2005－DBS－137（Ⅰ）（32）
2005／7／13

並列 Btree 構造における負荷分散処理の並行性制御への影響

吉原朋宏[†]　渡邊明嗣[†]　小林大[†]
田口亮[††]　上原年博[††]　横田治夫[†††,†]

我々が提案している並列 Btree 構造向けの並行性制御手法 MARK-OPT が，アクセスパターンが各 PE(Processing Element) で均一である環境において有効であることを示してきた．本稿では，アクセスパターンの偏りがある環境における従来手法との比較実験から，アクセス偏りによる MARK-OPT への影響を考察し，MARK-OPT がアクセス偏りがある環境においても有効であることを示す．また，そのアクセス偏り除去のための有効な手段であるデータマイグレーションの並行性制御に MARK-OPT を用い，データマイグレーションが発生する頻度を変化させた場合の実験から，データマイグレーションによる MARK-OPT への影響を考察し，MARK-OPT がデータマイグレーションの並行性制御として有効であることを示す．

Influence of Load-Balancing Operations on a Parallel Btree Concurrency Control

Tomohiro Yoshihara ,[†] Akitsugu Watanabe ,[†] Dai Kobayashi ,[†] Ryo Taguchi ,[††]
Toshihiro Uehara [††] and Haruo Yokota [†††,†]

We proposed a new concurrency control method, MARK-OPT, for parallel Btree structures, such as the Fat-Btrees. Previous experimental results demonstrated that the MARK-OPT was effective in the environment where the access frequencies were uniform. In this paper, we evaluate the influence of access skews on the MARK-OPT to show that the method is also effective even when the access pattern has skew. To mitigate the access skew, data should be migrated between data storages. We then evaluate the performance of the MARK-OPT with the data migration. Experiments with changing the pattern of the data migration indicate that the MARK-OPT is also effective as a concurrency control method for data migration.

1. はじめに

データベース用無共有並列計算機における検索，更新処理は，参照されるデータが配置されている各 PE(Processing Element) 上で並列に実行されることが望ましい．アクセス集中による負荷の偏りが存在する場合，負荷が大きい PE がボトルネックとなり，全体の処理性能が低下してしまう．したがって，各 PE で負荷を均等化することは処理性能の向上につながり，負荷を均等にするためのデータ分配方式が重要になる[1),2)]．

従来のデータ分配方式にはハッシュ分配方式や値域分配方式[3)] などがあるが，ハッシュ分配方式では領域指定された問い合わせや，連続したアクセスの I/O 回数を削減するクラスタ化に対応できないという欠点がある．一方，値域分配方式では，静的決定された分割境界に沿って分割するため，データ更新によってデータ配置の偏りが生じたときに均一化するコストが非常に大きくなる欠点がある．

データ分配方式として値域分配方式を採用した上で，インデックス構造に並列 Btree を用いる研究がある．並列 Btree を利用することによって，両方式の欠点が解消でき，同時に高速アクセスが可能になる．しかし，従来の並列 Btree ではディレクトリ更新によるスループットの低下や，少数の PE へのアクセスの集中といった問題が生じる．

これらの問題を解決するために，新しい並列 Btree 構造として Fat-Btree が提案されている[4)~12)]．ディレクトリ構成として Fat-Btree を用いることで，単一キー問い合わせ，レンジ問い合わせが並列に高速実行できることが確率モデルの検証[4)]，および nCUBE3[6)] や LAN 環境での PC クラスタ[12)] 上への実装による実験により明らかにされている．

† 東京工業大学 大学院 情報理工学研究科 計算工学専攻
Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology
†† NHK 放送技術研究所
NHK Science & Technical Research Laboratories
††† 東京工業大学 学術国際情報センター
Global Scientific Information & Computing Center, Tokyo Institute of Technology

Fat-Btree を含めた並列 Btree に適した並行性制御方式として INC-OPT 方式が提案されている[7),10)]．無共有並列計算機向けに設計されている INC-OPT は，従来の Btree の並行性制御方式である B-OPT[13)] や ARIES/IM[14)] より優れたパフォーマンスを示すことが明らかにされている[7),10)]．しかし，INC-OPT は Btree の構造変化を起こす操作 (SMO:Structure Modification Operation) が広範囲で行われるとき，ルートページからの木降下リスタートが多数必要となり，システムの処理性能の低下をもたらす．

