データベースシステム 125−69
（2001. 7. 19）

探索コスト評価による分散ディスク偏り制御

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

使用頻度を考慮した分散ディスクの偏り制御を，分散ディレクトリ探索コストの評価から述べる．Fat-Btree を並列ディレクトリ構造に用いた並列データベースシステムにおける，負荷評価の詳細度と精度の関係について議論し，分散ディレクトリ探索コストの詳細な評価が並列データベースシステムのアクセス頻度の分散とデータ量の分散に影響を与えることを示す．

キーワード Fat-Btree, 分散ディスク, 負荷分散, 分散配置, 偏り制御, 探索コスト評価

A Search-Cost Based Technique for Online Heat-balancing of Parallel Disks

WATANABE Akitsugu* 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

We discuss load balancing of parallel database systems in terms of access cost for searching parallel directories. We discuss the relation between accuracy of measure of load and quority of load barancing. We claim a low accuracy measure of load degrades a quority of load barancing, and it's more serious for a large systems. We conclude that a high accuracy measure of load and its efficient imprementation are important for databases in the future.

KEYWORD Fat-Btree,parallel disk,skew handling,search cost evaluation,load balancing,declustering

1 序論

データベース処理能力を向上させるという目的で，商業的な開発を含む多くの並列データベースシステムが発表されている [1]．なかでもプロセッサー間で共有のメモリおよびディスクを持たない並列無共有システム [2] はスケーラビリティーの観点から大いに注目されている [3]．

並列無共有システムにおけるプロセッサエレメント (PE) が格納しているオブジェクトに対する探索と更新の操作は，PE 間でメッセージを交換することで行われる [4]．負荷分散は性能および規模の向上を線形的にするための非常に重要な技術である [5]．負荷が片寄っている場合，実行時間は最も負荷の大きい PE が処理を完了するまで引き延ばされてしまう．したがって，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 [6, 7, 8, 9]．

クエリ処理の過程における中間データの生成のように負荷の偏りには何種類か理由があるが，負荷分散に最も大きな影響を与えるのはデータ配置である．データベースのデータの水平分割は並列データベースにおける各 PE での操作において特に重要である [6]．

データを水平分割する戦略は 3 つのタイプに分けられる．すなわち，値域，ラウンドロビン，ハッシュの 3 つである [3]．値域分割では，完全一致クエリに当てはまるオブジェクトデータがどの PE に含まれるべきかを決定できる．また，値域分割はレンジクエリを扱うことができ，クラスター I/O 操作を利用できるので，値が近いオブジェクトの I/O を減らすことができる．しかしながら，この戦略は潜在的にデータの配分に偏りをもたらす可能性があるので，負荷分散の質を下げる恐れがある．分割に使う境界が固定されていれば，初期のデータ配置に偏りがなかったとしても更新を繰り返すことでバランスが崩れるだろう．一方，ラウンドロビン分割では偏りは生じないが，データが格納されている PE についての情報が無いので，どのような完全一致クエリやレンジクエリに対しても全ての PE が処理しなければならない．ハッシュ分割は完全一致クエリを扱える上に偏りも非常に小さいが，レンジクエリや I/O のクラスタ実行にはむかない．

並列データベース環境においては，高速にアクセスできる手段を用意することもまた重要である．値域分割の長所を利用しつつ各 PE への高速アクセスを実現するための並列ディレクトリ構造として，並列 Btree が発表された [10]．並列 Btree は完全一致クエリとレンジクエリ，さらに I/O のクラスタ実行を扱うことができ，しかも各 PE に割り当てられるデータ量を Btree のインデクスノードによって分散させることができる．Fat-Btree は更新手続きのコストを考慮した並列 Btree 構造である [11]．アクセス頻度の高い根付近のノードを多数の PE に格納し，アクセス頻度の低い葉付近のノードを少数の PE だけに格納することで，インデクス更新に関わる PE の数を減らしつつ探索時の並列実行数を保っている．

