

HDD 上の Ext4 におけるシーケンシャルの 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^{†1}

概要: IoT の普及などにより、ビッグデータと呼ばれる巨大なデータの蓄積や解析に注目が集まっている。ビッグデータの蓄積や解析には大規模ストレージとして HDD を用いることも多く、その I/O 処理の高速化が重要となる。過去に Ext3 ファイルシステムにおいてファイルの HDD 内における格納位置を制御することによって、HDD におけるシーケンシャル I/O スループットを大きくする手法が提案されている。本稿では、まず Ext4 ファイルシステムがファイル削除によって空いた領域を積極的に活用せず HDD の低速領域を活用してしまうという問題を持つことを示す。そして、Ext4 にてファイル格納位置を制御する手法を実装し、性能評価によりその有効性を示す。

キーワード: Ext4, Filesystem, Sequential access

1. はじめに

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。そのため、HDD 上の I/O 処理の高速化は重要である。過去に、ext3 ファイルシステムにてファイル格納位置を制御することによって、Hadoop のシーケンシャル I/O 性能を向上させる手法[1][2][3]が提案されている。この手法では、HDD のゾーン毎に I/O 性能が異なる[1][2][3]という特徴を生かし、Hadoop にディスク外周部を積極的に活用させることによって、Hadoop の I/O 性能を向上させている。

また、著名なファイルシステムに ext4 がある。ext4 にはファイル削除によって生まれた空き領域を積極的に活用しないという特徴がある。そのため、Hadoop のような一時的ファイルを多く生成するビッグデータ処理基盤においては、ext4 のブロックアロケーションの特徴が性能に大きく影響すると考えられる。本稿では、ext4 のファイル削除によって生まれた空き領域を活用しないという特徴を示し、それを改善する手法を提案する。

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、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や ext4 の問題点について説明する。4 章で、ext4 におけるファイル格納位置制御手法を提案する。5 章で、ローカルファイルシステム上での提案手法の効果の検証や、HDFS (Hadoop Distributed Filesystem) 上での効果の検証を行う。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 ファイル格納位置制御手法

過去にファイル格納位置を制御することによって、

Hadoop のシーケンシャル I/O 速度を向上させる手法が提案されている[1][2]。この手法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、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 ext3 ファイルシステム[4][5]を用いて実装されている。Ext3 では、図 1 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それぞれ 1 と 0 で管理している。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、常にディスクの空き領域を監視し、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、使用可能領域を拡張する。また、ファイルが削除され使用可能領域が余剰になると、使用可能領域の内最も内周部の領域のブロックビットマップを使用中という情報に書き換える。

また、Hadoop ジョブの特徴に基づいてファイル格納位置を制御する手法も過去に提案されている[6][7][8][9][10]。この手法は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る
- (2) ジョブのファイルが恒久的ファイルである
- (1)に該当するジョブのファイルを高速な外周部側エリアに、(2)に該当するジョブのファイルを低速な内周部側エ

^{†1} 工学院大学大学院 工学研究科 電気・電子工学専攻
Electrical Engineering and Electronics, Kogakuin University Graduate School
^{†2} Advanced Computer and Information Systems (ACIS) Lab University of Florida

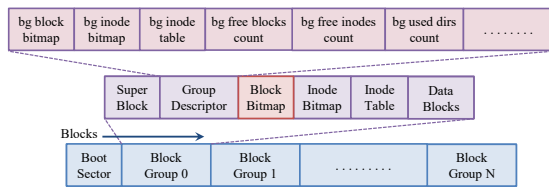


図1 ext3のディスク内レイアウトの概要

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している。

2.2 Hadoop と MapReduce

Hadoop は著名なビッグデータ処理基盤の一つであり、MapReduce モデル[11][12][13][14][15][16]の処理が行われ、データはHDFS (Hadoop Distributed Filesystem)というファイルシステムに保存される。MapReduce 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。

3. 基礎性能調査

3.1 HDD のシーケンシャルアクセス性能

測定用 HDD の、最初のアドレスから最後のアドレスまで 64MB の読込/書込を行うプログラムを実行し、HDD のアドレス（ゾーン）とシーケンシャルリード/ライト速度の関係を調査した。実験機の仕様を表 1 に、測定用 HDD の仕様を表 2 に示す。最初と最後のアドレスは、それぞれ最外側のゾーンと最内側のゾーンに対応する。図 2 にデバイスファイルに直接読込したときのシーケンシャルリード速度を、図 3 にデバイスファイルに直接書込したときのシーケンシャルライト速度の結果を示す。結果から、最外周部のゾーンの速度は最内周部のゾーンの速度の約 2 倍となっていることがわかる。