そこで我々は，並列 Btree 構造の新たな並行性制御として，INC-OPT を改良した MARK-OPT 方式を提案している[15)]．MARK-OPT は，楽観的なラッチカップリング中に SMO が発生するポイントのマーキングを行う．これにより，従来の INC-OPT 方式と比較して，SMO 発生時のリスタート回数を少なく抑え，SMO のための排他ラッチ獲得時間を短縮することができる．また MARK-OPT が，アクセスパターンが各 PE(Processing Element) で均一である環境において有効であることは明らかにされている[15)]．

本稿では，アクセスパターンの偏りがある環境における従来手法との比較実験から，MARK-OPT に対するアクセス集中の影響を考察する．また，データの移動によるデータマイグレーションは，アクセス偏り除去のための有効な手段であり，データ移動に並行性制御に MARK-OPT を用い，データ移動を発生する環境での実験から，MARK-OPT に対するデータ移動の影響を考察する．

以下に本稿の構成を述べる．まず 2 節で Fat-Btree とそのデータ移動について述べる．次に 3 節では従来の並行性制御手法と我々の提案する並行性制御手法について述べる．4 節では提案手法と従来手法の実験による評価について述べる．最後に 5 節でまとめと今後の課題を述べる．

2. Fat-Btree とデータマイグレーション

2.1 Fat-Btree 構造

Fat-Btree は，B^+-Tree 全体をページ単位で PE 間で分配する並列 Btree の一種であり，データページである Btree の葉ページを各 PE に均等に分配する．ディレクトリ部分である Btree の葉ページ以外は，各 PE に配置されている葉ページへのアクセスパスを含むインデックスページにのみ再帰的に配置する．これにより，各 PE のディスクに格納されるのは，Btree のルートから均等に分配された葉ページまでの部分木である (図 1).

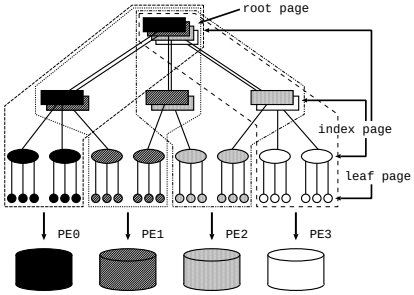

図 1 Fat-Btree の例

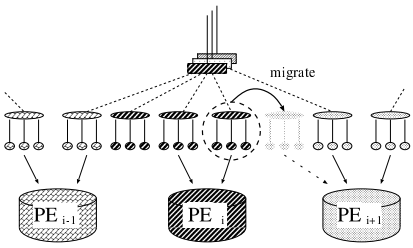

図 2 Fat-Btree におけるデータ移動の方法

更新頻度が高い下位のインデックスページほどそのコピーを持つ PE が減少していくため，ディレクトリ更新時に同期が必要となる PE 数が少なくなり，PE 間の局所的な通信によって更新処理を行うことができる．

また，各 PE では格納している葉ページの探索に必要のないインデックスページを持たないため，各 PE でインデックスページのキャッシュを行った場合にキャッシュのヒット率を高く保つことができる．高キャッシュヒット率により，更新処理だけでなく，探索処理も従来の並列 Btree 構造と比較して高速に行うことができる．

2.2 データマイグレーション

データマイグレーションは，並列データベースやストレージシステムにおいてアクセス偏り除去のための有効な手段である．分散配置された PE 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し，データ移動によりデータ再配置を行うことで，システム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．

2.2.1 Fat-Btree のデータ移動

Fat-Btree では，ある PE 内でインデックスが端である葉インデックスページおよび，その下のデータページをそのまま，インデックスにおいて論理的に隣の PE に移すことで，データを移動させることができる (図 2). この場合，実際にデータ移動に関わる PE は，隣接する 2PE だけで済むので，他の PE の処理を妨害することなくデータ移動を実現できる．

以下，ノードを移動するアルゴリズム[5)] の概要を述

表 1　ラッチマトリクス

mode	IS	IX	S	SIX	X
IS	○	○	○	○	×
IX	○	○	×	×	×
S	○	×	○	×	×
SIX	○	×	×	×	×
X	×	×	×	×	×

べる．この葉ページの移動の際，移動先の PE がその葉ページの上位のインデックスページを持たない場合，移動対象となる葉ページからルートページまでのアクセスパス上のインデックスページを，移動先の PE が持っていないページ分は移動元の PE からコピーする．逆に葉ページの移動によって移動元の PE から移動した葉ページの上位のインデックスページからその PE 内の下位のインデックスページのエントリーがなくなる場合，そのインデックスページを削除する．この操作でそれより上位のインデックスページからローカルページのエントリーがなくなった場合，再帰的に上位のインデックスページを削除する．

