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

2005－DBS－135　(5)
2005／1／20

索引分散管理システムにおける再編成処理の実装

田村　弘行†　　樋口　健‡　　都司　達夫‡

本研究では，複合オブジェクトに対する索引分散管理システムにおいて，オンライン再編成処理の実装を行う．索引分散管理システムは，オブジェクト間の参照関係をマルチインデックス手法で索引化し，その索引を分割，非共有メモリ型並列計算上に分散配置し，検索パス式を用いて検索処理を行う．このシステム上で並列に処理を行うことで応答時間の短縮が見込めるが，検索要求の傾向の変化や，大量の更新要求が行われると，システムが非効率的状態に陥り，その後の応答時間が悪化することもある．この効率の低下を改善するためには索引を再分割，つまり再編成が必要となる．そこで，本研究では索引分散システムにおけるオンライン再編成を実装し，実験を行うことで我々の提案する再編成手法の有効性を示す．

An Implementation of Reorganization in Distributed Index System

HIROYUKI TAMURA†　and KEN HIGUCHI‡　and TATSUO TSUJI‡

In this paper, we implement the on-line reorganization in distributed index system. In our Distributed Index System, an index of references between objects is constructed using multi-index technique. The index is divided and managed on a shared-nothing parallel computer, and our system retrieves the final result along the specified retrieval path expression. By processing queries in parallel, good response time would be expected in this system. However, if the tendency of retrieval queries changes or a large amount of index modification queries occurs, it would be in an inefficient condition and response time would be often degraded. In order to overcome such a situation, index reorganization is needed. We implement on-line reorganization to the distributed index system and it is shown in experimenting that our reorganization technique is effective.

1. はじめに

現在，データベースにおいては，非常に大規模なデータに対し，様々な高度な問い合わせを処理する必要が高まっている．我々はその１つの解決法として，索引を分割し，並列計算機上に分散配置し，並列処理することにより高速化を目指す研究を行ってきた[9), 10), 11), 12)]．しかし，これら索引分散管理システムでは，システムを稼動し続けることにより，データや索引が何度も更新されたりするなど，非効率な状態に陥ることが予想され，このようなデータや索引に対しては，再分割，つまり再編成することが必要となる．しかし，再編成処理とは索引を大きく更新する処理であり，非常に高コストである．そのため，他の処理への影響を少なくすることが求められ，オンライン再編成が望ましい．データベースに対するオンライン再編成に関しては，様々な研究がなされている[2), 3), 4), 5), 6), 7), 8)]．また，索引分散管理システムにおいてもオンライン再編成手法が提案されている[12)]．

†福井大学工学研究科
Graduate School of Engineering,.Fukui　University
‡福井大学工学部
Faculty of Engineering, Fukui University

本研究では，12)で提案されたオンライン再編成法を索引分散管理システムに実装し，実際の並列計算機上での有効性の検証を目的とする．

なお，本研究では，索引の状態が非効率的に陥り再編成が必要になったとき，他の処理への影響をより少なく抑えながら再編成を行う処理方法について考えるもので，再編成要求が必要になるタイミングや効率向上のための最適な分割方法を検知する方法については考慮しない．

2． 対象オブジェクトと索引

以下では，対象とする複合オブジェクトおよび索引，検索の定義を行う．

2.1 複合オジェクト

複合オブジェクトの検索パス式を以下のように定義する．

$$P = C_1\ A_1\ A_2 \cdots A_N$$

N をこのパスの長さと定義する．ここで，A_1 はクラス C_1 の属性，A_j はクラス C_j の属性であり，$1 \leqq j < N$ ならばその参照の定義域は C_{j+1} とする．ここで，各 A_j は単一オブジェクトとは限らず，オブジェクト集合も許す．つまり，C_j が属性 A_j の属性値として複数のオブジェクトを参照することが可能である．また，検索パス式Pの値を以下のように定義する．

$$o_1\ o_2 \cdots o_{N+1}$$

ここで，$o_1, o_2, \cdots, o_N$ はそれぞれクラス $C_1, C_2, \cdots, C_N$ のインスタンスであり，o_j（$1 < j \leqq N$）はオブジェクト o_{j-1} の属性 A_j の値である．また o_{N+1} はオブジェクト o_N の属性 A_N の値であり，単純値またはオブジェクト識別子（以下，OID）である．P の値の集合を O_P と表記する．

