社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－DBS－134 (II) (70)
2004／7／14

分散データ格納環境のための自律的並列偏り除去手法

渡邊 明嗣 * 山口 宗慶 * 小林 大 * 田口 亮 ‡ 林 直人 ‡ 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, † 東京工業大学 学術国際情報センター
〒 152-8552 東京都目黒区大岡山 2-12-1
{aki,muu,daik}@de.cs.titech.ac.jp, yokota@cs.titech.ac.jp
‡NHK 放送技術研究所 〒 157-8510 東京都世田谷区砧 1-10-11
taguchi.r-cs@nhk.or.jp, hayashi.n-gm@nhk.or.jp, uehara.t-jy@nhk.or.jp

概要

本研究ではアクセス偏りを短時間で解消する目的に適したデータ移動量算出式を提案する．提案手法はデータ分散機構が並列にデータ移動を行う際に生じる過渡的偏り増大の抑制に有効である．スケーラブルな分散データ格納システムを構築するためには，アクセスの均等分配を行うデータ分散機構が必要である．アクセスパターンがしばしば変化する環境ではそれによって生じるアクセス分布の偏りを短時間で解消することが重要である．データ移動によって偏り除去を行う場合，１回のデータ移動量によってその効果が異なってくる．提案手法は隣接ＰＥ間の負荷比率を取り入れることで，観測誤差の影響の軽減と大きな偏りの速やかな解消を目指し，過渡的偏り増大を抑制する．

キーワード： 分散ディスク, 負荷分散, 分散配置, 分散制御, 偏り制御

Autonomous Transit Skew Handling for Parallel Data Storage Systems

Akitsugu WATANABE* Munenori YAMAGUCHI* Dai KOBYASHI*
Ryo TAGUCHI‡ Naoto HAYASHI‡ Toshihiro UEHARA‡ Haruo YOKOTA†,*

* Department of Computer Science Graduate School of Information Science and Engineering
Tokyo Institute of Technology
† Global Scientific Information & Computing Center Tokyo Institute of Technology
2-12-1 Oh-Okayama, Meguro-ku Tokyo, 152-8552 JAPAN
aki@de.cs.titech.ac.jp, yokota@cs.titech.ac.jp
‡ NHK Science & Technical Research Laboratories 1-10-11 Kinuta, Setagaya-ku Tokyo, 157-8510 JAPAN
taguchi.r-cs@nhk.or.jp, hayashi.n-gm@nhk.or.jp, uehara.t-jy@nhk.or.jp

Abstract

We propose a data migration amount computation formula for parallel skew handling method targeting fast skew removal. This suppresses the transient skew increasement during parallel access skew handling. For scalable parallel data storing systems, it is important that the system has an online data declustering mechanism which balances skewed access requests. Access skews generated by changing access pattern should be resolved in short time. The effect of skew handling is affected by the amount of data being migrated by one execution. The goals of our proposal are fast removal of large skews and decreasing insufficient migration caused by error in load evaluation.

KEYWORD: Parallel Disk, Load Balancing, Parallel Data Placement, Skew Handling

1 はじめに

分散データ格納システムのスケーラビリティ向上は我々は分散データ格納システムのスケーラビリティ向上を実現するアクセス分布偏り除去の分散制御技術TCSHを提案している[7]．TCSHはアクセス分布偏り除去プロセスのスケーラビリティを高めることによって，構成規模が大きい場合でも，また，アクセスパターンに変化がある場合でも，性能低下につながるアクセス分布の偏りを速やかに取り除く．また，TCSHは値域分割を用いた分散データ格納システムを適用対象としているため，Fat-Btree[11]やaB$^+$-TreeのようなB-treeベースの索引機構を利用できる．これらの索引機構を用いたシステムでは部分木単位の移動に基づく効率的なデータ移動手続きを利用できるため，偏り除去を一層効果的に行うことができる[4]．しかしながら，TCSHはアクセス分布偏り除去プロセスにおけるデータ移動フェーズを全PEで独立に並列実行するため，データ移動フェーズ中に過渡的な偏りが生じる可能性がある[5]．また，データ移動が通常処理の性能に与える悪影響を限定したり，アクセス分布の観測誤差による不必要なデータ移動を抑制したりするために偏り除去の1サイクル中に生じるデータ移動量を間引く手法が提案されているが[2]，移動量の間引きが効果的に働くためには，システム構成要素だけでなく，解消すべき偏りが持つ性質なども考慮しなければならないため，簡潔で効果的な間引き手法が必要とされている[9]．

