Vol.2013-HPC-138 No.30
2013/2/22

疎行列のキャッシュ適合性に基づく Graph500 ベンチマークの特性解析

田邊昇[†1]　冨森苑子[†2]　高田雅美[†2]　城和貴[†2]

Graph500 はビッグデータ解析や疎行列処理のベンチマークとも言われており，近年注目を集めている．本論文では Graph500 の疎行列や，その Randomize や先行研究の Vertex sorting の有無を制御した行列のキャッシュメモリへの適合性を，空間的局所性の観点から解析した．その結果，Graph500 の疎行列の空間的局所性は極めて低く，疎行列ベクトル積(SpMV)を行う場合にはベクトルのアクセスの際に，キャッシュラインには平均して 1 個程度しか有効なデータは載っていないことが判った．ランダマイズによりその傾向が強まる． Vertex sorting は空間的局所性にはあまり作用しなかった．上記行列の SpMV 実行時の GPU の L1 キャッシュヒット率は，Vertex sorting の効果が小さく，グラフサイズ(SCALE)が大きくなるほど低下した．データサイズの大きさと非零要素配置のランダム性のため，キャッシュアウェアなソフトウェア最適化が困難である．特に空間的局所性の低さに伴う問題は現時点では未開拓な性能改善要因である．ゆえに，Graph500 や Green Graph500 の競争で優位に立つには，1 桁程度のメモリトラフィック削減による高速化や省電力化が期待できるメモリ側の Gather 機能が有望である．

Characterization of Graph500 Benchmark Based on Suitability for Cache of Sparse Matrices

NOBORU TANABE[†1]　SONOKO TOMIMORI[†2]
MASAMI TAKATA[†2]　KAZUKI JOE[†2]

Graph500 is a benchmark suite for big data analysis and sparse matrix processing which receives attention in these years. The spatial locality of sparse matrices used for Graph500 and their variations which are controlled in adoption of randomization and vertex sorting which is a technique of preceding work are investigated. We show the spatial locality of sparse matrices used for Graph500 is very low and there is about 1 or a little more valid data on a cache line for the memory accesses issued by sparse matrix-vector multiplications (SpMV) in average. The tendency is increased by randomization. The effect of vertex sorting does not have big effects on spatial locality. Effect of vertex sorting on L1 hit rate of GPU on processing SpMV for the above matrices is small and is degraded when the size of graph (i.e. SCALE) becomes larger. It is very difficult to solve the problem by just software approach because of the huge size of sparse matrices and the randomness of their accesses to degrade the optimization against their cache awareness. Especially problems caused by low spatial locality are frontiers of improvement, now. Therefore, hardwired gather functions at memory side is promising for taking advantage in the Graph500 lists, which improve the processing speed due to reduction of memory traffic in an order of magnitude.

1. はじめに

スーパーコンピュータの性能評価指標としては永年 Top500[1]ベンチマーク(密行列の LU 分解)が用いられてきたが，その指標が疎行列の反復解法を主体とする多くの技術計算の実状からかけ離れているため，問題視されることが多くなってきた．特に，Top500 が近年のコンピュータ製造上のネックになっていない演算性能に偏った指標になっていることが問題である．その結果，グラフ処理に対する社会的ニーズの高まりとあいまって， Graph500[2]ベンチマークが Top500 を補う評価指標として登場してきた．Graph500 はビッグデータ解析や疎行列処理のベンチマークとも言われており，近年注目を集めている．

Graph500 は Top500 と同様に毎年 2 回のランキングが発表されるため，HPC コミュニティにおいても Graph500 ランキングの国際競争が激しさを増してきている．その中核部分である BFS（幅優先探索）ループの電力効率を競う Green Graph500[3]も開始されている．

歴史のある Top500 のワークロードについては様々な研究の蓄積がなされており，最適化手法は十分に確立されている．一方，歴史の浅い Graph500 のワークロードの解析の研究はまだ数少ない．従来は主に並列化や通信を中心にした最適化の研究がなされてきた．グラフ解析ワークロードと従来型のコンピュータとの不適合は様々な観点で存在するが，その一つとして，ランダムアクセスに伴うキャッシュメモリの効率低下問題がある．この問題は Gather 機能付きメモリシステム[4]-[10]を軸に田邊らが取り組んできた課題であるが，この観点からの Graph500 に対する研究はこれまであまりなされていない．

システム全体として大容量な主記憶を有する PC クラスタや超並列計算機の上に分散並列処理される場合，解析対象のグラフ構造はストレージではなく，半導体メモリ上に保持される．Graph500 も事実上そのような状態での性能を測定するベンチマークである．Graph500 における並列化や

