負荷変動傾向を考慮したストレージ間データ移動と複製間アクセス回送の協調制御

小林　大[†]　渡邊　明嗣[†]　田口　亮[††]　上原　年博[††]　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
†† NHK 放送技術研究所
††† 東京工業大学 学術国際情報センター
E-mail: †{daik,aki}@de.cs.titech.ac.jp, ††{taguchi.r-cs,uehara.t-jy}@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障が困難である．本研究では負荷均衡化と要求性能保障の両立を目的とし，我々の提案する自律ディスクに対しレプリケーションによる負荷分散を導入する．各ストレージノード上のキャッシュを考慮した回送先アクセス対象の決定方法について考察し，ノード上キャッシュヒット率をパラメータとした回送先決定手法を提案する．また，自律ディスクが従来から備えるデータ移動による各負荷均衡化手法とレプリケーションの切り替え・併用制御手法を提案し，自律ディスクにより構成されたシステムにおいて要求性能保障を達成する．

キーワード　ストレージ技術, 複製キャッシュ, 性能評価

A combination control of data migration and data replication based on load trend

Dai KOBAYASHI[†], Akitsugu WATANABE[†], Ryo TAGUCHI[††], Toshihiro UEHARA[††], and Haruo YOKOTA[†††,†]

† Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology
†† NHK Science & Technical Research Laboratories
††† Global Scientific Information & Computing Center, Tokyo Institute of Technology
E-mail: †{daik,aki}@de.cs.titech.ac.jp, ††{taguchi.r-cs,uehara.t-jy}@nhk.or.jp, †††yokota@cs.titech.ac.jp

Abstract　In parallel storage systems, it is significant to handle skew of access-request distribution. It, however, is difficult to guarantee performance when dynamic load balancing functions are executed because they consume resources of systems. In this paper, we introduce a replication function to the Autonomous Disks, toward realization both skew handling and performance guarantee. In first, we discuss about a method to determine the access forwarding schemes efficiently, considering effective utilization of cache memories on each storage nodes. And we propose the method that uses a hit ratio of each cache memory to recognize the popularities of data. Additionally, we also propose a control method to coordinate replication and migration functions complementary.

Key words　storage system technology, replication and cache, performance evaluation

1. はじめに

近年，扱うデータの大規模化やそれに伴う管理コスト肥大化により，ストレージシステムの自律管理機能が注目されている．特に CPU やメモリ等を組み込んだ多数のストレージノードをネットワーク結合したシステムを構成し，各ストレージノード上の資源により自律管理を行うシステムが多数提案されている．

並列ストレージシステムにおいて，システムの資源を有効に利用し高いコストパフォーマンスを得るためには，アクセス負荷の均衡化が重要である．

そのような自律的パフォーマンスチューニングのひとつにデータマイグレーションがある．分散配置されたストレージノードに対して，各データのアクセス負荷を考慮してデータ配置を決定することによりストレージ資源の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．また，システム運用中に逐次計測している各ノードの負荷情報を利用し動的にデータ再配置を行うデータマイグレーションが並列データベースやストレージシステムにおける自律的性能管理手法として提案され，効果を得ている [1].

また，高機能ストレージによるシステムや並列データベースでは，ノード故障からデータ喪失を防ぐためデータの複製を複数の異なるノードに保持している．冗長符号による障害回復は，ノード故障時のデータ復旧処理性能低下を伴うためである．この障害対策のための複製へアクセスの一部を回送（転送）することで，ノード間の負荷分散を達成できる [2,3]. 加えて，複数の複製へのアクセス回送割合を，動的に変更することで，変化する負荷傾向へも性能低下なく適応可能である [4]

しかし，データマイグレーションによる負荷分散は，ディスクアクセスやネットワーク転送を伴い，ストレージ装置の性能を一時的に低下させることが問題となる．これにより，当該ディスク上のデータに対するアクセス要求のレスポンスタイムが悪化し，システムに要請されるサービス品質が満たされなくなる．一方で，複製のみによる負荷分散は，静的な複製配置においてはデータマイグレーションのように大きな負荷偏りの変化に追従できず，動的な複製配置においても負荷偏りの変化に起因する複製配置変更時のアクセスによる性能低下といった問題がある．

我々はこれまで，自律ディスク [5] と呼ぶ可用性やスケーラビリティに優れた高機能ストレージシステムを提案している．自律ディスクはシステムはネットワークに接続された高機能ディスクノードのクラスタにより構成される．自律ディスクでは，データマイグレーションによるノード間負荷均衡化機能を備えるため，性能低下の問題があった．我々はこれまでに，Replica-assisted Migration [6] および TLE レプリカアシスト [7] を提案してきた．これは，自律ディスク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マイグレーション時のみレプリケーションを併用し，マイグレーション移動元，移動先対象ノードの性能が性能用件を満たさない状態に陥ることを防ぐ手法である．

本稿では，さらにこの方式を拡張し，通常時においてもレプリカへのアクセス回送を利用する環境において，データマイグレーションをどのように行うかについて議論する．データレプリケーションの常用により少量で高頻度の負荷偏り変化に対し不必要なデータ移動を削減することが可能となり，システム資源をさらに有効利用可能となる．

まず，通常レプリケーションを効率よく行うため，自律ディスクで用いている複製配置戦略である Chained Declustering [8] 環境下における，キャッシュヒット率を考慮したアクセス回送方法について検討する．

