Vol.2011-DPS-146 No.9
Vol.2011-CSEC-52 No.9
2011/3/10

オンメモリ型ストレージ間 データマイグレーションの性能と負荷の評価

小山 芳樹[†] 山田 将也[††] 松村 政志[††] 山口 実靖[††]

ストレージ装置の I/O 速度の高速化手法の 1 個に，ストレージ間データマイグレーションによるストレージ間の負荷分散がある．しかし，データマイグレーションを行うためには高負荷ストレージからデータを読み込み，低負荷ストレージへのデータの書き込みを行う必要がある．よって，データマイグレーションは一時的に高負荷ストレージの負荷をさらに増加させてしまうという課題がある．

これに対して我々はファイルアクセス履歴を用いるオンメモリ型データマイグレーション手法を提案する．本手法では，オペレーティングシステムにファイルアクセス履歴取得機能を実装し，最近アクセスされたファイルをマイグレーション対象とする．よって，マイグレーション対象データはメインメモリにキャッシュされており，ストレージアクセスを伴わず読み込むことができる．これにより，高負荷ストレージへの負荷を軽減できると期待される．我々はオペレーティングシステムのファイルシステムにアクセスファイル履歴保持機能を追加し，提案データマイグレーション手法を実装した．性能評価実験により提案手法の性能と I/O 使用率を評価した結果，提案手法は従来のデータマイグレーション手法よりも低い負荷でデータマイグレーションが行えることが確認された．

Performance and I/O Usage Evaluation of Data Migration Method using Memory Cache

Yoshiki Koyama[†] Masaya Yamada[††]
Masashi Matsumura[††] and Saneyasu Yamaguchi[††]

Data migration between storage devices enables load balancing and improves storage performance. However, data migration from high-load storage to low-load storage requires data reading from the high-load storage and data writing to the low-load storage. Thus it humus the performance of high-load storage more. In this paper, we proposed a migration method, with which data in memory cache is migrated. This method requires data reading from not a high-load storage device but a cache memory. Hence, performance decline of high-load storage caused by this method is smaller than a traditional method. We evaluated the performance and I/O usage of the proposed method. The experimental results demonstrated that our method decrease performance decline.

1. はじめに

計算機システムの普及により，計算機が扱うデータの量は爆発的に増加した．これにより，計算機システムのストレージ装置には非常に高速なデータ I/O 処理能力が要求される様になった．しかし，HDD をはじめとするストレージデバイスは急速な容量の成長は達成しているが，性能の成長は比較的緩やかな速度でしか達成しておらず，必ずしも十分な性能を提供しているとは言えない．よって，性能向上はストレージシステムの重要な課題となっている．

ストレージ装置の I/O 速度の高速化手法の 1 個に，ストレージ間データマイグレーションによる負荷分散があり，成果を上げている [1)2)]．しかし，ストレージ間データマイグレーションによる負荷分散にはマイグレーション中に高負荷ストレージの負荷をさらに増加させてしまうという課題がある [2)]．この課題に対して，本研究ではマイグレーションシステムがファイルシステムと連携することによりメインメモリキャッシュを効果的に活用し，低負荷でのマイグレーションを実現する手法を提案する．具体的にはファイルシステムよりファイルアクセス履歴を取得し，メインメモリにキャッシュされているデータを検出し，そのデータをマイグレーション対象とする．これにより高負荷ストレージからの読み込みを発生させず，高負荷ストレージのさらなる負荷の増加を低減できると期待される．

2. ファイルシステム

ファイルシステムは記憶装置上のデータをファイルとして透過的に扱うために提供されているオペレーティングシステム内のソフトウェアである．Linux ファイルシステムの動作概念図を図 1 に示す．Linux ファイルシステムは基本機能が実装されている VFS（virtual filesystem）と ext2（second extended filesystem）などのローカルファイルシステムで構成されている．VFS は全てのローカルファイルシステムで共通に使われ，VFS によりアプリケーションが様々なローカルファイルシステムに対して同一の手法でアクセスできるようになる．本研究の提案手法ではファイルアクセス履歴保持機能を VFS に追加している．ファイルアクセス要求が発行された時にファイル名をカーネル空間に確保されたメモリ領域にコピーするものである．

ローカルファイルシステム ext2 のディスクとファイルの管理の仕方を説明する．ディスクの管理は図 2 のように複数のブロックをブロックグループにまとめて管理している．ブロックグループのデータブロックにはユーザが使用するファイルのデータ

[†] 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
[††]工学院大学 工学部 情報通信工学科
Department of information and Communications Engineering, Kogakuin University