図 2, 3 より最外周（最速）部におけるシーケンシャルリードライト速度は約 190MB/sec であり、最内周（最遅）部における速度は約 90MB/sec であることが分かる。

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

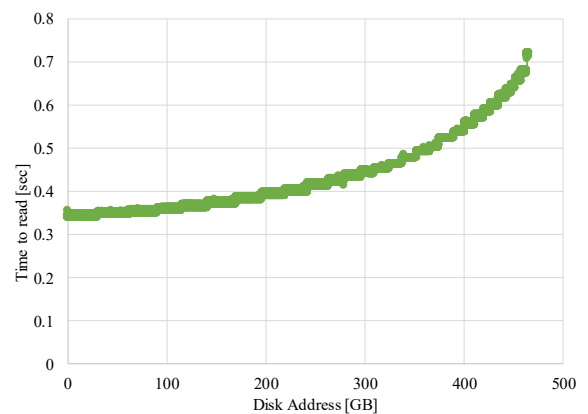


図 2 HDD のゾーン毎のシーケンシャル read 速度 (デバイスファイル)

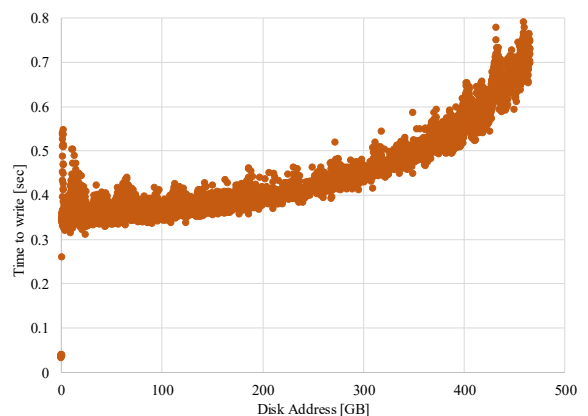


図 3 HDD のゾーン毎のシーケンシャル write 速度 (デバイスファイル)

3.2 dd コマンドによる各ローカルファイルシステムにお

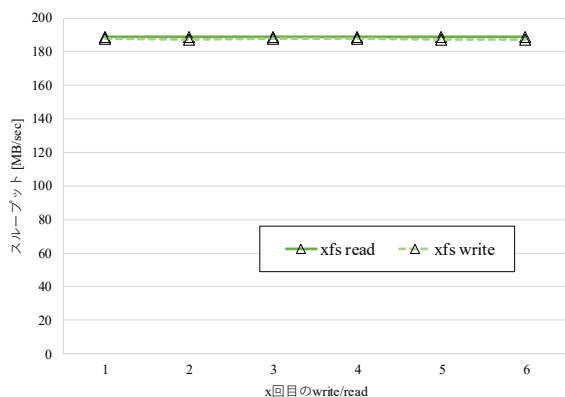


図4 XFS における read/write 速度 (dd command)

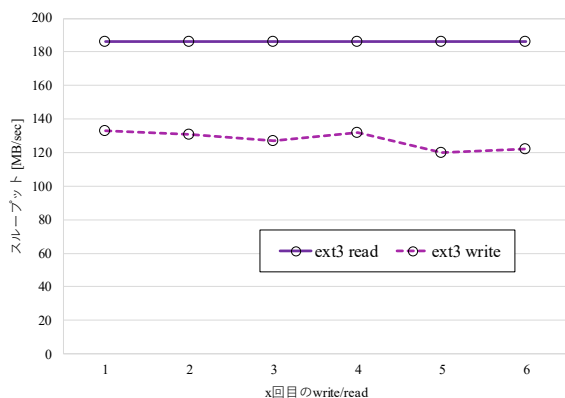


図5 ext3 における read/write 速度 (dd command)