また，負荷傾向の大きな変化に対応するために，レプリケーションとマイグレーションの切り替え・併用制御を行う RM-Combination を提案する．RM-Combination では，変化する負荷傾向を元に切り替え閾値を逐次求める．またマイグレーション実行時には，マイグレーションによる増加負荷量が可能な限り各ノードに均一に渉るようなデータ移動経路の選択を行う．さらに，マイグレーションによる増加負荷量をレプリケーションの回送率算出に利用し，負荷をより多くのノードに分散する．

本手法に対し，自律ディスク環境化において実験を行い，有効性について論ずる．

本稿の構成は以下のとおりである．つづいて 2. においてアクセス回送先判定手法に関する考察と計算機シミュレーションによる実験結果を述べる．3. で自律ディスクにおけるレプリケーションとマイグレーションの切り替え・併用制御法である RM-Combi について述べ，4. にて PC クラスタ上に実装された自律ディスクシミュレーションに提案手法を実装し，その有効性を示す．その後，5. で関連研究について述べ，最後に 6. にてまとめと今後の課題を述べる．

2. アクセス回送判定手法

高機能ストレージや，PC クラスタでは各ストレージノードは独立のキャッシュ（バッファ）メモリを有している．単純に全てのデータに対するリクエストを確率によって回送するか否か判定すると，1 種類の同じデータが複数のノードでキャッシュされ，ヒット率は減少する．データの複製が複数のノードに格納されている場合，回送の判定時にキャッシュのヒット率も考慮しなくてはいけない．

D-SPTF [9] では複製を有している全てのノードに対してリクエストをマルチキャストし，最初に返答があったもの，つまりキャッシュにヒットしたノード以外のリクエストをキャンセルすることで，システム内に同じデータが複数キャッシュされることを防ぎスループット向上を達成している．しかし，この方式では余分なネットワークアクセスが発生するため，非常に低いネットワークレイテンシ環境下のみでしか有効性を示せていない．

本稿では，各リクエストが目的とするデータのアクセス頻度によって回送するか否かを判定する手法について検討する．また，実装時にアクセス頻度順位を得ることはコストが高いため，その時点でのキャッシュヒット率を利用したアクセス回送判定手法を提案する．

2.1 条　　件

ここでは，Primary-Backup の one-copy [10] 複製を仮定する．そして，すべてのリクエストはまず Primary データを保持するノードに到着し，回送判定方法に基づきリクエストを回送するか否かを決定する．

各ノードは，Chained Declustering 複製配置戦略に基づき，それぞれがプライマリとなるデータと，他の隣接するノードの複製データの両方を格納している．各データはアクセスが最終的に到達したノードでアクセスされ，当該ノードのキャッシュ

表 1　ノード・キャッシュサイズ緒元

pattern	#page / node	#page / cache / node
web パターン	15,000,000	32,000

メモリに挿入される対象となる．ここでは，複製間の一貫性については常に取れていると仮定し，回送対象とするアクセスはデータ読出しアクセスとする．

2.2　アクセス頻度別回送決定のディスクキャッシュ性能への影響

キャッシュヒット率は，キャッシュの量と，そのキャッシュが対象とするデータ量に関連する．回送を行うと回送先においてはキャッシュ対象データ量が増えキャッシュヒット率が減少すると考えられる．一方，アクセス回送対象データ量は対象データのアクセス頻度に依存する．なぜならば回送率はノード間負荷量が均衡化するように，回送データとは独立に設定されるためである．

そこで，アクセス頻度順位上位のデータ，中位のデータ，下位のデータに対するアクセスをそれぞれ回送する手法についてキャッシュヒット率への影響を実験により観測した．

2.2.1　頻度別回送によるキャッシュヒット率の測定実験

一般的な追い出しアルゴリズムである LRU を利用したキャッシュと頻度優先追い出しアルゴリズムである 2Q [11] を利用したキャッシュをシミュレートするプログラムに対して，アクセス系列を入力とし，キャッシュヒット率を算出した．アクセス系列としては，我々の研究室の WEB サーバ(注1)のログから採取したパターンを用意した．アクセス数は年間 2 万件程度で，アクセス傾向は四半期毎程度に変化しており，一部のよく使われるファイルへのものである zipf 分布に近い．

実験の仮定として，ノードは回送元ノードと回送先ノードの 2 つのみを考え，回送元ノードには他のノードからリクエストが回送されることはなく，回送先ノードには複製のアクセスとプライマリへのアクセスの両方が到達することとした..（図 1）．複製配置でチェーンを構成する，Chained Declustering 複製配置において，格納されるデータの間に強い関連がなければ，アクセス回送によりアクセス負荷均衡化を図ると，アクセス回送されてこない（負荷の高い）ノードから始まる少数個のノードのチェーンの集合となると考えられる．今回は，その一番単純な形として 2 ノードのモデルを考えた．

データ総数，キャッシュ容量は表 1 の設定とし，回送率と，2 ノード間アクセス量比率を変化させ計測した．

その結果を図 2 に示す．左の 6 種類は回送元ノードと回送先ノードの負荷が均衡している状態で回送をしたものである．upper が上位，middle が中位，lower が下位回送方式をあらわす．いずれのパターンにおいても，下位回送方式が最もヒット率が高く，中位，上位の順で悪い．これは，下位回送方式では，全リクエスト中の頻度の高いデータへのリクエストが 2 ノード間に分散するのに比べ，上位回送方式では，高頻度アクセスデータがいずれも回送先ノードのキャッシュを利用してしまう

