Vol.2011-HPC-129 No.11
2011/3/15

GPU と拡張メモリによる疎行列ベクトル積性能の行列形状依存性軽減

田邊 昇[†] Boonyasitpichai Nuttapon[††] 中條 拓伯[††]
小川 裕佳[†††] 高田 雅美[†††] 城 和貴[†††]

GPU のメモリシステムにおける不連続アクセスにおける問題を解決するために筆者らは先行研究で Scatter/Gather 機能を有する拡張メモリシステムを提案した。疎行列ベクトル積に対する評価も行ない，提案拡張メモリシステム側の疎行列ベクトル積に対するスループットを PCI express のバンド幅以上にできることを示した。しかし，疎行列ベクトル積に関しては評価行列が 1 つのみであった。本報告ではその種類を増やした実験によって，提案方式の性能の行列形状依存性について評価した．その結果，提案方式は行列の形状に性能が鈍感で，高い実行性能を安定して提供できることがわかった．

Relaxing the Dependency of Matrix Shape on the Performance of Sparse Matrix Vector Product Using a GPU and an Extended Memory

Noboru Tanabe[†] Boonyasitpichai Nuttapon[††]
Hironori Nakajo[††] Yuka Ogawa[†††]
Masami Takata[†††] and Kazuki Joe[†††]

In order to overcome the problems of discontinuous accessing in memory systems of various processors, we have proposed an extended memory system which has an additional function of scattering and gathering. In this paper, we have confirmed the assumption on feasibility of throughput of the memory system is correct or not with a simulator based on DRAMsim2 with various matrices. As the result, we have confirmed proposed system has good stability for matrix figure.

1. はじめに

疎行列ベクトル積は連立一次方程式求解や固有値計算の中で用いられ，実行時間の大半を占めるため，最も重要な HPC 計算カーネルの一つである．

近年では GPU の高バンド幅を用いたその高速化の研究も盛んに行われている．広いビット幅構成にした GDDR 型 DRAM による GPU のメモリシステムは，同タイミングで動作するスレッド群が発生するアクセス群のアドレスが連続になるアプリケーション（例えば構造格子系）には効率的に動作する。一方、不連続アクセスになるアプリケーション（例えば非構造格子系）は，メモリシステムに多くの投資をしているベクトル型スーパーコンピュータを除いて，GPU を含む様々なプロセッサで大幅な性能低下が発生するという問題がある。特に対象問題が大きくなり列ベクトルが GPU のキャッシュには載りきらなくなると激しい性能低下が発生するようになると考えられる。

そのような問題を解決するために筆者らは先行研究[1]-[10]で Scatter/Gather 機能を有する拡張メモリシステムを提案した。文献[8] [9] [10]では疎行列ベクトル積においても評価を行ない、有効性を示してきた。ただし，文献[10]の疎行列ベクトル積についてのメモリシステムを含む詳細なシミュレーションは，GPU 上の従来方式で低い性能しかえられない 1 種類の疎行列とランダムアクセスでしか行っていなかった．このため，他の疎行列でも安定してスループットが得られるかどうかは明らかになっていなかった．また，疎行列の格納形式は最も広く用いられている CRS 形式のみで，アクセス順序は CRS 形式のインデックスそのものに基づく CPU 向けのものであり，文献[9]で提案した GPU 向けの順序にはなっていない．

一方，計算プラットフォームとアプリケーションごとに非零要素の配置の特徴を用いた専用の最適化方法を考案して，実装するアプローチを取る研究[]も存在するが，その開発にはセンスと労力を要する上，適用範囲が限られるという問題があった．高い絶対性能を，より汎用な枠組み上で，安定して実現できることが望ましい．

本研究では，プラットフォームとして提案拡張メモリを用いた GPU に，アプリケーションを疎行列ベクトル積に焦点を絞り，CRS 形式そのものだけでなく，GPU 向けのアクセス順序についても評価するとともに，より多くの疎行列に対して実験することで提案方式の行列形状依存性について評価を行う．

以下、本報告では第 2 章で適用対象とするアプリケーションである疎行列ベクトル積について述べる。第 3 章では現在の GPU のメモリシステムの課題を整理する。第 4

[†]株式会社 東芝
Toshiba corporation
[††]東京農工大学
Tokyo University of Agriculture and Technology
[†††]奈良女子大学
Nara women's university

章では近未来の GPU に関連するメモリシステムの動向について論じる。第 5 章では Gather 機能付き拡張メモリの基本アーキテクチャとそれに付随する実現技術について紹介する。第 6 章ではメモリシステム側のスループットのシミュレーションによる提案方式の行列形状依存性の評価を示す。第 7 章でその考察を述べ、第 8 章で関連研究を紹介したのち、第 9 章でまとめる。

