ハードウェアオフロードによる MapReduce の高速化

本庄利守[†1]　及川一樹[†1]

大規模データを処理する基盤として、MapReduce と呼ばれる分散処理フレームワークが広く使われてきている．MapReduce は、多数のサーバから並列にデータを読み書きすることで、I/O ボトルネックを克服するアーキテクチャとなっていることが特徴である．しかし、ディスクやネットワークの高速化が進むことで、従来想定していた I/O ボトルネックから CPU ボトルネックに移行することが予想される．そこで、本論文では、将来スタンダードになるであろう SSD や Infinband などの高速なディスクやネットワークを用いたベンチマークを通じて、実際に I/O ボトルネックから CPU ボトルネックとなること示し、この CPU ボトルネックを克服する手法として、ハードウェアオフロードによる MapReduce の高速化を提案する．今回は、Map 処理におけるデータのデシリアライゼーション、パースおよびキーによるソートを実行する箇所を Tilera 社のメニーコアプロセッサボードにオフロードするプロトタイプの実装、評価を通じて、本方式のフィージビリティを示す．

Hardware Accelerated MapReduce

TOSHIMORI HONJO[†1]　KAZUKI OIKAWA[†1]

1. はじめに

近年、ビッグデータの処理基盤として、Google により提案された MapReduce と呼ばれる分散処理フレームワークが広く利用されるようになってきた[1]．特に、オープンソースのクローン実装である Hadoop は、MapReduce のデファクトスタンダードとなりつつある[2]．現在の MapReduce は、現行のコモディティサーバと現行のネットワークを想定し、ディスクやネットワークの I/O 性能がボトルネックになることを想定したアーキテクチャとなっている．一方で、SSD (Solid State Drive)や Infiniband などのように、ストレージやネットワークなどのハードウェアの進化が進んでおり、これらがコモディティ化されると、従来の MapReduce が想定していた I/O ボトルネックに対して、CPU ボトルネックに移行することが予想される．本論文では、近い将来、スタンダードとなることが予測される SSD のような高速ストレージや Infiniband のような高速ネットワークを用いた Hadoop MapReduce のベンチマークを通じて、I/O ボトルネックから CPU ボトルネックに移行が現実に起こることを示す．そして、CPU ボトルネックを克服し、I/O の限界性能を引き出すようなデータ処理を実現するための一案として、ハードウェアオフロードによる MapReduce の高速化を提案し、Tilera 社のメニーコアプロセッサボードを用いたプロトタイプ実装と評価を通じて、本方式のフィージビリティを示す．

2. MapReduce に対する課題

2.1 MapReduce の概要

まず、MapReduce における処理の流れを説明する[3]．図 1 のように、MapReduce は、大きく Map と Reduce と呼ばれる 2 つの処理から構成される．各 Map および Reduce 内の処理シーケンスを図 2 に示す．Map 処理では、分散ファイルシステム上に蓄積されたデータを読み出し、そのデータのデシリアライゼーション、パースを行い Key と Value の値を決定する．そして、メモリ上に展開された Key と Value のペアに対し、Key によるソートを実施し、シリアライズ処理をしながら、ディスク上のファイルに書き出す．以上を繰返し、Map 処理の最終段階で、ディスクに書き出したソート済みファイルをマージし、1 つのソート済みファイルを作成する．Reduce 処理では、各 Map が出力したファイルをネットワーク越しに回収し、それらをマージして、1 つのソート済みファイルを作成した後に、各 Key に対して集約された Value に対し、具体的な Reduce 処理を適用し、結果をファイルシステムに格納する．

