ファイル格納位置最適化による Hadoop の I/O 性能向上に関する一考察

中上 誠[‡1]　Jose A. B. Fortes[‡2]　山口 実靖[‡3]

†1, †3 工学院大学大学院 工学研究科 電気・電子工学専攻 〒163-8677 東京都新宿区西新宿 1-24-2

‡2 Advanced Computer and Information Systems (ACIS) Lab University of Florida Gainesville, USA

E-mail: †1 cm19036@ns.kogakuin.ac.jp, †2 fortes@ufl.edu, †3 sane@cc.kogakuin.ac.jp

あらまし ビッグデータ処理などで用いられる重要なプラットフォームに Hadoop があり，その性能向上が重要である．ビッグデータ処理には大規模ストレージとして HDD を用いることも多い．HDD にはゾーン毎に異なる I/O スループットを持つという特性があり，それを考慮してファイルの格納位置を制御することによって大規模 I/O 処理の高速化を行うことが可能であると期待できる．本稿では，Hadoop ジョブの特性を考慮したファイル格納位置の制御による I/O 処理の高速化手法に着目し，性能評価によりその有効性について考察をする．

キーワード Hadoop，ファイルシステム，ストレージ，HDD

1. はじめに

Hadoop は，MapReduce モデルに基づいたビッグデータ処理プラットフォームである[1]．ビッグデータ処理には大容量の記憶媒体が必要となるため，容量単価が低いハードディスクドライブ（HDD）を用いることも多い．HDD はゾーン毎に I/O 性能が異なる[2][3]ので，MapReduce ジョブの特性を考慮し，ジョブが生成するファイルの HDD における格納位置を制御することにより，Hadoop の I/O 性能を向上させることができると考えられる．

過去に，Hadoop が生成するすべてのファイルを空き領域の内最も外周部に格納させることによって Hadoop のシーケンシャル I/O 性能を向上させる手法[2][3][4][5]や，ジョブの特性を考慮してファイル格納位置を制御することにより Hadoop の I/O 性能を向上させる手法が提案されている[6][7][8]．本稿では，Hadoop SWIM [9]を用いてこれらファイル格納位置制御手法の性能を評価する．

本稿の構成は以下の通りである．2 章で関連研究を紹介する．3 章で，Hadoop ジョブの基礎性能の調査，および HDD におけるファイル格納位置とシーケンシャルアクセス速度の関係について説明する．4 章で，Hadoop 性能を向上させるファイル格納位置制御手法について説明する．5 章で，4 章にて紹介したファイル格納位置制御手法の性能評価を行う．6 章にて考察を述べる．7 章にて本稿をまとめる．

2. 関連研究

2.1 MapReduce 処理

図 1 のように MapReduce ジョブの処理は，Map フェーズ，Shuffle フェーズ，Reduce フェーズの 3 つのフェーズで分けられる．Map フェーズでは，まず入力ファイルが Input Split と呼ばれる小さなファイルに分割される．そして，Mapper が Input split を受け取り，Input split に対してユーザーが定義した Map タスクを実行して中間ファイルとなる Key-Value ペアを生成する．Shuffle フェーズでは，mapper が生成した Key-Value ペアの中間ファイルをソートし，Key ごとにグループ化して reducer にファイルを転送する．Reduce フェーズでは，reducer が受け取った Key-Value ペアに対してユーザーが定義した Reduce タスクを実行し，出力ファイルが生成される[10][11][12][13].

2.2 Hadoop SWIM

Hadoop SWIM は，Hadoop を実行した際の履歴からそれを模擬した負荷を生成することができるベンチマークソフトである．SWIM によって生成されたジョブは Submit time seconds，Inter job submit gap seconds，Map input bytes，Shuffle bytes，Reduce output bytes の 5 つのパラメータを持っている[14][15]．SWIM には Facebook システムの負荷を模擬したジョブがあり，本稿ではこの Facebook トレースのジョブの持つ 5 つのパラメータを変更することによって得られるさまざまな MapReduce ジョブを用いて評価を行う．

3. 基礎性能評価

本章にて，SWIM ジョブの I/O 使用率や CPU 使用率などの振る舞いを調査する．これらの情報はジョブの特性を考慮したファイル格納位置制御手法にて必要となる．

3.1 SWIM ジョブの振る舞い

本節にて，Map-heavy，Shuffle-heavy，Reduce-heavy 各ジョブの使用資源(CPU や HDD)の特性について述べる．ジョブの特性を考慮したファイル格納位置制御に

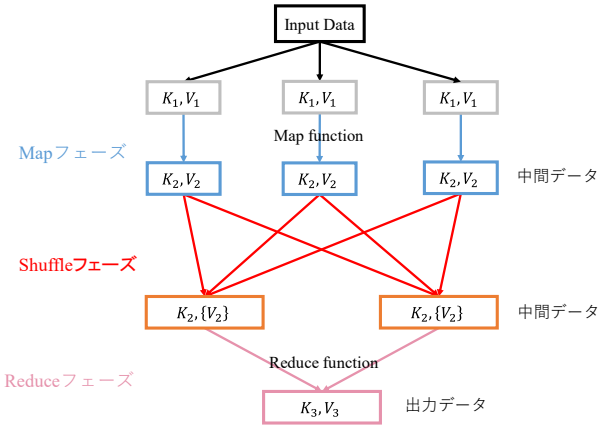

図 1　MapReduce 処理の概要

表 1　実験機の仕様

CPU	AMD Phenom 2 X4 965 Processor
OS	CentOS 6.10 x86_64 minimal
カーネル	Linux 2.6.32.57
メインメモリ	4GB
HDD	500GB(ext3)
Hadoop Ver.	2.0.0-cdh4.2.1