我々は，このデータ移動量を間引く戦術の核となるデータ移動量計算式に注目した．我々がデータ移動量計算式に注目した理由は2点ある．第1に，アクセス分布の偏り除去においてデータ移動量の決定手法に注目した研究が少なく，改善の余地が残されていると期待できること．第2に，負荷評価手法，分散ディレクトリ構造の改良を含むデータ移動手段の改善といった既存の研究成果との統合が容易に行えるためである．偏り除去に高いスケーラビリティを持たせるためには，データ移動量計算もまたスケーラブルでなければならない．そこで，我々はシステムの構成規模に依存しない計算量で算出できる隣接ＰＥ間の負荷比率に注目した．本研究では，データ移動フェーズを全PEで独立に並列実行する偏り除去機構のための隣接ＰＥ間の負荷比率を利用したデータ移動量計算式を提案し，その効果を考察する．

本論文の構成は以下の通りである．2節では，本論文が前提とするアクセス分布偏り除去について述べる．3節では，隣接ＰＥ間の負荷比率に注目したデータ移動量計算式について述べる．4節では，提案手法の過渡的偏り除去に対する有効性を実験により検証する．最後に，5節で本論文の結論を述べる．

2 アクセス分布偏り除去

2.1 分散ディレクトリ構造

本論文ではデータの水平分割戦略に値域分割戦略が用いられていることを前提とする．これは，値域分割を用いた分割を行うことで，レンジクエリ，完全一致クエリを効率的に扱うことができ，さらにクラスタI/Oの利用や，システムを構成するPEの追加・除去に伴う分割数変更時における効率的なデータ再構成を行うことができるためである．

B-treeを基底とした階層型インデクスは，値域分割戦略を実現する効率的な手法の一つである．典型的な階層型インデクス方式では，PEに割り付けられた値域とキーとの対応を決定するための大域インデクスと，PE内でのキーに対応するデータの所在を決定するための局所インデクスの2つのインデクスを用いてデータの管理を行う．Leeらが提案しているaB$^+$-Tree[4]は，階層型インデクス方式と同様に値域分割戦略を効率的に取り扱うことを目的としており，局所インデクスに高さに制限を設けたB$^+$-tree構造を用いることでデータ数に対する探索コストのスケーラビリティと，分割境界変更のコスト軽減を実現している．また，我々が提案しているFat-Btree[11]は，階層型インデクス方式と同様に値域分割戦略を効率的に取り扱うことを目的としているが，インデクス木の根からの距離に応じてノードの冗長性が減少する構成方法を用いることによって，インデクス探索の負荷を効率的に分散し，データ挿入や削除に伴うインデクス更新における同期コストを減らし，さ

らに，分割境界変更のコストをも軽減する特長がある．いずれのインデクス方式も，データ数に対する探索コストが $O(\log(\text{総データ数}))$ で，インデクス更新時間を含めたデータの挿入・削除に要する時間が $O(\text{移動するデータ数} \times \log(\text{総データ数}))$ である．

以後，データ数に対する探索コストが $O(\log(\text{総データ数}))$ で，かつ，インデクス再構成に要する時間を含む，隣接 PE 間でのデータ移動に要する時間が $O(\text{移動するデータ数} \times \log(\text{総データ数}))$ になるようなデータ移動手段の存在を許す分散ディレクトリ構造が利用されているものとして議論を行う．

2.2 偏り除去の分散制御

アクセスパターンが変化する環境では，アクセス分布の偏りを速やかに取り除くことが性能向上において重要である．我々はシステム構成規模が大きい場合でも偏り除去を短時間で行うことができる偏り除去の分散制御手法 TCSH(Tree-Communication-Skew-Handling) を提案している [7]．TCSH は二分木状のコミュニケーション経路を構築し，値域分割を用いた分散データ格納システムにおけるデータ移動の制限を考慮した移動計画立案を行い，データを全 PE で並列に移動することによって，$O(\log(PE\text{ 数}))$-時間で偏り除去を行う．