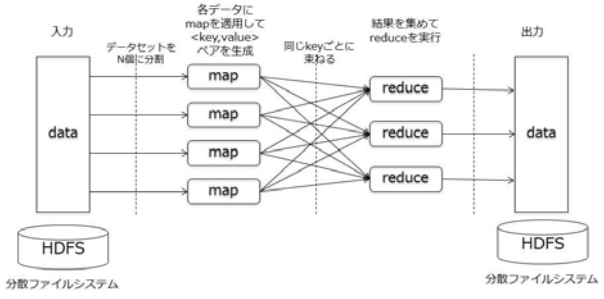

図 1：MapReduce の概要

†1 日本電信電話株式会社 NTT ソフトウェアイノベーションセンタ
honjo.toshimori@lab.ntt.co.jp

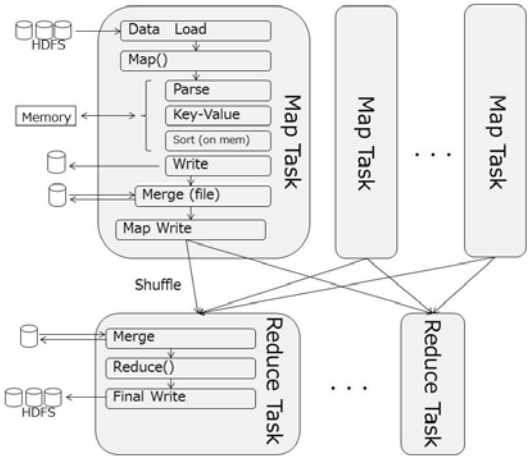

図 2：MapReduce の処理シーケンス

2.2 MapReduce の課題

MapReduce は、コモディティサーバを前提に大規模データを効率的に処理するために考案されたフレームワークである．基本的な考え方として、現行の CPU の処理性能に比べて、ディスクの読み出し/書き出し速度やネットワークの転送速度の方が遅いことを想定しており、コモディティサーバを大量に並べて、多数のディスクから並列に読み出すことで、ディスクの I/O 性能を稼ぐこと想定している．また、ローカルのディスク I/O に比べて、ネットワークによる転送の方が遅いことを想定し、可能な限りネットワーク転送を減らし、ローカルで処理するようにもデザインされている．通常、HDD の読み出し速度は、100MB/秒程度であり、各サーバには数台のディスクを搭載することが通例であり、数 100MB/秒の読み出し速度となる．また、ギガビットイーサネットの転送速度は、その名の通り、1Gbps 程度である．一方で、近年、ストレージやネットワークの進化も進んでおり、SATA 接続の SSD(Solid State Disk)では 1 台で 500MB/秒程度の読み出しができ、さらに PCI Express 接続の場合には、数 GB/秒の読み出しが可能となってきている．また、ネットワークに関しては、10 ギガビットイーサネットがコモディティ化してしてきていると共に、Infiniband においては 56Gbps の通信が可能となってきている．このように、最新のデバイスでは、従来に比べて性能が 1 桁近く向上してきており、MapReduce のアーキテクチャが前提としてきた I/O ボトルネックに変化を及ぼす可能性がある．最新のデバイスにより、ディスクやネットワークの I/O 性能が上がり、I/O のボトルネックが解消されると仮定すると、次にボトルネックになるのは CPU による処理となる．前節で述べた MapReduce の処理のシーケンスの中で、CPU 負荷になりそうな主な処理は、Map における(1)データのデシリアライゼーション、パース、(2)キーによるソート (3)ソート済みファイルのマージ、Reduce における(4)ソート済みファイルのマージなどである．これらが、CPU ボトルネックになり性能が律速されることが予想される．

2.3 ベンチマークによる評価

上記の予想を裏付けるため、実機によるベンチマーク実験を実施した．最新の高速なディスクとして、PCI Express 接続の SSD である Intel Solid-State Drive 910 (800GB) (公称での読み出し速度は、2GB/秒)を用いて、Hadoop の標準的なベンチマークである Grep および Terasort による評価を行った． 実験では、CPU Xeon E5-2690(2.9GHz、8core、16thread) × 2 台、RAM 128GB、Intel Solid-State Drive 910 (800GB)、Mellanox InfiniBand MCX354A-FCBT(Dual FDR 56Gb/s) を搭載したサーバ 6 台を用いて、1 台をマスター、残りの 5 台をワーカーとしてセットアップし、Hadoop のディストリビューションとしては、Cloudera Hadoop Dirtribution (CHD 4.1.2)を用いた．

まず、Grep による正規表現による文字列検索の性能をベンチマークした．検索対象のデータには、Teragen で生成した 400GB のデータを用いて、全データのうち、1 行だけヒットするような正規表現で検索を実施した．実験では、CPU 利用率をモニタリングしながら、実行時間を測定した．全処理時間は、2 分 55 秒であった．Map 処理に関して注目すると、約 3 分かかっており、5 台のサーバで処理したことから、1 サーバ当たり約 450MB/秒で処理している換算となる．ディスクの読み出し速度が 2GB/秒であることを考えると、I/O を十分には使いきれていないことが分る．また、図 3 は、あるワーカーにおける CPU 利用率をモニターした結果であるが、90%以上の使用率となっており、CPU ボトルネックとなっていることが分る．