(注1)：http://www.de.cs.titech.ac.jp/

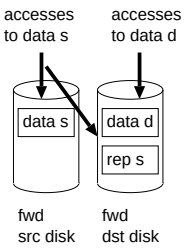


図 1　実験環境: 2 ノード構成の chained declustering

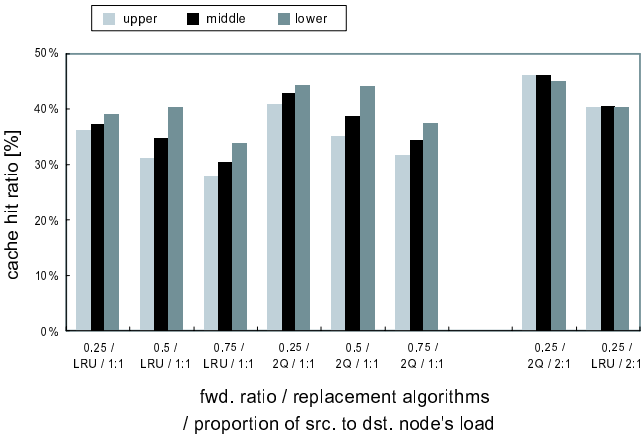


図 2　回送率・ページ追い出しアルゴリズムを変えたときの web パターンアクセスに対するキャッシュヒット率

ためであると考えられる．少量のデータが多量のアクセスを受けるアクセス傾向モデルは一般的であり，より高頻度アクセスされるデータをより長くキャッシュに置く方が，高頻度アクセスと低頻度アクセスをそれぞれのキャッシュ領域で住み分けるよりも効率がよい．

右の 2 種類は，回送元ノードと回送先ノードの負荷の比が 2:1 のときに，25%のアクセスを回送した結果を示している．この場合，上位回送方式が最も高いヒット率を示している．これは，全てのリクエストのうち高頻度のものが回送元ノードに格納されているプライマリデータ宛てであり，これらを 2 つのノードのキャッシュに分配することが，高頻度アクセスデータの分散と等価となったからである．しかし，下位回送方式は負荷に差がある場合でもあまり性能低下していない．

WEB パターン実験の一つに対して，回送元ノードと回送先ノードのキャッシュヒット率を図 3 に示す．図 3 からも，下位回送方式が効率よくキャッシュを扱えていることがわかる．上位回送方式では，回送先ノードのヒット率上昇に比べ，階層元ノードのヒット率減少が著しい．一方，下位回送方式では，回送先ノードのキャッシュヒット率減少量に比べ，回送元ノードのキャッシュヒット率が大きく上昇しており，高頻度アクセスファイルが常にキャッシュ上に配置されている．

2.3　キャッシュヒットを利用したアクセス回送決定

前節のように，キャッシュヒット率減少を抑えるための回送としては，頻度下位のものの回送が有効であった．しかし，現実のシステムにおいて，完全な頻度順位情報を得ることは非常にコストが高い．特に，アクセス回送率設定は，高頻度で実行

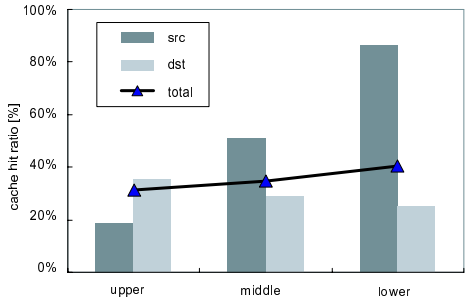


図 3　web パターン・LRU・50%回送時の回送元，回送先各ノードにおけるキャッシュヒット率

することにより現在の負荷傾向に合致した回送率が得られるため，よりコストが少なくキャッシュヒット率減少を抑える回送先決定手法が必要である．

そこで我々は，キャッシュのヒット率情報を利用したアクセス回送方法を提案する．これをキャッシュh/m(Hit or Miss) 分岐方式と呼ぶ．基本概念は下位回送と同様，低頻度利用データへのアクセスを回送対象とする．よって，キャッシュミスしたリクエストを優先的に回送する．キャッシュミスしたリクエストを全て回送して足りない場合はキャッシュヒットしたリクエストを確率的に回送する．

観測している最近のヒット率を h, 設定された回送率を r とすると，キャッシュh/m 分岐方式は以下のように判定を行う．

（1） あるリクエストがプライマリ側にきたら，まずプライマリ側のディスクのキャッシュをサーチする．

（2） プライマリデータ格納ノードのキャッシュにおけるヒット/ミスを見る．ここで，ヒットあるいはミスしたとしても，キャッシュ内容への変化は施さない

（3） 設定された回送率に従い，以下の式により決定する．

（a） ミス率 $(1-h)$ が回送率 r を上回るとき: ミスしたアクセスのうち $r/(1-h)$ を回送

（b） ミス率が回送率を下回るとき: ミスしたアクセスは全て回送する．ヒットしたアクセスのうち $(r+h-1)/h$ も回送

以上により，r によって指定された率のアクセスを回送することができる．ここで問題となるのが，h として用いている回送元ノードのキャッシュヒット率は，h/m 分岐方式に影響されることである．回送元ノードでは提案方式によりヒット・ミスが判定されたアクセスのうち，ヒットのものがより多く実際のキャッシュにアクセスする．そのため，回送判定の度にヒット率が上昇する．特に回送率が高い場合，キャッシュミスしたアクセスが全て回送され，キャッシュ内データの追い出しが起こらなくなり, 実際のキャッシュヒット率 h は 1 に収束してしまう．この場合，上記手法では目的の回送率を達成できなくなる．

