Vol.2009-HPC-121 No.25
2009/8/5

情報処理学会研究報告
IPSJ SIG Technical Report

ランダムアクセス型応用のための PCI express 越しに機能メモリをアクセスするアーキテクチャ

田邊 昇[†]　芳野裕子[††]　小川裕佳[††]
佐々木愛美[†]　高田雅美[††]　城 和貴[††]

本報告では, 筆者らが提案した可視化装置における「アクセラレータが PCI express 越しに機能メモリをアクセスする」という概念を，ポストベクトル機向けに整理する．ボリュームレンダリングを例として提案装置向けアルゴリズムを検討した．さらに，本用途への GPU 利用の可能性や PCI express の特性を評価した．乱数やボリュームレンダリングの履歴 index を用いた配列間接参照実験では、ミッドレンジ GPU は理論ピークの 1.3%の実効バンド幅でデバイスメモリをアクセスした．このバンド幅は PCI express と機能メモリにより置き換え可能と考えられる．よって，本概念はランダムアクセス型応用を加速するアクセラレータ向けのメモリ容量向上策や、ECC 付加による信頼性強化策として有望である。

Architecture based on Accesses to Functional Memory Through PCI express for Applications with Random Accesses

Noboru Tanabe[†]　Yuko Yoshino[††]　Yuka Ogawa[††]
Manami Sasaki[†]　Masami Takata[††]　and　Kazuki Joe[††]

In this report, we organize the concept on our visualization machine as for the post-vector supercomputer. The proposed concept is the architecture with accelerators accessing functional memory through PCI express. We investigate the volume rendering algorithm for the machine based on the concept. Moreover, we evaluate the feasibility of using GPU and the performance of PCI express for this use. In the experiment of indirect array accesses using random index and index trace on volume rendering, a mid-range GPU accessed device memory with 1.3% bandwidth compared to the theoretical peak. This bandwidth seemed to be able to be replaced with PCI express and functional memory. Therefore, this concept is promising for accelerator running applications with random accesses both to overcome limitation of device memory capacity, and to improve robustness of memory using ECC.

1. はじめに

ベクトル型スーパーコンピュータは大規模な数値シミュレーションを長い間支えてきた．しかし，近年 Top500 リスト[1]にランキングされている HPC 向け計算インフラは，COTS ベースの PC クラスタや，IBM BlueGene[2]を代表とする ASIC ベースの CPU を用いた超並列計算機や, その中間的な構造を有する Cray XT 系列[3]を代表とする超並列計算機で占められるようになった．その結果，ベクトル型スーパーコンピュータ市場は冷え込んでしまった．近年では国費を背景に次世代スーパーコンピュータ開発プロジェクトが進んでいるが，膨大な開発費用を伴う製造から参加企業が撤退した．今後はその代替インフラ(ポストベクトル機)をどのように作るかという課題がある．

ベクトル型スーパーコンピュータの演算能力は COTS の CPU や GPU などで代替可能なケースが多い．GPU の演算能力は既に 1 TFLOPS を超えており，それを生かした GPGPU 研究の成功例[5]は数多く報告されている．一方，キャッシュアーキテクチャや GPU の統合メモリアクセスでは救いきれない大容量メモリに対するランダムアクセスを主体にするアプリケーションは，メモリシステムとの不整合から必ずしも COTS がベクトル型スーパーコンピュータを代替しきれない．ベクトル型スーパーコンピュータの存在価値を形成する要因の本質は，その演算能力の高さなのではなく，大容量で不連続アクセスに強いメモリシステムであると言っても過言ではない．

GPU 基板上のデバイスメモリの容量は現状では最大でも 4GB であり，それを超える大規模データを処理する場合、工夫を行わないと PCI express のようにバースト転送しか効率的に実行できない通信経路がボトルネックになる．さらに大容量のメモリを安定稼動するにはエラー訂正コード(ECC)の導入が必須である．

一方，筆者らは十数 GB から数十 GB のボリュームデータをリアルタイムでレンダリングする可視化装置[6][7][8]を提案した．ボリュームレンダリングは、大容量メモリに対するランダムアクセスを主体にするアプリケーションの代表格である．その加速を行う提案システムは，高い演算能力を有するアクセラレータ（SpursEngine[9]または GPU）と，DIMMnet-3[10][11]のように Gather 機能を有する拡張大容量機能メモリを PCI express によって接続する並列システムである．

本報告では，上記の「アクセラレータが PCI express 越しに機能メモリをアクセスする」という概念を，ポストベクトル機向けに整理する．そのキラーアプリケーションの一つであるボリュームレンダリングをモチーフに，上記システム上での加速を行う

† 株式会社　東芝
Toshiba corporation
†† 奈良女子大学
Nara women's university

ためのアルゴリズムの検討を行う．さらに，ボリュームレンダリングのメモリアクセスパターンや, DIMMnet-3 と組み合わせた時に発生する PCI express アクセスパターンをベースに，提案システム構成における GPU の適合性も検討する．

2. 提案システム

2.1 既存アクセラレータの問題点

これまで，HPC 用途へのアクセラレータとしては Tsubame[12]システムに採用された ClearSpeed[13]や RoadRunner[14]システムに採用された Cell/B.E.(PowerXcell 8i[15]) が代表的である．さらに，CUDA 言語の登場に伴い，近年，研究レベルでは GPU の表示機能以外への一般応用である GPGPU の研究に注目が集まっている．Nvidia 社の Tesla などは HPC 用途をターゲットにしており, 倍精度演算をサポートするとともに，単精度演算 1TFLOPS クラスの演算能力を有する．