Fat-Btree を効率よく運用するためにはオブジェクトの動的な再配置が不可欠であるため，効率的な再配置手法が議論されてきた．横田, 宮崎らは 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動的なデータ量の偏り除去手法の素案を示した [11]．Feelifl, 喜連川,Ooi はアクセス頻度と両端の PE が特異点になる値域分割の特性に着目した動的な偏り除去手法を提案した [12, 13]．Feelifl らは偏り除去による負荷の見地から根に隣接する部分木を単位にアクセス頻度を推定する手法を取った．しかしながら，上位のノードを共有している Fat-Btree では，葉を探索する途中で他の PE へ処理が移行することがしばしばある．枝ノードと葉ノードのアクセス頻度を共に考慮することで，PE 間の負荷をさらに軽減するだけでなく，我々は負荷を詳細に評価することでデータ量の偏りを除去する効果が得られることをも期待している．

本稿では探索コストを詳細に計測した場合の負荷計測の精度をより大まかな計測のそれと比較し，精度の向上がシステムの性能に与える影響を評価する．2 節では Fat-Btree について述べる．3 節では偏り制御と負荷評価についての議論をする．4 節ではシミュレーションを用いて精度の向上がシステムの性能に与える影響を評価する．

2 Fat-Btree

2.1 並列 Btree

値域分割を用いることで，完全一致検索，レンジクエリ，近傍検索などを行う際にクラスタ単位での高速なアクセスが行えるのは大きな魅力である．これによって冗長な I/O 手続きの回数が減り，外部操作の並列性を内部操作の並列性と対等のものとして導入できるようになる．しかしながら，単純な値域分割ではページの挿入を繰り返す間にデータの偏りが生じてくる．単純な値域分割では分割に用いる値域が静的に規定されているため，偏り除去は非常に高くつく．もし，値域が動的に変更することができ，かつ PE 間のデータの再配置を安価に行えるならば，値域分割戦略は並列ディレクトリ構造のよき基盤となるだろう．

Btree を並列ディレクトリ構造として用いることを提案したのは Seeger と Larson である [10]．彼らはレコードディストリビューション，多ページ Btree，ページディストリビューションの 3 種類の並列 Btree を比較した．結局，彼らはページディストリビューションが並列ディレクトリ構造に適していると結論付け，負荷分散をページの分配によって行う LOB-tree を提案した．LOB-tree は大きなデータの偏りが無く，レンジクエリの負荷を複数の PE に分散できる．

ページディストリビューション並列 Btree は並列探索に適していたが，更新手続きを十分に考慮していなかった．[10] は，インデックスの分布については十分に取り上げていない．Btree の形状はデータアイテムを挿入／削除することで変形していくので，CWB(Copy-Whole-Btree)[1] 方式では挿入／削除が行われる度に全ての PE がインデックスの更新を行わなければならない．システムが更新操作を行うのであれば，システムのスループットは著しく低下する．一方，SIB(Single-Index-Btree)[2] 方式では並列探索に深刻なボトルネックが発生する．

[1] Btree のインデックス部分を全ての PE 上に複製する並列 Btree 構造

[2] Btree のインデックス部分を単一の PE のみが持つ並列 Btree 構造

2.2 Fat-Btree の構造

まず Fat-Btree の基礎となる Btree 構造について述べる．Btree にタプルを挿入すると，まず葉ノードが更新される．葉ノードが持つタプルの数がノードの最大数を越えたならば，その葉ノードは 2 つの葉ノードに分割され，元の葉ノードの親ノードが更新される．親ノードの更新の結果，親ノードが持つ葉ノードの数がノードの最大数を越えたならば，親ノードはさらに分割され，以下同じように繰り返す．したがって，更新頻度は葉に近いノードほど高くなる．

