Vol.2014-HPC-143 No.7
2014/3/3

疎行列ベクトル積性能を決める諸要因

田邊 昇[†1]　冨森苑子[†2]　高田雅美[†2]　城 和貴[†2]

疎行列ベクトル積(SpMV)は多くの場合にキャッシュアーキテクチャとの相性が悪い．並列処理においては負荷不均衡が性能に与える影響も大きい．これまでは SpMV 性能を決める要因として，キャッシュのヒット率や一行あたりの非零要素数の平均，最大値，分散が注目されていた．しかし，それらと性能との相関が不明瞭であり，SpMV の挙動は長年にわたり謎に包まれていた．それは SpMV の最適化や，効率的な疎行列ライブラリ構築の障害であった．本報告では，SpMV 性能を左右する様々な要因をアプリケーション依存の要因とプラットフォーム依存の要因に分けて考察した．それを踏まえて行列の非零要素配置から導かれる時間的局所性と空間的局所性等のアプリ依存パラメータを導入した SpMV 性能モデルを構築した．その上でフロリダ大コレクションから抜粋した 115 種の疎行列と GPU を用いて SpMV 性能モデルの評価実験を行った．その結果，GPU 上で実行する場合は Padding に関する補正と小さな行列での補正が必要であることと，長行を折り畳むなど適切な負荷分散がなされた場合はキャッシュのヒット率よりも，空間的局所性やインデックス転送の抑制の方が実効性能に敏感であることが明らかになった．

1. はじめに

近年，グラフ処理に代表されるビッグデータ解析処理が注目を集めている．これらにおいては，巨大で複雑な非零要素配置を有する疎行列で表現される処理が必要になる．Web サイトの重要性を与える PageRank[1][2]や，嗜好分析･リコメンデーションを行なうための処理の Random walk with restart[2][3]などにおいて，疎行列処理の大規模化と高速化が求められている．それらを含む重要なグラフ処理は一般化された疎行列ベクトル積(GIM-V)[3]によって実現することができる．

また，科学技術における国内の重点アプリケーションの多くが大規模疎行列を係数とする連立一次方程式(疎行列ベクトル積：SpMV)に帰着される．特に非構造メッシュに由来する疎行列の場合，アクセスアドレスがランダムに近くなり，階層構造やキャッシュアーキテクチャはこのようなアクセスパターンが非常に苦手である．

上記を背景に Graph500 ベンチマーク[4]や，Top500[5]を実状に近づけるための改良版ベンチマークである HPCG ベンチマーク[6]が近年注目を集めている．これらは，どちらも疎行列処理のベンチマークである．京コンピュータをはじめ，従来のキャッシュベースな CPU では疎行列処理への対応が不十分である．このため，国際競争の場にさらされるエクサスケールマシンにとって疎行列処理への対応は極めて重要である．

筆者らの研究[7][8][9]によって，Graph500 ベンチマークを代表とするグラフ解析処理における空間的局所性は極めて低い(キャッシュライン内に平均 1 個程度しか有効データがないこと)が明らかになっている．このため，解析対象のグラフ（疎行列）の時間的局所性がキャッシュ容量でカバーできる範囲に無い場合は，キャッシュは性能低下や電力浪費の元凶になる．

筆者らは上記のような課題に対して，Scatter/Gather 機能を有する Hybrid Memory Cube(HMC)[10] [11] [12]を提案してきた．この機構は空間的局所性を大幅に改善する．

本報告では，キャッシュアーキテクチャ（ライン単位の外部メモリアクセス）で動作する場合と，Gather 機能付メモリシステムで動作させる場合について，アプリケーションの特徴量を反映可能な SpMV 処理速度のモデルを示す．さらに，処理性能モデルの妥当性をフロリダ大コレクション[13]から抜粋した 115 種の疎行列とキャッシュベースの GPU を用いた評価により検証する．

本報告の構成は以下の通りである．2 章では SpMV 性能を変動させる様々な要因に関して考察する．3 章では処理速度のモデルに関して述べる．4 章では評価実験について述べる．5 章では敏感な性能変化要因について考察し，6 章でまとめと今後の課題について論ずる．

2. SpMV 性能変動要因の多様性

SpMV 性能はアプリケーション由来の疎行列の非零要素配置の多様性に起因して，同じソースコードに対しても疎行列が変わった時の性能変動が複雑かつ大きい．にもかかわらず，疎行列の種類に対して広範囲に適用でき，かつ精度が高い性能予測手法が確立されていなかった．それは SpMV の最適化や，効率的な疎行列ライブラリ構築の障害であった．以上を鑑み，本章では SpMV 性能を変動させる様々な要因に関して考察する．なお，本章は 4 章の実験で用いられる GPU に限らず，その他のプラットフォームにも通用する一般論として記述する．

2.1 疎行列の非零要素配置に由来する要因

SpMV においてはベクトルに対する間接参照(配列のインデックスが配列になっているメモリ参照)があり，疎行列の非零要素配置がそのインデックス配列の中身を決定する．それがアプリケーションごとに劇的に異なり，こうして生み出される複雑なメモリアクセスパターンとキャッシュと

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

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

の相性や，負荷分散に劇的な影響を与えることがある．本節では，疎行列の非零要素配置に由来する SpMV 性能変動要因を列挙する．

(1)行数

行数は SpMV のベクトルのサイズを意味する．そのベクトルは前述のように間接参照されるため，性能に大きなインパクトがある．具体的には，キャッシュサイズとベクトル用配列の容量の大小関係が最も顕著な変動を与える．SpMV が反復的に行なわれる利用形態においては，キャッシュ容量の中にベクトルが全部入る場合は，複雑なメモリアクセスが外部メモリには出なくなり，キャッシュの高いバンド幅で決定される高い処理性能が得られる. また, GPU などのメニーコア型のアクセラレータを用いる際には，一部の並列アルゴリズムでは総スレッド数となるため，行数は並列処理性能に大きなインパクトがある．

(2)非零要素総数

非零要素数は SpMV の総演算回数と直結した要因である．メモリアクセス効率が同じだった場合は，非零要素数に概ね比例した計算時間がかかる．GPU などのアクセラレータを用いる場合，ホスト～アクセラレータ間の転送やホストからの起動にかかるオーバーヘッドが，非零要素数が小さい行列においては計算処理時間と比較して無視できなくなり，顕著に処理性能が低下する．

(3)一行内の最大非零要素数

一行内の最大非零要素数は GPU 向け疎行列格納形式の一つである ELL 形式[14][15]における長方形配列の幅を与え，必要なメモリ容量を決定する．ELL 形式においては一行内の最大非零要素数に満たない行では足りない分を 0 で Padding し，GPU における高コストな条件分岐を回避する．そこでは 0 による無駄な計算が実行され，実質演算で換算する FLOPS 値を導く際の効率に直結する. 各スレッドのループ回数として，実行時間は最大非零要素数により左右される．京コンピュータにおける SpMV 部の最適化に関する実施例[16]においても，一行内の最大非零要素数がある程度以下に納まり，かつ，ほぼ一定なアプリケーションにおいてのみ, ソフトウェアパイプライニングの手法が使える．

