データベースシステム 128－1
（２００２．７．１７）

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

渡邊 明嗣 * 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@de.cs.titech.ac.jp, yokota@cs.titech.ac.jp

概 要

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，クラスタ構成の再構築手続きと負荷偏り除去が重要である．これらの手続きはいずれもデータの再配置を伴うため，データ移動要求の衝突を回避する機構が必要である．クラスタ再構成中の状態を負荷が偏った状況と捉えることで，これらの移動手続きを負荷偏り除去手続きの中に統合できる．再構成に伴う負荷の偏りと要求される移動量は定常状態のそれに比べて大きく，短時間での処理が困難である．本稿では，クラスタ再構成に伴う大きな偏りと定常状態の小さな偏りの両方を効率的に除去できる偏り除去の並列制御機構と分割数決定機構を提案し，その性能を評価する．

キーワード： 分散ディスク, 負荷分散, 分散配置, 分散制御, システム再構成, 偏り制御

A Reorganization Control Method to Unify Skew Handling and System Reconstruction with Distributed Directory

Akitsugu WATANABE* 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

Abstract

Cluster reorganization and skew handling procedure take important place to constructuring scalable distributed data storage systems. Because both procedures involve data reorganization, we should adjust migration requirements of these procedures. By dealing state under cluster reconstruction with highly load skewed situation, data migration by cluster reconstruction is unified to skew handling procedure. However, since skews generated by cluster reconstruction are much higher than usual skews, it is hard to remove it quickly. In this paper, we develop our parallel control method for skew handling and propose a high skew adaptive splitting method for migration. The experiment indicates that the proposed method can treat both ordinary skews and highly skews generated by cluster reconstruction.

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

1 はじめに

共有のメモリ装置やディスクを持たない処理ユニット (PE) 群で構成される並列無共有システムは，対費用効果，耐故障性，および，拡張性が優れていることから，大規模データの格納に適した構成として注目されている．並列無共有システムでは，独立したディスクおよびメモリを持つ多数の PE が共有のネットワークに接続されており，各 PE は唯一の共有資源であるこのネットワークを利用してメッセージ交換を行う．

高い拡張性を持つデータ格納システムを並列無共有システム上に構成するためには，規模の拡大に伴う性能向上を線形に近づける PE 間の負荷分散と規模の変更時にサービスの質を落とさない効率的な PE クラスタの再構成とが共に重要である．負荷分散が不十分であれば，負荷が集中した PE(ホットスポット) が発生し，システムの性能が低下する．分散データ格納システムにおける負荷の偏りはデータ配置と深い関連があり，そのため，データの水平分割によって負荷均衡を行う戦略が広く研究されてきた [6, 4, 8, 3]．

値域分割戦略はレンジクエリや近傍探索の取り扱いが効率的で，また，クラスタ I/O を用いた効率的なアクセス手法を提供できるなど，ハッシュ分割やラウンドロビンには無い優れた特長を有している．しかしながら，アクセス傾向が本質的に流動的であることから，また，運用中のデータの追加及び削除が予測不能であることから，初期のデータ分割をいかように行ったとしてもいずれは偏りが生じる．このような運用中に生じた偏りを取り除くためには，データの動的再配置による負荷分散機構が必要である．データの動的再配置ではデータ配置情報を更新する処理の性能が重要となるため，我々が提案している並列インデクス構造 Fat-Btree[13] ではデータ配置情報自体を分散格納することによって，更新処理およびデータアクセスの性能と拡張性を高めており，また，その性質を利用した負荷分散機構が提案されている [13, 5, 11]．

取り扱うデータ量の増大に伴う PE の追加や故障に伴う PE の取り外しなどの必要があることから，分散データ格納システムの拡張性の向上においては PE クラスタの再構成もまた重要な要素となる．我々はストレージ主導型の分散データ格納アーキティクチャとして自律ディスク [12] を提案している．自律ディスクの特長はストレージ PE の自律分散管理を行うことであり，これによって，複雑なデータ配置サーバが持つボトルネックと耐故障性の問題から分散データ格納システムを解放している．我々は自律ディスクにおける PE クラスタのオンライン再構築手法を提案しており [7]，サービス停止を必要としない PE クラスタの再構築が可能である．

