Vol.2012-ARC-202 No.13
Vol.2012-HPC-137 No.13
2012/12/13

Tightly Coupled Accelerators アーキテクチャ向け通信機構の予備評価

塙 敏博[1,a)]　児玉 祐悦[1,b)]　朴 泰祐[1,c)]　佐藤 三久[1,d)]

概要：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 を導入し，2012 年 2 月から運用を開始している．素粒子・宇宙物理・生命科学等の，極めて大量な演算を要求する大規模並列アプリケーションにおいて，数百〜千台規模の GPU を定常的に利用した大規模並列実行により，サイエンスの新しい分野を切り拓くことを目指している．HA-PACS は，既に稼働中のコモディティ技術による大規模 GPU クラスタ部分に加え，次世代のアクセラレータ間結合の要素技術として TCA (Tightly Coupled Accelerators) アーキテクチャに基づく実験クラスタからなる．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る．

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いる．

しかしながら，GPU の演算性能の高さに対して，GPU と CPU とをつなぐインタフェースである PCI Express (PCIe) の性能がボトルネックとなっている．さらに 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専用相互結合機構 TCA (Tightly Coupled Accelerators) の開発を行っている [2].

[1] 筑波大学 計算科学研究センター
[a)] hanawa@ccs.tsukuba.ac.jp
[b)] kodama@cs.tsukuba.ac.jp
[c)] taisuke@cs.tsukuba.ac.jp
[d)] msato@cs.tsukuba.ac.jp

1.1 HA-PACS とアクセラレータ間結合機構 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．

最終的に，HA-PACS は，既に稼働しているベースクラスタ部に加え，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，また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 で結合し，特に低レイテンシ・高バンド幅を要求するアルゴリズムにこれを適用する．このノードグループをミニクラスタと呼ぶ．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 の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．K20 は，ベースクラスタ部に搭載されている M2090 の約 2

倍の倍精度演算性能を持つ [3]．また詳細は次節以降で述べるが，K20 では，新たに 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．ノード構成についてはベースクラスタとほぼ同様の構成を考えているが，CPU などの各構成要素は調達時点での最新製品をバランス良く取り入れる予定である．

2. HA-PACS/TCA と PEACH2

2.1 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 [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．

2.1.1 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 に加えて，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, 約 8Gbps [*1] 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．

[*1] 厳密には，Gen3 だけは 128b130b 変換のため，7.88Gbps になる．

2.1.2 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．

2.1.3 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．

2.2 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 ボードは別の PCIe スロットに接続し，GPU メモリにアクセスする必要がある．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 CPU に内蔵された PCIe スイッチを介して直接 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

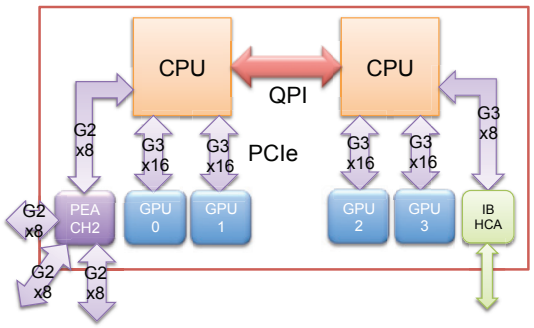

図 1 HA-PACS/TCA におけるノード構成

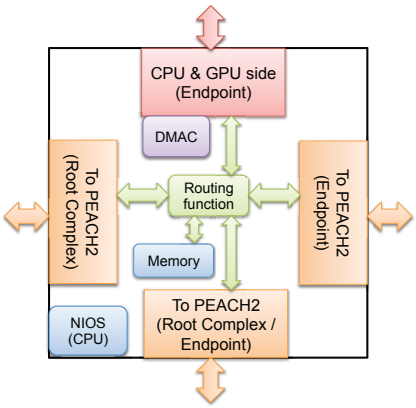

図 2 PEACH2 の構成

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，性能の問題があり無効にされている[*2]．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．ミニ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．

2.3 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 を用いる．図には示していないが，この他に，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．

