Vol.2013-HPC-141 No.18
2013/10/1

アーキテクチャの進歩を促進する Graph500 ベンチマークのあり方

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，Top500 を補うものとしても近年注目を集めてきた．従来の Graph500 参照実装は従来の計算機が苦手とする疎行列ベクトル積系応用全般を代表し，計算機の発展を促進する力があった. 一方, ビットマップ式訪問管理と Hybrid BFS アルゴリズムがデファクトになる流れにある．筆者らの予備調査の結果，それらを実装した Pre release 版 Graph500 はキャッシュに適合し，メモリアクセス負荷が大幅に軽減され，本来とは異質なワークロードになることが明らかになってきた．本報告では，計算機の発展を促進する観点から，スーパーコンピュータのランキングに利用されるベンチマークの役割と，その望ましいあり方を考察する．さらに Hybrid 動作における局所性とキャッシュ挙動の観点から上記の予備調査結果を補強するとともに，Graph500 の将来について考察する．

1. はじめに

スーパーコンピュータをランキングするための性能評価指標としては 20 年に渡り Top500[1]ベンチマーク(密行列の LU 分解である Linpack)が用いられてきた. ランキングに用いられるため，計算機ベンダーはそれが高速になるように設計する．つまり，計算機をランキングするための性能評価指標は，計算機アーキテクチャの進化の方向性を決める力がある．

ところが，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, Graph500[2]ベンチマークが Top500 を補うランキング用評価指標として登場した. Graph500 はビッグデータ解析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．特に，伝統的な技術計算よりも市場が大きいと考えられているビジネス用途を主たる背景にしている．Graph500 もランキングに用いられるため, 今後その価値を損なわないように発展し, 正しく認識されるようになると，Top500 より多くの計算機のアーキテクチャに影響を与える潜在力を秘めている．

Graph500 はグラフ処理の中心的な処理である BFS（幅優先探索）を主体とする．BFS を含む多くのグラフ処理のメモリアクセスパターンや通信パターンは，疎行列の反復解法の中核をなす疎行列ベクトル積と基本的に同じである．ここが Top500 ではドライブできずに残された未来の計算機アーキテクチャの重要な進化ポイントでもある. よって，Graph500 を大切に育成し，未来の計算機アーキテクチャの進化を促進することが，自然科学者やビジネスコミュニティを包含する多くの人々のためになる道と考えられる．

一方，アルゴリズム的に未成熟だった上に改変が自由な Graph500 は，アルゴリズムの競争の場としても使われてしまった．このため，優れたアルゴリズムを使ったエントリと，参照実装ベースの素朴なアルゴリズムを使ったエントリの間には，同等クラスのプラットフォームを用いていても桁違いの性能差が生じてしまい，プラットフォームのランキングとして Graph500 は正しく機能していなかった. 正しく機能しなければ Graph500 は Top500 を補うものとして安定成長できない．未来の計算機アーキテクチャの進化を促進することもできず，結果として後発のより良いベンチマークに取って代わられる．

現在では参照実装の Pre-release 版として，現時点で優れたアルゴリズムとして広く認識されている以下の二つ

(1)Agrawal らのビットマップ式の訪問管理方式[3]

(2)Beamer らの Hybrid BFS[4]

を適用したソースコード[5]が公開されている．Graph500 の公式サイト[2]はそれを用いることを推奨し，アルゴリズムレベルの実装の差を減らすよう促している．

筆者らは上記のような状況を鑑み，Graph500 参照実装の Pre-release 版で何が起きようとしているのかを，計算機アーキテクチャの観点から調査を行った．その最初の取り組みとして，メモリアクセス回数や，メモリアクセスの局所性に大きな変化が生じていることを確認した[6].

本報告では，Hybrid 動作とキャッシュの挙動を中心にその補強を行なうともに，ランキング用ベンチマーク全般の現状における Graph500 の位置づけをふまえつつ, 上記の変化がもたらす意味について考察する．

本報告の構成は以下の通りである．第 2 章ではランキング用ベンチマークとその役割について述べる．第 3 章では Graph500 の新旧アルゴリズムと計算機アーキテクチャの関係について論じ，Graph500 の運用に関する提言を行なう．第 4 章では Graph500 の新旧アルゴリズムの評価実験について述べる. 第 5 章で関連研究を紹介し, 第 6 章でまとめる．

2. ランキング用ベンチマークとその役割

スーパーコンピュータをランキングするための性能評価指標は，計算機アーキテクチャの進化の方向性を決める力がある．本章ではスーパーコンピュータをランキングする

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

ための各種性能評価指標を紹介し，それらの役割や問題点について考察する．

2.1 Top500 ベンチマーク

スーパーコンピュータをランキングするための性能評価指標としては20年に渡りTop500[1]ベンチマーク(密行列のLU分解であるHigh Performance Linpack:HPL)が用いられてきた．ランキングに用いられるため，計算機ベンダーはLinpack が高速になるように設計してきた．より正確にはCPU ベンダーは SPEC[7]ベンチマークを高速に実行するように設計していたが，Top500 は倍精度密行列積(DGEMM)を用いるアルゴリズムで Goto BLAS[8]などのキャッシュ向けに最適化されたライブラリを用いる場合は，多くのSPEC 内のベンチマークよりも性質の良いワークロードになっていた．このため，Top500 にはSPEC をもとに設計する慣習の変革を促がす力を持たなかった．これは，少なくとも性能面からはビジネス市場向けに設計された価格性能比が高いCOTS の CPU を Top500 用に流用できたことを意味する．

結果として COTS と基本的に同一なアーキテクチャであるSIMD 型ショートベクトル演算器（またはGPU）で浮動小数演算性能を強化したキャッシュベースのスカラ型プロセッサを用いた超並列システムが Top500 上位を独占するようになった． その結果，使い勝手の観点で自然科学者からの人気を集めていたベクトル型スーパーコンピュータは，Top500 ランキングと市場から一部の例外を除き殆ど消滅してしまった．

