並列ストレージにおけるサービス性能を保った複製利用負荷均衡化に対する更新リクエストの影響

小林　大[†,††]　田口　亮[†††]　横田　治夫[††††,†]

† 東京工業大学大学院情報理工学研究科計算工学専攻　〒 152–8552 東京都目黒区大岡山 2–12–1
†† 日本学術振興会特別研究員 DC
††† 日本放送協会放送技術研究所　〒 157–8510 東京都世田谷区砧 1–10–11
†††† 東京工業大学学術国際情報センター　〒 152–8550 東京都目黒区大岡山 2–12–1
E-mail: †, ††daik@de.cs.titech.ac.jp, †††taguchi.r-cs@nhk.or.jp, ††††, †yokota@cs.titech.ac.jp

あらまし　並列ストレージシステムではストレージノード間アクセス負荷の均衡化が肝要である．しかしワークロードの変化に対する動的負荷均衡化を，データ再配置によって行う手法は，それ自体がアクセス能力の一部を使用するため性能保証が困難である．著者らは，データマイグレーションによる負荷均衡化に，複製データを併用し，性能維持を行う手法を提案している．提案手法では，ストレージノード中に格納されるバックアップデータを用いたデータ移動経路変更及びアクセス回送により，円滑にマイグレーションを行う資源を確保する．本稿では，提案手法使用下において，データへの更新リクエストが存在する場合の影響を考慮した改善について考察を行った．その結果，アクセス回送の効果はあまり見られず，データ移動経路変更が有効である結果を得た．

キーワード　ストレージ技術, 性能保証, データマイグレーション, レプリケーション, 性能評価

Corresponding to Write Accesses on Replica-assisted Migration for Providing Generarl Storage QoS

Dai KOBAYASHI[†,††], Ryo TAGUCHI[†††], and Haruo YOKOTA[††††,†]

† Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology
2–12–1 Ookayama, Meguro-ku, Tokyo 152–8552, Japan
†† Research Fellow(DC) , Japan Society for the Promotion of Science
††† Science and Technical Research Laboratories, Japan Broadcasting Corporation
1–10–11 Kinuta, Setagaya-ku, Tokyo 157–8510, Japan
†††† Global Scientific Information and Computing Center, Tokyo Institute of Technology
2–12–1 Ookayama, Meguro-ku, Tokyo 152–8550, Japan
E-mail: †, ††daik@de.cs.titech.ac.jp, †††taguchi.r-cs@nhk.or.jp, ††††, †yokota@cs.titech.ac.jp

Abstract　Data migration is an efficient method to handle skew of access-request distribution in parallel databases and distributed storage systems. However, it also causes performance degradation during the migration process temporally because it uses system resources such as local I/O or networks. To solve this kind of problems, we have proposed a method *replica-assisted migration* that use temporally replica data in other storage nodes during executing data migration. In this paper, we evaluate an efficency of the method under workloads with update operations and also improve the method. Using a storage system simulator and file server trace workloads, migration routing algorithm in the method work better, while access forwaring algorithm in it can not show effective working.

Key words　Network storage , QoS, Data migration, Replication, Performance evaluation

1. はじめに

データマイグレーションは並列構成のデータベースやストレージシステムにおけるアクセス負荷均衡化手法である．ネットワーク結合されたストレージノード間において各データのアクセス負荷傾向を考慮しデータを移動することでデータ配置を変化させ，性能を大きく低下させるアクセス集中を回避することができる．このようなデータマイグレーション技術は自律的なパフォーマンスチューニング手法として提案され，効果を得ている [1,2].

しかし，データマイグレーションは，ディスクアクセスやネットワーク転送を伴うため，ストレージノードの性能を一時的に低下させることが問題となる [3–5]．アクセスパターンの変化が激しい場合や，負荷偏りの評価精度が低い場合，マイグレーションにより負荷を追い出そうとするディスク能力が既に飽和しており，マイグレーション処理によりディスク単体のスループットを低下させてしまう恐れがある．

我々はこれまでに，複製データを利用することでマイグレーションによる資源利用を他のディスクに分散する手法 Replica-assisted Migration を提案してきた [6–9]．Replica-assisted Migration は，我々の提案する並列ストレージシステムである自律ディスク [10] のデータ配置戦略において，データマイグレーションに起因するノード負荷上昇を効果的に分散するために複製データを利用し，マイグレーション移動元，移動先ノードの性能が性能要件を満たさない状態に陥ることを防ぐ手法である．

しかし，これまでの研究では，データ読出し要求とデータ更新要求が混在するアクセス環境下における評価を行っていなかった．提案手法ではバックアップデータを正規のサービスにも用いるために，同期更新が必要となり，データ更新要求により複製格納ディスクへのアクセスが発生する．このため，データ更新要求も考慮したマイグレーション戦略，アクセス回送戦略が必要となる．また，これまでの研究では人工的なワークロード下での実験しか行っておらず，より現実の運用に近いワークロードに対する評価と有効性の確認は行っていなかった．

本稿では，これまで提案してきた Replica-assisted Migration に対し，その適用可能性の評価を目的とする．本稿では，より現実的で更新要求の混在するワークロードとしてファイルサーバから採取されたワークロードを用い，これに対し本手法の適用を考察する．ファイルサーバは，近年のストレージセントリック構成によるシステム構築と，個人の扱うデータ量の爆発的な増大により大規模化する傾向にあり，ネットワーク並列ストレージシステムにおける重要な適用対象のひとつである．Hsu らの報告による近年のファイルサーバにおけるワークロードの特徴 [11] として，アクセスリクエストのうち 30%〜60%が Write リクエストである更新要求であり，また格納データサイズ，アクセスサイズとも均一でなくまたアクセス偏りが大きいことがあげられる．