なお，本研究では，検索パス式を１つに限定し，クラス C_j（$1 < j \leqq N$）はそれぞれ異なるクラスとし，任意の異なる二つのクラスのインスタンス集合は共通部分を持たないものとする．

2.2 マルチインデックス

9)，10)，11)，12)における索引分散管理システムでは，複合オブジェクトに対する索引として，索引要素更新のオーバーヘッドが他の方式に比べて小さく，マシン間で部分的に索引要素を移動することが容易なマルチインデックス方式 [1)] を採用している．

複合オブジェクトに対するマルチインデックスとは，オブジェクトの直線的な参照関係の逆関係をそのまま参照関係にすることであり，検索は，この逆参照関係を順にたどっていくことで結果を得る．

オブジェクト o_j の属性 A_j の値が o_{j+1} であるとき，以下を P の索引要素と定義する．

$$< o_j , o_{j+1} >$$

ここで，o_{j+1} はキー値，o_j はデータ値である．さらに，検索パス式Pの索引 IP は，すべての索引要素の集合である．

さらに，クラス C_i のインスタンスの OID をキー値とする索引要素集合を IP_i とし，A_N の値をキー値とする索引要素集合を IP_N とする．

マルチインデックス方式による検索要求“ 検索検索パス式 P が $P = C_1\ A_1\ A_2 \cdots A_N$ のとき，属性 A_N の値が o_{N+1} であるようなオブジェクトを検索検索パス式 P 上で参照しているクラス C_1 のインスタンスを求めよ ” は，

$$o_1 o_2 \ldots o_N o_{N+1} \in O_P$$

となる C_1 のインスタンス o_1 の集合を求めることであり，IP を用いて

$$< o_N , o_{N+1} >, < o_{N-1} , o_N >, \ldots, < o_1 , o_2 >$$

のようにオブジェクトの被参照関係から求めることである．ここで，検索パス式 P 上で要素 o に対する検索結果のオブジェクト集合を $R_P(o)$ と表す．また，検索パス式 P と A_N の値の集合 S に対して以下のように定義する．

$$R_P(S) = \cup_{o \in S} R_P(o)$$

検索パス式を用いた検索においては $R_P(o)$ が一般の検索における単純値検索にあたり，$R_P(S)$ が範囲検索に対応する．本研究における検索とは集

合 S に対して範囲検索である $R_P(S)$を求めることである.

3. 索引分散管理システム

索引分散管理システムについての概要を述べる（詳細に関しては 11）を参照）. 本研究における, 索引分散管理システムは索引を管理するシステムであり, オブジェクトデータ本体を管理するシステムについては特定しない.

3.1 システムの基本構成

システムを構築する環境として, メッセージ通信非共有型メモリ並列計算機を仮定する. 各プロセッサエレメント（PE）での処理は 1 プロセスの逐次処理とする. このような並列処理環境上において索引に対する処理を並列に行うために, 複合オブジェクトに対する索引を分割し, 分割された索引をそれぞれの PE に分散配置し, 並列処理を行う.

本システムでの PE は, 検索 PE, HOST, DETECTOR の 3 種類に役割が別れている. 以下にそれぞれの処理の概要を述べる.

3.1.1 検索 PE

実際に索引を格納し, ユーザからの検索, 更新要求を処理する PE である.

3.1.2 HOST

外部からの検索・更新処理要求を, 検索 PE や DETECTOR に送信し, その要求処理の結果の集計し, 最終的な終了判定を行う PE である. HOST は唯一つだけ存在し, 外部の処理要求に対して要求識別子（RID）を付加する. RID には自然数を用い, 小さい順に処理要求に付加する.

