ビッグデータ統合利用基盤における大規模 I/O 性能の向上

藤島 永太[†1]　中島 健司[†1]　近 丈一郎[†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, †3 工学院大学 工学部 情報通信工学科 〒163-8677 東京都新宿区西新宿 1-24-2

E-mail: †1 {cm15023, cm16032}@ns.kogakuin.ac.jp, †2 c513042@ns.kogakuin.ac.jp, †3 sane@cc.kogakuin.ac.jp

あらまし 近年，ビッグデータの利活用に注目が集まっている．医療やライフサイエンスなどのデータを利活用するためにはその暗号化や匿名化が重要となるが，暗号化によりデータサイズが拡大し，非常に規模の大きな I/O 処理が要求されることがある．本稿では，ビッグデータ統合利用基盤における I/O 速度に着目し，その向上手法について考察する．具体的には，ビッグデータ統合利用基盤における I/O 処理は大規模のシーケンシャル I/O が多いことを考慮し，シーケンシャル I/O アクセスに特化したファイルシステムを紹介する．そして，当該ファイルシステムが極めて大きなブロックサイズでストレージデバイスにアクセスすること，メタ情報の配置などによりデータ領域が分断されシーケンシャル性が低下するのを回避していることを示す．次に，性能評価により，他のファイルシステム使用時より高いビッグデータ統合利用基盤アプリケーション性能が得られることを示す．

キーワード ストレージ，ビッグデータ

1. はじめに

近年，IT 技術の進歩に伴い，地球上で生成されるデータ量が指数関数的に増加しており，それらのデータから有用な情報を収集・蓄積・分析し，活用することに注目が集まっている．ただし，医療・ライフサイエンスに代表されるセンシティブなデータを扱うセキュアなコンテンツ共有・流通基盤では，暗号技術の併用が欠かせない．この場合，大規模なデータを全て暗号化すると，データ量が大幅に増加し，実用的な処理時間を実現するためには大規模データ処理の I/O 速度の向上が重要となる．

本稿では, 大規模なデータを格納する HDD の特性に着目し，巨大な I/O に対して有効であると考えられる大きなブロックサイズでファイルの管理を行うファイルシステムを構築し，性能評価によりその有効性を示す．

本稿は次の様に構成される．2 章では，I/O 性能を向上させる対象となるゲノム配列の秘匿検索アプリケーションの概要について述べる．3 章では，当該アプリケーション実行中に発行される大規模な I/O の解析について述べる．4 章では，既存手法である大きなブロックサイズでファイル管理を行うファイルシステムについて説明する．5 章では，性能評価により当手法の有効性を示す．6 章では，関連研究について述べ，最後に，7 章で本稿のまとめと今後の課題について述べる．

2. ゲノム配列の秘匿検索アプリケーション

バイオインフォマティクス分野の研究では，ゲノム配列の検索が頻繁に行われる．しかし，個人ゲノムの利用はプライバシー保護への対策が必須であり，暗号化したままでゲノム配列の検索処理を行う必要がある．これを実現するための従来研究では，PBWT 離散データ構造と準同型暗号を組み合わせた手法が提案されている．本稿では，PBWT 離散データ構造と完全準同型暗号を用いたゲノム秘匿検索手法 [1] に焦点を当て，そのアプリケーションの性能について考察を行う．

当該手法を実装したアプリケーションは，コホート研究で得られるデータの利用など，クエリとデータベースの両方にセンシティブな情報が含まれている場合を想定している．このモデルでは，クライアント(検索者)はサーバ(データ保有者)に対してクエリ内容を秘匿し，サーバはクライアントに対してクエリの検索結果以外の情報を秘匿する．当アプリケーションの現在の実装では，サーバは長さ N のゲノム配列が M 本登録されているデータベースを持つ．性能評価では，1000 ゲノムプロジェクト [2] における SNP 配列 2,184 サンプルを持つデータベースを使用している．

