C1-3

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

渡邊 明嗣* 横田 治夫†

概 要

分散格納されたデータへの並列アクセスにおいて，負荷の均衡化は性能およびスケーラビリティの向上のために極めて重要であり，これまでにもデータ移動による負荷分散制御手法が広く研究されている．大規模化する近年のデータベースにおいては，負荷分散制御手法自体に対してもスケーラビリティが要求されている．負荷分散制御手法のスケーラビリティを高めるためには，集中制御は適当ではなく，偏り除去制御の分散化が重要となる．しかし，これまでの分散偏り除去制御はトークンベースで，スケーラビリティが十分とは言えない．我々は通信木を構築することによって偏り除去制御の分散化を図る分散制御手法を提案している．本稿では，その手法を従来の分散制御手法と比較し，提案手法が従来の手法に比べて高いスケーラビリティを有することを示す．また，提案手法による効率化がディレクトリ構造，負荷分布の評価手法との組み合わせによって受ける影響について考察を行なう．

キーワード Fat-Btree, 分散ディスク, 負荷分散, 分散制御, 分散配置, 偏り制御

*東京工業大学 大学院情報理工学研究科 計算工学専攻 aki@de.cs.titech.ac.jp
†東京工業大学 学術国際情報センター yokota@cs.titech.ac.jp http://yokota-www.cs.titech.ac.jp/yokota/index-jp.html

1 はじめに

データベース用並列無共有システムにおいて，データオブジェクトの操作を伴う処理は，関連するデータオブジェクトが配置されている各 PE 上で並列に実行されることが望ましい．各 PE 間で均等に処理が実行されることは，アクセス性能およびシステムのスケーラビリティーに大きな影響を与える．そのため，並列データベースにおける PE の負荷を平均化する技術が数多く研究されてきた [5, 10, 9, 6, 2].

データ配置は，負荷分散に影響を与える要素として特に重要である [5]．値域分割戦略による水平分割はクラスタ I/O が利用でき，かつレンジクエリや近傍探索をハッシュやラウンドロビンによる配置に比べて効率よく行なえることが魅力である．しかしながらデータベースの運用に伴い，データオブジェクトの追加・削除およびアクセスパターンの変化を原因とする負荷偏りが生じるため，これを取り除くためのデータの動的再配置，すなわち動的偏り制御が必要である．

値域分割された並列データベース動的偏り制御として隣接する PE 間でのみ負荷を比較する分散制御 [8]，トークンベースで分散制御を行なう [11, 3] がある．隣接する PE 間でのみ負荷を比較する分散制御は，分散制御の収束保証が無く，信頼性を欠く．トークンベースでは分散制御の利点を十分に活かしていないため，計算時間が O(PE#) となり規模の拡大に伴い計算時間が膨大なものになる．

我々は通信木を構築することによって計算時間を短縮する分散制御アルゴリズム TCSH (Tree-Communication-Skew-Handling) を提案している [12]．TCSH は通信木を用い，また 1 回当りの通信コストを定数オーダに抑えることで計算時間に大きな影響を与える通信コストを小さくする．我々は，TCSH を用いることでプラニング段階の通信時間が $O(\log \text{PE\#})$ に短縮されることを示したが，負荷評価アルゴリズムを変えての詳細な解析は行なっていない．

本稿では，TCSH の実行時間をシミュレータを用いて測定し，TCSH による実行時間の短縮を評価する．また，TCSH を各種の負荷評価アルゴリズムと組み合わせた場合の実行時間をシミュレータを用いて検証し，並列データベース動的偏り制御における分散制御方式の与える影響を明らかにする．