(4)一行内の非零要素数の分散(標準偏差)

一行内の非零要素数のばらつき具合は負荷分散に甚大な影響を与える．一行内の非零要素数の分散が小さく，かつ少数の飛びぬけて大きい最大非零要素数を有する疎行列では ELL 形式における Padding の比率が大きくなってしまい, スレッド間の実質的な負荷分散が破綻する．回路行列や，Graph500 の疎行列のようにスモールワールド性のあるグラフにおいては，上記のような負荷分散の破綻が起きてしまう．

(5)一行あたりの平均非零要素数

一行あたりの平均非零要素数は，上記の負荷分散の問題を改善する GPU 向け疎行列格納形式の一種である Fold 法[17][18]における長方形配列の幅を左右する. ELL では最大が実行時間やメモリ容量を左右するのに対し，Fold 法は平均が各スレッドのループ回数として実行時間を左右する．行方向に複数スレッドを割り当てる GPU 向けアルゴリズムにおいては, Warp 内の起動スレッド数と関連する. なお, SpMV とベクトルの内積を多用する CG 法などの反復法においては，全体における SpMV の割合を左右する要因でもある．一行あたりの平均非零要素数が小さい疎行列ほど，SpMV の比率が少なくなり，内積の比率が大きくなる．その場合，SpMV だけでなく内積の高速化も重要になる．

(6)時間的局所性指標

上記の(1)～(5)までは考慮されている先行研究が存在するが，たとえ(1)～(5)までの指標が同一でも，大幅に性能が異なりうる．CPU だけでなく Fermi[19]や Kepler[20]世代の GPU でもキャッシュアーキテクチャを基にしており，キャッシュにヒットした場合は外部メモリへのメモリアクセスが行なわれない．時間的局所性指標[21][22]とは，インデックス配列の内容がもたらすアクセスパターンにおいて，同一ラインが再びアクセスされるまでのアクセス回数の平均値である．プラットフォーム上で準備されているキャッシュのライン数がこれをカバーできる場合はキャッシュのヒット率が高く，不足分が大きくなるほどキャッシュのヒット率が下がる．

(7)空間的局所性指標

たとえ(1)～(6)までの指標が同一でも，大幅に性能が異なりうる．その原因はヒット率がたとえ同一でも，ミス時のペナルティが異なれば大幅に性能が異なりうるためである．ミス時のペナルティに直接的に関連する指標が疎行列の空間的局所性指標[22][23]である. CPU だけでなく Fermi 以降の GPU でもキャッシュアーキテクチャを基にしており, キャッシュにミスした場合はメモリアクセスがキャッシュライン (64～128B) 単位で行なわれる．空間的局所性指標はミス時のリプレース動作において転送されるライン中で直近に使われる有効なデータ数の平均値を表している．

2.2 プラットフォームに由来する要因

本節では，プラットフォームに由来する SpMV 性能変動要因を列挙する．

(1)外部メモリバンド幅

SpMV は 2 回の浮動小数演算に対して, 1 回のベクトルへの間接参照のみならず，同数の再利用性のないメモリアクセスが行列値およびインデックスに対して 1 回ずつ必要になるため，現状の CPU や GPU のバランスでは，性能のルーフラインは外部メモリバンド幅で決まる．このような観点からは GPU や Intel®* Xeon® Phi™ [24][25]のようなメニーコアプロセッサは外部メモリバンド幅が高く設定されているので, COTS な CPU より高い SpMV 性能が期待できる．ただし，ピークバンド幅ではなく実効バンド幅が SpMV 性能に効くので，必ずしもピークバンド幅に比例した性能に

なるとは限らない．例えば，同じ GPU 上でも多重化されているスレッド数が多い時と少ない時では実効外部メモリバンド幅は大きく異なる．

(2)キャッシュラインサイズ

SpMV はベクトルへの間接参照があるため，アクセスパターンがランダムに近くなることが多い．その際，前節の空間的局所性はキャッシュラインサイズが小さいほどランダムアクセスによるダメージが少なくて済む．ただし，HPC 用途で用いられているプロセッサのキャッシュラインサイズは 128 バイト固定（京コンピュータや GPU）または 64 バイト固定（Intel Xeon 等）である．SX-9[26]や SX-ACE[27]といったベクトル型スーパーコンピュータも ADB(Assignable Data Buffer)という小容量なキャッシュを併用するようになってきている．キャッシュである以上ラインサイズの影響を受けるがラインサイズは不明である．

(3)キャッシュ容量

SpMV はベクトルへの間接参照があるため，外部メモリバンド幅がボトルネックになりやすい．ただし，ベクトル全体が入るだけのキャッシュ容量を有する場合は，SpMV の反復的実行におけるベクトルへのアクセスが外部メモリに出なくなるため，高性能が実現できる．ただし，実際にはキャッシュ容量に全体が入ってしまうような小さな行列は最初から処理性能が問題にならない．GPU の場合は CPU よりキャッシュ容量が少ないため，全体が入ってしまうケースは稀である．多数のノードで並列処理する際には 2 次元分割をするなどでベクトルの全体ではなく一部だけ持てば良いようなアルゴリズムも存在する．その場合はベクトルが全てキャッシュ上に載るような状況を作れる可能性がある． ただし，ボトルネックがノード間通信に移動するので，通信性能が低い環境では必ずしも得策にはならない．

(4)キャッシュバンド幅

SpMV はベクトルへの間接参照があるため，アクセスパターンがランダムに近くなることが多い．このため通常 L1 キャッシュだけでは時間的局所性をカバーできない．一方，L2 キャッシュや L3 キャッシュを有することが一般的で，階層が低いキャッシュほど大容量だがアクセス遅延や実効バンド幅が低くなる．数十 MB の L3 キャッシュを有する CPU も存在するが，大容量な L3 キャッシュは外部メモリバンド幅の 2～4 倍程度しかバンド幅がない．このため，L3 キャッシュでベクトルが全て納まる場合でも，納まらない場合の 2～4 倍程度しか高速化しない．この点は最新のベクトル型スーパーコンピュータである SX-ACE の ADB の容量はチップあたり 4MB と COTS の CPU より小さい上にバンド幅は主記憶バンド幅の 4 倍になっており，比率は COTS の CPU の L3 キャッシュのそれとあまり変わらない．

(5)演算器スループット

SpMV はメモリバンド幅ネックなアプリケーションであるので，演算器のスループットが性能を規定することは稀であると考えられる．全ての配列が L1 キャッシュまたは共有メモリ（GPU の場合）上に載る場合は演算器のスループットが SpMV 性能を決める可能性がある．