そこで，各キャッシュ内に正規のヒットリクエスト数カウンタの他に，上記 (1) によるキャッシュ走査もアクセスとしてカウントする模擬キャッシュヒット率変数 h_{im} を用意しこれを h の代わりに用いることで，回送によるキャッシュヒット率変化時でも回送率を維持できる．

図 4 に WEB パターンを入力として，h と h_{im} をそれぞれ

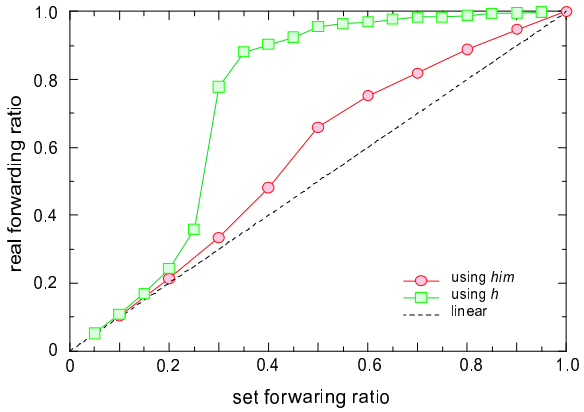


図 4　現実のキャッシュヒット率 h と模擬キャッシュヒット率 h_{im} を用いた場合の，設定された回送率に対する実際の回送率

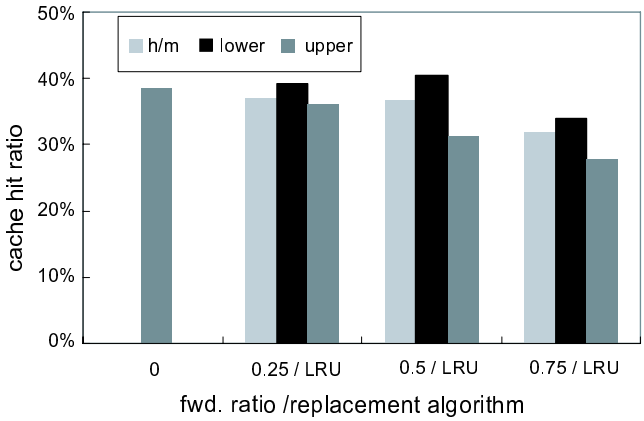


図 5　キャッシュh/m 判定方式を用いて回送を行った場合のキャッシュヒット率

用いた場合の，設定回送率に対する実際に得られた回送結果を示す．h を用いた場合，本来のキャッシュヒット率である約 30%の周辺で，得られる回送結果が入力を大きく外れてしまっている．一方，h_{im} を用いた場合，設定回送率が 0.5 の付近で，30%程度期待する結果から外れてしまっているが，h を用いる場合に比較し概ね合致している．

本手法についても 2.2.1 と同様にシミュレータ上で実験を行った．LRU+Web パターンによる結果を図 5 に示す．これより，本手法が下位回送方式と上位回送方式の中間程度の性能を示せることを確認した．

4. による実験では，本方式を自律ディスクに実装し，つづく 3. における制御手法の提案の評価に用いる．

3.　複数の技法の切り替え制御

データレプリケーションと，アクセス回送率の動的制御による負荷分散はコストが低いといった特徴をもつが，配置された複製の位置や数により能力が制限される．一方データマイグレーションは，より大きな変化に対応できる. より広範囲のアクセス傾向変化に対応するには，データの再配置が必須であるが，データの再配置には，ディスクアクセスとネットワーク転送が伴い，ストレージシステムがサービスに利用する能力を消費してしまう．

そこで，本稿では，自律ディスクにおいて，レプリケーションとマイグレーションを併用し負荷の変動へ対処するアプローチの下，これらの協調制御手法として RM-Combination を提案する．

RM-Combination は 2 種の手法の切り替え制御と，併用制御の 2 機能からなる．

切り替え制御　現在の格納データに対する負荷偏りをレプリケーションのみで解決不可能であると判断した時点でマイグレーションを発動する．

併用制御　マイグレーションによるデータ移動経路の選択と，マイグレーションによる各ノードの負荷上昇を考慮したレプリケーション戦略立案を行う．

以下では，まず自律ディスクの本提案に関係のある特徴を述べ，つづいてマイグレーションに起因する負荷見積もり方を紹介し，RM-Combination について述べる．

3.1 前提とする環境

3.1.1 自律ディスクとレプリケーション戦略

自律ディスクにおけるアクセス回送は次の手順で行われる．全てのリクエストは自律ディスクを構成する各ノードに均等に到着する．各ノードは，分散ディレクトリを走査し，リクエストが該当するデータ（のプライマリデータ）を保持しているノードにリクエストを転送する．プライマリ保持ノードは指定された回送率を保持しており，2.3 で述べたような判定手法を用いて，アクセスの回送の是非を決定し，必要があればリクエストを複製保持ノードへ転送する．回送率は常に全てのリクエストを見て，全てのノードへのリクエストが均衡化するよう，以下のように決定されることを前提とする．今回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．