この性質を考慮して，Fat-Btree では葉に近いノードは少ない PE のみが持ち，葉から遠く更新頻度の少ないノードを多くの PE にコピーして共有する．この制約をより体系的に表すことにしよう．各 PE は Btree の部分木を持つ．各 PE に含まれているのは，その PE に配置されたページと，そのページから Btree の根ノードまでの間にある枝ノード全てである．

形式的な定義のため，以下の記法を用いる．個々の木のノードを識別子 i で区別し，そのレベルを $L(i)$ と表記する．ノード i がノード j の親であるなら，$P(j) = i$ であり，$L(j) = L(i) + 1$ である．根ノード r においては $L(r) = 1$ である．木の高さを H と置くとき，$L(l) = H$ となるノード l は葉ノードである．

定義 1 (Fat-Btree) S_i をノード i を格納している PE の集合とする．以下の条件を満たす並列 *Btree* を

$$S_i \supseteq S_j \ \textit{if} \ \ i = P(j)$$

$$\textit{and} \ \ \forall p \forall i \{p \in S_i \Rightarrow \exists j \{i = P(j), p \in S_j\}\}$$

Fat-Btree と呼ぶ．

上の定義より，$C_i = \{k \mid P(k) = i\}$ とおくと

$$S_i = \bigcup_{j \in C_i} S_j$$

が成り立つ．したがって，根ノードについて $F_h = \{h \mid L(k) = h\}$ とおくと

$$S_r = \bigcup_{i \in C_r} S_i = \bigcup_{i \in C_r} \bigcup_{j \in C_i} S_j = \cdots$$

$$= \bigcup_{i \in F_2} S_i = \bigcup_{j \in F_3} S_j = \ldots = \bigcup_{l \in F_H} S_l$$

が成り立つ．これは根ノードが葉ノードを持つ全てのPEに格納されていることを表す．$|S_i|$ を S_i の濃度とおくと，S_i はノード i のコピーの数を表す．したがって，$|S_r| = |U|$ は根ノードを持つPEの数を表す．

CWB と SIB を同様の記法で表すと，CWB は $\forall i S_i = U$ を満たすものであり，SIB は $\forall i |S_i| = 1$ を満たすものである．

3 偏り制御と負荷評価

3.1 偏り制御

横田, 宮崎らは [11] 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データ量の動的な偏り除去手法を提案した．彼らは偏り除去を効率的に行うために，PE が取り扱う値域を変更することによって移動するデータページが常に連続しているという値域分割の特性と，インデクス木の構造に関する Fat-Btree の特性に注目した．移動するデータページが連続していることにより，偏り除去の際のページへのアクセスは高速なクラスタ I/O 操作を利用することができる．また Fat-Btree では，データページを移動するときにページおよびノードのロケーションとコピーを持つPEが変化するだけで，ディレクトリの構造自身は変化しない．また，連続するデータページの移動によってロケーションが変化する枝ノードは全てある枝ノードを根とする一つの部分木に含まれている．

横田, 宮崎らはこの動的データ量偏り除去手法のために偏り検査とページ移動を行うアルゴリズムの素案を提案した．この素案は，アクセス状況を PE 間で回覧し，偏りがある閾値を越えた場合に偏り除去を行うというシンプルなもので，偏りの評価や閾値の決定については触れられていない．

Feelifl, 喜連川,Ooi は [12, 13] 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 アルゴリズムと負荷評価の精度によって構成されている．

Full-Window アルゴリズムは [12] にまとめられている．このアルゴリズムは両端の PE が特異点になる値域分割の特性を利用している．偏り除去アルゴリズムによって移動することになったページが特定の PE 間を往復しないように，アルゴリズムは左右の端から中央方向に向かって注目する PE を変えながら，中央方向へ移動させるページを全て検出したのちに中央から左右の方向に向かって注目する PE を変えながら，左右方向に移動させるページを検出する．彼らはまた，偏りが大きいシステムでは偏り除去アルゴリズムによるページの移動が少なめになるように分割した方が効率が良いことを発見した．両端の PE が特異点になることに関しては，値域の両端を連結したリング状の設計が偏り分散に効果的であるという報告もある [13].