†1 (株)東芝
Toshiba corporation
†2 奈良女子大学
Nara Women's University

通信の最適化手法が確立されると，半導体メモリ上のグラフ構造へのアクセスのソフトウェア的あるいはハードウェア的な最適化の優劣が差を生むようになると予想される．

そこで本研究では，Graph500 のワークロードとメモリアクセスの効率化の鍵を握る，キャッシュとの相性に関する調査を行なう．

本論文の構成は以下の通りである．第 2 章ではターゲットとなるアプリケーションである Graph500 と Green Graph500 について紹介する．第 3 章では上記においても問題となる不連続アクセスに伴うキャッシュの問題について述べる．第 4 章では上記の問題の度合いを定量的に表す疎行列の空間的局所性に関する指標について提案する．第 5 章では提案指標を基にした評価について述べる．第 6 章で関連研究を紹介し，第 7 章でまとめる．

2. Graph500 と Green Graph500

近年，社会的ニーズとしては Web，ソーシャルネットワークサービス(SNS)の爆発的普及に伴い, グラフ解析のニーズが高まってきている．さらに，嗜好分析やリコメンデーション，車・電話・機械についた GPS やスマートメータなどの大量かつ多様なセンサからの時系列データの解析に基づくリアルタイム制御など，ビッグデータ解析のニーズが高まりを見せている．それらの多くはグラフ解析にカテゴライズされる．

グラフ解析のワークロードの特徴は，例えばある web サイトがどの web サイトをリンクしているという参照関係のような，大規模で不規則的なネットワーク構造を反映した大規模なランダム疎行列の処理に帰着される．これは Irregular processing[9]にカテゴライズされる処理であり，従来型のコンピュータでは効率的に処理することが困難とされてきた．しかし，この種のアプリケーションを効率よく実行できることは将来のコンピュータシステムにおける重要な要求となってきている．

Graph500[2]ベンチマークは上記のようなグラフ処理の特徴への適合性を測るベンチマークであるとともに，Top500[1]を補う評価指標として 2010 年 6 月の ISC'10 にアナウンスされ，同年 11 月の SC10 から継続的に実施されてきた．毎年 2 回（ISC および SC において）ランキングが発表されるため，HPC コミュニティにおいても Graph500 ランキングの国際競争が激しさを増してきている．

Graph500 ベンチマークのワークロードは Level synchronized BFS(レベルごとに同期した幅優先探索)である．Level synchronized BFS はグラフの隣接行列と列ベクトル(CQ に相当)の疎行列ベクトル積(SpMV)によって NQ を求め，次のレベルではそれを CQ として同様の処理を繰り返す処理ととらえることができることが明らかになっている．[17] ここで CQ は Current Queue，NQ は Next Queue の略である．CQ の初期値となる非零要素は重複せずランダムに与えられる 64 個の節点が入る．レベルが進むごとにその個数は一旦増加後，探索終了した節点の増加によって減少して，完全に無くなったときが探索の終了を意味する．各節点の探索済か否かを記録する配列への読み書きが，ランダムな不連続アクセスとなる．

グラフの隣接行列はクロネッカーグラフ[11]の隣接行列であり，非零要素の配置にランダム性が強いとともに，行内の非零要素数の変動幅が大きい．このため，0 パディングによって行内の非零要素数を一定とみなしてソフトウェアパイプライニングを行う最適化[12]はうまく動作しない．

並列処理における最適化手法としては隣接行列の二次元分割によって通信相手を行方向の分割数や列方向の分割数に減少する方式を多数のノードを用いている実装では殆どのグループが用いているものと考えられる．

Graph500 の中核部分である BFS ループの電力効率を競う Green Graph500[3]も 2012 年 11 月より開始される．このベンチマークでは Graph500 の前処理や後処理を除いた中心部分の処理速度と消費電力のバランスが求められる．このため, 電力や逐次処理部分の最適化をおろそかにしたまま，単純に PC のノード数を増やしてスケールアウトさせるという手法では Graph500 以上に Green Graph500 では上位にランクされるのは難しいと考えられる．BlueGene/Q など，徹底的に電力あたりの処理性能の高さを追求したシステムが有利である．