(6)Gather スループット

SpMV はベクトルへの間接参照があるため，Gather 処理のスループットが性能に影響を及ぼす可能性がある．Gather はベクトル型スーパーコンピュータや Intel Xeon Phi のようにプロセッサ側で行なうシステムが一般的である．Intel Xeon Phi は 64 バイトのキャッシュラインサイズの縛りがあるため Gather する対象が多数のラインに分散しがちなランダム性の高いインデックスではあまり高いスループットが期待できない．さらに，Gather する対象のデータが外部メモリにある場合は，空間的局所性の問題が解決できていない．筆者らはこの欠点を改善すべく，図に示すようなメモリ側で Gather を行なうメモリシステム(Gather 機能付き HMC)を提案している．その Gather スループットは，有効データの少ないキャッシュラインを用いて細い CPU～メモリ間配線を転送するより，劇的に高スループットが期待できる．

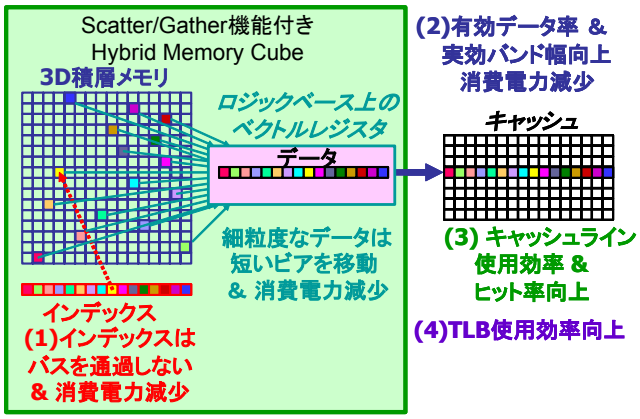

図 1 Scatter/Gather 機能つき Hybrid Memory Cube による間接参照の問題の解決

(7)メモリアクセス遅延隠蔽機能

SpMV においては行列値配列とインデックス配列は再利用性が無い逐次アクセスであるため，プリフェッチが効く．通常，現代的な CPU はハードウェアプリフェッチとソフトウェアプリフェッチの両方を持っている．2 つ程度の逐次アクセスストリームは通常ハードウェアプリフェッチが追随できる．プリフェッチが効かない場合はミス時のリプレースに伴う遅延が隠蔽されないため，性能低下が起きる．一方，GPU は大量のスレッドをサイクルごとに切替えながらリプレース遅延を隠蔽することができる．通常 SpMV のベクトルは間接参照のためインデックスの中身が連続になっていない限りプリフェッチがかからない．しかし，Gather 機能付きメモリでは間接参照が連続参照にコード変換されるため，行列値やインデックスのようにプリフェッチをかかる．

(8)インデックス配列の格納場所

SpMV は 2 回の浮動小数演算に対して, 1 回のベクトルへの間接参照のみならず，同数の再利用性のないメモリアクセスが行列値およびインデックスに対して 1 回ずつ必要になる．インデックス配列は再利用性が無いため，ADB を有する最新のベクトル型スーパーコンピュータにおいても，小容量な ADB には置けない．大規模な疎行列の計算ではインデックスも 64 ビットとなるため，単精度浮動小数 2 個分の大きなバンド幅を消費する．一方，メモリ側での Gather によるとインデックス配列はメモリ側からプロセッサ側には移動しないため，外部メモリバンド幅を形成するプロセッサ～メモリ間配線のボトルネックを通過しない．近年，インデックスを圧縮してインデックス配列が消費するメモリバスバンド幅を節約するアルゴリズム[29][30]が提案されているが, 上記ハードウェアはそれを皆無にする．

3. SpMV 性能モデル

本章は前章で示したアプリケーション由来の要因や，プラットフォーム由来の要因を考慮した SpMV 性能のモデリングを行なう．キャッシュアーキテクチャ（ライン単位の外部メモリアクセス）で動作する場合と，Gather 機能付きメモリシステムで動作させる場合について，SpMV 性能のモデルを整理する．

3.1 キャッシュの処理速度モデル

キャッシュベースのプロセッサを用いる場合は通常は Scatter/Gather をプロセッサ側で行なう．本節のモデリング対象は，キャッシュベースのプロセッサによる SpMV 処理速度である．1FLOPS あたりの必要バンド幅[B/FLOP]の式を作るにあたり，アプリケーション由来のいくつかのパラメータを以下のように定義する．

I : インデックス 1 個のデータサイズ[B]
F : 浮動小数 1 個のデータサイズ数[B]
L : キャッシュラインサイズ[B]
S(F,L) : 空間的局所性 (ライン内の有効データ数)
hit_x : 列ベクトル x へのアクセスのヒット率

1FLOPS あたりの必要バンド幅は以下の三つの値 α , β , γ の和で表される．

$$BPF_{cache} = \alpha + \beta + \gamma = F/2+I/2+(1- hit_x)*L/S(F,L)\ [B/FLOP] \quad (1)$$

α : index に必要なバンド幅 : I/2[B/FLOP] (連続アクセス＆再利用性なし)
β : A に必要なバンド幅 : F/2[B/FLOP] (連続アクセス＆再利用性なし)
γ : x に必要なバンド幅 : 有効データのみで F/2[B/s]を出すのに必要なバンド幅

また，x はキャッシュ容量より十分に大きく，キャッシュミスのリプレース動作によりキャッシュに取り込まれるものとしている．

上記の γ は 0.5 個の x(F[B])をリプレースで持ってくる時に消費されるバンド幅に対応する。ミス 1 回で S 個の x(F[B])をリプレースで持ってくるのに L[B]を 2 回移動する。ミス率 1 の時 0.5 個の x(F[B])をリプレースで持ってくるのに 0.5*2L/S＝L/S [B]を移動する。ミス率(1- hit_x)の時 1FLOP あたりに $\gamma = (1- hit_x)*L/S[B]$ を移動することになる．よって，主記憶(GPU の場合はデバイスメモリ)のバンド幅 W_{cache}[B/s]によって得られる処理速度は以下のようになる．

$$F_{cache}=W_{cache}/(F/2+I/2+(1-hit_x)*L/S(F,L))[FLOPS] \quad (2)$$

なお，GPU では分岐コストが高いため，0-Padding（本来値が 0 の要素を省略しないで記憶領域を割り当て 0 で初期化すること）を行なうことがある．その場合の処理速度は 0 による演算の水増し分を以下のように補正する必要がある．ここで Nnz は元の非零要素総数，Nnzp は Padding 後の非零要素総数である．

$$F_{cache0}=Nnz/Nnzp*W_{cache}/(F/2+I/2 + (1-hit_x)*L/S)[FLOPS] \quad (3)$$

3.2 提案メモリシステムの処理速度モデル

提案メモリシステムでは Scatter/Gather をメモリ側で行なう．メモリ側 Gather では index はメモリ側から動かないので α に相当するバンド幅消費は Gather スループット W_{gather}[B/s]の中に組込まれる．β はキャッシュの場合同様 F/2[B/s]であり，x はメモリ側 Gather によって連続化されるので γ は β 同様に F/2[B/s]であり，全体として以下の式で 1FLOPS あたりの必要なメモリバンド幅が表される．