Copeland は負荷を熱，すなわち，値域に対するアクセス頻度と定義した [14]．Feelifl らは熱の評価精度について議論し，評価の精度とはその値域を扱う際の詳細さのレベルであるとした [13]．彼らは値域を扱うレベルを { データページ，インデックス枝，インデックス木，PE} のように分類し，部分木レベルの評価[3]が PE レベルでの評価より詳細な戦略を立てられることから動的偏り除去に適していると主張した．

3.2 負荷評価のレベル

Fat-Btree を並列ディレクトリ構造に用いたデータベースシステム S について，負荷評価のレベルとその精度の関係を確率的手法を用いて評価する．以下では例として，インデックスノードの最大エントリ数は 150，S のインデクス使用率はどのインデクスでも $\ln 2$ にほぼ等しく，1PE あたりの平均ページ数は 1M，システムが許容する偏りの閾値 ζ は 10% であるとする．

[3] すなわち，PE それぞれの根ノードが持つ部分木のさらにそれぞれについて熱を評価する

θ	値域に含まれるページ数 R			
	10K	25K	50K	100K
0.1	0.11%	0.27%	0.54%	1.09%
0.2	0.25%	0.62%	1.25%	2.51%
0.3	0.44%	1.12%	2.25%	4.54%
0.4	0.73%	1.88%	3.85%	7.83%
0.5	1.19%	3.19%	6.68%	13.96%
0.6	1.97%	5.56%	12.13%	26.42%
0.7	3.28%	9.87%	22.69%	52.15%
0.8	5.38%	17.37%	42.20%	102.66%
0.9	8.55%	29.40%	75.08%	192.28%

表 1: zipf(θ) 分布における移動するページ数とそれら全体の熱の標準偏差 σ (1 つの PE が持つ熱を 100%としたとき)

3.2.1 PE レベルの評価

PE レベルの負荷評価は実装が簡単だが，評価の精度は悪い．たしかに，一定時間内に PE が持つ全てのオブジェクトがアクセスされた回数の和はその PE の負荷状態を良く表現する．しかしながら，偏り除去アルゴリズムはページを移動した際にどれだけの熱が移動するかという情報をも必要とする．

今，アルゴリズムが閾値 ζ に等しい量の熱を隣接する PE に移動しようとしている．PE レベルでの評価では PE が持つ全てのオブジェクトがアクセスされた回数の和のみがわかるため，アルゴリズムが知りうるのは PE が持つオブジェクトの熱の期待値のみである．仮にこの PE が平均と同じ $1M$ ページを持っているとするなら，アルゴリズムは期待値の和が移動しようとしている熱と等しくなるようにおおよそ $1M \times \zeta \simeq 100K$ ページを移動しようとする．システムにおけるページの熱分布が zipf(θ) モデルで表現されるとき，$1M$ ページのうちの $100K$ ページを移動した場合の標準偏差を算出すると，$0.3 \leq \theta \leq 0.5$ で 4.5%～14%程度ある[4](表 1)．すなわち，実際に移動する熱が期待している量より著しく少なかったり多かったりするような事象が起りうる．

[4]一般に $0.3 \leq \theta \leq 0.5$ における zipf(θ) モデルは現実のデータベースの良い近似であると言われている

3.2.2 部分木レベルの評価

部分木レベルの負荷評価は PE レベルのものよりも 多少コストはかかるが，比較的簡単に実装できる．しかしながら，環境によっては精度を高める効果はあまり期待できない．