一方，これらのアクセラレータはメモリ容量の拡張性，アプリケーションによっては演算能力に見合った通信能力，不連続アクセスに対する実効バンド幅に問題を抱えている．特に GPU ではメモリのエラー訂正機能の欠如にも問題があり，オンボードのメモリの大容量化やその並列化によって数十 GB クラス以上のシステムメモリ容量を長時間安定動作させるには課題がある．

2.2 提案アーキテクチャ

2.2.1 基本概念

図 1 に提案アーキテクチャの基本概念を示す．PCI express 等の高バンド幅な標準 I/O を介してアクセラレータと機能メモリを結合する．PCI express スイッチ等の共有アドレス空間上にデバイスをマップする機能を有する結合網を介してこれらを多数結合する．このような方式により，メモリ容量とメモリバンド幅と結合網バンド幅と演算能力のバランスを維持したスケーラビリティ向上，低消費電力化と低コスト化を実現する．

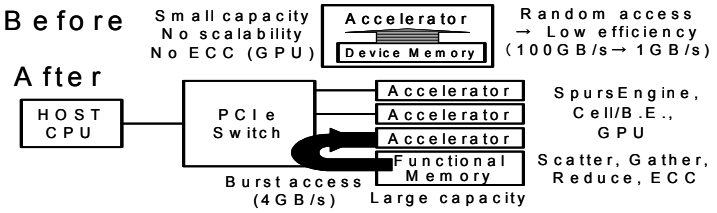

図 1　提案アーキテクチャの基本概念

機能メモリはアクセラレータの外付けデバイスとして，エラー訂正機能が付いた拡張メモリとして用いられる．さらに機能メモリはホストの主記憶と異なり, PCI express 等の標準 I/O を通過するデータ量を削減する機能や転送効率を向上させるための機能を有する．機能メモリの具体的な機能として代表的なものは, Scatter/Gather(分散/収集) 機能である．その他にも機能メモリ上の複数のデータから中央値(Median)等を出力するような簡単な演算機能(Reduce 機能)も限られた I/O 転送バンド幅の節約に寄与する．

PCI express スイッチを搭載し PCI express を木構造状に分岐増設する製品が複数市場に出てきている．一部のスイッチ製品[16]ではリーフノード間のルーティングをサポートしている．これらを階層的に接続すれば，低コストで PCI 空間上にマップされた多数のデバイス間での読み書きを多数並列に実行できる．

なお，上記の基本概念は国際会議 IWSV'09 における口頭発表および SWoPP'08 において論文発表[6]された大容量データ向け可視化装置を実例として述べられている概念である．ただし，この概念は可視化装置に限定されるものではない．

Scatter/Gather 機能を有する機能メモリとして最も代表的なものは筆者らが研究開発してきた DIMMnet-2[17][18][19]およびその改良版である DIMMnet-3[10][11]である．米国の二つの国立研究所(ORNL および SNL)が 2008 年より Scatter/Gather 機能を有する機能メモリを有する超並列計算機[20]の研究開発に着手しており，この種の機能メモリが HPC システムで実際に用いられていく兆しがある．膨大な国費を背景としてもベクトル型スーパーコンピュータが経済的問題から開発が困難な状況が現実化しており，この種の機能メモリが COTS にベクトル機的性質を付加する代替技術[17]として有望である．よって，DIMMnet-3 に類似した機能メモリが今後市場に出てきて，本基本概念を用いたシステム構築に応用できる可能性が高まりつつある．

つまり，本基本概念は，可視化装置のみならず，COTS の CPU と GPU 等の演算アクセラレータとベクトル機的なメモリアクセラレータ（機能メモリ）を組み合わせた将来の COTS ベースのベクトル機代替システムにも敷衍できる可能性を秘めている．

2.2.2 処理の流れ

図 2 に基本概念に基づく機能メモリアクセス処理の流れを示す．

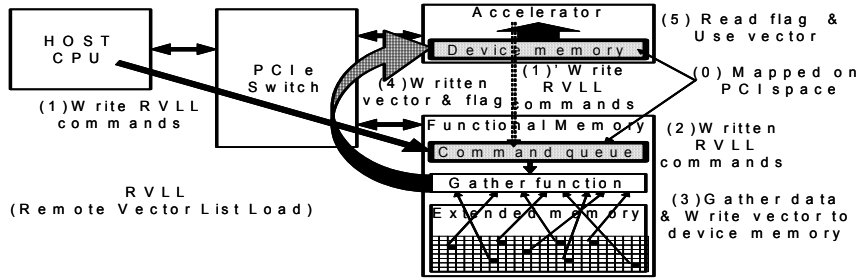

図 2　基本概念に基づく機能メモリアクセス処理の流れ

(1)機能メモリ(例えばPCI express子基板を実装したDIMMnet-3)へのコマンドキューはPCI空間上にマップされる．よって，ホストまたはアクセラレータはアクセスしたい機能メモリに割り当てられた上記のコマンドキューに対応するアドレスに所定フォーマットでコマンドを書き込む．

(2)上記に応じて，PCI expressスイッチによって構成された木状結合網によって上記書き込みトランザクションは実行され，アクセスする機能メモリのコマンドキューにコマンドが書き込まれる．

(3)機能メモリはコマンドキューからコマンドを取り出して，記載された内容の機能(例えば遠隔リストベクトルロード：RVLL)を実行し，指定があれば応答データや完了フラグをコマンドに記述されたアドレスに書き込む．

(4)上記に応じて，PCI expressスイッチによって構成された木状結合網によって上記書き込みトランザクションは実行され，コマンドは終了する．

(5)アクセラレータは十分に余裕のあるタイミングでコマンドを起動できない場合は必要に応じて上記完了フラグをポーリングし，完了していれば後続の処理(デバイスメモリ上に連続化かつアライメント調整されて格納済みのベクトルデータに対するSIMD演算ループ(Cell/B.E.,SpursEngine)やWarp (GPU))を実行する．