表 2　測定用 HDD の仕様

型番	DT01ACA050
インタフェース	SATA 3.0
インタフェーススピード	6.0Gbps
容量	500GB
バッファサイズ	32MB
回転数	7200rpm

必要な情報である I/O 使用率，CPU 使用率，ディスク使用量の推移を調査した．調査時のジョブのパラメータを以下に示す. Submit time と inter job submit gap は，すべてのジョブで 1 と設定し，Input file size は 20GB とした．Map input bytes を 10^{12}，Shuffle bytes と Reduce output bytes を 1 としたジョブを Map-heavy とし，Shuffle bytes を 10^{12}, その他のパラメータを 1 としたもの Shuffle-heavy とし．Reduce output bytes を 10^{12}，その他のパラメータを 1 としたものを Reduce-heavy とした. 計算機および測定用 HDD の仕様をそれぞれ表 1 および表 2 に示す.

Map-heavy，Shuffle-heavy．Reduce-heavy 実行時の平均 I/O 使用率と平均 CPU 使用率を図 2 に，実行時の最大ディスク使用量と実行後のディスク使用量を図 3 に示す．これらの結果から，Map-heavy は CPU バウンドであり，実行時に生成されるファイルは一時的ファイルであり実行中にほぼ全てのファイルが削除されることがわかる．Shuffle-heavy は I/O バウンドであり，実行中に生成されるファイルは一時的ファイルであることがわかる．Reduce-heavy は I/O バウンドであり，実行時に生成されるファイルは削除されず恒久的に残ることがわかる

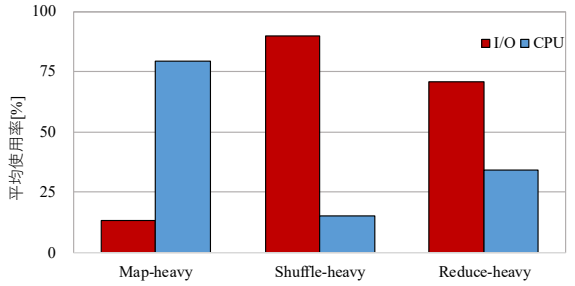

図 2　各ジョブの平均 I/O・CPU 使用率[8]

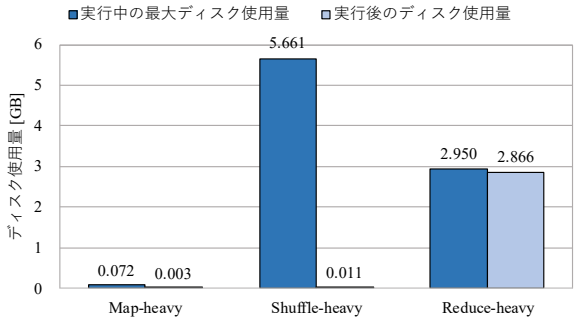

図 3　各ジョブの最大ディスク使用量と実行後のディスク使用量[8]

3.2　HDD のシーケンシャルアクセス性能

測定用 HDD に対して先頭のアドレスから最後のアドレスまで 64MB ずつ読込/書込を行うプログラムを実行し，HDD の各ゾーンにおけるシーケンシャル読込/書込速度を測定した．先頭アドレスはディスク最外周部ゾーンに最後のアドレスはディスク最内周部ゾーンにそれぞれ対応している．図 4 にゾーン毎のシーケンシャル読込/書込速度の測定結果を示す．この図から，ディスク最外周部ゾーンのシーケンシャル読込/書込速度はディスク最内周部と比較すると約 2 倍の速度であることがわかる．

3.3　SWIM ジョブのディスクアクセスのシーケンシャル性

ジョブ実行中の時間的かつ空間的に連続した I/O 要求を 1 つの要求として結合することによって得られる結合 I/O サイズ[16][17][18][19][20]について調査した．ジョブが I/O バウンドでありかつ結合 I/O サイズが大きい場合は，ファイルを外周部ゾーンに配置すること

により I/O 性能を向上することができると期待できる．Map-heavy ジョブは CPU バウンドであり，ファイル格納位置を最適化することによって性能を向上することはできないと予想される．本稿では，Shuffle-heavy ジョブと Reduce-heavy ジョブの結合 I/O サイズを調査した．図 11 と図 12 に，Shuffle-heavy ジョブと Reduce-heavy ジョブの結合 I/O サイズを示す．図より，Shuffle-heavy ジョブ，Reduce-heavy ジョブともに結合 I/O サイズの大きい I/O 要求が多く発行されており，HDD が非常に高いシーケンシャル性でアクセスされていることがわかる．したがって，これらのジョブに HDD の外側部ゾーンを積極的に利用させることによってシーケンシャル I/O 速度を向上させることができると予想される．

4. ファイル格納位置制御手法

本章にて Hadoop の I/O 性能を向上させるファイル格納位置制御手法[2][7][8]について説明する．

4.1 通常手法

ファイルの格納位置の制御を行っていない状態を我々は通常手法と呼んでいる．通常手法の時，Hadoop ジョブが生成するファイルはディスクの空き領域中の内周部側に格納されたり，外周部側に格納されたりする．そのため，高速スループット領域を活用することができていないことがある．

