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[†]　冨森 苑子[‡]　高田 雅美[‡]　城 和貴[‡]

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，それを変形した行列のキャッシュメモリへの適合性を，列インデックス列の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，疎行列ベクトル積時のベクトルのアクセスの際に，32 個のデータを持つ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明した．ランダマイズ過程を戻したり，Vertex sorting による 空間的局所性の向上は少なく，グラフが大きくなるほどその効果が減少する．ゆえに，Graph500 や Green Graph500 の競争で優位に立つには，ライン内部の空間的局所性を最大化することで，1 桁程度のメモリトラフィック削減が期待できるメモリ側の Gather 機能の採用が有望である．

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

NOBORU TANABE[†], SONOKO TOMIMORI[‡], MASAMI TAKATA[‡], KAZUKI JOE[‡]

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 effects of canceling randomization and vertex sorting have small effects on spatial locality. These are degraded when the size of graph becomes larger. Therefore, gather functions at memory side is promising for taking advantage in the Graph500 and Green Graph500 lists, which reduces memory traffic in an order of magnitude by maximizing intra line spatial locality.

1. はじめに

Graph500[1]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[2]を補う評価指標という意味合いからも，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．

本研究では Graph500 における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ワークロード疎行列の形状に着目して行なう．

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 4 章では提案指標を基にした評価について述べる．第 5 章でまとめる．

2. Graph500 と Green Graph500

近年，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，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の効率的実行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[1]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[2]を補う評価指標として近年，継続的に実施されてきた．毎年 2 回ランキングが発表されるため，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．Suzumura らによってその参照実装の評価[4]が行われている．Level synchronized BFS はグラフの隣接行列と列ベクトル(CQ に相当)

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

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．[5] 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[6]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．

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るグループは皆が行なうと考えられる．

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．

3. 疎行列の空間的局所性に関する指標

筆者らは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指標として「列インデックス列の空間的局所性」を提案した[7]．図 1にその概念を 1 ラインに 32 要素が入る場合につき示す．

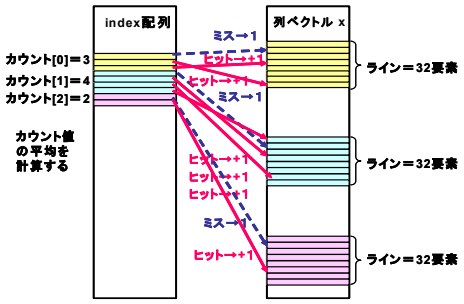

図 1　列インデックス列の空間的局所性

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と定義する．

4. 評価

本章では，Graph500 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

CPU	Intel® Xeon® CPU X5670 @ 2.93GHz
GPU	Nvidia Tesla C2050 (448 cores)
Device Memory	144GB/s, 3GB
Host I/F	PCI express x16 Gen.2 (Max. 8GB/s)
OS	RedHat Enterprise Linux Client release5.5
CUDA	Cuda 3.2
Octave	GNU Octave，version 3.2.4
Matlab	Matlab version R2011b

4.1 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．

表 2　評価に用いた疎行列

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

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．

図 2 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 1.03 個に過ぎない．つまり，Graph500 用疎行列にはライン内の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 3 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 5.6%に過

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

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．5%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．95%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[12]のようなメモリ側の Gather 機能[8]-[12]の効果は甚大であると考えられる．

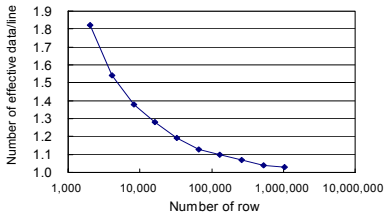

図 2　Graph500 用疎行列の列インデックス列の空間的局所性

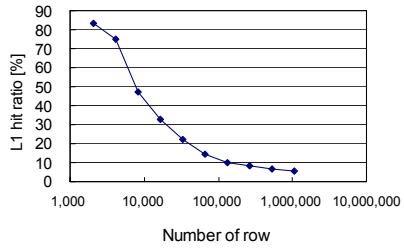

図 3　Graph500 用疎行列の疎行列ベクトル積実行時の GPU の L1 キャッシュヒット率(C2050)