（1） 全ノードは，一つのノードに負荷情報を送信する．

（2） 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回送率を計算する．

（3） 計算し終わった回送率を各ノードに送信する．

（4） 各ノードは与えられた回送率をセットする．

(2) では，まず現在負荷が平均を超えているノードを探し，その超過負荷を複製側へ回送する．回送されたノードでも負荷が平均を超えればその負荷をさらにそのノードの複製を保持するノードへ回送する．以上の作業を全てのノードに対して行う．最初のノードを探す作業が $O(n)$ である．その負荷を回送する作業はまた最悪の場合は $O(n)$ であるが，n が大きい場合，実際には $O(1)$ で計算できる．よってこの作業の平均の計算量は $O(n)$ である．Pentium4 2.4GHz を搭載した PC における 10000 ノード想定時の計算時間は高々10ms であった．これは十分現実的である．一方，(1)，(3) については，全てのノードがひとつのノードに直接負荷情報を送る場合スケーラビリティに欠けるが，ここでは我々が提案する TCSH [12] のように，ノード間に木構造のパスを形成し，段階的に収集する方法を仮定した．これにより負荷情報収集・発行時間は $O(log(n))$ であることが期待でき，十分現実的な範囲で上記アルゴリズムを完了することが可能であると考えられる．

3.1.2 データ移動による負荷の見積もり

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]. ここではその概略を述べる．

データマイグレーションによるマイグレーションデータ量を Q_{mig}[Byte] とする．マイグレーションによる負荷の平均を L_{mig} [Byte/s], マイグレーションにかかる時間 t_{mig}，及び Q_{mig} の関係は次式となる．

$$L_{mig} = \frac{Q_{mig}}{t_{mig}} = \frac{Q_{avg}}{t_1 \times s} \tag{1}$$

ただし，データ 1 つ辺りの平均転送時間を t_1 [s] とデータ一つあたりの平均サイズ Q_{avg} [Byte] とした．分母の 2 は読み出しと書き込みが逐次的に行われる場合のためである．読出しと書き込みが同時に行われる場合必要ない

今，システムに対する外部からのアクセスリクエストに対する性能要件として定義されたレスポンスタイムを R_{req} [s] とおく．データマイグレーションによるアクセスは，各ノードにおいては外部からのアクセスリクエストと等価に扱われるため，データ一つあたりの転送時間 $t_1 < R_{req}$ となる．よって，マイグレーションによる負荷は

$$L_{mig} = \frac{Q_{avg}}{2R_{req}} \tag{2}$$

により求められる．

実際のマイグレーション負荷が過小評価であり L_{mig} よりも大きい場合，誤った見積もりによる負荷均衡化戦略により当該ノードの負荷は計画よりも上昇する．その結果，ノード性能が落ち, マイグレーション速度は減少する．以上を繰り返すことにより，マイグレーション負荷は L_{mig} に収束すると考えられる．

3.2 切り替え制御

レプリケーションとマイグレーションの切り替え制御を行う．現在の格納データに対する負荷偏りをレプリケーションのみで解決不可能であると判断した時点でマイグレーションを発動し，マイグレーションによる負荷均衡化の一纏まりの処理が終了するまで両手法を併用する．本節では，各ノードが許容最大性能を超えないための 3 つの閾値の併用について述べる．

Chained Declutering + one-copy 複製配置では，あるデータに対する負荷は各ノード許容性能の最大 2 倍まで許容される．2 倍以内であれば許容性能を超える負荷を複製側に回送することで，回送先ノードにも十分なサービス余裕があればシステムは正常に動作可能である．あるノードに全ノードの負荷平均 2 倍以上のアクセスが集中すると，レプリケーションによる負荷均衡化により対処できなくなる．

そこで，この全ノード負荷平均の 2 倍を閾値 1(相対閾値) として用いる．$LP_i(t)$ を時刻 t におけるノード i に格納されたプライマリデータへの負荷，$LB_i(t)$ を複製データへの負荷とする．ノード i の複製はノード $i+1$ へ配置されており，またノード i はノード $i-1$ の複製を保持しているとする．ノード $i+1$ へ回送を行わなかった場合のノード i の負荷は $LP_i(t) + LB_{i+1}(t) + LB_i$ である．今，全 n ノードにか

かる負荷の平均 $L_{avg}(t) \triangleq \frac{1}{n}(\sum_i LP_i(t) + LB_i(t))$ に対し，その 2 倍を超える負荷のノードが存在するとき，すなわち $LP_i(t) + LB_{i+1}(t) + LB_i(t) > 2 \times L_{avg}(t)$ の時，時刻 t+1 において不均衡が発生すると考えられる．よってデータマイグレーションを行い，大域的な負荷を均衡化する．

負荷の偏りが 2 倍以内であったとしても，平均負荷がノード性能と比較して十分高い場合，破綻する可能性がある．そこで閾値 2(絶対閾値) を導入する．ノード i の許容最大負荷量を Lm_i とする．絶対閾値は，そのノードへの負荷が許容性能の 2 倍から，急激な変化に対応するためのマージンを除いた値とする．即ち，$max_i(LP_i(t) + LB_{i+1}(t) + LB_i(t)) > (Lm_i - L_{mig}) \times 2$ となる t において，マイグレーションを実行する．ここで L_{mig}(単方向マイグレーションのための負荷値) をマージンとして用いた．

また，負荷の偏りが大きいが，全体負荷量が低い場合，データを移動する必要はない．ここでは閾値 3(下限閾値) として，$max_i(LP_i(t) + LB_{i+1}(t) + LB_i) < (Lm_i - L_{mig})$ を置いた．