前提となるマイグレーション戦略を，ファイルサーバワークロード向けに改良し，その上で Replica-assisted Migration のアルゴリズムをデータ更新要求を含むものに改良した．シミュレーションでそれらの挙動を観察した．その結果，提案手法のうちマイグレーション経路変更が効果を得ていることを確認した．また，経路変更を適応的に行うことでマイグレーション実行時間の削減効果が得られることを確認できた．

本稿の構成を以下に示す．つづく 2. で，並列ストレージとデータマイグレーションについて説明する．3. でこれまで提案してきた Replica-assisted Migration について述べる．4. でファイルサーバワークロードへ提案手法を適用する手法を述べ，5. でシミュレーションプログラムによる実験とその結果を考察する．また，実験から得られた知見を基に提案手法を改良し，その効果を確認する．最後に 6. でまとめと今後の課題について述べる．

2. 並列ストレージシステムと負荷均衡化

2.1 並列ストレージシステムと自律ディスク

近年，大規模なストレージシステムは多数のストレージノードをネットワーク結合した並列ストレージシステムとして構成される．そしてデータ群をファイル，エクステント，ページ，ブロック等の粒度で分割を行い，分散格納する．データ配置管理の複雑さを利用者から隠蔽するため，システムはメタデータサーバや分散ディレクトリ [12,13] 等によるストレージ仮想化機構により仮想化され，単一の巨大ボリュームとして提供される．各クライアントからのアクセスリクエストは仮想化機構を通じて適切なストレージノードへと送られて処理される．並列ストレージシステムでは，ノード故障によるデータ喪失を防ぐため，他のノードのバックアップデータを保持する構成をとるものが多い．

我々はこれまで可用性やスケーラビリティに優れた高機能並列ストレージシステムである自律ディスク [10] を提案している．自律ディスクでは，Primary-Backup の one-copy [14] 複製により管理する障害復旧用バックアップデータを，Chained Declustering [15] 複製配置戦略に基づきストレージノード間で保持し合うことで信頼性を高めている．また，データ配置管理に分散 B^+-Tree による分散仮想化機構を有し，値域分割によりデータ格納ノードを決定する．

2.2 ストレージ装置におけるアクセス集中の弊害

多数のストレージ装置により構成される分散ストレージシステムではアクセス負荷偏りにより著しく性能が低下するため，アクセス負荷偏りは常時解消する必要がある．

アクセス負荷が各ノード間で均衡していた場合に比べて，偏りが生じた場合には，アクセス集中ノードのレスポンスタイムが著しく増加する [16]．そのため，システムに余剰資源があるにも関わらず，求められるサービス品質を満たせなくなる可能性がある．

2.3 ノード間データ移動による負荷均衡化

ストレージノード間のアクセス負荷を均衡化する手法として，データマイグレーションが提案されている．データマイグレーションでは，各ノードの性能を考慮し，格納データに対するアクセス負荷が均等になるようにデータを動的に再配置する．適切な負荷評価と移動戦略を用いることで，アクセス負荷偏りを

除去可能である．

2.3.1 マイグレーション戦略立案

前提とした均衡化アルゴリズムは次の通りである．全てのノードは，自身の現在の負荷を記録しており，あるインターバル時間ごとに次の作業を行う．まず全ノードは，ある一つのノード (以下では coordinator と呼ぶ) を選定し，coordinator に対し直接，あるいは段階的 [17] に負荷情報を送信する．負荷情報を受け取ったノードは，全てのノードで負荷が平坦化するような移動戦略（移動元ノード，移動先ノード，移動負荷量）を計算する．計算結果である負荷量にスピードファクター値 $\theta(0 \leqq \theta \leqq 1)$ を乗じ，各ノードに戦略の該当部分を送信する．各ノードは自身の格納データに関する負荷情報を走査し，与えられた移動負荷量を満たすデータセットを特定する．各ノードは指定された移動先ノードに必要なデータを送信する．

本稿におけるデータマイグレーションでは，上記のようにノードごとの負荷情報を集約し，戦略を立案することを仮定する．また格納データごとの負荷情報を集約することは難しいため，移動データの特定は各ノード内で行う．

2.3.2 マイグレーションによる負荷上昇の問題

ノードがそのサービスに利用するためのディスクアクセスやネットワーク転送能力の一部がデータを移動するため利用され，負荷が集中してからデータマイグレーションを行うことは一時的にさらに性能を悪化させる．負荷評価精度の問題や利用者傾向の変化，あるいはノード故障などのシステム構成の変化により，あるノードの負荷が急激に高まることは往々にしてあり得る．一度負荷が偏ると，単純なマイグレーションではレスポンスタイムの低下によりシステムに求められるサービス品質を満たせなくなる．

データマイグレーション中のサービス性能維持に関して，既存研究は主に階層構成のストレージシステムにおける階層間のデータ移動を目的としており，多くの研究は適応的にマイグレーション速度を調節するアプローチを取る [3–5]．しかし，負荷均衡化のためのマイグレーションの場合，変化するワークロードに対しマイグレーション速度を維持することも重要であるため，速度調節によらないアプローチが肝要である．

3. Replica-assisted Migration

我々は，適応的速度調節によらない，マイグレーション中のサービス性能維持手法として Replica-assisted Migration を提案している [6–9]．本節ではその一部を述べる．