4.2 外周部優先格納手法

この手法[2]はファイルを常にディスク空き領域中の最外周部領域に格納するように制御することによって，Hadoop が常に最もシーケンシャル I/O 性能の良い部分を活用できるようにする．この手法は ext2/3/4 ファイルシステム[21][22][23]を用いて実装されている．これらのファイルシステムでは，図 7 のようにディスクは 4KB を 1 つのブロックとして管理され，複数のブロックを集めてブロックグループを構成する．ブロックグループ毎にブロックビットマップ，inode ビットマップ，inode テーブル，データブロックが用意されている．この内，ブロックビットマップはブロックグループ内の各データブロックが使用中であるか否かを管理している．この手法では，ブロックビットマップを書き換えディスク最外周部領域以外の領域をすべて使用中という情報に書き換えることによって，内周部にファイルが格納されることを回避する．そして，ファイルが格納され使用可能領域が少なくなると，使用禁止領域のブロックビットマップを元の状態に戻し，使用可能領域を閾値する．この手法にはディスクの空き領域サイズを監視する機能，空き領域の容量が指定された閾値を下回ったときに空き領域を動的に拡張する機能，空き領域の容量が指定された閾値を上回ったとき

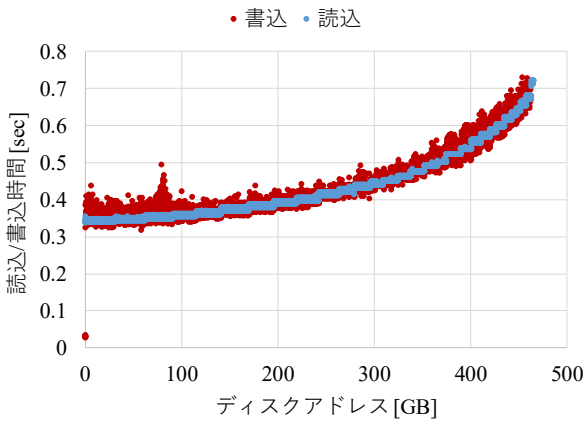

図 4　HDD のゾーン毎のシーケンシャル読込/書込速度[7]

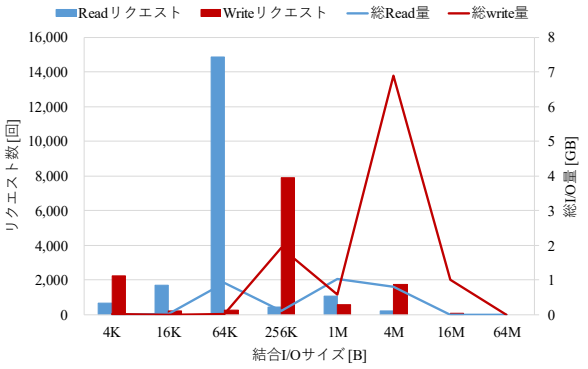

図 5　Shuffle-heavy の結合 I/0 サイズ[7]

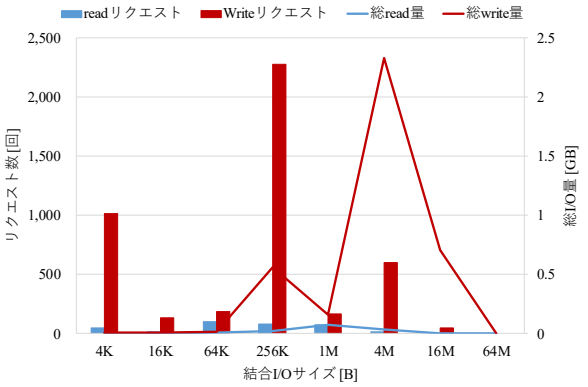

図 6　Reduce-heavy の結合 I/0 サイズ[7]

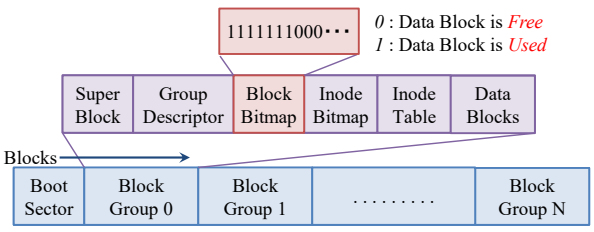

図 7　ext2/3/4 の概要

に空き領域を動的に縮小する機能がある．空き領域の動的拡張機能は，空き領域の監視機能により使用可能領域の容量が指定された閾値を下回ったときに起動され，使用禁止領域内のブロックをディスクの外周側から順に使用可能領域の容量が閾値を上回るまで使用可能状態へと変更していく機能である．空き領域の動的縮小機能は監視機能により使用可能領域の容量が指定された閾値を上回ったときに起動され，使用可能領域内のブロックをディスクの内周側から順に使用可能領域の容量が閾値を下回るまで使用禁止状態へと変更していく機能である．

実行されるジョブが 1 つの場合は，この手法を用いてディスク最外周部領域にジョブのファイルを格納させるのが最も良いと考えられる．対して，複数のジョブが実行される場合はジョブ特性を考慮してファイル格納位置を制御した方が良いと考えられる．

4.3 ジョブ特性を考慮したファイル格納位置制御手法

この手法[7]複数種類のジョブが順次実行される場合に有効である．この手法では，HDD を 2 つのエリアに分ける。そして，以下の優先順位に従ってジョブのファイルを外周部側エリアに格納するか，内周部側エリアに格納するかを決める．