2.2.3 SpursEngineとDIMMnet-3の組合せ

SWoPP'08において発表した可視化装置のハードウェア構想[6]では，前述の基本概念を適用し，アクセラレータとしてSpursEngineを用いる図 3 左図に示すようなシステム構成を提案した．SpursEngineを用いる場合はH.264ハードウェアエンコーダが内蔵されているため，遠隔可視化における表示処理において有利であるとともに，情報家電機器に利用されることを想定された設計であるため GPU と比べて消費電力とコストの将来性の面で有利である．

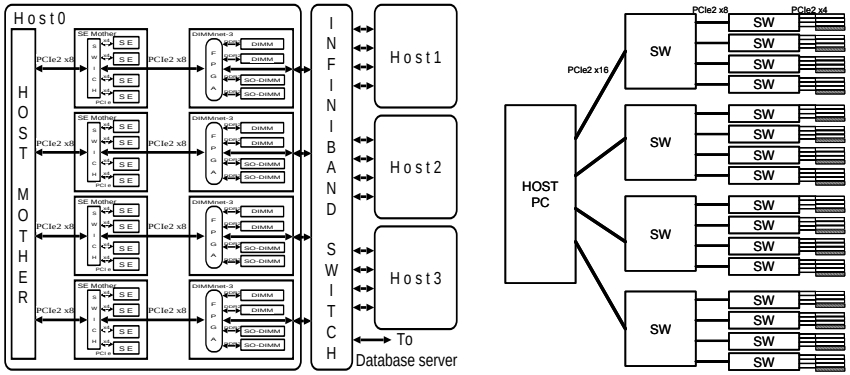

図 3　市販 PCI express スイッチ BOX を用いた結合方法（左：旧，右：新）

SpursEngine と DIMMnet-3 を接続する下位階層は PCI express スイッチで接続されるが，その構想段階では図 3 左図に示すように上位階層を Infiniband スイッチによって 4 台の PC が接続されることとしていた．その後，図 3 右図に示すように Infiniband の部分も市販の PCI express スイッチ BOX に置き換える案を検討中である．この変更によりスイッチ部分の低コスト化と，PCI express と Infiniband 間のルーティング機能といった新規開発部分が省略でき，開発リソースや部品コストの削減を図ることができる．なお，図 3 に示すように最上位階層のスイッチのトポロジーがクロスバ接続(Infiniband)からトリー接続(PCI express)に変更になるため，下位階層は PCI express Gen1 や x4～x8 構成で，上位階層は PCI express Gen2 や x8～x16 構成での接続とするなどで，上位階層ほど高いバンド幅の通信路で接続することが望ましい．

ただし上記の変更案はコスト重視の戦略であり，性能面ではホストの負荷や，上位階層の PCI express の負荷が上昇するなど，採用においては更なる検討が必要である．本報告では DIMMnet-3 へのコマンド投入時の負荷面からの検討を行う．

2.2.4 GPUとDIMMnet-3の組合せ

5 年 10 年のレンジで考えると Nvidia 社や AMD 社がハイエンド GPU でビジネスを継続できるか否かに疑問が残るものの，現状では HPC ユーザからも魅力的な製品を市場に供給できており，CUDA 言語の登場に伴って GPGPU 研究は大変な盛り上がりをみせている．

GPU は PCI express バンド幅と演算能力のバランスという観点では，SpursEngine よりも演算能力の高さを重視した仕様設定がなされている．一方，これまでの最適化前のボリュームレンダリングプログラムによる予備評価[7][8]から，SpursEngine で 3 個の SPU に実質的な演算をさせる場合は PCI express x4 を飽和させるだけの演算能力が引き出せていない．

上記を踏まえ，本報告では SpursEngine だけでなく，GPU の利用も短期的には有望な選択肢であるという観点から検討を行う．

複数の GPU と DIMMnet-3 を PCI express 経由で接続する場合，PCI express スイッチのピン数制約によりレーン数は x1～x8 の範囲で接続することになる．ハイエンド GPU の一つである Nvidia 社 GTX280 のデバイスメモリの最大バンド幅は 141.7GB/s にも及ぶが，それに比べると PCI express x1～x8 のバンド幅は 1/400～1/50 に過ぎない．ただしボリュームレンダリングが発生するランダムなアクセスに対する実効バンド幅は GPU の場合も Cell/B.E.や SpursEngine 同様に大幅に低下してしまう．

つまり，ボリュームレンダリング実行時に大幅に低下した実効バンド幅でデバイスメモリをアクセスしている GPU と同等クラスの実効バンド幅を PCI express や DIMMnet-3 が供給できるならば，メモリ容量制約の克服法として本アプローチは有望となる．

なお，理研 HPC シンポジウム'08 における成瀬氏の発表スライド[21]によると 100MB

の 4 バイト乱数入り配列に対するランダムアクセス遅延は Nvidia 社 GTX280 の場合 520ns かかるということが報告されている．これはスレッド数 32 での測定であり，実効バンド幅に換算すると 246MB/s になる．スレッド数を増やせばバンド幅が向上すると考えられるが，内蔵するプロセッサ数と同数の 240 スレッドまでバンド幅がスレッド数に比例すると仮定したとしても高々1.845GB/s に過ぎない．このバンド幅は PCI express Gen1 x4 の 1.845 倍でしかなく，上記のシナリオが成立すると予想される．