2. 疎行列ベクトル積高速化の課題

疎行列ベクトル積は連立一次方程式求解や固有値計算の中で用いられ，実行時間の大半を占めるため，最も重要な HPC 計算カーネルの一つである．反復解法である CG 法などを包含するクリロフ部分空間法の中で大半の計算時間を消費する。疎行列はメモリ容量と計算量の節約のため、通常、非零要素のみを値とインデックスを格納する 1 次元配列の組として格納する。疎行列ベクトル積の処理の特徴は 1 次元インデックス配列を添字とした 1 次元配列の間接参照にあり、その結果として様々なプロセッサ上でメモリバンド幅律速に陥る。

解析対象とするアプリケーションによって疎行列の非零要素の配置（インデックス配列）が大きく異なるため、アプリケーションごとに異なる最適化技法を適用した実装を行うことが一般的であった。例えば、プラットフォームを Cell/B.E.に限定し、アプリケーションを QCD 計算に限定し、それらの組合せでアーキテクチャと行列形状の特徴を生かした最適化を行う研究もある。ただし、そのようなアプローチの開発にはセンスと労力を要する上，適用範囲が限られるという問題があった．

最適化技法の選択を自動チューニングの枠組みで、複数の事前に用意された最適化技法をライブラリに選択させる研究も行われている。ただし、準備したどの最適化技法もあまり高い性能を示さない場合は低い性能で我慢しなければならない。利用者サイドとしては、高い絶対性能を，より汎用な枠組み上で，安定して実現できることが望ましい．

近年は GPU のデバイスメモリバンド幅の高さに着目し、メモリバンド幅律速である疎行列ベクトル積の高速化を目指した研究が数多く行われている。構造格子系の離散化によって得られる疎行列のように GPU にフィットする疎行列の場合は高い性能が出るが、そうでない場合は大きな性能低下が起きるのが一般的であった。

3. 現在の GPU のメモリシステムの課題

3.1 統合アクセス成否による性能変動

GPU ではデバイスメモリへの複数のアクセス要求がまとめて実行される統合(Coalesced)アクセスが発生するようにプログラミングされないと、メモリバンド幅バウンドなアプリケーションの性能は大幅に低下する。グループ内アクセス間の順序、アラインメントのずれや、グループ内アクセス群が例えば「4 バイトアクセスの場合に 128 バイト以内」等の一定の領域内にないと Coalesced 転送が発生せず、実効バンド幅が大幅に低下する。GPU 側で世代が新しくなるごとに性能低下が発生する条件や性能低下率は改善されてきているが、現時点でも上記の範囲の条件は存在するので、デバイスメモリへの不連続アクセスとなる場合は大きな性能低下が起きる。

3.2 デバイスメモリ容量の少なさ

現時点ではキャッシュベース CPU を用いたシステムでは主記憶容量が 512GB 程度まで拡張できる道があるが、GPU のデバイスメモリは最新のものでも 6GB のものが出たばかりである。通信と演算をオーバーラップさせて十分に通信遅延を隠蔽できるアプリケーションの場合は GPU を並列で用いることでこの制約を回避できる。しかし、この程度のメモリ容量では並列 GPU にすると細粒度で不規則な通信が発生してしまう場合もあり、デバイスメモリ容量の限界を拡張可能にする方法が望まれる。

3.3 オンチップメモリ容量の少なさ

ISCA2010 で Intel(R)*が発表した研究[17]では、SpMV(疎行列ベクトル積)計算において GPU(Nvidia GTX280)は Intel(R) Core(TM) i7 と比較して 2.5 倍の演算あたりメモリバンド幅を消費した理由として、列インデックスが Intel(R)Core(TM) i7 は半分がキャッシュに載るのに対して、GPU ではオンチップメモリに十分に載り切らない点を指摘している。キャッシュや共有メモリなどの GPU 上のオンチップメモリに再利用性のあるデータを載せきれない場合はデバイスメモリアクセスが多くなるため性能低下の原因となる。

3.4 PCI express ボトルネック

GPU のデバイスメモリ容量では十分でないアプリケーションでは、ホスト上の主記憶とデバイスメモリ間の転送や、ホスト上の主記憶を経由したノード間通信が必要になる。この場合、PCI express のバンド幅またはそこに接続されたネットワークのバンド幅がボトルネックとなる。特にバースト長が小さなデータを PCI express やネットワークに通す場合の実効バンド幅は、バースト長が大きなデータよりも大幅に低下する。