（1）ジョブのファイルが一時的ファイルであり，実行時 I/O バウンドである

（2）ジョブのファイルが一時的ファイルであり，実行時 I/O バウンドでない

（3）ジョブのファイルが恒久的ファイルであり，実行時 I/O バウンドである

（4）ジョブのファイルが恒久的ファイルであり，実行時 I/O バウンドでない

本稿では，(1)，(2)に該当するジョブのファイルを外周部側エリアに，(3)，(4)に該当するジョブのファイルを内周部側エリアに格納する．

今後アクセスされる可能性が低い恒久的ファイルがディスク外周部に格納されると，他アプリケーションはその領域を使うことができなくなる．当該手法では一時的ファイルを優先的に外周部側エリアに格納させることによって，複数回高速スループット領域を活用することができるようにする．

この手法はファイルを外周部側エリアに格納するか，内周部側エリアに格納するかの制御のみをおこなっており，外周部側エリア中の外周部側領域と，内周側エリア中の外周部側領域を活用する考慮がされていない．

4.4 外周部の活用とジョブ特性を考慮したファイル格納位置制御手法

この手法[8]では 4.3 節で紹介した優先順位に従って，ジョブのファイルを外周部側エリアに格納するか，内周部側エリアに格納するかの制御をおこなうのに加え，各エリアの空き領域の内最も外周部側の領域に必ずファイルを格納するように制御している．

4.2 節で紹介したディスクの空き領域サイズを監視する機能，空き領域の容量が指定された閾値を下回ったときに空き領域を動的に拡張する機能，空き領域の容量が指定された閾値を上回ったときに空き領域を動的に縮小する機能を 4.3 節の手法に加えることによって実装している．

5. 性能評価

本章にてファイル格納位置制御手法の性能評価を行う．本稿では Map-heavy, Shuffle-heavy, Reduce-heavy のジョブが順次連続して実行されるケースを取り扱う．ジョブを連続して実行しているときに，外周部優先活用手法を適用した時の動作を図 8 に示す．この手法を適用すると，Reduce-heavy のファイルが格納された後，Reduce-heavy のファイルが格納されている領域の後ろの領域に他ファイルが格納される．そのため，Reduce-heavy のファイルが多く格納されると，全てのジョブはディスク内周部側を使わなければならなくなる．次に，ジョブ特性を考慮したファイル格納位置制御手法を適用した時の動作を図 9 に示す．この手法では，ファイルが恒久的である Reduce-heavy のファイルは内周部側エリアに，ファイルの多くが一時的である Map-heavy, Shuffle-heavy のファイルは外周部側エリアに格納させる．そのため，外周部側エリアに格納されたファイルのほとんどはジョブ実行後に削除されるので，Map-heavy と Shuffle-heavy は外周部側エリアを複数回活用することができる．ただし，こちらの手法は外周部側エリアと内周部側エリアの外周部側領域を積極的に活用することができていないという問題点がある．図 10 に外周部とジョブ特性を考慮したファイル格納位置制御手法の動作を示す．この手法では，図 9 の問題点が改善され，各エリアの外周部側領域を積極的に活用することができている．

これら手法の性能評価のために，ジョブセットを実行し，実行時間を測定した．図 11 に示すジョブセット 1 は，Map-heavy グループ（20 個の Map-heavy ジョブで構成）と，Shuffle-heavy グループ（20 個の Shuffle-heavy ジョブで構成）と，Reduce-heavy グループ（20 個の Reduce-heavy ジョブで構成）が繰り返し連続して実行するものである．各ジョブグループは 9 回ずつ，合計 27 回実行される．ジョブセットを実行すると，HDD のほどんどが使用された状態になるようにしている．各ジョブのパラメータ設定は 3.1 節のものと同様である．