Vol.2011-DPS-146 No.9
Vol.2011-CSEC-52 No.9
2011/3/10

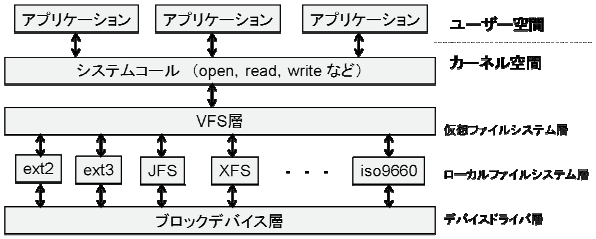

図 1　Linux ファイルシステムの動作概念図

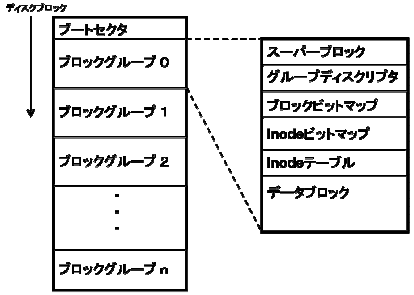

図 2　Ext2 パーティションの構造

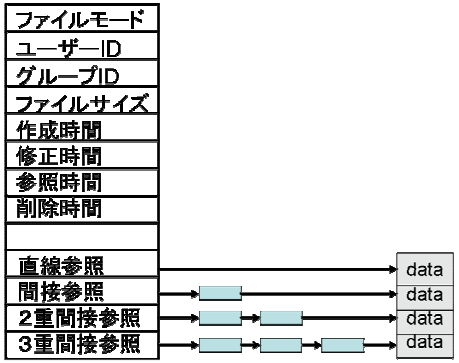

図 3　inode 構造体

が保持される．そして，inode テーブルには各ファイルの inode が並んでいる．inode 構造体の内容は図 3 のようになっている. inode 構造体にはファイルのパーミッション，オーナー，ファイルサイズや作成時間などの属性情報やデータブロックの参照値が記録されている．本研究ではマイグレーションを行う際にはデータを別の HDD に移動した後に inode 構造体のデータブロック参照の値を書き換えることで，移動先のデータにアクセスが行われるように変更している．

3. 関連研究

3.1. ストレージ間データマイグレーション

複数の物理ストレージデバイスを用いる計算機があり，図 4-a のようにある一つのストレージにアクセスが集中している場合，その高負荷なストレージが原因でシステム全体の性能が低下してしまう．そこで図 4-b のように高負荷ストレージ内のデータを低負荷ストレージに移動し負荷を分散し，性能を向上させる手法がデータマイグレーションである．アプリケーション層におけるデータマイグレーションとして文献 [1] がある．この研究では，データベースアプリケーションにおいて動的かつ自律的にデータを移動させる手法を提案し，性能の向上を実現している．ストレージ層におけるデータマイグレーションとして，横田ら自律ディスク [3]と，それにおけるデータマイグレーションがある [2]．横田らは自律ディスクというディペンダブルで高性能な先進的なストレージシステムを提案しており，小林らは自律ディスクにおけるデータマイグレーションの研究を行っており，詳細な考察や性能向上を行っている [2].

アプリケーション層などの上位層においてマイグレーションを行うことの利点はアプリケーションの動作の詳細を把握した上でのマイグレーションが可能であり，より適切なマイグレーションが期待できることである．その反面，提案手法の適用範囲が制限されてしまう欠点がある．これに対してストレージ層などの下位層でデータマイグレーションを行う場合，任意の OS，任意のアプリケーションに対して提案手法を適用することが可能となる．また，小林らによりデータマイグレーションの課題も指摘されている [2]．すなわち，データマイグレーションを行うためには高負荷ストレージよりデータを読み込み，低負荷ストレージに書き込む必要があるため，一時的に高負荷ストレージにさらに負荷を与えてしまう．これに対して，自律ディスク(ストレージ層)におけるマイグレーション負荷の影響に考察や，解決手法が提案されている [2]　.