2 TCSH

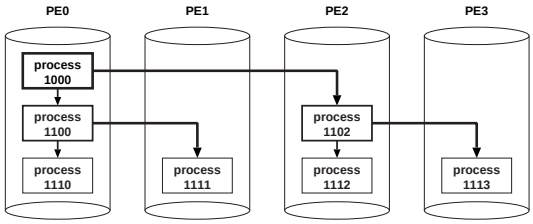
横田らは無共有並列システム用ディレクトリ構造 Fat-Btree と，その上での動的データ量偏り除去のためにアクセス状況を PE 間で回覧し，偏り検査とページ移動を行う detect_skew_&_invoke_migration アルゴリズムを提案した [11]．Feelifl らは [4, 3] において Fat-Btree を並列ディレクトリ構造に用いた並列無共有システムにおけるアクセス頻度に着目した動的な偏り除去手法を提案した．これは偏り除去を行う Full-Window アルゴリズムと根に隣接する部分木を単位とした負荷評価から構成される．

detect_skew_&_invoke_migration アルゴリズムや Full-Window アルゴリズムは全ての PE を順に巡回するために PE 数の増加とともに処理時間が $O(\text{PE\#})$ で増加する好ましくない特性を持っている．これらトークンが PE を巡回する方式の偏り制御の並列処理方式では，トークン到着後に発生した偏りが次のトークンの到着まで解消されないため，負荷分布が動的に変化する環境において，処理時間が長いことは負荷が偏った状態が長く続くことを意味する．また，detect_skew_&_invoke_migration アルゴリズムでは，トークンが全 PE の負荷情報を格納したベクトルを伴って PE 間を巡回するため，偏り制御中のトークンによる通信コストは $O(\text{PE\#}^2)$ となり，スケーラビリティを悪化させる．

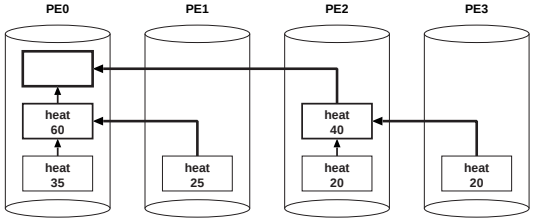
我々は、より高いスケーラビリティを持った負荷制御を行うために，1 対 1 の通信と PE 単位での並列処理を前提とした TCSH(Tree-Communication-Skew-Handling) アルゴリズムを提案している．従来の偏り制御では移動計画を立てる際に全ての PE が全 PE の負荷情報を取得していた．しかし，値域分割された並列データ構造では，偏り制御によるデータの移動は隣接 PE 間でのみ行なわれるため，実際には，全ての PE が全 PE の負荷情報の取得を行なう必要はない．隣接する PE へ送るべき負荷の決定に必要な動的情報は，PE 間の負荷平均 $L_{avg.}$，右側にある PE 全ての負荷の合計 L_{Right}，左側にある PE 全ての負荷の合計 L_{Left} の 3 つだけであり，これに加えて静的に決定されている右側にある PE の数 $PE\#_{Right}$ と左側にある PE の数 $PE\#_{Left}$ がわかっていれば，左 (右) 側の PE に移動すべき負荷 $\Delta L_{\{Left/Right\}}$ は下式を用いて求められる．

$$\Delta L_{\{Left/Right\}} = L_{avg.} \times PE\#_{\{Left/Right\}} - L_{\{Left/Right\}}$$

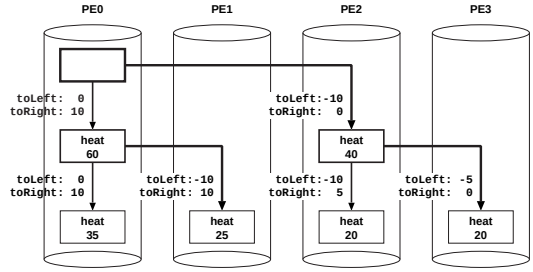
TCSH は図 1 のような通信木を構築する手続きを用いることによってこれらの動的情報を効率よく集計し，必要な情報のみを各 PE に伝えることで高い並列度を持った並列処理を実現している [12]．TCSH は負荷制御を 4 つのフェーズに分割して行なう．フェーズ 1 で TCSH は全 PE を再帰的に 2 分割し，コーディネーターと全ての PE をつなぐ通信木を構築する．フェーズ 2 で通信木の葉プロセスは自らの負荷を計算し，親プロセスに送る．負荷を受け取ったプロセスは 2 つの子プロセスから送られた負荷を記憶し，その和を自らの親プロセスに送る．フェーズ 3 でコーディネーターは負荷の平均を決定し，それを通信木を介して全ての PE に送る．このとき同時にフェーズ 2 で各中間ノードに記憶した負荷を用いて移動すべき負荷を決定する．フェーズ 4 で，葉プロセスは隣接する PE の葉プロセスと通信を行ない，フェーズ 3 での決定にしたがってデータオブジェクトを移動す