負荷分散と PE クラスタの再構築はいずれもデータの再配置を伴うため，データの移動を協調して行うための機構が重要である．我々は自律ディスクの再構成に伴うデータの再配置を，負荷分散機構を介して行うことで，PE クラスタ再構築機構と負荷分散機構の移動要求における衝突を回避する手法を提案している [7]．この提案ではノード間の領域使用量を平均化する負荷分散機構 [13] の利用を前提としており，そのため，[2, 5] によって指摘されているオブジェクト間にアクセス偏りがあるような状況での負荷分散における問題を残している．一方で，従来のオブジェクト間にアクセス偏りがある状況を考慮した負荷分散機構 [5, 11] では PE クラスタ再構築を考慮していないため，ディスクの追加に伴う大量のデータ移動を速やかに解決できない問題がある．

本研究では PE クラスタ再構築を考慮した負荷分散機構についての考察を行い，PE の追加に伴う負荷の偏りを効率良く解消できる負荷分散機構の構成案を示す．また，提案構成案についてシミュレーションを用いた評価とその考察を行う．

2 章では自律ディスクの概要について述べる．3 章では分散ディレクトリ構造について述べる．4 章ではクラスタの再構築を考慮した負荷分散機構の構成案を示す．5 章ではシミュレーションを用いた提案構成案の評価とその考察を行う．6 章では本稿の結論を述べる．

2 自律ディスク

大規模データ処理の分野では，ストレージセント

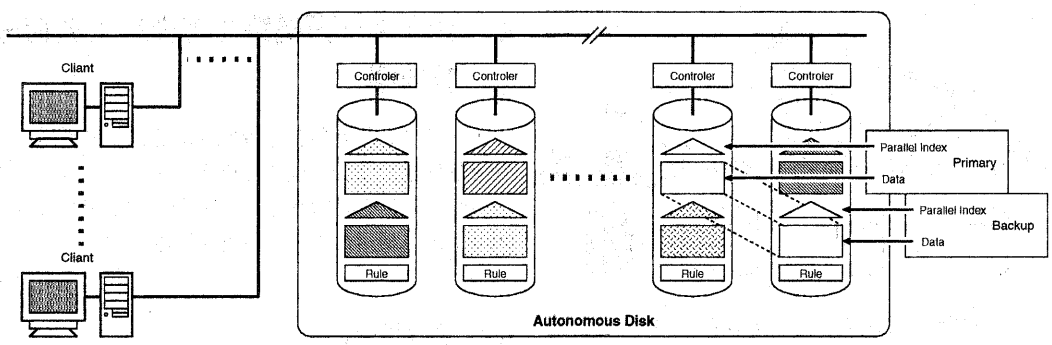


図 1: 自律ディスク

リック，すなわち，ストレージをシステムの中心に据える構成が注目されている．LAN に接続された共用ストレージである NAS(Network Attached Strage) や複数のストレージを専用のネットワークで接続したものである SAN(Storage Area Networks) はストレージセントリックな構成の 1 つであり，スケーラブルなストレージ構成の実現として商業的にも注目を集めている．

しかしながら，NAS や SAN ではストレージはパッシブデバイスとして扱われるため，外部にデータ管理を行うサーバーを設置する必要がある．このようなサーバーは潜在的なボトルネックであり，また機能の集中点があることは耐故障性の観点からも好ましくない．スケーラブルな大規模ストレージを構築するためには，ボトルネックおよびクリティカルポイントになるような機能の集中点を持たない構成，すなわち，ストレージ自身が自律分散管理を行うような構成が有効である．我々が提案している自律ディスク (Autonomous Disk)[12] はこのような自律分散管理を行う大規模ストレージのための構成手法であり，現在ディスクコントローラとしてディスクに搭載されている PU を利用することで，自律分散構成を安価に実現することを狙っている．

自律ディスクは以下のような機能的特徴を備えた，独立した PU，メモリとディスク装置を備えた PE 群をネットワークで連結した分散ストレージクラスタである．

透過的アクセス　自律ディスクは，クライアントに均一で透過的なアクセスを保証する，すなわち，同一の自律ディスククラスタに属するデータに対するアクセスを試みる際に，クライアントはその所在に注意を払う必要が無い．

