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

樋口 健[†] 都司達夫[†]

本論文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けるオンライン再編成法の提案を行う．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，その索引を非共有メモリ型並列計算機上で分割管理し，検索パス式に基づく問合せ処理を行う．この際，複数の要求を並列処理することで応答時間の短縮が見込まれる．しかし，検索要求の傾向の変化や大量の索引の更新が起こると，その後の処理の応答時間が悪化することがある．これはオブジェクト間の参照関係集合や検索要求により最適な索引分割が異なるためである．このような事態に対処するには索引の分割を再編成する必要がある．我々の再編成手法は索引要素の更新処理を基本としたオンライン再編成手法であり，他の通常の索引処理と並列に実行される．シミュレーションにより我々の手法の再編成処理が並列実行中の処理への影響をおさえつつ実行可能であることを示す．

On-line Reorganization for Distributed Index System for Complex Objects

Ken Higuchi[†] and Tatsuo Tsuji[†]

This paper concerns an on-line reorganization scheme for distributed index system for complex objects. In our index system, an index of references between objects is constructed using the 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. However, if the tendency of retrieval queries changes or a large amount of index modification queries occurs, response time would be often degraded. This degradation is caused by the fact that according to the set of references between objects and a retrieval query, the optimal index partitioning differs. In order to overcome such a situation, index reorganization is needed. Our index reorganization scheme is based on the index modification and performed on-line together with other ordinary index operations such as retrieving and modification. The simulation result proves that our reorganization scheme can be performed parallelly with these ordinary index operations, keeping the influence on such index operations being minimal.

1. はじめに

データベースが非常に大規模で，種々の高度な問題を扱い，停止してメンテナンスすることが難しくなっている現在では，様々な要求をオンラインで処理することが求められている．また，システムを稼働し続けていくことにより，データや索引が非効率的な状態になる可能性もあり，これらを解消するためのシステムの再編成も重要な問題である．

文献 13)，18) ではメッセージ通信非共有メモリ型並列計算機上で，複合オブジェクト集合に対する大容量索引をクラスタリングして分散配置し，並列操作を行うことにより処理効率を向上させる手法が提案されている．この方式の利点はデータ本体と索引を分離可能であり，索引上の処理がデータ本体に影響を及ぼさない点である．しかし，索引分散管理システムでは稼働している間に処理効率が悪化することがある．この現象には，(1) 負荷の不均等による処理効率の悪化，(2) 負荷は均等であるが，全体として処理効率の悪化の 2 種類がある．(1) の原因には，検索条件の偏りにより索引の一部に負荷が集中する場合や，索引要素の更新によりその後の処理における索引の一部に負荷が集中する場合が考えられる．(2) の原因には，索引を分散管理するシステムの場合，検索要求数，検索条件，要求間隔などによって最適な索引分割が異なり，検索要求に対して最適な索引分割ではないために起こる場合が考えられる．たとえば，範囲の狭い範囲検索と範囲

† 福井大学工学部
Faculty of Engineering, Fukui University

の広い範囲検索では最適な索引分割はまったく異なり，処理負荷は均等でも全体として非効率的な状態になることがある．これらの処理効率の悪化を解消するためには索引の再分割，つまりシステムの再編成が必要となる．ここで注意すべきは (1) の解消に必要となる再編成は比較的小規模の再編成であるが，(2) の解消に必要な再編成は大規模な再編成であることが少なくないことである．よって，索引分散管理システムにおいては様々な規模での再編成に対応した再編成手法が必要となる．また，再編成処理は非常に高コストな処理であるため，他の処理への影響を少なくすることが求められる．したがって，オンライン再編成は重要な問題である．

データベースにおける再編成に関しては様々な研究がなされている[2),3),5),6),10)~12),14)]．文献 2)，6)，12) ではデータに対する索引の再編成の方式が提案されており，文献 14) では参照関係に関するデータの再編成も行っている．これら手法も含めた一般的な再編成手法に基づくと，索引分散管理システムにおける再編成は以下の手順で行われる．

1. 新しい索引分割の決定.
2. 関係する索引要素に対するロック取得.
3. 索引要素の移動.
4. 必要なデータの更新.
5. 索引要素に対するロックの開放.

ここで重要となるのは「関係する索引要素のロック取得」である．つまり,「関係する索引要素」の特定とそれを格納している場所の特定のための処理が必要となる．既存の手法では再編成のみに利用する索引や機構を用意し，この問題を処理している．したがって，これら再編成用の索引や機構を適切に管理することが必要となり，他の処理に影響を及ぼす.

そこで，本論文では，これらの欠点を考慮し，索引分散管理システムに適した再編成手法を提案する．以下は我々の提案する手法の概要である.

- 索引の分割決定にハッシュ関数を用いる.
- 再編成はハッシュ関数の変更とそれによって必要な索引要素の移動によって実現する.
- 再編成の対象範囲を索引構成に基づいて分割し，段階的に行う.
- 再編成処理は索引要素の更新手法を基本に行う.

本システムでは再編成処理中も他の検索，更新要求に対する処理が可能であり，また，データ本体の情報を利用しない．したがって，索引分散管理システムにおけるデータ本体と索引の分離という利点を損なうことはない．本システムを分割なしのオンライン手法，およびオフライン再編成手法とシミュレーションで比較し，その優位性の検証を行う．なお，本論文では，何らかの理由により再編成が必要な場合に，他の処理への影響をいかに少なくおさえながら再編成を高速に行うかを議論の対象とし，再編成をすることでどの程度，処理の効率が向上するかに関しては議論の対象としない.

2. 対象オブジェクトと索引

以下では対象とするオブジェクトおよび索引，検索の定義を行う.

2.1 複合オブジェクト

複合オブジェクトの検索パス式を

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

とし，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，$1 \leq j < N$ 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_i が属性 A_i 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，P の値を

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

とする．ここで，$o_1, o_2, \ldots, o_N$ はそれぞれ $C_1, C_2, \ldots, C_N$ のインスタンスであり，o_j $(1 < j \leq N)$ はオブジェクト o_{j-1} の属性 A_{j-1} 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 ID（以下，OID）である．P の値の集合を O_P と表記する.

なお，本論文においては検索パス式は 1 つに限定し，クラス C_i $(1 \leq i \leq n)$ はそれぞれ異なるクラスとし，任意の異なる 2 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．また，各 C_i 間の参照関係が，独立したオブジェクトどうしの参照関係であるか，内包関係であるかは索引上では何ら違いはないため，オブジェクト間の関係の種類は特に限定はしない.

2.2 マルチインデックス

文献 18) 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はマルチインデックス手法[1),4)] 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接的な参照関係の逆関係をそのまま索引要素とするもので，検索はその索引要素を順に検索していくことで行われる（リバーストラバーサル）.

オブジェクト o_j の属性 A_j の値（集合値の場合はその要素）が o_{j+1} であるとき，

$$\langle o_j, o_{j+1} \rangle$$

を P の索引要素という．ここで o_{j+1} はキー値，o_j はデータ値となる．ただし，実際の索引要素のキー値，データ値には OID（または単純値）を用いることとする．さらに，IP をすべての索引要素の集合とし，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の集合を IP_i とする．また，A_N の値をキー値とする索引要素集合を IP_{N+1} とする．

マルチインデックスにおける検索要求 “要素 o_{N+1} を属性 A_N の値としているオブジェクトを検索パス式 P 上で参照している C_1 のインスタンスを求める” 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

$$\langle o_N, o_{N+1}\rangle, \langle o_{N-1}, o_N\rangle, \ldots, \langle o_1, o_2\rangle$$

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

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

と定義する．

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．本論文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. 索引の分散管理システム

文献 18) における索引分散管理システムの概要について述べる（詳細に関しては文献 18) を参照のこと）．索引分散管理システムを構築する並列環境としてメッセージ通信非共有メモリ型並列計算機を仮定する．また，各 PE での処理は 1 プロセスの逐次処理とする．

上記並列処理環境において索引に対する処理を並列に行うために，IP を分割し，各 PE にそれぞれ格納して管理する．ただし，索引の分割には “キー値が同じ索引要素は同じ PE で管理する” という制限以外は加えない．したがって，IP_i が特定の一部の PE 集合に分割して格納される場合や，IP_i がすべての PE によって分割して格納される場合など，自由な索引分割が可能となる．