3.1.3 DETECTOR

HOST と検索 PE の処理のみでは, 各要求に対する処理が終了したか否かを判定することはできない. そこで, DETECTOR（以下, DE）と呼ばれる要求の終了判定を行う PE を用意する. DE は IP_i ごとに一つずつ用意され, 検索 PE より送られるメッセージ受信数・送信数を集計し, HOST・DE より送られるメッセージ受信予定数を用いて, RID ごとに IP_i の処理の終了判定を行っている. これは, 検索パス式の検索処理の方向が一定であり, IP_i への検索要求は IP_{i+1} の処理結果として送信されたメッセージのみであることを利用している. つまり, IP_i の終了は, 検索結果として IP_{i+1} から IP_i へ送られるメッセージの総数と, IP_{i+1} が終了した際送られる IP_i へのメッセージ受信予定数（IP_i へ送信したメッセージの総数）が等しいときのみである. ただし, i=N+1 のとき, “ IP_{i+1} に関する処理 ”は HOST の検索要求の送信処理であり, “ IP_{i+1} の検索結果として送信されたメッセージ ”は HOST からの検索要求のメッセージである. また, i=1 のとき, “ IP_i への検索要求のメッセージ ”は, HOST への検索結果である. ここで, IP_i に関する終了判定を行う DE を DE_i で表すこととする.

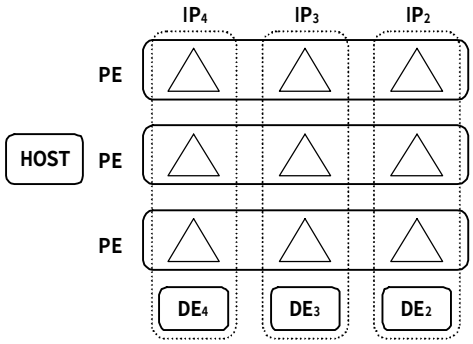

図 1 索引分散管理システムの各 PE

4. オンライン再編成

索引分散管理システムにおいては処理要求により最適な分割は異なる. つまり, 処理要求によっては, 処理能力が低下することがある. したがって, ユーザからの処理要求の傾向に基づいて索引を分割し直し, 索引を最適な状態に再編成することで, 索引の非効率化による処理能力の低下を解消することができる.

ただし, 更新と同様に, 再編成中に起こる索引

の移動時の排他ロックも考える必要がある. 最も安全な方法として，HOST で再編成要求以降の要求をすべて保留し, 再編成要求以前の処理要求が全て終わり次第再編成処理を行う, つまりオフラインで行う方法が考えられる. しかし, 索引を再編成することは高コストであり, この方法で再編成処理を行っている間は, 他の要求が一切処理されないため，他の処理への影響が大きい.

そのため，12)では索引分散管理システムにおけるオンライン再編成手法が提案され, シミュレーションによる検証が行われている. 本研究においては 12)の手法を採用し，その実装，改良を行う.

4.1 オンライン再編成

オンライン再編成手法を記述する（詳細に関しては 12）を参照).

12)の手法では以下の事項を仮定する.

- 索引の分割にハッシュ関数を用い，各 PE は分割決定に用いたハッシュ関数を保持し，それを用いてメッセージ送信先を決定する.
- 再編成要求は，分割決定に用いたすべての PE のハッシュ関数とそれに伴う索引要素の移動を変更要求とする.
- 各 DE は, 索引要素移動用管理区画と呼ばれるＰＥ集合を管理する.

なお, ハッシュ関数はすでに決定されているものとし, 最適なハッシュ関数の選択・決定方法については考慮しない.

上記の前提において再編成処理を行う. 再編成は以下の３つのフェーズからなる

- 索引に対するロックの取得
- 索引要素の移動
- ロックの開放およびハッシュ関数の変更

ここで，ロックの粒度が問題となる．12)では移動対象となる索引要素が IP_i に含まれる場合は, IP_{i+1} に属する索引要素を使用する処理要求をすべて保留状態とする方式を採用している．また, 移動対象の属する IP_i においては，以下のタイミングで索引要素移動を行う.