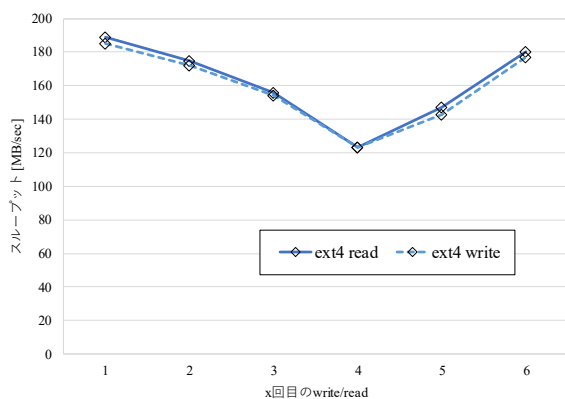


図6 ext4 における read/write 速度 (dd command)

けるシーケンシャル I/O 性能の評価

dd コマンドにて XFS, ext3, ext4 のシーケンシャルアクセス性能を調査した。この時、「100GB のファイルを write→100GB のファイルを read→ファイル削除」という動作を6回繰り返した。

XFS と ext3 と ext4 における6回の各 read/write のスループットをそれぞれ図4と図5と図6に示す。

図4から、XFS の read/write スループットは約 190MB/sec で一定となっていることがわかる。図5から、ext3 の read スループットは約 180MB/sec で一定に、write スループットは約 130MB/sec で一定となっていることがわかる。図6か

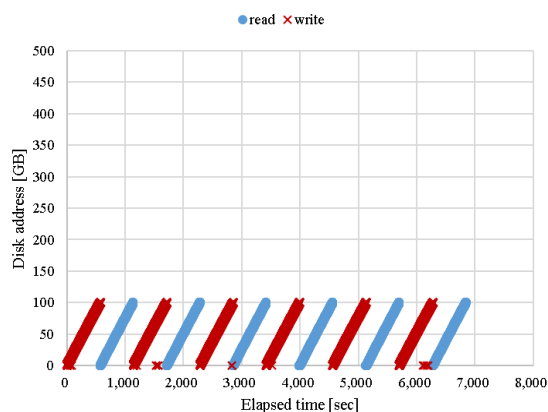


図7 XFS における I/O アドレス (dd command)

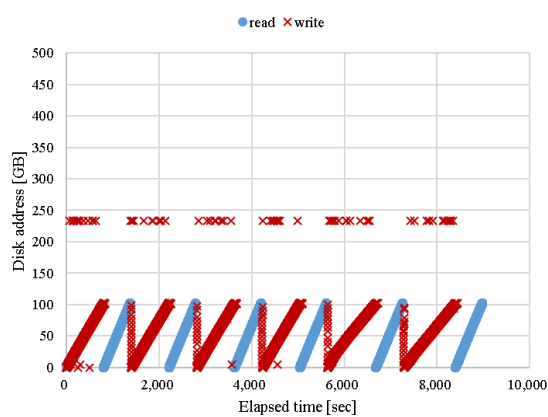


図8 ext3 における I/O アドレス (dd command)

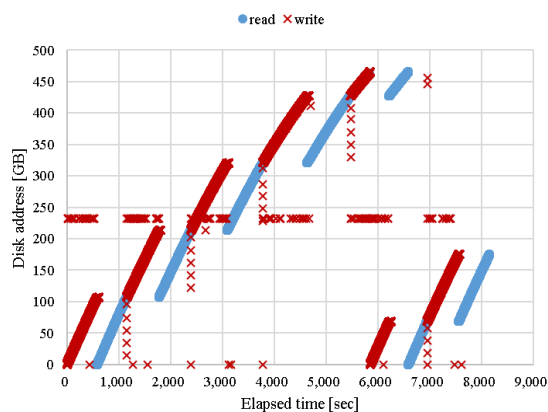


図9 ext4 における I/O アドレス (dd command)

ら、ext4 では read/write スループット共に実行回数によって変動しており、read/write スループットは最大で約 190MB/sec, 最小で約 120MB/sec となっていることがわかる。

3.3 各ローカルファイルシステムにおけるファイル格納位置についての調査

カーネル監視ツール[17][18][19]を用いて、dd コマンドによる read 処理と write 処理において I/O 要求が発行され

たディスクアドレスを監視した。XFS と ext3 と ext4 での測定において発行された I/O 要求をそれぞれ図 7 と図 8 と図 9 に示す。

図 7 と図 8 から、XFS と ext3 では全てのファイルがディスク外周部のアドレス約 100GB の領域に格納されていることがわかる。これが、複数回ファイル読み書きと削除を行ってもスループットが一定になる要因であると考えられる。