3.2. ファイルシステム層におけるデータレプリケーション

ファイルシステム層における関連する研究として，Huang らによる FS2 の研究がある [4]．これはファイルシステム層おけるデータレプリケーションの研究であり，ディスクの空き領域にファイルの複製を配置し I/O 性能を向上させる．具体的には，連続アクセスされるファイルを把握し，それらが近隣に存在できるように各ファイルのレプリケーションを空き領域に配置する．

アプリケーション層，ストレージ層におけるデータマイグレーションの研究は，OS のメモリキャッシュの有効利用を目指すものではなく本研究とは趣旨や実装レイヤが異なっている [5]．また，レプリケーションの研究はファイルシステム層において実装している点が本研究と類似しているが，空き領域にレプリケーションを配置することを主たる目的としており，本研究とは趣旨が異なる．

Vol.2011-DPS-146 No.9
Vol.2011-CSEC-52 No.9
2011/3/10

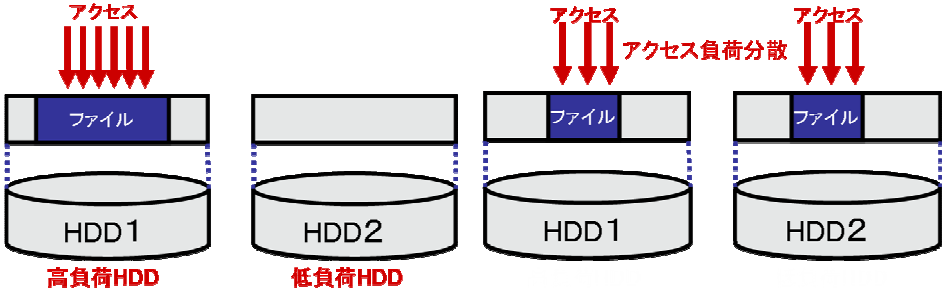

図 4-a ある HDD にアクセスが集中　　図 4-b データマイグレーション後

図 4　データマイグレーションによる負荷分散

4.　基本性能測定（マイグレーションの効果）

マイグレーションによる負荷分散の効果を確認するために，マイグレーション前とマイグレーション後の性能を測定し，比較した．測定を行った実験環境を図 5，実験機の詳細を表 1 に示す．2 台の HDD 上に単一の論理ボリュームを構築し，この論理ボリューム内でデータを別の HDD に移動することによりマイグレーションを実現した．論理ボリューム内における移動であるためファイル名を保ったままの移動が可能となる．論理ボリュームは LVM(Logical Volume Manager)を用いて構築した．

測定では図 6-a のように論理ボリュームの前半(HDD1 の領域)に 4MB のファイルを 1024 個配置し，それらに対してランダムに読み込みをかけるベンチマークを実行し得られた性能を測った．次に図 6-b のように論理ボリュームの前半にファイル 512 個，後半(HDD2 の領域)にファイル 512 個配置した場合の性能を測定し比較した．前者がマイグレーション前，後者がマイグレーションによる負荷分散後と見なすことができる．測定結果を図 7 に示す．結果よりマイグレーションにより負荷を分散させることで読み込み速度が向上することが確認された．

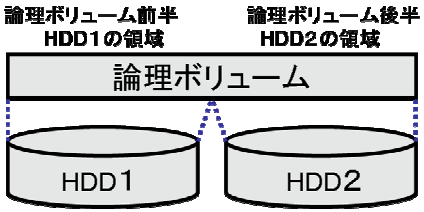

図 5　実験環境（論理ボリューム）

表 1　実験機の詳細

CPU	AMD Athlon 1640B 2.7GHz
メインメモリ	DDR2　1GB
HDD	「HGST Deskstar7k 1000.B」×2 容量 1TB，回転数 7200rpm
論理ボリューム	LVM を使用し 2 台の HDD を単一の論理ボリュームとする
OS	Fedora 8 Linux (カーネル Linux-2.6.23.1)
ファイルシステム	Ext2