$$BPF_{gather} = F\ [B/FLOP] \quad (4)$$

Gather スループット W_{gather}[B/s]の場合の得られる処理速度は以下のようになる．

$$F_{gather} = W_{gather}/BPF_{gather} = W_{gather}/F[FLOPS] \quad (5)$$

なお，GPU の外部メモリとして Gather 機能付きメモリを用いる場合は 0-Padding が適切であり，その場合も前述のとおり 0-Padding に伴う水増し分の補正が必要である．その場合の処理速度は以下のように補正する．

$$F_{gather} = Nnz /Nnzp*W_{gather}/F[FLOPS] \quad (6)$$

4. 評価

4.1 実験環境と評価行列

今回の実験に用いた計算機環境を表 1 に示す．また，評価に用いた行列は University of Florida Sparse Matrix Collection から抜粋した 115 個の正方行列であり，文献[31]の評価行列のサブセットである．図 2 と図 3 は行列特性の中で性能を左右させる非零要素数と行数の分布を示したものである．これらの図から読み取れるように偏りのない分布になっているため，特定のアプリ種や形状を持っていないと判断できる．

表 1 測定環境

CPU	Intel® Xeon®CPU　X5670 @ 2.93GHz
GPU	Nvidia Tesla C2050 (コア数 448) L1 キャッシュ:16KB, L2 キャッシュ:768KB デバイスメモリ: 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	Cuda3.2

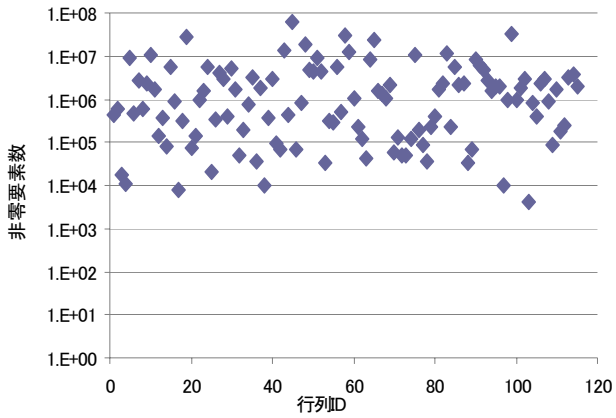

図 2　評価行列の非零要素数の分布

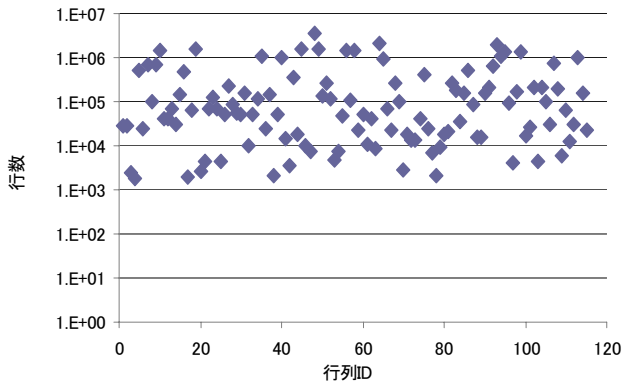

図 3　評価行列の行数の分布

ここで，実測性能を計測するために，実行時間及びキャッシュヒット率を計測する．実行時間については，各評価行列に対して疎行列ベクトル積（SpMV）の GPU カーネルを 100 回計算し，その平均値を算出し，SpMV に必要な浮動小数点演算回数をその平均値で割り，FLOPS 値を測定する．また，評価に用いた GPU（TeslaC2050）には L1・L2 キャッシュがあり，CUDA3.2 においてはプロファイラの性能カウンタでそれらのキャッシュヒット率の測定が可能である．

測定環境においては数式(3)の W_{cache} には GPU のデバイスメモリバンド幅の 144GB/s，また，実測ヒット率は A，index，x の 3 つのアクセスのヒット率を示す．測定に用いた SpMV の CUDA プログラムは A，index は隣り合うスレッドが連続する領域をアクセスする状態となるため，スレッドが多数多重化されている(行数が多い)状況では実質的にプリフェッチが効いてほぼ全てがヒットしている状態と考えられる．従って，ヒット率を左右しているのは列ベクトルのアクセスによるミス回数でだけある．行列値 A，index，列ベクトル x は同じ回数アクセスされるので，数式(3)の中の hit_x は実測ヒット率の約 1/3 と近似できる．

L1 ヒット率は本来測定環境で実行してみないとわからないパラメータであるが，たとえ測定環境が無かったとしても疎行列の非零要素配置さえわかっていれば，筆者らの先行研究で用いた時間的局所性測定ツール[21][22]によって，大まかな予測を行なうことができる．

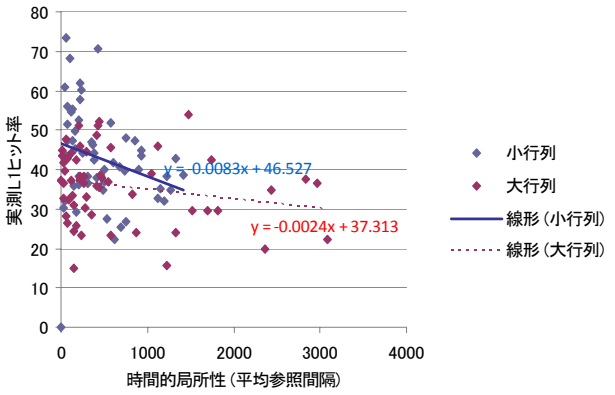

図 4　時間的局所性(参照間隔)と L1 キャッシュヒット率の相関（閾値＝7 万行）

図 4 に時間的局所性と L1 キャッシュヒット率の相関を示す．特に時間的局所性が低く出ている領域での誤差が大きいことがわかる．ここで仮に 7 万行を境に小行列(紫の点)と大行列(青の点)に分類すると，近似直線の方程式などの分布傾向に若干の違いが見られることがわかる．ヒット率の予測を正確にするには，行列サイズを考慮して，大きい場合と小さい場合に分類して，別の近似式を用いることが

有効であることが示されている．ただし，いずれのグループでも近似直線から 30%程度以内には入っており，右肩下がりの傾向と，右にいくほど散らばりが減るという定性的な傾向は同じである．

ここで，近似直線から外れる散らばりが出ている原因としては，時間的局所性測定ツールの想定するリプレースメントポリシーが Denning 流(FIFO)になっているのに対し，測定環境の GPU のリプレースメントポリシー(詳細不明だが一般的には LRU がベースになっていることが多い)が一致していないことが考えられる．

次に，現状の 2 本の近似直線(大行列と小行列の 2 グループで 2 方程式)を用いることで生じている誤差が，最終的に予測したい SpMV 処理性能にどれ位のインパクトを与えるかについて検証する．誤差により SpMV 処理性能が激変する場合は問題である．