3. 並行性制御

木構造の一貫性を保証するために，Btree に並行性制御は必須である．通常，Btree や他のアクセスパスには，ロックの代わりにデッドロック検出機構を持たない高速かつ単純なラッチが用いられる．

3.1 ラッチモード

本稿では，5 種類のモードを持つラッチを仮定する．各モードは IS，IX，S，SIX，X であり，これらのモードの適合性は表 1 に示される[16]．表中の“ ○ ”は同時に複数のラッチが獲得可能なモードである．

あるインデックスページの複製が複数の PE に存在する場合を考える．もしラッチ処理が各 PE で分散して行われるならば，表 1 より，IS および IX モードはローカル PE のみで獲得するだけでよい．しかし，S，SIX および X モードは，コピーを持つすべての PE でラッチを獲得する必要がある．

3.2 INC-OPT 方式

Fat-Btree を含めた並列 Btree 向けの並行性制御方式として INC-OPT 方式は提案された[7),10)]．

INC-OPT の検索時のプロトコルは，IS モードによるラッチカップリングが使用される．まず，ルートページを IS モードでラッチした後，以下の処理を繰り返す．

(1) 親ページのキーを比較して，子ページのポインタを取得
(2) 子ページの IS ラッチを取り，親ページのラッチを解放

葉ページまで辿り着けば，葉ページに S ラッチを獲得しデータを読む．

INC-OPT での更新処理時のプロトコルは，二つのフェーズで行われる．

第 1 フェーズ: IX ラッチカップリングで葉ページまで辿る (葉ページは X ラッチが獲得される)．もし，SMO が起きないならば葉ページを更新して終了．もし葉ページがスプリットを起こし SMO が起こるならば，葉ページの X ラッチを解放し，第 2 フェーズに移行する．

第 2 フェーズ: 葉ページとその親ページを X モードでラッチする．もし親ページもスプリットするならば，同様にして X ラッチの範囲を拡大していく．十分な X ラッチが獲得されたならば更新操作を実行する．

この手続きの厳密な定義は[7),10)] を参照されたい．

INC-OPT は，更新処理のプロトコルからわかるように，SMO 発生時のリスタートが複数回必要になることがある．ルートページまで更新が及ぶ場合には，木の高さと同じフェーズ数が必要となる．このことは更新命令のレスポンスタイムを増加させるとともに，上位ページで複数回 X ラッチを獲得することで，システム全体のスループットも低下させる．

3.3 MARK-OPT 方式

MARK-OPT(MARKing OPTimistic) 方式[15)] は，並列 Btree 向けとして我々が提案した並行性制御手法である．従来の INC-OPT は，広範囲に SMO が発生するときに，X ラッチ拡大のために複数回のリスタートが必要だった．それに対し MARK-OPT は，SMO が発生する木の高さをマークすることにより，広範囲の SMO 発生時も，ほとんどの場合 1 回のリスタートで更新フェーズに移ることができるので，リスタート回数を少なく抑えることが可能である．よって，リスタート回数を少なく抑えることによるレスポンスタイムの向上と，中間の X ラッチ拡大フェーズの除去による不必要な X ラッチの獲得の減少から，高更新環境において従来手法より高いスループットを得ることが期待できる．MARK-OPT は INC-OPT の拡張であるため，検索処理時のプロトコルは同様であるが，更新処理時のプロトコルが異なる．

MARK-OPT での更新処理時のプロトコルは次のように行われる．

第 1 フェーズ: IX ラッチカップリングで葉ページまで辿る (葉ページは X ラッチが獲得される)．フルエントリー (次にエントリーが増えたときにスプリットする状態のこと) ではないインデックス

ページが存在したら，そのページのルートページからの高さをマークする．マークする高さは，フルエントリーではないページに遭遇するたびに逐次更新される．もし，SMO が起きないならば葉ページを更新して終了．もし葉ページがスプリットを起こし SMO が起こるならば，葉ページの X ラッチを解放し，第 2 フェーズに移行する．