ただし，GPU を用いる戦略の場合，GPU の処理効率向上のためには GPU 内部でのスレッド数を多く取る必要がある．このため，ボリュームレンダリングの場合は投影画面のピクセル数に限界がある(2K 解像度の高精細ディスプレイを使用するとしても光線の並列度は 2K の 2 乗に過ぎない)ため，GPU の個数をあまり多くしても効率が落ちる可能性がある．

3. 検討対象とするアプリケーション

3.1 ボリュームレンダリング

本報告ではランダムアクセス型アプリケーションの典型例としてボリュームレンダリングを検討対象とする．ボリュームレンダリングは大規模な数値シミュレーション結果や CT・MRI・油田探索等各種センサーによる観測データを分析するために広く利用されている．半透明度であるボクセルと呼ばれるデータの三次元配列をボリュームと呼ぶ．ボリュームレンダリングは，半透明なボリュームをある視点から眺めた時の見え方を描画するアプリケーションである．

ボリュームレンダリングにはいくつかの基本アルゴリズムがあるが，そのうち最も近似が少なく良い画像が得られるのが Ray casting 法[22]である．Ray casting 法は視点から二次元投影面(スクリーン)上の画素を貫くように光線を延ばし，光線がボリュームデータと交差する位置のボクセル値をαブレンディングと呼ばれる計算方法で積算してピクセルの輝度を決定する方法である．計算量が多い上に，メモリアクセスが大容量メモリに対してランダムになるため，高速化にあたっては工夫が必要である．

3.2 予備評価で用いた単純 Ray casting

筆者らはこれまでの評価研究[7][8]においてあまり工夫はしていない単純な Ray casting 法のプロトタイププログラム(以下 Proto-ray と呼ぶ)を Playstation®3 上で実行させることによって提案方式のフィージビリティを検討してきた．これによりいくつかの重要な知見を得ることができたが，実用的な処理速度を現実的なハードウェア量で実現するには，様々な最適化技法によって改善する必要がある．

3.3 C-ray

C-ray[23]は IBM 社主催のコンテストである Cell/B.E. Challenge’07 において America region の第 3 位に輝いた Cell/B.E.向けのボリュームレンダリングプログラムである．C-ray も上記同様に Ray casting 法をベースにしている．後述するように様々な最適化が施されており，ボリュームデータ全体が主記憶に納まる範囲では高い完成度を持ったプログラムである．我々はその作者である Kim 氏よりソースコードの提供を受けることができた．本報告では，論文に記載されている最適化手法とソースコードを照合しつつ，Proto-ray への最適化手法移植について検討を行う．

4. 提案システム向けアルゴリズムの検討

本節ではホスト PC とアクセラレータと機能メモリからなるヘテロジニアス環境における処理の分担を中心に，実例に即してアルゴリズムを検討する．主に C-ray の論文やソースを参考にしながらターゲットシステムへの適合性を検討する．

4.1 負荷分散戦略

ターゲットシステムでは各アクセラレータまたはホスト PC から Gather 機能付メモリ(DIMMnet-3)に対して遠隔リストベクトルロードコマンドによってデバイスメモリへのプリフェッチを起動することができるものとする．

DIMMnet-3 上の大容量メモリ(最大 28GB)に入ってしまう範囲では 1 個の DIMMnet-3 上にボリュームデータ全体を格納できるので，PCI express スイッチによる木状結合網のバンド幅制約をあまり気にすることなく，個々のアクセラレータから最もアクセスしやすい位置にある DIMMnet-3 に対して遠隔リストベクトルロードコマンドによってプリフェッチを起動する．

このようなシステム上では単純な光線単位の負荷分散が有効と考えられる．その理由は以下の 3 点である．

(1)Ray casting 法に基づくボリュームレンダリングでは，一般に光線間の演算に依存関係はないため並列化に伴う計算途中の通信が不要である．
(2)上記の機能メモリを用いた遠隔リストベクトルロードを用いたプリフェッチには光線方向に分割されないようにすることで PCI express 上での転送のバースト長を大きく取ることができ転送効率が向上する．
(3)1 本の光線に伴う計算は 1 個のアクセラレータで閉じるので後述する Early ray termination(ERT)法が有効に機能する．

4.2 Empty space skipping

Empty space skiping(ESS)法[24]はボリュームデータ内で,何も表示するものが無い空領域の計算を省略する方法である．ボリュームが空である情報を格納することで,光線を飛ばしている間,計算を省略する階層的なデータ構造を構成する．本手法はボリュームに空領域が多い時に役立つ．この手法で見込める加速率は，例えば空領域が 90%程度と多いボリュームでは 10 倍程度となる．

本手法は歴史が古く，C-ray をはじめとする実用的なボリュームレンダリングプロ

グラムにはほぼ必ず組み込まれている方法である．

ESS は Proto-ray には組み込まれておらず，効果が高い方法なので今後，実装する．ただし，C-ray に実装されている空領域判定アルゴリズムは 8 分木構造にボリューム上のデータ有無を階層的に管理し，リストをたどるという処理になるため，ターゲットシステムのアクセラレータ部分である SpursEngine や GPU には不向きな方法であると考えられる．よって，Cell/B.E.と異なって PPE がない SpursEngine 上の SPE で実行するとどの程度の性能低下が見られるか調査する必要がある．

4.3 Early ray termination

Early ray termination(ERT)法[24]は輝度値が所定の閾値に達した後の光線を飛ばす作業を中止することで計算量を削減する高速化手法である．輝度値が所定の閾値に達すると，一つの光線上のある点より奥のサンプリング点に関する計算が最終的な見え方に殆ど影響がないことに基づく一種の近似である．本手法はボリュームに空領域が少ない時に役立つ．