図 9 から、ext4 ではアドレス 0~100GB に格納されたファイルが削除された後、ファイル削除によって生まれた空き領域を活用せずアドレス 100GB~200GB にファイルを格納していることがわかる。Ext4 ではファイル削除によって生まれた空き領域を積極的に活用せずディスク内周部を活用してしまうため、複数回ファイル読み書きと削除を行うとスループットが変動すると考えられる。

3.4 TeraSort のディスクアクセスのシーケンシャル性について

TeraSort は Hadoop のサンプルアプリケーションの一つで、1 行 100B からなる複数行のテキストデータをソートするサンプルベンチマークツールである。3.3 節で紹介したカーネル監視ツールを用いて、TeraSort 実行中に発行された I/O 要求を監視した。測定環境は 3.1 節の環境と同様のものを用い、ファイルシステムは ext4、TeraSort の Input ファイルサイズは 24GB とし、Hadoop は疑似分散モードにて実行した。

図 10 に TeraSort 実行中の結合 I/O サイズを示す。結合 I/O サイズとは、カーネル監視ツールにて観察した I/O 要求のうち、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したもの[17][18][19]である。図 10 から、TeraSort 実行中は Hadoop は非常に高いシーケンシャル性でディスクにアクセスしていることがわかる。

3.5 TeraSort の中間ファイルと恒久ファイルの関係について

df コマンドを定期実行することによって、TeraSort 実行中のディスク使用量を監視した。図 11 に TeraSort 実行中に生成された全てのファイルのサイズと TeraSort 実行後に残った全てのファイルのサイズを示す。図 11 から TeraSort 実行中には合計 113GB のファイルが生成されたが、その内 90GB (80%) のファイルが中間ファイルであり実行後には削除されていることがわかる。3.2, 3.3 節で紹介したように、ext4 にはファイル削除によって空いたディスク外周部領域を積極的に活用しないという問題があるため、TeraSort のような多くの一時的ファイルを生成するジョブを連続実行する場合、そのシーケンシャル I/O 性能が大きく低下すると考えられる。

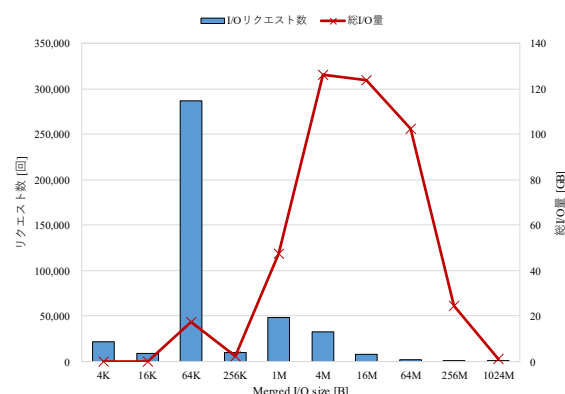


図 10 TeraSort の結合 I/O サイズ (ext4)

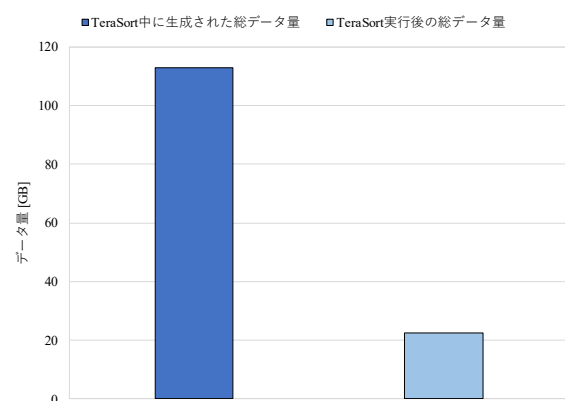


図 11 TeraSort の総生成ファイル量と
実行後のファイル量

4. 提案手法

本章では、ext4 におけるファイル格納位置制御手法を提案する。Ext4 のディスク内レイアウトの概要を図 12 に示す[20]。2.1 節で紹介した ext3 におけるファイル格納位置制御手法は block bitmap を書き換えることによってファイル格納位置を制御する。しかし、ext4 にて block bitmap の書き換えのみを行ってもファイル格納位置制御をすることはできない。その原因は 2 つある。1 つ目の理由は ext4 で新たに実装された bg_flags と bg_checksum によって Block bit map のみを書き換えるとエラーが出力されることである。2 つ目の理由は、ext4 では block bitmap の内容と Group descriptor 内の Bg_free_blocks_count_lo の値が不一致であるとエラーが出力されることである。