第 2 フェーズ: 第 1 フェーズと同様に木の高さのマークは行う．前フェーズでマークした高さ以下のインデックスページと葉ページを X モードでラッチする．もし獲得した X ラッチの範囲が SMO の範囲に対して十分でなければ，同様にマークを用いて X ラッチの範囲を変化させていく．十分な X ラッチが獲得されたならば更新操作を実行する．

この手続きの厳密な定義は[15] を参照されたい．

MARK-OPT はマークした前フェーズの状態をもとにラッチ範囲を決定している．そのため，前フェーズからの木構造の変化による SMO 範囲の拡大により，複数回リスタートが必要となる場合もあるが，INC-OPT 同様に最大フェーズ数は高々木の高さ H に抑えられる．SMO 範囲が複数のレベルに渡って同時に拡大することは極めて稀であり，多くの場合は 1 回のリスタートで処理を完了できる．よって，上位ページまで SMO が及ぶときでも，INC-OPT のように多くのリスタートを必要としない．

4. 実　　験

アクセスパターンの偏りがある環境における MARK-OPT の有効性を示すために，分散ディレクトリとして Fat-Btree を採用している自律ディスク[17)~19)] に提案手法を実装し実験を行う．また，データ移動の並行性制御としての MARK-OPT の有効性を示すために，データ移動が行われている環境における実験も行う．

4.1 実 験 環 境

実験は，我々の提案する分散ストレージ技術である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．

4.1.1 初期 Fat-Btree の構築

初期 Fat-Btree として各 PE に葉ページを 1 ページずつ作成し，これらの葉ページのキーをその PE に格納されるキーの下限値とする．PE 番号の昇順に，初期葉ページのキーの値が大きくなるようにしておくことにより，以後の挿入処理では各 PE に格納される葉ページが静的に分割される．この初期 Fat-Btree に対してランダムな要素を繰り返し挿入したものを実験に用いる．今回の実験における Fat-Btree の構成パラメータはを表 3 に示す．

表 2 実験システムの構成

ノード	128 台 (Storage)，16 台 (Clients)
CPU	AMD Athlon XP-M1800+ (1.53GHz)
メモリ	PC2100 DDR SDRAM 1GB
ディスク	TOSHIBA MK3019GAX (30GB, 5400rpm, 2.5inch)
OS	Linux 2.4.20
Java VM	Sun J2SE SDK 1.4.2_04 Server VM

表 3 実験システムのパラメータ

ページサイズ	4KB
タプルサイズ	400B
インデックスページのキー数	最大 64
葉ページのタプル数	最大 8

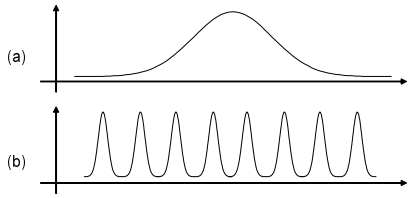

図 3 実験に用いたアクセスパターン:
(a) ピークが 1 つ (b) ピークが 8 つ

4.2 実 験 方 法

このような環境下の自律ディスク上に上記の手法で，約 400,000 タプルの初期 Fat-Btree を構築した後，各クライアント PC から同時にリクエストを送信する．

アクセス分布については zipf 分布[20)] に従った図 3 に示すような 2 つのパターンを用意した．それぞれのパターンは，ピーク値が (a)1 つの場合，(b)8 つの場合を想定している．また，今回は純粋にデータ移動の並行性制御としての提案手法の有効性を示すため，負荷評価に基づくデータ移動ではなく，クライアントからの連続的なリクエスト送信によってデータ移動を行う．

16 台のクライアント PC(1 台あたり並行して 8 スレッド) から検索と挿入のリクエストを送信し，10 秒間操作したときのスループットから性能を評価する．

以下に本実験の概要を述べる．まず，リクエストのアクセスパターンを変化させたときの各手法のスループットを測定し，従来手法と提案手法の比較を行う．次に，データ移動が実行されている環境でのスループットの測定を行う．