本手法も歴史が古く，C-ray をはじめとする実用的なボリュームレンダリングプログラムにはほぼ必ず組み込まれている方法である．

ERT は Proto-ray には組み込まれておらず，ターゲットシステムへの適用上の否定的な特殊事情は無い．むしろ，一本の光線方向にプロセッサへの分割境界が入ってしまうタイプ(例えば部分画像を生成した後で合成する並列処理)のシステムよりも，1 本の光線を 1 つの SPE が最初から最後まで担当する負荷分散は ERT によって省略できる場合が多くなる傾向にある．よって，ERT はターゲットシステムと相性は良いと考えられる．

4.4 Streaming

C-ray では Cell/B.E.の PPE 上に，Empty space skiping を適用しつつ空でない領域の Ray が交差するサブボリューム(光線セグメント)を決定して SPE に仕事(光線の視点からのオフセットと長さ)を割り当てるステージを割当て，ERT を適用しながら不透明度の積算をするステージを複数の SPE 上に設けてそれらをパイプライン的処理する．この場合，PPE がネックになりやすいので，次節のような PPE の負担を軽減する最適化が必要になる．

この最適化は Proto-ray 上では行われていない．ただし，ターゲットシステムを想定した状況での性能評価では別スレッドがプリフェッチを行うことが仮定されており，実質的にはこの最適化を行うことが前提になっている．プリフェッチスレッドは C-ray の PPE 上の処理と同様な処理を行うことになると考えられる．

4.5 近似による空判定計算の削減

PPE の負担を軽減する最適化として C-ray は近似による空判定計算削減と，それに伴う SPE 側の処理増加の削減を，ハッシュテーブルを用いた Refining という方法で行っている．その効果は高いので，空領域判定部分が相対的に重くなるターゲットシステム上では有効性が高いと考えられる．

4.6 PPE 有無の違いへの対応

C-ray は Cell/B.E.上での最適化を行ったプログラムであり，PPE の処理を最適化することをメインにしている．ところが，ターゲットシステムで用いられる SpursEngine や GPU 上には分岐処理に強いとされる PPE は存在しない．そのため，C-ray で PPE に実行させていた処理を分岐処理に強くない SPE や GPU に担当させることが良いかどうか検討する必要がある．

なお，Proto-ray では分岐処理に弱い SPE での処理速度を向上させるために if 文の代わりにマスク演算処理に置き換えて分岐命令の実行を削減する最適化は既に適用済みである．GPU においても似たような技法が使えると思われる．

4.7 プリフェッチスレッドの動作場所

ターゲットシステムのアーキテクチャ上は機能メモリへのプリフェッチ(遠隔リストベクトルロードコマンド)を発行するプリフェッチスレッドはホスト PC 上や，アクセラレータ上の CPU コア(例えば SpursEngine の SPU)で実行させることもできる．この項目については双方を実装し，評価によって優れた方式を選択するものとする．

5. 性能評価

5.1 GPU を用いる場合の基本性能

本節の評価は GPU を本提案システムのアクセラレータとして使う場合の適性を PCIexpress の特性と，デバイスメモリのランダムアクセス性能の観点から評価する．

5.1.1 評価環境

本節の評価に用いられた評価環境を表 1 に示す．

表 1　GPU 評価環境

CPU	Intel®[a] Core(TM)2 Duo CPU E8400 @ 3.00GHz
主記憶	3.25GB (DDR2 dual channel)
GPU	Nvidia GeForce9800GT (MP 数 16，メモリバンド幅 57.6GB/s)
マザーボード	MSI 社 MS-7360
GPU 装着スロット	PCI express x16 Gen1 (最大バンド幅 4GB/s)
OS	Microsoft®[b] Windows®XP Profesional Version 2002
コンパイラ	Microsoft® Visual Studio® 2005
CUDA	Version 2.0 Beta2 for 32-bit or 64-bit Windows Vista® or XP

a Intel、Intel Core は、アメリカ合衆国およびその他の国における Intel Corporation の商標です。
b Microsoft、Windows、Visual Studio、Windows Vista は、米国 Microsoft Corporation の、米国、日本およびその他の国における登録商標または商標です。

5.1.2 PCI express におけるバーストアクセスバンド幅

DIMMnet-3 のリストアクセスにおいては読み出すデータ型にもよるが，読み出しデータと同程度の大きさの index 情報をコマンドとして書き込む必要があるので，コマンド書き込み時に実現できるバンド幅は軽視できない．cudaMemcpy 関数を用いて GPU が PCI express 越しにホストの主記憶をアクセスする場合のバンド幅を測定した結果を図 4 に示す．GPU から見て DIMMnet-3 はホスト主記憶の一部に見え，バースト的な書き込みについては DIMMnet-3 のコマンドキューもホスト主記憶もほぼ同等である．よって，このバンド幅は GPU が DIMMnet-3 に対してコマンドを書き込む場合に GPU→ホスト方向の PCI express で消費されるバンド幅の近似値を与える．1 回だけ転送した場合のバンド幅はデータサイズ 16KB で 500MB/s 弱，1MB で 1.1GB/s 程度であり，意外に低いものであることが判った．リストアクセスにおいてはこのバンド幅がボトルネックになる可能性があることが判った．一方，1000 回繰り返し転送を行った時に得られる継続バンド幅は右図のようになる．16KB で単発の時より高い 663MB/s のバンド幅が得られる．PCI express 利用時は繰り返し転送して PCI express の空き時間を埋めることが望ましい．ただし cudaMemcpy 関数起動オーバーヘッドなどによりデータブロックサイズがあまり小さいと繰り返しによる効果にも限界があるので，DIMMnet-3 のコマンドキューのサイズは大きいほど良いと考えられる．