以上のように，$max_i(LP_i(t) + LB_{i+1}(t) + LB_i) > (Lm_i - L_{mig}) \times 2 \bigvee LP_i(t) + LB_{i+1}(t) + LB_i(t) > 2 \times L_{avg}(t)$ である閾値によりマイグレーションを発動する．

3.3 併用制御

切り替え制御によりマイグレーションを行うことが決定したら，改めて全てのノードから負荷情報を取得し [12] データ移動戦略を決定する．ここで，我々はマイグレーション時にレプリケーションを行い負荷を分散する Replica-assisted Migration を提案しているが，その手法を用いる．

3.3.1 データ移動経路決定, 回送率算出

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．

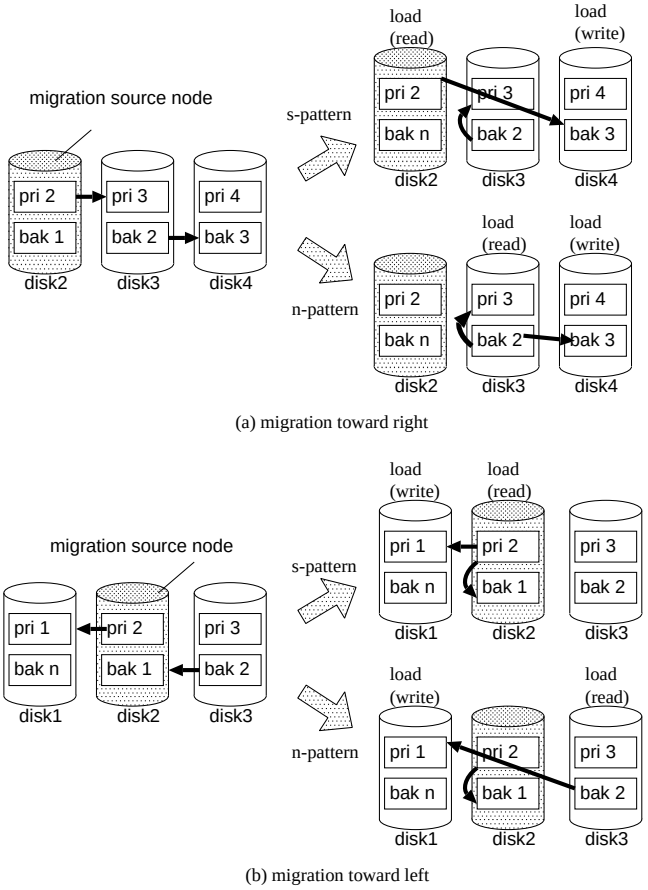
よって，右方向と左方向マイグレーションの 2 種類があり，各方向のマイグレーションには，2 通りのデータ経路が選択できる（図 6）．ここでは，マイグレーション元ノードに負荷がかかる経路を S(Self)-pattern，マイグレーション元ノードの両サイドに負荷がかかる経路を N(Neighbor)-pattern と呼ぶ．図 6(a) では左図のように pri 中のデータを pri3 に，耐故障性を考慮して bak2 中の複製を bak3 へ移動する．この場合，disk3 中では bak2 → pri3 を，ポインタ張替えにより行うことができる．よって，pri2 → bak3(S-pattaern) もしくは bak2 → bak3(N-pattern) の何れかにより，目的のデータ移動を行うことができ，マイグレーションに起因する負荷は disk4 への write オペレーションと，disk2 または disk3 への read オペレーションの 2 つである．

データ移動経路決定アルゴリズムは以下に記す．

（1） 経路パターンに拠らない負荷を，各ノードの負荷リストに加える．

（a） ノード i が右方向マイグレーションを行うなら，ノード $i+2$ へ L_{mig} を加える．

（b） ノード i が左方向マイグレーションを行うなら，ノード $i-1$ へ L_{mig} を加える．



(a) migration toward right

(b) migration toward left

図 6 マイグレーション実行時各方向 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.

（2） ノード 1 から順にマイグレーションを行うかを見る．方向に拠らず，ノード i とノード $i+1$ の負荷値を比べ少ない方へ L_{mig} を加える．ノード i に加えたなら経路パターンを S-pattern に設定する．ノード $i+1$ に加えたなら経路パターンを N-pattern に設定する．

（3） 以上を全てのマイグレーション元ノードにおいて行う．

以上により，各マイグレーションによる負荷は可能な限り，より余裕のあるノードへとかかるようにデータ経路が設定される．

この後，現在の負荷情報と，マイグレーションにより増加する見積もり負荷情報を加えたリストに対し，通常レプリケーションと同様に全ての負荷が均衡化するように回送率を決定する．以上の結果を全てのノードに対し配信し，終了を待つ．

4. 実　　験

提案する RM-Combination の性能保障面での有効性を示すために，自律ディスク模擬実装上に提案手法を実装し実験を行う．自律ディスクに提案手法であるキャッシュh/m 法および RM-combination を実装し，クライアントからのアクセス負荷偏り・高負荷環境下における各ノードの挙動を観察した．

表 2　ストレージノード性能緒元

#nodes	4 台 (Storage)
CPU	AMD Athlon XP-M 1800+ (1.53GHz)
MEM	PC2100 DDR SDRAM 1GB
Network	1000BASE-T
HDD	TOSHIBA MK3019GAX
	(30GB，5400rpm，2.5inch)
OS	Linux 2.4.20
Local File System	ext3 FS
Java VM	Sun J2SE 1.5.0_01