2.4 DMA コントローラ

DMA コントローラには，chaining DMA 機能を備えたコントローラを用い，PCIe の性能を最大限引き出せるように考慮している．現時点では，Altera 社の PCIe IP に含まれている chaining DMA 実装 [12] を利用している．この IP では，ホスト上で，あらかじめサイズ，内部アドレス，PCIe アドレスを指定したディスクリプタを最大 255 個まで作成しておき，ディスクリプタを登録することにより指定した個数を連続して DMA 処理することができる．

2.5 PCIe アドレスマッピングとルーティング

PEACH2 では，ミニ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．

図 3 に PEACH2 におけるアドレス割り当ての考え方を示す．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[*3]．このノード・デバイスの割り当てア

[*2] 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．

[*3] 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．

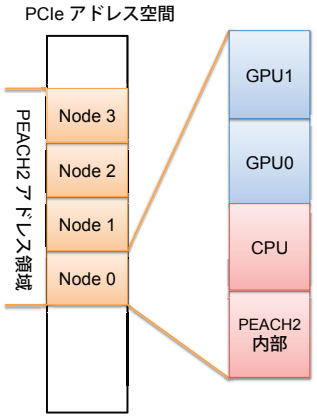

図 3 PEACH2 におけるアドレス割り当て

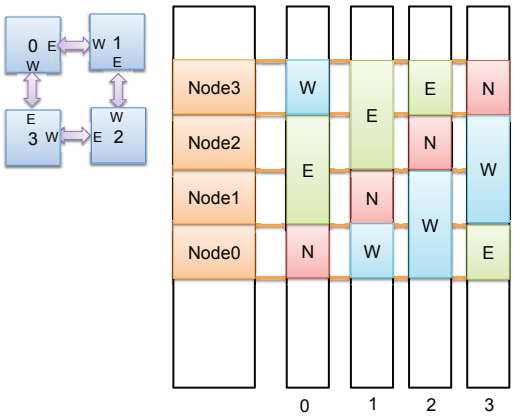

図 4 4 ノードリングの場合の出力ポート設定

ドレス[*4]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．**図 4** に 4 ノードリング構成の場合の例を示す．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．

2.6 PEACH2 プロトタイプボード

図 5 に PEACH2 プロトタイプボードを示す．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．また，テスト用の液晶モジュール，Gigabit Ethernet ポート，JTAG ポートなどが搭載されている．

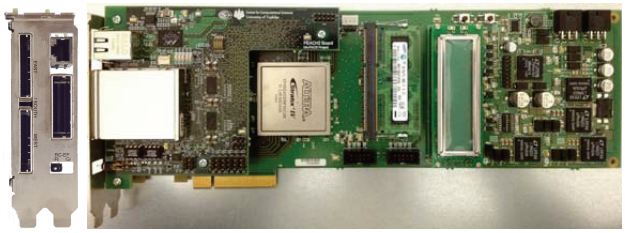

図 5 PEACH2 試作ボードの写真 (左：パネル面，右：基板面)

現在，HA-PACS/TCA に搭載するための量産版を実装中である．機能についてはほぼ変更はないが，基板サイズの縮小や，信頼性向上のための修正などを施している．

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．

2.7 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[*5]．これを拡張し，ノードをまたがった 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．

ミニクラスタ内では，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 のミニ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．

3. 予備性能評価

本節では，PEACH2 試作ボードおよび現時点での FPGA 論理を用いた性能の予備評価について述べる．テスト環境は**表 1** に示すように，HA-PACS ベースクラスタと同じ構

[*4] 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．

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

表 1 テスト環境

ハードウェア	
CPU	Xeon E5 2.6GHz × 2
メモリ	128GB
GPU	NVIDIA K20
マザーボード	Intel S2600IP, W2600CR
	SuperMicro X9DRG-QF
ソフトウェア	
OS	Linux, CentOS6.3
	kernel-2.6.32-279.{9,14}.1.el6.x86_64
GPU ドライバ	NVIDIA-Linux-x86_64-304.{51,64}
プログラミング環境	CUDA 5.0

成になっている．

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．

3.1 DMA 基本転送性能

PEACH2 試作ボードを用いて，FPGA 内部メモリとの間での DMA 転送性能を測定した．これにより，
(1) FPGA の PCIe 性能
(2) Chaining DMA コントローラの性能
を確認することができる．

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 からの転送終了の割込みで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．