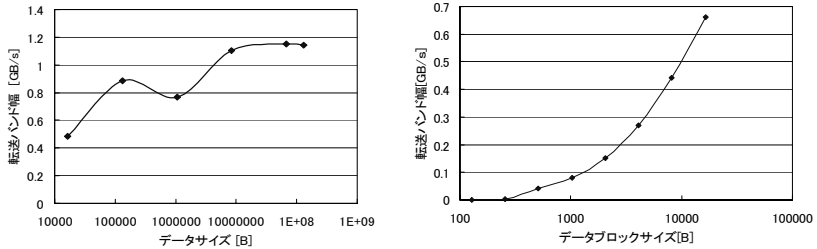

図 4　cudaMemcpy 関数による転送バンド幅（GPU→ホスト，左：単発，右：継続）

一方，DIMMnet-3 が Gather されたベクトルデータを GPU のデバイスメモリに書き込む場合，これは DIMMnet-3 上のハードウェア(DMA コントローラ)が GPU のデバイスメモリに向かってバースト書き込みを行うことになる．そのバンド幅はホスト上のソフトウェアオーバーヘッドも含む cudaMemcpy 関数を用いてホストが PCI express 越しに GPU のデバイスメモリをアクセスする場合のバンド幅を下回ることは無いと考えられる．上記の cudaMemcpy 関数を用いた GPU のデバイスメモリへのバースト転送バンド幅を測定した結果を図 5 に示す．逆方向よりは高いバンド幅が得られるが，1 回だけ転送した場合のバンド幅は左図のようになりデータサイズ 16KB で 500MB/s 程度，1MB で 1.4GB/s 程度であり，意外に低いことが判った．一方，1000 回繰り返し転送を行った時に得られる継続バンド幅は右図のようになる．16KB で単発の時より高い 663MB/s のバンド幅が得られる．PCI express 利用時は繰り返し転送して PCI express の空き時間を埋めることが望ましい．

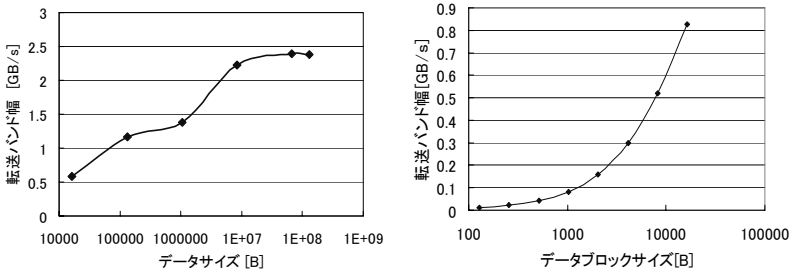

図 5　cudaMemcpy 関数による転送バンド幅（ホスト→GPU，左：単発，右：継続）

5.1.3 デバイスメモリへのランダムアクセスバンド幅

本評価では乱数で発生したホストの主記憶および GPU のデバイスメモリ上のアドレスに対する GPU からのアクセスバンド幅を測定する．現状の GPU には Cell/B.E.や SpursEngine の DMA リスト[24]に相当する機能は無いので，ホストの主記憶をアクセスさせる場合は cudaMemcpy 関数を用いて実装する．

本測定により，ボリュームレンダリングのメモリアクセスパターンが乱数で生成したアドレス列と同程度にランダムだった場合に，そのアプリケーション処理性能を支配すると考えられるデバイスメモリや GPU から見たホスト主記憶へのアクセスバンド幅の近似値を得ることができる．

測定においては，まず，共有メモリに index 配列 sid を 2048 要素分，index 配列におけるスレッドごとの担当場所の先頭のオフセット値配列 snum_p を 512 要素分，データ読み込み用配列 s_data を 1024 要素分確保した．その上で，ホスト上で index 配列を乱数で初期化後に, 9 個おきに 1 個をスレッドが担当する index 値の個数 8 で初期化する．その後，デバイスメモリ経由で共有メモリ上の sid および snum_p に転送する．そこまでのポインタ情報準備時間をあらかじめ測定しておき，上記ポインタ情報を用いたデバイスメモリ上データ配列 g_data への間接参照を実行する．バンド幅は全転送処理時間からポインタ情報準備時間を差し引いた純粋な間接参照時間を元に算出した．

その結果，デバイスメモリ上の 4 バイトデータ型の配列(要素数 4096)への 512 スレッド，16 ブロックという設定でのランダムアクセスバンド幅は 758MB/s であった．これは図 3 で示されているバンド幅で近似される、PCI express gen2 x8 化された

DIMMnet-3 により 16KB 程度に連続化後の DIMMnet-3 から GPU への転送バンド幅と概ね同程度である．

また，このバンド幅は純粋に間接参照部分のみの時間に基づくものであるので，ボリュームレンダリングをさせる場合には，これ以外に視線とボリュームの交差点から index を生成する処理や, 間接参照によって読み込まれたデータを用いてのαブレンディング計算などの処理が必要になる．よって，実際にデバイスメモリに対して発生するメモリバンド幅はその数分の 1 に落ちることが予想される．

つまり，本実験で用いたミッドレンジの GPU を用いた場合には，DIMMnet-3 との接続に用いる PCI express は GPU1 個あたり PCI express gen1 x4(1GB/s)程度で十分にデバイスメモリを直接アクセスしているような体感速度が得られることがわかる．

