Vol.2014-HPC-143 No.18
2014/3/3

不揮発性メモリを用いた Hybrid-BFS アルゴリズムの最適化と性能解析

岩渕 圭太[1,2,a)]　佐藤 仁[1,2]　溝手 竜[1,2]　安井 雄一郎[2,3]　藤澤 克樹[2,3]　松岡 聡[1,2]

概要：近年，科学技術計算におけるデータ量は急速に増大しており，膨大な量のデータを処理するには，将来のスーパーコンピュータでは消費電力や価格面でのコスト，ローカルに取り付けられた不揮発性メモリに注目が集まっている．しかし，不揮発性メモリの活用方法や性能への影響は明らかでない．一方で，大規模グラフ処理に対する需要が様々な分野から高まっており，データインテンシブ問題の代表例であるグラフ処理問題を対象とし，不揮発性メモリを活用することで，DRAM 容量を増設することなくより大規模な実行が行えることを目指している．具体的にはグラフ処理において最も重要なアルゴリズムの 1 つである，BFS を高速に行う Hybrid-BFS アルゴリズムに対して，アクセスパターンの詳細な分析から性能に影響の少ないデータの退避や I/O デバイスの特性を考慮した I/O を行うことで，性能低下を抑えながら省メモリ化を行い，DRAM のみの場合に比べ 4 倍のサイズのグラフ問題を実行可能とし，性能低下率を 6.9%まで抑えることができた．また，不揮発性メモリを活用することで，DRAM 容量やシステムの規模を縮小することができ，大規模グラフ処理における電力効率を競う，Green Graph500 の 2013 年 11 月の BigData カテゴリに対して，現在の最高値より 1.8 倍の電力効率性能を達成できた．

1. はじめに

現在，データ量が増大するビッグデータと言われる現象がおきており，デジタルデータの量は，今後，2020 年までに約 40 ゼタバイト（2012 年の約 14 倍）になるとの報告もある [1]．また，Internet of Things 発展による大量のセンサーデータから送信される情報量は膨大なものとなる．さらに，これらのデータはそのサイズだけが問題になるのではなく，非構造なデータがリアルタイムにシステムに流れ込み，これらのデータに対する演算は非常に大容量，且つ，高いメモリ性能が要求される．その結果，大量の DRAM が必要となり，非常に価格の面でのコストが高く，また，消費電力も増大してしまうため望ましくない．

一方で, 近年, フラッシュデバイスなどのような, DRAM と比較するとアクセスレイテンシやスループットなどの性能面で劣るものの，容量あたりの価格，消費電力の点で優れたデバイスが登場し，普及している．このようなフラッシュデバイスを DRAM に対して補助的に利用することで，メモリを多階層化し，単一の計算ノード上の DRAM には収まらない容量のグラフを実行できる可能性があるものの，その具体的手法やどの程度の性能低下が起きるのかなどの定量的な指標は明らかではない．

また，近年，SNS 解析，道路ネットワークの経路探索，スマートグリッド，創薬，遺伝子解析等の応用で，大規模グラフが出現しており，数百万頂点〜数兆頂点，数億エッジ〜数百兆エッジからなる超大規模なグラフに対する高速処理が求められている．例えば，現存する SNS における交友関係は 8 億頂点，1000 億エッジからなる大規模グラフによって表現される．一般に，グラフ処理は，参照の局所性が低く，また，メモリアクセスは非構造なものとなるため，全てのデータを DRAM へロードして実行しなければ，妥当な性能で実行することは難しい．

そこで，データインテンシブ問題の代表例であるグラフ処理を対象とし，DRAM 容量を超える環境でも性能低下を最小限に抑えた大規模実行を目指すことで，不揮発性メモリをデータインテンシィブ問題に使用した場合の性能への影響と，具体的な実現手法を明らかにし，将来の HPC システムにおける I/O のあり方を考察することを目的とする．

これまで著者らは Graph500 ベンチマークにおいて，Hybrid-BFS アルゴリズムに対して不揮発性メモリを補助的に利用した場合に，どのようなアクセスパターンがあり，それらが I/O にどのような影響を与えているのかを調べる

[1] 東京工業大学
Tokyo Institute of Technology
[2] JST CREST
[3] 中央大学
Chuo University
[a)] iwabuchi.k.ab@m.titech.ac.jp

と共に，アルゴリズムの特性を考慮しアクセス数の少ないデータを不揮発性メモリに退避することで，どの程度性能の低下が起きるか予備評価を行ってきた．その結果，一部データを不揮発性メモリに退避することで DRAM 用量が半分の環境において性能低下を 47.1%まで抑えることができ，また，次数の小さい頂点が大量にあることによる非効率な I/O が，性能低下の要因となっていることを確認した．

本論文では，細粒度な I/O が頻発するグラフ処理において，アルゴリズムの特性を分析しデータ配置を行うことで，より DRAM 容量を削減できる手法を提案すると共に，一般的な SSD を並列的に接続することで高い IOPS を実現できる I/O アーキテクチャを用いることで，シングルノード上で 6.9%の性能低下で DRAM のみの場合に比べて 4 倍のサイズの問題を実行できた．また，D データインテンシブベンチマークの電力の効率性を競う，Green Graph500 の Big Data カテゴリにおいて現世界一の 1.8 倍の電力効率を達成することができた．