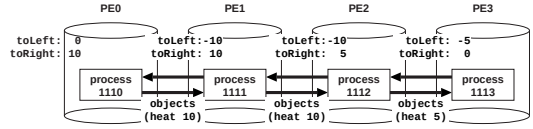
フェーズ 1　再帰的にプロセスを生成し，通信木を構築する．



フェーズ 2　葉プロセスは親プロセスにその PE の負荷を送信する．枝プロセスは親プロセスに子プロセスから受信した負荷の和を送信する．



フェーズ 3　親プロセスは左の子プロセスに { 最も左の葉が左に送る負荷, 最も右の葉が右に送る負荷 } を送信する．右の子プロセスにも同様に送信する．



フェーズ 4　葉プロセスは受信した負荷の移動量に従ってオブジェクトを隣接 PE に移動する．

図 1: TCSH の動作

る．通信木を用いることで全 PE が同じ移動計画を共有するまでの処理時間を $O(\log \text{PE\#})$ に短縮し，さらにデータオブジェクトの移動を全ての PE で独立に並行して行なうことを可能にする．

TCSH のフェーズ 1–3 では，高々定数オーダの負荷情報のやり取りを行なうだけであり，通信コストに与える影響の大きい PE 間通信のコストは $O(\text{PE\#})$ に抑えられる．TCSH ではこの通信は並列に行なわれるため，フェーズ 1–3 における通信に要する時間は高々$O(\log \text{PE\#})$ のオーダであり，高いスケーラビリティーを持つ．

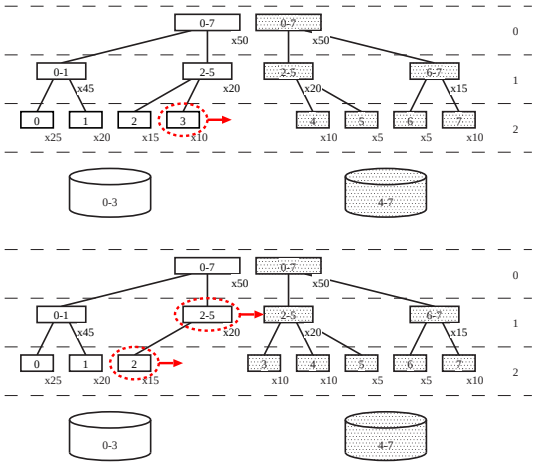
TCSH は Feelifl らが提案した，段階的な偏り制御を行なえる．偏り制御を段階的にすることにより，偏り制御によるサービスの停止は段階的でないものに比べ短くなる．[4] では段階的な偏り制御の表現のため，熱移動量の正規化係数である speed parameter α を導入している．speed parameter α による正規化を例を用いて説明する．負荷がそれぞれ 80%,120%であるような 2 つの PE，A,B があるとする．今ここで B から A へ負荷が 20%に相当するだけのデータオブジェクトを送れば，負荷は完全に均等になるだろう．この 20%を $\alpha = 0.25$ を用いて正規化すると 20%×0.25 =5%となる．すなわち実際には，負荷が 5%に相当するだけのデータオブジェクトが B から A へ送られる．このように，speed parameter α を用いた正規化を行なうことによって，移動するオブジェクトの量は減少し，偏り制御によるサービス停止が短くなる．段階的な移動はまた，負荷評価の誤差による過剰な移動を軽減する．

TCSH アルゴリズムに用いる負荷評価アルゴリズムはシステム負荷の和の平均と各 PE 毎の負荷の差分と PE の局所的な情報から，移動するデータ量を決定可能なものでなければならない．次章で述べる負荷評価アルゴリズムの DTC，値域分担熱はこの条件を満たしている [12].