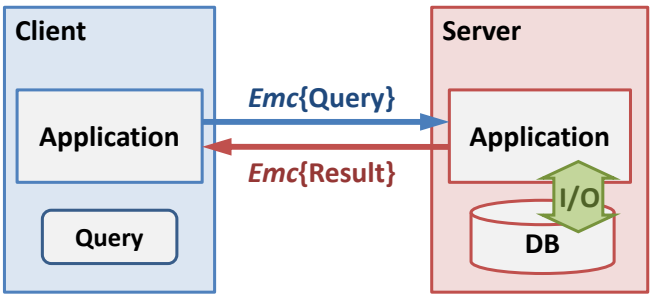

図 1. ゲノム配列検索の概略図

3. ディスク I/O 解析

前章のゲノム配列秘匿検索アプリケーション実行中のディスク I/O を解析するために，Linux カーネルを改変し，SCSI 層における I/O 要求の内容を調査した．測定環境は，物理計算機 1 台でサーバとクライアントを並列に動作させ，データベースの容量は 2 GB とした. 測定に使用した物理計算機の仕様は表 1 の通りであり，データベースを格納する HDD の仕様は表 2 の通りである．

表 1. 測定用物理計算機の仕様

CPU	Intel Core i5-4670 3.40 GHz
OS	CentOS 7 x86_64 minimal
Kernel	Linux-3.10.103 (modified)
SSD	512 GB (ext4)
HDD	1 TB (ext3)
Main Memory	16 GB

表 2. 測定用 HDD の仕様

型番	WD1003FZEX
インタフェース	SATA 3.0 (6.0 Gbps)
容量	1 TB
バッファ容量	64MB
回転数	7,200 rpm

測定結果を図 2 に示す．横軸はアプリケーション実行中の経過時間 [sec]，縦軸はアクセスされたディスク内のアドレス [GB]を示している．経過時間 10～22 秒の時間帯で，サーバにおけるデータベースの読込処理が行われている．この部分のグラフを拡大したものを図 3 に示す．図より，大きな範囲に渡り巨大なシーケンシャルアクセスが行われていることが分かる．この結果から，当該アプリケーションの I/O 性能を向上させるためには，巨大なファイルに対する連続的な読み込み性能を向上させる必要があることが分かる．

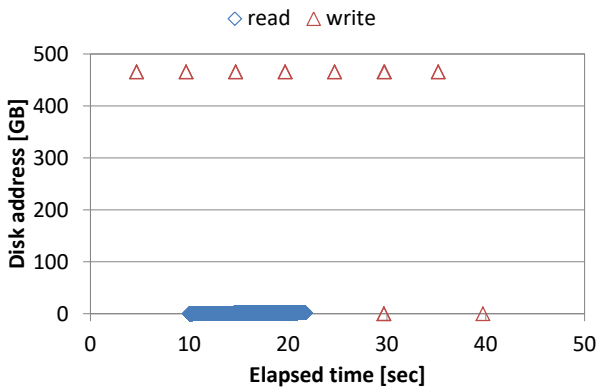

図 2. SCSI 層における I/O 要求

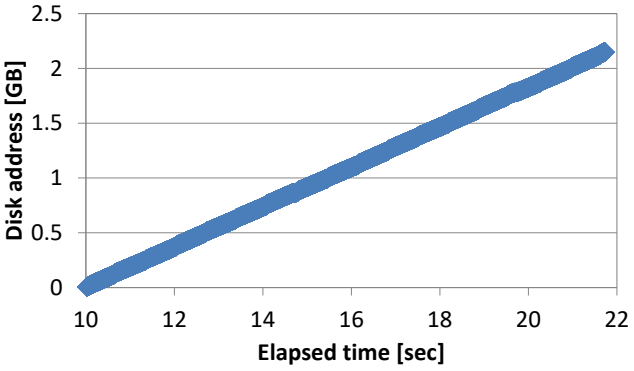

図 3. SCSI 層における I/O 要求 (拡大図)