部分木レベルの評価では，部分木全体が移動の対象になるような場合を正確に扱うことができる．むろん，多くの場合には「端数」を処理するために部分木の一部だけを移動する場合を含むだろう．部分木が持つページの数 $|S|$ が移動するページの数 i に比べて十分少ない場合について．移動するページのうち，実際の熱が正確にわからないものの期待値は $|S|/2$ である．しかしながら，部分木全体の熱は既知なので部分木が持つ移動しないページの熱の和が求められるのであれば，移動するページの熱の和も求められる．したがって，実際の熱と期待値との間の誤差は $\min(|S|-i, i)$ 個のページの熱の分散から推定できる．

例としたインデックスノードの最大エントリ数が 150，S のインデクス使用率がどのインデクスでも $\ln 2$ にほぼ等しい場合では，根ノードは約 100 個の部分木を持つことになる．システムの PE 数が高々4 個程度なら部分木が持つページ数は 40K 程度であり，実際に移動するページの熱の標準偏差は 0.44%～1.19%程度になるだろう (表 1 の 10K の列)．しかし，システムの PE 数が 20 個なら部分木が持つページ数は 200K 程度であり，実際に移動するページの熱の標準偏差は 2.25%～6.68%程度に増える (表 1 の 50K の列)．

多くの PE があるような場合には別の問題から高い精度を期待できなくなる．システムの PE 数が 100 個を越えるような場合には部分木と PE がほぼ 1 対 1 対応してしまうために PE レベルの評価との間に大きな差は無くなる．すなわち，前節で述べたような問題が発生しうる．

3.2.3 データページレベルの評価

データページレベルの評価は最も高い評価精度を保証する．多少コストはかかるが，PE レベルの評

価や部分木レベルの評価と異なり静的な分布においてはページの移動量が余分だったり，あるいは不足だったりすることはない．ページの移動量を正確に評価できるため，偏り除去操作の頻度を低く抑えることができる．偏り除去操作はI/Oを伴うためにそれ自身が負荷を増やす．その頻度を低く抑えることで，これはシステム全体の負荷を低くすることができる．実装に必要なコストはPEレベルの評価や部分木レベルの評価に比べて高いが，ページ／インデックスのアクセス回数をページ／インデックスと共にキャッシュに保存し，それらがキャッシュから出るときにはじめてディスクに書き出すことで実装のコストを下げることができる．

データページレベルの評価に加え，枝ノードの熱を評価することでデータ量の偏りを除去する効果が期待できる．単純なデータページレベルの評価では，データページにおける熱の偏りと同様の偏りをその祖先のノードたちに仮定するが，複数のPEに格納されている枝ノードでは負荷をPE間で分散するため，かならずしも同じPEが持つページの熱の偏りを反映せず，よりなだらかな熱分布を持っている．こうした根ノードに近い枝ノードの平均的な熱分布を正確に評価することでデータ量の偏りを除去する効果が期待できる．

図1はデータページレベルの評価と枝ノードの熱を評価するアルゴリズムである．`heatOf`は与えられたノードを移動したときにどれだけの熱が移動するかを計算する．移動する熱は，与えられたノードの熱だけではない．与えられたノードが同じPEに兄弟を持たないならば，その親ノードもまた移動する．そのために同じPEに兄弟を持つことを調べるサブルーチンが`isOnlyChildIn`である．

```
sub isOnlyChildIn(node t, PE e) {
/* t はノードまたはページ */
    boolean ans = true;
    node p = parent(t);
    if(countOfChildIn(p,e)<>1) {
        ans = false;
    }
    return ans;
}


algorithm heatOf(node t) {
/* t はページ */
    PE e = whereIs(t);
    lheat = 0;
    lheat = heat(t,e);
    for (node f=t;
         f<>root and isOnlyChildIn(t,e);
         f=parent(f) ) {
        node p = parent(f);
        lheat += heat(p,e);
    }
    return lheat;
}
```

図 1: 全オブジェクトの熱を評価するアルゴリズム

4 シミュレーションによる解析

4.1 解析手法

データページレベルの評価と部分木レベルの評価をシミュレーションを用いて比較する．目的は偏り除去後の状態の比較であるため，シミュレーションでは，ページの熱についてアクセス分布モデルに沿ったシンドロームを作り，各レベルでの評価にしたがって偏りを除去した後に，定められた回数の探索を行い，その平均レスポンス時間を比較する．さらに，簡単のため，以下の条件を仮定する．