自律負荷分散機構　自ら負荷状態を管理し，データ配置を動的に変更して負荷の偏りを除去する．偏り除去処理はオンラインで行われ，かつクライアントからは隠蔽される．

オンライン再構築機構　PE の追加／削除を自律管理する．再構築処理の開始は，クライアントまたはクラスタ自身によって宣言される．

障害回復　自動的に障害が発生した PE を隔離し，クラスタ内に置かれたバックアップからのデータ復元を試みて，残りの PE 群で本来の機能の維持を図る．

これらの機能は PE 上のディスクに記述されたルールと，ルールを実行するルールエンジンによって実現される．ルールエンジンの詳細については [12] を参照されたい．透過的アクセスは分散ディレクトリ構造 Fat-Btree[13] の利用によって可能になる．Fat-Btree の説明は 3 章で行う．自律負荷分散機構については 4 章で議論する．オンライン再構築機構はクラスタ構成情報の更新とデータ移動の 2 つの段階から成り，データ移動は自律負荷分散機構を用いて行われる [7]．クラスタ再構築を考慮した自律負荷分散機

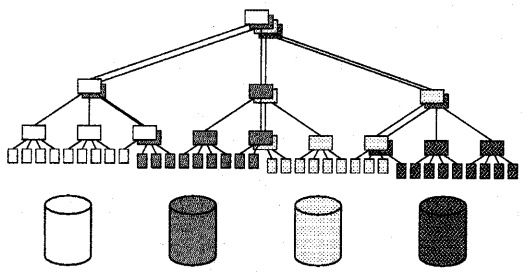


図 2: 並列ディレクトリ構造 Fat-Btree

構によるデータ移動については 4 で議論する．障害回復は本稿との関連が薄いため本稿では紹介しない．[7] を参照されたい．

自律ディスク構成の概念図を図 1 に示す．

3 分散ディレクトリ構造

レンジクエリの取り扱いに優れ，また，クラスタ I/O を用いた効率的なアクセスが可能となることから，本稿ではデータが値域分割戦略によって分割されていることを仮定する．

スケーラビリティーの観点から，分散ディレクトリ構造の基底構造には B-tree が適しており，いくつかの提案が行われている [10, 9, 13] ．我々が提案している Fat-Btree[13] は冗長なコピーの配置手法を工夫することによって (図 2)，探索および更新が効率的に行われる分散ディレクトリ構造である．Fat-Btree はインデクス構造に他の PE に格納されたデータへのポインタを含めることで，全てのデータに到達可能なメカニズムを可能にする．また，PE 間のデータ移動が容易であることと，システム性能の低下が少ない BULK 転送方式 [1] を扱えることから，データの再構成を用いる負荷分散に適している [5].

本稿では，データの移動を高速に行うことができ，また，インデクスの再構成が効率的な Fat-Btree を分散ディレクトリ構造として用いる．また，データ移動の最小単位がインデクスの葉ノードであることを仮定する．

4 クラスタ再構築を考慮した負荷分散機構

オンライン並列負荷分散機構は並列制御機構，負荷評価機構，移動速度決定機構の 3 つの要素から構成される．負荷評価機構は，PE 負荷評価機構と部分木負荷評価機構によって構成される．

4.1 並列制御

我々は実行時間が $O(\log(\#\text{ of PE}))$ であるスケーラブルな負荷分散の並列制御 TCSH(Tree Communication Skew Handling parallel control) を提案している [11]. しかしながらこの提案では，クラスタを構成する PE が均質であることと，クラスタ構成が変化しないことを仮定しているため，クラスタ再構築を扱うためには若干の変更が必要である．すなわち，各 PE の負荷情報だけではなく，各 PE の (均質でない) 処理能力をも集計する必要がある．

図 3 はクラスタ再構築を考慮した負荷分散機構の並列制御 TCSH の改良案である．TCSH は以下の 5 つの実行フェーズを経て，クラスタの負荷を平均化する．

1. 通信木の構築

   コーディネーターとなる PE が負荷分散を管理するコーディネータープロセスを作り，通信木を構築する．通信木の頂点はプロセスに，辺はプロセス間通信路に対応する．コーディネータープロセスを根とするバランス 2 分木で，各 PE と 1 対 1 対応したプロセスである葉ノードを持つ．