Block bit map は data block の「未使用/使用」の情報を「0/1」で管理している。Group descriptor 内の Bg_free_blocks_count_lo はブロックグループ内の空きデータブロック数を管理している。Bg_flags は block bitmap, inode bitmap, inode table が初期化されているか否かを管理している。Bg_checksum は block bitmap の checksum である。

任意のブロックグループの block bitmap を全て 1 に、bg_free_blocks_count を 0 に、bg_flags の block bitmap のフ

ラグを0に書き換えることによって、そのブロックグループにファイルが格納されることを禁止する。また、Ext4のソースコードを改変し checksum の検証がされないようにした。

5. 性能評価

5.1 dd コマンドによるシーケンシャル I/O 性能評価

Ext4 に提案手法を実装し、3.2 節と同様の測定を行った。提案手法では、ディスク外周部 110GB 以外へのファイル格納を禁止した。通常の ext4 と提案手法の ext4 における 6 回の各 read/write のスループットを図 13 に、平均スループットを図 14 に示す。また、提案手法において I/O 要求が発行された I/O アドレスを図 15 に示す。

図 13 から、提案手法によって、複数回ファイル読み書きと削除を行ってもスループットが一定になり、最大で read 処理のスループットを 53.6%，write 処理のスループットを 50.4% 向上させたことがわかる。また、図 14 から提案手法によって read 処理の平均スループットを 17.5%，write 処理の平均スループットを 19.3% 向上させたことがわかる。

また、図 15 から提案手法によってファイルが常に外周部に格納されていることが確認できる。これが ext4 のスループットを向上させることができた要因であると考えられる。

5.2 TestDFSIO

TestDFSIO は Hadoop のサンプルアプリケーションの一つで、HDFS の I/O 速度を測定するためのベンチマークツールである。ファイルサイズ数 1、ファイルサイズ 100GB とし、「write→read→ファイル削除」という動作を 5 回繰り返した。測定環境は表 1.2 と同様である。ローカルファイルシステム ext4 の上に分散ファイルシステム HDFS を構築し、提案手法では ext4 にてディスク外周部 110GB 以外へのファイル格納を禁止した。

通常手法と提案手法における TestDFSIO の各 read 処理、write 処理のスループットを図 16 に、平均スループットを図 17 に示す。通常手法と提案手法において I/O 要求が発行されたディスクアドレスをそれぞれ図 18 と図 19 に示す。

図 16 から、提案手法によって最大で read 処理のスループットを 39.4%，write 処理のスループットを 42.6% 向上させたことがわかる。また、図 17 から提案手法によって read 処理の平均スループットを 18.2%，write 処理の平均スループットを 18.6% 向上させたことがわかる。

図 18 と図 19 から、通常手法時にはディスク全体を活用しているが、提案手法時にはディスク外周部領域のみを活用していることがわかる。

5.3 TeraSort

TeraSort の input file は 24GB とし、4 回連続で実行した。提案手法では、ディスク外周部 200GB 以外へのファイル格

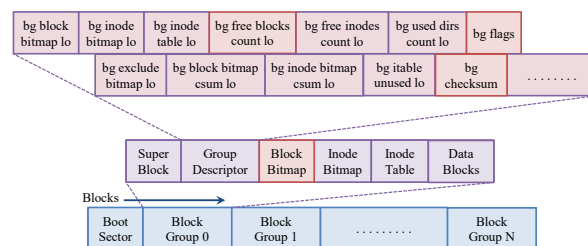


図 12 ext4 のディスク内レイアウトの概要

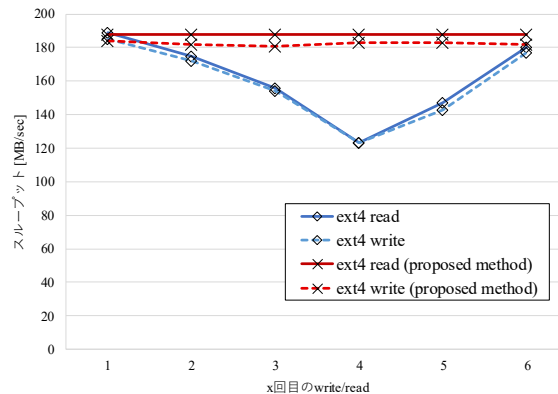


図 13 通常手法時と提案手法時の ext4 における read/write 速度 (dd command)