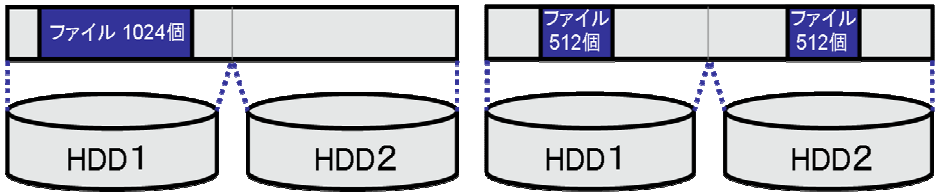

図 6-a マイグレーション前を想定　　図 6-b マイグレーション後を想定

図 6 ファイル作成の状態

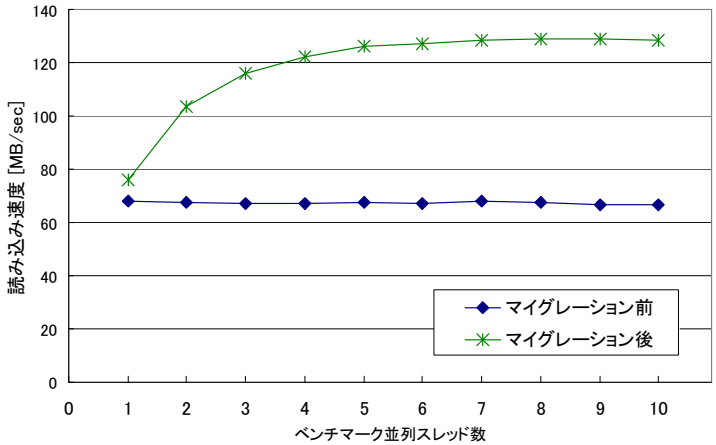

図 7 マイグレーション前後の性能

Vol.2011-DPS-146 No.9
Vol.2011-CSEC-52 No.9
2011/3/10

5. ファイルアクセス履歴を用いたオンメモリ型ストレージ間データマイグレーション手法の提案

本章において，ファイルシステム層においてマイグレーションを行い，マイグレーションによる高負荷ストレージに与える負荷を軽減する手法を提案する．本手法では，ファイルシステムにおいて最近アクセスされたデータを管理し，これをマイグレーションの対象とする．最近アクセスされたファイルはメインメモリにキャッシュされており，これをマイグレーションする場合は(高負荷ストレージからではなく)メインメモリからデータを読み込み，低負荷ストレージに書き込みを行うことになる．よって，高負荷ストレージに加える負荷を軽減することが可能となる．

システム構成と動作を図 8，図 9 に示す．図 8-a のようにアプリケーションから HDD データへのアクセスが発生すると，そのデータはメモリ内にキャッシュされる．以降の同一データへのアクセスは図 8-b のようにメモリキャッシュから読み込むことにより行われる．本手法では図 9-a のようにマイグレーションツールがファイルシステムよりアクセス履歴を取得し，メインメモリにキャッシュされているファイルの情報を取得する．そして，そのファイルをマイグレーション対象とする．結果として図 9-b のようにマイグレーションツールは対象ファイルをメモリキャッシュから読み込み，高負荷ストレージに与える負荷を小さく抑えることが可能であると予想される．次に図 9-c のように読み込んだデータを低負荷 HDD に書き込み，最後に図 9-d のように inode ポインタを張り替える．マイグレーション後のファイルアクセスは図 10 のように移動先のデータに対してアクセスされる．

本稿では性能測定用の簡易実装を作成し，それを用いて性能測定を行った．簡易実装ではマイグレーションツールはユーザ空間で動作し，デバイススペシャルファイルを用いてストレージイメージに直接書き込みを行う．具体的には，ファイルデータのストレージへの書き込み，inode テーブルの更新をスペシャルファイルに対して行う．よって，本実装はローカルファイルシステム実装(ext2)に依存する形になっており，他のファイルシステムでは動作しない．また，マイグレーション中の書き込みには対応していない．ファイルアクセス履歴保持機能は，仮想ファイルシステム内に実装されている．よって，任意のローカルファイルシステムで用いることができる．本履歴保持機能では，ファイルアクセス要求が発行された時にファイル名をカーネル空間に確保されたメモリ領域にコピーする．そして，本データをファイルシステムを介してユーザ空間から読み込むことができる．