2. PE 負荷の評価およびその伝達

   葉ノードで PE 負荷評価を行い，結果と PE の処理能力を通信木の枝を上りながら合計する．

3. 負荷分布の伝達

   各枝は，子から受け取った負荷情報を元に子の処理能力に応じた負荷分担と子の間での負荷移動量を決定する．子は，親から負荷分担と負荷移動量を受取り，それを自らの子に分担させる．

```
Start_TCSH() {
    p はこのプロセスを表す;
    D は クラスタに属する PE の全体集合を表す;
    任意の PE にプロセス TCSH() を作り，D, p を送る;
    作成した TCSH() から負荷 h, 処理能力 a を受信する;
    作成した TCSH() に負荷割当 h, 0, 0 を送信する;
}

TCSH() {
    p はこのプロセスを表す;
    親プロセス pp から，PE 集合 S, pp を受信する;
    if(|S| >1) {
        S を左集合 L と右集合 R に等分する;
        L,R それぞれから任意の PE lpe, rpe を選ぶ;
        /* フェーズ 1 */
        lpe に TCSH() プロセス lp を作り L, p を送る;
        rpe に TCSH() プロセス rp を作り R, p を送る;
        /* フェーズ 2 */
        lp から 負荷 lh, 処理能力 la を受信する;
        rp から 負荷 rh, 処理能力 ra を受信する;
        pp に (lh + rh),(la + ra) を送る;
        /* フェーズ 3 */
        負荷割当 ah, 左/右へ移動する負荷 tlh,trh を
            pp から受信する;
        L への負荷割当 lah = ah × la/(la + ra));
        R への負荷割当 rah = ah × ra/(la + ra));
        L に 負荷割当 lah, 左/右に移動する負荷
            tlh, ((lh − tlh) − lah) を送信する;
        R に 負荷割当 rah, 左/右に移動する負荷
            ((rh − trh) − rah), trh を送信する;
    } else {
        /* フェーズ 2 */
        PE 負荷評価戦略 GLE を用いて PE p の負荷を測定し，
            ch に代入する;
        PE p の処理能力を測定し，ca に代入する;
        pp に ch,ca を送る;
        /* フェーズ 3 */
        負荷割当 ah, 左/右へ移動する負荷 tlh,trh を
            pp から受信する;
        /* フェーズ 4 */
        移動速度決定戦略 SF を用いて tlh と ca から
            今回移動する負荷 ctlh を決定する;
        trh,ctrh についても同様;
        /* フェーズ 5 */
        ctlh が正なら，部分木負荷評価戦略 LLE を用いて
            ctlh から移動する部分木の大きさを決定し，
            部分木を左側の隣接 PE に移動する;
        ctrh についても同様;
    }
}
```

図 3: クラスタ再構築を考慮した TCSH の改良案

4. 移動速度の決定

   過剰な移動を抑制するために，移動速度決定機構はフェーズ 3 で受け取った負荷移動量に移動速度係数を掛け，移動をより小さな単位に分割する．

5. 部分木負荷評価機構による移動する部分木の決定と移動

   フェーズ 4 で決定された負荷移動量を基に，部分木負荷評価機構が隣接 PE に移動する部分木を決定し，移動する．

フェーズ 1, 2, 3 は通信木に沿って実行されるため，その実行時間のオーダーは $O(\log(\#\text{ of PE}))$ である．フェーズ 4,5 は各 PE で独立に実行されるため、その実行時間のオーダーは $O(1)$ である．したがって，TCSH の実行時間は $O(\log(\#\text{ of PE}))$ である．

4.2 負荷評価機構

近年におけるネットワーク，ディスク，プロセッサ 3 者の速度向上の現状から，ディスクアクセスはボトルネック発生原因の最も有力な候補である．ディスクアクセスに注目したシステム負荷の評価は熱と呼ばれる指標に反映される [2]．我々は熱の概念を発展させ，キャッシュへのヒット率によって 1 アクセスあたりの重み付けを変える負荷評価機構 DTC(Directory Traverse Cost based load evaluation) を提案している [11]．DTC はインデクスノードの冗長なコピーによってもたらされる負荷分散効果と，また，キャッシュによる負荷軽減効果を考慮することで，システム負荷の評価精度を高めている．

DTC には記録するアクセス履歴の粒度によるバリエーション DTC_n が用意されている．DTC_n は，インデクス木の根からの距離が n 以下のノードについてのみアクセス履歴を記録し，その先では負荷が均等に分配されていることを仮定する．細かい粒度は負荷評価の精度を高め，正確な偏り除去を可能にするが，アクセス履歴の記録コストが大きい．粗い粒度はアクセス履歴の記録コストが小さいが，負荷評価の精度において劣る．我々のシミュレーションで

は，DTC_1が記録コストと除去精度の良いトレードオフとなっていた[11].