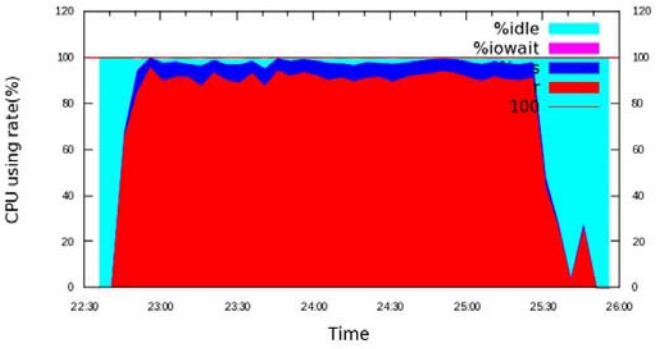

図 3：Grep によるベンチマーク

次に、Terasort によるソート性能のベンチマークを行った．Grep と同様に Teragen で生成した 400GB のデータを用いた．全処理時間は、7 分 53 秒であった．同様に、Map 処理に関して着目すると、約 6 分かかっており、1 サーバ当たり約 228MB/秒で処理した換算となる．こちらも、2GB/秒の読み出し速度に対して、10 分の 1 程度の速度しか出せていない．前述の Grep との比較としては、Map 処理で key によるソートなどの処理が入るため、Map での処理速度の低下している．同様に、図 4 は、あるワーカーにおける CPU 利用率

をモニターした結果であるが、90%以上の使用率となっており、CPU ボトルネックとなっていることが分る.

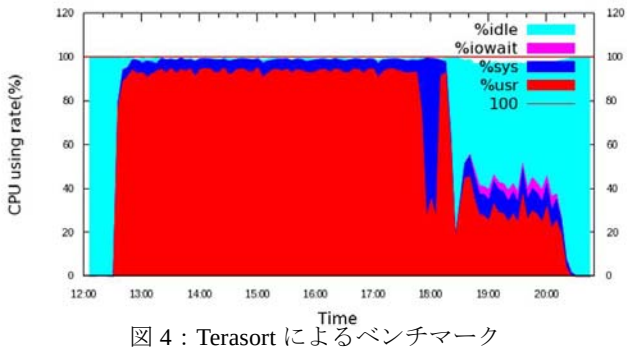

図 4：Terasort によるベンチマーク

以上のように、簡易なベンチマーク実験であるが、最新のデバイスを用いると Grep や Terasort などのように、比較的単純な処理でも CPU ボトルネックとなることが示された. 上記の実験では、PCI Express 接続の SSD を高々1 台しか接続していないが、複数台搭載することも可能であり、その場合にはさらに顕著な結果となることが予想される.

3. ハードウェアオフロードによる MapReduce の高速化

3.1 MapReduce におけるハードウェアオフロード

前章で述べたように、I/O ボトルネックを前提としてデザインされた MapReduce であるが、ハードウェアの進化に伴ってディスクやネットワークの I/O 性能が向上してくると、今度は CPU がボトルネックとなり、全体としての処理性能が頭打ちになる. 一方で、サーバホストに搭載されるメインの CPU のクロックや CPU の数、コアの数などが、今後、飛躍的に向上してくとは考えにくく、各サーバの性能をフルに使いきれないまま、サーバ台数を増やすことによって性能を向上させるしかない. さらに、各サーバの性能をフルに使い切らないまま、台数を増やすことは、電力利用の観点でも非効率である.