4.2 Graph500 疎行列へのランダマイズ解消の効果

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．

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．

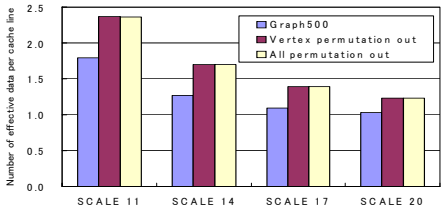

図 4　Permutation の Graph500 用疎行列の列インデックス列の空間的局所性への効果

4.3 Graph500 疎行列への Vertex sorting の効果

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，Ueno[5]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(行中の非零要素数に対応) が降順になるようソーティングして節点番号を入れ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 ベクトルにより入れ替えて 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 5および図 6に示す．

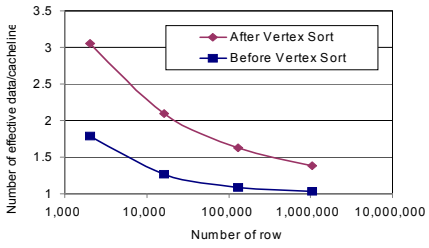

図 5　Vertex sorting の Graph500 用疎行列の列インデックス列の空間的局所性への効果

図 5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 が大きくなるほど改善度が鈍る．空間的局所性の値自体も SCALE20 で 1.4 程度と低い．つまり，Vertex sorting は Gather 機能付きメモリ[8]-[12]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．

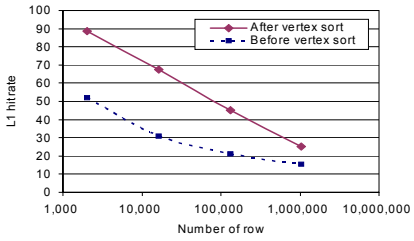

図 6　Vertex sorting の Graph500 用疎行列の疎行列ベクトル積実行時の GPU の L1 キャッシュヒット率への効果(C2050)

一方，図 6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 6 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[8]-[12]を用いる場合は，原理的に行列サイズによらずに安定した効果が期待できる．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．

5. 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およびや，その Randomize や Vertex sorting の有無を制御した行列を用い，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．ランダマイズ過程を戻したり，Vertex sorting による 空間的局所性の向上は少なく，グラフが大きくなるほどその効果が減少する．ゆえに，Graph500 や Green Graph500 の競争で優位に立つには，ライン内部の空間的局所性を最大化することで，1 桁程度のメモリトラフィック削減が期待できるメモリ側の Gather 機能の採用が有望である．決定版のアルゴリズムが流布した際に，このメモリを持つ者が持たない者に大きく差を付けられる可能性が示された．

より大きな SCALE に対する評価，マルチノード化や最新のアルゴリズムに対する評価，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は今後の課題である．

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] Graph500 : http://www.graph500.org/.

[2] Top500 : http://www.top500.org/.

[3] Green Graph500 : http://green.graph500.org/.

[4] Toyotaro Suzumura, Koji Ueno, Hitoshi Sato, Katsuki Fujisawa and Satoshi Matsuoka : "Performance Evaluation of Graph500 on Large-Scale Distributed Environment", IEEE IISWC 2011 (2011).

[5] 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).

[6] 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).

[7] 冨森, 田邊, 高田, 城 : "疎行列のキャッシュへの適合性分類に関する予備評価", 情報処理学会 HPC 研究会, Vol.2012-HPC-135 (2012).

[8] 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).

[9] 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).

[10] N. Tanabe, Y. Ogawa, M. Takata, K. Joe : "Scaleable Sparse Matrix-Vector Multiplication with Functional Memory and GPUs", Euromicro PDP2011, pp. 101-108 (2011).

[11] 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).

[12] 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).

[13] National Institute of Standards and Technology : "Matrix Market I/O Functions for Matlab", http://math.nist.gov/MatrixMarket/mmio/matlab/mmiomatlab.html