ところが，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 が近年の計算機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，計算機アーキテクチャの進化の観点から問題である．

一方，Top500 の派生的なランキングとして Linpack の電力あたりの性能を競う Green500[9]ベンチマークも近年ランキングされるようになってきた．これは，年々ハイエンドなスーパーコンピュータの最大の制約要因になってきた消費電力の問題の解決の方向に，計算機アーキテクチャの進化を促進する働きがある．ただし，Top500 同様に通信やメモリアクセスに負荷がそれほどかかっていない状態における電力を競うので，計算機アーキテクチャの進化を必ずしも正しい方向に導けるとは限らないと考えられる．

2.2 HPC Challenge ベンチマーク

HPC Challenge ベンチマークは Top500 の問題点を補うために，スーパーコンピュータのメモリバンド幅やノード間通信性能などの多角的な性能を測定するものとして提案されたベンチマーク群である．ベンチマークごとに負荷をかける場所が異なっているという意味で，マイクロベンチマーク的な色彩が強い．Top500 と同様のタイミングでランキングが発表されている．Top500 と異なり複数の指標で順位がつき，必ずしも 1 台のマシンが全ての項目で 1 位を取るとは限らない．このため HPC Challenge のランキングは一般人にはわかりにくく，ランキングとしての社会への影響力においては Top500 より大幅に低いのが現状である．

2.3 Graph500 ベンチマーク

Graph500[2]ベンチマークは，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， Top500 を補うランキング用評価指標として登場した．Graph500 はビッグデータ解析のベンチマークとも言われており，米国を発端にする政治的な流れ，資金の流れを背景に盛り上がりをみせているビッグデータブームにも乗って，近年注目を集めている．特に，伝統的な技術計算市場よりも市場規模が大きいと考えられているビジネス用途を主たる背景にしているため，従来のスーパーコンピュータユーザである自然科学者コミュニティを大幅に超えた範囲や, Web 検索や SNS など日常生活により密着した領域に影響を及ぼす可能性が高い．Graph500 も Top500 と同じタイミングで発表されるランキングに用いられるため，今後その価値が正しく認識されるようになると，Top500 より多くの計算機のアーキテクチャに影響を与える潜在力を秘めている．

Graph500 はグラフ処理の中心的な処理の一つである BFS（幅優先探索）を主体とする．一見，疎行列の反復解法とは無縁の話のように見えるが，グラフ解析のデータ構造は疎行列であり，全てではないものの BFS を含む多くの重要なグラフ解析処理は GIM-V[10]（Generalized Iterative Matrix-Vector multiplication：一般化された反復的疎行列ベクトル積）で表現できる．大規模グラフ処理フレームワークである PEGASUS[10]，国内でも JST CREST で開発された ScaleGraph[11]は GIM-V モデルを基本概念として設計されている．つまり，疎行列の反復解法とグラフ処理は見掛けが似ていない親戚なのであり，GIM-V のメモリアクセスパターンや通信パターンは，疎行列の反復解法の中核をなす疎行列ベクトル積と基本的に同じである．ここが Top500 ではドライブできずに残された未来の計算機アーキテクチャの重要な進化すべきポイントでもある．よって，ビッグデータの流行を追い風に利用しながら Graph500 を大切に育成し，未来の計算機アーキテクチャの進化を促進することが，自然科学者やビジネスコミュニティを包含する多くの人々のためになる道と考える．

一方，アルゴリズム的に未成熟だった上に改変が自由な Graph500 は，アルゴリズムの競争の場としても使われてしまった．このため，優れたアルゴリズムを使ったエントリと，参照実装ベースの素朴なアルゴリズムを使ったエントリの間には，同等クラスのプラットフォームを用いていても桁違いの性能差が生じてしまい，プラットフォームのランキングとして Graph500（特にメモリアクセス性能がボトルネックとなる小規模なグラフのもの）は正しく機能して

いなかった．正しく機能しなければ Graph500 は Top500 を補うものとして安定成長せず，未来の計算機アーキテクチャの進化を促進することもできず，結果として後発のより良いベンチマークに取って代わられる．そこで，現在では参照実装の Pre-release 版として，現時点で優れたアルゴリズムとして広く認識されている(1)Agrawal らが SC10 で発表したビットマップ式の訪問管理方式[3]と，(2)Beamer らが SC12 で発表した Hybrid BFS[4]，を適用したソースコード[5]が公開されている. Graph500 の公式サイトはそれを用いることを推奨し，アルゴリズムレベルの実装の差を減らすよう促すようになった．

筆者らは上記のような状況を鑑み，Graph500 参照実装の Pre-release 版で何が起きようとしているのかを，アーキテクチャの観点から調査[6]を行った．その最初の取り組みとして，メモリアクセス回数や，メモリアクセスの局所性に大きな変化が生じていることを確認した．メモリアクセスがボトルネックとなる小規模なグラフにおいては，全く別のベンチマークを走らせて，同じ土俵で戦わせている状態にあったと言える．

よって，プラットフォームの評価指標として Graph500 を機能させるためには，少なくともメモリアクセス回数を統一する必要がある．何に統一すべきなのかについては，後続の章で考察する．

一方，Graph500 の派生的なランキングとして BFS 部分の電力あたりの性能を競う Green Graph500[12]ベンチマークも存在する．スーパーコンピュータの最大の制約要因になってきた消費電力の問題の解決の方向に，計算機アーキテクチャの進化を促進する働きがある．ただし，元の Graph500 が変質してしまえば，Green Graph500 の価値も低下してしまう．Green Graph500 は特に小規模なプラットフォームを中心に競われるので，メモリアクセスをどの状態で測定するかの重要性が高い．