TCSH は偏り除去を以下の 5 つのフェーズに分割して実行する（図）．

1. PE 負荷の計測
2. 負荷分布の伝達
3. 移動計画の素案の立案
4. 分割境界移動についての実行時判断
5. 実際のデータオブジェクト移動

TCSH は 3 フェーズで全体の負荷偏りを完全に解消するようなデータ移動計画を立案し，各 PE にこの計画に沿うような両隣接 PE へのデータ移動量の組を与える．例えば，4 つの PE から構成されるシステムにおいて，各 PE の負荷が順に

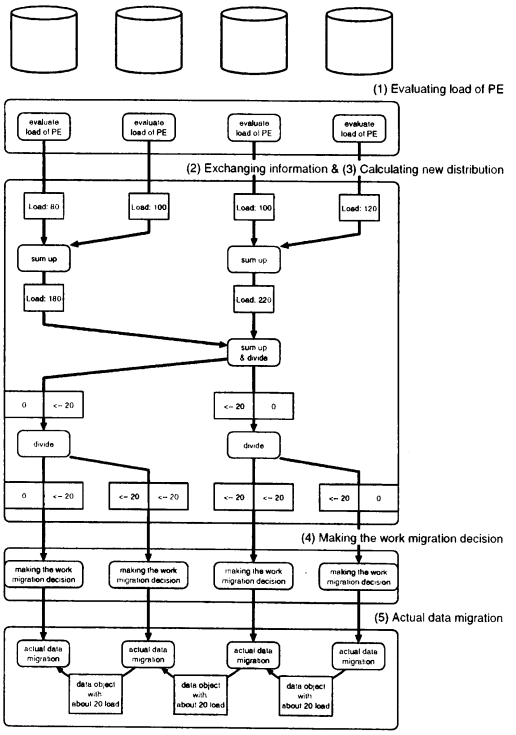

図 1: TCSH

PE0=50,PE1=125,PE2=125,PE3=100 となっていたならば，PE1 には PE0 へ 50 負荷に相当するデータを移動するように指示し，PE2 には PE1 へ 25 負荷に相当するデータを移動するように指示する．

2.3 データ移動量の決定

前述の 3 フェーズでは全体の負荷偏りを完全に解消するようなデータ移動計画の素案を立案する．しかしながら，立案された素案通りにデータ移動を行うと，いくつかの問題が生じる．

まず，(1) データの読みだし，書き込みといったシステムの基本動作の待ち時間が増加する問題がある．これは，全体の負荷偏りを完全に解消するようなデータ移動計画で，しばしば要求される大量のデータ移動が原因となって起こる．大量のデータ移動がディスクやネットワークアダプタを長時間利用するため，

	PE0	PE1	PE2	PE3	PE4
heat	150	100	150	50	50
heat to migrate	→50	→50	→100	→50	
after subdivision	→50	→50	→100	→50	
$heat_0$	150	100	150	50	50
$\Downarrow migrate_0$	→50				
$heat_1$	100	150	150	50	50
$\Downarrow migrate_1$	→50				
$heat_2$	100	100	200	50	50
$\Downarrow migrate_2$	→100				
$heat_3$	100	100	100	150	50
$\Downarrow migrate_3$	→50				
$heat_4$	100	100	100	100	100

図 2: 過渡的偏り増大の問題，並列にデータを移動することによって偏りが一時的に増大する

その間，基本動作の待ち時間が増加する．

次に，(2) 不必要な移動が生まれる問題がある．これは，前述の 1 フェーズで得られる負荷情報に誤差が含まれていることと，アクセスパターンが変化する状況ではデータ移動中のアクセスパターン変化によって偏りが自然に解消される可能性があることに因る．

最後に，(3) 過渡的偏り増大の問題がある．これは，個々のデータサイズが大きい場合により顕著に現われる．図 2 は過渡的偏り増大が生じるような状況を示したものである．行 "heat" は負荷に偏りがある初期の負荷分布状態，行 "heat to migrate" はその偏りを完全に解消するような移動計画である．$heat_0$ から $heat_4$ はデータ移動によって負荷偏りを解消していく様子を表している．偏り除去の 1 サイクルが完了した後の状態である $heat_4$ では負荷の偏りが完全に解消されているが，$heat_2$ では PE2 に初期の負荷を大きく上回る負荷が集中しており，全体の偏りは初期状態よりもむしろ増大している．負荷の偏りが非常に大きく，かつ，ホットスポットが連続しているような場合にこの問題は生じる．