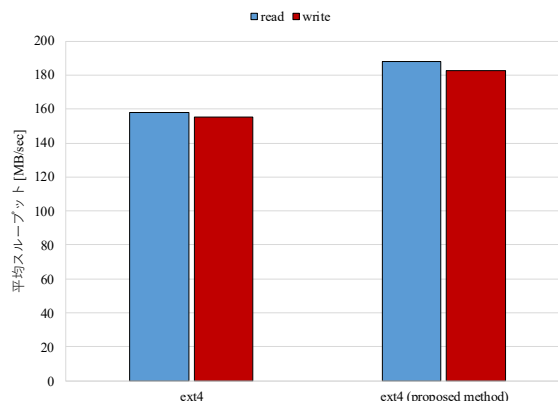


図 14 通常時と提案手法時の ext4 における平均シーケンシャル read/write 速度

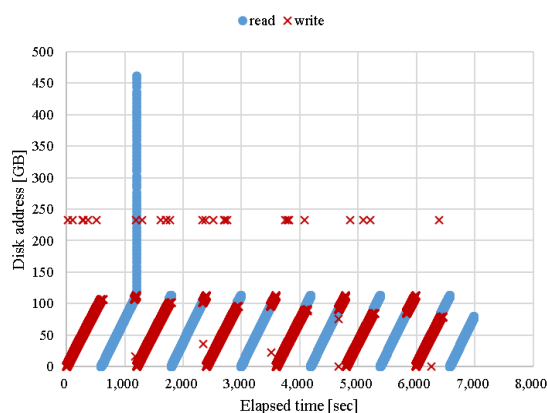


図 15 提案手法適用時の ext4 における I/O アドレス (dd command)

納を禁止した。

通常手法と提案手法における TeraSort の各実行時間を図 20 に、合計実行時間を図 21 に示す。通常手法と提案手法において I/O 要求が発行されたディスクアドレスをそれぞれ図 22 と図 23 に示す。

図 20 から、提案手法によって最大で実行時間を 30.1%短縮させたことがわかる。また、図 21 から提案手法によって合計実行時間を 15.4%短縮させたことがわかる。

図 22 と図 23 から、通常手法時にはディスク全体を活用しているが、提案手法時にはディスク外周部領域のみを活用していることがわかる。

6. 考察

図 15, 19, 23 を見ると、使用禁止にしているブロックグループの Block bitmap を一度読み込んでデータブロックの空き情報を確認していることがわかる。例えば、図 15 の 1100 秒付近におけるアドレス 110GB から 465GB への read アクセスがこれにあたる。今回の実験では、ディスクアドレス 110GB~465GB の全てのブロックグループの Block bitmap を読み込んだ場合でも、処理時間は約 1.5 秒と小さいため性能に大きな影響を与えていない。ただし、ext4 のブロック確保ポリシー自体を修正することにより、この時間も削減できると考えられる。

Ext4 にディスク外周部を積極的に活用させる方法として、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のみにファイルを格納させるという方法も考えられる。しかし、この方法でもパーティション内における外周部の仕様をさせることはできない。また、提案手法はファイルシステム情報を書き換えるだけで容易にファイルを格納させる位置を決めることができるため、より多くの状況に適用できると考えられる。

文献[21]にて、Hadoop において Shuffle heavy であるジョブの判別手法が提案されている。Shuffle 処理が多いジョブは中間ファイルが多く、I/O バウンドであるという特徴を持つ。判別手法によって Shuffle heavy であると判明したジョブを連続実行する場合、我々の手法を適用することによって大きく性能を向上させることができると期待できる。