3.1 概　　要

Replica-assisted Migration では，2. で述べた他のノードに配置された複製データを利用することで負荷集中ノードから一時的にリソースを捻出し，ノードに負荷が集中した後でも，データマイグレーションが行うことを可能とする．複製の利用方法として，アクセス回送 [6] と，マイグレーション経路変更 [8] がある．上記に加え，マイグレーション時以外にも複製データへのアクセス回送を行う手法 [8] を提案しているが，本稿では説明を割愛する．

提案手法の特徴として，移動元ノードの負荷量が最大許容量に近い場合でもデータマイグレーションによる負荷均衡化を行うことが可能であることが挙げられる．

なお，Replica-assisted Migration で用いるアルゴリズムの多くは，2.1 で述べた Chained Declustering と値域分割のもつデータ配置に対する制約を暗黙的に用いているが，その他のシステムにも拡張可能である．

3.2 複製へのアクセス回送併用制御

提案手法では障害復旧用の複製を用いて，移動元のディスクに格納されるデータに対するアクセスの一部をデータマイグレーション実行中のみ複製データを持つノードに回送することで，負荷集中ノードから一時的にリソースを捻出し，ノードに負荷が集中した後でも，データマイグレーションを行うことを可能とする．

提案手法は，2.3.1 で述べたマイグレーション実行時に次に述べる処理を加える．coordinator において負荷評価により算出された負荷量から，マイグレーションを行うデータ量，及びアクセス回送率 r を決定する．複製との一貫性保持が非同期であれば同期に切り替え，複製に対してまだ適用されていない更新を適用する．移動元ディスクに対するアクセス要求を回送率 r の割合で，複製の存在するディスクへと回送するようにセットする．データを移動する．その間のアクセス要求は r に基づき複製へ回送される．データ移動完了後, 複製への要求回送率を 0 にする．

3.2.1 回送率決定アルゴリズム

coordinator における回送率決定は次のように行う．ここで $L_i(t)$ は採集されたノード i のプライマリデータへのアクセス負荷，$L_i^{\mathrm{TMP}}(t)$ は計算中に更新されたノード i 格納データへのアクセス負荷を表す．

まず，現在の負荷 $L_i^{\mathrm{TMP}}(t)$ とマイグレーション負荷 L_{MIG} の合計が，性能要件負荷 Lm_i を超えるノード i を探す．ノード i からの回送負荷量 h_i がノード i のプライマリ負荷量より小さく回送できる負荷量が存在する場合，次にノード i の超過負荷 $L_i^{\mathrm{TMP}}(t) - L_{\mathrm{MIG}}$ を，$L_i^{\mathrm{TMP}}(t)$ から引き複製ノードの負荷 $L_{i+1}^{\mathrm{TMP}}(t)$ に加える (回送)．ノード i の回送負荷量 h_i も同様に増加する．続いて回送により複製保持ノードの負荷 $L_{i+1}(t)$ が性能要件 Lm_{i+1} を超過したら，それを解消するために，同様にノード $(i+1)$ からノード $(i+2)$ への回送による負荷量値更新を行う．

以上を $h_j > L_j(t)$ となるか，全てのノード i に対して $L_i^{\mathrm{TMP}}(t) < Lm_i$ となり回送割合の更新が起こらなくなるまで行う．最後に得られる回送負荷量に対するノード i のプライマリ負荷量 $h_i/L_i(t)$ が回送率となる．

3.2.2 マイグレーション負荷見積もり

データ移動に伴う各ノードの負荷上昇量については，各ノードのスループット対レスポンスタイム性能曲線を用いて推定する方法を提案している [7]．データ一つあたりの平均サイズ Q_{avg} [Byte]，システムに対する性能要件として定義された単位アクセスあたりのレスポンスタイムを R_{req} [s] を用い，1 回のマイグレーションによる単位時間当たりのアクセス負荷は $Q_{avg}/2R_{req}$ [Byte/s] と考えられる．ここで分母の 2 は移動元

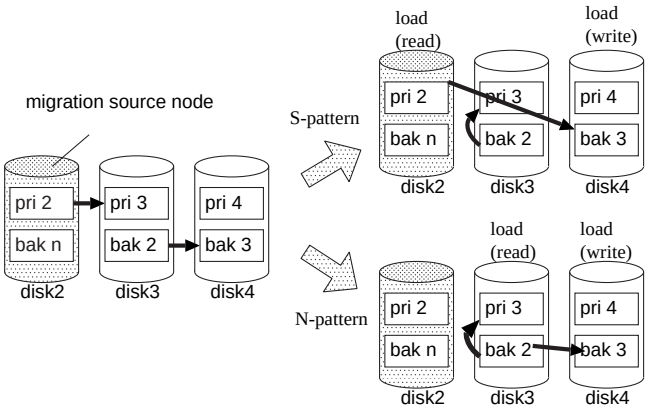

図 1　右方向マイグレーション実行時の 2 種類のデータ経路. disk_i に格納されたプライマリデータ領域を pri_i, その複製データ領域を bak_i と表記した.

からの読出しと移動先への書き込みが逐次実行であることを表しており，同時実行可能な場合は必要ない.

ただしこの式は，ディスクの性能が IO サイズに対して線形であることを仮定している．実際には，磁気ディスク及び磁気ディスクを用いたシステムの性能はサイズに線形でない場合が多い．この是正については本稿 4.2.1 において述べる.

3.3　複製配置を利用したマイグレーション経路変更

複製配置を利用することでマイグレーションによるアクセス負荷が発生するノードの割り当てが可能になる.

自律ディスクはデータ配置戦略として値域分割を，複製配置戦略として Chained Declustering を用いている．これは，格納されるデータを一意の順序付けで並べ，その部分範囲を各ノードに割り当て，さらにデータ配置戦略によって関連ができた”隣の”ノードに対して自身の持つデータの複製を配置している.