4.3 アクセス偏りを変化させたときの比較

図 4 および図 5 は，それぞれ図 3(a) および (b) のアクセスパターンに従い，更新比率 40%，横軸として

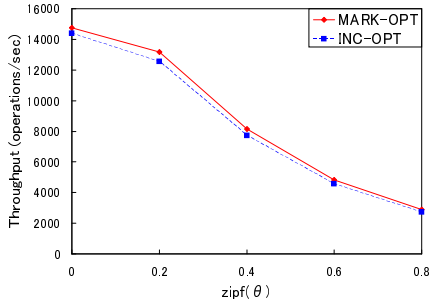

図 4 アクセス偏りのピークが 1 つのときの並行性制御方式の比較

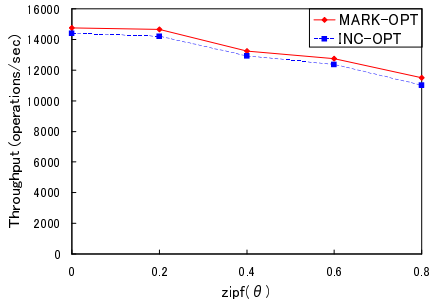

図 5 アクセス偏りのピークが 8 つときの並行性制御方式の比較

zipf 分布の母数を 0 から 0.8 に変化させ，スループットを縦軸として比較したグラフである．

zipf 分布の母数 θ が大きくなり，アクセス偏りが大きくなるに従い，ピーク値が 1 つの場合は，急激にシステム全体のスループットが低下する．これは，アクセスが集中している PE での処理が限界に達しているためである．これに対し，ピーク値が 8 つの場合は，アクセスが 8 つの値域に分散されているため，急激なシステム全体のスループットの低下は起こらない．

このように，システム全体のスループットは低下しているのにも関わらず，更新要求増加時[15] と異なり，MARK-OPT の INC-OPT に対する改善率の上昇がわずかである．Fat-Btree は全体として 1 つの Btree を形成しており，アクセス偏りは各 PE ごとの部分木の形態には影響するものの，木構造全体の形態には影響しない．更新要求による SMO 発生は，部分木ではなく，木構造全体の状態によるため，アクセス集中は SMO の発生率にほとんど影響しない．アクセス集中による MARK-OPT の改善率上昇がわずかであったのは，これらの理由からである．

しかし，どのようなアクセスパターンにおいても，MARK-OPT はスループットを改善しており，MARK-OPT はアクセス偏りがある環境においても有効であると言える．

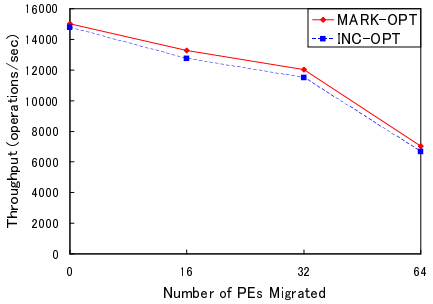

図 6 データ移動の頻度を変化させたときの並行性制御方式の比較 (アクセス偏り無)

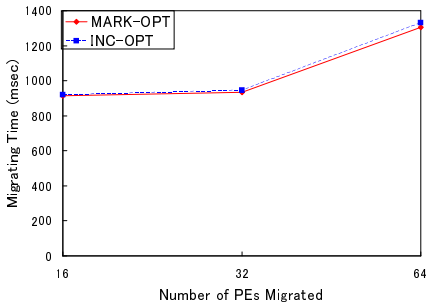

図 7 データ移動にかかるによる並行性制御方式の比較 (アクセス偏り無)

4.4 データ移動の頻度を変化させたときの比較

データ移動の頻度は，データ移動が行われる値域分割のパーティション数により変化させる．そのパーティションを挟むディスク間で，100 個の葉ノードを移動しては戻すことを繰り返すように，クライアントからリクエストを送信する．

4.4.1 アクセス偏りがないときの比較

図 6 は，均一なアクセスパターンに従い，それぞれ更新比率 0%および 20%，横軸としてデータ移動を行うパーティション数を 0 から 64 に変化させ，スループットを縦軸として比較したグラフである．また，図 7 は，各 PE ごとに 100 個の葉ノードを移動するのにかかる時間を縦軸として比較したグラフである．

INC-OPT と比較して，MARK-OPT は常に高いスループットが得られているが，データ移動の増加による改善率上昇はわずかである．これは，データ移動による影響はほとんどの場合，パーティション付近のみにとどまり，アクセス負荷が均一であると，リクエストのスループットへの影響がわずかなためである．また，データ移動にかかる時間もわずかであるが改善している．これは，MARK-OPT がマークにより，SMO 発生時のリスタート回数を減少させているためである．