- 再編成要求以前に送られた要求をすべて処理後，移動開始する.
- 再編成要求以降の要求を処理する前に, 索引の移動を完了させる.

このタイミングは DE の終了判定情報を用いて決定可能である. しかし, 移動の対象が索引全体に及ぶような大規模な再編成では, オフライン再編成と大差なくなってしまう. そこで処理を IP_i 単位で分割し, 連続的に処理することで, 再編成処理と他の処理との並列性を高めることができる.

上記オンライン再編成の手順は以下のようになる.

（１）ロックの取得

HOST が再編成要求を受けると，RID を付加後，IP_i ごとに再編成要求を分割し，それらを DE_{N+1} に送る. HOST は，IP_{N+1} に対する再編成が終了するまで, この再編成要求以降の要求の送信を保留する.

以下の記述においては特に断りのない場合は IP_j に対する再編成処理として記すこととする.

DE_i におけるの再編要求に対するロック処理は以下の手順で行われる

(a) 管理する全ての PE にロック要求を送る

(b-1) $i>2$ ならば DE_{i-1} に再編成要求を送る

(b-2) $i=2$ ならば DE_j にロック完了通知を行う

ロック要求を受け取った検索 PE においては再編成要求の RID より大きい RID で，かつ，移動対象索引に関するもののみ保留状態にする.

（２）索引要素の移動

ロック完了通知を受け取った DE_j は，再編成要求の RID より小さな RID の処理要求が IP_j において全て終了後, 全 DE に対象索引の再編成開始通知を送る.

再編成開始通知を受け取った各 DE は, 自分の

管理する PE 全てに索引要素移動許可を与える．索引要素移動許可を受けた検索 PE は索引の移動を行い，送受信情報を自分の管理を担当する DE に送る．各 DE はそれらを用いて索引要素移動の終了判定を行う．

（3）ロックの解除とハッシュ関数の変更

各 DE は自分の管理する PE において全ての移動が終了したならば，DE_j に対して索引要素移動終了通知を送る．DE_j は全ての DE から索引要素移動終了通知を受けたならば，全 DE にロック解除通知を送り，HOST に再編成処理終了通知を送る．ロック解除通知を受けた DE は管理 PE 全てに対しロック解除通知を送り，ロック解除通知を受けた検索 PE はロックを解除する．また，同時にハッシュ関数を変更する．

4.2 更新の粒度を考慮した再編成

4.1 節においてオンライン再編成の処理の概要を述べた．しかし，この手法をそのまま実装した場合，再編成許可が出た後の，索引要素移動のための索引要素の削除・挿入が一度に行われてしまう．大量の索引要素を移動する再編成処理においては，再編成に関する索引要素の削除，挿入にかかる時間は非常に大きくなり，その間，他の処理は完全にストップしてしまう．また，再編成処理を IP_i ごと分割したため，先に開始した再編成処理により，その他の処理要求に対する処理がストップし，結果的に残りの再編成処理の開始が遅れる可能性もある．

そこで，再編成を行う際，一度に索引を更新する方法とは別に，再編成中の索引要素の削除，挿入処理を細かい単位に分け，他の要求に対する処理の優先度を高めることにより，他の処理への影響をより抑える方法をとる．

ここで，挿入，削除をまとめて行う索引要素数を粒度とする．つまり，粒度 i で再編成処理を行う場合は，再編成処理における i 個の索引要素の削除または挿入を行うたびに再編成処理以外の処理が可能な要求に対する処理が可能か否かを探査し，もし可能なら再編成処理以外の処理を優先して行う．粒度を無限大とする場合は，従来のシステムである．