3. 不連続アクセスとキャッシュ

本章では不連続アクセスによるキャッシュの問題と，筆者らが提案してきたメモリ側の Gather 機能によるこの問題の解決についても述べる．

3.1 不連続アクセスによるキャッシュの問題

キャッシュアーキテクチャでは間接参照などの不連続アクセス時にキャッシュラインの利用効率が低下する．例えば，Graph500 ベンチマークの参照実装における疎行列の保持方法には CSR 方式（CRS 方式と呼ばれることも多く，これと同等の方式）と CSC 方式の二つのデータ構造による実装が示されている．これらのデータ構造で格納された疎行列とベクトルの積を計算する際には，ベクトルを格納した配列に対するインデックス配列を引数とした間接参照が発生する．グラフ解析アプリケーションにおいては疎行列のランダム性が高い．このため，従来のキャッシュベースな CPU・GPU や従来のメモリを用いる場合は，上記の間接参照における図 1 に示すような問題の発生が顕著になると考えられる．

(1)インデックスはパッケージ間配線を通過するため，メモリバンド幅を消費するとともに，大きな電力も消費する．
(2)ライン単位の転送により，有効データが少ないゆえのパッケージ間配線バンド幅と電力の浪費が発生する．
(3)有効データが少ないゆえのキャッシュエントリの浪費

と，それに伴うリプレースの多発やヒット率の低下が発生する．

(4)有効データが少ないゆえの TLB エントリの浪費とヒット率の低下が発生する．

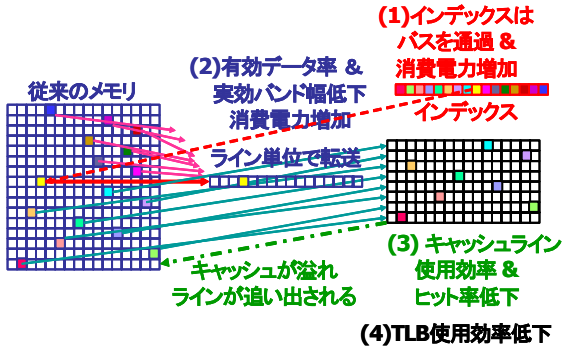

図 1　間接参照における従来システムの問題

Figure 1　Problems caused by indirect accesses on conventional systems.

3.2 ランダム性が高い場合の解決方法

ランダム性が高い大きなデータの際は，タイリングなどのキャッシュの挙動を十分に意識したソフトウェア的最適化が困難となり，キャッシュ容量内に有効なデータを置ききれず，問題解決が困難である．特にソフトウェアのみではライン内にごく少数の有効データしかないようなアクセスパターンでは，上記の(2)(3)の問題を解決できない．

一方，図 2 に示すように田邊らが提案してきた Gather 機能付 Hybrid Memory Cube[10]のようなメモリ側の Gather 機能により，(1)から(4)の問題を解決することが可能である．

Green Graph500 の観点からは，インデックスがパッケージ間配線を通過しない点と，Gather のための細粒度アクセスが積層メモリ間の短い配線で転送される点と，使用しないデータがパッケージ間配線を通過しない点が電力効率向上に貢献する．

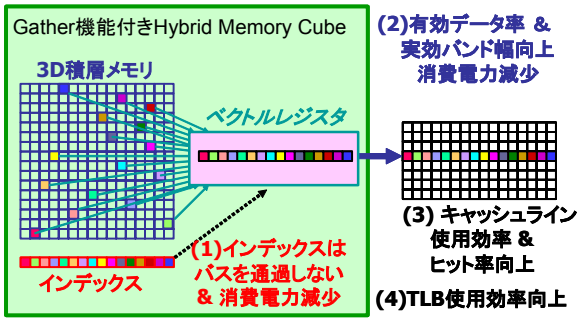

図 2　Gather 機能つき Hybrid Memory Cube による間接参照の問題の解決

Figure 2　Solution for problems caused by indirect accesses using Hybrid Memory Cube with hardwired gather.

4. 疎行列の空間的局所性に関する指標

疎行列のキャッシュへの適合性分類に資する疎行列の特性に関する新しい指標として「列インデックス列の空間的局所性」を提案する．図 3 にその概念図を示す．行列の特性値を表す場合の定義を以下に示す．「非零要素のみを CSR 形式で格納し，列ベクトル読み出しに用いるインデックス配列を先頭から順に読み出した際に，下位 5bit 以外が継続して一致している回数をカウントする．不一致が生じた時のカウント値を出力し，1 からカウントしなおす．その出力されたカウント値の数列の平均を取ったものを，列インデックス列の空間的局所性 $S_{locality}$ と定義する．」