2.4 HPCG ベンチマーク

Top500 や HPC Challenge ベンチマークの主催者であるテネシー大学の Dongarra 教授らは Top500 の 20 周年記念にあたる本年 6 月の ISC2013 において，Top500（Linpack）に代わる新たなランキング用評価指標として HPCG ベンチマークの構想[13][14]を発表した. 疎行列を係数行列とする連立一次方程式の Gauss Seidel 型の前処理を有する共役勾配法(Conjugate Gradient 法)による反復解法の性能を測定するものである．これにより Linpack 同様に 1 つの指標で明快な順位付けを行なった上で，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の乖離を防止できるようにする. さらに，演算偏重であった計算機アーキテクチャの進化の方向性を，より多くのユーザのニーズにあったバランスの取れた方向に導くことを目指している．

ここで，疎行列の非零要素配置はアプリケーションごとに異なり，動作時の挙動や最適化手法がそれによって大きく変わるため，どのような疎行列をベンチマークの対象にするかについては極めて重要な選択肢である．ところが，構想発表段階ではその点が未確定になっている．ランキングには用いられてこなかったものの従来から存在する NAS CG ベンチマークとは差別化したいことや，NAS CG や Graph500 のようなランダム性の高い疎行列は選択されない方向性が，Dongarra 教授らによる文書[13]には記載されている. ランダム性が高い疎行列を捨てるということは，連続場を扱うことが多い伝統的なスーパーコンピュータ市場しかカバーしないということを意味する．この考え方は High Performance Computing(HPC)向けのプラットフォームと Internet data center(IDC)向けのプラットフォームの要求仕様が近づいてきており，やがて統合に向かうであろうことが考慮されていないと言わざるを得ない.

筆者の意見としては，疎行列を扱う紛らわしいランキング用指標が乱立すべきではないと考える．よって，可能な限り HPCG はもう一つの疎行列処理系のベンチマークである Graph500 と平和的に融合（統合）させるべきと考える．具体的には現状の計算機にとって最も厳しい条件に近い性質を有する Graph500 用の疎行列の非零要素配置を全面的(または部分的に)用いることを提案する．これにより PageRank 実行時のような実効性能の下限を示し，それをうまく処理できる計算機への進化を促すことで，それより緩い条件の疎行列も効率的に処理できる強力なプラットフォームを提供できるようにすべきと考える．従来のスーパーコンピュータの伝統的な応用領域のみならず，Graph500 が想定するようなビジネス領域への適合性も担保することで，スーパーコンピュータと IDC の両市場向けのプラットフォームが共通の方向に進化し，やがて統合され，COTS 化によるコスト低減を促進すべきと考える．

3. Graph500 の新旧アルゴリズムとアーキテクチャの関係

ランキング用ベンチマークのボトルネックになっている部分の改善が進むように，計算機アーキテクチャは進化していくと期待される. 本章では Graph500 ベンチマークの新旧の参照実装のアルゴリズムが計算機のどこの部分に負荷をかけるものかという点と，それを踏まえた Graph500 運用に関する提言を述べる.

3.1 旧参照実装のアルゴリズム

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．[15] 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(BFS 木を記録する配列と兼用)への読み書きが, ランダムな不連続アクセスとなる．従来の参照実装ではこの配列のデータ型が 64bit 整数であり, 通常 64 バイトから 128 バイトのキャッシュライン単位のアクセスにおいては非常に効率が悪いメモリアクセスを多発する．節点数分の 8 バイトデータは現在の最大クラスのオンチップキャッシュ（例えば 32MB）から 400 万節点（SCALE22 程度）で溢れてしまうため，それ以上の SCALE では主記憶へのランダムアクセス実効バンド幅が単体性能を決める．

以上の性質から，Graph500 の旧参照実装のアルゴリズムは, Top500 が助長してきたキャッシュ中心のメモリシステムを有する従来の主流派の計算機アーキテクチャからの進化を促す力を持っていたと言える．

3.2 新参照実装のアルゴリズム

本稿執筆時点で Graph500 公式 web サイトの参照実装配布ページ上からもリンクされている Pre-release 版が別サイト(Graph 500 project on Gitorious[5])で，Agrawal らのビットマップ式の訪問管理方式[3]と Beamer らの Hybrid BFS[4]適用された参照実装の OpenMP ソースコードが公開されている．既に Pre-release 版が推奨版であることがアナウンスされており，今後，ビットマップ式訪問管理と Hybrid BFS が主流として用いられていく流れにあると考えられる．新参照実装のアルゴリズムの特徴は大別して以下の二つがある．

(1)ビットマップ式訪問管理

前節の従来手法と異なり，Pre-release 版の Graph500 においては訪問済みか否かを管理する配列をビットマップとして圧縮して管理する．このためそのアクセスにはビット演算などの余計な手間がかかるものの，外部メモリではなく大容量のオンチップキャッシュを持つ CPU を用いた場合は SCALE28 程度までオンチップキャッシュに載せたままで実行することができるため主記憶へのランダムアクセスが抑制される．SCALE28 の必要メモリ容量は大容量の主記憶を有するノードの主記憶容量と釣り合う程度であるので，この方式はビットマップ圧縮が可能なタイプのグラフ解析においては有効である．

この技法が適用された場合は，ラストレベルキャッシュのバンド幅がノード内性能を制約する．通常，このバンド幅は L1 キャッシュや L2 キャッシュよりは大幅に低く，主記憶の 2 から 4 倍程度に過ぎないので，キャッシュがヒットしている割には劇的な効果はない．よって，この技法が適用された新参照実装においてはノードに搭載される主記憶容量の増加に見合った比率でのオンチップキャッシュ容量の増加と，ラストレベルキャッシュのバンド幅を向上させるように促すにすぎないので，計算機アーキテクチャの進化はあまり劇的ではなく，物量が電力制約や集積度の制約の範囲で割り当てられるにすぎない．