よって，シングルノードにて不揮発性メモリをメインメモリの延長として使用することの有効性を確認すると共に，並列的な I/O（API，デバイス構成共に）を行うことでフラッシュメモリを有効活用でき，さらに，不揮発性メモリをメインメモリの延長として使用する場合には，ユーザによる多階層メモリへの明示的なデータの配置が重要であることを示す．

2. Graph 500 ベンチマーク

Graph 500 ベンチマークとはグラフ探索の性能を利用したベンチマークである．ここではまず，Graph 500 ベンチマークで使用される幅優先探索 (BFS) について説明し，次に Graph 500 ベンチマークの概要を示し，そして最後に，我々が対象とする BFS アルゴリズムについて述べる．

2.1 幅優先探索（BFS : Breadth First Search）

幅優先探索 (BFS) とは，重み無しグラフの探索において，ある頂点から他の頂点への最短経路を求める手法の一つである．探索は始点となる頂点からレベル（深さ）単位でステップを進めながら行い，あるレベルの頂点が全て探索されてから次のレベルの探索を行う．まず始点となる頂点（level 0）から隣接する全ての頂点（level 1）を探索する．次に，level 1 の頂点を始点とし，同様に，隣接する頂点（level 2）を全て探索する．レベル $i+1$ の頂点は，レベル i の頂点に隣接し，且つ，これまでに訪問されていない全ての頂点である．探索の始点となる頂点をレベル 0 とすれば始点に隣接する頂点の集合はレベル 1 となり，レベル i で訪問した頂点は始点から i ホップで隣接していることになる．

2.2 Graph 500 ベンチマークの概要

Graph 500 ベンチマークは，データインテンシブなワークロードに対するスーパーコンピュータの性能ランキングである．従来の性能ランキングである Top500 は計算インテンシブなワークロードに対して競われてきたのに対し，Graph 500 ではデータインテンシブなワークロードの一つであるグラフ探索の性能が競われる．具体的には，クロネッカーグラフと呼ばれる，実グラフと似た性質を持つグラフを生成し，そのグラフに対して，ランダムに選ばれた探索開始点から繰り返し BFS を行い，始点から他の全頂点までの最短経路を求め，最後に結果の検証を行う．Graph 500 で使用される，クロネッカーグラフには以下のような性質がある．

(1) スケールフリー性 (次数分布のべき乗則)
　各頂点の保持するエッジ数（次数）の分布がべき乗則に従っている性質である．一部の頂点が他の多くの頂点とエッジで繋がっており，大きな次数を持っている一方で，他の多くの頂点はごくわずかな頂点としか繋がっておらず，次数は小さい．この特性はロード・インバランスを引き起こし，分散メモリ環境での性能劣化を引き起こす原因となり得る．

(2) スモールワールド性
　任意の頂点間の距離が頂点数が多くても極めて小さい性質である．例えば Graph 500 ベンチマークでは頂点数は数億個であったとしても任意の頂点間は 6 ホップ程度で探索できる．

次にベンチマークの流れについて説明する．ベンチマークは以下 4 つのステップからなる．

(1) エッジリストの生成
　無向グラフのエッジリストを生成する．グラフサイズは頂点数が 2^{SCALE}，エッジ数は頂点数*$Edgefactor$ で表される．

(2) データ構造の変換
　生成されたエッジリストを BFS の実行に適切なデータ構造に変換する．

(3) BFS の実行
　変換されたグラフデータに対して BFS を実行する．この時の実行時間とグラフが持つ全エッジ数から性能が決まる．性能は単位時間に処理されたエッジ数である TEPS（Traversed Edges Per Second）値として表される．

(4) 探索結果の検証
　BFS によって求めた BFS 木に対して探索結果の検証を行う．

Graph 500 ベンチマークでは，ベンチマークは探索開始点としてランダムに 64 個の点が選ばれ BFS を行う．よって，BFS の実行と探索結果の検証の組み合わせが 64 回行

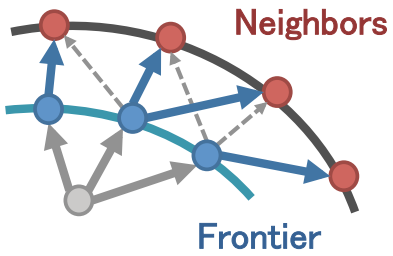

図 1　Top-down アプローチの概要と擬似コード

われ，64 回の BFS における TEPS 値の中央値がランキングに使用される．

3. Hybrid-BFS アルゴリズム

我々の提案手法は，Hybrid-BFS[2] アルゴリズムを対象とする．Hybrid-BFS アルゴリズムは，探索する頂点数を削減するために，従来の Top-down で頂点を探索するアプローチに加えて，Bottom-up で頂点を探索するアプローチを補助的に利用して，この両者のアプローチをハイブリッドに切り替えながら，BFS を行う手法である．この手法は，現在の Graph500 ランキング上のカスタマイズされた実装において主流となっているものである．ここでは，Hybrid-BFS のアルゴリズムの詳細について述べる．

3.1 Top-down アプローチ

Top-down アプローチは，従来よりよく知られた古典的な BFS のアルゴリズムであり，Graph 500 ベンチマークのリファレンス実装などでも使用されている手法である [3]．**図 1** にその概要と擬似コードを記す．各レベル毎の探索の際に，現在訪問している頂点の集合を frontier とし，frontier に属している各頂点に対しそれぞれ隣接している全ての頂点を neighbors とする．このとき neighbors をチェックし，未訪問であれば訪問済とし、さらにその頂点を次のレベルの探索での frontier に追加する．訪問済であれば特に操作は行わない．