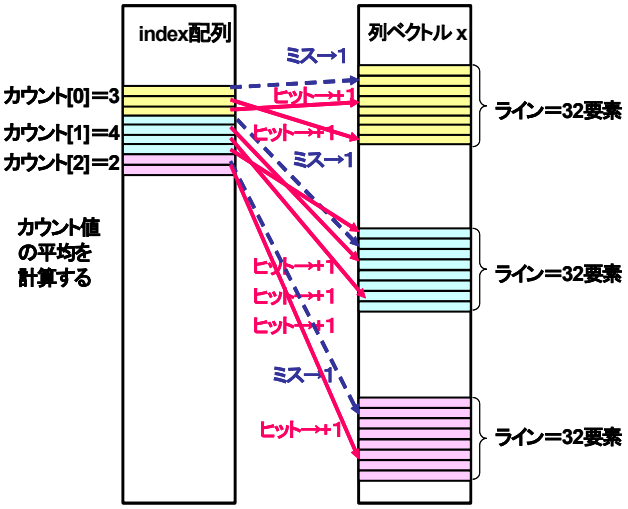

図 3　列インデックス列の空間的局所性 $S_{locality}$

Figure 3　Spatial locality of column index sequence $S_{locality}$.

上記の値は，1 本のラインしかないキャッシュに対して疎行列ベクトル積を CSR 形式の疎行列に対して行なう際の列ベクトルをアクセスする際のキャッシュラインあたりにいくつの有効データが存在するのかという平均値を与える．その逆数は上記のアクセスに対するメモリバンド幅が，どれ位薄まってしまうのかを意味する．ただし，実際のキャッシュは多数のラインを用意することで時間的局所性を引き出すので，疎行列とキャッシュの相互作用の一面しか見ていないことから，上記の平均値は実機特性を厳密に表すものではなく，近似値である．

なお、上記の定義で下位 5bit としているのは 128 バイト(GPU 等の一般的なキャッシュラインサイズ)のラインの中に存在する 32(2^5)個の 4 バイトデータのいずれかへのアクセスは，キャッシュがヒットすることに対応している．

5. 評価

本章では，グラフ解析ワークロードに対するメモリアクセス列のキャッシュメモリへの適合性を，空間的局所性の観点から評価を行う．評価環境を表 1 に示す．

表 1　評価環境

Table 1　Evaluation environment.

CPU	Intel®Xeon®*CPU　X5670 @ 2.93GHz
GPU	Nvidia Tesla C2050 (コア数 448)
デバイスメモリ	メモリバンド幅　144GB/s,3GB
ホスト I/F	PCI express x16 Gen.2 (最大バンド幅 8GB/s)
OS	RedHat Enterprise Linux Client release5.5
CUDA	Cuda 3.2
Octave	GNU　Octave, version 3.2.4
Matlab	Matlab version R2011b

5.1 Graph500 用疎行列による評価

Graph500 のグラフに対するメモリアクセス列のキャッシュメモリへの適合性を，空間的局所性の観点から評価を行った．Graph500 で扱う問題のサイズはグラフの頂点数＝2^{SCALE} であるような SCALE 値を用いて表す[3]．本評価において，ベンチマークを実行するプログラムと同様，グラフ生成において枝数が頂点数の 16 倍となるようなクロネッカーグラフを生成する．次に，生成された枝リストからグラフデータ構造に変換する．その際，生成された疎行列を表 2 に示す．SCALE は単一 GPU の測定環境でデバイスメモリに入りきる 20 まで測定した．本実験では疎行列の生成には Graph500 の Reference code2.1.4 の Matlab 互換の Octave 版の kronecker_generator.m および kernel_1.m を用いて作成した．kernel_1.m により生成された疎行列をテキストファイルに出力し，Matrix market 形式に変換後，自作プログラムに入力して評価を行った．

表 2　評価に用いた疎行列

Table 2　Experimented matrices

SCALE 値	*非零要素数*	*行数*
11	45,536	2,048
12	97,010	4,095
13	203,826	8,192
14	426,578	16,384
15	883,126	32,768
16	1,818,824	65,536
17	3,730,586	131,072
20	31,398,208	1,048,576

評価は 4 章に示した疎行列の空間的局所性を計測した．図 4 に SCALE が 11 から 20 までの Graph500 用疎行列の空間的局所性を示す．その結果は行列の行数が増加するにつれて単調減少し，Toy クラス(SCALE=26)の 1/64 である SCALE=20 においてすら，ライン内の有効データは平均 1.23 個に過ぎない．つまり，Graph500 用疎行列には空間的局所性はほとんど存在しないと言ってよいレベルにある．