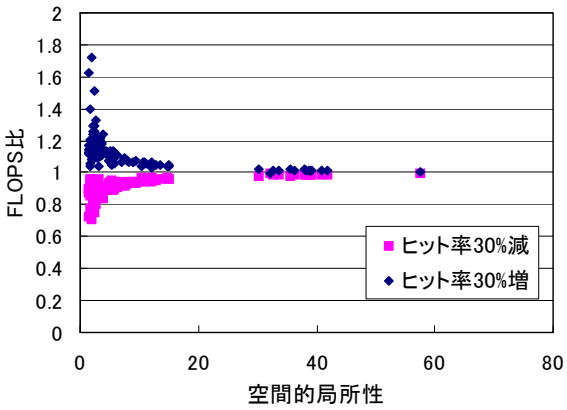

図 5　空間的局所性と誤差 30%のヒット率を用いた場合のモデル性能比の相関

図 5 に空間的局所性と誤差 30%のヒット率を用いた場合のモデル性能比の相関を示す．ヒット率が 30%上に外れた予測値を用いても平均で 12.6%しか SpMV 性能には反映しない．30%以上の変化を見せるのは，115 サンプルのうちわずか 5 例しかない．ヒット率が小さい方向に見積もった場合は全てのサンプルで 30%以上の変化は見られなかった．つまり，ほとんどの場合ではヒット率が SpMV 性能に与える影響はヒット率向上率の 1/3 程度の小幅な影響しかない．SpMV 性能により高い影響力のある要因がある場合は，上記の誤差は大きな問題ではないと考えられる．

次に，GPU の実験環境を用いた測定による SpMV 処理性能と，モデルから得られる SpMV 処理性能の間の相関について検証を行なう．図 6 は数式(3)で示された Padding 補正後のモデルに前述の hit_x の近似値を代入して得られる SpMV 性能と実測の SpMV 性能の相関をとったものである．相関係数は 0.20 であり，弱い相関となっている．全体としてモデルによる値より実測値が下方向にぶれているサンプルが多い．以降ではこの原因を探ることにする．

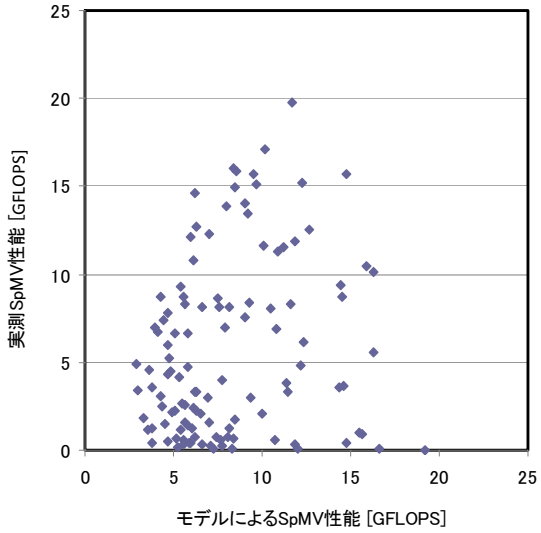

図 6　実測性能とモデル性能の相関

図 7 と図 8 ではそれぞれの局所性指標値と測定環境のモデルの GFLOPS の相関をとっている．時間的局所性指標と比較すると，空間的局所性指標の方が強い相関が得られることがわかる．これは，空間的局所性を変化させた場合と時間的局所性（キャッシュヒット率）を変化させた場合を比べると，空間的局所性指標を変化させた方がより SpMV 性能が敏感に変化することを示唆している．

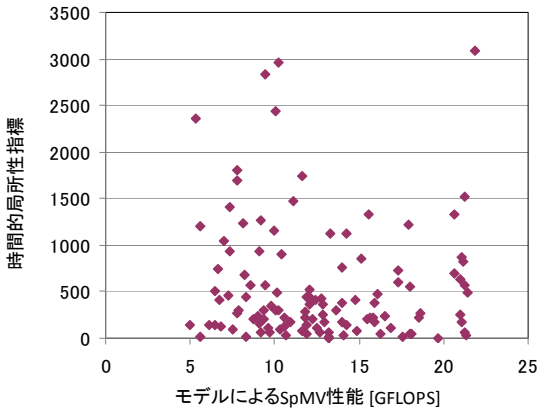

図 7　時間的局所性指標値とモデルの相関

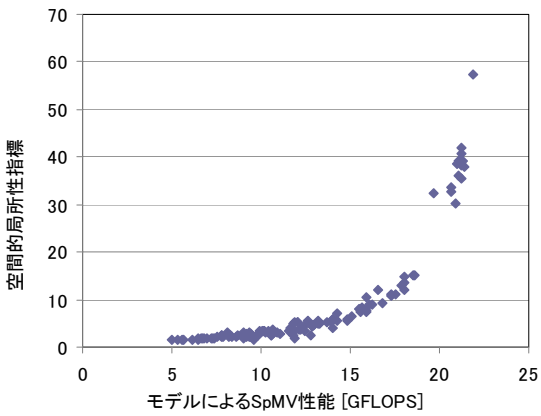

図 8　空間的局所性指標値とモデルの相関

一方，実測性能とモデルを見比べると，行数が比較的大きいものについては相関があるが，小さい行数(3 万行以下)になると相関が小さいことが見受けられる．これは，性能モデルでは SpMV が 128 バイト固定長ラインのキャッシュベースで動き，GPU(Tesla　C2050)の場合，1Warp＝32 スレッドが同時に連続する 32 個の float をアクセスしている．メモリアクセス遅延を隠蔽するのに十分な数のスレッドが同じ SM（Streaming Multiprocessors）上で実行されるようなプログラムを開発しなければ，A や index の配列は 100%の効率でメモリロードが行なわれない．行数が小さい行列に対しては，SM 上で多重化されるべきスレッド数が不足し，メモリアクセス遅延が隠蔽されきれないため，再実行が多発する．このメモリアクセス遅延を隠蔽するには，多重度がクロックサイクルタイムあたりのアクセス遅延より大きいときは隠蔽が可能だが，小さいと隠蔽できない．さらに，行内の非零要素数が少ない行列ではキャッシュライン内に複数の行が詰め込まれることも多く，他の Warp が参照した行列 A が L1 キャッシュにヒットしてしまう．これらが，モデルと実測性能差が生じる要因と考察する．