7. おわりに

本稿では、ext4 がファイル削除によって生まれた空き領域を積極的に活用しないことを示した。そして ext4 におけるファイル格納位置制御手法を提案し、提案手法によって ext4 にディスク外周部の空き領域を積極的に活用させることができることを示した。また性能評価によって、ext4 のローカルファイルシステム上での性能を最大で 53.6%、HDFS 上での性能を最大で 42.6%向上させていることを示

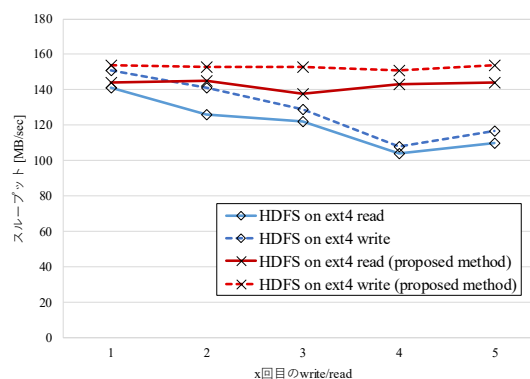


図 16 TestDFSIO の各スループット

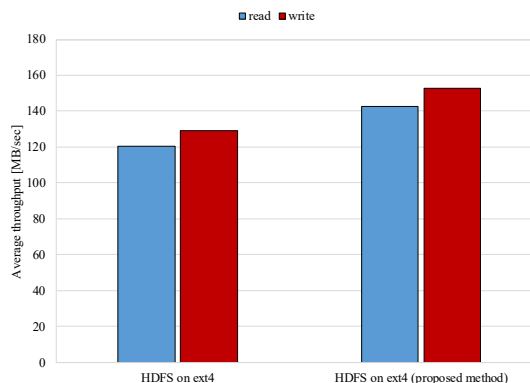


図 17 TestDFSIO の平均スループット

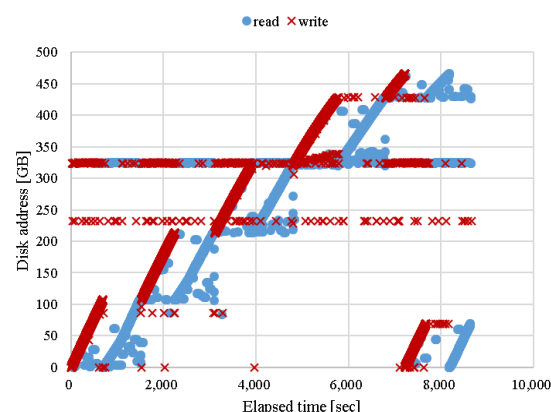


図 18 通常手法時の I/O アドレス (TestDFSIO)

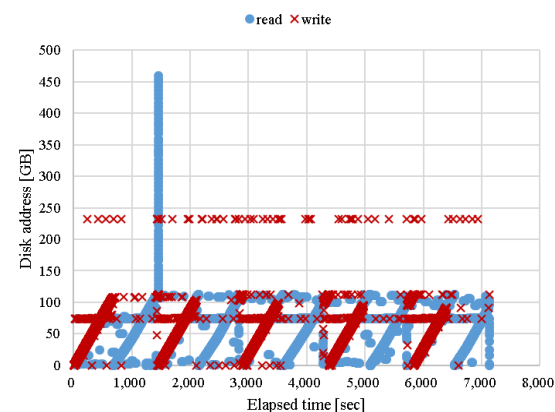


図 19 提案手法時の I/O アドレス (TestDFSIO)

した。

今後は、完全分散モードにて同様の手法の評価をおこなう予定である。

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] 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
- [2] E. Fujishima and S. 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.
- [3] 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
- [4] R.Card and T.Ts'o and S.Tweedle, "Design and Implementation of the Second Extended Filesystem," First Dutch International Symposium on Linux, 1994
- [5] 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
- [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] 中上誠, Jose A.B. Fortes, 山口実靖, "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," DEIM Forum 2020 H2-1
- [10] Makoto Nakagami, Jose A.B. Fortes, Saneyasu Yamaguchi, "Job-aware File-storage Optimization for Improved Hadoop I/O Performance", IEICE TRANSACTIONS on Information and Systems, Volume and Number: Vol.E103-D, No.10. 2020.
- [11] 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>
- [12] GitHub - SWIMProjectUCB/SWIM: Statistical Workload Injector for MapReduce - Project at UC Berkeley AMP Lab, <https://github.com/SWIMProjectUCB/SWIM>
- [13] Faraz Ahmad, Seyong Lee, Mithuna Thottethodi and T. N. Vijaykumar, "MapReduce with Communication Overlap