疎行列ベクトル積性能と強い相関関係がある L1 ヒット率を CSR 形式で保持した疎行列ベクトル積を GPU Nvidia 2050 の上で実行し測定した結果を図 5 に示す．L1 ヒット率は行数に対して単調減少し，SCALE=20 における L1 ヒット率は 15.4%に過ぎない．このヒット率の中には間接参照されるベクトルだけではなく，行列値の配列への連続アクセスも含まれている．10%程度でヒット率の低下が止まるのは行列値の配列への連続アクセスがこの程度の比率を占めるためであると考えられる．ライン内の有効データは平均 1 個であり，キャッシュラインは通常 128 バイトである．このため，ベクトルのデータ型が 8 バイトである場合は，間接参照は有効なデータだけ Gather した場合の 16 倍のバンド幅を消費する．データ型が 4 バイトである場合は 32 倍のバンド幅を消費する．90%程度のメモリアクセスがこのような非常に効率の悪いメモリアクセスを行なうため，Gather 機能付 Hybrid Memory Cube[10]のようなメモリ側の Gather 機能の効果は甚大であると考えられる．

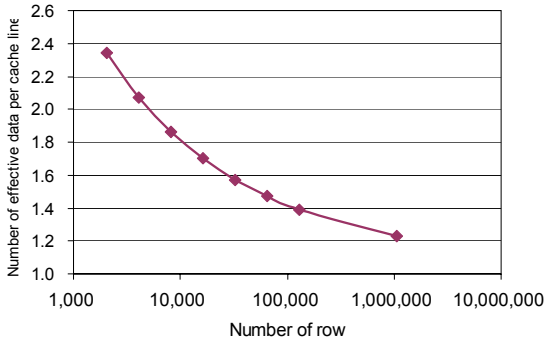

図 4　Graph500 用疎行列の列インデックス列の空間的局所性

Figure 4　Spatial locality of the sequence of column index of sparse matrix for Graph500.

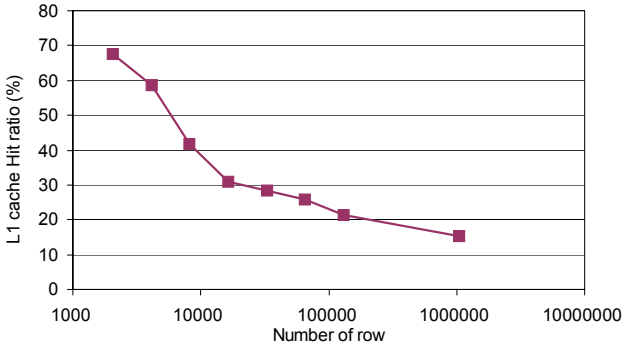

図 5　Graph500 用疎行列の疎行列ベクトル積実行時の GPU の L1 キャッシュヒット率(G2050)

Figure 5　L1 hit rate on executing SpMV for sparse matrix for Graph500 on GPU (G2050).

* Intel，Xeon は，米国およびその他の国における Intel Corporation の商標です．

5.2 Graph500 用疎行列におけるランダマイズ過程の評価

前節の実験により Graph500 用疎行列には空間的局所性はほとんど存在しないと言ってよいレベルにあることが判ったが，その性質は疎行列生成の際のどの過程によって生み出されたのかを評価するため，ランダマイズを実行している 2 箇所を全部、または部分的にコメントアウトし，空間的局所性を測定した．その結果を図 6 に示す．

空間的局所性は Permute vertex labels だけをコメントアウトした場合と Permute the edge list も両方ともコメントアウトした場合の差は出なかった．上記の双方の実験とも Graph500 用疎行列より若干空間的局所性が上がっている．つまり，Permute vertex labels が空間的局所性を低下させる効果をもたらしたと考えられる．

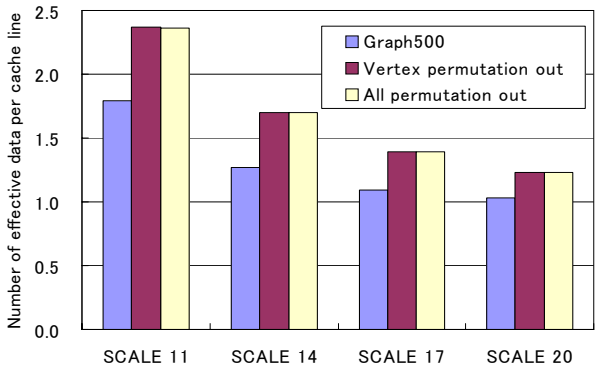

図 6　Permutation の Graph500 用疎行列の列インデックス列の空間的局所性への効果

Figure 6　Effect of permutation on spatial locality of the sequence of column index of sparse matrix for Graph500.