PE は HOST，検索 PE，DETECTOR の 3 種類に分類される．以下にそれぞれの PE の処理の概要を記す．また，**図 1** は検索時の論理的な処理の流れを，**図 2** は検索時の実際のシステムにおける物理的な処理の流れを表し，**図 3** は索引要素更新時の処理の流れを表す．ここで，図中の三角形は索引の一部を表し，同じ色の三角形は格納する索引要素が同じ IP_i に属することを示している．また，図 2 における交差した実線の矢印は HOST，検索 PE 間での相互の通信を簡略化して表現しており，実際には，検索 PE への通信は検索要求であり，HOST への通信は検索結果の送信である．

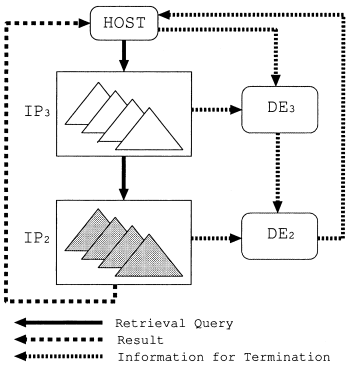

図 1 論理的な検索の流れ
Fig. 1 Logical retrieval process.

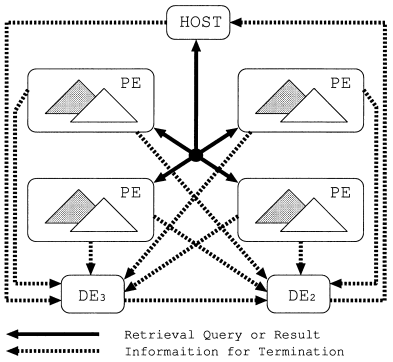

図 2 物理的な検索の流れ
Fig. 2 Physical retrieval process.

なお，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，その索引のもとである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 HOST

検索要求，更新要求などの外部からの処理要求を検索 PE，DETECTOR に送信し，その処理結果を集計し，最終的な処理の終了を判定するための PE である．このとき，外部から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．また，処理要求として送信したメッセージ総数の情報を DETECTOR に送信する．

3.2 検 索 PE

実際に索引を格納し，検索，更新要求などを処理するための PE である．

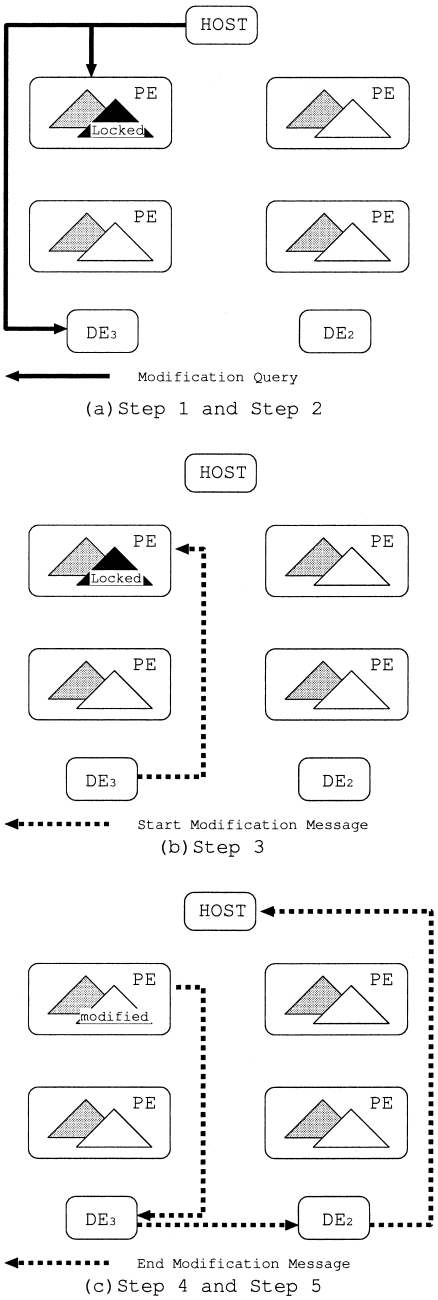

図 3 索引更新の流れ
Fig. 3 Scheme for modification.

検索処理は以下の 1.〜5. の手順で実行される．

1. HOST は検索条件中の A_N の値をキー値とする索引要素が格納されている検索 PE に対して検索要求を送信する．
2. 各検索 PE は検索要求が到着したならば，その検索条件の OID（または単純値）を自分が管理する IP の部分索引で検索する．
3. 2. における検索結果がクラス C_1 のインスタンスの OID ならば HOST に検索結果として送信する．そうでない場合，その OID をキー値とする索引要素を管理する検索 PE に対し，その OID を検索条件とする検索要求を送信する．
4. 各検索 PE に対して検索要求があるかぎり 2. と 3. を繰り返す．
5. すべての検索 PE の処理が終わり，HOST にすべての検索結果が到着したならば検索終了となる．

ここで，1. および 3. において検索要求の送信先 PE を特定する必要がある．つまり，索引要素がどの PE で格納されているかの情報が必要となる．送信先 PE を特定するための方法としては，(a) ハッシュ関数を用いて分割を決定し，それと同じハッシュ関数を用いて送信先 PE を決定する方法，(b) 何らかの方法で分割を決定し，索引要素自体に送信先 PE の情報を埋め込み，それを参照する（つまり，索引要素のデータ値として OID とその OID をキー値とする索引要素を格納する PE の位置の組を用いる）方法，(c) 何らかの方法で分割を決定し，索引要素の位置情報を別に管理し，その情報を用いて送信先 PE を決定する方法の 3 種類が考えられる．また，後述の DETECTOR による終了判定を行うために，他の検索 PE（または HOST）からいくつのメッセージを受信し，それを処理した結果，いくつのメッセージを送信したかの情報を DETECTOR に送信する．

3.3 DETECTOR

DETECTOR（以下，DE）は HOST から発行された要求に対する処理が終了したか否かの判定を行うための PE である．DE は IP_i ごとに 1 つずつ用意され，検索 PE，HOST，DE からもたらされた IP_i に関するメッセージの送信数，受信数の情報を用いて，RID ごとに IP_i の処理が終了したかどうかを判定する．これは，図 1 で示されるように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} の検索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する．つまり，IP_i に関する検索処理の終了は，IP_{i+1} に関する処理がすべて終了し，かつ，IP_i への検索要求のメッセージの総数と IP_{i+1} の検索結果として送信されたメッセージ総数が等しいときに限られる．ただし，$i = N+1$ の場合，“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 からの検索要求である．また，$i = 1$ の場合，“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．この性質を用いて検索の終了を判定する．

ここで，IP_i に関する DE を DE_i と表記することとする．DE_i は図 1 で示した論理的な検索処理の流れに基づいて IP_i に関する処理の終了判定を行う．したがって，実際のシステムにおいては，索引分割によっては，DE_i は図 2 で示した物理的な検索処理の流れのようにすべての検索 PE と情報のやりとりをする場合や，特定の検索 PE 集合のみと情報のやりとりを行う場合など様々な状況がありうる．逆に各検索 PE はすべての DE に対して情報のやりとりを行う場合や，ただ 1 つの DE のみと行う場合など様々な状況がありうる．

この終了判定を用いて索引更新の処理のタイミングを決定する．

索引の更新に関しては以下の 1.〜5. の手順で実行される．ただし，更新要求は IP_i に属する索引要素を更新対象とする．

1. HOST は更新対象の検索要素を管理する検索 PE と DE_{N+1} に対し，更新要求を直接送信する．
2. 更新要求を受信した検索 PE（更新対象 PE）は DE から更新許可が到着するまで，処理可能な要求に対して処理を行う．
3. 更新要求の到着した DE（DE_j とする）は $i \neq j$ ならば DE_{j-1} に対し更新要求を転送する．$i = j$ ならば IP_i に関する処理が更新要求の RID より小さな RID を持つ処理要求すべてに対して終了した場合，更新対象 PE に更新許可を発行する．
4. 更新許可を受信した更新対象 PE は更新処理を行い，その終了を DE に伝える．
5. 更新終了のメッセージを受信した DE（DE_j とする）は DE_{j-1} に更新終了メッセージを送信する．ただし，$j = 2$ ならば HOST に対して送信を行う．