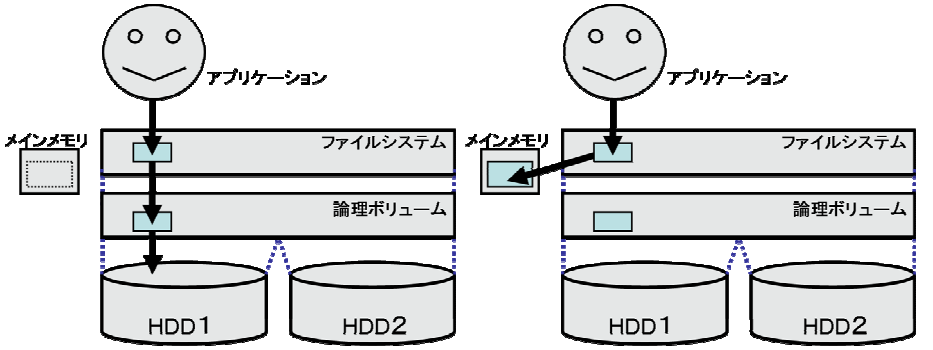

図 8-a　HDD から読み込む場合　　図 8-b　キャッシュにデータがある場合

図 8 通常のファイルアクセス

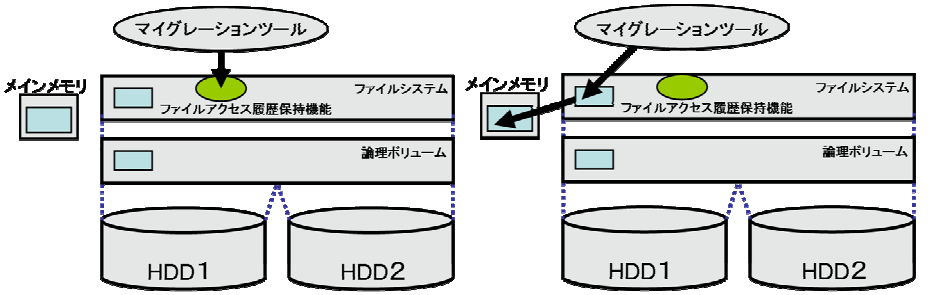

図 9-a　最近アクセスされたファイル名を取得　　図 9-b　キャッシュから読み込む

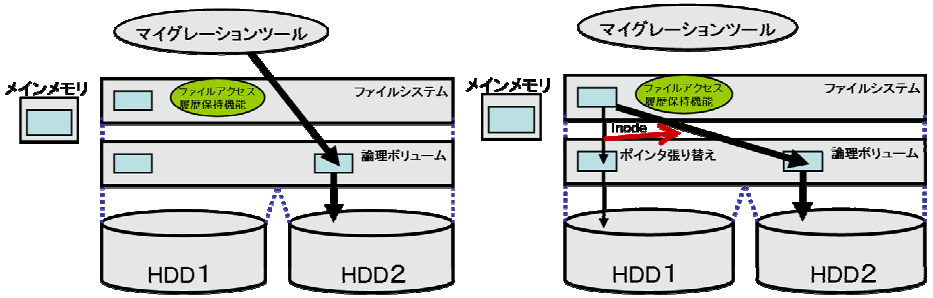

図 9-c　読み込んだデータを書き込む　　図 9-d　inode を書き換え

図 9　提案マイグレーションツールの手順

Vol.2011-DPS-146 No.9
Vol.2011-CSEC-52 No.9
2011/3/10

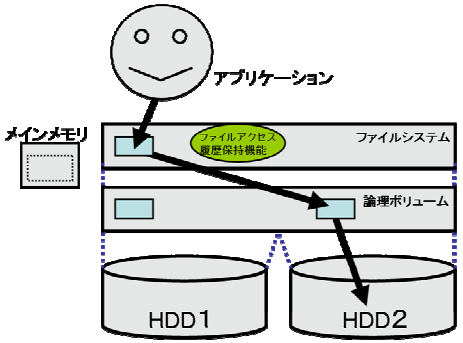

図 10 マイグレーション後のファイルアクセス

6. 性能評価

6.1 測定環境

提案手法の性能を評価した．測定環境は第 4 章と同一である．2 台の HDD で単一の論理ボリュームを構築し，この論理ボリューム内でデータを別の HDD に移動することによりマイグレーションを実現した．

6.2 ランダムリードプログラムによる性能評価

まず，ランダムにファイルをリードするプログラムを作成して提案手法の性能評価を行った．論理ボリュームの前半(HDD1 の領域)に約 4MB のファイルを 1024 個配置する（これがマイグレーション前となる）．そして作成した 1024 個のファイルの半分 512 個のファイルを論理ボリュームの後半(HDD2 の領域)に移動させる（これがマイグレーション後となる）．このマイグレーションを 1024 個のファイルをランダムに読みこむベンチマークプログラムの実行中に行い，ベンチマークプログラムへの影響を測定した．実験はベンチマーク並列スレッド数 1，2，3 にて行った．

最初にスレッド数 3 における実験結果を 図 11 に示す．従来手法（非オンメモリ型）のマイグレーション手法の性能を図 11-a，提案手法（オンメモリ型）のマイグレーション手法の性能を図 11-b である．横軸が実験時間，縦軸がランダム読み込みベンチマークにおける読み込み時間である．また，マイグレーション実行前，マイグレーション実行中･序盤, マイグレーション実行中･中盤, マイグレーション実行中･終盤, マイグレーション実行後の平均読み込み時間が図 11-c である．図 11 より従来手法においてマイグレーションにかかった時間は約 186 秒であり，図 11 より提案手法ではマイグレーションにかかった時間は 130 秒であることが分かる．よって，提案手法によりマイグレーションにかかる時間が短くなったことがわかる．また，図 11-c のマイグレーション中のファイル読み込み時間を比較すると本手法はマイグレーション負荷を低減できており，ベンチマークプログラムに与える影響が小さくなっていることがわかる．ただし，ベンチマークプログラムに与える影響は依然として小さくなく，さらなる改善が望まれることも確認された．