3　負荷評価

偏り制御の精度においては，負荷評価アルゴリズムが重要になる．Copeland らは [1] でアクセス頻度を負荷の偏りの指標として用いることを提案し，それを熱と呼んだ．Feelifl らは Copeland が提案した熱を，粒度の扱いにおいて発展させ，値域に対するアクセス頻度の累計を用いた指標を提案した [3]．以下では Copeland らの提案した指標を熱，Feelifl らの提案した指標を値域分担熱と表記する．オブジェクト O を粒度とする値域分担熱は O の値域 $\{Rmin \ldots Rmax\}$ へのアクセス数の累計を用いた負荷の指標である．本論文では根からの距離が n であるノードを粒度とした値域分担熱を RANGE_n と表記する．RANGE_n では根からの距離が n 以下のノードについてアクセス数を計測し，より深い階層ではアクセスが均等に分配されていることを仮定する．

我々は Fat-Btree による負荷分散効果の影響を考慮した精度の高い負荷評価アルゴリズム DTC(Directory-Traverse-Cost based load-evaluation) を提案している [12]．このアルゴリズムは Fat-Btree を並列ディレクトリ構造に用いたシステムのために考案されたものである．DTC は，移動元の PE が持つオブジェクトの熱分布のみからオブジェクトの移動に伴う熱の損失の概算を得る負荷評価アルゴリズムであり，他の PE が持っている情報を必要としないため並列処理に適する．DTC では負荷評価にオブジェクトの負荷を用いる．負荷はオブジェクトのアクセスコストの累計である，すなわち，キャッシュされたオブジェクトのアクセスコストは小さく，ディスク上のキャッシュされていないオブジェクトのアクセスコストは大きい．並列ディレクトリ構造がアクセス負荷を分散させている場合のオブジェクトのアクセスコストは DTC では小さく見積もられ，過剰なオブジェクトの移動を抑える．負荷の粒度にバリエーションを与えることは，実装コストに幅を持たせる．DTC は粒度の概念を持つ．根からの距離が n であるノードを粒度とした DTC 負荷評価を DTC_n と表記する．DTC_n では根からの距離が n 以下のノードについてアクセス数を計測し，より深い階層ではアクセスが均等に分配されていることを仮定する．



DTC は根に近いノードにおける負荷分散を考慮し，葉の移動によって変化する負荷だけでなく，葉の移動が引き起こす枝の移動によって変化する負荷をも考慮することで負荷評価の精度を高める

図 2: DTC による負荷評価

4　シミュレーションによる性能評価

本論文では 2 つの目的から TCSH に関するシミュレーション実験を行なった．第 1 の目的は，TCSH によるスケーラビリティ向上の確認である．第 2 の目的は，TCSH との組み合わせに適した負荷評価アルゴリズムの選択である．

本シミュレーションでは偏り除去を評価する指標として，負荷が偏った状態が解消されるまでの時間を表す，偏り除去に要する時間を用いる．シミュレーションに用いるデータベースは初期状態において個々の葉と PE 毎のデータ配置のそれぞれが zipf 分布 [7] で表される偏りを持っている．本シミュレーションではデータベースの偏りを評価する指標として最大負荷率を用いる．最大負荷率は最も負荷が大きい PE の負荷の平均負荷に対する百分率であり，性能の低下を招くホットスポットの状況を良く反映する．本シミュレーションではシステムにおいて許容される

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
オブジェクトサイズ	4KB
インデクスノードの最大エントリ数	200
インデクスノードの利用率	ln 2
メッセージのセットアップ時間	200μs/message
オブジェクトの移動速度	8K-object/sec =32MByte/sec
オブジェクトへのアクセスに要する時間	
ディスク	10ms/object
キャッシュ	1μs/object
PE 数	2–64
データページ数/PE	64K =256MByte
キャッシュ連想度	8
キャッシュセット数	128 =4MByte
アクセス偏り (zipf 分布の母数)	0.5, 0.1, 0.9
クエリの到着間隔	64msec
偏り除去の間隔	32K ≅2100sec
偏り除去の speed parameter α	0.25
アクセスコスト比	10^4