この更新方法により，更新要求より前に発行された処理要求はすべて更新処理前に実行され，更新要求より後に発行された処理要求はすべて更新処理後に実行されることが保証される．ここで問題となるのは 2. の “処理可能な要求” の決定である．文献 18) では以下の条件のどちらかを満たす処理要求に関しては “処理可能な要求” として処理を行う．

- 更新要求の RID より小さな RID を持つ処理要求．
- 更新対象の索引要素の属する IP_i に属さない索引要素に関する処理要求．

この終了判定および索引更新手法により検索処理と更新処理が同時実行可能なシステムとなる．つまり，更新処理においては検索 PE における IP_i 単位で排他ロックが行われ，他の索引要素にはまったく無関係に処理が可能であり，検索 PE としてはアイドル状態に陥らずに何らかの処理が可能となる．ただし，索引分散管理システムにおける排他ロックは，実際に索引を排他ロックするのではなく，処理要求の保留として取り扱う．

4. オンライン再編成法

4.1 索引分割決定法と再編成

本論文で扱う再編成とは検索 PE に分散格納されている索引要素集合を再分割し，それらを検索 PE に再分配することである．したがって，検索 PE 間で索引要素の移動が生じ，それに対する処理が必要となる．

まず，システム全体として以下を前提とする．

- 索引の分割にはハッシュ関数を用いる．
- 各 PE は上記の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の送信先を決定する．
- 再編成要求は HOST が発行し，RID を付加する．
- 再編成要求はハッシュ関数の変更要求とし，すべての PE のハッシュ関数を変更する．
- 索引要素移動は各検索 PE 間で直接通信を行って移動させる．

3.2 節で述べたように索引分散管理システムでは処理要求（検索要求，索引要素の更新要求）をどの PE に送るべきかを知る必要がある．この方法として 3 つの方法をあげた．これら 3 つの方法に対して再編成処理を考慮して検証を行う．3.2 節 (b) の索引要素自体に送信先 PE の情報を埋め込む方法では，再編成時にこれらの情報も書き換える必要があるために，変更が必要な索引要素数は増加し，ロックが必要な範囲も拡大する．つまり，ある索引要素を移動するためには，そのキー値をデータ値とする索引要素のすべてが変更対象となり，移動時にはこれらの索引要素をすべてロックする必要がある．また，この変更対象となる索引要素を特定するためには索引要素と逆の方向の索引を新たに管理するか，すべての検索 PE にブロードキャスト通信でロックの要求を行う必要がある．これは，非常に高コストな処理であり，再編成処理を考慮するならば 3.2 節 (b) の方法を用いることは妥当ではない．3.2 節 (c) の位置情報を別に管理する方法でも位置情報の更新を行うことが必要となる．つまり，索引要素を移動する場合は，移動時にその位置情報の更新もそれぞれ行う必要がある．再編成時には大量の索引要素が移動することがあり，そのために位置情報用索引の更新も大量に行う必要がある．これは，非常に高コストな処理である．また，3.2 節 (c) の方法では

HOST，検索 PE において検索要求を送信する前に検索条件の各キー値に対して検索を行う必要がある．この位置情報用索引へのアクセス数は複合オブジェクトに対する索引でのアクセス回数と同数であり，検索要求に対する応答時間が悪化することは明らかである．したがって，3.2 節 (c) の方法を用いることは妥当ではない．一方，3.2 節 (a) のハッシュ関数を用いる場合はハッシュ関数の変更のみで完了する．ただし，すべての PE のハッシュ関数を再編成に同期して変更する必要がある．また，ハッシュ関数を用いることは他の方法に比べて分割の制限になる．しかし，ハッシュ関数を複雑化させることで任意の索引分割に近づけることは可能であり，この分割の制限の問題に対してはある程度までは解消可能である．ただし，ハッシュ関数を複雑化し，任意の索引分割を可能とすると，本質的には (c) の方法と同じものとなり，(c) と同様の問題を招くことになる．したがって，比較的単純な関数で実装環境，対象索引に応じたハッシュ関数の決定が必要となる．これらを考慮し，本システムでは索引の分割，処理要求の送信先 PE の決定にハッシュ関数を使用することとし，再編成は索引要素の移動とハッシュ関数の変更によって達成される．また，索引要素移動においてはハッシュ関数を用いることで移動先の検索 PE を特定可能であり，検索 PE 間で直接的に移動をすることが可能となる．

以上を前提とし，索引分散管理システムにおける新たな再編成手法を提案する．なお，分割の決定にどのハッシュ関数を用いるかや，どのハッシュ関数に変更するかに関しては本論文の議論の対象外とする．

4.2 ロックの取得

再編成において重要となるのは再編成要求に関係する索引要素に対するロックの取得である．つまり，ロックの範囲とタイミングをどのように決定するかである．ロックする範囲によって他の処理との並列度が変化し，また，ロックの取得が成功する可能性も変化する．したがって，再編成の処理時間や他の処理への影響も変化する．

ロックが必要となる最低限の範囲は再編成により移動する索引要素である．しかし，この移動する索引要素のみでは，再編成中に処理要求の喪失が起きる可能性がある．つまり，索引要素が移動する再編成では格納している検索 PE が再編成前と再編成後では変更されるため，処理要求の送信先 PE を変更する必要があり，この変更が矛盾なく行われない場合は処理要求の喪失が起こってしまい，処理結果に対する正当性が失われる．したがって，ロックの対象には移動する索引要素のキー値の OID を対象とする処理要求も含まれる．ここで，再編成によって移動を要求されている索引要素を “移動対象索引要素”，移動対象索引要素のキー値の OID（または単純値）が検索条件に含まれる検索要求または移動対象索引要素のキー値と更新対象の索引要素のキー値が等しい更新要求を “移動対象関連要求”，移動対象関連要求の処理対象（検索要求ならば検索条件であり，更新要求ならば更新対象の索引要素のキー値）をデータ値とする索引要素を “移動対象関連索引要素” と呼ぶことにする．

移動対象関連要求は HOST または検索 PE により送信される．HOST においては送信保留にすることで移動対象関連要求に対する処理は達成可能である．一方，検索 PE における移動対象関連要求に対する処理としては以下の 4 つが考えられる．

(a) 移動対象関連索引要素をロックする．
(b) 処理要求を送信時に移動対象関連要求であるか確認し，移動対象関連要求ならば送信を保留する．
(c) 処理要求の到着時に移動対象関連要求であるか確認し，移動対象関連要求ならばそれに対する処理は行わず，正しい送信先に転送する．
(d) 再編成時にはすべての索引要素をロックする．

(a) の方法は移動対象関連要求が検索 PE 内で生じない方法である．また，索引要素をロックする機構は既存の更新処理を行う機構を用いて容易に実現可能である．

(b) の方法は移動対象関連要求を送信しない方法である．(a) の方法と比べると検索前に保留するか，検索後に保留するかの違いであり，検索 PE が高負荷な状況では本質的な違いはない．つまり，移動対象関連索引要素を使用する処理以外にも行わなければならない処理（再編成に関する処理や他の検索，更新処理）が大量にあり，どちらの処理を優先したとしても処理の順序だけの差となり，全体としての処理時間としては違いはないと考えられる．

(c) の方法は受信した処理要求が移動対象関連要求ならば転送する方法である．この方法の場合，(a)，(b). の方法に比べて通信回数が明らかに増加する．また，転送のための検査をいつまで行えばいいかの判定や，転送の繰返しが起こる可能性を否定できず，効率的とはいえない．