スレッド数 2 における実験結果を図 12 に示す．図よりスレッド数 3 の場合と同様にマイグレーションにより性能が大きく向上すること，提案手法によりマイグレーション負荷が減少することが確認された．

スレッド数 1 における実験結果は図 13 の通りである．スレッド数 1 では提案手法によるマイグレーション負荷の低減は確認されたが，マイグレーションによる大きな性能向上は確認されなかった．これはスレッド数 1 では複数の HDD が同時に稼働することが無いためである

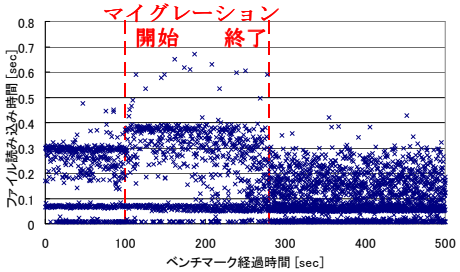

図 11-a 従来手法による結果

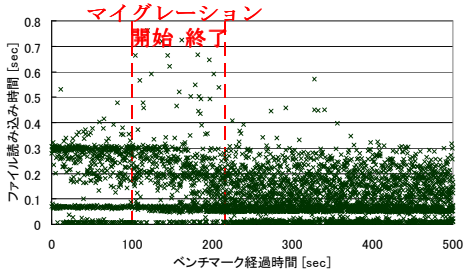

図 11-b 提案手法による結果

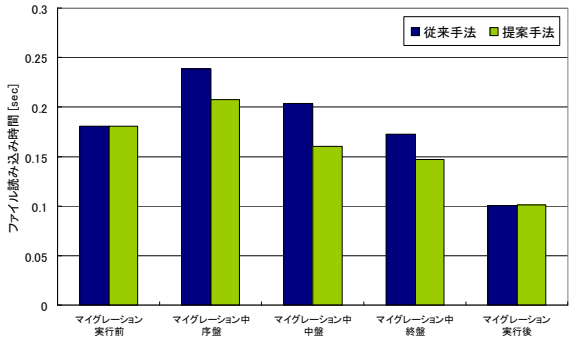

図 11-c 平均ファイル読み込み時間の比較

図 11 ベンチマークスレッド数 3 における結果

Vol.2011-DPS-146 No.9
Vol.2011-CSEC-52 No.9
2011/3/10

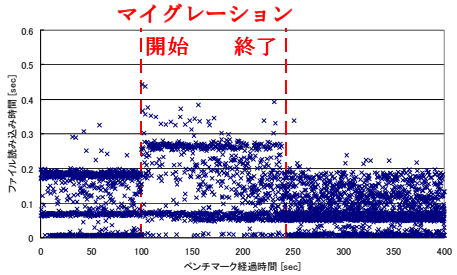

図 12 -a　従来手法による結果

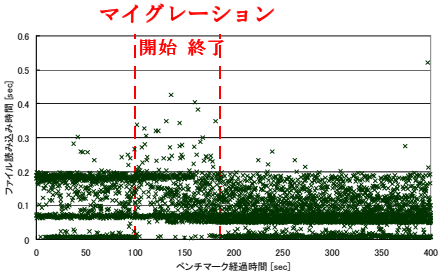

図 12-b　提案手法による結果

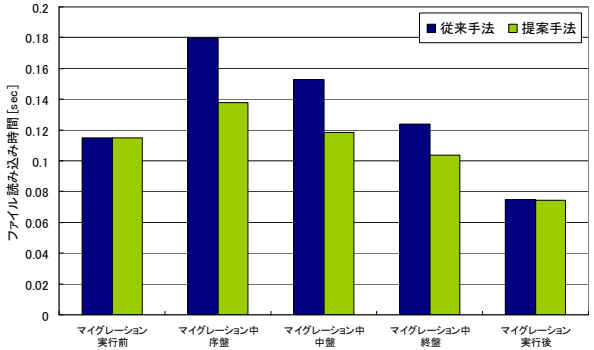

図 12-c　平均ファイル読み込み時間の比較

図 12　ベンチマークスレッド数 2 における結果

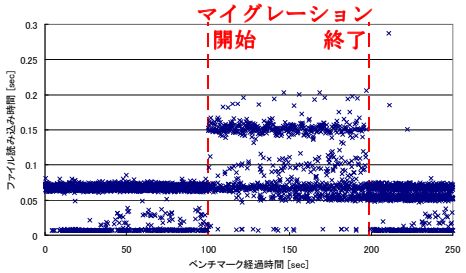

図 13-a　従来手法による結果

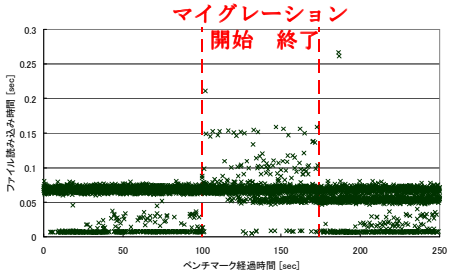

図 13-b　提案手法による結果

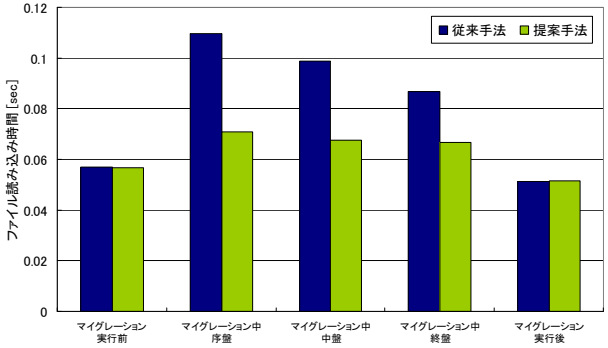

図 13-c　平均ファイル読み込み時間の比較

図 13　ベンチマークスレッド数 1 における結果

ランダムリード実行中の HDD1 と HDD2 の I/O 使用率（I/O キューが空でない確率）を測定した．これらを図 14～図 16 に示す．図 14-a と図 14-b を比較すると，従来手法ではマイグレーションの開始により HDD1 の使用率が一時的に増加しており，マイグレーション終盤まで HDD1 の使用率の大きな減少がないことが分かり，提案手法では HDD1 の使用率の増加はなくマイグレーション序盤より HDD1 の使用率の大きな現象があることがわかる．同様に図 15-a と図 15-b や，図 16-a と図 16-b を比較すると，提案手法ではマイグレーションの中盤から使用率が減少するが，従来手法では終盤まで減少がほとんど見られないことがわかる．