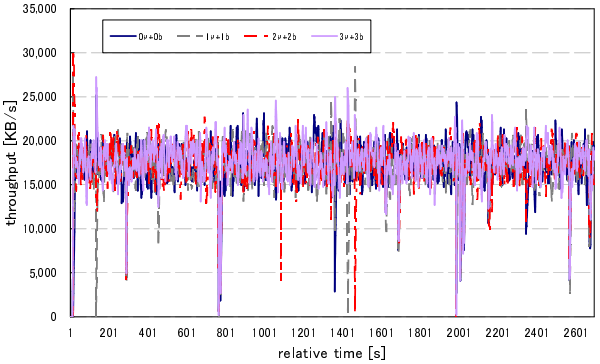


図 7　各ノードごとに格納されているデータへの合計スループット

4.1　実 験 環 境

実験は，自律ディスクの模擬実装上で行う．これは Linux クラスタ上に Java を用いて模擬実装されている．今回の実験では表 2 に示す構成の PC と十分なバックボーン性能を持つネットワークスイッチを用いて，実験環境を構成した．ユーザに対して透過的なデータ配置を実現する分散ディレクトリには，aB$^+$-Tree [13] を利用したため，データ配置の変更毎に更新情報のブロードキャストが発生する．また，クライアントと自律ディスクのインターフェースに HTTP を用いる実装 [14] を利用した．

このような環境の自律ディスクに対し，1MB のデータを合計 3000 個挿入した，それらに対して Pentium4 2.8cGHz のクライアントから 6 個のスレッドを用いて，値域のあるデータ付近に正規分布上の偏りがおこるようアクセスを発生させた．また，正規分布の山の部分が 1 時間に 1 回同じデータにあたる速度で変化させた．

4.2　実験結果と考察

負荷をかけはじめてからおよそ 40 分間，各ノードのワークロードと回送率の変化を観測した．

図 7 に各ノードに格納されているデータへの合計スループットを示す．4 本の線それぞれが各ノードの時間ごとのスループットを示している．以下に示すグラフは全て，横軸は開始してからの相対時間であり，縦軸は 2 秒毎に集計したスループットである．どの曲線も概ね 17MB/s 前後を推移している．

また，図 8 に時間ごとのシステム全体の合計スループットから算出した 1 台あたりの平均スループットを示す．各ノードごとのスループットに比べより分散が小さい．これより，自律ディスクに実装した回送率設定手法がうまく動作しレプリケーションによって．

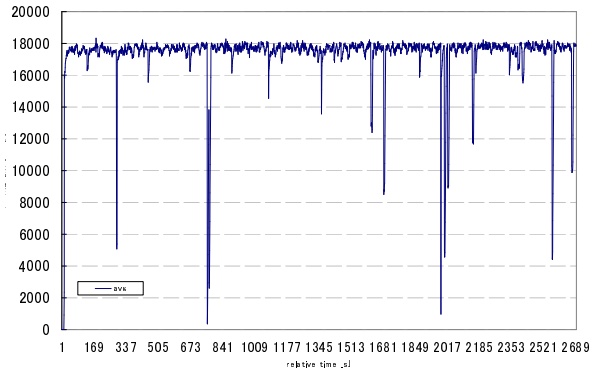


図 8　システム全体の平均スループット: 4 台のノードに格納されている全てのデータへのアクセスの平均

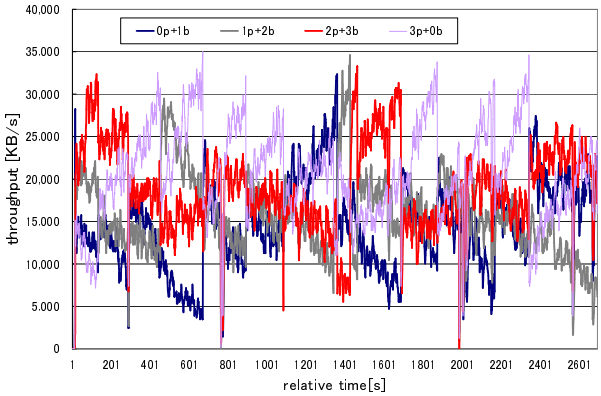


図 9　各ノードごとにおける格納されているプライマリデータへの回送を考慮しなかった場合の合計スループット

図 9 に各ノードごとの，他のノードに回送したリクエストを含めた格納されているプライマリデータへの合計スループットを示す．実際のデータへのアクセス偏りを確認できる．各折れ線グラフが急激に変化しているところはデータマイグレーションが実行されている時点である．システムスループット (図 8) と比較すると，図 9 のマイグレーション実行時点で図 8 が減少している傾向は余り見られない．これにより，我々の提案する RM-Combination がマイグレーションによる負荷を適切に吸収し分散できていることがわかる．

図 8 において急激にスループットが落ちている点が散見される．例として t=792 の時点で特に顕著に下がっている．図 9 と比較するとその時点ではマイグレーションが動作している傾向（各曲線の急激な傾向の変化）はない．また全てのノードのスループットが同期的に急激に落ち，直後に回復しているため，これらはクライアント側においてガーベジコレクタ等で一時的に処理が停止したためであると考える．

5.　関 連 研 究

高機能ストレージにおいて，耐故障性と負荷分散のためにデータの複製を用いているシステムとして D-SPTF [9] や GoogleFS [3] がある．D-SPTF はキャッシュとディスクシーク