5.1.4 アクセストレースを用いたデバイスメモリへのアクセスバンド幅

本評価ではこれまでの予備評価で用いた Proto-ray で発生させたホストの主記憶および GPU のデバイスメモリに対する GPU からのアクセスバンド幅を測定する．これは前述の乱数アドレス列よりも実際のアプリケーション実行時の挙動を正確に表している．Proto-ray は ERT や ESS を全く行っていないので，画像の内容にはアクセスパターンは依存しない. 4 バイトデータ型の配列(要素数 4096)に対して Ray あたり 16 または 17 回のアクセスリストを有する DMA リスト転送[24]を行っている．その DMA リスト生成部分のプログラムを改造し，1 本の Ray に対応する index 列とその個数をファイルに出力してトレースファイルとした．前述のランダムアクセスバンド幅測定プログラムの index 配列を，このトレースファイルで初期化するように改造して同様のバンド幅測定を実施した．

その結果，index 配列を乱数により初期化した場合の結果(758MB/s)と，Proto-ray で発生させた index 列により初期化した場合の結果(748MB/s)には，大きな違いがないことを確認できた．前者より後者がややバンド幅低下した理由はアドレスのパターンの違いというよりは, 各スレッドが担当するアクセス回数のばらつき(前者は 8 回固定で, 後者は 16 または 17 回)によるものと考えられる．

5.2 空領域判定部の実行場所選定のための予備評価

Empty space skipping は処理すべき光線の本数を大幅に削減し，性能向上に有効な方法である．しかし，この中核をなす空領域判定部は C-ray においては，PPE 上で実行するようにプログラムされている．PPE は予定しているターゲットシステムには存在しない．そこで本節では，これを SPE やホスト CPU に移動したらどのような性能が得られるかについて評価を行う．

方法は C-ray における Empty space skipping 処理における空領域判定部の関数の実行時間を元に，PPE・ホスト CPU の 2 つのプロセッサ上での性能比較評価を行った．評価環境は PPE と SPE については Playstation ®3，ホスト CPU は Intel® Xeon(TM) 3.00GHz，主記憶 1GB，OS は Linux，コンパイラは全て gcc である．

表 2　空領域判定部の実行時間

Volume size	PPE [ms]	HostPC [ms]	Ratio
16x16x16	0.00011476	0.0000450227	2.549
32x32x32	0.0001182896	0.0000464604	2.546
64x64x64	0.0001778318	0.0000624361	2.848
128x128x128	0.0005335296	0.0000808137	6.602
256x256x256	0.001187948	0.0000954884	12.441

その結果を表 2 に示す．サイズが 128 を超えて大きくなるにつれ PPE とホストの性能差は拡大する．PPE もホスト PC もキャッシュベースのほぼ同じ周波数の CPU であるが，キャッシュ容量は PPE が 512KB に対し，ホスト PC は 2MB のキャッシュを内蔵しており大きな差がある．ボリュームデータそのものをアクセスする処理が SPE 側に移動すると，空領域判定のためにたどる木状構造体の全てまたは大半がホスト PC ではキャッシュに納まり，PPE では溢れて大きな差が生じていると考えられる．キャッシュの容量が性能に敏感であることから，アクセス範囲をより小容量で済ませなければ性能が出にくい SPE にこの処理を行わせるのは適切ではないと考えられる．ホスト CPU は PPE よりも大幅に高性能という結果は，空領域判定部をホストに移動すべきことを示唆している．しかし，1 台のホスト PC を図 3 の右図のように木状の PCI express の頂点とする単純なシステム構成では問題があるので，今後，ホスト PC の構成や台数，DIMMnet-3 へのコマンドの与え方について改善すべきであると考える．

6. おわりに

本報告では，これまでの筆者らが提案し評価してきた可視化装置における機能メモリとアクセラレータを PCI express ネットワークを用いて組み合わせる基本概念を整理するとともに，本概念を適用したターゲットシステムに適するボリュームレンダリングアルゴリズムの検討を行った．さらに，GPGPU への近年の期待と課題を鑑み，アクセラレータとしてこれまでの研究で採用を仮定してきた SpursEngine だけでなく，GPU 利用の可能性や PCI express の特性を定量的手法により検討した．

本報告でまとめた基本概念は可視化装置のみならず，COTS の CPU と，GPU 等の演算アクセラレータと，DIMMnet-3 等のベクトル機的なメモリアクセラレータ（機能メモリ）を組み合わせた将来の COTS ベースのベクトル機代替システムにも敷衍できる可能性を秘めている．

本概念を適用したターゲットシステムに対するボリュームレンダリングアルゴリズムの検討においては，Cell/B.E.向けのアプリケーションである C-ray をベースに最適化技法の検討を行った．その結果，アクセラレータとして SpursEngine を用いる場

合と GPU を用いる場合の双方において，C-ray における Cell/B.E.の PPE で実行されていた処理をどこで実行させるかが当面の最大の検討課題であるという認識に至った．その予備評価として C-ray の Empty space skipping 処理における空領域判定部を PPE・とホスト CPU 上で実行した場合の性能比較評価を行った．その結果，ホスト CPU 上の実行が大幅に性能が良いが，アクセラレータを多数用いる場合はホスト CPU をマルチプロセッサ化するなどの強化が必要である．

GPU 利用の可能性に関する評価では，乱数 index やボリュームレンダリングプログラムから抽出した index による間接参照を，ミッドレンジ GPU は約 750MB/s の実効バンド幅でメモリアクセスしていることを観測した．PCI express のバースト転送バンド幅も測定し，上記バンド幅より高いバンド幅が PCI express Gen1 でも得られることを確認した．以上から，ランダムアクセス系アプリケーション実行時のデバイスメモリアクセスバンド幅は，DIMMnet-3 のような Gather 機能付き機能メモリと組み合わせてバーストアクセス化した際に期待できる PCI express の実効バンド幅で置き換え可能なレベルであることが明らかになった．つまり，GPU 等のアクセラレータが PCI express 越しに機能メモリをアクセスすることによりメモリ容量やエラー訂正能力向上を図る戦略は，ランダムアクセス系アプリケーションにおいて有望であると言える．