そこで、MapReduce における CPU ボトルネックを克服するため、サーバのバス上に拡張できるメニーコア CPU や FPGA などを搭載したアクセラレータボードを用いた MapReduce の高速化を提案する. このような方式であれば、性能要件に応じて、バス上にアクセラレータボードを追加することで、比較的スケーラブルに性能を上げることが可能となる. 我々は、アクセラレータによる MapReduce の高速化を目指すが、現行の MapReduce のフレームワークをでいきるだけ維持しつつ、アセラレータを活用できる枠組みの実現を目指す. 2 章で述べたように MapReduce では、Map と Reduce の 2 つの処理から構成されると共に、key-value 形式のデータ構造を扱うなど、比較的定型的な処理の組み合わせから成り立っている. なおかつ、各処理の独立性は高く、並列化も比較的容易である. さらに、処理内容をブレイクダインすると、Map における(1)デシリアライゼーション、パース、(2)キーによるソート (3)ソート済みファイルのマージ、Reduce における(4)ソート済みファイルのマージ、さらに、Map および Reduce 内における key-value に対する演算処理(機械学習処理における数値計算など)が CPU ボトルネックになり性能が律速されることが予想され、これらをアクセラレータボードで処理することで、ホスト CPU の負荷を下げ、さらにはアクセラレータボード上で高速処理することで、MapReduce 自体の高速化につながる. 今回は、Map 処理における(1)デシリアライゼーション、パース、(2)キーによるソートに着目し、メニーコアプロセッサを搭載するアクセラレータボードへオフロードする方式の検討およびプロトタイプによる評価を行う.

3.2 Tilera メニーコアプロセッサによる Map 処理のオフロード

Map 処理をオフロードするアクセラレータボードとして、Tilera 社のメニーコアプロセッサボード(Tilera TILEncore-Gx36)を用いた[4]. Tilera 社のメニーコア CPU は、各コア間をメッシュで接続するホモジニアスな構成のメニーコア CPU である. また、低消費電力も売りとしているプロセッサであり、今回使用したプロセッサの消費電力の公称値は、35W である. また、今回使用したアクセラレータボードには、36 コアを搭載した CPU 1 台(クロックは 1GHz)、RAM は 8GB が搭載されている. 本ボードは、ホストサーバの PCI Express 上に拡張できるボードで、ボード上の CPU では、SMP Linux が動作する.

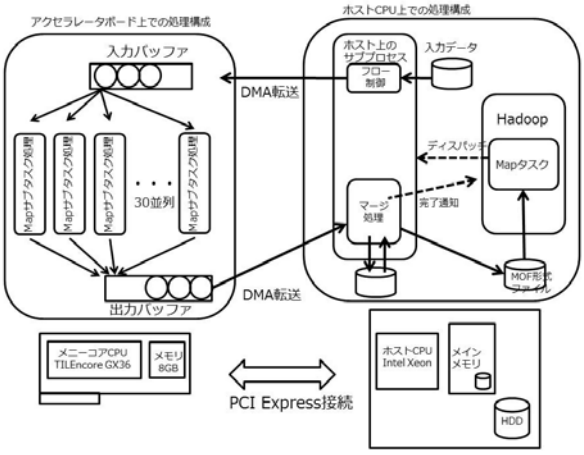

図 5：Tilera メニーコア CPU ボードへのオフロード

ホスト CPU 上での Map 処理のうち、CPU 負荷の高いデータのデシリアライゼーション、パース、キーによるソートの部分(以後、Map サブタスク処理と呼ぶ)を、Tilera 社のメニーコアプロセッサを搭載したボード(TILEncore-Gx36)にオフロードした. ただし、ボード上のメモリ容量が限られているため、Map 処理の最終段階で行われるソート済みフ

ファイルのマージはホスト側で行うデザインとした. 図 5 に全体構成およびデータの流れを示す. ボード上では、Linux が動作しており、36 個のコアを使った並列処理が可能である. Hadoop の Map 処理部にフックを仕掛け、外部にディスパッチするように改造した.

ホスト側には、Tilrea のメニーコアプロセッサボードとのデータの転送およびソート済みファイルのマージを行うプロセスを配置し、ボード上には、Map サブタスク処理をマルチスレッドで行うプロセスを配置した. これらは、C 言語で実装されており、オブジェクト生成コストなどの Java 固有のオーバーベッドは発生しない.

ホスト側で、HDFS やローカルディスクからブロック(64MB)単位のデータを読みだし、ボード上に DMA 転送し、キューに積み、ボード上では、30 並列のスレッドにより、キューからのデータを取り出し、Map サブタスク処理を行う. 出力結果は再び、DMA 転送によりホスト側に戻され、全データ処理が完了した時点で、それらのファイルのマージを実施し、Hadoop の Map に処理を戻す. 以降は、Hadoop のフレームワークの中で処理が進んでいく.