(d) の方法は索引全体をロックするため，検索 PE 間では移動対象関連要求は存在しない．しかし，これはオンライン再編成ではなく，オフライン再編成である．

これらの問題点と索引分散管理システムの既存の機能を考慮し，(a) の方法を基本とする再編成方法を提

案する．ところが，(a) の方法をそのまま使用すると移動対象関連検索要素の特定を行わなければならない．そのためには，検索の逆方向，つまり，オブジェクトの参照関係をたどることが必要となる．したがって，データ本体の情報を利用するか，そのための新たな索引が必要となる．これは，本システムにおけるデータ本体と索引の分離という利点を著しく損なうか，膨大な量の検索が必要となり，本システムにおいては採用することはできない．一方，移動対象索引要素は決定していることから，移動対象関連索引要素がどの IP_i に属するかを特定することは可能である．そこで，移動対象関連索引要素をロックするのではなく，移動対象関連索引要素を使用する可能性のある処理要求を保留状態にすることで実現する．つまり，移動対象関連索引要素が IP_i に含まれる場合は，IP_i に属する索引要素を使用する処理要求はすべて保留状態とする．

4.3 再編成処理の分割

4.2 節における処理要求の保留のみで索引要素移動を開始した場合，再編成中の他の処理が正しく行われる保証はない．それは，処理要求の保留開始前に送信された移動対象関連要求に対する処理が終了しないかぎり，索引要素移動後に移動元の検索 PE に移動対象関連要求が到着する可能性があり，その結果，処理結果が正しいことの保証がなくなる．しかし，処理要求の保留開始前に送信された移動対象関連要求が処理されたか否かの判定は，すべての検索 PE に対して問い合わせる必要があり，膨大なコストがかかる．そこで，索引分散管理システムにおける処理の終了判定を利用することで索引要素移動のタイミングを決定することにする．つまり，以下のタイミングで索引要素移動を行うものとする．

- それ以前に発行された要求に対する処理が終了後に索引要素移動を開始．
- それ以降に発行された要求に対する処理の開始前に索引要素移動を完了．

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り，他の処理の正当性に影響はない．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の場合，オフライン再編成との違いはなく，再編成処理とそれ以外の処理との並列性は望めない．そこで，再編成処理を IP_i 単位で分割し，それらを処理することで全体の再編成が完了する方法を採用する．つまり，再編成処理を，IP_{N+1} に関する処理，IP_N に関する処理，...，IP_2 に関する処理に分割し，連続的に処理する．これにより再編成処理と他の処理との並列性が高まる．

IP_i を対象とする再編成は，以下の手順で処理される．

1. すべての検索 PE において再編成要求の RID より大きな RID を持ち，IP_{i+1} に関する処理要求を保留状態にする．
2. 再編成要求の RID より小さな RID を持つ処理要求に対して IP_i に関する処理終了後に索引要素を移動．
3. ハッシュ関数を変更し，1. の保留状態の処理要求に対し処理開始．

上記 1.，2. の処理は索引要素の更新処理における処理要求の保留と更新開始許可と同じである．したがって，索引要素の更新処理の機構を利用することで実現可能である．また，1.，2. においては IP_i に関する処理要求はシステム中に存在しないことが保証されるため，移動対象索引要素に対するロックは索引要素の削除，移動，挿入のためのロックになり，同時実行制御のためのロックは必要なくなる．

4.4 索引要素移動の開始

4.3 節の 2. を行うためには 1. の状態であることが必要となる．したがって，すべての検索 PE に 1. の要求を送信する必要がある．また，2. 自体も多くの検索 PE に送信する必要があり，3. の処理もすべての検索 PE に送信する必要がある．これにはブロードキャスト通信を行うことになり，高コストな処理となる．HOST や DE は他の処理要求に対する処理も行う必要があり，長時間 1 つの処理を行うことは全体の処理効率に影響する場合がある．そこで，DE 集合を利用して検索 PE に送信する手法を導入する．再編成処理においては，各 DE はそれぞれ検索 PE 集合の管理を担当し，各検索 PE はただ 1 つの DE によって管理されるものとする．ただし，この管理は再編成処理に関するもののみであり，その他の検索，更新処理に関する検索 PE と DE の関係には無関係である．以下に IP_i に関する再編成におけるすべての検索 PE に対して保留要求を送信する手順を示す．

i. HOST は DE_{N+1} に再編成要求を送信．
ii. 再編成要求を受信した DE_j は自分の管理する検索 PE に保留要求を送信し，DE_{j-1} が存在するならば，再編成要求を送信する．

また，4.3 節の 2. の条件を満足したか否かは DE の終了判定情報を利用する必要がある．したがって，IP_i に関する処理の終了判定を行う DE_i において索引要素移動開始可能か否かの判定を行う．そこで，以下の処理を追加する．

iii. 再編成要求を受信した DE_2 は DE_i に対してす

べての検索 PE に保留要求が送信されたことを伝える．

以上の処理により，DE_i は自らの終了判定情報を利用することで索引要素移動開始の許可を与えることが可能となる．なお，再編成対象が全 IP_i にいたる場合，iii. の処理は全 DE への送信となる．しかし，通常，DE は検索 PE に比べて少数であるため，ブロードキャスト通信より短時間で終了する．

この DE と検索 PE の関係を利用してすべての検索 PE に索引要素移動開始の許可，ハッシュ関数の変更，保留解除の許可を送信することも可能である．

4.5 索引要素移動の終了

上記再編成においては各検索 PE における索引要素移動が同時に終了するとは限らない．したがって，索引要素の挿入要求がすべて到着したか否かは分からない．保留状態の処理要求を処理開始後に索引要素の挿入要求が到着した場合は間違った処理結果となる可能性がある．つまり，処理結果の正当性が失われる．したがって，保留状態の処理要求を処理開始するには全体の索引要素移動の終了を待たなければならず，そのためには全体の索引要素移動の終了を判定する必要がある．索引要素移動は，移動元の検索 PE における移動対象索引要素の削除，送信と移動先の検索 PE における受信，挿入の手順で行われる．ただし，同じ RID の索引要素移動に関しては各索引要素は 2 回以上移動することはない．そこで DE では管理している「受信予定メッセージ数」「受信メッセージ数」の 2 つのカウンタと DE 数に対応した数の「送信メッセージ数」のカウンタを用意し，終了の判定を行う．ここで,「受信予定メッセージ数」は管理する検索 PE が最終的に受信するメッセージの総数,「受信メッセージ数」は管理する検索 PE が実際に受信したメッセージの総数,「送信メッセージ数」は他の検索 PE に送信したメッセージ数を送信先の検索 PE を管理している DE ごとに集計した数である．また，DE_j の「受信予定メッセージ数」は，すべての索引移動に関する送信が終了した時点での「送信メッセージ数」の中で送信先が DE_j であるものの総和となる．したがって，すべての索引移動に関する送信が終了し，かつ，すべての DE より「送信メッセージ数」に関する情報を得ないかぎり,「受信予定メッセージ数」は確定しない．上記カウンタは，再編成が IP_i 単位で行われるため，再編成対象の IP_i ごとに用意する必要があり，各カウンタの値は検索 PE と DE からの索引移動に関するメッセージ数の情報を用いて更新を行う．これは，検索処理における IP_i ごとの処理終了の判定を拡張したものであり,「受信予定メッセージ数」と「受信メッセージ数」を比較することで索引要素移動の終了を判定する．つまり，DE_j において，IP_k に関する「受信予定メッセージ数」と「受信メッセージ数」が等しいならば，DE_j の管理する検索 PE 集合においては IP_k に関する索引要素移動は終了したと判定する．

ここで問題となるのは「受信予定メッセージ数」である．これは，他の DE における「送信メッセージ数」により決定されるが，送信がすべて終了した状態での「送信メッセージ数」でなければ正しい終了判定は行えない．そこで，管理する検索 PE 集合のすべてから関連する索引要素移動に関する送信が終了し，その送信メッセージ数の情報が到着後に，DE 間の「送信メッセージ数」情報のやりとりを行うこととする．そして，自分を含めたすべての DE から「送信メッセージ数」情報を取得したとき，その合計を「受信予定メッセージ数」として確定することができる．

また，すべての DE において IP_i に関する索引要素移動の終了が確認されたならば，IP_i に関する索引要素移動が終了したことになる．これを判定するために各 DE は索引要素移動が終了した場合，DE_i に対して索引要素移動が終了したことを伝える．そして DE_i がすべての DE から IP_i に対する索引要素移動の終了情報を得たとき，IP_i に関するすべての索引要素移動が終了したことになる．