この粒度を考慮したアルゴリズムは，前述のアルゴリズムに以下の処理を加える．

［対象索引要素の削除の場合］

検索 PE は索引要素移動許可通知を受信すると，移動対象となる索引要素の削除を開始する．ここで，削除した索引数を保持するカウンタを用い，その値が規定の値になるまで削除を行う．カウンタが規定の値（粒度）になると別の処理に実行権を移し，再編成以外の処理可能な要求がなくなったならば，カウンタを 0 に戻し，索引要素の削除を再開する．すべての索引要素が削除し終わるまでこれを繰り返す．

［対象索引要素の挿入の場合］

再編成に対する索引要素の挿入要求を受信すると，受信した索引要素の挿入を開始する．挿入した索引要素数を保持するカウンタを用意し，その値が規定の値（粒度）になるまで挿入を行う．規定の値に達すると別の処理に実行権を移し，再編成以外の処理可能な要求がなくなったならば，カウンタを 0 に戻し，索引要素の挿入を再開する．すべての索引要素の挿入が終わるまでこれを繰り返す．

5. 実験

提案した再編成アルゴリズムを索引分散管理システム上に実装し，実験を行うことにより，本システムの有効性の検証を行う．

5.1 実験条件

実験に用いる索引を以下のものとする．

- 索引の対象となる複合オブジェクトは N=5 で行う．各クラスのインスタンス数は 10000 である．
- 各オブジェクトは 1 つまたは 2 つの参照を持つ．参照数および参照先オブジェクトはランダ

ムに決定し，2 つの場合は異なるオブジェクトを参照する．

- 分割に用いるハッシュ関数は以下のものを使用する．

  Type1：OID mod 7

  Type2：(OID/2) mod 7

ここで，OID は索引要素のキー値のオブジェクト ID である．

これらをマルチインデックス手法で索引化し，それを 3 セット用意して，それぞれに対し実験を行う．これらの索引を，索引１，索引 2，索引 3 とする．

実験対象として，オフライン再編成，オンライン再編成，粒度を考慮したオンライン再編成の 3 つを用いる．粒度を考慮したオンラインシステムは，粒度が N=100，50，25，10，5，1 の場合をそれぞれ測定する．

実装を行う並列計算機は以下のものを用いた．

【SUN Enterprise 5500】

- CPU：UltraSPARCⅡ×14
- クロック速度：400MHz
- 主記憶容量：５GB
- 通信には MPI を用いる

実験条件を以下に挙げる．

- 総 PE 数は１４とする．その内訳として，HOST は１PE，検索 PE は７PE，DE が５PE である．
- １回の実験においては９個の連続的な処理要求に対して処理を行う．５個目の要求を再編成処理とし，その他の要求は検索要求とする．
- 再編成要求は索引の初期分割により，Type1 から Type2 へハッシュ関数に変更する再編成か，Type2 から Type1 へハッシュ関数に変更する再編成かのいずれかのものとする．
- 各検索要求はランダムな 1000 個のキー値に対する要求である．

以上の条件を用いて，各索引に対し 10 回シミュレーションを行い，その層処理時間の平均を結果とする．

再編成により Type１から Type2 へのハッシュ関数変更およびその逆の変更は，索引の全体の 6／７の索引要素が移動する．この大規模な再編成において粒度によって総処理時間がいかに変化するかを実験により検証する．