Top-down アプローチの欠点としては，frontier の数が多い場合に冗長な探索が増大していまい探索効率が悪くなってしまう，という点が挙げられる．これは，Top-down アプローチでは frontier に所属する頂点は自分が隣接する全ての頂点に対して訪問済みかどうかの探索を行う．しかし，このアプローチでは，既に他の頂点によって訪問済みの頂点に対しても再度探索を行ってしまうため，冗長が探索が増えてしまう．また，ある頂点に対して訪問済みとする操作を行う際にはアトミックな操作が必要になり，性能が低下してしまうという問題もある．

3.2 Bottom-up アプローチ

Bottom-up アプローチは，Top-down アプローチとは逆の方向に探索を行う．**図 2** にその概要と擬似コードを記す．Top-down アプローチが訪問済みの頂点を始点として，隣接している未訪問の頂点を目指し探索を行うが，Bottom-up アプローチは全ての未訪問の頂点を始点とし，隣接する頂点の中に 1 つでも frontier に属する頂点があればその未訪問頂点を訪問済とし，親の頂点をエッジの存在した frontier に属する頂点とする．この時，frontier に所属する頂点の一覧はビットマップを使用して保存されていると効率が良い．Bottom-up アプローチでは，未訪問の頂点が 1 つでも frontier に属する頂点へのエッジを見つければその時点で探索は終了するため冗長な探索を削減することができ，性能の向上が期待できる．

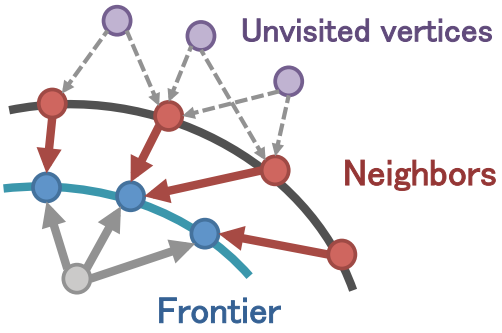

図 2　Bottom-up アプローチの概要と擬似コード

3.3 アプローチの切替

Top-down アプローチと Bottom-up アプローチを探索の状況によって切り替えるのが Hybrid アルゴリズムである．具体的には frontier の数や探索されるエッジ数を用いてヒューリスティックなモデルにてアプローチの切り替えを行っている．アプローチの切り替えについては，Beamer らが 2012 年にモデルの提案と詳細な分析を行っており，詳細はそちらを確認されたい [4]

3.4 データ書き出しの局所性を高めたグラフ分割

著者の 1 人である安井らは，現在，Hybrid-BFS アルゴリズムを DRAM のみを使用した実装を行っており [5]，2013 年 6 月の Graph 500 リストにおいて 1 台の計算ノード上（4-way Intel Xeon E5-4640）で，11.1GTEPS を記録している．この実装では，NUMA ノードを考慮し，Top-down アプローチと Bottom-up アプローチにおいてそれぞれデータ書き出しの局所性を高めた探索を行うため，探索用のグラフデータについて．Top-down アプローチと Bottom-up アプローチ各々に関して探索用のグラフデータを CSR の形式で持っている．

Top-down アプローチでは，frontier にある頂点を始点とし，隣接する頂点を終点として探索を試みる．通常，各頂点は隣接する頂点の情報を全て持つが，今回，対象としている実装では，全ての探索対象の頂点は，NUMA ノード

毎に分割されて管理され，例えば，ある NUMA ノードに属する始点の頂点は，同じ NUMA ノードに属する終点の頂点のみを探索し，終点の頂点が他の NUMA ノードに属するような場合は，探索を他の NUMA ノードに属する頂点に委ねる．このようなデータ配置を行うことによって，ある頂点を訪問済みとする書き込みを確実にローカルの NUMA ノードに行うことができる．

一方で，Bottom-up アプローチに使用する backward graph については，探索の方向が逆になり，未訪問の頂点から frontier に含まれる頂点を探す．そのため，backward graph では，各 NUMA ノードは担当する終点の頂点を分割して持ち，始点の候補となる頂点の情報を各 NUMA ノードで持つことにより，frontier へ探索を行った場合でもある頂点を訪問済みとする書き込みを確実にローカルの NUMA ノードに行うことができる．

我々は，このデータ書き出しの局所性を高めた実装を元に不揮発性メモリを活用する機構を拡張した．今回対象とする Hybrid-BFS アルゴリズムを用いた実装で使用する具体的なデータを示す．

(1) エッジリスト
生成されたクロネッカーグラフのエッジ一覧をタプル形式で保持している．

(2) BFS 探索用グラフデータ
Top-down アプローチと Bottom-up アプローチの探索用に，それぞれエッジリストを CSR 形式に変換して持つ．以下，Top-down アプローチ用グラフデータを forward graph，Bottom-up アプローチ用のグラフデータを backward graph とする．

(3) BFS ステータスデータ
探索結果（BFS 木）や探索に用いるキューやビットマップなどを持つ．

Graph500 ベンチマークの各ステップにおいて，グラフ生成ではエッジリスト，BFS 探索用グラフデータ構築ではエッジリストと BFS 探索のためのグラフデータ，Hybrid-BFS アルゴリズムを用いた探索時は探索用グラフデータと BFS ステータスデータ，そして，探索した結果の検証時にはエッジリストと BFS ステータスデータを使用する．よって，既存実装ではこれらのデータを全て DRAM 上に配置しながら，Graph500 ベンチマークの実行を行っている．