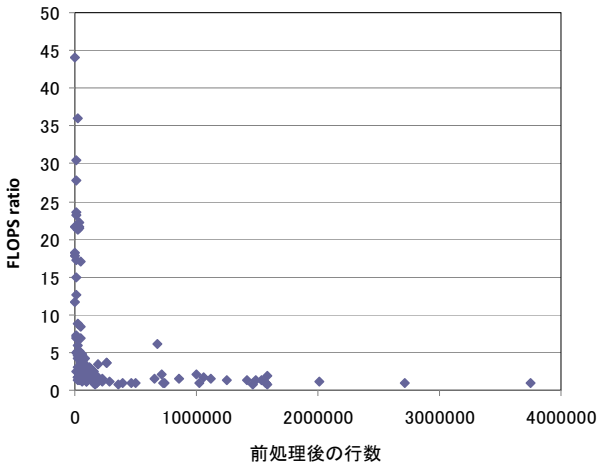

図 9　実測/とモデル性能比と前処理後の行数の関係

図 9 の縦軸は実測性能/モデル性能の比を表している．前処理後の行数が少ないものほど実測値とモデルの差が大きくなることが判る．また，スレッド数（行数）が少ない時にはメモリアクセスが非効率となり，性能比に影響を及ぼす可能性がある．そこで，ずれが大きくなっている行列を中心にいくつかの小行列を抜粋し，比較を行う．

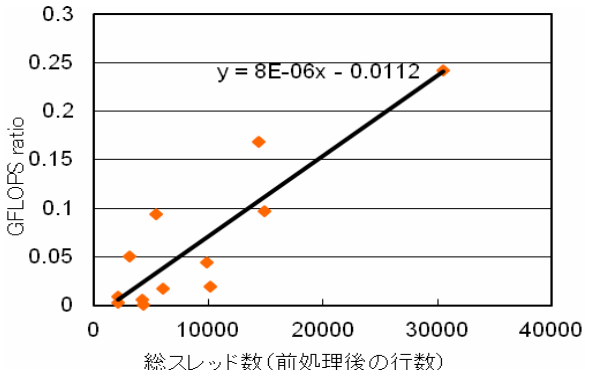

図 10　実測/モデル性能比とスレッド数（一部抜粋）

図 10 は抜粋した小行列の実行結果をグラフ化したものである．図 10 より，スレッド数(前処理後の行数)が少ないと実測値との差が大きくなることが判る．カーネルの起動や計測時間のオーバーヘッドは，実行時間が短いときほど演算回数あたりの性能値に影響が大きくなると考えられる．

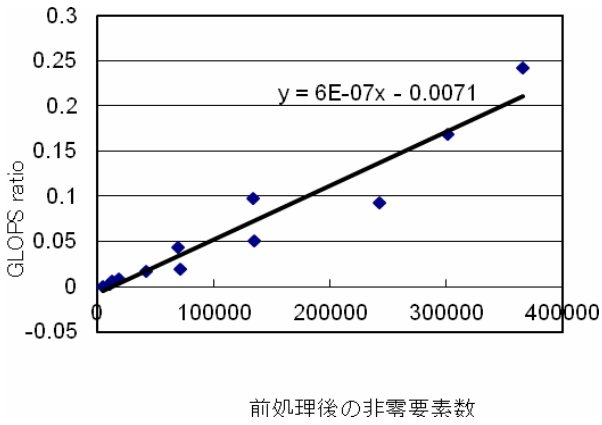

図 11　実測/モデル性能比と非零要素数（一部抜粋）

図 11 では，図 10 と同じ縦軸を使って，実行時間を決定する行列の非零要素数と比較する．図 10 と比較して，非零要素数との相関をとる図 11 の方がより相関が強いといえる．したがって，この近似直線の係数である 6e^-7 を測定環境における小行列用のモデル式の補正係数とする．式(3)のモデル式に I=4,F=4,L=128 を代入し，小行列用の補正をしたモデル式が式(7)である．ここで N は前処理後の非零要素数である．

$$F'_{cache} = W_{cache}/(6+(1\text{-}hit)*128/S)*(N*6e\text{^-}7)\ [FLOPS] \qquad (7)$$

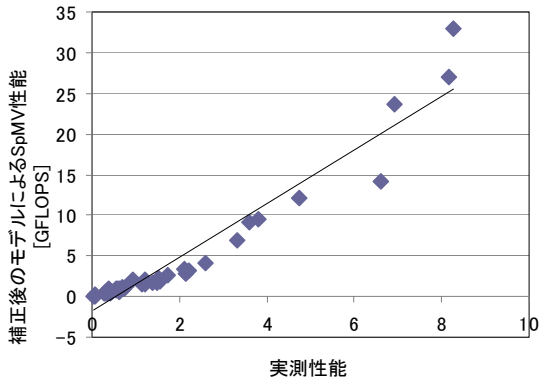

図 12　小行列における実測性能と補正後のモデル性能

図 12 は抜粋した小行列に対して，数式(7)のモデル式を適応した結果である．補正係数を加えたモデルと実測性能は強い相関があることが分かる．そこで，全てサンプルで数式(7)のモデル式を適用し，補正を行うべき行数の閾値を決める．

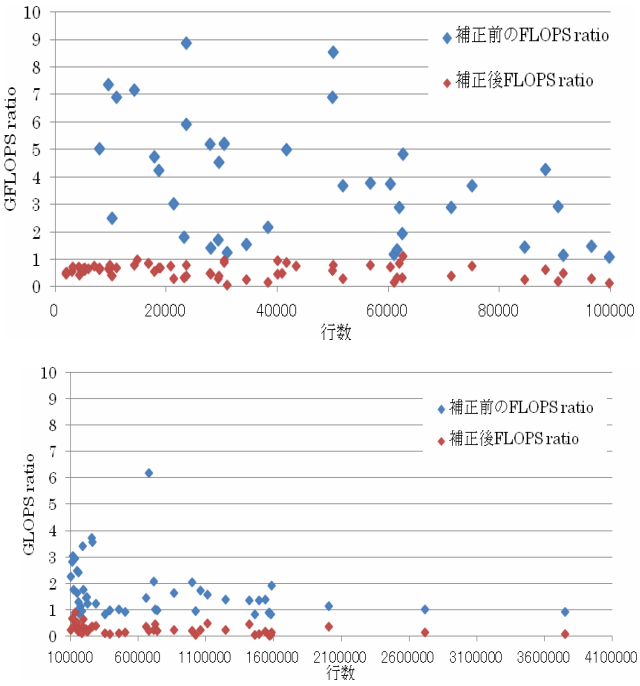

図 13　小行列用補正前後での性能比と行数の関係

図 13 では，数式 2 を用いたモデル性能と実測性能の比（補正前）と数式 5 を用いたモデル性能と実測性能の比(補正後)を 2 色に分けてグラフにしている．注意点として，10,000 行以下の小行列は縦軸の最大値を超えた大きなずれが起きている．グラフより，補正の効果があるといえるのは 60,000 から 70,000 行のあたりであると考察できる．したがって，70,000 行を閾値とする．

次に，図 14 において，閾値より小さい行列には数式 5 を，大きい行列には数式 2 を適用して実測性能と比較を行う．外れ値を一部含むが，相関係数は 0.47 となり，図 9 より相関が強くなった．