Feelifl らは (1) 待ち時間の問題を解消するために，speed factor と呼ばれる定数を使ってデータ移動を分割する手法を提案した [2]．この方法では，全体の負荷偏りを完全に解消するようなデータ移動計画の素案における各 PE 間のデータ移動量に speed factor と呼ばれる定数を掛けることで，データ移動を分割する．また，移動に閾値を定め，speed factor を掛けた後の移動量が一定量を下回っていた場合，移動を取りやめる．例として，データ移動計画の素案において PE1 には PE0 へ 50 負荷に相当するデータを移動し，PE2 には PE1 へ 25 負荷に相当するデータを移動するように指示されていた場合を考える．speed factor が 0.5，閾値が 10 であれば，PE1 には PE0 へ 25 負荷に相当するデータを移動するように指示し，PE2 には PE1 へ 12.5 負荷に相当するデータを移動するように指示する．speed factor が 0.5，閾値が 20 であれば，PE1 には PE0 へ 25 負荷に相当するデータを移動するように指示し，PE2 にはデータ移動を行わないように指示する．この手法では移動後も負荷の偏りが残るが，残された偏りは偏り除去を繰り返すことによって解消される．この手法は 1 回の移動量を減らすため，(1) 待ち時間の問題に対して有効である．また，(2) 不必要な移動 に対しては，speed factor に小さな値を設定することで移動するデータの量を減らし，誤差やアクセスパターン変化による不必要な移動を減らすことができるが，speed factor に小さな値を設定すると，偏り除去が完了するまでの時間が大きくなるため，大きな偏りが生じる環境での性能低下が顕著となる．そのため，偏り除去をシステム再構成に用いるときやアクセスパターンが変化するような環境における (4) 大小の偏りが共存する環境での最適パラメタ設定 の問題が生じる [8]．(3) 過渡的偏り増大 に対しても，speed factor に小さな値を設定することでその影響を減らすことができるが，不必要な移動の場合と同様に，偏り除去が完了するまでの時間が大きくなるため，大きな偏りが生じる環境での性能低下の問題がある．

我々は [8] で，(4) 大小の偏りが共存する環境での最適パラメタ設定 の問題に注目し，各 PE 間のデータ移動量に各 PE 間のデータ移動量に比例するよう定められた係数を掛けることで，この問題を解決しようとした．この手法は偏り除去をシステム再構成に用いるときやアクセスパターンが変化するような環境のような大小の偏りが共存する環境に対して効果を示したが，(3) 過渡的偏り増大 の問題については未解決のままだった．

3 隣接間負荷比率を利用したデータ移動量計算式

データ移動量の決定は移動計画の立案や負荷評価と独立に行われるプロセスであるため，偏り除去の他の構成要素との相互依存が少なく，既存の研究成果との統合が比較的容易に行える．2.3 節で示した問題を解決することができれば，大小の偏りが共存したり，アクセスパターンが変化するような環境での偏り除去をより効率的に行うことができる．

しかしながら，これらの問題を解決するためにデータ移動量の決定を複雑にすることは好ましくない．なぜならば，データ移動量の決定プロセスが複雑になると負荷評価が行われてから実際のデータ移動が行われるまでの時間が大きくなり，その間のアクセスパターンの変化によって評価が実際の負荷から遠ざかってしまったり，偏りによる性能低下が長く続いたりするためである．また，データ移動量の決定がシステムの構成規模に対してスケーラブルであることも重要である．そこで，我々はシステムの構成規模に依存しない計算量で算出できる隣接ＰＥ間の負荷比率に注目した．図 3 は我々が提案している隣接ＰＥ間の負荷比率に注目したデータ移動量計算手法 Adaptive Migration Grain Optimization(AMiGO) である．AMiGO を使って求められた係数は全体の負荷偏りを完全に解消するようなデータ移動計画の素案における各 PE 間のデータ移動量に掛ける係数として利用される．