- 1つのタプルに対する検索のみを扱う
- Btreeにおけるデータページの配置はバランスしている
- PEに対して均等にアクセス要求が行われる
- 各タプルに対するアクセス頻度は静的に定められた分布にしたがう

タプル数 (キャッシュ)	除去無し	部分木	ページ
128K(無)	68.2	68.6	69.1
64K(有)	326.7	333.0	340.2
128K(有)	235.6	238.6	240.6

表 2: PE 数 4 における負荷評価のレベルとスループット (探索/sec) の比較

- タプルの更新は行わない
- ディスク，キャッシュへのアクセス時間と他の PE との通信時間のみを考慮してレスポンスタイムを算出する

偏り除去には以下の 3 つの負荷評価レベルを用いる．

1. 偏り除去を行わない
   熱の分布を考慮せずに，ページ数を均等に分配する
2. 部分木単位で評価する
   各 PE の根ノードの各部分木までの熱の分布を考慮する
3. データページ単位で評価する
   全てのオブジェクトにおける熱の分布を考慮する

4.2 シミュレーション結果

表 2 は PE 数 4 における負荷評価のレベルとシミュレーションにおけるスループットである．

このような比較的小規模な構成でも部分木レベルの負荷評価だけで数%の性能向上が見込める．さらに，負荷をノードおよびページを考慮して評価することでその効果が倍増しているのが見て取れる．

残念ながら，実験規模が小規模であったなどの要因からデータ量の偏りを除去する効果は確認できなかったが，負荷評価の精度がスループットに少なからず影響を与えることが確認できた．

5 結論

本稿では，探索コストを詳細に評価した場合の負荷評価の精度をより大まかな評価のそれと比較し，また負荷評価の精度の向上による偏り除去の精度向上がシステムの性能に与える影響を評価した．負荷評価の精度が低いことにより偏り除去の精度を下がり，偏り除去の頻度を増やし，偏り除去による負荷を増やす．多くの PE を用いる大規模な並列データベースシステムでは，精度の低下はより深刻な問題である．

本稿ではシミュレーションを用いることで，また負荷評価の精度と偏り除去の精度の関係を確認した．このシミュレーションは小規模のデータベースのものであり，今後は，大規模な並列データベースにおける負荷評価の精度と偏り除去の精度の関係の調査，および精度の高い負荷評価を低いコストで実現する手法が必要である．また，本実験で確認できなかったデータ量の偏りを除去する効果についての研究および測定を継続する．

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] M. G. Norman and P. Thanisch. *Parallel Database Technology: An Evaluation and Comparison of Scalable Systems.* The Bloor Research Group, 1995.

[2] M. Stonebraker. The Case for Shared Nothing. *Database Eng.*, 9(1), 1986.

[3] David DeWitt and Jim Gray. Parallel Database Systems: The Future of High Performance Database Systems. *Communications of the ACM*, 35(6):85–98, June 1992.

[4] G. Graefe. Query Evaluation Techniques for Large Databases. *ACM Computing Survey*, 25(2):73–170, 1993.

[5] C. Xu and F. C. M. Lau. *Load Balancing in Parallel Computers*. Kluwer Academic Publishers, 1997.

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

[7] 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.

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

[9] K. A. Hua and J. X. W. Su. Dynamic Load Balancing in Very Large Shared-Nothing Hypercube Database Computers. *IEEE Transactions on Computer*, 42(12):1425–1439, December 1993.

[10] B. Seeger and P. Larson. Multi-Disk B-trees. In *Proc. of ACM SIGMOD Conf.. '91*, pages 436–445, 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*, pages 448–457, 1999.

[12] 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.

[13] 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.

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