5.2 実験結果

Type１から Type2 へハッシュ関数を変更するシミュレーションの測定結果を図 2，図 3，図 4 に示す．

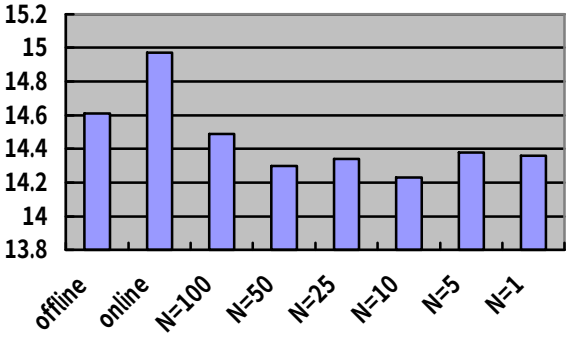

図 2 索引１の Type1 から Type２への再編成

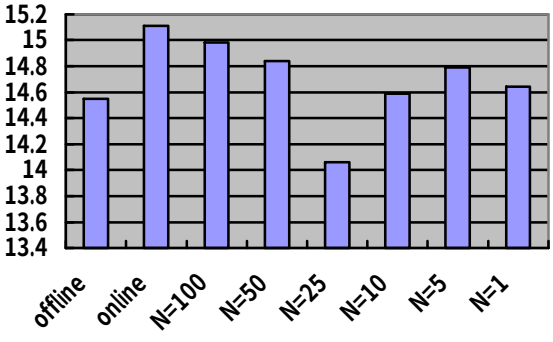

図 3 索引２の Type１から Type２への再編成

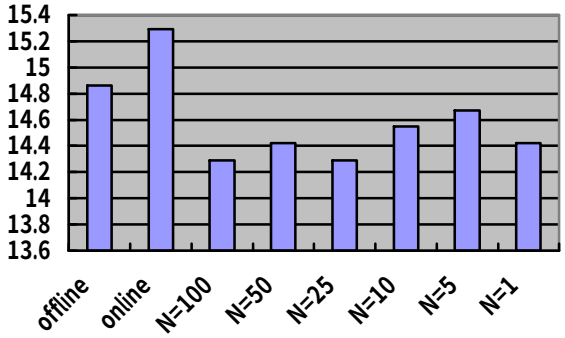

図 4　索引３の Type１から Type２への再編成

Type2 から Type1 へハッシュ関数を変更するシミュレーションの測定結果を図 5，図 6，図 7 に示す.

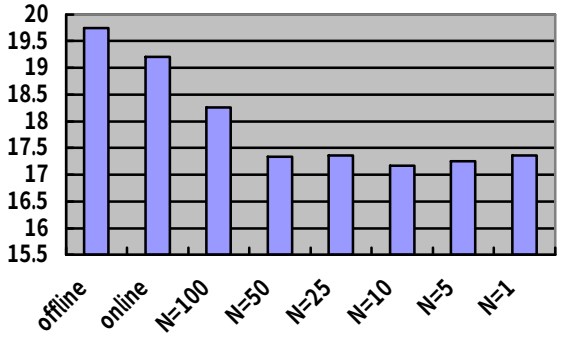

図 5　索引１の Type２から Type１への再編成

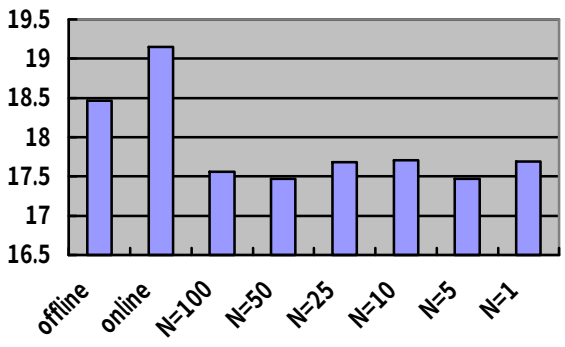

図 6　索引 2 の Type1 から Type1 への再編成

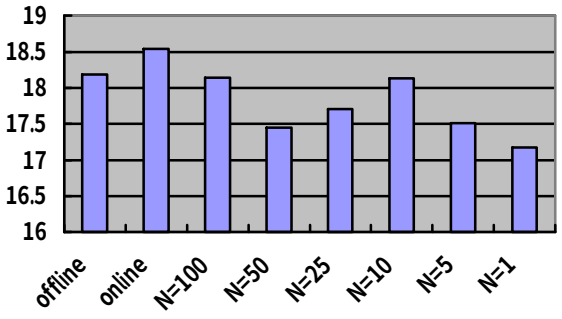

図 7　索引３の Type2 から Type1 への再編成

5.3 考察

実験を行った結果, オンライン再編成はオフライン再編成より総処理時間が長いことがわかる. これは, 先に開始された再編成処理により, 他の検索要求が保留となり, それによって他の部分の再編成処理の開始が遅れ, 最終的には全体の処理が遅れてしまうことによるものと推察される. また, 実験結果より, 粒度を考慮したオンライン再編成がもっとも高速である. これは, 再編成処理中も他の処理が可能か否かの問い合わせが可能となることで, 前述の再編成処理の分割による弊害を解消されたことによるものである.