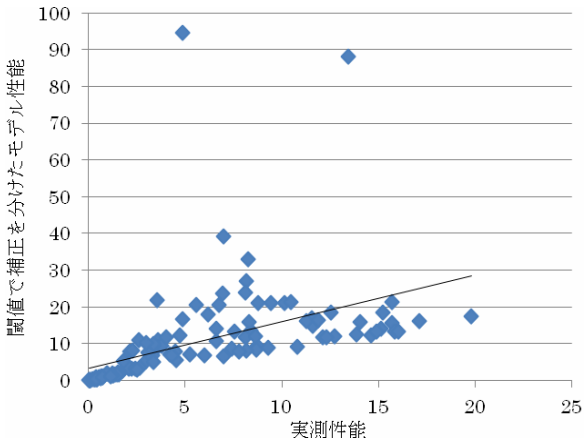

図 14　小行列補正後のモデル性能と実測性能の相関（閾値 7 万行の場合）

これらの結果をまとめると，評価行列では行数の大きい行列は実測性能とモデル性能の相関が大きく，行数が大きいほど正確さが増す傾向がある．この領域では空間的局所性がキャッシュ性能を決定づけていることが分かる．また，行数の小さい行列（約 7 万行以下）では前処理後の行数や実行時間に大きく関わる非零要素数と実測性能は相関がある．

なお，現実問題としては，補正が必要になるような 7 万行以下の疎行列では GPU においてはスレッド数不足に伴うメモリアクセス効率の低さだけでなく，ホスト間通信に伴うボトルネックの観点からも，GPU で動かすのではなく CPU 上で実行してしまった方が結果的に短時間で済んでしまう可能性も高い．

5. 敏感な性能変化要因の考察

チューニングにおいては様々な要因の中から目的とする性能に敏感に作用する要因を特定することが極めて重要である．図 7 および図 8 の傾向から，従来チューニングにおいて最も重視される傾向があったヒット率の向上より，空間的局所性の向上の方が SpMV 性能に敏感に作用することが示唆された．本章ではその点を掘り下げて考察を加える．

従来の HPC 全般におけるチューニングにおいて最重要視されてきたのがキャッシュのヒット率である．SpMV 実行時のヒット率を 30%上げることは，チューニングの専門家がアプリケーション固有の特質を考慮した高度な努力によって達成できるかどうかという困難度の高い作業である．それが計算機を専門としない自然科学者のチューニングに対する参入障壁の一つになっていると考えられる．よって，その障壁を取り除くことは科学技術全体の進歩にも寄与しうる重要なことと考える．

図 5 や図 7 および図 8 の傾向から，キャッシュのヒッ

ト率は SpMV 性能に与える影響は%の絶対値上でもあまり大きくなく，他にもっと大きい影響を持つ要因があることがわかる．キャッシュのヒット率をアプリ依存のオーダリングで上げても性能に対するインパクトがあまり大きくないという現象は幾つか報告されている．例えば，Graph500 の最適化においてオーダリングの工夫(Vertex sorting)を行なっても 10%程度しか性能に変化が出なかった例[29]や，京における有限要素法のアプリケーションにおいてオーダリングの工夫(物理的に近接する節点が近接する節点番号を持つようにする手法)を用いた場合 37%(ピーク比 5.9%→8.1%)の性能向上が限界だった例[16]などがある．

オーダリングは主に時間的局所性に作用する手法であり，副作用として空間的局所性に影響が出る時もありえるが，それはアプリケーション依存である．Graprh500 の例[29]のようにランダム性が高く，行内非零要素数変動が大きく，空間的局所性が 1 近辺であるような悪性のアプリケーションでは順番を少し変えたくらいでは空間的局所性には大きな変化が出ない．よってオーダリングによる加速率は小さい．それに対して京における有限要素法の最適化[16]は比較的幸運なケースである．物理空間上の局所性を節点番号の局所性に反映させ，行列の対角付近に非零要素が集まり，キャッシュライン内要素数(32)に近い近接節点数(上限 28 付近で安定)である性質が利用できて，空間的局所性を高めることに成功している．ただしそれでも加速率は 37%が限界であり，Gather 機能付きメモリが悪性な行列の空間的局所性を 1 から 32 に引き上げてしまう効果に比べると格段に小さいと考えられる．

そこで次に，ヒット率の代わりに空間的局所性が 32 に引き上げられた場合の SpMV のモデル性能の変化を観察する．

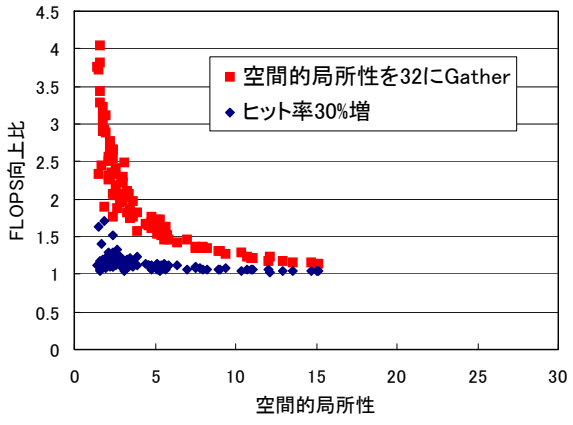

図 15　ハードウェアで空間的局所性を 32 に引き上げた場合とヒット率を 30%増加した場合のモデル性能向上比の違い(インデックス配列転送抑制分は除く)

図 15 の赤い点は元の空間的局所性とハードウェアで 32 に引き上げられた空間的局所性を用いた場合のモデル性能比の相関を示している．なお，ツールの出力上で 32 以上になっている最初から良好な場合は，そのままの値を用いている．比較のために図 5 のヒット率を 30%向上できた場合の性能比を併せて図示している．これより，明らかに空間的局所性を向上させた場合は全ての場合においてヒット率 30%増の最適化を行なった場合よりも格段に高い性能向上を見せることがわかる．

なお，この条件設定においては Gather 機能付きメモリを用いた場合に不要になるインデックス配列が消費するバンド幅の排除の効果が入っていない．よって，図 15 で表示されている加速率は控えめな予測になっている．その効果は同サイズの配列 3 本だったものが 2 本のロードへの変化になるので、あと 1.5 倍程度性能が向上するということになる．1.5 倍という効果の大きさは前記の京における有限要素法の最適化で最終的に得られた効果よりも大きい．インデックス配列移動の抑制と空間的局所性改善は全く独立して作用するので，これらの二つの敏感な要因の改善を組み合わせた場合の効果は高いと考えられる．

現状の日本の HPC 戦略のトレンドは，京と同様の保守的なアーキテクチャのままで，対象を重点アプリに絞り込み，チューニングの専門家を割り当てて，主にヒット率の改善によって有限要素法の例のように運が良い時にのみ 40%程度の加速を得るという方向性が優勢と考えられる．ただし，その戦略は重点アプリに選ばれなかった広い裾野への波及力に乏しい．