ジョブセット 1 を 5 回実行した時の平均実行時間を図 12 に示す．性能評価の結果，外周部優先活用手法を

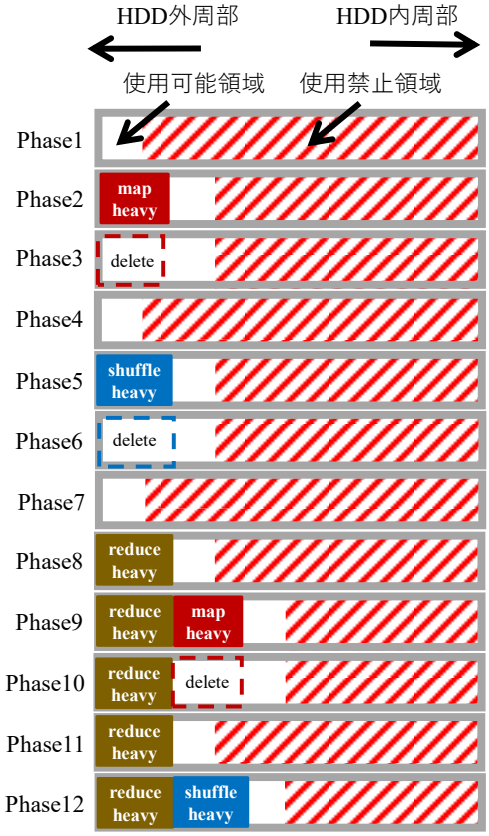

図 8　外周部優先格納手法の動作

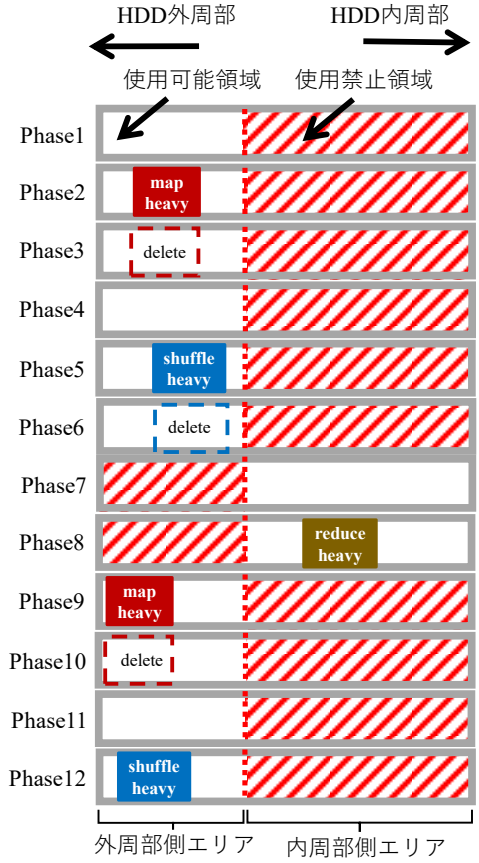

図 9　ジョブ特性を考慮したファイル格納位置制御手法の動作

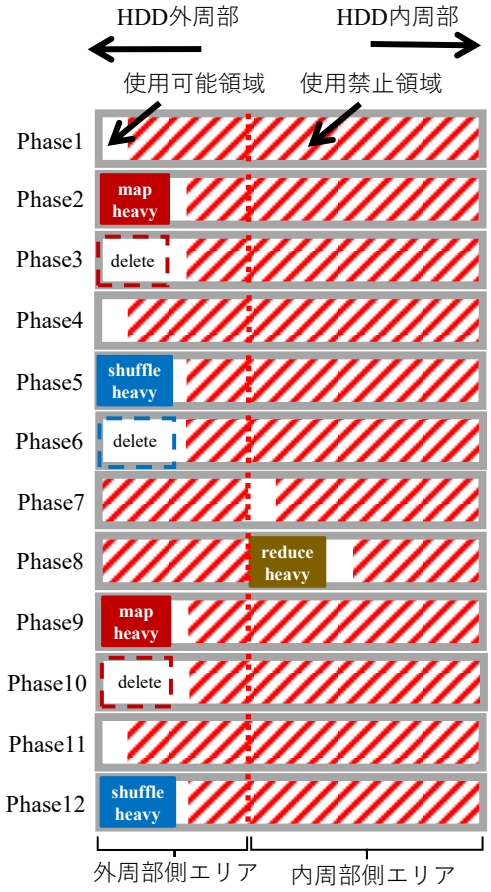

図 10　外周部の活用とジョブ特性を考慮したファイル格納位置制御手法の動作

適用した時，通常時から 4.8%，ジョブ特性を考慮した手法を適用した時，通常時から 13.2%，外周部とジョブ特性を適用した時 15.4%実行時間を短縮することができた．

図 13 にジョブセット 2 を示す．ジョブセット 2 の Map-heavy グループは Map input bytes が $10^{10.5}$，10^{11}，$10^{11.5}$，10^{12}，$10^{12.5}$のジョブがそれぞれ 4 つずつ，合計 20 個の Map-heavy ジョブで構成されている．Shuffle-heavy グループ，Reduce-heavy グループについても同様である．ジョブグループは 3 回ずつ，合計 9 回実行される．こちらのジョブセットでも，実行後に HDD のほとんどが使用された状態になるようにしている．

図 14 にジョブセット 2 を 5 回実行した時の平均実行時間を示す。性能評価の結果，外周部優先活用手法を適用した時，通常時から 3.4%，ジョブ特性を考慮した手法を適用した時，通常時から 6.1%，外周部とジョブ特性を適用した時 13.0%実行時間を短縮することができた．

図 15 にジョブセット 3 を示す．ジョブセット 3 の Map-heavy グループは Map input bytes が $10^{10.5}$，10^{11}，$10^{11.5}$，10^{12}，10^{12}のジョブがそれぞれ 5 つずつ，合計