4. 大規模シーケンシャルアクセス性能の高いファイルシステムの構築

前章のディスク I/O 解析の結果より，アプリケーションの I/O 性能を向上させるためには巨大なファイルに対するシーケンシャル I/O 性能を向上させる必要があることが分かった．そこで，巨大なブロックサイズでファイルの管理を行うファイルシステムを構築し，そのファイルシステム上にデータベースを格納することにより，I/O 性能を向上させることが可能であると期待できる [3] ．本章では，本手法とそれに基づくファイルシステムの実装について述べる．

4.1. 手法の概要

本手法は，我々の過去の研究 [4] で得られた知見に基づいて動作する．具体的には，定記録密度方式の HDD に対してシーケンシャルアクセスを行う場合，ディスク外周側の連続領域に対して行う方が，内周側に対して行うよりも高い I/O 性能が得られるという知見に基づく．

また，連続性の高い I/O と連続性の低い I/O では，連続性の高い I/O の方が HDD の持つアクセス性能を活かせるため，巨大なブロックサイズでファイル管理を行う方式とした．

4.2. 手法の実装

本手法の実装は，FUSE (Filesystem in Userspace)を用いて行った．図 4 に本手法の概略図を示す．ユーザ空間上での実装であり，ディスクへの I/O 処理はデバイススペシャルファイルに対して直接行い，各ファイルのディスク内のオフセット位置などのファイル管理情報は配列を用いてメモリ上で管理している．ファイルシステムのブロックサイズは 256 MB に設定しており，ディスク領域はこのブロック単位で分割される．そして，ディスク領域を全てファイルのデータ領域として使用することにより，連続性の高いディスクアクセスを実現している．シーケンシャルアクセス性能の高いディスク外周部を積極的に使用するために，新規ファイルの格納位置検索方法は，ディスクアドレスの低い方から順に空き領域の検索を行う仕様とした．また，シーケンシャルアクセスに対して効果が低いとされるページキ

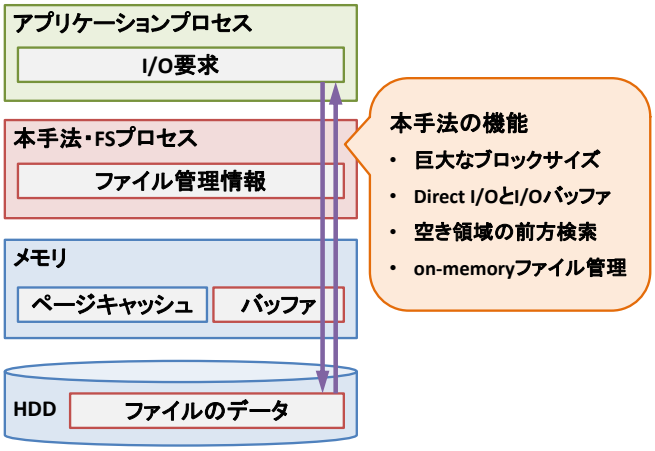

図 4. 本手法の概略図

ャッシュを無効化するために，ディスクアクセスには Direct I/O を使用している．このディスクアクセスをブロックサイズ単位で行うために，ブロックサイズ単位のバッファを設けている．

5. 性能評価

本章では，前章で述べた我々の提案手法の性能評価について述べる．

5.1. 測定方法および環境

ゲノム配列検索アプリケーションのデータベースを格納するファイルシステムを Ext3 または提案手法としたときのデータベース読込時間を比較することにより性能評価を行った．アプリケーションはそれぞれ 10 回ずつ実行し，平均読込時間を比較対象とした．測定環境は，データベースの格納に用いるファイルシステムを変更した点以外は 3 章と同様である．

5.2. 測定結果

データベースの平均読込時間の測定結果を図 5 に示す．ファイルシステムに Ext3 を使用した場合の平均読込時間は 11.72 秒，XFS を使用した場合は 11.63 秒となっている．そして，提案手法を使用した場合の平均読込時間は 9.80 秒となり，Ext3 と比較して 16.4%，XFS と比較して 15.8%の性能向上を確認した．