4.6 ロックの開放とハッシュ関数の変更

索引要素移動の終了が確認されたならば，4.3 節の 3. の処理を行う．そのため，全検索 PE に対してその要求を送信する必要がある．これに関しても DE と検索 PE の関係を利用して送信可能となる．そして，すべての検索 PE において 4.3 節の 3. の処理が行われたときに再編成が終了となる．

4.7 アルゴリズム詳細

以下にアルゴリズムの詳細を記す．なお，再編成要求に付加された RID を rid とする．

［HOST］

1. 再編成要求をハッシュ関数の変更として DE_{N+1} に送信．
2. 1. を処理後，IP_{N+1} を対象とする再編成処理が終了するまで次の処理要求を送信しない．

［DETECTOR（DE_i における処理）］

3. 再編成要求を受信したならば，管理する検索 PE に各 IP_j に関する処理の保留を要求．
4. 3. を処理後，$i \neq 2$ ならば DE_{i-1} に再編成要求を送信．$i = 2$ ならば各 DE_j に対して保留完了を送信．

5. 4. の保留完了の情報を受信し，$rid-1$ までの処理要求の IP_i における処理が終了したならば，全 DE に IP_i に対する索引要素移動の開始を許可する．
6. 5. の許可を受信したならば，管理する検索 PE に索引要素移動開始を要求する．
7. IP_j における索引要素移動の終了が確認されたならば（後述），DE_j に対し索引要素移動の終了を伝える．
8. すべての DE から IP_i に関する索引要素移動の終了情報を受信したならば，各 DE に対し，保留解除要求を送信し，HOST に対して IP_i に関する再編成が終了したことを伝える．
9. 保留解除要求を受信したならば，管理する検索 PE に対して保留解除とハッシュ関数の変更を要求する．

［検索 PE］

10. IP_i に対する保留要求が到着したならば，rid より後の処理要求で，かつ，IP_i に対する処理を保留状態にする．
11. 索引要素移動開始要求が到着したならば，対象となる索引要素を削除し，移動先へ送信する．
12. 11. を処理後，管理している DE に対し，索引要素移動の終了判定に関する情報を送信．
13. 索引要素が到着したならば，索引要素を挿入し，管理している DE に対し索引要素移動の終了判定に関する情報を送信．
14. 保留解除要求が到着したならば，対応する保留を解除し，ハッシュ関数を変更する．

［索引要素移動の終了判定（DE_i における IP_j に関する処理として記す）］

15. 管理する検索 PE からの情報により「送信メッセージ数」,「受信メッセージ数」をカウンタに登録，更新する．
16. 管理するすべての検索 PE から「送信メッセージ数」情報を得たならば，管理する検索 PE において IP_j の索引要素移動の送信処理が終了したものとして，各 DE に対応する「送信メッセージ数」を伝える．
17. 各 DE から「送信メッセージ数」情報が到着したならば，その累計を「受信予定メッセージ数」としてカウンタを更新．
18. すべての DE から「送信メッセージ数」情報が到着したならば,「受信予定メッセージ数」を確定する．
19. 「受信予定メッセージ数」が確定し,「受信メッセージ数」と等しいならば，管理する検索 PE における IP_j の索引要素移動の処理は終了したとして DE_j に処理の終了を伝える．

4.8 既存手法との比較

既存の手法を用いて索引分散管理システムにおける再編成を行う場合，移動対象索引要素を移動データ，移動対象関連索引要素を移動データに関係するデータ（または索引）と見なすことで適用可能である．

文献 2)，6)，12)，14) ではデータの移動に対して，関係する索引に対する更新方法について述べられている．索引分散管理システムの場合は移動対象関連索引要素がこの関係する索引となる．文献 2)，6)，12) では関係する索引の位置は一意に定まることを前提としている．一方，文献 14) では移動対象関連索引要素の位置を索引化し，各 PE がそれを管理している．いずれにせよ，移動対象関連索引要素の位置を特定するための機構を保持していることが前提となる．しかし，索引分散管理システムが管理している索引はオブジェクト間の逆参照関係であるので，その逆の関係を索引要素とする位置特定のための機構はオブジェクト間の参照関係に対する索引となる．つまり，索引分散管理システムが管理している索引と同規模の索引となる．しかも，単なる索引要素の位置ではなく，オブジェクト間で参照関係にあるという条件のため，我々の提案しているようなハッシュ関数を用いて索引要素の位置を特定する方法だけでは不十分であり，完全に索引として管理する必要がある．また，この位置特定用索引は参照関係と同期する必要があり，参照関係の変更時には元々の索引とともに変更する必要がある．さらに，再編成時に移動対象関連索引要素を把握するためには，移動対象索引要素 1 つに対して，位置特定用索引を用いて 1 回検索する必要がある．文献 2)，6)，12) において位置特定用索引を集中管理とした場合は，その管理する PE に位置特定のための要求が集中し，ここがボトルネックとなることが予想される．一方，文献 2)，6)，12) において位置特定用索引を分散管理とした場合や文献 14) の場合は，移動対象索引要素の移動とともにこの位置特定用索引の索引要素も移動する必要がある．つまり，再編成時には索引を実際に管理している B^+ 木などに対して倍の削除，挿入処理が必要となり，索引要素の移動量がほぼ倍になる．これらのことを考慮するならば，これら既存の手法は，本手法に比べて空間的，時間的コストが非常に高いことは明らかであり，索引分散管理システムにおけるオンライン再編成手法としては提案手法に比べ劣ることは明らかである．

文献 8)，9) では B^+ 木におけるオンライン再編成法について述べられている．文献 8) では葉ノードの未使用部分が多くなり，非効率的になったときに葉ノードを統合し，内部ノードを更新する再編成手法をオンラインで行っている．文献 9) ではデータベースにおけるレコード移動にともなう二次索引の更新をオンラインで行う手法を提案している．これらを組み合わせることで索引分散管理システムにおける索引要素の移動（文献 8) を利用）とそれに関係する索引要素の更新（文献 9) の二次索引の更新を利用）を行うことは可能である．しかし，これらの手法では索引要素移動時にそれに関係する部分のロックを取得する必要があるが，それを行うためには移動対象関連索引要素の位置を特定するための機構が必要となる．これは前述の方法と同様の問題点を持ち，索引分散管理システムの再編成法としては直接適用することはできない．

文献 15)，16)，17) においては並列 B 木における負荷の偏り制御に関して述べられている．この手法では負荷の偏りが発見された場合，部分木を PE 間で移動することで偏り制御を行っている．しかし，これらの手法では，索引をキー値の範囲ごとに区切って分割して各 PE に格納し，再編成においてもこの前提を崩すことは許されていない．つまり，部分木の移動先はキー値として隣接している他の部分木を管理する PE のみと限定されている．したがって，任意の PE に移動するためには何度も再編成を行う必要があり，大規模な再編成には向かない．これはこの手法が負荷の偏り制御を目的とした再編成手法であるためであり，大規模な再編成も必要となる索引分散管理システムの再編成法としては適さない．

文献 3)，5)，10) などのオブジェクト指向データベースにおけるガーベージコレクションに関する方法では参照されていることが重要であり，参照しているデータの位置の特定をしないか，すべての索引要素を調べて特定する方法を用いている．したがって，ガーベージコレクションに用いる手法は再編成法としては適さない．

5. 計算機シミュレーション

計算機シミュレーションにより，本方式の有効性について検証を行う．本シミュレーションは実計算機上に実現した仮想的な並列計算機上でのシミュレーションであり，単位時間を基準としてイベントの発生時刻と終了時刻を決定することで総処理時間を計測する．また，実験結果は単位時間で表すものとする．