よって，右方向と左方向マイグレーションの 2 種類があり，各方向のマイグレーションには，S(Self)-pattern, N(Neighbor)-pattern の 2 通りのデータ経路 [8] が選択できる（図 1）．そこで，マイグレーション戦略，及びアクセス回送率決定時に同時に，各ノードのアクセス負荷を考慮しデータ経路を選択することで，マイグレーションに起因する負荷上昇を遊休ディスクに割り当てることが可能となる.

4.　ファイルサーバワークロードへの対応

本稿では，提案している Replica-assisted Migration の現実の有効性を評価するため，ファイルサーバから採取したワークロードによる評価を行う.

本節ではまずファイルサーバワークロードの特徴を述べ，提案の前提となるマイグレーション戦略の変更について述べる．その後， Replica-assisted Migration がファイルサーバワークロード下で動作可能となるような改良について述べる.

4.1　アクセス傾向の特徴

IBM の Hsu らは 15 の PC と 5 つのファイルサーバ上で採取したトレースログの解析を基に，ファイルサーバにおけるワークロードの一般的な特徴を報告している [11]．本節では以降の議論のためその特徴として，Write リクエスト比率，データサイズ，アクセス偏りについて紹介する.

[性質 1]　更新要求が混在したワークロードである．アクセスリクエスト中に含まれる Write リクエストはリクエスト数，リクエストサイズいずれも全体の 30〜60% 程度である.

[性質 2]　アクセス粒度は小さい．単一アクセス要求の平均リクエストサイズは 7〜9KB で，ほぼ全てのアクセスはリクエストサイズが 100KB 以下である.

[性質 3]　アクセス偏りが激しい．1 日に c 回以上アクセスされるファイル合計サイズ $f(c)$ は正定数 a, b を用いて $f(c) = a/c^b$ と近似できる．また 1 日にアクセス対象となるデータ量は格納データ量の 4〜7% であり，ほとんどのファイルはアクセスされない.

[性質 4]　Read,Write 両要求が混在するファイルは少ない．多くのファイルは Read もしく Write の一方の要求のみが集中する．Read,Write 共に利用されるファイルは全体の 25%以下である.

4.2　前提とするマイグレーション戦略の変更

まず，2.3.1 で述べた，マイグレーションのための移動負荷量決定戦略の変更を行う.

4.2.1　異なる粒度のリクエストサイズへの対応

性質 2 より，異なるサイズのリクエストが混在するが，ストレージノードの応答性能は一般にリクエスト IO サイズに対して非線形であり，サイズが異なるリクエストが混在する場合で特にリクエストサイズが小さい場合，MB/s や IOPS といった単位で性能要件を規定するのは難しい.

今回は，あらかじめ用いるストレージノードに対し各サイズごとのリクエストに対する許容性能を測定し，負荷測定機構内に保持しておくことで，各ノードではアクセス頻度とリクエストサイズの列を測定した値に基づき，負荷値に変換する事とする.

4.2.2　極度に激しいアクセス偏りへの対処

性質 3 により，用いるワークロードではアクセス偏りが極度に激しい．2.3.1 で述べたようにアクセス偏りを平準化する場合，多くのストレージノード上でマイグレーションが発生し，マイグレーションデータ量がワークロード負荷の大きさに比べ膨大になる.

そこで，ノードごとに 4.2.1 で述べた負荷値により定義された許容最大性能を設定し，移動戦略立案時点において，あるノードの負荷が，与えられた許容最大性能値の一定割合以上になったとき，現在のノードの負荷が最大性能以下になるようにマイグレーション戦略を立案することとする.

この変更により必要なマイグレーション量が大きく減少する．後述の実験環境では，平準化の場合ではノード一台あたり 100GB を超えるマイグレーション量が算出される．それに対し，許容最大性能維持の場合ノード一台あたりのマイグレーション量は 1GB 程度になり，かつ全てのノードでアクセス性能を維持し続けた.

4.2.3 バックアップ更新負荷の通知

各ノードのアクセス負荷は格納するプライマリ負荷と，バックアップへの更新負荷の合算値である．これは提案手法がバックアップへのアクセス回送を利用しており，複製のダーティ状態を許容できないためであり，更新負荷は常にプライマリとバックアップ両方のデータに伝播する．

一方，2.3.1 において移動戦略を立てる場合，マイグレーションによって移動されるロードの算出が必要であり，これには上記のように負荷中の更新処理の割合が必要となるが，移動戦略を行う coordinator では各データごとの更新割合情報を閲覧することはスケーラビリティの観点から困難である．よって，戦略立案時に用いる更新割合は，各ノードの負荷収集部分で合算値を取得し，負荷情報と共に収集することになる．

ここで，性質 4 により各ファイルごとの更新割合は 0 か 1 に偏るため，ファイルごとの合算した値を元にバックアップの更新負荷を算出すると誤差が大きい．ノード全体としては更新割合が低い場合でも移動対象となるデータ群の更新割合がそれよりも実は高い場合もあり，負荷評価の誤差を大きくする要因となる．

そこで本稿では，各データの更新割合をマイグレーション実行時に動的に取得することとする．各ノード上にはマイグレーションプロセスが動作しており，移動対象データの複製の移動先に対し，当該データの負荷値に更新割合を乗じた負荷値が加算されることをそのたびに通知する．まだ通知された側のノードにおいてマイグレーションが行われていなければ，残りの移動負荷量に通知量を合算する．