4. 近未来の GPU に関連するメモリシステムの動向

本章では対象とする近未来の GPU における不連続アクセスに伴う課題の解決に関連し、考慮しておきたいメモリシステムの最近の動向について、2 つの観点から論じる。

4.1 GPU のキャッシュへの依存度増加

GPU の一般用途への応用である GPGPU の普及のトレンドにおいて、Nvidia 社の

* Intel 、Intel Core は、米国およびその他の国における Intel Corporation の商標です。

Fermi アーキテクチャのように GPU プログラミングへの負担軽減の観点からキャッシュへの依存度が高まってきている。特に不連続アクセスを主体とするアプリケーションでは、キャッシュのラインサイズでのアクセスはバンド幅やキャッシュ容量を浪費する。よって、キャッシュ容量が CPU より早く底をつく GPU においては、不連続アクセス対策は今後も重要性が増していくトレンドにあると考えられる。

4.2 三次元実装

三次元実装は現在 2 つの方向からメモリシステムの変革に採用されようとしている。1 つは CPU チップの上にメモリチップを複数積み重ねていく方向での採用である。この方向は GPU 内のキャッシュ容量を増加させる目的に応用される可能性があると考えられる。ハイエンド GPU はチップあたりの発熱量が CPU より大きいため，放熱の観点からも CPU 以上のキャッシュ用メモリを積層するのは困難である．GPU のキャッシュの容量には CPU 以上に限度があり，疎行列ベクトル積の列ベクトルを全部載せきれないアプリケーションには，効果は限定的になると考えられる．

もう 1 つの方向はメモリパッケージ内でメモリチップを複数積み重ねていく方向での採用である。DDR4 は 4Gbps 程度の高周波を実現するためコントローラとメモリパッケージ間を Point-to-point 接続する．同一パッケージ内のチップ間をシリコン貫通ビアで接続することで反射の影響を回避し、高周波への対応と容量の増加を両立する。これは CPU の HPC に対する適性を向上させるように作用する．

一方，GPU に用いられている GDDR 系のメモリはコントローラとメモリ間を Point-to-point 接続することで DDR3 などの汎用メモリとの周波数的な差別化をはかっている．このため，DDR4 との差別化を考慮するとゲーム等の GPU 本来の用途で大幅な大容量化のニーズが出てこないかぎり，GDDR 系メモリの将来製品にはパッケージ内の三次元実装は採用されないと考えられる。このように GPU のデバイスメモリのバンド幅と容量にはトレードオフがあり，バンド幅重視設計の GPU には GDDR5，容量重視の GPU には DDR4 が用いられていくと考えられる．疎行列ベクトル積を多用する HPC 用途においては高バンド幅と大容量を同時に要求するため，三次元実装技術は HPC 用途における GPU（主に GDDR を用いる）の CPU（DDR4 を用いる）に対する優位性を現在以上に向上する方向には作用しないと考えられる．

5. Gather 機能付き拡張メモリ

5.1 基本アーキテクチャ

前章での課題の解決策として，DIMMnet-2 と同様の連続化ハードウェア（分散/収集機構）を COTS プロセッサのコアから見て内部ネットワークよりメモリに近い場所に追加することを提案する．提案方式の基本コンセプトを図 1 に示す．

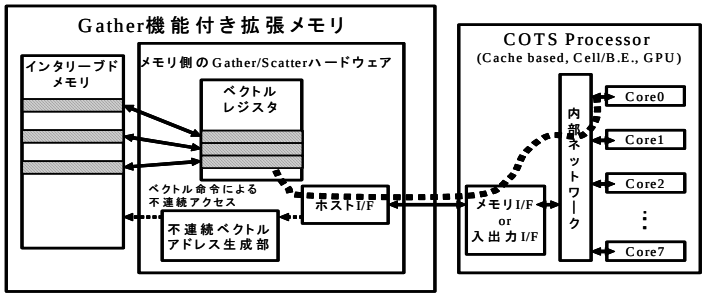

図 1　　提案する基本アーキテクチャ

表 1 に DIMMnet-2 の主なベクトル型アクセスコマンドを示す．このうち，等間隔ロード/ストア，リストロード/ストアのコマンドが不連続アクセスの連続化を実行するものである．ロード系が外部メモリから一種のベクトルレジスタである Prefetch Window への収集(Gather)処理を行い，ストア系が一種のベクトルレジスタである Write Window からの外部メモリへの分散(Scatter)処理を行なう．