提案手法との比較対象はオフライン手法と再編成処理を分割せずに行う手法とする．既存の手法ではデータ本体へのアクセスや再編成専用の大規模な索引が必要であったり，処理要求の転送などの余分な処理が必要であり，本手法より再編成処理のコストが高いことは明らかであるため，比較対象としては適さない．一方，オフライン再編成は提案手法にくらべ，他の処理との並列性は劣るが，再編成処理のみを行った場合は複雑な処理を行わない分，提案手法より高速であることが予想される．さらに，索引全体を対象とするような大規模な再編成の場合は提案手法においても他の処理との並列性はそれほど望めないため，提案手法が必ず優位であるとはいいきれない．そこで，オフライン手法を比較対象とする．また，再編成処理を分割することによる高速化を検証するために，再編成処理を分割せずに行う手法も比較対象にあげる．よって，再編成対象を一部のものから索引全体となるものまで変更してシミュレーションを行い，提案手法，再編成処理を分割せずに行うオンライン手法，オフライン手法の 3 つの手法を比較，検証する．ただし，本実験においては再編成処理以外の検索，更新処理は 3 つの手法ともまったく同一の処理であり，再編成手法による差のみを比較，検証するものである．

5.1 シミュレーション条件

以下の条件の複合オブジェクト集合を 10 セット用意する．

- 索引の対象となる複合オブジェクトは $N = 8$ で各クラスのインスタンス数は 10000.
- 各オブジェクトはたかだか 2 個のオブジェクトを参照する．ただし，参照数および参照先オブジェクトはランダムに決定され，2 つの場合は異なるオブジェクトを参照する．

上記の複合オブジェクト集合をマルチインデックス手法で索引化し，それぞれに対してシミュレーションを行う．

索引分散管理システムとして以下を仮定した．

(a) 検索用 PE 数は 64 であり，0 から 63 まで ID が付加されているとする．

(b) 通信パケットは最大 100 の OID を格納可能である．

(c) C_s を通信初期化時間，C_t を 1 つの OID を送信するのに必要な時間，T_r を 1 つの索引要素を検索するのに必要な時間としたとき，その時間比を $C_s : C_t : T_r = 100 : 1 : 100$（case 1），$C_s : C_t : T_r = 100 : 1 : 500$（case 2）とする．これらのパラメータは C_t を 1 単位時間として設定する．また，1 つの索引要素を削除する時間，挿

入する時間は T_r と等しいものとする．

(d) 索引分割に用いるハッシュ関数は，
Type 1： $(\mathrm{CID}-2)\times 8+(\mathrm{IID} \mod 8)$
Type 2： $\mathrm{IID} \mod 64$
の 2 種類を用意する．ここで，CID は索引要素のキー値のオブジェクトの属するクラスに対する ID で，クラス C_i に属するオブジェクトは CID= i とし，A_n の値の場合は CID= 9 とする．また，IID はそのクラス内のインスタンスに固有な ID である．索引要素ごとにこのハッシュ関数を用いて格納する検索 PE を決定する．

(e) 再編成処理における DE_i は $(i-2)\times 8$ から $(i-2)\times 8+7$ までの ID を持つ検索 PE を管理する．

それぞれのハッシュ関数を用いて分割した索引に対してシミュレーションを行う．ここで，(d) のハッシュ関数はハッシュ値が IP_i ごとにまとまっているタイプ（Type 1）と IP_i に関係なく全体に分散されているタイプ（Type 2）の 2 種類を用意した．よって，1 つの検索要求に対する処理においては，Type 1 は 1 つの検索要求に対する処理内での並列性の低いハッシュ関数であり，Type 2 は 1 つの検索要求に対する処理内での並列性の高いハッシュ関数であるといえる．一方，複数の検索要求に対する処理においては，Type 1 は IP_i ごとにパイプライン的に処理されるため，各検索要求に対する処理間での並列性が高いハッシュ関数であり，Type 2 は 1 つの検索処要求を集中的に処理する傾向が強く，各検索要求に対する処理間での並列性が低いハッシュ関数であるともいえる．一方，1 つの検索 PE の送信先の検索 PE 数の点では Type 1 は 8 であり，Type 2 は 64 である．したがって，Type 1 は通信回数の少ないハッシュ関数で，Type 2 は通信回数の多いハッシュ関数である．これらのことより，2 つの関数は単純な式ではあるが非常に性質の異なる関数ということができる．しかも，Type 1 から Type 2 または Type 2 から Type 1 に分割を変更した場合，移動が必要となる索引要素は全体の 7/8 であり，非常に大規模な再編成となる．

また，(c) の時間比は IBM 製非共有メモリ型並列計算機 RS/6000SP 上で通信に MPI ライブラリ，索引要素の格納に二次記憶上の B^+ 木を使用した場合の実測値を参考に，ディスクアクセス速度に大きく依存する索引に対する処理時間 T_r と通信速度に大きく依存する通信に対する処理時間 C_s，C_t の 3 者の比の異なる 2 種類の状況を想定した．

5.2 実 験 1

索引分散管理システムにおいて負荷は均等であるが全体としては非効率的な場合には，ハッシュ関数を根本的に変更し，非常に多くの索引要素を移動させる必要があり，大規模な再編成が行われる．このような場合における提案手法の優位性を検証する．

5.2.1 シミュレーション

シミュレーションにおける条件として，以下を仮定する．

(a) 開始時には Type 2 のハッシュ関数を用い，HOST からの再編成命令により Type 1 に変更する場合（case a）と開始時には Type 1 のハッシュ関数を用い，HOST からの再編成命令により Type 2 に変更する場合（case b）の 2 種類の再編成を行う．つまり，case 1，case 2 のそれぞれに対して行い，4 つの場合（case 1a，case 1b，case 2a，case 2b）についてそれぞれシミュレーションを行う．

(b) 要求発行間隔は 2500 とし，HOST は 10 の検索要求または更新要求を発行する．処理要求が更新要求である確率（更新確率，P_m）を 0 から 1 まで 0.1 きざみで変更するものとする．

(c) 検索要求における検索確率は 0.1 とする．

(d) 更新要求はオブジェクトの 1 つの参照関係の変更にともなう索引の更新とし，変更対象はランダムに決定する．

(e) 再編成の対象範囲の IP_i の数（NA）を 1 から 8 まで変更してシミュレーションをそれぞれ行う．ただし，対象となる IP_i は IP_9，IP_8，...，IP_2 の順に選択していくものとする．たとえば，$NA=2$ ならば IP_9，IP_8 が再編成の対象となる．ただし，更新確率が 0 ではない場合は再編成の対象範囲を 8 と固定する．再編成対象ではない索引要素に関しては適用されるハッシュ関数の変更は行われないものとする．

(f) HOST は再編成要求の RID を 5 とし，RID が 4 の処理要求を送信後に再編成要求を送信する．

(g) 再編成処理において，各検索 PE において移動対象索引要素を 1 つ探査するのに要する時間を T_r 単位時間とする．

以上の条件において各索引に対し 3 回シミュレーションを行い，その平均を結果とする．

case a，case b とも，移動する索引要素数は全体の $7\times NA/64$ であり，最大で全索引要素の 7/8 が移動することになり，大規模な再編成であるといえる．また，更新要求が多くなった場合，(b) の要求発行間隔

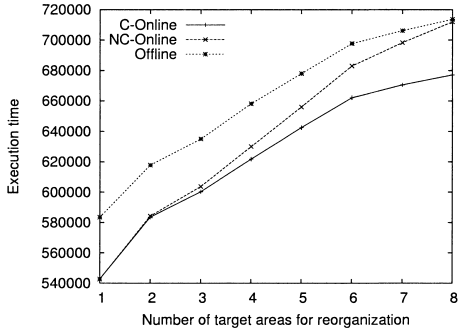

図 4 case 1a における再編成対象範囲と処理時間（$P_m = 0$）
Fig. 4 Target area for reorganization and execution time of case 1a ($P_m = 0$).

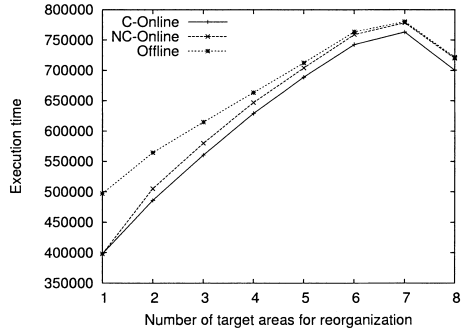

図 5 case 1b における再編成対象範囲と処理時間（$P_m = 0$）
Fig. 5 Target area for reorganization and execution time of case 1b ($P_m = 0$).