4.3 Replica-assised Migration の改良

続いて，提案する Replica-assisted Migration の改良について述べる．

4.3.1 一貫性保持

プライマリとバックアップは常に同期更新されるとする．

本提案の適用対象として前提としている自律ディスクではプライマリバックアップ方式の one-copy 複製を利用している．クライアントからのアクセスはストレージ仮想化装置を経て，まずプライマリデータを格納するストレージノードへ到達する．プライマリ格納ノード上あるいは仮想化装置上でロック取得等の並行性制御を行うことで，アクセス系列は直列化でき，プライマリとバックアップの一貫性は保たれる．その後，回送判定が行われ，必要であればバックアップ格納ノードへ回送される．

4.3.2 回送率決定アルゴリズムの改良

アクセス回送は Read リクエストのみで可能であるため，ノード i 格納データの平均更新割合 $u_i(0 \leqq u_i \leqq 1)$ を用い，回送率の計算を次のように計算する．

まず，3.2.1 の戦略のうち，各ノードの負荷量をプライマリデータへのアクセス負荷量に，同期更新によりバックアップに伝播されるバックアップデータのアクセス負荷量を加えたものとする．また回送可能な余剰負荷量がまだ存在するか否かを判断する際，現在の自分のプライマリへのアクセス負荷量 $L_i(t)$ ではなく，そのうちの Read リクエストの量 $L_i(t)(1-u_i)$ を用いる．同様に，3.2.1 の戦略で得られた回送対象負荷量 h_i について，$h_i/\{L_i(t)(1-u_i)\}$ をノード i の Read リクエスト回送率とする．以上により，ノード i へのアクセス負荷のうち Read リクエストのみを対象とした回送率を得ることができる．

表 1　シミュレーション構成設定

simulation parameter	value
Rotational Speed (RPM)	7200
# of surfaces	4
# of sectors per cylinder	2520 - 5184
# of zone	29
Full stroke seek time (msec)	14.7
Single track seek time (msec)	0.8
Head switch time (msec)	1.4
buffer size (KB)	8196
disk node	16
cache-size (MB)	64
Block-size (KB)	32
Network-bandwidth (Mbps)	800

5. 実　　験

これまでに述べた手法について，実際のファイルサーバから採取されたトレースログを基にしたワークロードに対する挙動をシミュレーションプログラム上で観察した．

5.1 実験環境

5.1.1 システム構成

待ち行列を利用したイベント駆動型のストレージシミュレーションプログラムを構築し，実験に用いた．シミュレーションはイベントタイマー，イベント，モジュール，ジョブで構成される．イベントタイマーにより起動されたイベントがジョブをモジュールに投入することでシミュレーション時間が進む，イベント駆動型を取る．本シミュレーションプログラムの詳細については以前の発表 [18] を参考にされたい．シミュレーション内におけるディスクアクセス時間は，HGST deskstar T7k250 [19] を基にした表 1 に示すパラメータと前回アクセス時のヘッド位置から計算する．

本稿における実験において特筆すべき Write アクセスに対する挙動を以下に述べる．Read アクセスについては，Read-ahead キャッシュを利用するが，Write アクセスはライト・スルー設定とし，同期的にディスクに書き込む．これらは現在用いられているファイルサーバの設定とは異なり，Write アクセスに対する応答性を大きく減少させるが，障害に対するデータ喪失を考慮し，本実験では上記設定を用いた．また複製との一貫性保持も同期的に行い，複製保持ノードからの書き込み終了を待ってからクライアントへ終了通知を返す．ディスクアクセス待ちキュー内では，重複したブロックへの書き込みリクエストの除去のみを行い，コマンドリオーダリングは行わないこととした．

5.1.2 予 備 実 験

表 1 で設定されたシミュレーション内のディスク 1 台の許容性能を算出するための予備実験を行った．ディスク 1 台に対し，同一サイズのデータを合計 1GB 格納し，クライアントノード

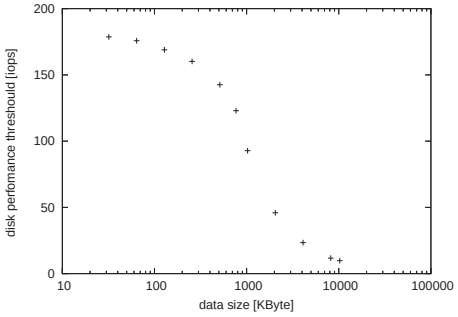

図 2　サイズごとのディスク性能限界点

表 2　ワークロードの特徴

workload parameter	a	b	c	d	e
Head Timestamp (hour)	15	16	17	18	19
Time span (hour)	1.1	1.0	0.9	1.0	0.9
Write Ratio on # (%)	46.9	41.7	38.5	42.7	44.4
Write Ratio on size (%)	43.4	42.2	39.6	42.0	44.7
Accessed size (GB)	36	40	40	42	40
# of accessed files (10^3)	14.0	13.4	11.6	13.4	11.9
# of requests	1.2×10^5				
total file size(GB)	450 × 2 (Primary & Backup)				
# of total files	1.0×10^6				

から平均到着率一定, アクセス偏りなしのポワソン過程による読み出しリクエストを発生させ，平均レスポンスタイムを測定した．そして，各サイズにおいて平均到着率 1MB/s における平均レスポンスタイムの 10 倍を超えるレスポンスタイムを示す平均到着率を当該サイズのアクセスに対する性能限界点であると定義した．各サイズに対する性能限界点となる平均リクエスト到着率を図 2 に示す．768KB 前後でグラフの傾向が変化している．これは，ディスクノードからクライアントノードへのネットワークを 800Mbps の待ち行列として設定しており，800KB を越えるあたりでネットワークキュー側がボトルネックになるためである．