表 1　　DIMMnet-2 の主なベクトル型アクセスコマンド

Load	Burst	VL
	Strided	VLS
	Indexed	VLI
Store	Burst	VS
	Strided	VSS
	Indexed	VSI

5.2 不連続アクセス向けメモリ構成

上記の基本アーキテクチャに基づく Scatter/Gather を効率的に行うには大容量メモリに対する不連続アクセスのスループットが高くなるようにする必要がある。本節では不連続アクセスのスループットを向上させる際の 4 つのポリシーを列挙する。詳細は文献[]で述べているので参照されたい。なお、後述する性能評価では DDR3 型の現時点で最も一般的に使用されている DRAM を用いており、以下の最後の項目のみ本評価の対象外である。

- 狭いビット幅のチャネル
- 深いインタリーブ

- Open ページポリシー
- 低サイクルタイム型メモリの活用

5.3 ホストインタフェース

本節では GPU と接続する場合の提案拡張メモリのホストインタフェースの候補について述べる。

5.3.1 PCI express

PCI express は現在もっとも一般的に用いられている標準 I/O インタフェースである。レーン数は 16 のものが主に GPU 向けに、8 のものがネットワークインタフェースなどのその他のサーバーの I/O 製品向けに使われる。GPU と拡張メモリの接続を行う場合は PCI express を利用するのが適切と考えられる。ただし、現時点で得られる最高速のバンド幅が x16 Gen.2 で 8GB/s に留まる。よって、この 2 倍のバンド幅を実現するには Gen.3 の普及を待つ必要がある。

5.3.2 GDDR5

GPU ベンダー側での対応が必要になるものの GDDR5 デバイスメモリインタフェースの利用が考えられる．GDDR5 DRAM は 1 チップあたり現段階で 7Gbps ×32 ビット幅で 28GB/s が得られるので，NVIDIA® Tesla™ C2050/2070 が 384 ビット分(32 ビット×12)に設置していた GDDR5 ポートを将来の GPU 上で機能メモリ用に 1～2 チップ分増設または切換え可能に改良することで PCI express より高いバンド幅を機能メモリ側に提供することが可能であると考えられる．理論上，GDDR5 はさらに 4 倍の 28Gbps まで向上できるとされており，本用途への応用は有望と思われる．

5.3.3 専用インタフェース

GPU はこれまでも Nvidia 社の SLI などのように独自の拡張用インタフェースを搭載してきた。特に Nvidia 社の Tesla 系列の製品群には ECC の採用や倍精度浮動小数演算の強化など HPC にフォーカスした改良を積極的に取り入れた歴史がある。よって、HPC 用途での重要性が認識されれば、機能メモリの装着や GPU 間通信を目的とした高バンド幅な専用インタフェースが HPC 系列の GPU に搭載される可能性もあると考えられる。

6. 性能評価

6.1 評価方法

6.1.1 シミュレータ

(1) ベースとして用いたシミュレータ

本研究の性能評価に際して、Maryland 大学の DRAMsim2[15]をシミュレータのベースとして用いた。DRAMsim2 はアドレストレースファイルを入力として動作する。旧バージョンの DRAMsim[13][14]はシミュレータ内で CPU とメモリシステムが一体になっている。評価対象は提案拡張メモリを装着する相手によってホスト CPU は異なる上、提案拡張メモリ内にあるベクトル型のアドレス生成部と DRAMsim で用意された CPU は、スループットが異なると思われる。さらに、メモリシステム構成や、メモリ種類の追加変更のしやすさも考慮して、本研究では DRAMsim2 を選択した。

(2) 改造内容

現在、DRAMsim2 は開発途上にあり、本研究においてはいくつかの不足分を独自に追加改造して用いた。その改造内容を以下に示す。

1) チャネル数を可変にした
2) アドレスマッピングをインタリーブに対応させた
3) トランザクション投入部多重度を可変にした

6.1.2 評価対象のメモリシステム

(1) DRAM チップ

評価に用いた DRAM チップのパラメータは DRAMsim2 に添付されている DDR3_micron_64M_8B_x4_sg15.ini の二種類である。その主なパラメータ値を表 2 に示す。

表 2　評価に用いた DRAM チップの主なパラメータ

DRAM 種類	DDR3
容量	2Gbit
バンク数	8
行数	32768
列数	2048
tCK(転送サイクルタイム)	1.5ns
CL(CAS レイテンシ)	10
BL(バースト長)	8
tRAS(RAS レイテンシ)	24
tRCD(RAS to CAS レイテンシ)	10
tCCD(CAS to CAS レイテンシ)	4

