet+[16),17)] 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．

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[4]．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[18]．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．

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．さらに，基本転送の性能評価によって，その有効性を示した．PEACH2 は，PIO 転送により最小 0.8μ 秒の低レイテンシである一方，chaining DMA

により理論ピークバンド幅の 90%を超える性能が得られた．

HA-PACS/TCA については，PEACH2 チップの DMA 性能改善，および PEACH2 ボードの量産を 2012 年度中に完了し，2013 年秋には大規模実験クラスタが稼働する予定で、ベースクラスタ部と合わせて 1FPlops 以上のシステムが完成する．HA-PACS ベースクラスタでは，既に大規模 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行く予定である．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 部の有効性を示すことができると考えている．

謝辞

本研究に際し御協力，御助言をいただいた Dale Southard, Joel Scherpelz を始めとする NVIDIA 社，および NVIDIA JAPAN 諸氏に深く感謝する．日頃より御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，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, 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 (アーキテクチャ), Vol. 2012-ARC-201, No. 26, pp. 1–8 (2012).
3) NVIDIA Corp.: *NVIDIA Tesla Kepler GPU Computing Accelerators*. http://www.nvidia.co.jp/content/tesla/pdf/Tesla-KSeries-Overview-LR.pdf.
4) NVIDIA Corp.: *NVIDIA GPUDirect*. http://developer.nvidia.com/gpudirect.
5) 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).
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, *Proceedings of 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*. http://www.altera.co.jp/literature/lit-stratix-iv.jsp.
11) NVIDIA Corp.: *Developing A Linux Kernel Module Using RDMA For GPUDirect*. 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*. http://developer.nvidia.com/category/zone/cuda-zone.
15) Gudmundson, J.: Enabling Multi-Host System Designs with PCI Express Technology, http://www.plxtech.com/products/expresslane/techinfo (2004).
16) 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).
17) Rosetti, D. et al.: Leveraging NVIDIA GPUDirect on APEnet+ 3D Torus Cluster Interconnect (2012). http://developer.download.nvidia.com/GTC/PDF/GTC2012/PresentationPDF/S0282-GTC2012-GPU-Torus-Cluster.pdf.
18) Mellanox Technologies: *Mellanox OFED GPUDirect*, http://www.mellanox.com/content/pages.php?pg=products_dyn&product_family=116&menu_section=34.