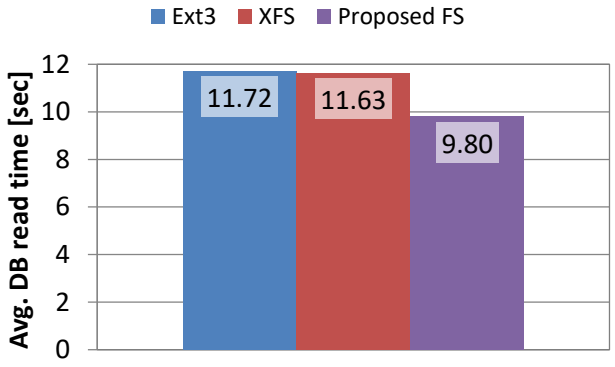

図 5. データベースの平均読込時間

6. 関連研究

本章にて，本研究と関連する研究を紹介する．

シーケンシャルアクセス性能が深く考慮されているファイルシステムとして XFS [7] , ZFS [9] , Ext4 [11] などがある. XFS [7] は著名なジャーナリングファイルシステムの一つであり，巨大なファイルへのアクセスの性能を考慮した初期のファイルシステム実装である．発表当時 [10] より巨大なデータや巨大なサイズのファイルをサポートしており，大きなブロックをまとめて扱えるエクステントの機能や，可変のブロックサイズを大きくして大きなファイルへのアクセス性能を高める機能などを有している．ZFS は極めて大きなサイズのデータの扱いを可能とするとともに，実装の簡略化も考慮されている．Ext4 ファイルシステム [11] [12] は，これまで伝統的に使用されてきた Ext2/Ext3 ファイルシステム [13] を拡張したものであり，Ext3 同様にジャーナリングファイルシステムである．巨大なサイズのストレージをサポートしており，エクステント機能により巨大な連続領域の確保による巨大アクセスの高速化などが図られている．また，ファイルシステムの性能を評価した文献として [14] [15] などがある．しかし，シーケンシャルアクセスのみに特化して作成された著名なファイルシステムはなく，この点において本研究とは異なっている．

ディスク上のレイアウトに関する研究は，文献 [16] にてまとめられ，紹介されている．初期のディスクレイアウトの理論に関する研究として文献 [17] があり，シミュレーションによる研究として文献 [19]-[22] がある．文献 [17] では，最高頻度アクセスデータをストレージの中央に配置する organ pipe 手法がラ ンダムアクセスに適していることが示されている．

ファイルシステムレベルのレイアウトに関する研究としては，以下の研究がある．FFS [23] やその後続の研究 [24] [25] にて，関連するデータブロックと inode をディスク上の近隣に配置することにより I/O 速度を向上させる方法が提案されている．文献 [26] にて，参照の局所性ではなく，微小ファイルの距離を近づけることに着目して性能を上げる方法が提案されている．また，アクセス頻度の高いファイルをディスクの外周に配置することによりログ構造化ファ イルシステムをさらに高速化する研究がなされている [27] ．HFS [28] の Hot File Clustering や Smart File System [29] では，ファイルシステムが実行時アクセスパターンを観察し，高頻度アクセスデータを予約領域に移動を行っている． FS2 [18] では，ファイルの複製を用いて連続アクセスされるファイル，あるいはその複製を近隣に配置することによりさらなる高速化を実現している．これらの研究は，シーク距離を削減し性能を向上させるという点において本研究と類似しているが，シーケンシャルアクセス速度の速いディスク外周部の使用や，シーケンシャルアクセスに特化したファイルシステムの実装は考慮されておらず，本研究とはシーケンシャルアクセス性能側面において大きく異なっている．

シーケンシャルアクセス性能を考慮した我々の既存の研究としては，Ext2/Ext3/Ext4 ファイルシステムを外部から静的に制御し高速領域(外周部)のみを使用させることによるシーケンシャルアクセス性能の向上研究 [5] や，動的に制御し性能向上させる研究 [4] や，これを用いて DWH アプリケーションの性能を向上させる研究 [6] がある．これらは，ストレージ上におけるファイル格納位置の制御はファイルシステムが司ることに着目し，その制御をファイルシステム外から擬似的に実現した手法であるが，簡易な実装となっておりファイル格納位置を完全に制御することができず，制御の精度において本稿より劣っている．