(2)システム構成

評価したメモリシステムのシステム構成パラメータを表 3 に示す。

表 3　評価したメモリシステムのシステム構成パラメータ

システム構成パラメータ	値
チャネル数	16
1 サイクルで発生する Transaction 数	8
チャネルあたりのビット幅	8
ランク数	4

チャネル本数は 16 に固定した。データラインが 8bit 幅のチャネルを 16 本実装するコントローラは全体として 128bit 分のデータ用ピンを消費する。このピン数は DIMMnet-3 の半分、DIMMnet-2 や現在市販されている SMB の仕様と同等である。

(3) アドレスビットマッピング

インタリーブ構成のアドレスマッピングはアドレスの下位から固定バースト分、チャネル、バンク、ランクの順に割り当てた。これにより、固定バースト長単位でチャネルが切り替わり複数チャネルの並列動作を促進した。ランクの切り替えには 1 サイクル待ちが入るので、一番上位に割り当て、ペナルティの発生頻度を抑制している。

6.1.3 ワークロード

(1) ランダムアクセス

変動確認の基準としてランダムアクセスの際のバンド幅を測定する。ランダムアクセスのアドレストレースを作成した。データサイズは 8 バイト、アドレスの範囲は 0～4G で 4096 個の乱数を生成し、トレースを作成した。

(2) 疎行列ベクトル積のベクトルへのアクセス

疎行列ベクトル積において提案拡張メモリにオフロードすることを想定し、その際のベクトルへの間接アクセスのトレースを University of Florida Sparse Matrix Collection[16]から比較的小規模な疎行列によって作成した。上記コレクションの拡張子 mtx のファイルの index 部分を index としてデータサイズ 8 バイトとして 0 番地から配置される配列をアクセスする際のアドレストレースを DRAMsim が受け付ける形式で生成した。本評価に使用した行列を表に示す。Na5 の場合でも非零要素数が約 16 万あるので、等間隔アクセスやランダムアクセスの評価の 4096 回の場合より約 40 倍シミュレーション時間がかかる。その際のシミュレーション実行時間が現在の測定環境では 1 日程度かかる。シミュレーション時間は index 列の大きさにほぼ比例して増加する。さらに、index ファイルのサイズが大きくなりすぎると現行の DRAMsim2 は異常終了することがわかった。このため、今回の実験は行列のサイズはあまり大きくせず、文献[8][9]でも取り上げた行列の中から小さい順に 3 個と、文献[22]で GPU 上での疎行列ベクトル積の FLOPS 値が評価されている行列の中から小さい順に 10 個を選択し、評価を行った。表 4 に評価に用いた疎行列の特徴を示す。

また、今回の実験では GPU 向けの評価として、文献[8][9]にて提案した提案拡張メモリと GPU を組み合わせたシステム向けのアルゴリズムの前処理部分を適用したアドレストレースを用いた場合のバンド幅も測定した。なお、今回用いた前処理では折り畳み幅の個別調整は行っていないものを用いた。また、0 パディングが index ファイルには入っているが、値は 0 に固定されるためメモリアクセスを行わなくてもコントローラ内部のレジスタまたはキャッシュなどで代用できるため、0 パディングに対応するアクセスはトレースファイルから省略されている。

表 4　評価に用いた疎行列の特徴

行列名	行数	非零要素数			
		合計	行平均	行最大	標準偏差
msc01440	1440	23855	16	40	12.3
bcsstk13	2003	42943	21	84	14.68
fv1	9604	47434	4	5	2.92
nasa4704	4704	54730	7	20	4.28
bcsstk15	3948	60882	15	39	6.83
aft01	8205	66886	8	11	2.56
Dubcova1	16129	134569	8	12	3.57
s2rmq4m1	5489	143300	26	30	5.04
bcsstk16	4884	147631	30	42	9.66
Na5	5832	155731	26	185	35.71
msc10848	10848	620313	57	300	49.4
exdata_1	6001	1137751	189	1501	390.27

6.2 結果

図 6 は格納方式を変えた場合の 8 バイトデータの疎行列ベクトル積実行時のベクトルアクセスに対する DDR3 の実効バンド幅を示したものである。図 7 は疎行列ベクトル積実行時のベクトルアクセスの実効バンド幅比較を示した図である。比較のためにランダムアクセスの際のバンド幅や文献[24]の Cevahir の結果の等価バンド幅も併記している。一部の例外を除き、ランダムアクセスのバンド幅より 10～20%程度高い実効バンド幅を観測した。一部の例外はランダムアクセスのバンド幅より GPU 向けに生成した index アクセスのほうが若干低い実効バンド幅を示しているが、ランダムアクセスより 10%以下の低下に留まっている。