タイムを考慮した，ブロードキャストベースのアクセス回送手法を提案している．GoogleFS では，ノード故障や負荷の変更に従い，複製を動的に増やす機構を備えている．

複製間のアクセス分配率を変更することにより，動的負荷分散を行う技術に DASD dancing [4] や Chained Declustering [8] があげられる．これらは複製配置チェーンを通じて負荷を移動伝播させることでデータの移動を行うことなく負荷を他のストレージに移動している．

データマイグレーションに起因するアクセス負荷からレイテンシ増加を抑える手法として Aqueduct [15] がある．Aqueduct では移動データを細分化し，各タイミングごとに，その直前のレイテンシからフィードバックして次のマイグレーションスピードを求めることで，QoS を達成している．しかし，マイグレーション実行時間が 5 倍以上遅くなることもあり，高頻度の負荷分散には適さない．

6. まとめと今後の課題

本稿ではストレージノード間の負荷均衡化と要求性能保障の両立を目的とし，我々の提案する自律ディスクに対しレプリケーションによる負荷分散機能を導入した．

効率のよいレプリケーションを実現するために，アクセス回送判定手法について考察を行った．また，アクセス頻度順位による回送判定を計算機シミュレーション上で評価した結果を元に，効率のよい回送判定方法であるキャッシュh/m 判定方式を提案した．

また，自律ディスクが従来から備えるデータ移動による負荷均衡化手法と，レプリケーションの切り替え・併用制御手法として RM-combination を提案した．RM-Combination では，データマイグレーションの発動回数を抑えると同時に，データマイグレーション実行時の負荷をレプリケーションの戦略立案に含めることによる要求性能保証を行う．

自律ディスクにより構成されたシステムに対して，提案手法を実装し実験した結果，われわれの手法が大小のアクセス負荷偏りを解消できることを確認した．

今後，レプリケーションのみの場合，マイグレーションのみの場合といった環境との詳細評価や，今回設定した各パラメータの妥当性評価等が必要であると考えている．

今後の課題としては，書き込みアクセスを考慮した複製間一貫性保持との協調動作や，3 つ以上の複製を保持するシステムにおける挙動等が考えられる．その他に，負荷が急に変動したときの制御方法，アクセス履歴を用いた予測的閾値変更，ヘテロ環境への適応といったことが課題として挙げられる．

謝　　辞

本研究の一部は，科学技術振興機構戦略的創造研究推進事業 CREST，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 (16016232) 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] G. Weikum, A. Mönkeberg, C. Hasse and P. Zabback: “Self-tuning database technology and information services: from wishful thinking to viable engineering.”, VLDB, pp. 20–31 (2002).

[2] G. R. Ganger, J. D. Strunk and A. J. Klosterman: “Self-* storage: Brick-based storage with automated administration.”, Technical Report CMU-CS-03-178, Carnegie Mellon University (2003).

[3] S. Ghemawat, H. Gobioff and S.-T. Leung: “The Google file system.”, SOSP, Boton Landing, NY, pp. 29–43 (2003).

[4] J. L. Wolf, P. S. Yu and H. Shachnai: “Disk load balancing for video-on-demand systems”, Multimedia Systems, **5**, 6, pp. 358–370 (1997).

[5] H. Yokota: “Autonomous Disks for Advanced Database Applications”, Proc. of International Symposium on Database Applications in Non-Traditional Environments (DANTE’99), pp. 441–448 (1999).

[6] 小林, 渡邊, 山口, 田口, 上原, 横田：“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, **3**, 2, pp. 65–68 (2004).

[7] 小林, 渡邊, 田口, 上原, 横田：“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，電子情報通信学会，DE2004-112 電子情報通信学会, pp. 35–40 (2004).

[8] H.-I. Hsiao and D. J. DeWitt: “Chained declustering: A new availability strategy for multiprocessor database machines”, Proceedings of the Sixth International Conference on Data Engineering, February 5-9, 1990, Los Angeles, California, USA, IEEE Computer Society, pp. 456–465 (1990).

[9] C. R. Lumb, R. Golding and G. R. Ganger: “DSPTF: Decentralized request distribution in brickbased storage systems”, Proceedings of ASPLOS’04, Boston, MA (2004).

[10] P. A. Bernstein and N. Goodman: “An algorithm for concurrency control and recovery in replicated distributed databases”, ACM Trans. Database Syst., **9**, 4, pp. 596–615 (1984).

[11] T. Johnson and D. Shasha: “2Q: A low overhead high performance buffer management replacement algorithm”, VLDB’94, Proceedings of 20th International Conference on Very Large Data Bases, Santiago de Chile, Chile, pp. 439–450 (1994).

[12] 渡邊, 横田：“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, **85-D1**, 9, pp. 877–886 (2002).

[13] M. L. Lee, M. Kitsuregawa, B.-C. Ooi, K.-L. Tan and A. Mondal: “Towards Self-Tuning Data Plaement in Parallel Database Systems”, SIGMOD Record, **29**, 2, pp. 225–236 (2000).

[14] 花井, 横田：“自律ディスクを用いた Web サーバーの構成”, 第 13 回データ工学ワークショップ論文集, DEWS2002 C3–4 電子情報通信学会データ工学研究専門委員会 (2002).

[15] C. Lu, G. A. Alvarez and J. Wilkes: “*Aqueduct:* online data migration with performance guarantees”, Conference on File and Storage Technologies (FAST’02), Monterey, CA, pp. 219–230 (2002).