4. 評価

Cloudera Hadoop Dirtribution (CHD 3 update4)をベース実装した上記のプロトタイプを用いて、単一のホスト上で Hadoop の疑似分散モードにより Grep、Wordcount および Terasort による評価を実施した. ホストには、CPU Intel Xeon E5-1660 3.3GHz 6 コア(HT)、メモリ 128GB を搭載したマシンを使用した.

4.1 Grep

「50GB のランダムな辞書データ(スペースで区切り)から、各単語をパースし、正規表現(/^a{4}/)にマッチしたもののみを抽出する」という MapReduce 処理を用いて、ホスト CPU のみ(Map は 12 並列)の場合とハードウェアオフロードした場合の比較を行った. ただし、双方とも同じ処理で比較するため、Hadoop にサンプルとして付属する Grep ではなく、文字列を Tokenizer により切り出して、各文字列に対して正規表現比較を実施する Java の MapReduce コードを実装して評価した. また、双方とも、ディスクの I/O がボトルネックとならないように、メモリ上に作成したファイルシステム(tempfs)を用いた. (図 6 には参考のため、HDD を用いた場合の結果も含めている) 図 6 に実験結果を示す. ホスト CPU のみの場合は、211 秒でレートに換算すると 242MB/s であった. 現状の SATA 接続の SSD からの読み出し速度が 500MB/s 程度であることを考えると、I/O 性能を使い切れないことが分る. 一方、ハードウェアオフロードした場合には、44 秒で、レートに換算すると 1163MB/s であった. ただし、Tilera 上での Map サブタスク処理のみに着目すると、35 秒で完了しており、1462MB/s に相当する. この場合、SSD 2 台程度であれば、I/O 性能限界まで引き出せることとなる.

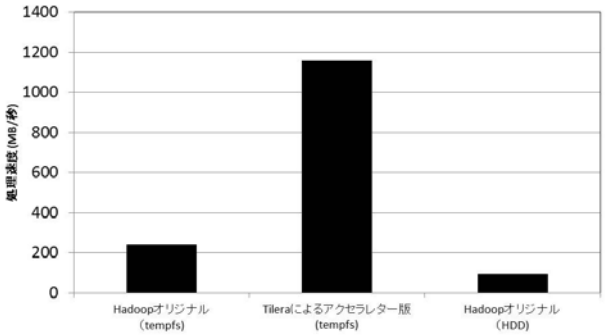

図 6 : Grep による性能評価

4.2 Wordcount

次に 20GB のランダムな辞書データ(スペースで区切り)に対する各単語数の出現頻度計算処理である Wordcount を実施した. こちらも、ホスト CPU のみ(Map は 12 並列)の場合とハードウェアオフロードした場合の比較を行った. 双方とも、ディスクの I/O がボトルネックとならないように、メモリ上に作成したファイルシステム(tempfs)を用いた. 図 7 に実験結果を示す. ホスト CPU のみの場合は、1050 秒でレートに換算すると 19MB/s であるのに対して、ハードウェアオフロードした場合には、162 秒でレートに換算すると 126MB/s であった.

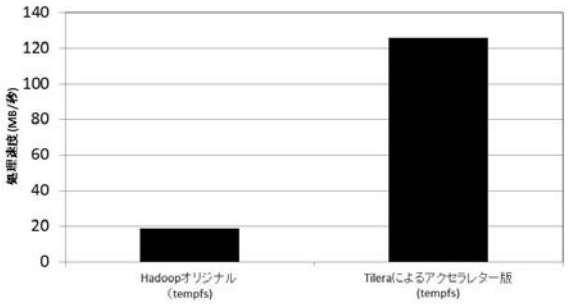

図 7 : Wordcount による性能評価

4.3 Terasort

最後に Terasort による性能評価を実施した. Teragen で生成した 10GB のデータを用いた. こちらも、ホスト CPU のみ(Map は 12 並列)の場合とハードウェアオフロードした場合の比較を行った. 双方とも、ディスクの I/O がボトルネックとならないように、メモリ上に作成したファイルシステム(tempfs)を用いた. 図 8 に実験結果を示す. ホスト CPU のみの場合は、401 秒でレートに換算すると 25MB/s であるのに対して、ハードウェアオフロードした場合には、138 秒でレートに換算すると 74MB/s であった. 参考までに、Tilera 上での Map サブタスク処理のみに着目すると、30 秒で完了しており、409MB/s に相当する.