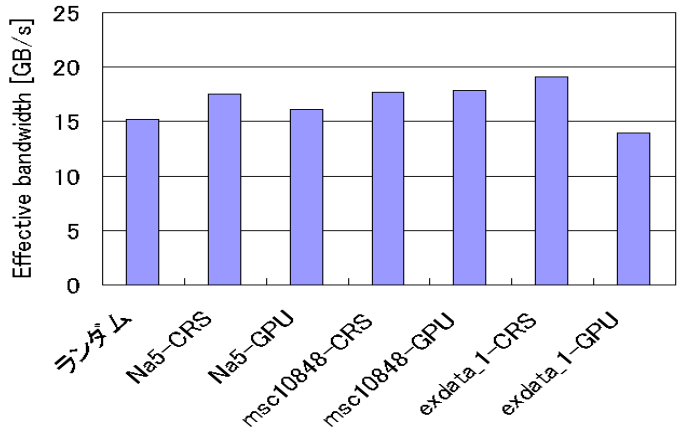

図 6　格納方式ごとの疎行列ベクトル積実行時のベクトルアクセスの実効バンド幅

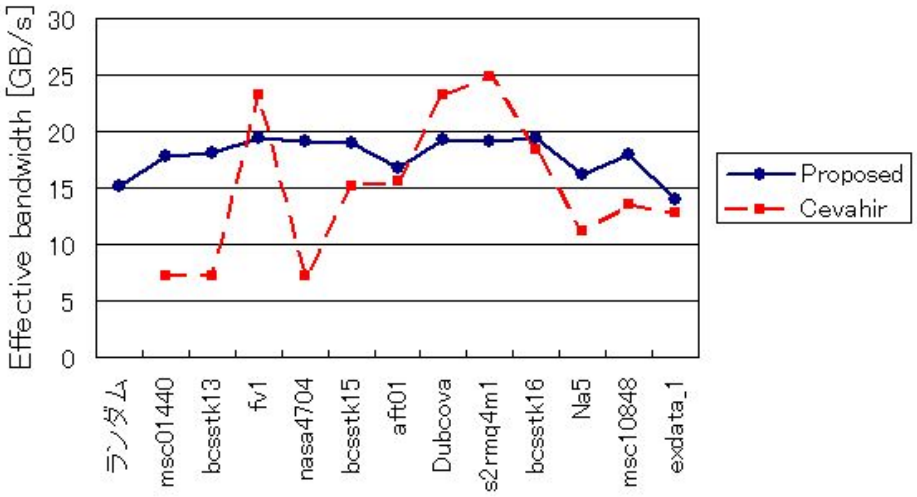

図 7　疎行列ベクトル積実行時のベクトルアクセスの実効バンド幅変動の比較 (DDR3)

7. 考察

7.1 実効バンド幅の行列形状依存性

まず、提案システム上での処理速度はどこがボトルネックになるのかによって決まる。具体的には(1)GPU の演算速度、(2)デバイスメモリアクセスバンド幅、(3)提案拡張メモリアクセスバンド幅、(4)提案拡張メモリからデバイスメモリへの転送路のバンド幅、の 4 つがボトルネックになる可能性がある。(1)の GPU 演算速度は現状の製品は十分に高速であり、行列値一次元配列は元からコアレスドアクセスで、提案拡張メモリで整列後は列ベクトルコアレスドアクセスになるので(2)のデバイスメモリアクセスバンド幅も十分高速である。さらに、(4)の提案拡張メモリからデバイスメモリへの転送路のバンド幅は GDDR5 や GPU の専用インタフェースを仮定すれば十分に大きくできるので、ここでは(3)の提案拡張メモリアクセスバンド幅がボトルネックになって性能が決定される状態を考える。

提案方式のバンド幅は平均 16.5GB/s、標準偏差 1GB/s 以下である。 (3)が性能を決めるモードでは 8 バイトを読み出して 2 浮動小数演算を行うことになるので、4 B/FLOPS である．これを元に文献[24]の Cevahir ら研究における GFLOPS 値から換算したキャッシュを用いた場合の列ベクトルの実効バンド幅を図 7 に併記した。提案方式はこれより平均が高く、ばらつきは大幅に小さい。つまり従来研究と比べて絶対性能で高く、安定した性能を実現できている傾向がわかる。