3.1.1 PEACH2-CPU 間

PEACH2 ドライバの内部に DMA streaming 用バッファを確保し，PEACH2 から chaining DMA によりバッファの内容を読み書きすることにより性能を測定した．ここで，DMA write は PEACH2 からホストへの転送，DMA read はホストから PEACH2 への転送である．

DMA write, DMA read をそれぞれ 255 回繰り返した結果を**図 6** に示す．この結果より，いずれも 1 回の DMA サイズ 4Kbyte で 3.3 Gbyte/sec を超える結果を得られた．但し DMA write においては，PEACH2 における DMA 転送が完了した時点で割込みがかかるため，それが CPU メモリに反映される前である可能性はある．

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，

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

になる．従って、ピーク性能の 93%の結果が得られており，PEACH2 の PCIe IP および Chaining DMA コントローラが十分な性能を持つことが分かる．DMA write は送信し終えたら応答を待たずに次の送信ができるのに対して，DMA read は CPU からの応答パケットを待つ必要があるため，write より性能が低くなるが，転送単位のサイズが 2Kbyte を超えると write に近い性能を示している．

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 7** 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 255 回の場合と比べて，大きく性能が低下していることが分かる．ディスクリプタを転送するためのオーバヘッドが無視できないため、ディスクリプタを使わない DMA 機構も持つことが望ましいと考えられる．**図 8** に，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 7 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．

3.1.2 PEACH2-GPU(K20) 間

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．いずれも CUDA driver API を用いている．
(1) cuMemAlloc() を用いて GPU メモリを確保
(2) 確保した GPU メモリ (の一部) を PCIe 空間にマップ
(3) FPGA から chaining DMA で GPU メモリに書き込み
(4) cuMemHostAlloc() でホストメモリを確保し，cuMemcpyDtoH() 関数で GPU メモリの内容をホストに転送し，FPGA からの書き込み内容と一致することを確認

ここでは (3) に要する時間を計測した．PEACH2-CPU 間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 GPU のベースアドレスレジスタ (BAR) 中にマップされた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 6 から図 7 に，それぞれ，データサイズを変えながら 255 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，GPU の DMA 書き込み性能は，CPU に対する DMA 書き込みとほぼ同等であることが分かった．これにより，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．

DMA 読み出し性能は最大 830MB/s 程度で，CPU の場合に比べて性能が出ていない．これは，GPU の内部で，PCIe アドレスにマップするためのアドレス変換を行っており，これが原因ではないかと考えられるが，詳細は不明である．

今回グラフには示していないが，QPI を超えた GPU へ

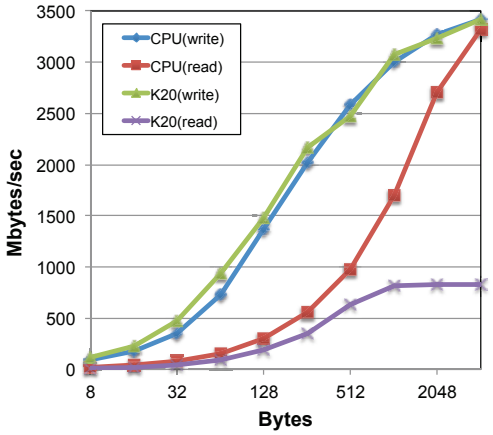

図 6 PEACH2-CPU, GPU 間の DMA 性能 (255 回連続)

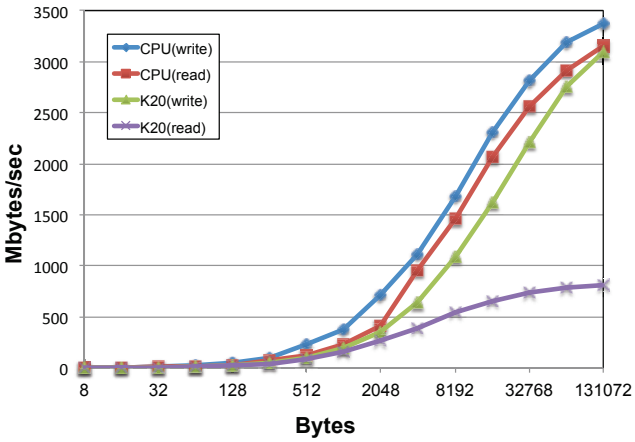

図 7 PEACH2-CPU, GPU 間の DMA 性能 (1 回のみ)