によっては再編成処理中には他の処理要求が存在しないこともありうる．そこで，予備実験により 1 つの検索要求に対する処理時間を求め，要求発行間隔をその処理時間の 1/20 以下となる 2500 とした．なお，予備実験による 1 つの検索要求に対する処理時間の平均は case 1 の場合は 56712 であり，case 2 の場合は 131572 となった．

5.2.2 シミュレーション結果

本シミュレーションにおいては，各処理が並列に実行されるため，再編成のみの処理時間の比較ではその他の処理との並列性に対する評価が正しく行えない．よって，本論文においては総処理時間を用いて評価することとした．また，再編成処理後にある程度の要求を処理した後のシステムの状態は，手法にかかわらず，ほぼ同じ状態になり，手法による総処理時間の差は一定になることが予想される．一方，総処理時間自体は再編成以外の要求に対する処理により変化するため，手法による総処理時間の差の比較対象を総処理時間自体とすることはあまり意味のないことである．したがって，総処理時間の差の比較対象として 1 つの検索要求に対する処理時間の平均を用いることとする．

図 4，**図 5**，**図 6**，**図 7** は更新確率を 0 とした場合の再編成対象範囲と総処理時間の関係を表したものである．**図 8**，**図 9**，**図 10**，**図 11** は再編成対象範囲数を 8 とした場合の更新確率と総処理時間の関係を表したものである．また，**表 1** は更新確率が 1 の場合の総処理時間を表にしたものである．ここで C-Online は提案手法，NC-Online は再編成処理を分割せずに行うオンライン手法，Offline はオフライン手法である．

これらの結果より，ほぼすべての場合において提案手法が他の手法より高速であることが分かる．また，更新確率が 0 の場合において，提案手法とオフライン手法の総処理時間の差の平均は case 1 で 39912 あり，case 2 で 85092 であり，予備実験による 1 つの検索要求に対する処理時間が case 1 で 56712，case 2 で 131572 であったことを考慮するならば，提案手法とオフライン手法において有意に差が開いているといえる．ただし，更新確率が 1 の場合は総処理時間にほとんど違いは表れていない．これは，更新は検索に比べて非常に短時間で処理可能であり，再編成要求の到

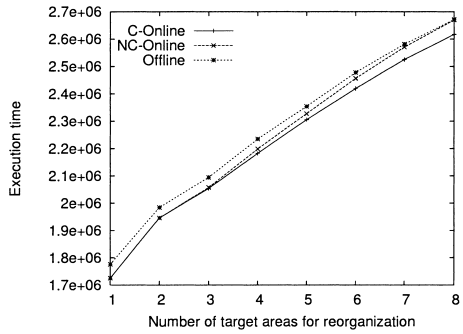

図 6 case 2a における再編成対象範囲と処理時間（$P_m = 0$）
Fig. 6 Target area for reorganization and execution time of case 2a ($P_m = 0$).

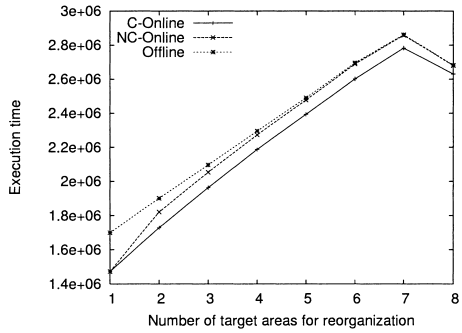

図 7 case 2b における再編成対象範囲と処理時間（$P_m = 0$）
Fig. 7 Target area for reorganization and execution time of case 2b ($P_m = 0$).

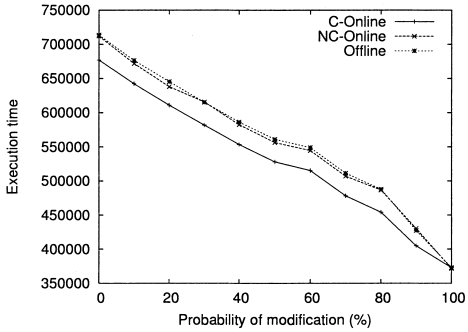

図 8 case 1a における更新確率と総処理時間（$NA = 8$）
Fig. 8 Probability of modification and execution time of case 1a ($NA = 8$).

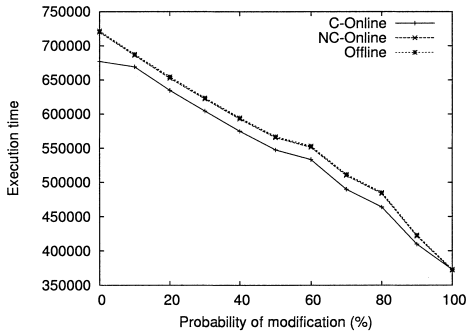

図 9 case 1b における更新確率と総処理時間（$NA = 8$）
Fig. 9 Probability of modification and execution time of case 1b ($NA = 8$).

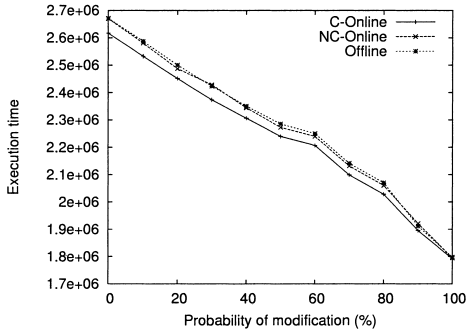

図 10 case 2a における更新確率と総処理時間（$NA = 8$）
Fig. 10 Probability of modification and execution time of case 2a ($NA = 8$).

着時には索引要素の移動が可能な状況であるため，どの手法であっても，各検索 PE における処理手順，タイミングはほとんど同じとなり，総処理時間にほとんど違いは表れなかったことによる．また，再編成処理を分割せずに行うオンライン手法に関しては，対象範囲が広い場合はオフライン手法と同程度の処理時間であり，対象範囲が狭い場合は提案手法と同程度の処理

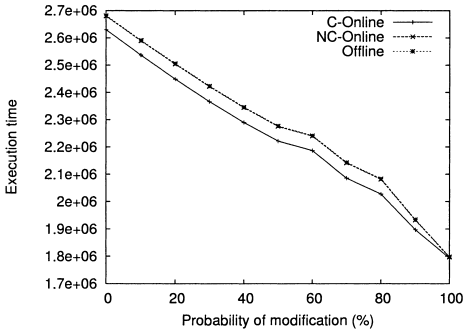

図 11 case 2b における更新確率と総処理時間（$NA = 8$）
Fig. 11 Probability of modification and execution time of case 2b ($NA = 8$).

表 1 更新確率 1 における総処理時間（$NA = 8$）
Table 1 Execution time of the case that $P_m = 0$ ($NA = 8$).

	case 1a	case 1b	case 1a	case 1b
C-Online	372751	372605	1791491	1793270
NC-Online	371614	371876	1794549	1796401
Offline	372890	372891	1797922	1797922

時間である．オンライン処理による効果は対象範囲が狭い場合に現れ，再編成における処理の分割による効果は対象範囲が広い場合に現れることが分かる．つまり，オンライン手法は対象範囲が広い場合にはオフライン手法と比べてロックの範囲に差があまりなく，処理時間の差が小さいが，対象範囲が狭い場合はロックの範囲の差が処理時間の差に現れる．一方，再編成処理を分割する手法は対象範囲が広い場合には他の処理との並列度が高くなり，処理時間に差が現れるが，対象範囲が狭い場合は分割数が少なくなり，処理時間に差はなくなる．したがって，提案手法がオンライン処理による効果と再編成処理の分割による効果を効率的に取り入れた手法であることが分かる．ただし，case b の場合は case a より差が小さくなっている．これは，Type 1 のハッシュ関数の場合，検索 PE が管理する索引要素集合はすべて 1 つの IP_i に属しており，この IP_i に対してロックがされた場合にはまったく検索処理が行えなくなる．したがって，各処理の並列性が阻害され，オフライン手法との差が小さくなったものと推察される．

なお，case b の $NA = 8$ において対象範囲が広くなっても総処理時間が減少する現象は，Type2 のハッシュ関数に完全に変更されていない状態（$NA \neq 8$ のとき）では，一部の検索 PE の管理する索引要素数が非常に減少し，非常に負荷が不均等となる状態が起こり，再編成後の検索処理で逆転現象が起きるためで

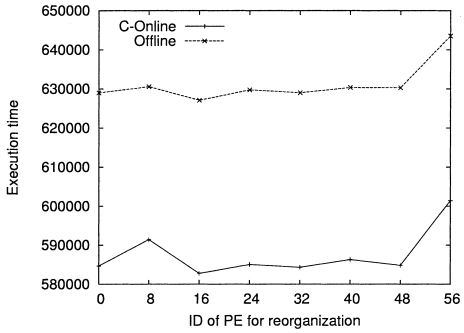

図 12 case 1a における再編成対象 PE と処理時間（$P_m = 0$）
Fig. 12 Target PE for reorganization and execution time of case 1a ($P_m = 0$).