4. Hybrid-BFS アルゴリズムと I/O の特性を考慮した省メモリ化

ここでは，省メモリ化を行うための手法について提案する．BFS においてグラフデータを退避する既存手法として，グラフデータを全て不揮発性メモリに退避する手法があるが，これでは性能が大きく低下してしまう．不揮発性メモリを使用して大規模グラフに対する BFS を実行する方法としては，BFS ステータスデータのみを DRAM 上に乗せグラフデータは不揮発性メモリに載せる手法がある [6]．BFS ステータスデータを DRAM 上に固定する理由としては，BFS ステータスデータは探索中に使用するキューやビットマップであり，探索中に頻繁にアクセスが行われ探索の性能に大きな影響を及ぼすためである．

そこで我々の提案手法は，BFS ステータスデータを DRAM 上に固定すると共に，全てのグラフデータを退避するのではなく，Hybrid-BFS アルゴリズムの特性を考慮し，探索用グラフデータの一部を DRAM 上に残す方針を取る．

4.1 不揮発性メモリを用いた場合のデータ配置方針

4.1.1 Dual モデル

著者らはこれまで不揮発性メモリを用いた場合のデータ配置方針を提案している．まずそのモデルについて述べる．3.4 で述べたように，我々が元にしている実装では NUMA ノードに最適化したメモリアクセスを行うために，それぞれのアプローチで，別のグラフを持っている．しかし，Hybrid-BFS アルゴリズムでは Bottom-up アプローチが性能の大半を占めているため，Bottom-up アプローチにて使用する backward graph は DRAM 上に固定すると共に，性能への影響が少ない Top-down にて使用する forward graph を不揮発性メモリへ退避する手法を提案した．

その結果，DRAM 容量が半分の環境においても不揮発性メモリとして SSD を使用することで，性能の低下率を 47.1%まで抑えることができた．

このモデルではアクセスパターンの分析が容易であり，且つ，両アプローチとも NUMA に対応するため BFS Status へのアクセス効率は高まり高速な実行が見込まれる．しかし，退避できるデータは，元の実装の半分のみであり，DRAM と NVM の性能差，容量差を考慮した柔軟な容量の変更を行うことはできないという問題がある．そこで，さらなる大規模実行に向けて新たな手法を検討する．

4.1.2 Single モデル

これまで我々はアクセスパターンの分析のために 2 つグラフを使用したモデル（Dual モデル）について評価を行ってきたが．Hybrid-BFS ではどちらか一方のアプローチについて，NUMA ノードへの対応を行わないことで 1 つのグラフデータを共有できるようになる．そこで，新しく両アプローチで性能に大きな影響がある backward graph のみを使用し，さらに，backward graph についても一部のみを DRAM 上に保持し残りの部分は不揮発性メモリに退避することで省メモリ化を行う手法について検討する．

4.2 Hybrid-BFS アルゴリズムの特性と I/O における課題

4.2.1 細粒度な I/O と性能への影響

著者らは，これまで主に 4.1.1 で述べた Dual モデルに対してアクセスパターンの評価を行ったきた，その結果から，不揮発性メモリに退避されたデータを参照する際の著しい性能低下の要因として，サイズが 8 バイト程度の細粒度の I/O が大量にあることを明らかにした．この問題に対して，Hybrid-BFS アルゴリズムのパラメータを調整し，DRAM 上に全てのデータがある backward graph を使用することである程度性能低下を抑えることはでき，その後，著者らはデバイスを SATA 接続の SSD から高速な PCIe 接続のフラッシュデバイスに切り替えることでさらなる性能の改善を得ることはできた．さらには，I/O の API に対して mmap を使用することによって，キャッシュを有効活用するものや，libaio ライブラリを使用することで同時に複数の I/O を発行可能し，フラッシュに効率のよい I/O を行うことで，最大性能の改善には至らなかったが，性能が著しく低いものに対しては性能の改善を達成することができた．

よって，一見すると細粒度の I/O でも I/O を工夫することで高い性能を得られるように考えられるが，実際には全てのデータが DRAM にある Bottom-up アプローチを DRAM の場合に比べて多様することによる性能改善率が高かく，Single モデルのように一部のデータのみ DRAM に保持する場合では，この最粒度な I/O が大きな課題となると考えられる．

4.2.2 Bottom-up アプローチの特性分析

backward graph には，各頂点毎に隣接している頂点 ID が連続した領域に保持されており，この領域に先頭からアクセスし frontier の頂点が見つかるまで探索を行う．よって，frontier の頂点へのエッジが見つかった時点で探索を終了するため，参照されないエッジが存在する．Graph500 ベンチマークで使用するようなスモールワールド性のあるグラフでは，特に次数が大きい頂点において参照される可能性の低いエッジが多数ある．そこで，実際に各頂点毎にどの程度のエッジが探索されているのかを計測した．その結果を**図 3** に示す．x 軸は頂点毎に探索されたエッジ数である，y 軸は全探索に対する割合であり，値は左から右に向かって積算値となっている．青色の点は Top-down と Bottom-up の合計であり，赤色の点は青から Bottom-up の場合のみを取り出したものである．このことから，Bottom-up アプローチでは先頭から数個のエッジをみる場合が大半を占めており，且つ，全体の 30%から 40%程度の探索は各頂点ごとに先頭から数個のエッジのみ探索を行うだけで終わっていることが分かる．