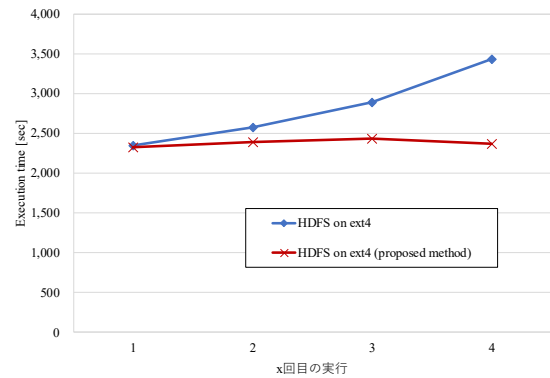


図 20 TeraSort の各実行時間

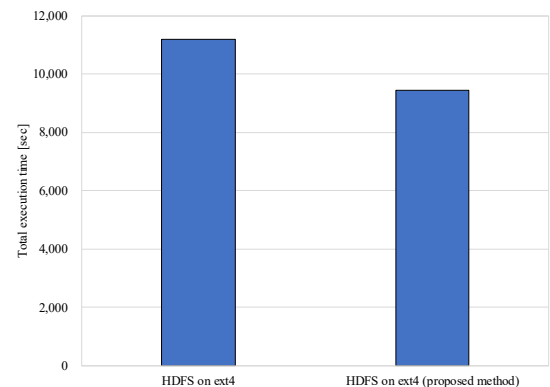


図 21 TeraSort の合計実行時間

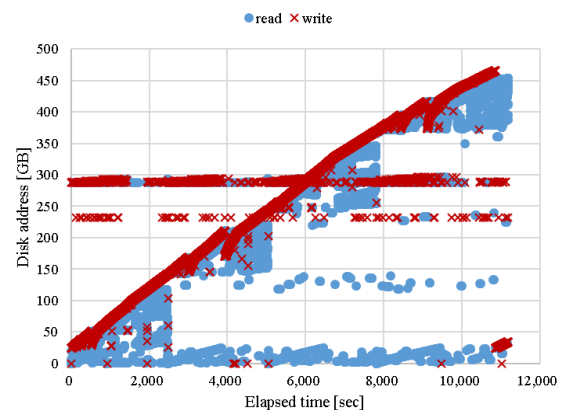


図 22 通常手法時の I/O アドレス (TeraSort)

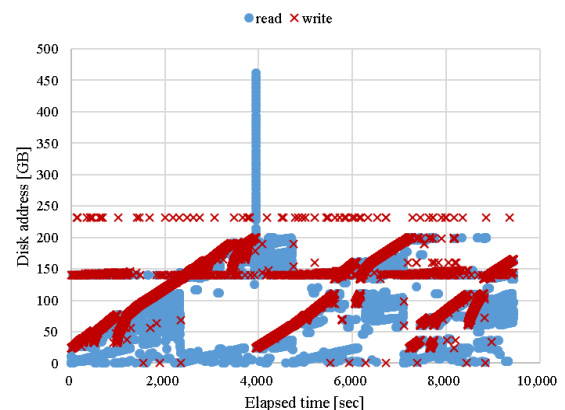


図 23 提案手法時の I/O アドレス (TeraSort)

- (MaRCO)", Journal of Parallel and Distributed Computing, May 2013, Pages 608-620 ,DOI: <https://doi.org/10.1016/j.jpdc.2012.12.012>
- [14] Navya Francis, Sheena Kurian K," Data Processing for Big Data Applications using Hadoop Framework ", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 3, March 2015, DOI: 10.17148/IJARCCCE.2015.4343
- [15] 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
- [16] 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)
- [17] 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
- [18]. 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
- [19] 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
- [20] 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
- [21] Wataru Muro, Masayuki Hirono, Jun Matsumoto, Takehiro Sato, Satoru Okamoto and Naoaki Yamanaka, "Proposal of Shuffle-Heavy job discrimination method in Hadoop ~ Application to hybrid optical/electrical switching datacenter network architecture(HOLST) ~", IEICE Technical Report, vol. 116, no. 498, PN2016-90, pp. 37-44, March 2017