このグラフから最小二乗法により，本実験に用いる設定のディスクに対するデータサイズ s[byte] のアクセス負荷量 $l(s)$ を次の式で得ることができる．

$$l(s) = \begin{cases} s \times 3.38 \times 10^{-9} + 5.49 \times 10^{-3} & (s \leqq 800KB) \\ s/(96 \times 10^6) & (s > 800KB) \end{cases} \quad (1)$$

ただし，ディスクの最大許容負荷量を 1 とした．以降の実験ではこの式を用いて，各アクセスを負荷量に変換する．

5.1.3　ワークロード設定

シミュレーションで用いるワークロードは，Hewlette Packard 社で公開している File System Traces [20] のデータから抽出した．公開されているワークロードは 1 週間分あり，その中の複数の時点から約 1 時間のセットを切り出したものを用いた．各ワークロードの特性の詳細を表 2 に示す．表中の Head Timestamp が各ワークロードの切り出し時点を表す．格納データセットはトレースログと共に配布される ls コマンドのアウトプット中の記述，及びトレースログ中の offset 値を参考に事前に作成した．offset 値を基準としてサイズを決定しているため，中間領域にディスクを割り当てないスパースファイルが実装されたファイルシステムが利用されていた場合，実際より大きなファイルセットが作成されている可能性がある．

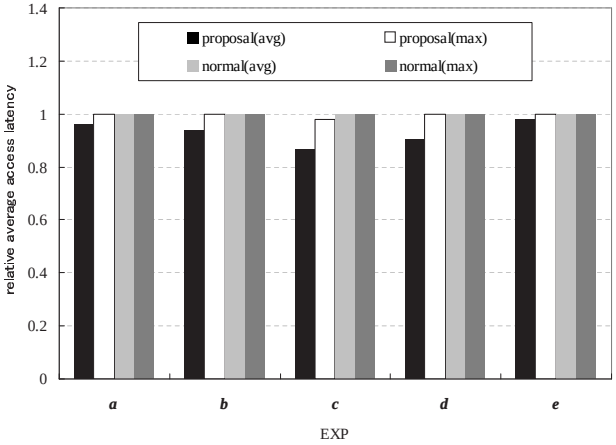

図 3　Replica-assisted Migration の有無によるマイグレーション実行中のアクセスレイテンシ（Replica-assisted Migration を用いないものを 1 とした相対値）

アクセスは，トレースログ中の READ, READV, WRITE, WRITEV コールを抽出し，offset 値と size とあわせ，ステートレスな単一アクセスの集合とした．このため，MMAP によりメモリ空間にマップされたファイルへのアクセスは再現できていない．

我々の手法におけるマイグレーションは値域分割を前提としているため，ファイルごとの値域は重要である．ここでは，トレースログに出現した順に振った仮 ID（sid）に対し，式 (2) に示す変換により値域に分散するような設定で実験を行った．

$$id = sid \oplus ((\overline{sid} \wedge \texttt{0xff}) \times 2^{12}) \quad (2)$$

この変換により，元の sid は 2^{20}(大よその格納ファイル数）の範囲で，連続する値が距離を持つよう分散される．

5.2　提案手法の挙動確認

提案する Replica-assisted Migration の有無のみが異なる上記の設定のシステムにおいて，5 つの時点から取得したワークロードそれぞれに対する各モジュールの動作を記録した．

5.2.1　実 験 結 果

図 3 は，各実験における最初のマイグレーションが発生してから終了するまでの単位ブロックあたりの平均アクセスレイテンシと最大アクセスレイテンシの比較である．横軸がワークロード種別で，縦軸が Replica-assisted Migration を用いない場合の値を 1 とした相対アクセスレイテンシである．

図より，平均アクセスレイテンシに関してはいずれの実験についてもわずかに改善していると言えるが，最大アクセスレイテンシはほぼ変化していないことがわかる．

図 4 に，図 3 と同じ実験における最大アクセスレイテンシの絶対値を示す．縦軸はマイグレーション発生期間中における単位ブロックあたりの最大アクセスレイテンシで単位は [秒/ブ

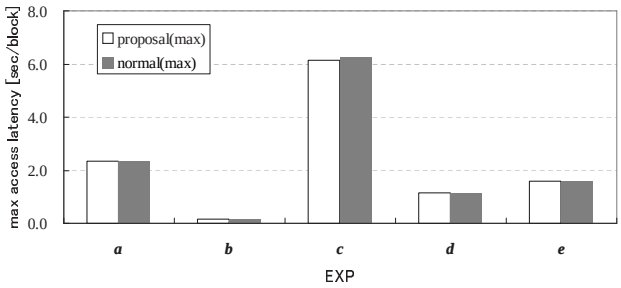

図 4　Replica-assisted Migration の有無によるマイグレーション実行中の最大アクセスレイテンシ．

ロック] である．特に実験 c では長時間待たされているリクエストが存在し，これがほぼ改善されていないことがわかる．

5.2.2　考　　察

実験では提案手法は，平均アクセスレイテンシはわずかに削減できているが，当初の目的としているサービス性能保証の点で効果が低いという結論を得た．これまでの人工的なワークロードに対する挙動との違いについて，各モジュールの動作ログより以下の 2 点が原因として考えられる．

第一に，性質 3 のアクセス偏りの大きさがある．そのため，4.2.2 においてマイグレーション戦略の変更を行い，マイグレーションされるデータ量を削減した．極端なアクセス偏りと併せ，実験中に立案されたマイグレーション戦略の多くは，ひとつの負荷集中ノードから単一方向へのマイグレーションのみであった．現在提案手法による経路選択を行わない場合，マイグレーションによる負荷は図 1 に示す N-pattern を必ず選択する設定としている．よって負荷集中ノードへマイグレーションが負荷をかけることがなく，想定するマイグレーションによる性能低下が発生していない場合が多い．