4.3 移動速度決定機構

負荷分散の過程では，一度移動したデータを元の場所に戻すような移動がしばしば起こる．このような“無駄な移動”は負荷評価の誤差や負荷分布の変化などによって起こるもので，システムに余分な負荷を与え，性能を低下させる．このような無駄な移動を抑制には，負荷の移動量に速度係数を掛けてデータの移動を分割するアプローチが有効である．

Feeliflらはデータ移動量の閾値ζと速度係数αを用いて無駄な移動を抑制するアプローチを提案している[5]．このアプローチでは，負荷の移動量のPEの負荷に対する比が閾値ζ以下であるような移動を抑制し，また，負荷の移動量に速度係数$\alpha(0<\alpha<1)$を掛けてデータ移動を複数回に分割する．次式は，このアプローチにおける移動予定の熱と移動量に掛けられる係数fkの関係を示したものである．

$$\mathrm{fk}_{\alpha,\zeta}(h,a)=\begin{cases}0 & (h/a<\zeta)\\ \alpha & (h/a\geq\zeta)\end{cases}$$

ただし，hは移動予定の熱，aは移動分割を行うPEの負荷．関数fkは移動予定の熱hを$h=h\times\mathrm{fk}(h,a)$のように正規化する．すなわち，$\mathrm{fk}=\alpha=0$の時は一切の移動を行わず，$\mathrm{fk}=\alpha=0.5$の場合には観測された偏りの半分に相当するページを移動し，$\mathrm{fk}=\alpha=1$の場合には観測された偏りの全部に相当するページを移動する．

このアプローチは定常状態における小さな偏りの除去に伴う無駄な移動を効果的に抑制する反面，非常に大きな偏りの除去が遅くなる欠点を持ち，クラスタ再構築に伴うPEの追加/削除によってもたらされる非常に大きな偏りを効率的に除去することが難しい．大きな偏りを速やかに除去しようとしてαを大きな値に設定すると定常状態における無駄な移動が増え，定常状態の無駄な移動の抑制を期待してαを小さな値に設定すると大きな偏りの除去に対応できない．

この問題は，システムに与える影響の度合いが異なる大きな偏りと小さな偏りを同じ速度係数で取り扱うことから生じている．速度係数を定数ではなく偏りの大きさに応じて変化する値とすることで，大きさの異なる偏りを効率よく扱うことができる．我々は負荷偏りによるシステムの性能低下率がシステムの過剰負荷率に対応していると予想し，次のような移動速度決定関数rsh(Reconstruction consious Skew Handling)を用意した．

$$\mathrm{rsh}_{\lambda,\chi,\zeta}(h,a)=\begin{cases}0 & (h/a<\zeta)\\ \lambda(h/a) & (\zeta\leq h/a\leq\chi/\lambda)\\ \chi & (h/a>\chi/\lambda)\end{cases}$$

ただし，hは移動予定の熱，aは移動分割を行うPEの負荷．λは過剰負荷率が与える影響の大きさ，χは移動速度の上限，ζは移動の閾値を表す．関数rshは移動予定の熱hを$h=h\times\mathrm{rsh}(h,a)$のように正規化する．この関数は，偏りが非常に小さい場合には移動を行わず，偏りの負荷に対する比が移動の閾値ζを上回るとその比に応じた割合でページの移動を行う．また，移動速度に上限χを設けることによって，移動予定の熱が大きい場合でも無駄な移動を抑制する．

この関数はクラスタの再構築を負荷分散機構で取り扱うという本稿の目的を満たすものである．我々は次章で，この関数のシミュレーションによる評価を行う．

5 シミュレーションによる評価

この実験では，4.3で提案した移動速度決定関数を用いた負荷偏り除去を行い，ディスク追加時の負荷偏り除去速度と，定常状態での“無駄な移動”量を測定する．比較のため，同時に[5]の手法を用いた負荷偏り除去も行い，結果を示す．