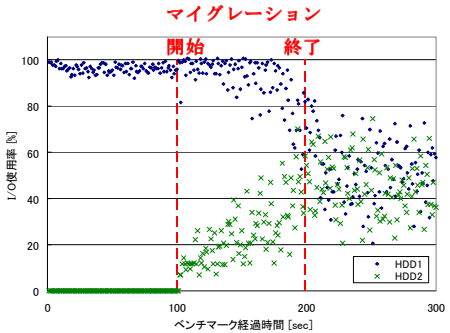

図 14-a　従来手法による結果

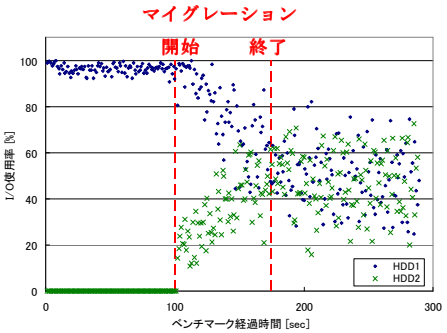

図 14-b　提案手法による結果

図 14　HDD I/O 使用率（ベンチマークスレッド数 1）

Vol.2011-DPS-146 No.9
Vol.2011-CSEC-52 No.9
2011/3/10

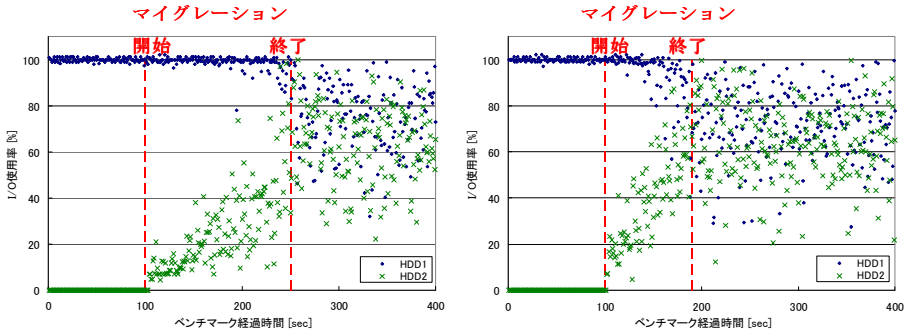

図 15-a　従来手法による結果　　図 15-b　提案手法による結果
図 15　　HDD I/O 使用率（ベンチマークスレッド数 2）

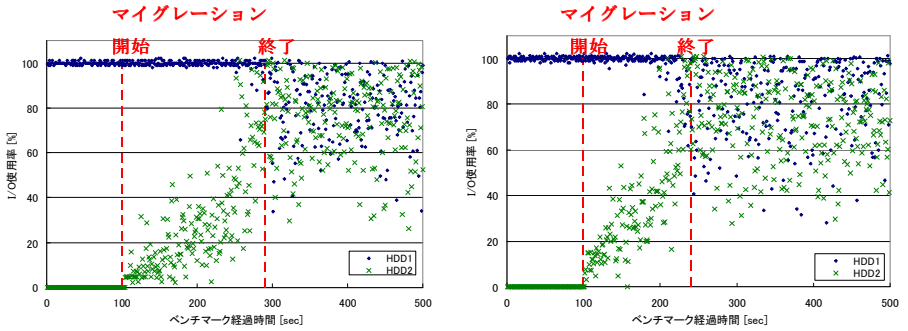

図 16-a　従来手法による結果　　図 16-b　提案手法による結果
図 16　　HDD I/O 使用率（ベンチマークスレッド数 3）

6.3　FFSB による性能評価

次に，ベンチマークソフト FFSB を用いて性能測定を行った．測定時に使用した FFSB の設定はファイルサイズ 4MB，ファイル数 1024 個，Read のみとした．FFSB 実行中に 512 個のファイルのマイグレーションを行い，FFSB にて得られた性能を測定した．実験結果は図 17 の通りである．提案手法適用時の方が FFSB ベンチマークの処理性能が高く，本結果からも従来の非オンメモリ型のマイグレーション手法に比べオンメモリ型の提案手法の方が優れていることが確認された．

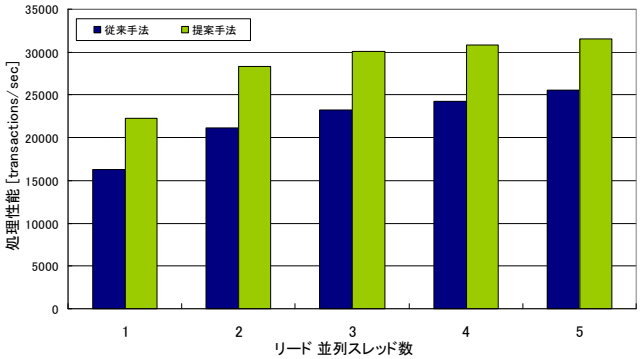

図 17　　FFSB による処理性能の比較

7.　おわりに

本稿では，ストレージ装置の I/O 速度の高速化手法の 1 個であるストレージ間データマイグレーションに着目し，その課題であるマイグレーション負荷を軽減する手法を提案した．具体的には，ファイルシステム層においてマイグレーションを実行し，ファイルアクセス履歴を用いてメインメモリにキャッシュされているファイルを対象としてマイグレーションを行う手法を提案した．提案手法を実装し性能評価を行ったところ，従来の非オンメモリ型より小さい負荷でデータマイグレーションを実行可能であることが確認された．