データ移動元の PE がデータ移動先の PE よりも頻繁にアクセスされている状況においては，AMiGO で求められる係数が大きな値になり，全体の負荷偏りを完全に解消するように求められたデータ移動計画の素案に近い量のデータ移動が行われる．素案に近い量のデータ移動が行われることによって，負荷が大きいデータ移動元の PE は速やかに高負荷状態から解放され，アクセスが負荷が小さいデータ移動先の PE に適切に配分されるようになる．また，データ移動元の PE がデータ移動先の PE よりも小さな負荷を持っている場合には，AMiGO で求められる係数が小さな値になり，データ移動の量が小さく抑えられる．

	PE0	PE1	PE2	PE3	PE4
heat	150	100	150	50	50
heat to migrate	→50	→50	→100	→50	
result of AMiGO	→30	→20	→75	→25	
$heat_0$	150	100	150	50	50
$\Downarrow migrate_0$	→30				
$heat_1$	120	130	150	50	50
$\Downarrow migrate_1$	→20				
$heat_2$	120	110	170	50	50
$\Downarrow migrate_2$	→75				
$heat_3$	120	110	95	125	50
$\Downarrow migrate_3$	→25				
$heat_4$	120	110	95	100	75

図 4: 係数=1.0 とした際の AMiGO による偏り除去の例

	PE0	PE1	PE2	PE3	PE4
heat	150	100	150	50	50
heat to migrate	→50	→50	→100	→50	
after subdivision	→25	→25	→50	→25	
$heat_0$	150	100	150	50	50
$\Downarrow migrate_0$	→25				
$heat_1$	125	125	150	50	50
$\Downarrow migrate_1$	→25				
$heat_2$	125	100	175	50	50
$\Downarrow migrate_2$	→50				
$heat_3$	125	100	125	100	50
$\Downarrow migrate_3$	→25				
$heat_4$	125	100	125	75	75

図 5: speed factor=0.5 を用いた偏り除去の例

これによって，既に高負荷状態になっている移動先の PE にさらなる負荷が与えられることを抑制する．

3.1 過渡的偏り増大の抑制

図 4 は係数 factor を 1.0 とした場合の AMiGO による偏り除去の様子を示したものである．この例では，図 2 と同様の，過渡的偏り増大が生じる移動順序でデータ移動が行われる．すなわち，ホットスポットである左から 2 番目の PE を移動先とするデータ移動が先に行われ，ホットスポットに過渡的な負荷の集中が生じるようになっている．

図 5 は speed factor を 0.5 とした場合の移動である．図 5 も図 4 も同じ標準偏差が 50 であるような PE 負荷の初期分布とそれに対する同じ移動計画の素案を元にしているが，移動途中の偏りと移動終了後に残る偏りが異なる．speed factor を 0.5 とした場合の移動では，途中で負荷の標準偏差が 53.0 まで増大し，

```
AMiGO(load(Source), load(Destination), threshold, factor) begin
    SpeedFactor = factor * load(Source) / (load(Source) + load(Destination));
    If (SpeedFactor < threshold)
        return 0;
    Else
        return SpeedFactor;
end.
```

図 3: Adaptive Migration Grain Optimization (AMiGO)

	PE0	PE1	PE2	PE3	PE4
heat	150	100	150	50	50
heat to migrate	→50		→50	→100	→50
after subdivision	→10		→10	→20	→10
$heat_0$	150	100	150	50	50
$\Downarrow migrate_0$	→10				
$heat_1$	140	110	150	50	50
$\Downarrow migrate_1$	→10				
$heat_2$	140	100	160	50	50
$\Downarrow migrate_2$	→20				
$heat_3$	140	100	140	70	50
$\Downarrow migrate_3$	→10				
$heat_4$	140	100	140	60	60

図 6: speed factor=0.2 を用いた偏り除去の例

移動終了後の負荷の標準偏差も 25.0 である．一方，AMiGO では，途中の負荷の標準偏差は 51.0 までしか増大せず，移動終了後の負荷の標準偏差も 17.0 と大幅に小さい．