偏りを最大負荷率で 120%までと設定し，初期状態から偏りが許容される量に減少するまで偏り除去を繰り返し，その実行時間を測定する．ただし，この実行時間に偏り除去の間に行なわれたクエリ処理の実行時間は含まれない．

本シミュレーションでは，各 PE を時計カウンタとデータ構造を持つインスタンスとして表現し，探索/偏り除去ごとに時計カウンタとデータ構造を更新して偏り除去に要する時間を測定する．本シミュレーションに関する主要なパラメータを図 1 に示す．現在のネットワーク通信速度，CPU 性能，ディスクアクセス速度の状況から，オブジェクト移動におけるボトルネックがディスクのシーケンシャルアクセス速度であるとの仮定は公平さを欠かない．本シミュレーションにおいては，ディスクのシーケンシャルアクセス速度の現状を鑑みて，毎秒 8K 個までのデータオブジェクトが隣接するディスクに移動できるものとする．また，典型的なディスク装置に付随しているメモリ容量は数 MB 程度であるため，シミュレーションではキャッシュに連想度 8，セット数 128 の LRU を用いる．過剰な移動を抑制し，サービスの停止を短くするための speed parameter α は 0.25 とする．評価に用いるキャッシュとディスクアクセスの負荷比は 10^4 とする．

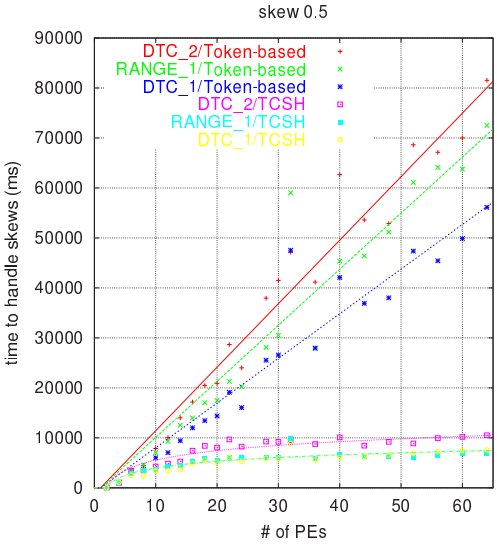


図 3: 初期偏りが中程度の場合 (zipf(0.5)) の偏り除去時間

DTC_n は根からの距離が n であるノードを粒度とした DTC 負荷評価，RANGE_n は根からの距離が n であるノードを粒度とした値域分担熱を用いた負荷評価を表す．

TCSH による偏り除去に要した時間の短縮　図 3,4,5 は，TCSH とトークン方式 (図中，Token-based と表記) の実行時間を初期の偏りと PE 数を変化させながら測定したものである．実験結果は TCSH の時間がトークン方式の分散制御を行なった場合に要した時間に比べて短縮されていることを示している．初期の偏りが小さい場合の測定結果の精度が低くなっているものの，PE 数の増加に伴う TCSH の時間の増加は理論的な予想である log(*PE*#) にほぼ一致しており，TCSH が高いスケーラビリティーを有することが確かめられた．図 6 は TCSH のトークン方式に対