論文 [30] の仁科らは，HDD と比較してランダムアクセス性能が高い SSD に着目し，SSD を HDD のディスクキャッシュとして用いるための Linux デバイスドライバの実装を行った．著者らは，SSD ディスクキャッシュの既存手法には利用制約があることを提示し，それらの制約を低減させた手法を提案し，性

能評価によりその有効性を示した．仁科らの研究は，HDD のランダム I/O 性能の低さを補うための SSD ディスクキャッシュ手法であり，本研究の HDD のシーケンシャル I/O 性能を活かすための手法と補完し合える関係にあると考えられる．

7. おわりに

本稿では，セキュアなコンテンツ共有・流通基盤を発展させるために作成されたゲノム配列の秘匿検索アプリケーションの I/O 性能向上について考察し，巨大なブロックサイズでファイル管理を行うファイルシステムを提案し，性能評価によりその有効性を確認した．

今後の課題として，キャッシュに影響を及ぼす巨大な I/O に対して Direct I/O を適用することや，ユーザ空間で実装を行った当手法を，よりオーバヘッドが小さいと考えられるカーネル空間で実装することなどが考えられる．

謝　　辞

本研究は，JST，CREST の支援を受けたものである．

本研究は，JSPS 科研費 25280022，26730040，15H02696 の助成を受けたものである．

参 考 文 献

[1] 石巻 優, 清水佳奈, 縫田光司, 山名早人, “完全準同型暗号を用いた高速なゲノム秘匿検索”, 2016 Symposium on Cryptography and Information Security (SCIS2016), (2016.1).

[2] The 1000 Genome Project Consortium: "An integrated map of genetic variation from 1,092 human genomes." Nature, Vol.491, pp.56-65, (2012).

[3] 藤島永太, 中島健司, 山口実靖, “高速大規模 I/O が可能なファイルシステムによるビッグデータ統合利用基盤の性能向上”, 情報処理学会 第 79 回全国大会.

[4] Eita Fujishima, and Saneyasu Yamaguchi, "Dynamic File Placing Control for Improving the I/O Performance in the Reduce Phase of Hadoop." 10th International Conference on Ubiquitous Information Management and Communication (ACM IMCOM’16), No.8-2, (2016. 1) DOI=http://dx.doi.org/10.1145/2857546.2857595.

[5] Eita Fujishima, and Saneyasu Yamaguchi, “Improving the I/O Performance in the Reduce Phase of Hadoop,” The Third International Symposium on Computing and Networking (CANDAR2015), pp.82-88, Sapporo, Japan, (2015. 12). DOI: 10.1109/CANDAR.2015.24.

[6] Eita Fujishima, Kenji Nakashima, and Saneyasu Yamaguchi, “Performance Improvement of I/O Intensive OLAP with Dynamic Control of File Storing Location,” 11th International Conference on Ubiquitous Information Management and Communication (ACM IMCOM’17), Beppu, Japan, (2017. 1). DOI=http://dx.doi.org/10.1145/3022227.3022305.

[7] Adam Sweeney, Doug Doucette, Wei Hu, Curtis Anderson, Mike Nishimoto, and Geoff Peck, “Scalability in the XFS file system.” In Proceedings of the 1996 annual conference on USENIX Annual Technical Conference (ATEC '96), USENIX Association, Berkeley, CA, USA, 1-1, (1996).

[8] Ray Bryant, Ruth Forester, and John Hawkes, “Filesystem Performance and Scalability in Linux 2.4.17.” In Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference, Chris G. Demetriou (Ed.), USENIX Association, Berkeley, CA, USA, 259-274, (2002).

[9] Jeff Bonwick, Matt Ahrens, Val Henson, Mark Maybee, and Mark Shellenbaum, “The zettabyte file system.” In Proc. of the 2nd Usenix Conference on File and Storage Technologies, (2003).