ただし，PageRank[16]処理のようにランダムアクセスされる配列を実数で持たねばならないタイプの重要なグラフ解析処理も存在するため，そのような処理においてはビットマップ方式で主記憶へのランダムアクセスを逃れることができない．このため，ラストレベルキャッシュの容量やバンド幅を多少向上させたりしても，大き目の問題ではヒット率が低い状態で使わざるを得ない PageRank や疎行列の反復解法に対する効果は限定的である．このように，この技法を入れたもので Graph500 が運用されると計算機アーキテクチャへの十分な進化の促進が行なわれない．

(2)Hybrid BFS アルゴリズム

Beamer の Hybrid BFS アルゴリズムは，前節で説明した従来手法（Top down アプローチ）と，その逆方向に探索を行なう Bottom up アプローチを切り換えながら処理を進める点にある．Bottom up アプローチは訪問済みの節点が検出されるとそこで Break し，それ以上その節点と共通の節点と接続している他のエッジの先にある節点の探索を省略する．この動きによってアクセスされるエッジが特に次数が大きな節点において多く省略され，結果としてアクセス回数が大幅に減少し，局所性が高まり，高速化が達成される．筆者らの先行研究[6]では Bottom up アプローチのメモリアクセス回数は Top down アプローチより大幅に減少することが確認された．プロセッサからメモリシステムに対するアクセス要求の数が桁違いに減ったものと減っていないものとは，アーキテクチャの観点からは負荷をかけている場所が全く異なる別物のベンチマークと言える． これが混在してランキングされている現状はベンチマークの用を果たしていない. つまり，メモリアクセス回数を統一する必要がある．

さらに，この技法を入れたもので Graph500 が運用されるとアクセスの時間的局所性が良くなりすぎてしまい，メモリシステムへの負荷がかからず，ベンチマークとしては計算機アーキテクチャへの十分な進化の促進を行なえなくなる．

3.3 Graph500 の運用に対する提言

Graph500 は旧参照実装か新参照実装か，あるいは他の何らかの方式にメモリアクセス要求回数をそろえ，プラットフォームのランキング用に機能するように運用（ルール）を改善することが必要である．

その際，前節の(1)(2)ともに新参照実装のアルゴリズムで Graph500 のランキングが運用されると，限られた性質の良いグラフ処理の性能しか反映できない上に，計算機アーキテクチャへの進化の促進を行なう力が失われる．

アーキテクチャの進化への影響度の観点からも，そのラ

ンキングが意味を持つユーザ数への影響度の観点からも，Graph500 の運用においてはあえて(1)(2)の最適化を差し控え，少なくともアルゴリズムレベル（メモリアクセス要求回数レベル）ではメモリシステムに負荷がかかっている状態(Top500 とは差異が明確な Irregular application[17]に分類される状態)のベンチマークとして存在すべきと考える．

4. 評価

本章では，ビットマップ式訪問管理と Hybrid BFS を適用した graph500 参照実装の Pre release 版のメモリアクセス列のキャッシュメモリへの適合性の評価を行う．評価環境を表 1 に示す．

表 1　評価環境

Table 1　Evaluation environment.

CPU	Intel®Xeon®*CPU　W3565 @ 3.2GHz （コア数 4，ハードスレッド数 2/コア）
キャッシュ	L1 = 32KB, L2 = 256KB, L3 = 8MB
主記憶	DDR3×3 チャネル, 25.6GB/s, 6GB
OS distribution	Fedra17
OS カーネル	Linux カーネル 3.3.4
コンパイラ	gcc4.4
プロファイラ	PAPI 5.2.0
Graph500	Pre release 版(tar ボール生成日 6/20)

本評価において，ベンチマークを実行するプログラムによって，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．対称行列化を行う過程などで一行あたりの非零要素数はさらに 2 倍程度増える．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

Table 2　Experimented matrices

SCALE 値	*非零要素数*	*行数*
10	20,998	1,024
11	45,536	2,048
12	97,010	4,096
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
18	7,609,740	262,144
19	15,481,872	524,287
20	31,398,208	1,048,576
21	63,538,102	2,097,152

graph500 ではこの生成過程の時間は対象外になる．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．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．

graph500 参照実装の Pre release 版として公開されている Hybrid アルゴリズム版のほかに，2 つのアプローチの切り替え条件を常に片方になるように固定することによって，Top down アプローチ，Bottom up アプローチの測定も行なった．

4.1 配列のサイズ

SCALE 値とビットマップのサイズの関係を図 1 に示す．測定環境の L3 キャッシュ容量は 8MB(64Mbit)である．測定の範囲ではビットマップ全体が L3 キャッシュに収まってしまうと考えられる．つまり，訪問先の管理をビットマップで管理する方式を導入してしまうと，他にキャッシュ容量を消費する部分がないならば，SCALE28 程度までは 8 MB のキャッシュを搭載している汎用 CPU を用いて，ビットマップ全体がほぼ常時 L3 キャッシュの中に入ってしまうことになると考えられる．

一方，BFS 木の結果(親の節点番号)を格納する配列 bfs_tree への書込みは逐次アクセスとなり，プリフェッチが効いてキャッシュは概ね L1 ヒットするが，プリフェッチやリプレースに伴う主記憶へのアクセスがバンド幅を消費すると考えられる．このアクセスは何も工夫しないとキャッシュを汚染し，ビットマップをキャッシュから追い出してしまう可能性がある．bfs_tree が L3 容量(8MB)になるのは SCALE20 の時である．

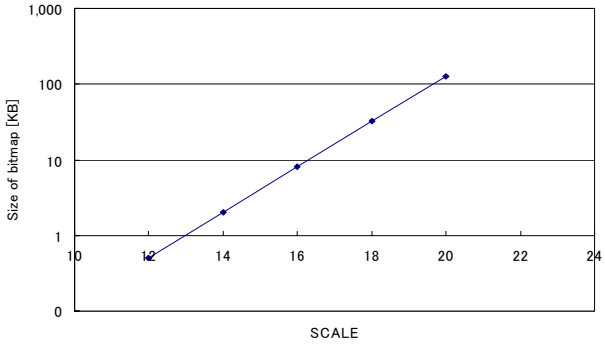

図 1　SCALE とビットマップのサイズの関係（Graph500 OpenMP 版）

Figure 1　Relation between SCALE and size of bitmap (graph500 OpenMP version).