7.2 実効バンド幅の行列格納方法依存性

今回実験した範囲では CRS 形式のほうがバンド幅が高くなる行列と、GPU 向け前処理をした格納法のほうがバンド幅が高くなる行列があることが観測されている。exdata_1 のように CRS 形式のほうが連続アクセスが多く、メモリシステムとの相性が良い場合はランダムアクセスより向上する率が他の行列より高い。よって、これを並び替えると不連続になったりバンクコンフリクトが多くなったりして顕著に悪くなっていると考えられる。ただし、性能低下が発生したとしてもランダムのバンド幅より若干落ちる程度であり、GPU のみを用いた場合で格納法を変更した場合に数倍の性能差が発生することと比べると十分に安定しているといえる。

その他の行列のように元がランダムに近く、ランダムと比べてさほど相性が良いわけではない場合は、偶然バンクコンフリクトが増減して若干変動している程度のわずかな変動と考えられる。

8. 関連研究

問題を解決するために筆者らは先行研究[1]-[10]で Scatter/Gather 機能を有する拡張メモリシステムを提案した。文献[8] [9] [10]では疎行列ベクトル積においても評価を行ない、有効性を示してきた。ただし，文献[8] [9]はメモリシステムのスループット

が十分であるという過程の元での評価であり、文献[10]についてのメモリシステムを含む詳細なシミュレーションを行っているものの，従来方式で低い性能であった 1 種類の疎行列でしか評価を行っていなかった．このため，他の疎行列でも安定してスループットが得られるかどうかは明らかになっていなかった．また，疎行列の格納形式は最も広く用いられている CRS 形式のみで，アクセス順序は CRS 形式のインデックスに基づく CPU 向けのものであり，本報告のように文献[9]で提案した GPU 向けの順序にはなっていない．

一方，計算プラットフォームとアプリケーションごとに非零要素の配置の特徴を用いた専用の最適化方法を考案して，実装するアプローチを取る研究[16]も存在する。しかし，そのような開発にはセンスと労力を要する上，適用範囲が限られるという問題があった．

自動チューニングの枠組みで、最適化技法として通信方法[18]や行列格納法や負荷分散方法[8][9][19][20]で複数の事前に用意された最適化技法をライブラリに選択させる研究も行われている。特に、最近では GPU 上に自動チューニングを適用する研究[9][20]も行われている。ただし、準備したどの最適化技法を選ばせるタイプの自動チューニング[20]ではあまり高い性能を示さない場合は低い性能で我慢しなければならない。利用者サイドとしては、高い絶対性能を，より汎用な枠組み上で，安定して実現できることが望ましい. 文献[8][9]の自動チューニングは複数手法から選択するのではなく、1 行あたりの非零要素数の平均と標準偏差の 2 つの行列形状指標に乗ずるパラメータを計算面積(パディングを含めた計算量)を考慮しつつ調整する。これは汎用的な一つの手法のパラメータを調整する枠組みの一つと言える。なお、本研究で用いたアドレストレースは文献[8][9]のアルゴリズムが出力したトレースを用いている。

また従来行われている CPU や GPU 上での疎行列ベクトル積高速化の研究は、筆者らの研究を除き、ほとんどが列ベクトルはキャッシュ上に載ることが多い状態での性能評価となっており、今後計算対象が大きくなってきた場合の性能低下の問題解決には取り組んでいない。本研究は行列サイズや並列ノード数のスケーラビリティ（細粒度ランダム通信の排除）と、GPU のキャッシュメモリ容量の限界性を重視してキャッシュには頼らない実装になっている。

9. おわりに

GPU のメモリシステムにおける不連続アクセスにおける問題を解決するために筆者らは先行研究で Scatter/Gather 機能を有する拡張メモリシステムを提案した。その中で疎行列ベクトル積でも評価を行ない，提案拡張メモリシステム側の疎行列ベクトル積に対するスループットを PCI express のバンド幅以上にできるということを示した。しかし，疎行列ベクトル積の評価に限ると評価対象の行列が 1 つのみであった。

本報告ではその種類を増やした実験によって，提案方式の性能の行列形状依存性について評価した．その結果，提案方式は行列の形状に性能が鈍感で，高い実行性能を安定して提供できることがわかった。

今後の課題としては、より大きな行列群での評価や、今回評価に用いた DDR2 や DDR3 より高性能が期待できる XDR-DRAM、DDR4 DRAM、MRAM 等のメモリを用いた場合の評価がある。さらに PCI express 上に GPU と機能メモリが同時に装着された場合に相互の連携を行うための基本ソフトウェアの整備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．