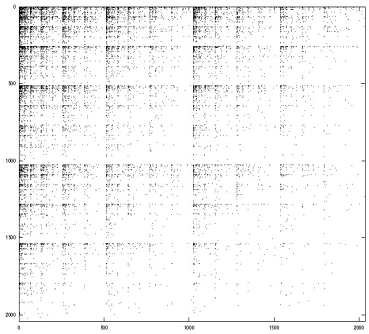

図 7　Permutation を外した Graph500 用疎行列の非零要素配置 (SCALE=11)

Figure 7　Mapping of non-0 elements of sparse matrix for Graph500 which is removed all permutation. (SCALE=11)

図 7 はランダマイズを実行している 2 箇所を全部コメントアウトした場合の SCALE11 における非零要素配置を示している．この図は Octave 上で spy 関数により容易に作成できる．この非零要素配置図から permutation をしない場合は周期的な模様の存在を感じ取ることができる．この規則性を利用した何らかの最適化が可能であるかもしれない．しかし，もしランダマイズを戻すのと同等な最適化手法が考案されたとしても，その空間的局所性改善の効果は SCALE20 において 20%程度に過ぎない．SCALE が大きくなるほど効果が少なくなる傾向が観測できている．

5.3 Graph500 用疎行列における Vertex sorting の評価

Graph500 用疎行列処理のキャッシュヒット率の改善に関する先行研究としては，上野[17]らが Vertex sorting というキャッシュに関する最適化を提案評価している．Vertex sorting は隣接行列の degree(行列においては行中の非零要素数に対応) が降順になるようにソーティングして節点番号を入替える前処理である．本節では Graph500 用疎行列を生成し，その後に colperm 関数を実行して非零要素数によって昇順の permutation ベクトルを生成後に，flipud 関数で降順に変換し，行と列をその permutation ベクトルによって入替えて Vertex sorting 前処理を施した疎行列を生成した．Octave 上では疎行列を Matrix market 形式に変換する mmwrite.m[13]が正常動作しないため Matlab 上で上記を行なった上で，mmwrite.m を用いて Matrix market 形式の疎行列を生成した．これを前述の測定に用いた空間的局所性測定プログラムや，CSR 形式用 SpMV プログラムの入力とした．このようにして測定した空間的局所性と GPU 上の L1 キャッシュヒット率をそれぞれ図 8 および図 9 に示す．

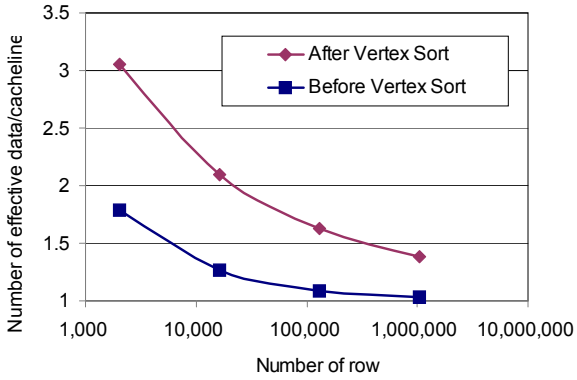

図 8　Vertex sorting の Graph500 用疎行列の列インデックス列の空間的局所性への効果

Figure 8　Effect of vertex sorting on spatial locality of the sequence of column index of sparse matrix for Graph500.

図 8 で明らかなように Vertex sorting はランダマイズ過程が施された後の Graph500 用疎行列の空間的局所性には SCALE が大きくなるほど変化を与えない．空間的局所性の値自体も SCALE20 で 1.4 程度にすぎない．つまり，Vertex sorting は Gather 機能付きメモリ[4]-[10] と同様にランダムメモリアクセスの問題を軽減することを目的とする．後者は行列サイズによらずに空間的局所性をハードウェアが強制的に 30 程度(キャッシュライン 128B，データ型 32bit の場合)に高める．その伸び代は Vertex sorting によってはほぼ未開拓であると言える．