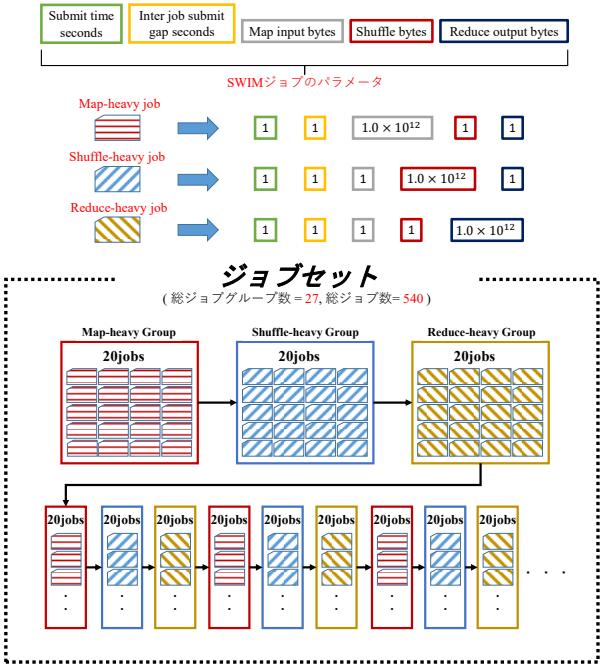

図 11　ジョブセット 1 の概略

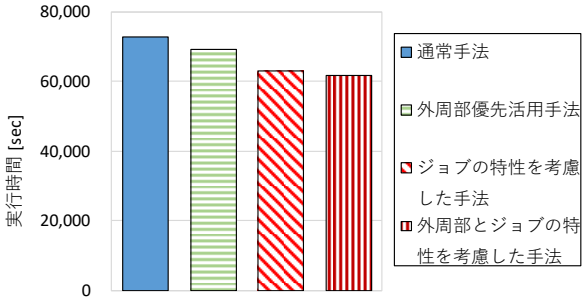

図 12　ジョブセット 1 の実行時間

20 個の Map-heavy ジョブで構成されている．Shuffle-heavy グループ，Reduce-heavy グループについても同様である．ジョブグループは 5 回ずつ，合計 15 回実行される．

図 16 にジョブセット 3 を 5 回実行した時の平均実行時間を示す。性能評価の結果，外周部優先活用手法を適用した時，通常時から 3.2%，ジョブ特性を考慮した手法を適用した時，通常時から 5.5%，外周部とジョブ特性を適用した時 9.0%実行時間を短縮することができた．

ジョブのパラメータが全て10^{12}（ジョブセット 1）であったときは，ジョブ特性を考慮した手法と外周部とジョブ特性を考慮した手法の性能差は小さかったが，ジョブのパラメータが複数ある場合（ジョブセット 2,3）は，両者の性能差が大きくなっている．外周部エリアと内周部エリアの境界は，一時的ファイルを生成するジョブの最大ディスク使用量をもとに決める．ジョブセット 1 実行時にジョブ特性を考慮した手法を適用し

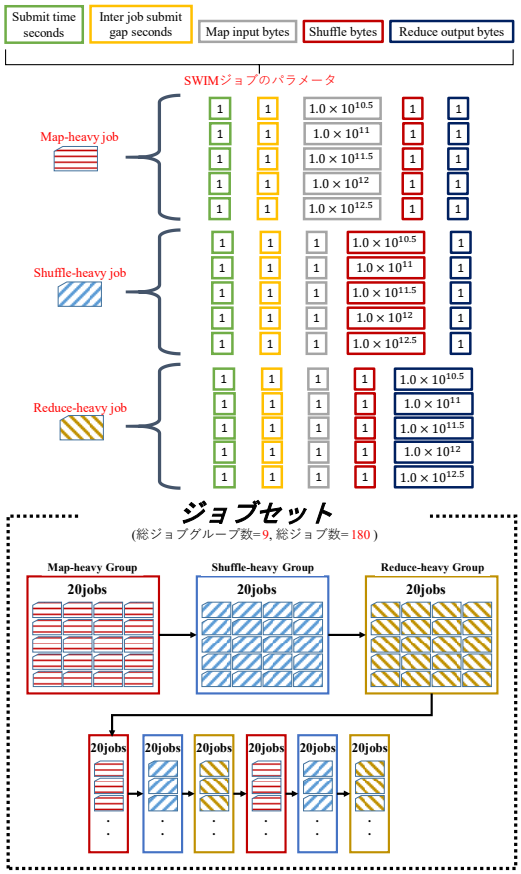

図 13　ジョブセット 2 の概略

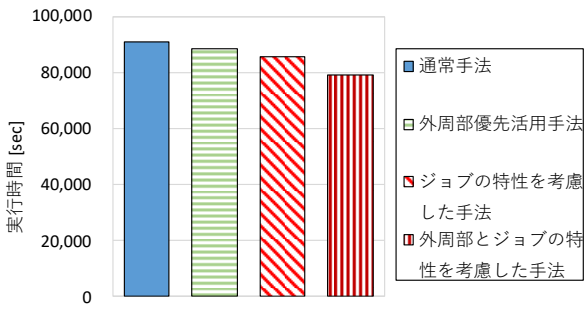

図 14　ジョブセット 2 の実行時間

た場合，Shuffle-heavy(Shuffle bytes = 10^{12})の最大ディスク使用量から境界を決めるため，全ての Shuffle-heavy が外周部エリアを有効的に活用することができると考えられる．しかし，ジョブセット 2,3 実行時にジョブ特性を考慮した手法を適用した場合，パラメータサイズが最も大きい Shuffle-heavy の最大ディスク使用量から境界を決めるため，パラメータサイズの小さい Shuffle-heavy は外周部エリアの内周側領域のみを活用してしまう可能性がある．そのため，外周部を活用する考慮をした手法との性能差が大きくなったと考えられる．