実験結果より, 粒度を下げることで総処理時間が減少する傾向にあることがわかる. しかし, 粒度を５０以下にした場合はは増減は不安定になっている. これは, 他の処理が可能か否かの問い合わせの処理の回数が増加することによる処理時間の増加の影響と思われる.

以上のことから, 粒度を考慮したオンライン再編成手法が有効であることは明らかである.

6. まとめ

複合オブジェクトに対する索引分散管理システムにおいて, 処理の粒度を考慮したオンライン再編成処理の実装を行い, 実験による検証を行った. その結果, 粒度を考慮したオンライン再編成手法がもっとも高速であることが示され, 有効性が実証された.

しかし,最適となる粒度は対象索引やハッシュ関数により異なり,最適な粒度の決定法の確立が今後の課題として残る.

参考文献

1) Bertino, E.: A survey of indexing techniques for object-oriented database systems, Query Processing for Advanced Database , Freytag, J.C., Maier,D and Vossen, G.(Eds.), Morgan Kaufmam, pp.383--418, (1995).

2) Achyutuni, K., Omiecinski, E. and Navathe, S.: Two techniques for on-line index modification in shared nothing parallel database, Proc. 1996 ACM SIGMOD International Conference on Management of Data, pp. 124-136,(1996).

3) Zou, C. and Salzberg, B.: On-line Reorganization of Sparsely-populated B+trees, Proc. 1996 ACM SIGMOD International Conference on Management of Data, pp. 115-124, (1996).

4) Sockut,G. H., Beavin, T. A. and Chang, C.: A Method for On-Line Reorganization of a Database, IBM System Journal, vol 36, no. 3, pp. 411-436, (1997).

5) Zou, C. and Salzberg, B.: Safely and Efficiently Updating References During On-line Reorganization, Proc. 24th International Conference on Very Large Data Bases, pp. 512-522, (1998.)

6) 樋口 健, 小倉 一泰, 都司 達夫, 宝珍 輝尚: 複合オブジェクトに対する索引の分割を決定する確率アルゴリズムの実験的評価,信学論, vol.J82-D-I, no.1, pp.14--23, (1999).

7) Lakhamraju, M. K., Rastogi, R., Seshari, S. and Sudarshan, S.: On-line Reorganization in Object databases, Proc. the 2000 ACM SIGMOD International Conference on Management of Data, pp.58-69, (2000).

8) Feelifl, H., Kitsuregawa, M., and Ooi, B.: A Fast Convergence Technique for Online Heat-Balancing of Btree Indexed Database over Shared-Nothing Parallel Systems, 11th International Conference of Database and Expert Systems Applications,pp.846--858, (2000).

9)中野究,「通信コスト軽減を目的とした複合オブジェクト索引の分散管理システムの改良」, 平成13年度福井大学大学院工学研究科修士論文

10)樋口健 都司達夫 宝珍輝尚,「複合オブジェクトに対する索引のオンライン更新が可能な分散管理システム」, 情報処理学会論文誌：データベース Vol.43, No.SIG12(TOD16), pp.64-79(2002)

11)瀧澤淳,「複数の検索パス式に対応した複合オブジェクト索引の分散管理システム」, 平成16年度福井大学大学院工学研究科修士論文

12)樋口健 都司達夫,「複合オブジェクトに対する索引分散管理システムにおけるオンライン再編成法」,情報処理学会論文誌：データベース Vol.45, No.SIG10(TOD23), pp.1-17(2004)