第二に，データ更新アクセスの負荷がある．性質 1 の通り，Write アクセスの割合は表 2 からアクセスの半数程度であるが，ワークロード中では単一ファイルへの Write が集中する傾向が得られた．さらに Read アクセスは多くがキャッシュにヒットし，ディスクに到達しない．実験では，ヒット率の低いマイグレーションによるアクセスを除くとキャッシュヒット率は 70% から 90%程度であった．この値は Hsu らの報告 [11] における LRU キャッシュミス率調査とおよそ合致している．性質 4 より，Write アクセスの集中するファイルに対しては回送のための Read アクセスがほとんど存在しないため QoS に貢献できない．

5.3　提案手法の改良

このように，ファイルサーバにおけるワークロードでは更新アクセスによる負荷が高く，提案手法のうちアクセス回送はあまり効果が得られない．

そこで，提案手法の経路選択を拡張し，更新アクセスが高い時点で発生したマイグレーションの実行時間の削減を試みる．

5.3.1　適応的経路選択の導入

左方向マイグレーションが起こった時，Write アクセスの割合が 100%に近い時を考える．図 1 において disk2 上の pri2 に更新負荷が集中する場合，同量のアクセスが disk3 上の bak2 にも集中する．いずれの経路においても負荷集中ノードの性能を悪化させてしまうため，その他のアクセスを考慮しないとマイグレーション経路選択が難しい．

そこで，マイグレーションによる負荷を両ノードに適応的に分散することによって，マイグレーションにかかる時間の削減を目指す．マイグレーション時間の削減は直接的にはサービス性能保証には結びつかないが，アクセス負荷評価時とマイグレーション終了時点が近づくことで結果的にアクセス負荷評価の精度向上へとつながる．

提案手法の改良は次の通りである．4.2.3 において，マイグレーション中に動的に更新要求割合の通知を行うこととした．この機構により，更新要求割合に加え，短期的アクセス負荷に関する情報を送信することとする．例えばディスクキュー長等が短期的アクセス負荷を表す．

マイグレーションプロセスは，ひとつのファイルのマイグレーションごと，あるいはあらかじめ定められたまとまったサイズのファイル群のマイグレーションごとに，マイグレーション経路を変更する．短期的アクセス負荷が小さいノードが移動元となるよう，S-pattern もしくは N-pattern を選択する．

これにより，その時点でよりアクセスの少ないノードを選択することができ，結果的にマイグレーションによる負荷上昇を二つのノードに等分に分散することが可能になり，マイグレーション時間を削減することが可能となる．

5.3.2　適応的経路選択の効果

更新負荷の多い場合のマイグレーションに対し，適応的経路選択を行った．ここでは短期的アクセス負荷として，ディスクキュー長を用いた．結果を図 5 に示す．6 つのワークロード下における計 11 回分のデータマイグレーションに対し，静的経路選択に対する時間削減率を縦軸に示す．ここでワークロードは表 2 の a,b,c 及びそのアクセス間隔を 30%縮めたものの 6 つを用いた．先ほどの実験では各実験の最初のマイグレーションのみの比較を示したが，ここでは，同ワークロード下で，同じタイミングで同じ戦略（移動元，移動先，移動量）が立てられたものであれば 2 回目以降も同一のマイグレーションであると考え，比較を行った．図における白い棒は最初の 1 回目を示し，黒い棒はその実験における 2 回目以降のマイグレーションを示している．

図より，多くの場合において時間削減が行われていることを確認できた．平均で 2.6% の時間削減効果が得られ，手法の有効性を確認できた．また，1 回目よりも 2 回目以降の時間削減効果が大きい傾向がある．詳細な解析は今後の課題としたい．

右から 2 番目の実験については適応的経路選択がわずかに劣っている．これは，短期的アクセス負荷が低いにもかかわらず，実は経路対象となったノードの方が負荷が高かったことにより，マイグレーション動作が阻害されたためである．負荷評価の精度向上は今後の課題とする．

6.　まとめと今後の課題

本稿では分散ストレージにおいて，負荷分散のためのデー

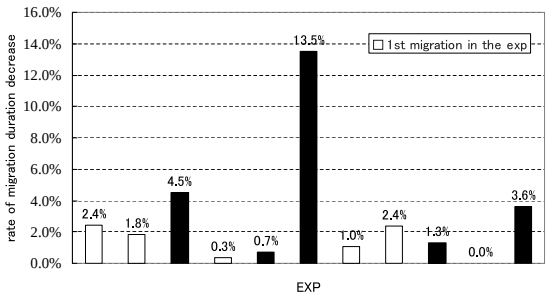

図 5　適応的経路選択によるマイグレーション時間削減: 静的経路選択に対するマイグレーション時間削減率を表す.