今後は C-ray 上の最適化技法の多くを移植するなどして性能向上を図る．また，機能メモリへのコマンド投入スループット向上とホストへの負担増加への対応策の検討も今後の課題である．さらに本アーキテクチャの CG 法(疎行列ベクトル積計算)などのボリュームレンダリング以外のランダムアクセス系アプリケーションへの適応性に関する検討も今後の課題である．

謝辞 本研究の一部(DIMMnet-3 の開発)は総務省戦略的情報通信研究開発推進制度(SCOPE)の一環として行われたものである．C-ray のソースコードをご提供いただいた Cell/B.E. challenge’07 米国第 3 位入賞者 Jusub Kim 氏に感謝いたします．

参考文献

[1] Top500 project : "Top500 Supercomputer site", http://www.top500.org/
[2] IBM : "BlueGene", IBM Journal of Research and Development, Vol.49, No.2/3 (2005)
[3] Cray Inc. : "Cray XT5", (2009) http://www.cray.com/Assets/PDF/products/xt/CrayXT5Brochure.pdf
[4] 理化学研究所, NEC : "次世代スーパーコンピュータ・システムの構成を見直す", http://www.riken.jp/r-world/info/release/press/2009/090514/index.html
[5] Nvidia : "CUDA Zone", http://www.nvidia.co.jp/object/cuda_home_jp.html
[6] 田邊, 佐々木, 中條, 城 :“大容量データ向け対話的実時間遠隔可視化装置の実現性検討”, 電子情報通信学会コンピュータシステム研究会 2008-CPSY-18, pp.43-48 (Aug. 2008)
[7] 田邊, 佐々木, 中條, 高田, 城 :“Cell/B.E.と DIMMnet を併用した大容量ボリュームレンダリングの並列処理性能", 情報処理学会ハイパフォーマンスコンピューティング研究会 2009-HPC-119, pp.7-12 (Mar. 2009)
[8] N. Tanabe, M. Sasaki, H. Nakajo, M. Takata, K. Joe : “The Architecture of Visualization System using Memory with Memory-side Gathering and CPUs with DMA-type Memory Accessing”, PDPTA’09 (Jul. 2009).
[9] Toshiba Corp. : “Toshiba starts sample shipping of SpursEngine(TM) SE1000 high-performance stream processor”, Press release 08 April, 2008, http://www.toshiba.co.jp/about/press/2008_04/pr0801.htm
[10] 田邊, 北村, 宮部, 宮代, 天野, 羅, 中條 :“主記憶以外に大容量メモリを有するメモリ/ネットワークアーキテクチャ", 情報処理学会計算機アーキテクチャ研究会 (Mar. 2007)
[11] N. Tanabe, H. Nakajo : " An Enhancer of Memory and Network for Cluster and Its Applications", IEEE PDCAT’08 (Dec. 2008)
[12] 松岡聡 : "TSUBAME の飛翔：ペタスケールへ向けた「みんなのスパコン」の構想", 情報処理学会研究報告 2006-HPC-107 pp.37-42 (Jul. 2006)
[13] ClearSpeed corp. http://www.clearspeed.com/
[14] Koch : "Road Runner System Overview", http://www.lanl.gov/orgs/hpc/roadrunner/pdfs/Koch%20-%20Roadrunner%20Overview/RR%20Seminar%20-%20System%20Overview.pdf
[15] IBM Corp. : "PowerXCell 8i processor product brief", http://www-03.ibm.com/technology/resources/technology_cell_pdf_PowerXCell_PB_7May2008_pub.pdf
[16] PLX Technology : "PCI Express 2.0 Switches - PCIe ExpressLane I/O Interconnect", http://www.plxtech.com/products/expresslane/gen2.asp
[17] N. Tanabe, M. Nakatake, H. Hakozaki, Y. Dohi, H. Nakajo, H. Amano : "A New Memory Module for COTS-Based Personal Supercomputing", Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'04), pp.40-48 (2004)
[18] 田邊, 安藤, 箱崎, 土肥, 中條, 天野 :"プリフェッチ機能を有するメモリモジュールによる PC 上での間接参照の高速化", 情報処理学会論文誌コンピューティングシステム,Vol. 46, No. SIG12 (ACS11), pp. 1-12 (Aug. 2005)
[19] N. Tanabe, H. Nakajo : “High Performance Computing and Database Processing with COTS and Extended Memory Modules”, HPC Asia’09 (Mar. 2009).
[20] Sudip S. Dosanjh : "Exascale Computing and The Institute for Advanced Architectures and Algorithms (IAA)", HPC User Forum, (Apr. 2008) , http://www.hpcuserforum.com/presentations/Norfolk/Sandia%20IAA.hpcuser.ppt
[21] 成瀬彰: " GPGPU クラスタの性能評価”, http://accc.riken.jp/HPC/Symposium/2008/naruse.pdf
[22] M. Levoy : “Display of surface from volume data”, IEEE Computer Graphics and Applications, Vol.8, No.3, pp.29-37 (May 1988)
[23] Jusub Kim, Joseph JaJa : “Streaming model based volume ray casting implementation for Cell Broadband Engine”, Scientific Programming, Vol.17, No.1-2, pp.173-184 (2009)
[24] M. Levoy : “Efficient Ray tracing of volume data”, ACM Trans. Graphics, Vol.9, No.3, pp.245-261 (1990)
[25] M. Kistler, M. Perrone, and F. Petrini: “Cell Multiprocessor Communication Network: Built for Speed” IEEE Micro, vol. 26(3) pp. 10-23, (May-June 2006)