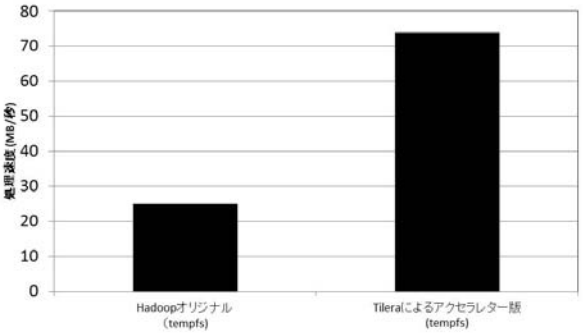

図 8：Terasort による性能評価

4.4 考察

Grep、Wordcount および Terasort による性能評価を通じて、ハードウェアオフロードによる性能向上を実現することが示せた．今回は、36 コアの CPU を 1 機搭載したボード 1 枚を用いた実験であったが、バスに追加するという形態であることから、必要な処理性能に応じて、スケーラブルに性能を伸ばすことが可能である．従来のホスト CPU の性能を改善するスケールアップ方式に比べて、比較的容易にある意味、スケールアウト的に性能を改善していくことができる．

一方で、今回の実験で用いたホスト側の CPU である 3.3GHz の 6 コア(12 スレッド) Xeon と、ボード上の CPU である 1.0GHz の 36 コアの Tilera Gx36 は、SPECint ベンチマークの値では大差はなく、本来であれば、同じような性能が得られそうなものである．これは、Java によるオーバーヘッドが原因であると考えている．すなわち、Hadoop 側を C 言語で書き直すことでも、上記の実験程度の結果が得られると考えられる．しかし、上述したように、ホスト側の CPU の性能の飛躍的な向上は難しいことから、アクセラレータボードによるスケーラブルな方式は有用である．

また、消費電力という観点では、大きなアドバンテージがある．今回用いたホスト CPU の消費電力は 130W であるのに対して、Tilera のメニーコア CPU は 35W である．Grep 処理に関して言うと、ホスト CPU のみでは 1.86MB/W に対し、Tilera 側は 46.5MB/W と約 25 倍以上の差がある．ビッグデータ処理では、これらのサーバを多数並べて処理することから、低消費電力も大きなアドバンテージとなる．

Wordcount および Terasort に関しては、SSD などの高速ストレージの I/O を使い切るまでには至らないとの実験結果であった．これを改善するためには、アクセラレータボードを追加して、今回の Map サブタスクの処理をさらに高速化すると共に、ホスト CPU で実施している Tilera 上で処理したファイルのマージ処理、および Reduce 側におけるマージ処理の高速化も必要となってくる．これらに関してもハードウェアオフロードを検討しており、今後の課題である．

5. 関連研究

Hadoop のベンチマークに関する研究は、多数行われているが、内部まで踏み込んだ解析を行っているものは少ない．Singapore 大学の Dawei Jiang らは、MapReduce の性能改善手法を提案し、ベンチマークにより効果を示している．この中で、データのパースコストが高いと共に、Java による影響の大きさを指摘している[5]．Massachusetts 大学の Edward Mazur らは、MapReduce 処理における CPU 利用率に着目したベンチマークを実施しており、Map 処理においては、Map に記述された処理に加えて、key によるソート処理も大きな割合を示すことが指摘されている[3].また、Berkeley 大学の Matei Zaharia らは、RDD(Resilient Distributed Datasets)と呼ばれる分散共有メモリを用いた MapReduce の高速化を提案している．本論文の中で、データのロード、いわゆるデシリアライゼーションのオーバーヘッドが思いのほか、大きいことを指摘している[6].