タマイグレーションに起因する性能低下を抑えるための手法 Replica-assisted Migration に対し，更新要求の混在する現実的なワークロード下での評価を行った．前提となるマイグレーション戦略を，ファイルサーバワークロード向けに改良し，その上で Replica-assisted Migration のアルゴリズムをデータ更新要求を含むものに改良した．ファイルサーバから採取されたワークロードに対し，シミュレーションで挙動を観察した結果，提案手法によるマイグレーション経路変更は効果を得られるが，アクセス回送が行えない更新要求集中が頻発するため，サービス性能保証効果は低いという結論が得られた．また，更新要求集中時のマイグレーション実行時間の削減のため，経路変更を適応的に行う改良を行い，実験によりその効果を確認できた．

今後の課題を以下に述べる．今回の実験設定では，複製間の一貫性保持やキャッシュ上での書き込みアクセスの取り扱いについて強い制約をおいていた．非同期バックアップやライトバックキャッシュ等を選択することで，システムの振舞いは大きく変わるため，評価を行う必要がある．

また，今回の実験では前提とするマイグレーション戦略自身や負荷の定義，データ配置制約に関する評価は一切行わなかったが，これらの精度は重要であるため，これらの評価と改善も今後の課題としたい．

謝　　辞

本研究の一部は，独立行政法人科学技術振興機構戦略的創造研究推進事業 CREST，独立行政法人日本学術振興会科学研究費補助金特別研究員奨励費, 情報ストレージ研究推進機構 (SRC)，文部科学省科学研究費補助金特定領域研究 (18049026) および東京工業大学 21 世紀 COE プログラム「大規模知識資源の体系化と活用基盤構築」の助成により行なわれた．

文　　献

[1] Gerhard Weikum, Axel Mönkeberg, Christof Hasse, and Peter Zabback. Self-tuning database technology and information services: from wishful thinking to viable engineering. In *VLDB*, pp. 20–31, 2002.

[2] Hisham Feelifl, Masaru Kitsuregawa, and Bengchin Ooi. A fast convergence technique for online heat-balancing of btree indexed database over shared-nothing parallel systems. In *11th Int'l Conf. on Database and Expert Systems Applications*, pp. 846–858, Sep 2000.

[3] Koustuv Dasgupta, Sugata Ghosal, Rohit Jain, Upendra Sharma, and Akshat Verma. Qosmig: Adaptive rate-controlled migration of bulk data in storage systems. In *the 21st International Conference on Data Engineering (ICDE2005)*, pp. 816–827, 2005.

[4] Chenyang Lu, Guillermo A. Alvarez, and John Wilkes. *Aqueduct:* online data migration with performance guarantees. In *Conference on File and Storage Technologies (FAST'02)*, pp. 219–230, Monterey, CA, January 2002.

[5] Jianyong Zhang, Prasenjit Sarkar, and Anand Sivasubramaniam. Achieving completion time guarantees in an opportunistic data migration scheme. *SIGMETRICS Perform. Eval. Rev.*, Vol. 33, No. 4, pp. 11–16, 2006.

[6] 小林大, 渡邊明嗣, 山口宗慶, 田口亮, 上原年博, 横田治夫. 複製データを併用した効率的なデータマイグレーションの検討. 日本データベース学会 Letters, Vol. 3, No. 2, pp. 65–68, Sep. 2004.

[7] 小林大, 渡邊明嗣, 田口亮, 上原年博, 横田治夫. 負荷分散のためのデータ移動による性能低下を抑制するアクセス回送制御. 信学技報，DE2004-112, pp. 35–40. 電子情報通信学会, October 2004.

[8] 小林大, 渡邊明嗣, 田口亮, 上原年博, 横田治夫. データ移動コストとキャッシュを考慮した複製へのアクセス分散制御. 日本データベース学会 Letters, Vol. 4, No. 1, pp. 125–128, Jun. 2005.

[9] Dai Kobayashi, Akitsugu Watanabe, Ryo Taguchi, Toshihiro Uehara, and Haruo Yokota. An efficient access forwarding method based on caches on storage nodes. In *International Special Workshop on Databases For Next Generation Researchers (SWOD 2005)* , pp. 188–191, 2005.

[10] Haruo Yokota. Autonomous Disks for Advanced Database Applications. In *Proc. of International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pp. 441–448, Nov. 1999.

[11] Windsor W. Hsu and Alan Jay Smith. Characteristics of I/O traffic in personal computer and server workloads. *IBM Systems Journal*, Vol. 42, No. 2, pp. 347–372, February 2003.

[12] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. In *7th USENIX Symposium on OSDI*, pp. 205–218, Nov. 2006.

[13] Haruo Yokota, Yasuhiko Kanemasa, and Jun Miyazaki. Fat-Btree: An Update-Conscious Parallel Directory Structure. In *Proc. of the 15th Int'l Conf. on Data Engineering*, pp. 448–457, 1999.

[14] Philip A. Bernstein and Nathan Goodman. An algorithm for concurrency control and recovery in replicated distributed databases. *ACM Transactions on Database Systems*, Vol. 9, No. 4, pp. 596–615, December 1984.

[15] Hui-I Hsiao and David J. DeWitt. Chained declustering: A new availability strategy for multiprocessor database machines. In *Proceedings of the Sixth International Conference on Data Engineering*, pp. 456–465, Los Angeles, CA, February 1990. IEEE Computer Society.

[16] Huseyin Simitci. *Storage Network Performance Analysis.* Wiley Technology Publishing, 2003.

[17] 渡邊明嗣, 横田治夫. 分散ディレクトリ探索コストを考慮した並列データアクセス偏り制御. 電子情報通信学会和文論文誌 D1, Vol. 85-D1, No. 9, pp. 877–886, September 2002.

[18] 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. 信学技報 vol.106, No.291, DE2006-129, pp. 19–24. 電子情報通信学会, October 2006.

[19] Hitachi Global Storage Technologies. *Deskstar T7K250 Hard Disk Drive Specification.* http://www.hitachigst.com, ver. 1.7 edition, 2006.

[20] Storage Systems Lab, Hewlett-Packard Labs. Publicly-available storage traces("file system traces"). http://-tesla.hpl.hp.com/public_software/.