4.2 キャッシュミス率

Graph500 は Kernel1(グラフ構築部)と Kernel2(BFS 木生成部)の二箇所の時間を測定して，ランキングを決めるための TEPS 値を計算する．この計測される箇所のうち，主たる処理を行なう Kernel2 部分の L1,L2 キャッシュミス率を Top down アプローチ，Bottom up アプローチおよび Hybrid アルゴリズムによる実行時に PAPI を用いて測定した． SCALE とミス率の関係を図 1(逐次実行時)および図 2(並列実行時)

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

に示す.

逐次実行の Hybrid アルゴリズムにおいては, L1 ミス率は約 40%で止まるのに対し，L2 ミスは測定範囲では SCALE にほぼ比例して多くなり SCALE18 で 70%を超える. SCALE18 では全アクセスの 60%で L1 が，12%で L2 が，28%で L3 がアクセスされることになる. 測定範囲では BFS 木の結果(親の節点番号)を格納する配列 bfs_tree が 8MB 以下なので L3 はほぼ常にヒットしていると考えられる.

前記 3 種類のアプローチでは Hybrid アルゴリズム，Bottom up アプローチ, Top down アプローチの順に L1 ミス率が低い．ただし，SCALE20 を超えると L1 ミス率のアプローチによる差はなくなってくる．一方，L2 ミス率は Bottom up が最も低い.

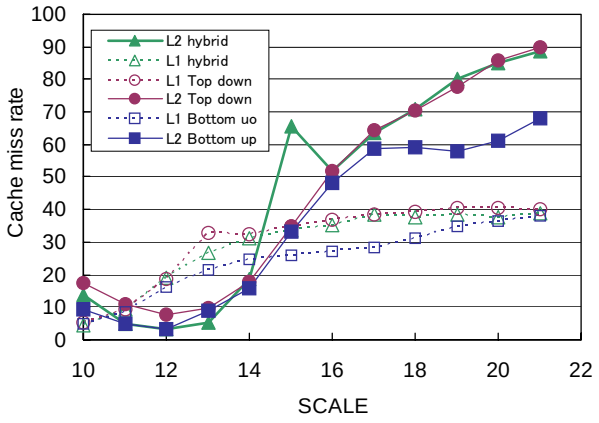

図 2　SCALE と kernel2 におけるミス率の関係（Graph500 OpenMP 版逐次実行時）

Figure 2　Relation between SCALE and cache miss rate for kernel 2 (graph500 OpenMP version, serial execution).

OpenMP 並列実行時には 16 以上の SCALE で L2 キャッシュのヒット率が 50%より低くなる．よって，16 以上の SCALE では L3 キャッシュのバンド幅によって性能が決まると考えられる．L2 キャッシュは 256KB(2Mbit)であるため，その全てを訪問管理用のビットマップ用に用いることができれば SCALE21 まで L2 キャッシュから溢れないことになるが，実際には BFS 木の結果(親の節点番号)を格納する配列 bfs_tree などの他の変数のアクセスにも L2 キャッシュは利用されるため，それより小さい SCALE で溢れが生じたと考えられる.

前記 3 種類のアプローチでは Hybrid アルゴリズム，Bottom up アプローチ, Top down アプローチの順に L1 ミス率が低い．ただし，SCALE20 を超えると L1 ミス率のアプローチによる差はなくなってくる．一方，L2 ミス率は Bottom up が最も低い.

L1 ミス率は逐次の場合よりも若干低くなる．これは，コアで分担する節点が分散されるため，コア数分 L1 容量が増えた状態になるためであると考えられる．一方，L2 ミスについては逐次の場合より並列の場合の方が多くなり，約 Hybrid では 84%，Top down では 89%まで L2 ミス率は上昇する. L1 で拾えないような時間的局所性の低いアドレスへのアクセスは 256KB 程度の L2 キャッシュでは 4 コアで並列化したくらいでは拾いきれず, 配列全体を格納できる L3 くらいの容量がないと効果が薄くなるためであると考えられる.

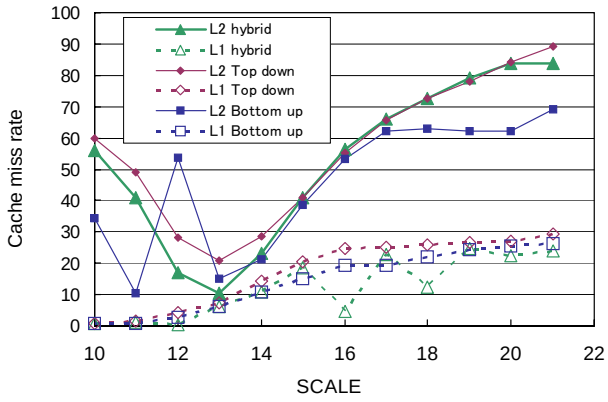

図 3　SCALE と kernel 2 におけるミス率の関係（Graph500 OpenMP 版並列実行時）

Figure 3　Relation between SCALE and cache miss rate for kernel 2 (graph500 OpenMP version, parallel execution).

4.3 GTEPS 値

測定環境で OpenMP 版 graph500（Hybrid アルゴリズム）の逐次実行時および並列実行時の SCALE 値と GTEPS 値の関係を図 4 に示す.