以下の実験では，表1に示したパラメータを用いる．偏り除去の並列制御には本稿で提案したTCSHの改良案を用いる．PE負荷評価戦略GLE(Global Load Evaluation)として，一定時間内のデータアクセスに要した時間の総計を用いる．部分木負荷評価戦略LLE(Local Load Evaluation)として，我々が提案しているDTC_1[11]を用いる．移動速度決定関数として，$\mathrm{rsh}_{2,1,1/32}$

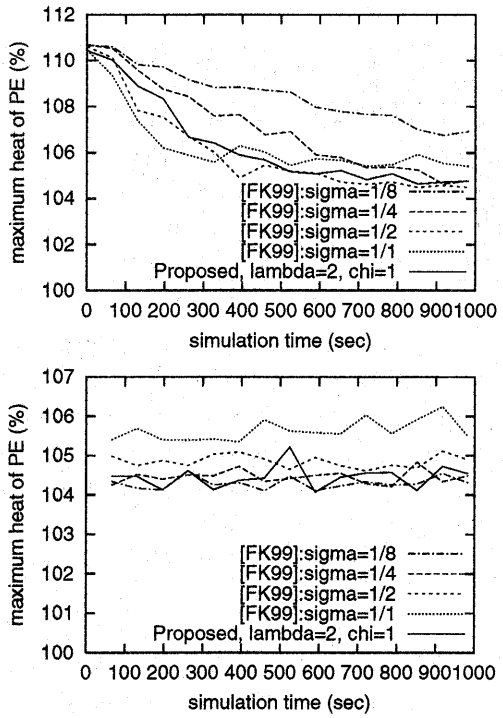
表 1: シミュレーションに用いたパラメータ

パラメータ	値
インデクスノードのサイズ	4KB
データページのサイズ	4KB
メッセージセットアップ時間	200μs/メッセージ
ディスクのアクセス速度	10ms/4KB
メモリのアクセス速度	1μs/4KB
PE 数	16
キャッシュ連想度	8
キャッシュセット数/PE	128
データページ数	1M ($\simeq 4GB$)
インデクスノード	
最大エントリ数	200
エントリ利用率	ln 2
インデクス木の高さ	4
アクセス偏り (zipf 分布の母数)	0.5
クエリの到着間隔	2.0 ms
偏り除去の間隔	32K クエリ ($\simeq$ 1 分)

を用いる．rsh の母数 $\lambda = 2$ は偏りの無いクラスタに PE の追加された場合の負荷偏りの比率 $\simeq 0.5$ における移動速度を最大にすることを狙って選択した．$\chi = 1$ は移動速度が 1 を越えないという [5] の制約を踏襲した選択である．$\zeta = 1/32$ は，偏りの無い PE 数 15 のクラスタに新しい PE を追加した場合の負荷偏り $1/15 \simeq 1/16$ を基に，それに 2 倍のマージンを持たせるよう選択した．

ホットスポットになる可能性がある PE への負荷の集中の度合いを示すため，最大負荷，すなわち最も負荷の大きい PE の負荷の平均に対する比を用いる．例えば，最大負荷 200%は，少なくとも 1 つの PE に負荷が完全に分散されている場合に比べて倍に相当する負荷が与えられていることを表す．大きな最大負荷は負荷が集中した PE の存在と，その PE がボトルネックになることによる性能低下の可能性を示唆する．

図 4 はディスク追加時および定常状態での最大負荷率の偏り除去に伴う推移を示したものである．提案手法は，ディスク追加時においては中程度の移動速度であり，ディスク追加に対する偏りの収束速度と収束結果に優れた $\mathrm{fk}_{1/2,1/32}$ に近い推移を示す．また，定常状態では，移動速度が大きく，無駄な移動が行われにくい $\mathrm{fk}_{1/8,1/32}$ とほぼ同等の安定した推移を示す．図 2 に定常状態における無駄な移動の量を示



Skew=0.5, Cashe Set=128/PE, ζ=1/32

図 4: ディスク追加時の負荷偏り除去過程 (上) と定常状態での負荷偏り除去過程 (下)