移動を分割しない場合（図 2），移動終了後の負荷の偏りは無くなるが，途中での負荷の標準偏差が 61.2 と著しく大きな値になる．また，speed factor を 0.2 と小さく設定すると（図 6），途中での負荷の標準偏差は 50.5 までしか増大しないが，移動終了後の負荷の標準偏差は 40.0 と高いままである．

4 実験

我々は提案手法の有効性を確かめるために，我々が提案している自律ディスクを用いたマルチメディアコンテンツサーバーを用いた提案手法と speed factor を用いた手法の比較実験を行った [6]．自律ディスクを用いたマルチメディアコンテンツサーバーは動画等のマルチメディアコンテンツを HTTP を介して提供するための分散サーバであり，自律ディスクが持つ基本的な機能に加えて，メタデータを用いたデータ管理機能を備えている．分散制御手法には TCSH，負荷評価アルゴリズムには DTC[7] を利用した．実験の諸元を表 1 に示す．

表 1: 実験の諸元

Parameter	Values
System Parameters:	
HDD access ratio	22MB/sec
CPU	Intel Pentinum (R)III 933MHz
# of PEs in the cluster	6
message setup time	200μs/message
network	1000Base-SX
Java^{TM} Runtime	Sun JRE 1.4.1 Server VM
Database Parameters:	
index node size	4KB
contents size	≅50MB
number of contents	100
Query Settings:	
number of queries	4200
mean arrival rate	1sec
distribution	zipf(1)
hotspot	PE 0
Workload Migration:	
workload evaluation	DTC[7]
workload migration control	TCSH[7]
interval of skew handling	60sec

我々は以下に示す手順で実験を行った．まず，データが全ての PE に均等に分配されるように初期の Fat-Btree を構成し，各 PE に 50MB の動画コンテンツが 16–17 個格納されるようにした．次に，要求頻度が減衰係数 δ の Zipf 分布 [3] に従うような要求系列を生成した．この生成された要求を用いて，データ移動量計算式だけを変えた同一環境下でのスループット測定実験を行った．

図 7 は speed factor を用いる手法で speed factor を 0.5 に設定した場合のスループットの推移である．300

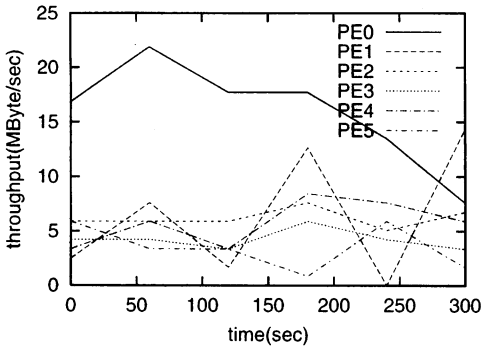

図 7: speed factor を用いた偏り除去過程, 係数=0.5

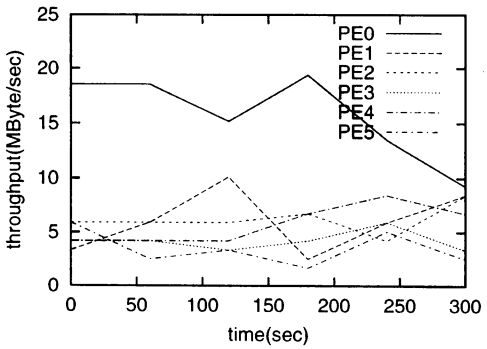

図 8: AMiGO を用いた偏り除去過程, 係数=1.0

秒の時点で，PE1 が PE0 を上回っている．すなわち，過渡的偏り増大が生じている．また，PE1 に対するアクセス量が時間とともに激しく変化している．

図 8 は提案手法で係数を 1.0 に設定した場合のスループットの推移である．過渡的偏り増大は一切見られず，PE1–5 のスループットが段階的に分散されている．

いずれの手法もホットスポットとなっている PE へのアクセスを他の PE に分配するが，その過程は大きく異なる．speed factor を用いる手法では移動の過程で別のホットスポットができるのに対し，AMiGO では全ての PE に対するアクセスが速やかに均等配分された状態に近づいていく．図 9 は偏り除去の過程における，スループットの標準偏差を示したものである．AMiGO は偏り除去が行われている期間の大