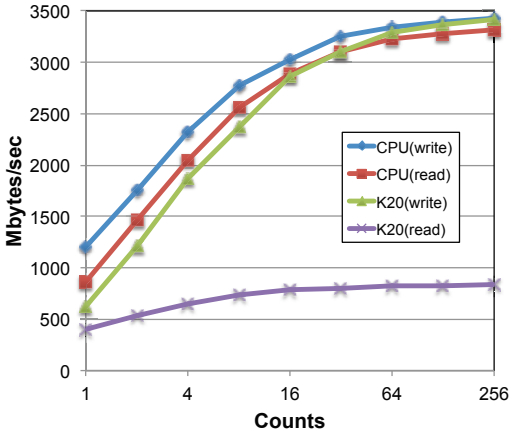

図 8 PEACH2-CPU, GPU 間の DMA 性能 (4096byte 固定)

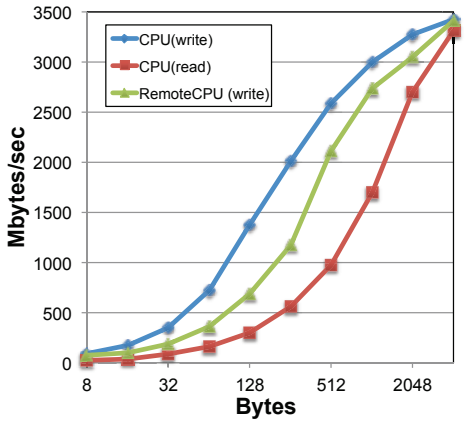

図 9 PEACH2-CPU, 隣接ノード CPU 間の DMA 性能 (255 回連続)

の DMA 書き込みについては，数 100MB/s 程度となり，著しく性能が低下することが分かった．

3.2 PEACH2 ボード間の転送遅延時間，転送性能

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．

測定方法は，以下の通りである．

(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，ボード A, B 間を PCIe 接続ケーブルで接続，経路制御のためのレジスタを適切に設定
(2) PEACH2 ドライバ中でクロックカウンタを読み出し，書き込む先のバッファを用意
(3) ボード A の PCIe 空間の中で，ボード B に割り当てられたアドレス領域としてバッファに対して 4byte store
(4) ボード A からボード B に自動的にパケットが転送
(5) ボード B から対象のバッファに書き込まれる
(6) バッファの値が書き換わるまでポーリングし，クロックカウンタを読み出し，(2) との差を測定

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はある．

次に，DMA を用いた，PEACH2 内のバッファから隣接ノードの CPU メモリへの DMA 書き込み性能を測定した．**図 9** に，255 回連続転送した場合の結果を示す．PEACH2 間の転送遅延が影響し，短いサイズではスループットの低下が見られるが，4Kbyte ではローカルの場合とほぼ変わらない結果が得られた．

なお，現状の DMA コントローラでは，実装上の制約からローカルホストからの DMA 読み出しとリモートホストへの DMA 書き込みを同時に実行することができない．そのため，ローカルホストのメモリ内容をリモートホストに送る場合，一旦 PEACH2 内部のメモリに読み出し，終了した後で PEACH2 内部メモリからリモートホストに書き込む必要がある．しかしながら性能に与える影響が大きいため，DMA 読み出しを実行しながら，到着したデータを

順にパイプライン的に DMA 書き込みを行えるよう，DMA コントローラの改良を行っている．

4. 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．

2.2 節でも述べたように，NVIDIA 社 GPU 向けの次期プログラム開発環境 CUDA 5 では，CUDA Computing Capability 3.5 以上を持つ 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 を利用するが，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．

5. おわりに

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 年度中に完了する予定である．また，2012 年度中に 8 ノードの実験用ミニクラスタを導入し，動作確認や性能改善を行う．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．

2013 年 10 月には数十ノード程度からなる HA-PACS/TCA 部のクラスタ導入を予定しており，800TFlops の性能を持つベースクラスタ部と合わせて，1PFlops を超える HA-PACS システムが完成する予定である．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 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, 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-KSeriesOverviewLR.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] Altera Corp.: *IP Compiler for PCI Express user guide*. http://www.altera.com/literature/ug/ug_pci_express.pdf.
[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.