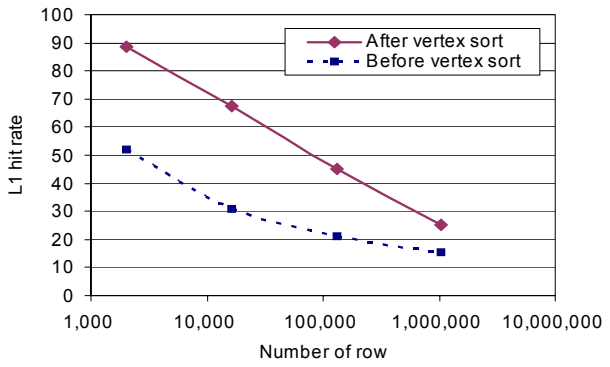

図 9　Vertex sorting の Graph500 用疎行列の疎行列ベクトル積実行時の GPU の L1 キャッシュヒット率への効果 (G2050)

Figure 9　Effect of vertex sorting on L1 hit rate on executing SpMV for sparse matrix for Graph500 on GPU (G2050).

一方，図 9 で明らかなように L1 キャッシュヒット率は小さい行列ほど大きな性能向上を示すものの，SCALE＝20 ではその効果はそれほど大きくない．Vertex sorting は比較的密な行には時間的局所性を高めるのに有益だが，疎な行にはあまり効果が期待できないことが原因と考えられる．また図 9 から行列サイズが大きくなるほどその効果が薄くなる傾向が見て取れる． Graph500 用疎行列ではサイズが 8 倍になっても一行あたりの非零要素数は 1 個程度しか増えないため，密な行の比率が低下する．よって，その傾向に応じて効果もサイズが大きくなるにつれて薄まると考えられる．

これに対して，キャッシュの代わりに Gather 機能付きメモリ[4]-[10] を用いる場合は，原理的に行列形状やサイズによらずに安定[9]した効果が期待できる．特に等間隔なアクセスと異なり，グラフ処理に顕著なランダムな非零要素配置は Gather 機能付きメモリにおけるバンクコンフリクトの問題が少なく，性能が安定することが先行研究[8]によってわかっている．よって，Vertex sorting と比較して加速率の差は行列サイズが大きくなるほど大きくなると考えられる．

6. 関連研究

Suzumura[18]らは Graph500 課題のリファレンス実装の解析を様々な観点から行っている．この中ではキャッシュに関する解析は行なわれていない．上野[17]らはこれを踏まえて，Graph500 課題の隣接行列の 2 次元分割の実装について報告している．Vertex sorting というキャッシュに関する最適化を提案評価しているが効果は 10%程度に過ぎない．これは時間的局所性に関する改善であり，空間的局所性の改善に着目したものではない．

Graph500 が代表するグラフ処理の高速化にカテゴライズされる研究としては PageRank の高速化に関する研究がある．CPU ベースの PC クラスタによるものは Rungsawang [14]などの研究，GPU によるものは Yang[15]や Cevahir[16]などの研究がある．疎行列ベクトル積はメモリバンド幅で性能が決まるため，GPU の実装は CPU よりも GPU のメモリバンド幅が大きいことによる加速が得られる．これらにおいても，キャッシュに関する解析はあまり行なわれていない．

疎行列ベクトル積処理で表せるグラフ処理を Hadoop により実装した PEGASUS[19]や GBASE[20]も発表されている．これらは多数ノードでスケールさせることに主眼があり，キャッシュに関する解析は行なわれていない．

7. おわりに

Graph500 ベンチマーク課題に用いられているクロネッカーグラフの疎行列と，その Randomize や Vertex sorting の有無を制御した行列を用い，Graph500 ベンチマークに対するメモリアクセス列のキャッシュメモリへの適合性を，空間的局所性の観点から解析した．

その結果，Graph500 ベンチマーク課題のクロネッカーグラフの疎行列の空間的局所性は極めて低く，疎行列ベクトル積を行う場合にはベクトルのアクセスではキャッシュラインには平均して 1 個程度しか有効なデータは載っていないメモリアクセスを繰り返すことになることが判った．これより，この問題をメモリ側の Gather 機能によって解消するならば，1 桁程度の計算時間短縮が期待できることが判った．

これらは Graph500 の競争においてソフトウェア的には拾い出せなかった未開拓な性能や省電力の伸び代であり，これを取り込んだものが将来の Graph500 およびその消費電力あたりの性能を競う Green Graph500 における勝者となる可能性が高いと考えられる．

また，Graph500 には本来いくつかの permutation がランダマイズの過程として組込まれているが，これを外すと空間的局所性が若干増えるとともに，人工的行列特有の規則性が観察できることがわかった．これを用いることができたとしてもその効果は高々20%程度に過ぎないことがわかった．