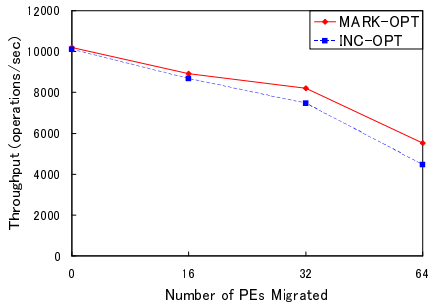

図 8 データ移動の頻度を変化させたときの並行性制御方式の比較 (アクセス偏り有)

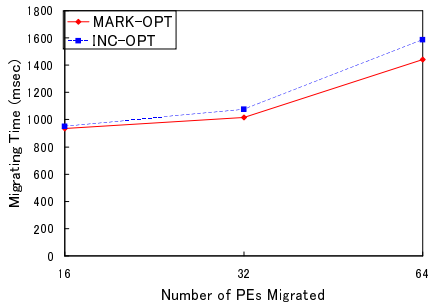

図 9 データ移動にかかる時間による並行性制御方式の比較 (アクセス偏り有)

4.4.2 アクセス偏りがあるときの比較

図 8 は，図 3(b)，zipf(0.6) のアクセスパターンに従い，更新比率 20%，横軸としてデータ移動を行うパーティション数を 0 から 64 に変化させ，スループットを縦軸として比較したグラフである．また，図 9 は，各 PE ごとに 100 個の葉ノードを移動するのにかかる時間を縦軸として比較したグラフである．

データ移動の頻度が大きくなるに従い，INC-OPT のスループットが低下していくのに対し，MARK-OPT も徐々に低下するものの，INC-OPT よりは高いスループットを維持している．また，データ移動の頻度の増加によるデータ移動にかかる時間の増加も MARK-OPT は抑えている．アクセスパターンに偏りがない場合と比較して，その差が顕著になったのは，アクセスが集中しているディスクでデータ移動が発生することにより，X ラッチ衝突が多く起きるためである．

4.4.3 データ移動の並行性制御としての評価

アクセス偏りの有無およびデータ移動の頻度によらず，INC-OPT と比較して，MARK-OPT は高いスループットが得られる．さらに，データ移動にかかる時間自体も MARK-OPT の方が小さく，すばやく負荷を分散することができることを意味する．このことより，MARK-OPT はデータマイグレーションにおけるデータ移動の並行性制御として有効であると言える．

5. まとめと今後の課題

本稿では，従来手法との比較により，我々の提案してきた並列 Btree の並行性制御手法に対する MARK-OPT のアクセスパターンの偏りに影響の考察を行った．また，それによって発生するデータ移動の MARK-OPT 対する影響の考察を行った．

Fat-Btree の全体として 1 つの Btree を形成するといる特性から，アクセス集中による SMO の増加はわずかである．そのため，アクセス集中による改善率の増加はわずかであったが，どのようなアクセスパターンにおいても，MARK-OPT は常にシステムスループットを改善しており，MARK-OPT がアクセス偏りがある環境においても有効であることを示した．

また，MARK-OPT がデータ移動の頻度の増加に伴うシステムスループットの低下を抑えることを示した．さらに，データ移動の頻度の増加に伴うデータ移動時間の増加も抑えることから，MARK-OPT がデータマイグレーションにおけるデータ移動の並行性制御として有効であることを示した．

本稿ではクライアントからリクエストを送信することにより，データ移動を発生させたが，本来は負荷値により負荷偏りを検出し，それを基にデータ移動を行うべきである．自律ディスクには，それらの操作を自律的に行う機能を備えており，その機能を用いて実験を行う必要がある．

また，MARK-OPT のリカバリーについても検討しなければならない．

さらに，優れた並行性制御を実現できることが知られている B-link[21),22)] の Fat-Btree への適用を検討している．B-link は，サイドポインタにより隣のインデックスノードにリンクをもっている．サイドポインタがあることにより，ラッチカップリングを用いず，単一ノードラッチによる並行性制御を行うことができる．また，リスタートを行うことはなく，すべての処理が 1 フェーズで行われる．B-link の Fat-Btree への適用方法を検討し，適用したものと従来手法との比較が必要である．

謝辞 Fat-Btree の並行性制御に関して奈良先端科学技術大学の宮崎純助教授に助言を頂いた．ここに感謝の意を表します．また本研究の一部は，独立行政法人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．