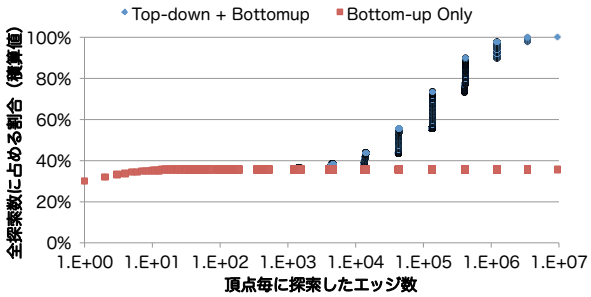

図 3 実際に探索されるエッジ数の計測

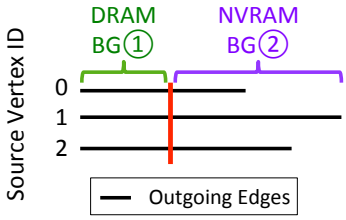

図 4 BG の一部を退避する手法

4.3 アルゴリズムと I/O の特性を考慮したグラフデータの省メモリ化手法

提案手法

4.2 で述べたように，Hybrid-BFS アルゴリズムでは，大量の細粒度な I/O と Bottom-up アプローチでは各頂点ごとに探索するエッジ数が少ないという特性がある．そこで，これらの特性を考慮したデータ分割手法について述べる．その様子を**図 4** に示す．具体的には各頂点毎に DRAM 上に配置する最大のエッジ数を定め，一部を DRAM に保持し，残りの領域を不揮発性メモリに退避することで，細粒度の I/O と性能の大半を占める Bottom-up アプローチへの I/O を DRAM 内に納めるようにする．また，このモデルでは DRAM 上に置く最大のエッジ数を調整することで，状況に合わせた DRAM 使用量の柔軟な変更ができる，

4.4 EBD-I/O

高速なランダム I/O を行うデバイスは商用，科学技術とわず様々な分野に大きなインパクトを与えている [7]．そのために，近年では DIMM スロットに DRAM のように設置することで，低レイテンシーを実現しているデバイスも発表はされているが，未だ普及はしていない．また，高速なランダムアクセスを提供するデバイスとして，例えば Fuison-io 社は PCIe 接続の NAND-Flash デバイスを提供しており，上位機種においては 100 万 IOPS を達成しているものもあるが，コモディティタイプの SSD と比較をすると非常に高価なデバイスであり，このデバイスをスパーコンピュータのような数千〜数十万台のノードから構成されるシステムにローカル I/O デバイスとして取り付けるの

は難しい．

そこでコモディティタイプの Flash デバイスを並列的に接続することで，低コストで高いバンド幅や IOPS を得ることが考えられ [8]，そのようなデバイスを想定し，プロトタイプとしてコモディティタイプの SSD を並列に接続したデバイスを構築した（EBD-IO:Extream Big Data I/O）．

白幡らはこれをバンド幅に重点をおき評価を行っているが [9]，我々はランダム I/O において重用視される IOPS の観点から評価を行う．

具体的な構成

複数の SSD を 1 つまとめる手法として，RAID カードに複数の SSD を取り付け，さらに複数枚の RAID カードを 1 つのノードに取り付けている．SSD は Crucial 社の mSATA 256GB を使用し，アクティブ時の消費電力は 200mW 程度である．RAID カードは Adaptec ASR-7805Q を使用している．今回は mSATA は 1 つの RAID カードに対して 8 枚づつ取り付け，この RAID カードを 2 つそれぞれ PCIe3.0 8 レーンの接続している．よって，容量は 1RAID カード辺り，2TB となり，合計で 4TB となる．RAID カードと mSATA SSD の間は SATA3 で接続し，RAID カード単位で RAID 0 を構成している．ストライプサイズは RAID カードの最小値である 64KB とした．ハードウェア RAID を使用しているためアプリケーション側からは 1 つの RAID カードは 1 つの 2TB の容量を持つブロックデバイスとして認識される．

性能の予備評価

mSATA デバイスのカタログスッペクは，read バンド幅が 500MB/s，write バンド幅が 260MB/s，4KB ランダム read が 45000IOPS，4KB ランダム write が 50000IOPS であり，理論上は 8 個のデバイスを同時に並列的に使用すればランダム read において 360000IOPS も達成可能である．

そこで，fio というベンチマークツールを用いて I/O ベンチマークを行った．ブロックサイズはグラフ処理を想定し，512 バイトとし，スレッド数と I/O-Depth を変化させた場合のランダム読み込みの性能を計測している．CPU は Intel Xeon E5-2690（8 コア）搭載のマシンを使用し，ハイパースレッドを使用することで，16 スレッドになり，これを 2 ソケット備えている．今回は同時に発行する I/O リクエスト数を表す I/O-Depth の値を 1，32，64 と変化させ，スレッド数を 2，4，8，16，32 と変化させており，全スレッドが合計で 32GB のデータを読み込むようにした．なお，プロトタイプデバイスは 2 つあるため，それぞれのデバイスに I/O を行うスレッドを均等に割り振り，2 デバイスをまとめた性能を計測している．

ベンチマークの結果を図 5 に示す．x 軸はスレッド数と I/O-Depth 数を示しており，上段が I/O-Depth，下段がスレッド数である．まず，I/O-Depth の値が上昇するにつれて，どのスレッド数の場合でも性能が最大となっていることが分かり，スレッド数が 8，I/O-Depth が 64 の場合において，性能が最大となり，443539IOPS となった．よって，コモディティタイプの SSD を使用しても高い IOPS を得られることが確認できた．