先行研究である Vertex sorting は性能改善率やそのスケーラビリティが，Gather 機能付きメモリが空間的局所性を改善する場合に期待できる劇的な改善に比べ低かった．Vertex sorting は Graph500 用疎行列の空間的局所性にあまり大きな変化を与えなかった．つまり，空間的局所性の改善は，Vertex sorting においても未開拓な領域と言える．アルゴリズム的な Graph500 の最適化技法が成熟した際に，同程度あるいは多少小さめの規模のマシンを用いて他国より抜きん出るために，将来のエクサスケールマシンには，Gather 機能付きメモリを採用することが有望であると考える．

今後の課題としては，GPU のデバイスメモリ容量より大きな SCALE に対する評価，メモリ側の Gather 機能を有する環境での Graph500 の実装と，それに基づく TEPS 値における加速率の評価，電力モデルや性能モデルの構築，Vertex sorting 以外の最適化技法の調査と Gather 機能付きメモリとの整合性の検討が挙げられる．

謝辞 本研究の一部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) Top500 : http://www.top500.org/.
2) Graph500 : http://www.graph500.org/.
3) Green Graph500 : http://green.graph500.org/.
4) 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, pp.40-48 (2004).
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 (2010).
6) N. Tanabe, Y. Ogawa, M. Takata, K. Joe : "Scaleable Sparse Matrix-Vector Multiplication with Functional Memory and GPUs", Euromicro PDP2011 (2011).
7) N. Tanabe, B. Nuttapon, H. Nakajo, Y. Ogawa, J. Kogou, M. Takata, K. Joe : "A memory accelerator with gather functions for bandwidth-bound irregular applications", Proceedings of the first workshop on Irregular applications: architectures and algorithm (IAAA'11) in conjunction with SC11，pp.35-42 (2011).
8) 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 ARC 研究会, Vol.2010-ARC-192 (2010).
9) 田邊，Nuttapon，中條，小川，高田，城 : " 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"，情報処理学会 HPC 研究会, Vol.2011-HPC-129 (2011).
10) 田邊，堀，Nuttapon，中條 : "Gather 機能を有する Hybrid Memory Cube の FPGA を用いた予備評価"，情報処理学会 HPC 研究会, Vol.2012-HPC-133 (2012).
11) J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos, "Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication", in Conf. on Principles and Practice of Knowledge Discovery in Databases (2005).
12) 南，井上，堤，前田，長谷川，黒田，寺井，横川 :"「京」コンピュータにおける疎行列とベクトル積の性能チューニングと性能評価"，ハイパフォーマンスコンピューティングと計算科学シンポジウム 2012 (HPCS'12), pp.32-41 (2012).
13) National Institute of Standards and Technology : "Matrix Market I/O Functions for Matlab", http://math.nist.gov/MatrixMarket/mmio/matlab/mmiomatlab.html
14) A. Rungsawang, B. Manaskasemsak : "PageRank Computation Using PC Cluster", Euro PVM/MPI 2003, Recent Advances in Parallel Virtual Machine and Message Passing Interface, LNCS Vol.2840, pp.152-159 (2003).
15) X. Yang, S. Parthasarathy, P. Sadayappan : "Fast sparse matrix-vector multiplication on GPUs: implications for graph mining", Proc. VLDB Endowment, Vol.4, No.4, pp.231-242 (2011).
16) A. Cevahir, C. Aykanat, A. Turk, B. B. Cambazoglu, A. Nukada, S. Matsuoka : "Efficient PageRank on GPU Clusters", 情報処理学会 HPC 研究会 Vol. 2010-HPC-128, No.21, pp.1-6 (2009).
17) K. Ueno and T. Suzumura "Highly Scalable Graph Search for the Graph500 Benchmark", 21st International ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC'12), pp.149-160 (2012).
18) Toyotaro Suzumura, Koji Ueno, Hitoshi Sato, Katsuki Fujisawa and Satoshi Matsuoka, "Performance Evaluation of Graph500 on Large-Scale Distributed Environment", IEEE IISWC 2011 (2011).
19) U. Kang, C. E. Tsourakakis, and C. Faloutsos : "PEGASUS : A Peta-Scale Graph Mining System Implementation and Observations", 2009 Ninth IEEE International Conference on Data Mining (ICDM '09), pp.229-238 (2009).
20) U. Kang, H. Tong, J. Sun, C.Y. Lin, and C. Faloutsos : "GBASE: a scalable and general graph management system", 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11),pp.1091-1099 (2011).