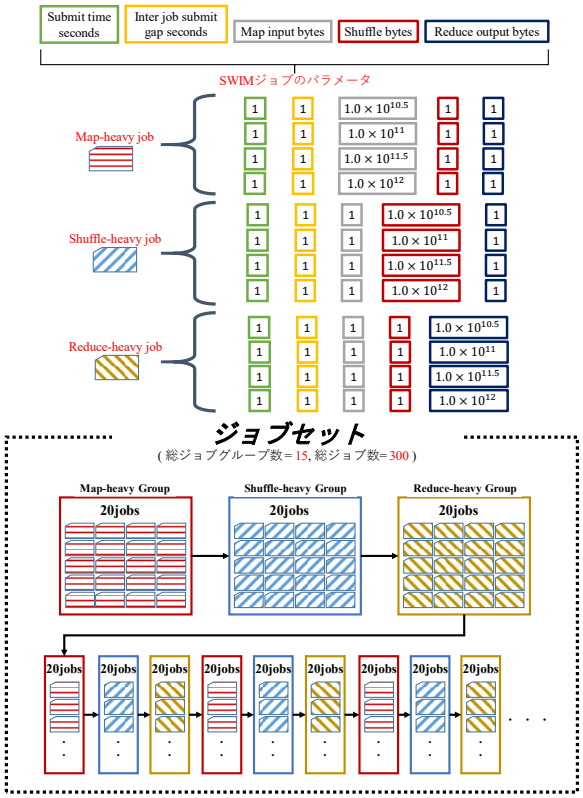

図 15　ジョブセット 3 の概略

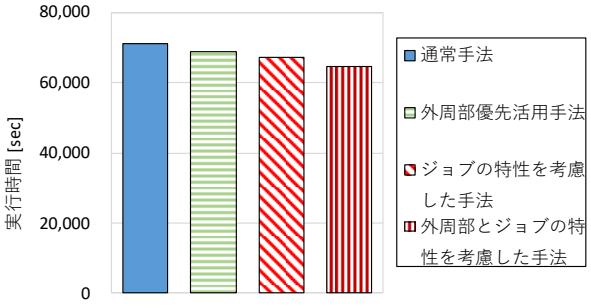

図 16　ジョブセット 3 の実行時間

6. 考察

本稿で提案した手法は，実行される Hadoop ジョブの特性，すなわち CPU や I/O の使用率やファイルが一時的か恒久的であるかといったジョブの特性が既知であることを前提としている．本稿で用いた SWIM 以外のアプリケーションであっても類似の特性を持つジョブが繰り返し実行されるようなアプリケーションであれば，本稿の手法を利用することができると考えられる．例えば，検索エンジンシステムは頻繁にインデックスを更新するが，これは Shuffle-heavy の特性と似ている．ショッピングサイトでも同一のオンライントランザクション処理（OLTP）ジョブが毎日実行される．オンライン分析処理（OLAP）アプリケーションの場合も，繰り返すジョブの特性は非常に似ている．よって，それぞれのジョブの特性を調査して本稿の提案手法を用いることができると考えられる．また，我々が提案した手法が必要とするジョブの特性は容易に得ることができる．I/O 使用率と CPU 使用率はそれぞれ iostat コマンドと vmstat コマンドによって求めており，ディスク使用量の推移は df コマンドを繰り返すことによって求めている．したがって，我々が提案した手法が適用可能である状況は多くあると考えられる．

ディスク外周部に一時的なファイルのみを格納させる方法として，ディスク外周部のパーティションを作成しそのパーティション領域に一時的ファイルを格納させる実装方法も考えられる．しかし，この実装方法ではパーティション領域の内どの領域にファイルが格納されるか制御できない．一方，外周部とジョブ特性を考慮した手法は，領域内のどの部分にファイルが格納されるかも制御することができる．また，ビットマップを変更することによって簡単に外周部の領域と内周部の領域のサイズを動的に変更することができる．よって，より多くの状況に適用できると言える．

外周部優先活用手法及び外周部とジョブ特性を考慮したファイル格納位置制御手法における空き領域サイズ監視機能のオーバーヘッドについて考察を行う．監視プロセスは定期的にファイルシステムの空きブロック数情報を監視しているが，この情報は一度アクセスすると常にページキャッシュに格納される．よって，監視機能の I/O 性能への影響は極めて小さいと考えられる．また，監視プロセスの CPU 使用率は平均 0.4% であった．よって，CPU 性能への影響も極めて小さいと考えられる．

7. おわりに

本稿では，SWIM ジョブを Map-heavy ジョブ，Shuffle-heavy ジョブ，Reduce-heavy ジョブに分類し，それらのジョブの I/O 使用率，CPU 使用率，実行ファイルサイズといった特性を調査した．そして，ジョブの特性を考慮してファイル格納位置を制御することによって I/O 性能を向上させる手法を改善した手法を提案した．また，性能評価により，提案手法は既存手法の性能を向上させることを確認した．

今後は，実行時に（オンラインで）ジョブの調査を行い，事前調査を行わない手法についての考察を行う予定である．

謝辞

本研究は JST，CREST JPMJCR1503 の支援を受けたものである．

本研究は JSPS 科研費 17K00109, 18K11277 の助成を受けたものである．

本研究は NSF ACI 1550126 and supplement DCL NSF 17-077 の支援を受けたものである.

参 考 文 献

[1] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107-113. DOI: https://doi.org/10.1145/1327452.1327492

[2] Eita Fujishima and Saneyasu Yamaguchi. 2016. Dynamic File Placing Control for Improving the I/O Performance in the Reduce Phase of Hadoop. In Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication (IMCOM '16). ACM, New York, NY, USA, Article 48 , 7 pages. DOI: http://dx.doi.org/10.1145/2857546.2857595

[3] Eita Fujishima and Saneyasu Yamaguchi, "Improving the I/O Performance in the Reduce Phase of Hadoop," 2015 Third International Symposium on Computing and Networking (CANDAR), Sapporo, 2015, pp. 82-88. doi: 10.1109/CANDAR.2015.24