参 考 文 献

1) Copeland, G., Alexander, W., Boughter, E. and Keller, T.: Data Placement in Bubba, *Proc. of ACM SIGMOD conf. '88*, pp.99–108 (1988).
2) Ghandeharizadeh, S. and DeWitt, D. J.: HybridRange Partitioning Strategy: A New Declustering Strategy for Multiprocessor Database Machines, *Proc. of VLDB Conf. '90* (1990).
3) DeWitt, D. and Gray, J.: Parallel Database Systems: The Future of High Performance Database Systems, *Communications of the ACM*, Vol.35, No.6, pp.85–98 (1992).
4) 金政泰彦,宮崎純, 横田治夫：並列データベースシステムにおける更新を考慮したディレクトリ構成，信学技報，DE97-77(AI97-44)，電子情報通信学会，pp.63–68 (1997).
5) Yokota, H., Kanemasa, Y. and Miyazaki, J.: Fat-Btree: An Update-Conscious Parallel Directory Structure, *Proc. of IEEE ICDE Conf. '99*, pp.448–457 (1999).
6) 宮崎純, 横田治夫：無共有並列計算機向けディレクトリ構造 Fat-Btree の実装とその評価，情処学会研究会報告，データベースシステム DBS-119-68，情報処理学会，pp.407–412 (1999).
7) 宮崎純, 横田治夫:並列ディレクトリ構造 Fat-Btree の並行性制御とその評価，情処学会研究会報告，データベースシステム DBS-124-69，情報処理学会，pp.37–44 (2001).
8) 宮崎純, 横田治夫:並列ディレクトリ Fat-Btree のリカバリについて，信学技報，DE2001-107，電子情報通信学会，pp.17–24 (2001).
9) 宮崎純，横田治夫：高信頼 Fat-Btree 構成への neighbor-WAL プロトコルの適用，第 13 回データ工学ワークショップ論文集，DEWS2002，C1-2，電子情報通信学会 (2002).
10) Miyazaki, J. and Yokota, H.: Concurrency Control and Performance Evaluation of Parallel B-tree Structures, *IEICE Transactions on Information and Systems*, Vol.E85-D, No.8, pp.1269–1283 (2002).
11) 鈴木裕通, 横田治夫：並列ディレクトリ構造 Fat-Btree における負荷分散の手法とその実装，第 11 回データ工学ワークショップ論文集，DEWS2000，4B-4，電子情報通信学会 (2000).
12) 風戸広史, 横田治夫：並列ディレクトリ構造 Fat-Btree におけるレンジ問い合わせの取り扱い，第 12 回データ工学ワークショップ論文集，DEWS2001，7A-7，電子情報通信学会 (2001).
13) Bayer, R. and Schkolnick, M.: Concurrency of Operations on B-trees, *Acta Informatica*, Vol.9, No.1, pp.1–21 (1977).
14) Mohan, C. and Levine, F.: ARIES/IM: an Efficient and High Concurrency Index Management Method Using Write-Ahead Logging, *Proc. of ACM SIGMOD Conf. '92*, pp.371–381 (1992).
15) 吉原朋宏,小林大,田口亮,上原年博, 横田治夫：並列ディレクトリ構造 Fat-Btree の並行性制御の改善とその評価，第 16 回データ工学ワークショップ論文集，DEWS2005，2A-o4，電子情報通信学会 (2005).
16) Gray, J. and Reuter, A.: *Transaction Proceeding: Concepts and Techniques*, Morgan Kaufmann, San Francisco (1993).
17) Yokota, H.: Autonomous Disks for Advanced Database Applications, *Proc. of International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pp.441–448 (1999).
18) 風戸広史, 横田治夫：自律ディスクへの Fat-Btree の実装，情処学会研究会報告，データベースシステム DBS-125-71，情報処理学会，pp.45–52 (2001).
19) 伊藤大輔,風戸広史, 横田治夫：負荷分散機構と組み合わせた自律ディスクのクラスタ再構築，信学技報，FTS2001-20，電子情報通信学会，pp.119–126 (2001).
20) Zipf, G. K.: *Human Behavior and the Principle of Least-Effort*, Addison-Wesley, Cambridge, MA (1949).
21) Lehman, P. L. and Yao, S. B.: Efficient Locking for Concurrent Operations on B-trees, *ACM Trans. Database Syst.*, Vol.6, No.4, pp.650–670 (1981).
22) Lanin, V. and Shasha, D.: A Symmetric Concurrent B-tree Algorithm, *Proc. FJCC*, pp.380–389 (1986).