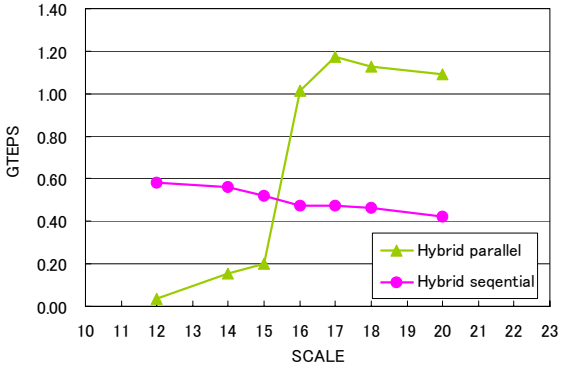

図 4　SCALE と GTEPS 値の関係（Graph500 OpenMP 版）

Figure 4　Relation between SCALE and GTEPS (graph500 OpenMP version).

結果として, SCALE が小さい時は逐次実行の方が高速であるが SCALE16 を境に並列実行の GTEPS 値が上回るようになる．ただし 8 スレッドで実行されている割にはあまり高速化しない．SCALE16 で急激に加速するが SCALE17 付近で並列実行の伸びが飽和する. これは前節の実験から L2 キャッシュからの溢れが SCALE16 から始まり，アクセスするようになった L3 キャッシュのスループットが SCALE17 で限界に達し, 実質 2 コア分程度の要求にしか応

えられないためボトルネックになっていると考えられる．

3 つのアプローチに対する OpenMP 並列実行時の SCALE と GTEPS の関係を図 4 に示す．

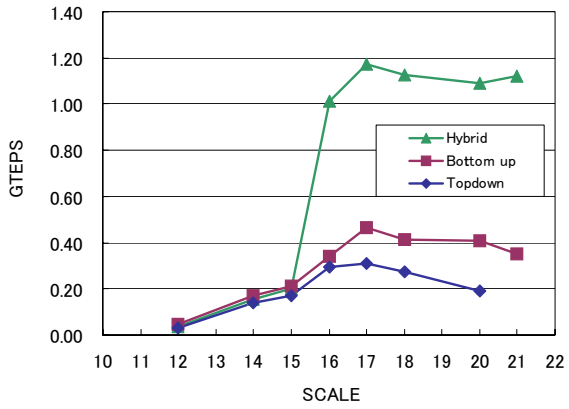

図 5　アプローチと SCALE と GTEPS 値の関係（Graph500 OpenMP 版並列実行）

Figure 5　Relation between Approach, SCALE and GTEPS (graph500 OpenMP version, parallel execution).

4.4 Hybrid 化に伴うアクセス回数の変化

SCALE 値と Top down アプローチおよび Bottom up アプローチによるビットマップアクセス回数比の関係を図 6 に示す．

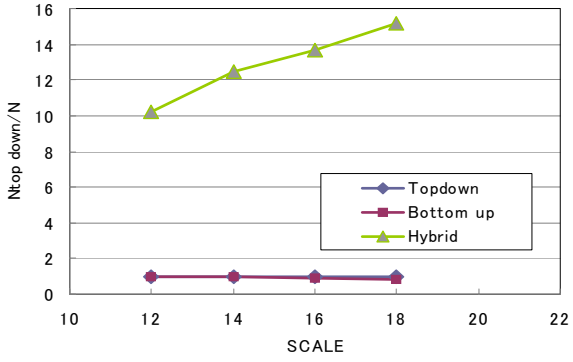

図 6　SCALE とビットマップアクセス回数比の関係（Graph500 OpenMP 版）

Figure 6　Relation between SCALE and ratio of the number of bitmap accesses (graph500 OpenMP version).

Bottom up アプローチでは Frontier を発見するとアクセスループを中断して抜けることにより，それ以降に予定されていたアクセスを省略する．このため，従来の Top down アプローチと比較して Bottom up アプローチではメモリアクセス回数が約 1/10.2 から 1/15.2 に減少している．大きい SCALE のものほどこの差は拡大する．

これほど大きな差がある状態のエントリを同じ土俵で比較していたら，そのランキングはプラットフォームの差を現すことはなく，ランキング用ベンチマークとして正しく機能しないのは明らかである．

この変化は「BFS を実行する」という問題設定(現状のルール)からは逸脱していないものの，疎行列ベクトル積と同等だったメモリアクセスに関する負荷を，それとは全く異なる軽い負荷の問題に置き換えてしまったと言える．

4.5 Hybrid 化に伴う空間的局所性の変化

Top down アプローチ, Bottom up アプローチおよび Hybrid アルゴリズムによるビットマップアクセスの空間的局所性指標値(ライン内の有効データ数)[18][19]の関係を図 7 に示す．

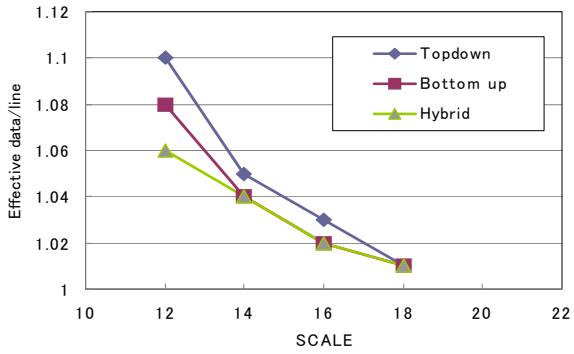

図 7　SCALE とライン内の有効データ数の関係（Graph500 OpenMP 版逐次実行）

Figure 7　Relation between SCALE and average number of effective data per line (graph500 OpenMP version serial execution).