[4] Makoto Nakagami，Joichiro Kon，Gil Jae Lee，Jose A.B. Fortes，Saneyasu Yamaguchi，“File Placing Location Optimization on Hadoop SWIM，” 2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)，Takayama，2018，DOI: 10.1109/CANDARW.2018.00100

[5] 近丈一郎，中上誠，Jose A.B. Fortes，山口実靖，“ファイル格納位置制御による大規模 I/O 性能の向上に関する一考察，” DEIM Forum 2019 J4-3

[6] 中上 誠, Jose A. B. Fortes, 山口 実靖, "特性を考慮した Hadoop ジョブの I/O 性能の向上", 情報処理学会 研究報告システムソフトウェアとオペレーティング・システム（OS）, 2019-OS-146, 13, pp. 1-8

[7] Makoto Nakagami，Jose A.B. Fortes，Saneyasu Yamaguchi，“Job-aware Optimization of File Placement in Hadoop，” BDCAA 2019 The 1st IEEE International Workshop on Big Data Computation，analysis，and Applications，COMPSAC 2019，July 2019.

[8] Makoto Nakagami，Jose A.B. Fortes，Saneyasu Yamaguchi，“Usable Space Control Based on Hadoop Job Features，” 2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW)，Takayama，2019

[9] GitHub-SWIMProjectUCB/SWIM: Statistical Workload Injector for MapReduce - Project at UC Berkeley AMP Lab, https://github.com/SWIMProjectUCB/SWIM

[10] 室航，廣野将行，松本隼，佐藤丈博，岡本聡，山中直明，“Hadoop におけるシャッフルヘビータスク判別手法の提案”，フォトニックネットワーク研究会，信学技報，vol. 116，no. 498，PN2016-90，pp. 37-44，2017 年 3 月

[11] Faraz Ahmad, Seyong Lee, Mithuna Thottethodi and T. N. Vijaykumar, ” MapReduce with Communication Overlap (MaRCO)”, Journal of Parallel and Distributed Computing, May 2013, Pages 608-620 ,DOI: https://doi.org/10.1016/j.jpdc.2012.12.012

[12] Navya Francis, Sheena Kurian K,” Data Processing for Big Data Applications using Hadoop Framework ”, nternational Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 3, March 2015, DOI: 10.17148/IJARCCE.2015.4343

[13] Vikram A. Saletore, Karthik Krishnan, Vish Viswanathan, and Matthew E. Tolentino, “ HcBench: Methodology, Development, and Full-System Characterization of a Customer Usage Representative Big Data/Hadoop Benchmark ”, T. Rabl et al. (Eds.): WBDB 2013, LNCS 8585, pp. 73–93, 2014. DOI: 10.1007/978-3-319-10596-3_7

[14] Yanpei Chen, Archana Ganapathi, Rean Griffith, Randy Katz,” The Case for Evaluating MapReduce Performance Using Workload Suites ”, 2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems , July. 2011 ,10 pages, DOI: 10.1109/MASCOTS.2011.1

[15] Yanpei Chen, Sara Alspaugh, Randy Katz,” Interactive Analytical Processing in Big Data Systems: A Cross-Industry Study of MapReduce Workloads “, Proceedings of the VLDB Endowment (PVLDB), Vol. 5, No. 12, pp. 1802-1813 (2012)

[16] Eita FUJISHIMA Kenji NAKASHIMA Saneyasu YAMAGUCHI, "Hadoop I/O Performance Improvement by File Layout Optimization", IEICE TRANSACTIONS on Information and Systems, Vol.E101-D No.2 pp.415-427, doi: 10.1587/transinf.2017EDP711

[17] S. Yamaguchi, M. Oguchi and M. Kitsuregawa, "Trace system of iSCSI storage access," *The 2005 Symposium on Applications and the Internet*, Trento, Italy, 2005, pp. 392-398. doi: 10.1109/SAINT.2005.65

[18] Saneyasu Yamaguchi, Masato Oguchi, Masaru Kitsuregawa, "iSCSI analysis system and performance improvement of sequential access in a long-latency environment," Electronics and Communications in Japan (Part III: Fundamental Electronic Science), Volume 89, Issue 4, Pages 55-69, Wiley Subscription Services, Inc., A Wiley Company, April 2006. DOI: 10.1002/ecjc.20238

[19] Yuta Nakamura, Kyosuke Nagata, Shun Nomura, and Saneyasu Yamaguchi. 2014. I/O scheduling in Android devices with flash storage. In Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication (ICUIMC '14). ACM, New York, NY, USA, Article 83 , 7 pages. DOI: https://doi.org/10.1145/2557977.2558025

[20] Joichiro Kon，Kenji Nakashima and Saneyasu Yamaguchi，“A Deletion Aware Usable Space Control for SD2，” 2017 Fifth International Symposium on Computing and Networking (CANDAR)，Aomori，2017，DOI: 10.1109/CANDAR.2017.31

[21] R.Card and T.Ts’o and S.Tweedle, “Design and Implementation of the Second Extended Filesystem,” First Dutch International Symposium on Linux, 1994

[22] Theodore Y. Ts'o, Stephen Tweedie, “ Planned Extensions to the Linux Ext2/Ext3 Filesystem ”, Conference: Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference, June 10-15, 2002, Monterey, California, USA

[23] Avantika Mathur, Mingming Cao, Suparna Bhattacharya, “ The new ext4 filesystem: current status and future plans ”, Proceedings of the Linux Symposium, Volume Two, June 27th–30th, 2007, Ottawa, Ontario Canada