[10] Randolph Y. Wang and Thomas E. Anderson, "xFS: a wide area mass storage file system." Proceedings of IEEE 4th Workshop on Workstation Operating Systems, WWOS-III, Napa, CA, pp. 71-78, (1993). doi: 10.1109/WWOS.1993.348169

[11] Avantika Mathur, Mingming Cao, Suparna Bhattacharya, Andreas Dilger, Alex Tomas, and Laurent Vivier, “The new ext4 filesystem: current status and future plans,” In Proceedings of the Linux symposium, Vol. 2, pp. 21-33, (2007, June).

[12] Avantika Mathur, Mingming Cao, Suparna Bhattacharya, Andreas Dilger, Alex Tomas, and Laurent Vivier, “The New Ext4 filesystem: Current Status and Future Plans,” In Ottawa Linux Symposium (OLS ’07), Ottawa, Canada, (2007, July).

[13] Stephen C. Tweedie, “Journaling the Linux ext2fs File System,” In The Fourth Annual Linux Expo, Durham, North Carolina, (1998, May).

[14] Lanyue Lu, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, and Shan Lu, “A Study of Linux File System Evolution,” ACM Transactions on Storage (TOS), Article 3, Vol. 32, No. 3, pp.10-17, (2014, January). DOI: http://dx.doi.org/10.1145/2560012

[15] Ray Bryant, Ruth Forester, and John Hawkes, “Filesystem Performance and Scalability in Linux 2.4.17,” In Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference, Chris G. Demetriou (Ed.). USENIX Association, Berkeley, CA, USA, 259-274, (2002).

[16] 山田将也, 山口実靖, “仮想計算機環境における二重ファイルシステム構造を考慮した仮想 HDD イメージファイルの配置”, 第 4 回 Web とデータベースに関するフォーラム (WebDB Forum 2011), (2011. Nov).

[17] C.K.Wong, “Algorithmic Studies in Mass Storage Systems,” Computer Sciences Press, (1983).

[18] Hai Huang, Wanda Hung, and Kang G. Shin, “FS2: Dynamic Data Replication in Free Disk Space for Improving Disk Performance and Energy Consumption," ACM SIGOPS Operating Systems Review. Vol. 39. No. 5. ACM, (2005).

[19] Robert English and Stephnov Alexander, “Loge: A Self-Organizing Disk Controller,” Proceedings of the Winter 1992 USENIX Conference, (1992).

[20] David Musser, “Block Shuffling in Loge,” HP Technical Report CSP-91-18, (1991).

[21] C.Ruemmler and J. Wilkes, “Disk Shuffling,” HP Technical Report, HPL-CSP-91-30, (1991).

[22] P.Vongsathorn and S. D. Carson, “A System for Adaptive Disk Rearrangement,” Software Practice Experience, 20(3): pp.225-242, (1990).

[23] M.K.McKusick et al., “A Fast File System for UNIX,” ACM Transactions on Computing Systems (TOCS), 2(3), (1984).

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

[25] Stephen Tweedie, “Journaling the Linux ext2fs Filesystem,” LinuxExpo, (1998).

[26] Greg Ganger and Frans Kaashoek, “Embedded Inodes and Explicit Groups: Exploiting Disk Bandwidth for Small Files,” USENIX Annual Technical Conference, 1-17. 1997. File System,” ACM Transactions on Computer Systems, pp.26-52, (1992).

[27] Jun Wang and Yiming Hu, “PROFS.Performance-Oriented Data Reorganization for Logstructured File System on Multi-Zone

Disks," The 9th International Symposium on Modeling, Analysis and Simulation on Computer and Telecommunication Systems, pp.285-292, (2001).

[28] HFS Plus Volume Format, http://developer.apple.com/technotes/tn/tn1150.html

[29] C.Staelin and H.Garcia-Molina, "Smart Filesystems," USENIX Winter, 45-52, 1991. 2001.

[30] 仁科圭介, 佐藤未来子, 並木美太郎, "SSD をディスクキャッシュとして用いる Linux デバイスドライバの実装", 情報処理学会 第 53 回 プログラミング・シンポジウム, pp.29-36, (2012.1).