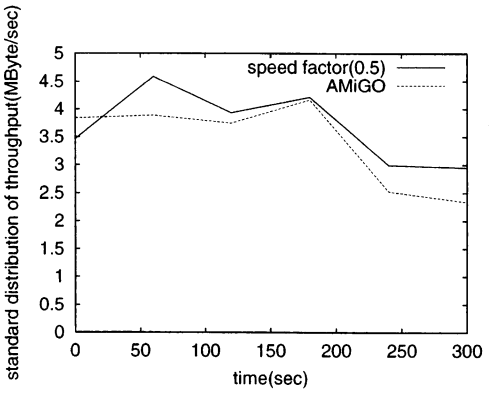

図 9: 偏り除去過程におけるスループットの標準偏差推移

半において，speed factor を使った手法よりもスループットの偏りが小さい．これは，AMiGO による過渡的偏り増大の抑制が期待通りに行われている事を示している．また，AMiGO は speed factor を使った手法よりも，短時間で偏りを解消している．

5 結論

本研究で提案した隣接ＰＥ間の負荷比率を利用したデータ移動量計算式は，偏り除去機構がデータ移動フェーズを全 PE で独立に並列実行した場合に生じる過渡的偏り増大を抑制する．過渡的偏り増大が抑制されることによって，偏り除去処理を行っている間のシステム性能低下を抑えることができる．

また，提案手法は隣接ＰＥ間の負荷比率のみを利用しているため，規模を拡大した場合でも実行コストが増大しないという特長を持っている．これはアクセスパターンが変化する場合に特に重要な短時間で偏りを除去するという目的にも適している．

我々は提案手法を自律ディスクを用いたマルチメディアコンテンツサーバ上に実装し，その効果を確認した．従来手法では過渡的な偏り増大が生じ，また，偏りが解消されるまでの時間も長い．提案手法は過渡的な偏り増大を抑制し，短時間で偏りを解消する．

今後は，大小の偏りが共存する環境での適切な移

動量の算出が為されているかどうかを検証する．また，より規模が大きいシステムに実装し，提案手法のスケーラビリティを確認する．

謝辞

本研究の一部は，文部科学省科学研究費補助金基盤研究（16016232），情報ストレージ研究推進機構（SRC），日本学術振興会 21 世紀 COE プログラム，NHK 放送技術研究所，および，科学技術振興機構戦略的創造研究推進事業（CREST）の助成により行なわれた。

参考文献

[1] G. Copeland, W. Alexander, E. Boughter, and T. Keller. Data Placement in Bubba. In *Proc. of ACM SIGMOD Conf. '88*, pp. 99–108, 1988.

[2] Hisham Feelifl, Masaru Kitsuregawa, and Beng-Chin 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*, Sep 2000.

[3] Donald E. Knuth. *Sorting and Searching*. Addison-Wesley Publishing Company, 1973.

[4] Mong Li Lee, Masaru Kitsuregawa, Beng-Chin Ooi, Kian-Lee Tan, and Anirban Mondal. Towards self-tuning data plaement in parallel database systems. *SIGMOD Record*, Vol. 29, No. 2, pp. 225–236, Sep. 2000.

[5] 渡邊明嗣, 花井知広, 山口宗慶, 田口亮, 林直人, 上原年博, 横田治夫. 過渡状況を考慮した 分散データ格納環境のための並列偏り除去手法. In *DEWS 2003*, March 2003. 1-B-3, http://www.ieice.org/iss/de/DEWS/proc/2003/program.html.

[6] 渡邊明嗣, 花井知広, 山口宗慶, 横田治夫. 自律ディスクを用いたマルチメディアコンテンツサーバ. Technical Report DE2002-86, DC2002-22, 電子情報通信学会, October 2002.

[7] 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, Vol. J85-D-I, No. 9, pp. 877–886, September 2002.

[8] 渡邊明嗣, 横田治夫. 分散ディレクトリ偏り制御とシステム再構成を統合する再配置制御. *DBSJ Letters*, Vol. 1, No. 1, pp. 3–6, October 2002.

[9] 渡邊明嗣, 横田治夫. 並列データアクセス偏り制御におけるスケーラブルな並列制御. In *DEWS 2002*, March 2002. C1-3, http://www.ieice.org/iss/de/DEWS/proc/2002/index.html.

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