3 通りとも SCALE の増加とともに空間的局所性が単調減少し，絶対値的にもライン内に 1 個程度まで減少する．Hybrid アルゴリズムも含めて，いずれの場合も L3 キャッシュのバンド幅を大幅に浪費することが明白である． Top down アプローチ，Bottom up アプローチ，Hybrid アルゴリズムの順にラインあたりの有効データ率が高いが，3 者の差はそれほど大きくない．極端に低かったものがさらに少し悪化した状態にある．

L3 キャッシュに全部載ってしまうビットマップの導入により，測定範囲の SCALE ではビットマップへの外部メモリアクセスがほとんど排除されているために，Top down アプローチと他の 2 つの差は顕在化しないと考えられる．

4.6 Kernel1,2 の時間比率

SCALE 値と Hybrid アルゴリズムによる kernel1 と kernel2 の時間比率の関係を図 8 に示す．比率が一定にならず，SCALE が大きくなるほど kernel1 の比率が増加する．SCALE21 では kernel1 は kernel2 の約 2 倍の時間を消費している．この比率はできるだけ一定になるようにすべきである．HPCG ベンチマークの場合，前処理時間は反復回数で正規化される．Graph500 においても類似した調整機構が導入されるべきと考える．

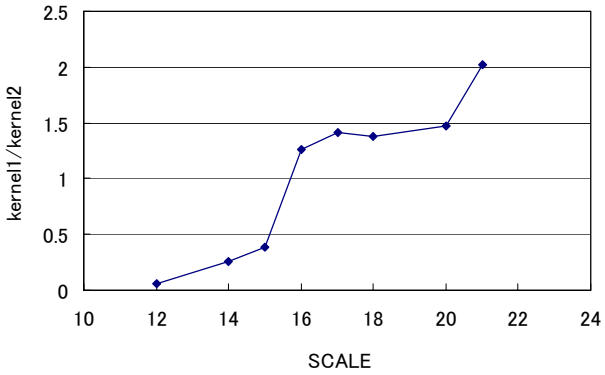

図 8　Hybrid アルゴリズムによる kernel1 と kernel2 の時間比率の関係（Graph500 OpenMP 版逐次実行）

Figure 8　Relation between SCALE and ratio of time between kernel1 and 2 (Graph500 OpenMP version serial execution).

5. 関連研究

近年，ノード内の処理に関するアルゴリズムとして，Agrawal らが SC10 で発表したビットマップ式の訪問管理方式[3]や，Beamer らが SC12 で発表した Hybrid BFS [4]が主流になってきている. 本稿執筆時点で Graph500 公式 web サイトの参照実装配布ページ上からもリンクされている Pre-release 版が別サイト(Graph 500 project on Gitorious)で，これらの 2 つのアルゴリズムが適用された参照実装の OpenMP ソースコードが公開されている．既に Pre-release 版が推奨版であることがアナウンスされており，今後，Hybrid アルゴリズムが主流として用いられていく流れにあると考えられる.

国内では安井らが Hybrid BFS の改良版を開発し，更なる高速化を達成し，2013 年 6 月の Green Graph500 リスト[12]の小規模カテゴリで上位をほぼ独占している．しかし，筆者が知る限りにおいてはソースコードが未公開であり，Beamer のように詳細が論文化されていないため現時点では主流にはなっていない．安井が共著になっている岩渕らの論文[20]には安井の手法が若干記載されている．そこには，グラフデータを Top down アプローチ用と Bottom up アプローチ用に別々に CSR 形式で持った上で, 分割方法を工夫することで，並列実行時の訪問済フラグ書込み時のアトミックオペレーションを排除しているとされる．平櫛らの GPU クラスタへの実装[21]においても，連続した頂点を同じスレッドが処理するような分割によりアトミックオペレーションを排除している.

Suzumura[22]らは Graph500 課題の参照実装の解析を様々な観点から行っている．古い参照実装に関するものであるとともに，キャッシュに関する解析は行なわれていない．上野[15]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 Hybrid アルゴリズムでもなく，かつ，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．田邊[23][24]らは Graph500 の課題疎行列の空間的局所性に関して，上記の Vertex sorting も含めた評価を行っている．ただし，評価対象が従来の参照実装に基づいている.

Graph500 が本来位置づけられていた Irregular application[17]をアーキテクチャレベルで解決しようというアプローチは超並列マルチスレッド共有メモリアーキテクチャの Cray MTA-2[25] ，Cray XMT[26]やそれらを用いた Bader らの研究[27][28]，Cray XMT-2 ベースの Cray/YarcData Urica[29]，FPGA と CPU を併用する Convey 社 HC-1[30][31]，HC-2[32]，MX[33]，Inteligent メモリを志向するアプローチとしては Graph500 の作者である Micron 社の Murphy らの Distributed PIM[34]，COTS CPU とシステムインテグレート可能なモジュールレベルでの Inteligent メモリである Tanabe らの DIMMnet[35]，Memory accelerator[36]，Hybrid Memory Cube として実現される Memory accelerator[37]などがある．Graph500 が本来の意義を失わなければ, これらのような先進的な試みが促進され，それらの中から将来の HPC と IDC を統合するプラットフォームに貢献できる革新的な技術の実用化が促進されて行くと考えられる.

6. おわりに

本研究ではビットマップ式の訪問管理方式や Hybrid BFS アルゴリズムが適用された Pre Release 版 Graph500 コードを用い，Graph500 におけるメモリアクセス列のキャッシュメモリへの適合性を，キャッシュヒット率や局所性の観点から解析した.

その結果，従来法よりもメモリアクセス回数が劇的に変化してしまっており，メモリアクセス回数や時間的局所性がグラフサイズに比例して楽な状況になる傾向を確認した.