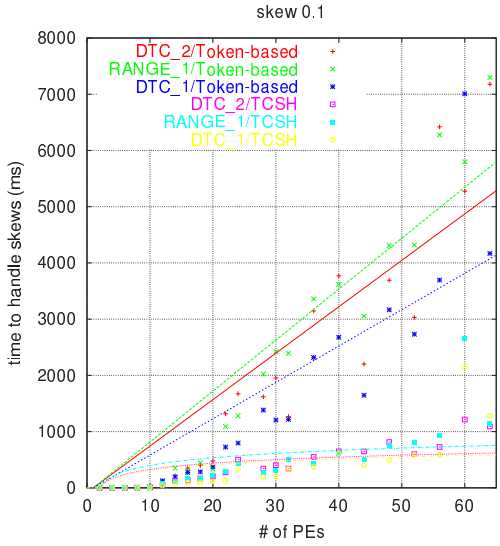


図 4: 初期偏りが小さい場合 (zipf(0.1)) の偏り除去時間

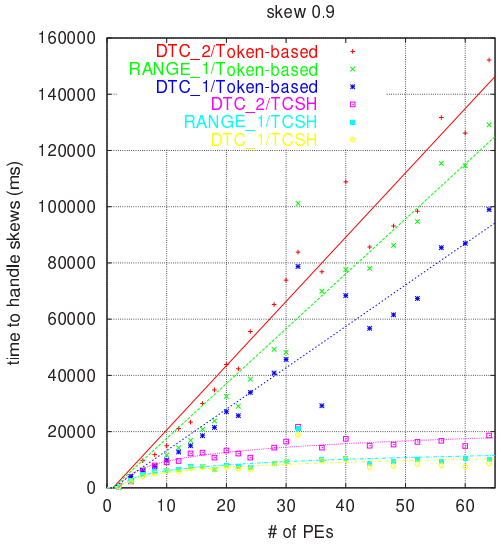


図 5: 初期偏りが大きい場合 (zipf(0.9)) の偏り除去時間

する実行時間の改善比を初期の偏りごとにまとめたものであり，TCSH がトークン方式の性能を改善していることと，その改善比が PE 数の増加と共に増すことを示している．実験結果から初期の偏りと改善比の関係を議論するのは難しいが，初期の偏りが大きいほど性能の改善比が向上する傾向が伺える．なお，初期の偏りと PE 数の両方が小さい場合は偏り除去の必要が無いため，グラフから省かれている．

負荷評価と性能　図 7 は TCSH のトークン方式に対する実行時間の改善比を負荷評価方式ごとにまとめたものである．PE 数が多い場合に注目すると，偏り除去に要した時間の短縮は，評価精度の低い値域分担熱による評価を用いた場合に最も大きく，評価精度の高い DTC_2 による評価を用いた場合に最も小さい．TCSH による偏り除去に要した時間の改善は測定精度の低い負荷評価において特に有効であり，その傾向は PE 数の増加に伴ってより強まっている．

図 3,4,5 に示した結果は DTC との組み合わせが値域分担熱との組合わせに比べて高い性能を導くことを示している．

我々は [12] で負荷評価における熱の粒度を細かくすると偏り除去のステップ数が減ることを報告した．しかし，図 8 は負荷評価における熱の粒度を細かくすると偏り除去に要する時間が増加することを示している．これは，負荷評価における熱の粒度を細かくしたことによる負荷計測コストの増加が偏り除去に要する時間に影響を与えていることを示している．

まとめ　TCSH は偏り除去に要する時間の短縮に有効であり，その効果は PE 数が大きいほど高い．PE 数が小さい場合，特に PE 数 2 の場合には他の場合と異なる挙動を示しており，特別な扱いが必要になる．実験の結果，負荷評価 DTC_1 が TCSH との組み合わせに最も適しているとわかった．また，TCSH の実行時間のオーダが $O(\log PE\#)$ であり，優れたスケーラビリティを有することを確認した．