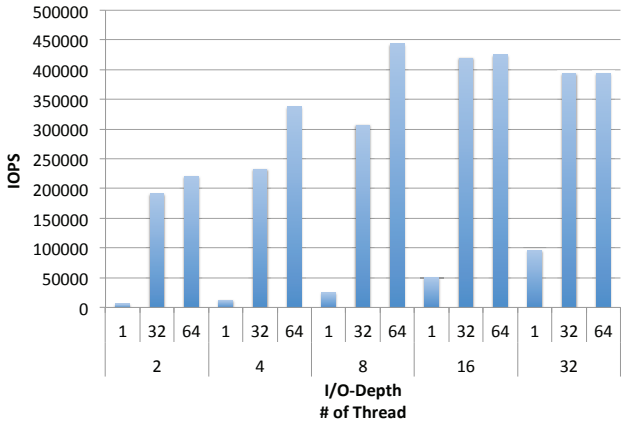

図 5 EBD-IO システムのベンチマーク結果

表 1 Machine Configurations(EBD-I/O プロトタイプマシン)

CPU	Intel(R) Xeon(R) CPU E5-2690 0 @ 2.90GHz (8 cores, 16 threads) × 2 sockets
DRAM	256GB
NVM	EBD-I/O プロトタイプ 2.0TB × 2
OS	CentOS 6.4(kernel 3.12.6)

5. 評価

5.0.1 評価環境

まず，評価環境は SCALE の大きいグラフを実行するために，提案手法の省にて述べた EBD I/O のプロトタイプデバイスを使用する．マシン環境を表 1 に示す．ここでは DRAM のサイズは 256GB とし，NVM として SSD を並列的に接続し，ハードウェア RAID0 を組んだ EBD-I/O のプロトタイプデバイスを 2 つ（合計 4TB）使用した．OS 側からは 2TB の NVM が 2 つに見え，且つ，NUMA ノードも 2 つあるため，NUMA ノードと EBD-I/O を 1 対 1 に対応させグラフデータのファイルを保存している．

5.1 DRAM 容量を超える状況での BFS 性能

ここでは，DRAM 容量を超える状況における BFS 性能について評価を行う．問題のスケールを上げた場合に，性能がどのように変化するかについて，SCALE を 23 から 31 まで変化させながら評価を行った．BFS 中に使用するデータは 3 つであり，グラフを CSR 構造にて表すための index 配列と value 配列，そして探索に必要な bitmap や BFS 木

を格納するための配列などをまとめた BFS Status である．今回評価を行った環境では，DRAM サイズが 256GB であるため，全てのデータを DRAM のみに保持する場合には，SCALE29 が限界となった．そこで，不揮発性メモリを活用し，各頂点毎に DRAM 上に保持するエッジの数を調整することで，CSR-Value の部分についてのデータ量を削減でき，SCALE31 の問題まで実行が可能となった．SCALE31 が最大のサイズとなる理由としては，エッジリストから CSR 構造のグラフを生成する際に，現在の実装では最大でエッジリスト本体とは別に，エッジリストの 4 倍の作業スペースが要求されるためである．SCALE31 ではエッジリストのみで 384GB，グラフ構築のための領域として連続した領域に 768GB×2 が必要となり，不揮発性メモリが 2.0TB× の環境では，SCALE31 が最大のサイズとなった．なお，この点については今後実装を変更し，余計な作業スペースを使用しないグラフ構造の構築を行っていく予定である．

図 6 に SCALE を 23 から 31 まで上げた場合の性能を示す．今回，評価を行ったのは BFS に必要な全てのデータを DRAM に保持した，DRAM Only のシナリオ，不揮発性メモリを活用した，DRAM + EBD-I/O シナリオについて評価を行った．まず，両シナリオについて，SCALE24 までは性能が向上し，SCALE が 26 以上からは緩やかに減少している．DRAM Only では SCALE29 まで実行が可能であるため，グラフは SCALE29 までとなっており，SCALE29 にて 4.11GTEPS を得られた．DRAM + EBD-I/O では DRAM 上のグラフサイズを削減することで，SCALE29 を超えた場合にも実行可能となり，さらに，著しい性能低下率を引き起こすことはなく，グラフサイズが 4 倍になる SCALE31 では 3.8GTEP を得た．これは，DRAM Only の SCALE29 に対して 6.9%のみの性能低下率となった．

ここで，DRAM 容量を超えた場合での低い性能低下率を得ることができた点について，分析を行った．その結果，SCALE31 の場合では，全体の I/O のうち約 60%が不揮発性メモリに対して行われているが，性能の大半を示す Bottom-up アプローチは DRAM 内で完結していることがわかった．また，iostat コマンドを用いて I/O リクエストの平均長を計測したところ，約 65KiB となり提案手法を用いることで細粒度な I/O を削減することができ，効率の良い I/O を行うことができており，性能低下を最小に抑えることができたと考えられる．

5.2 DRAM 使用量を変化させた場合の BFS 性能

ここでは，DRAM 容量を超えるグラフサイズにて，各頂点毎に DRAM に配置するエッジの最大数を調整することで，DRAM 使用量を変化させた場合に性能にどのように影響があるのかについて評価を行った．図 7 にその結果を示す．x 軸は各頂点毎に DRAM に配置するエッジの最大