また、Hadoop のハードウェアアクセラレーションに関しては、Map や Reduce 内の数値計算などの演算処理をオフロードする研究がいくつか行われている．Catalonia 工科大の Yolanda Becerra らは、IBM Cell BE プロセッサを搭載したアクセラレータボードにオフロードすることによって、暗号処理などの CPU インテンシブな処理が高速化できることを示している[7]．一方で、このような方式では、Grep や Terasort のような大量の I/O が発生するようなデータ処理に関しては、高速化できなかったことも述べられている．また、オフロード先に GPGPU を利用する研究も多数行われている．Hong Kong 大学の Bingsheng He らによる、Mars と呼ばれるフレームワークなど、いくつかの提案が行われている[8]．これらも主に数値計算をオフロードすることで、高速化を実現している．

また、Exar 社からは、PCI Express に搭載するハードウェアボードによりデータの圧縮、解凍処理をオフロードするプロダクトが製品化されている．[9]

高速ネットワークである Infiniband を用いた Hadoop 高速化の研究も行われている．Ohio 州立大学の Sayantan Sur らは、Infiniband を適用した Hadoop のベンチマークを実施し、HDFS が高速化できることを示している．Infiniband による効果を述べると共に、SSD を用いることによるストレージ高速化の効果が高いことも述べられている [10]．また、Aurbun 大学の Yondong Wang らは、Infiniband の RDMA を活用し、Reduce におけるソート済ファイルのマージと Reduce 自体の処理をシリアライズせず逐次的に処理することで、効率的に処理する手法を提案している [11].

6. まとめ

本論文では、大規模分散処理基盤である MapReduce において、今後デバイスの進化に伴って、ストレージやネットワークの高速化に伴って、CPU のボトルネックが顕著化することを述べた. MapReduce における CPU ボトルネックを克服するため、サーバのバス上に拡張できるメニーコア CPU や FPGA などを搭載したアクセラレータボードを用いた MapReduce の高速化を提案し、今回は Map 処理の一部を Tilera 社のメニーコアプロセッサにオフロードするプロトタイプの実装および評価を通じて、本方式のフィージビリティを示した.

参考文献

1) Dean, J. and Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters, *Proc. Sixth Symposium on Operating System Design and Implementation (OSDI'04)*, pp.137-150 (2004).
2) Welcome to Apache Hadoop (online), available from <http://hadoop.apache.org>.
3) Mazur, E. Li, B. Diao, Y. and Shenoy, P.: Towards Scalable One-Pass Analytics Using MapReduce, *Proc. IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum (IPDPSW '11)*, IEEE, pp. 1102-1111 (2011).
4) Tilera Corporation: TILE-Gx8036 Processor Specification Brief (online), available from <http://www.tilera.com/sites/default/files/productbriefs/TILE-Gx8036_PB033-02_web.pdf>.
5) Jiang, D. Ooi, B.C. Shi, L. and Wu, S.: The performance of MapReduce: an in-depth study, *Proc. VLDB Endowment, Volume 3 Issue 1-2, September 2010*, pp. 472-483, (2010).
6) Zaharia, M. Chowdhury, M. Das, T. Dave, A. Ma, J. McCauley, M. Franklin, M. J. Shenker, S. and Stoica, I.: Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, *Proc. 9th USENIX conference on Networked Systems Design and Implementation (NSDI'12)*, 2-2, (2012).
7) Becerra, Y. Beltran, V. Carrera, D. Gonzalez, M. Torres, J. and Ayguade, E. Speeding Up Distributed MapReduce Applications Using Hardware Accelerators, *Proc. 2009 International Conference on Parallel Processing (ICPP '09)*, IEEE Computer Society , pp. 42-49, (2009).
8) He, B. Fang, W. Luo, Q. Govindaraju, N.K. and Wang, T.: Mars: A MapReduce Framework on Graphics, *Proc. 17th international conference on Parallel architectures and compilation techniques (PACT '08)*, pp.260-269, (2008).
9) Robert Reiner: Maximizing Hadoop Performance with Hardware Compressoion (online), available from <http://www.exar.com/common/content/document.ashx?id=21230>.
10) Sur S., Wang, H. Huang, J. Ouyang X., and Panda, D. K.: Can High-Performance Interconnects Benefit Hadoop Distributed File System?, *Workshop on Micro Architectural Support for Virtualization, Data Center Computing, and Clouds (MASVDC).* (2010).
11) Wang, Y. Que, X. Yu, W. Goldenberg, D. and Sehgal, D. :Hadoop Acceleration through Network Levitated Merge. *Proc. International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11)*, (2011).