5　結論

本稿では，TCSH による偏り除去に要した時間をシミュレータを用いて測定し，TCSH による時間短縮

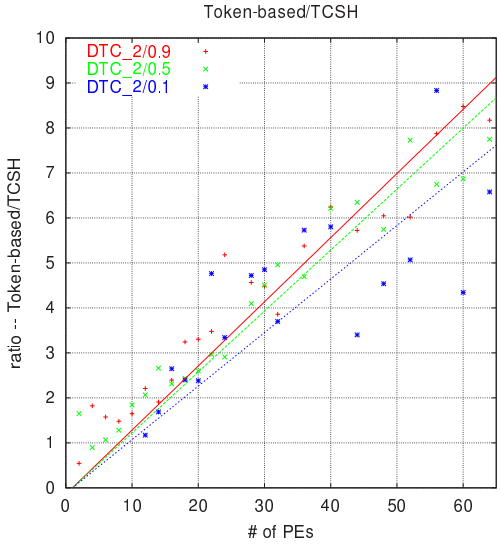


図 6: 初期偏りと性能比

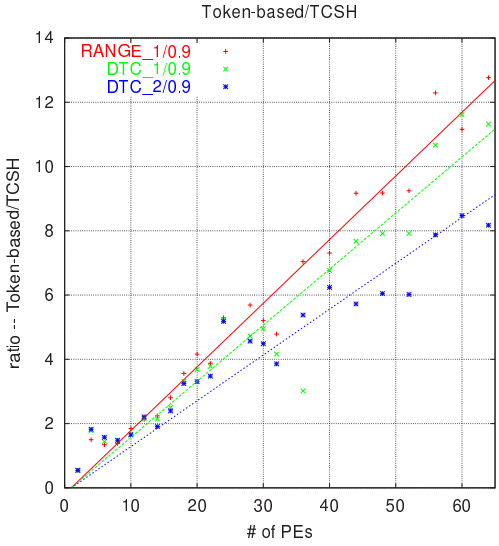


図 7: 負荷評価と性能比

を評価した．TCSH は通信木を構築することによって処理の並列度を上げるため，PE 数が多い場合に大きく時間を短縮する．TCSH による実行時間の短縮は，偏りのある状況を速やかに解消し並列データベースの性能を向上させる．シミュレータによる実験結果は TCSH の実行時間が $O(\log PE\#)$ であるとする [12] の理論的予想と一致し，TCSH が高いスケーラビリティを持つ偏り制御の並列処理を行なうことを示した．また，負荷評価アルゴリズムとの組み合わせ実験では，DTC_1 と TCSH との組み合わせが効率よく並列データベースの偏りを除去するという結果を得た．

今後は，さらに大きな PE 数での性質を測定し，TCSH をさらに詳しく評価する．また，偏り制御による探索/更新操作への影響を軽減する技術を異なるアプローチによって研究することが必要である．これについては，偏り制御の平常動作への影響を偏り制御が一度に移動するデータ量を制限することで軽減する案を検討中である．さらに，精度の高い負荷評価を低いコストで実現する手法もまた必要になるだろう．

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) 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] 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] 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.

[3] Hisham Feelifl and Masaru Kitsuregawa. The simulation evaluation of heat balancing strategies for btee index over parallel shared nothing machines. Technical Report DE99-88, 電子情報通信学会データ工学研究会, Nov 1999.

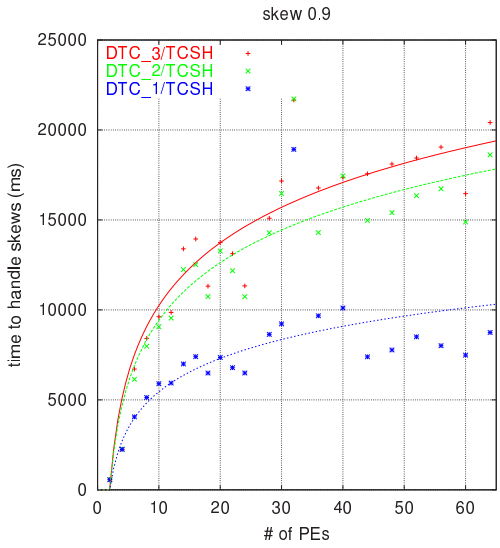


図 8: 初期偏りが zipf(0.9) の場合の偏り除去時間の粒度による差

[4] 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.

[5] 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.

[6] K. A. Hua and J. X. W. Su. Dynamic Load Balancing in Very Large Shared-Nothing Hypercube Database Computers. *IEEE Transactions on Computer*, Vol. 42, No. 12, pp. 1425–1439, December 1993.

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

[8] 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.

[9] H. Lu and K.-L. tan. Dynamic and Load-Balanced Task-Oriented Database Query Processing in Parallel Systems. In *Proc. of Third EDBT*, pp. 357–372, 1992.

[10] C. B. Walton, A. G. Dale, and R. M. Jenevein. A Taxonomy and Performance Model of Data Skew Effects in Parallel Join. In *Proc. of 17th Int'l. Conf. on VLDB*, 1991.

[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.

[12] 渡邊明嗣, 横田治夫. ディレクトリ探索コスト重視分散ディスク偏り制御の評価. Technical Report DE2001-110, 電子情報通信学会データ工学研究会, Oct 2002.