した．結果は $\mathrm{fk}_{\alpha,1/32}$ は α の値が大きい場合に多くの無駄な移動を行うとの予想に一致しており，$\alpha = 1/1$ の場合の無駄な移動は平均で 500 ページ程度でデータの移動時間にして約 5 秒に相当する．提案手法は無駄な移動が少なく，負荷分散機構がシステムに与える負荷が小さい．

6 結論

本稿では，負荷分散機構にクラスタ再構築におけるデータ移動手段としての利用があることを前提に，クラスタ再構築におけるデータ移動において有効な特長を有する負荷分散機構の提案を行った．我々は，従来の並列制御機構を拡張した並列制御機構を提案した．また，クラスタ再構築において有効な移動速

表 2: 定常状態における，"無駄な移動" が行われたページ数の平均

[5] $\alpha = 1/8, \zeta = 1/32$	35.77 ± 24.08
提案手法	$\mathbf{37.37 \pm 28.92}$
[5] $\alpha = 1/4, \zeta = 1/32$	86.45 ± 49.04
[5] $\alpha = 1/2, \zeta = 1/32$	209.31 ± 103.08
[5] $\alpha = 1/1, \zeta = 1/32$	526.29 ± 259.71

度決定機構の案 $\mathrm{rsh}_{\lambda,\chi,\zeta}$ を示した．我々は提案移動速度決定機構の性能をシミュレーションによって評価し，それが定常状態において定常状態のために調整された従来手法と同等の性能を持ち，また，ディスクの追加の際にはディスクの追加のために調整された従来手法に近い性能を持つことを示した．我々が提案した負荷分散機構はより単純な機能部品の組み合わせによって構成されており，システムの状況に合わせた調整が可能であることから，様々な構成の分散データ格納システムへの応用が期待できる．また，提案負荷分散機構の実行時間は $O(\log(\#\text{ of PE}))$ であり，クラスタの自律再構成との組合わせによって高いスケーラビリティを持つシステムの構築が可能である．ディスクの取り外しを含めた提案手法の評価及び適切な移動速度決定の手法は将来の課題である．

謝辞

本研究の一部は，文部科学省科学研究費補助金基礎研究 (12680333,13224036,14019035) および情報ストレージ研究推進機構 (SRC) の助成により行われた．

参考文献

[1] Kiran J. Achyutuni, Edward Omiecinski, and Shamkant B. Navathe. Two techniques for on-line index modification in shared nothing parallel databases. In *ACM SIGMOD*, pp. 125–136, Montreal, Canada, June 1996.

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

[3] R. Devine. Design and Implementation of DDH: Distributed Dynamic Hashing. In *Proc. of FODO'93*, 1993.

[4] David DeWitt and Jim Gray. Parallel Database Systems: The Future of High Performance Database Systems. *Communications of the ACM*, Vol. 35, No. 6, pp. 85–98, June 1992.

[5] 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.

[6] K. A. Hua and C. Lee. Handling Data Skew in Multiprocessor Database Computers Using Partition Tuning. In *Proc. of VLDB '91*, pp. 525–535, 1991.

[7] 伊藤 大輔, 横田 治夫. 自律ディスクにおけるクラスタ再構築アルゴリズムの実装. 信学技報, 電子情報通信学会, FTS2001-20. 電子情報通信学会, July 2001.

[8] W. Litwin, M.-A. Neimat, and D. A. Schneider. LH* – Linear Hashing for Distributed Files. In *Proc. of SIGMOD Conf. 93*, pp. 327–336, 1993.

[9] W. Litwin, M.-A. Neimat, and D. A. Schneider. RP* : A Family of Order-Preserving Scalable Distributed Data Structures. In *Proc. of VLDB'94*, pp. 342–353, 1994.

[10] B. Seeger and P. Larson. Multi-Disk B-trees. In *Proc. of ACM SIGMOD Conf. '91*, pp. 436–445, 1991.

[11] WATANABE, Akitsugu and YOKOTA, Haruo. A Directory-Traverse-Cost Based Skew Handling for Parallel Data Access. *IEICE Trans. Inf. & Syst. (Japanese Edition)*, 2002. in press.

[12] 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.

[13] 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.