これに対し，メモリモジュールを改善し，空間的局所性の改善とインデックス転送の抑制をすることで，桁違いな SpMV 性能の加速がアプリケーション(非零要素配置)を問わず，チューニングの専門家を割り当てずに得られることが示された．これは一種のゲームチェンジと言える．どちらがより多くの国内の自然科学者・ビッグデータ解析(グラフ解析)ユーザや，スパコンの開発費や電気代を支払って日本の広範囲な技術力・産業競争力向上を買う納税者にとって幸せな状態に近いかは言うまでもない．ただし，新型メモリモジュール(HMC)の実用化にはそれなりのリソース投入と開発期間が必要である．

6. おわりに

本報告では，SpMV 性能を左右する様々な要因をアプリケーション依存の要因とプラットフォーム依存の要因に分けて考察した．それらは SpMV の最適化において有益な指針を含んでいる．それらを踏まえて行列の非零要素配置から導かれる時間的局所性と空間的局所性等のアプリ依存パラメータを導入した SpMV 性能モデルを構築した．その上でフロリダ大コレクションから抜粋した 115 種の疎行列と GPU を用いて SpMV 性能モデルの評価実験を行った．その結果，GPU 上で実行する場合は大きな行列では概ね相関が高いが，Padding に関する補正と小さな行列での補正が必要であることが明らかになった．さらに，長行を折り畳む

など適切な負荷分散がなされた場合は,キャッシュのヒット率よりも，空間的局所性やインデックス転送の抑制の方が実効性能に敏感であることが明らかになった．これは近年ニーズが高まっている疎行列処理に強い将来の計算機の設計において，重要な設計指針を与えていると考える．

今後の課題は, Kepler 世代の GPU や Xeon Phi などのより新しいプラットフォームを用いたモデルの評価，性能モデルと実測性能の差を生む L2 キャッシュヒット率を加味したモデルの再構築，負荷分散やスレッド数(アクセス遅延隠蔽効果)向上における Fold 法を用いた前処理での折り目のパラメータの最適化などがある．

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) L. Page, S. Brin, R. Motwani and T. Winograd : "The PageRank citation ranking: Bringing order to the Web", Technical Report Stanford Digital Library Working Paper SIDL-WP-1999-0120, Stanford University, 1999.
2) 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, Jan. 2011.
3) U Kang, C. E. Tsourakakis and C. Faloutsos: "PEGASUS: A Peta-Scale Graph Mining System - Implementation and Observations", International Conference on Data Mining 2009, pp.229-238, 2009.
4) Graph500 : http://www.graph500.org/.
5) Top500 : http://www.top500.org/.
6) M. A. Heroux and J. Dongarra: "Toward a New Metric for Ranking High Performance Computing Systems", Sandia National Lab. Report, SAND2013-4744 (2013).
7) 田邊，冨森，高田，城 :“グラフ解析ワークロードのキャッシュ適合性”, 電子情報通信学会コンピュータシステム研究会 vol. 112, no. 237, CPSY2012-42, pp. 67-72, Oct. 2012.
8) 田邊，冨森，高田，城 :“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”, 情報処理学会研究報告 2012-HPC-138, Feb. 2013.
9) N. Tanabe, S. Tomimori, M. Takata and K. Joe: "Character of Graph Analysis Workloads and Recommended Solutions on Future Parallel Systems", 13th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'13), Dec. 2013.
10) 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.
11) 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", 情報処理学会研究報告 2010-HPC-133, Mar. 2012.
12) N. Tanabe, J. Kogou, S. Tomimori, M. Takata and K. Joe: "Future Irregular Computing with Memory Accelerators", FUTURE COMPUTING2013, pp.74-80, 2013.
13) Tim Davis : " The University of Florida Sparse Matrix Collection”, http://www.cise.ufl.edu/research/sparse/matrices/.
14) N. Bell, M. Garland : "Eficient Sparse Matrix-Vector Multiplication on CUDA", NVIDIA Technical Report NVR-2008-004, Dec. 2008.
15) 大島，金子，片桐 : "Xeon Phi における SpMV の性能評価",情報処理学会研究報告 2013-HPC-140，Aug. 2013.
16) 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, Jan.2012.
17) N. Tanabe, Y. Ogawa, M. Takata, K. Joe : Scaleable Sparse Matrix-Vector Multiplication with Functional Memory and GPUs, 19th Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP 2011), pp. 101-108, 2011.
18) 田邊, 小郷, 小川, 高田, 城:"長行を折畳む疎行列ベクトル積方式と Gather 機能付メモリによる高速化", 情報処理学会 ACS 論文誌, pp. 112-124 Aug. 2012.
19) NVIDIA : "ホワイトペーパー NVIDIA の次世代 CUDA™ コンピュートアーキテクチャ Fermi ™", http://www.nvidia.co.jp/docs/IO/81860/NVIDIA_Fermi_Architecture_Whitepaper_FINAL_J.pdf
20) NVIDIA : "ホワイトペーパー NVIDIA の次世代 CUDA™ コンピュートアーキテクチャ Kepler ™" GK110 ", http://www.nvidia.co.jp/content/apac/pdf/tesla/nvidia-kepler-gk110-architecture-whitepaper-jp.pdf
21) 冨森，田邊，高田，城 : “時間的局所性を考慮した疎行列のキャッシュ適合性”, 情報処理学会研究報告 2012-HPC-137, Dec. 2012.
22) N. Tanabe, S. Tomimori, M. Takata and K. Joe: "Locality Analysis for Characterizing Applications Based on Sparse Matrices", 19th International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2013) Jul. 2013.
23) 冨森，田邊，小郷，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”, 情報処理学会研究報告 2012-HPC-135, Aug. 2012
24) George Chrysos : "Knights Corner, Intel’s first Many Integrated Core (MIC) Architecture Product ", Hotchips 24, Aug. 2012. http://www.hotchips.org/wp-content/uploads/hc_archives/hc24/HC24-3-ManyCore/HC24.28.335-XeonPhi-Chrysos-Intel.pdf
25) Intel : " Intel® Xeon® Phi™ Coprocessor Data sheet", Nov. 2012.
26) NEC: “SX-9 ベクトルスーパーコンピュータ”, http://jpn.nec.com/hpc/sx9/
27) NEC: “SX-ACE ベクトルスーパーコンピュータ”, http://jpn.nec.com/hpc/sxace/
28) 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, Jun. 2011.
29) W. T. Tang, W. J. Tan, R. Ray, Y. W. Wong, W. Chen, S. Kuo, R. S. M. Goh, S. J. Turner and W. F. Wong: "Accelerating sparse matrix-vector multiplication on GPUs using bit-representation-optimized schemes", International Conference for High Performance Computing, Networking, Storage and Analysis (SC'13), Nov. 2013.
30) S. Xu, W. Xue and H. X. Lin: "Performance modeling and optimization of sparse matrix-vector multiplication on NVIDIA CUDA platform", The Journal of Supercomputing, Vol.63, No.3 pp.710-721, Mar. 2013.
31) 椋木, 高橋: "GPU における高速な CRS 形式疎行列ベクトル積の実装", 情報処理学会研究報告 2013-HPC-138，Feb. 2013.