今後はさらなる負荷低減についての考察，ファイルシステム内(カーネル空間内)での実装を行っていく予定である．

謝辞　本研究は科研費（22700039）の助成を受けたものである．

参考文献

1)　Gerhard Weikum, Axel Moenkeberg, Christof Hasse, Peter Zabback: Self-tuning Database Technology and Information Services: from Wishful Thinking to Viable Engineering, Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002
2)　小林 大，田口 亮，横田 治夫: 並列ストレージにおけるサービス性能を保った負荷均衡化の影響，電子情報通信学会 信学技報，DE2006-129, DC2006-36

Vol.2011-DPS-146 No.9
Vol.2011-CSEC-52 No.9
2011/3/10

3) Haruo Yokota: Autonomous Disks for Advanced Database Applications, in Proc. of 1999 International Symposium on Database Applications in Non-Traditional Environments (DANTE'99), pp.441-448, 1999.11
4) Hai Huang, Wanda Hung, Kang G.Shin: FS2: Dynamic Data Replication in Free Disk Space for Improving Disk Performance and Energy Consumption, SOSP 2005 pp. 263-276
5) 小山 芳樹, 山口 実靖: ファイルアクセス履歴を用いたオンメモリ型ストレージ間データマイグレーション, 情報処理学会 全国大会, 4L-3