一方，ビットマップ式の訪問管理方式や Hybrid BFS アルゴリズムの導入は「BFS を実行する」という問題設定(現状のルール)からは逸脱していないものの, 疎行列ベクトル積と同等だったメモリアクセスに関する負荷を，それとは全く異なる軽い負荷の問題に置き換えてしまった．BFS はグラフ解析においては一部であり，BFS 固有の高速化だけでは，それが通用しない PageRank などの疎行列ベクトル積と同等なメモリアクセス要求を有する重いグラフ解析アプリケーションを高速実行するプラットフォームへの進化を促進できない.

元来の Graph500 は Top500 ではドライブできずに残された未来のプラットフォームの進化を促進するものであった. よって，それを阻害する大幅な質的変化のベンチマークへの正式導入は慎重であるべきと考える．そのような意味で Graph500 は現在重要な岐路に差し掛かっており，これを適

切に乗り越えないと，HPC 市場にしか強い訴求力を持たない後発の HPCG ベンチマークにその座を譲る結果になる可能性が高いと考える．

今後の課題としては，アトミックオペレーションを排除可能な割り当て方式を導入した場合の評価，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) V. Agarwal, F. Petrini, D. Pasetto and D. Bader: "Scalable Graph Exploration on Multicore Processors", SC10 (2010).
4) S. Beamer, K. Asanovic, D. Patterson: “Direction-Optimizing Breadth-First Search”, SC12 (2012).
5) “Gitorious : Working space for the graph500 reference implementation”, https://www.gitorious.org/graph500.
6) 田邊，冨森，高田，城 :“Graph500 の Hybrid 解法に内在する局所性”，情報処理学会 HPC 研究会, Vol.2013-HPC-140 (2013).
7) Standard Performance Evaluation Corporation: "SPEC Benchmarks", http://www.spec.org/benchmarks.html.
8) K. Goto, K. Milfeld: “GotoBLAS2”, http://www.tacc.utexas.edu/tacc-projects/gotoblas2/.
9) Green500 : http://www.green500.org/.
10) 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).
11) ScaleGraph : https://sites.google.com/site/scalegraph/
12) Green Graph500 : http://green.graph500.org/.
13) University of Tennessee: “Professor Jack Dongarra Announces New Supercomputer Benchmark” (2013). http://www.utk.edu/tntoday/2013/07/10/professor-jack-dongarra-announces-supercomputer-benchmark/
14) M. A. Heroux and J. Dongarra: “Toward a New Metric for Ranking High Performance Computing Systems”, Sandia National Lab. Report, SAND2013-4744 (2013). http://www.netlib.org/utk/people/JackDongarra/PAPERS/HPCG-Benchmark-utk.pdf
15) 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).
16) 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）.
17) Workshop on Irregular applications: architectures and algorithm (IAAA) in conjunction with SC : http://cass-mt.pnnl.gov/irregularworkshop.aspx
18) 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).
19) 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) (2013).
20) 岩渕，佐藤，安井，藤澤，松岡 :“不揮発性メモリを用いた Graph500 ベンチマークの大規模実行へ向けた予備評価”，情報処理学会 HPC 研究会, Vol.2013-HPC-138 (2013).
21) 平櫛，高橋 :“GPU クラスタにおける幅優先探索の高速化”，情報処理学会 HPC 研究会, Vol.2013-HPC-139 (2013).
22) T. Suzumura, K. Ueno, H. Sato, K. Fujisawa and S. Matsuoka: "Performance Evaluation of Graph500 on Large-Scale Distributed Environment", IEEE IISWC 2011 (2011).
23) 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 HPC 研究会, Vol.2013-HPC-138 (2013).
24) 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”，先進的計算基盤システムシンポジウム（SACSIS 2013）(2013).
25) W. Anderson, R. Rosenberg, M. Lanzagorta: "Experiences using the Cray Multi-Threaded Architecture (MTA-2)," Proceedings of User Group Conference 2003, pp.378-383 (2003).
26) D. Mizell: “Introducing Cray XMT” (2010) http://wwwjp.cray.com/downloads/XMT-Presentation.pdf
27) D. A. Bader and K. Madduri: "Designing Multithreaded Algorithms for Breadth-First Search and st-connectivity on the Cray MTA-2", ICPP 2006, pp.523-530 (2006).
28) D. Ediger and David A. Bader : "Investigating Graph Algorithms in the BSP Model on the Cray XMT", Workshop on Multithreaded Architectures and Applications (MTAAP 2013) in conjunction with IPDPS’13 (2013).
29) Cray Inc./YarcData Inc.: “YarcData Urika Big Data Graph Appliance”, http://www.cray.com/Products/BigData/uRiKA.aspx
30) T. M. Brewer: "Instruction Set Innovations for the Convey HC-1 Computer," Micro, IEEE , vol.30, no.2, pp.70-79 (2010).
31) K. K. Nagar and J. D. Bakos : "A Sparse Matrix Personality for the Convey HC-1", 19th International Symposium on Field-Programmable Custom Computing Machines (FCCM) (2011).
32) Convey Computer corp. : Hybrid-core: The “Big Data” Computing Architecture” (2012) http://www.conveycomputer.com/files/9013/5515/7619/The_Big_Data_Computing_Architecture-Graph500.pdf
33) Convey Computer corp. : “Big Data Computer Architecture: The Convey MX Series” (2012) http://www.conveycomputer.com/files/6713/5266/3042/CONV-12-043.1MXdatachureWeb.pdf
34) R. C. Murphy, P. M. Kogge, A. Rodrigues : "The Characterization of Data Intensive Memory Workloads on Distributed PIM Systems", Second International Workshop of Intelligent Memory Systems (IMS 2000), LNCS vol.2107, pp.85-103 (2001).
35) 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).
36) 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).
37) N. Tanabe, J. Kogou, S. Tomimori, M. Takata and K. Joe: “Future Irregular Computing with Memory Accelerators”, FUTURE COMPUTING2013, pp.74-80 (2013). http://www.thinkmind.org/download.php?articleid=future_computing_2013_3_40_30074