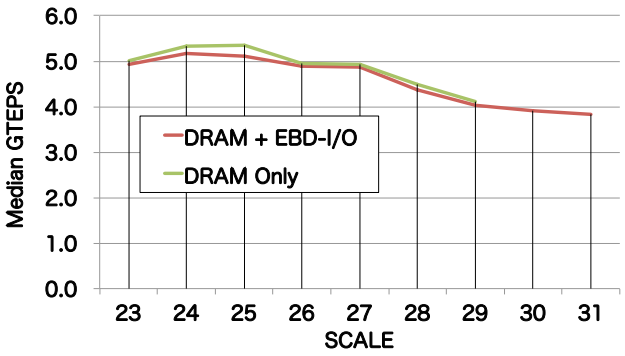

図 6 SCALE を上げた場合の BFS 性能

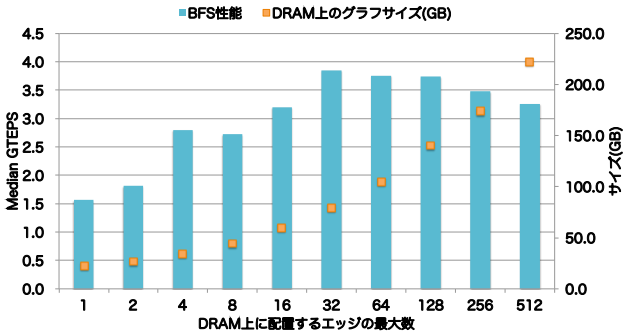

図 7 各頂点が DRAM 上に保持するエッジの上限数を変化させた場合の DRAM 上のグラフのサイズ：$SCALE = 31, edgefactor = 16$

数を調整した場合の DRAM 上に作成されるグラフのサイズ（GB）を表す（Index と Value 配列の合計）．エッジの最大数は，1 から 512 まで，2 倍づつに増やしながら評価を行った．左側の y 軸は Median GTEPS の値を示し，右側の y 軸は DRAM 上のグラフサイズ (GB) を表す．エッジの最大数が増加するにつれ，DRAM 上のグラフサイズは上昇する．しかし，Median TEPS はエッジの最大数が 1 から増加するにつれ 32 までは，性能が向上し，最大で 3.9GTEPS を得ることができたが，エッジ数が 64 以上の場合ではエッジの最大数が増加するにつれ性能が低下し始め，512 の場合には 3.3GTEPS となった．

この点について性能分析を行ったところ，例えば最大のエッジ数が 256 の場合は 32 の場合に比べて，実際にデバイスからの I/O 量が 10%程度増加しており，これは DRAM のサイズに空きができることで OS 側のページキャッシュが有効に作用し，I/O 量の減少とそれに伴う性能向上が達成できたと考えられる．

5.3 Green Graph500 による電力効率性能評価

提案手法を用いることで，より DRAM 容量が少ない環境においても性能低下を抑えながら大規模グラフの実行ができることが確認できた．我々の研究の目的の 1 つは，不揮発性メモリを活用することで，データインテンシィブな問題においても，DRAM 容量やノード数の増設を抑え，消

表 2　電力効率（EBD-I/O プロトタイプマシン）

SCALE	30	31
Edgefactor	16	
GTEPS	4.04	3.72
Power(W)	331.68	386.83
MTEPS / W	12.19	9.60

費電力の増加を防ぐことである．そこで，データインテンシブ問題に対する電力効率性のベンチマークである Green Graph500 ベンチマークを行い，電力効率の評価を行った．その結果，SCALE30 では電力効率は 12.19MTEPS/W となり，SCALE31 では電力効率は 9.60MTEPS/W となった（表 2）．ここで，2013 年 11 月時点での Green Graph500 BigData カテゴリーの第 1 位は 6.72MTEPS であり，本研究ではそれより高い値を得ることができた．これは，消電力な不揮発性メモリを活用することで，DRAM 容量やシステムの規模を縮小しながらも，高速に BFS を実行できたためである．

6. 関連研究

不揮発性メモリをグラフ処理に用いた研究として Kyrola らはグラフデータを一定のチャンクに分割しディスクに保存するこで，DRAM への読込みをシーケンシャルに行い，レイテンシーの隠蔽を行うモデルがある [10]．しかし，Hybrid-BFS の場合は，特に Top-down アプローチにおいてメモリアクセスがランダムになってしまうため，この手法の適応は難しい．

Pearce らは BFS 以外のグラフ処理にも使用できるアルゴリズムを採用し，コア数を超える大量のスレッドを使用することで不揮発性メモリとのレイテンシーを隠蔽している．しかし，アルゴリズムの特性上、全てのエッジを探索するため，BFS 時の性能は Hybrid-BFS に比べ低くなってしまう [11] [6]．

マルチノードに関する研究では，Beamer らが Hybrid-BFS のマルチノード化を行っており [12]，Pearce らも不揮発性メモリを搭載したマルチノード環境での研究を行っている [13]．

7. まとめ・今後の方針

科学技術計算におけるデータ量は急速に増大しており，膨大な量のデータを処理するには，将来のスーパーコンピュータでは消費電力や価格面でのコストの増加からローカルに取り付けられた不揮発性メモリに注目が集まっており，このようなデバイスを DRAM に対して補助的に利用することで，メモリを多階層化することで，単一の計算ノード上の DRAM には収まらない容量のグラフを実行できる可能性があるものの，その具体的手法やどの程度の性能低下が起きるのかなどの定量的な指標は明らかではない．