参考文献

1) N. Tanabe, M. Nakatake, H. Hakozaki, Y. Dohi, H. Nakajo, H. Amano : "A New Memory Module for COTS-Based Personal Supercomputing", 7th International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA2004), pp.40-48 Jan. 2004
2) N. Tanabe, H. Nakajo : " An Enhancer of Memory and Network for Cluster and Its Applications", IEEE PDCAT'08, pp.99-106, Dec. 2008.
3) N. Tanabe, H. Nakajo : " High Performance Computing and Database Processing with COTS and Extended Memory Modules", HPC Asia2009 (Best paper award), Mar. 2009.
4) 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", International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'09) , pp. 427-433, Jul. 2009.
5) N. Tanabe, H. Hakozaki, Y. Dohi, Z. Luo, H. Nakajo : " An enhancer of memory and network for applications with large-capacity data and non-continuous data accessing", The Journal of Supercomputing, Vol. 51, No. 3, pp. 279-309, Mar. 2010.
6) N. Tanabe, T. Tsukamoto, A. Ohta, H. Nakajo : " Efficiency Improvement for Discontinuous Accesses of Cell/B.E. Using Hardwired Scatter/Gather on Memory-side ", IEEE ICCEE'10, Nov. 2010
7) 塚本，田邊，大田，中條 :“ベクトルアクセス機構を有するメモリモジュールによる不連続な DMA の効率化”，情報処理学会 HPC 研究会, Mar. 2010.
8) N. Tanabe, Y. Ogawa, M. Takata, K. Joe : " Scaleable Sparse Matrix-Vector Multiplication with Functional Memory and GPUs", Euromicro PDP’2011, Feb.2011
9) 小川，田邊，高田，城 :“機能メモリと GPU の PCI express 接続によるヘテロ環境における超大規模疎行列ベクトル積の性能予測”，情報処理学会 HPC 研究会 Vol.2010-HPC-126 No.20, Aug. 2010.
10) 田邊, Nuttapon，中條 :“Gather 機能付き拡張メモリのアクセス性能の評価”，情報処理学会 HPC 研究会, Vol.2010-HPC-128, Dec. 2010.
11) D. Wang, B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel, B. Jacob : "DRAMsim: a memory

system simulator", SIGARCH Computer Architecture News Vol.33, No.4, pp.100-107, Sep.2005
12) B. Jacob : "DRAMsim: A Detailed Memory-System Simulation Framework", http://www.ece.umd.edu/dramsim/v1/
13) B. Jacob : "DRAMSim2", http://www.ece.umd.edu/dramsim/
14) Tim Davis : " The University of Florida Sparse Matrix Collection", http://www.cise.ufl.edu/research/sparse/matrices/
15) Victor W. Lee et al. : "Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU", ISCA 2010
16) 木原, 多田野, 櫻井 :"精度混合型 Krylov 部分空間反復法における疎行列ベクトル積の Cell BE 上での実装と性能評価", 情報処理学会論文誌・コンピューティングシステム(ACS) Vol.1 No.1, pp. 51-60 Jun. 2008.
17) 西田 :"複数アーキテクチャ上での疎行列ベクトル積の性能最適化手法", 情報処理学会 HPC 研究会 Vol.2009-HPC-123 No.7, Nov. 2009.
18) 工藤,黒田,片桐,金田 :"並列疎行列ベクトル積における最適なアルゴリズム選択の効果", 情報処理学会 HPC 研究会 Vol.2002 No.22, 2001-HPC-089 pp.151-156, Mar. 2002.
19) 櫻井,直野,片桐,中島,黒田,猪貝 :"自動チューニングインターフェース OpenATLib における疎行列ベクトル積アルゴリズム", 情報処理学会 HPC 研究会 Vol.2010-HPC-125 No.2, Jun. 2010.
20) 久保田,高橋 :"GPU における格納形式自動選択による疎行列ベクトル積の高速化", 情報処理学会 HPC 研究会 Vol.2010-HPC-128 No.19, Dec. 2010.
21) N. Bell, M. Garland : "Eficient Sparse Matrix-Vector Multiplication on CUDA", NVIDIA Technical Report NVR-2008-004, Dec. 2008
22) M. M. Baskaran, R. Bordawekar : "Optimizing Sparse Matrix-Vector Multiplication on GPUs", IBM Research Report, RC24704, Apr. 2009
23) A. Cevahir, A. Nukada, S. Matsuoka : "CG on GPU-enhanced Clusters", 情報処理学会 HPC 研究会 Vol.2009-HPC-123 No.15 Nov. 2009.
24) A. Cevahir, A. Nukada, S. Matsuoka : " An Efficient Conjugate Gradient Solver on Double Precision Multi-GPU Systems", 先進的計算基板システムシンポジウム SACSIS2009, pp.353-360, May 2009.