一方で，近年，様々な大規模なグラフに対する高速処理が求められており，データインテンシブ問題の代表例であるグラフ処理を対象とし，DRAM 容量を超える環境でも性能低下を最小限に抑えた大規模実行を目指すことで将来のスーパーコンピュータにおける I/O のあり方について検討を行っている．

そこで，これまでの研究から明らかになった大きな性能低下要因となる，細粒度な I/O が頻発するグラフ処理において，アルゴリズムの特性を分析しデータ配置を行うことで，性能低下を抑えながら DRAM 容量を削減できる手法を提案した．さらに，SSD を並列的に接続することで高い IOPS を実現できる I/O アーキテクチャを用いることで，シングルノード上で 6.9%の性能低下で DRAM のみの場合に比べて 4 倍のサイズの問題を実行できた．また，データインテンシブベンチマークの電力の効率性を競う，Green Graph500 の Big Data カテゴリにおいて現世界一の 1.8 倍の電力効率を達成することができた．よって，シングルノードにて不揮発性メモリをメインメモリの延長として使用することの有効性を確認すると共に，並列的な I/O（API，デバイス構成共に）を行うことでフラッシュメモリを有効活用でき，さらに，不揮発性メモリをメインメモリの延長として使用する場合には，ユーザによる多階層メモリへの明示的なデータの配置が重要であることを確認した．

今後の方針としては，以下がある．

- I/O アーキテクチャのさらなる検討
  よりフラッシュに最適化されたファイルシステムの使用，I/O の API は何が適切か
- メモリの多階層化と性能モデル構築
  メインメモリの延長とストレージとしての NVM をどう活用していくのか
- マルチノード化と性能モデル構築
  データインテンシィブアプリケーションにおける，ローカルノードのデバイスとリモートの DRAM へアクセスをする場合の性能特性
- データの局所性を高めるプログラミングモデルの提案
  NVM を DRAM の延長として使うためには，アルゴリズムの特性を考慮した明示的なデータ配置とキャッシュを組み合わせることが必要

謝辞　本研究の一部は JST CREST「ポストペタスケールシステムにおける超大規模グラフ最適化基盤」と「ビッグデータ統合利活用のための次世代基盤技術の創出・体系化」の援助による．

参考文献

[1] IDC: The Digital Universe in 2020: Big Data, Bigger Digi tal Shadows, and Biggest Growth in the Far East, http://www.emc.com/leadership/digital-universe/iview/index.htm.
[2] Beamer, S., Asanovic, K. and Patterson, D.: Search-

ing for a Parent Instead of Fighting Over Children: A Fast Breadth-First Search Implementation for Graph500, *Technical Report UCB/EECS-2011-117, EECS Department, University of California, Berkeley*, pp. 1–9 (2011).

[3] Suzumura, T., Ueno, K., Sato, H., Fujisawa, K. and Matsuoka, S.: Performance Characteristics of Graph500 on Large-Scale Distributed Environment, *Proceedings of the 2011 IEEE International Symposium on Workload Characterization (IISWC 2011)*, pp. 149–158 (2011).

[4] Beamer, S., Asanović, K. and Patterson, D.: Direction-optimizing breadth-first search, *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'12)*, Salt Lake City, USA, IEEE Computer Society Press, pp. 12:1–12:10 (2012).

[5] Yasui, Y., Fujisawa, K. and Goto, K.: NUMA-optimized Parallel Breadth-first Search on Multicore Single-node System, *2013 IEEE International Conference on Big Data (IEEE BigData 2013)* (2013).

[6] Pearce, R., Gokhale, M. and Amato, N. M.: Multithreaded Asynchronous Graph Traversal for In-Memory and Semi-External Memory, *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC10)*, IEEE, pp. 1–11 (online), DOI: 10.1109/SC.2010.34 (2010).

[7] Hey, A. J., Tansley, S., Tolle, K. M. et al.: The fourth paradigm: data-intensive scientific discovery (2009).

[8] Zheng, D., Burns, R. and Szalay, A. S.: Toward millions of file system IOPS on low-cost, commodity hardware, *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*, ACM, p. 69 (2013).

[9] 白幡晃一, 佐藤仁, 松岡聡：GPU アクセラレータと不揮発性メモリを考慮した I/O 性能の予備評価, 情報処理学会研究報告. [ハイパフォーマンスコンピューティング] (2013).

[10] Kyrola, A., Blelloch, G. and Guestrin, C.: GraphChi : Large-Scale Graph Computation on Just a PC Disk-based Graph Computation, *Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation*, pp. 31–46 (2012).

[11] Van Essen, B., Pearce, R., Ames, S. and Gokhale, M.: On the Role of NVRAM in Data-intensive Architectures: An Evaluation, *2012 IEEE 26th International Parallel and Distributed Processing Symposium*, Ieee, pp. 703–714 (online), DOI: 10.1109/IPDPS.2012.69 (2012).

[12] Beamer, S., Buluc, A., Asanovic, K. and Patterson, D.: Distributed Memory Breadth-First Search Revisited: Enabling Bottom-Up Search, *Workshop on Multithreaded Architectures and Applications (MTAAP2013) in conjunction with 27th IEEE International Parallel & Distributed Processing Symposium (IPDPS2013)*, Boston, pp. 1 – 10 (2013).

[13] Pearce, R., Gokhale, M. and Amato, N. M.: Scaling Techniques for Massive Scale-Free Graphs in Distributed (External) Memory, *2013 IEEE International Parallel & Distributed Processing Symposium (IPDPS2013)*, pp. 825–836 (online), DOI: 10.1109/IPDPS.2013.72 (2013).