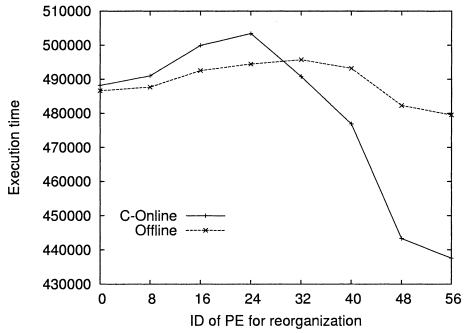

図 13 case 1b における再編成対象 PE と処理時間（$P_m = 0$）
Fig. 13 Target PE for reorganization and execution time of case 1b ($P_m = 0$).

ある．

以上の結果より，大規模な再編成においては提案手法が優位であることは明らかである．

5.3 実　験　2

索引分散管理システムにおいて特定の検索 PE に負荷が集中し，負荷の分散を行うための再編成を行う場合における提案手法の優位性を検証する．

5.3.1 シミュレーション

シミュレーションにおける条件としては以下を仮定する．

(a) 再編成の対象範囲を ID が $8 \times i$ である検索 PE の管理する索引要素とし，i を 0 から 7 まで変更してそれぞれシミュレーションを行う．再編成対象ではない索引要素に関しては適用されるハッシュ関数の変更は行われないものとする．ただし，更新確率が 0 ではない場合は再編成の対象範囲を ID が 0 の検索 PE の管理する索引要素に固定する．

(b) その他の条件は実験 1 と同様とする．

このとき，case a では全索引がロックの対象となり，case b ではただ 1 つの IP_{i+2} がロックの対象となる．したがって，case a の場合は再編成処理を分割せずに行うオンライン手法はオフライン手法とまったく同じ処理となり，case b の場合は再編成処理を分割せずに行うオンライン手法は提案手法とまったく同じ処理となるため，再編成処理を分割せずに行うオンライン手法を比較対象から外すこととし，提案手法とオフライン手法との比較のみを行う．

以上の条件において各索引に対し 3 回シミュレーションを行い，その平均を結果とする．

5.3.2 シミュレーション結果

図 12，**図 13**，**図 14**，**図 15** は更新確率を 0 とした場合の再編成対象範囲と総処理時間の関係を表したものである．**図 16**，**図 17**，**図 18**，**図 19** は再編成対象を ID が 0 の検索 PE とした場合の更新確率と総処理時間の関係を表したものである．

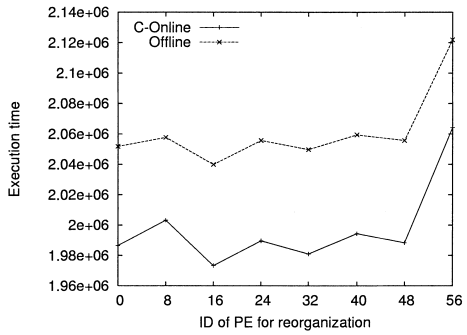

図 14 case 2a における再編成対象 PE と処理時間（$P_m = 0$）
Fig. 14 Target PE for reorganization and execution time of case 2a ($P_m = 0$).

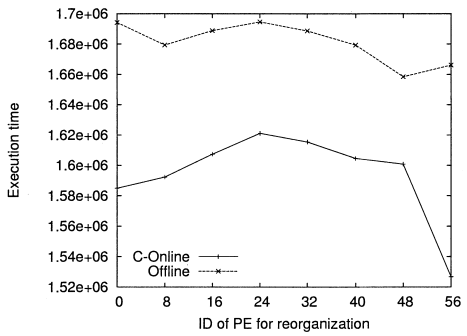

図 15 case 2b における再編成対象 PE と処理時間（$P_m = 0$）
Fig. 15 Target PE for reorganization and execution time of case 2b ($P_m = 0$).

これらの結果より，case 1a，case2a，case2b の場合は提案手法がオフライン手法より優位であることは明らかである．しかし，case 1b の場合には，いくつかの場合はオフライン手法の方が総処理時間が短かっ

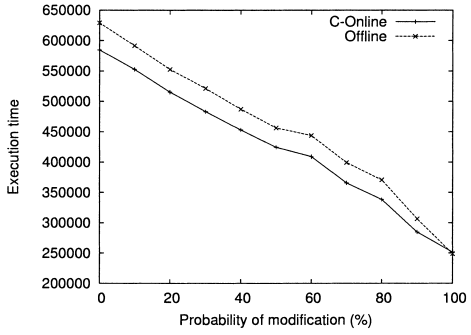

図 16 case 1a における更新確率と総処理時間（PE 0）
Fig. 16 Probability of modification and execution time of case 1a (PE 0).

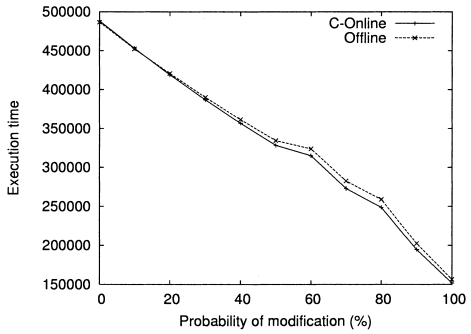

図 17 case 1b における更新確率と総処理時間（PE 0）
Fig. 17 Probability of modification and execution time of case 1b (PE 0).

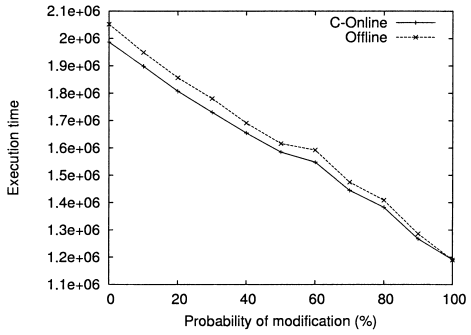

図 18 case 2a における更新確率と総処理時間（PE 0）
Fig. 18 Probability of modification and execution time of case 2a (PE 0).

た．case b の場合，再編成対象の PE が管理している索引要素は 1 つの IP_i に属する索引要素のみであり，提案手法においても再編成処理の分割を行うことはできない．しかも，Type 1 のハッシュ関数の特性によって，ロックが必要となる IP_i は 8 つの PE に占有的に格納されているため，これら PE が再編成処理中は

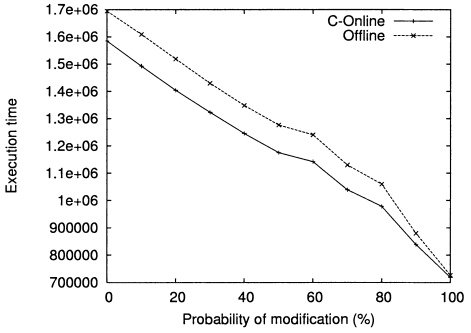

図 19 case 2b における更新確率と総処理時間（PE 0）
Fig. 19 Probability of modification and execution time of case 2b (PE 0).

他の処理がまったく不可能となり，これら PE がボトルネックとなり全体の処理の並列性が阻害されたものと推察される．しかし，このような状況でも提案手法とオフライン手法の差の最大は 8947 であり，1 つの検索要求対する処理時間の平均が 56712 であったことを考慮するならば，提案手法はオフライン手法に大きく劣るとはいえず，その他の場合における時間差を考慮するならば，提案手法の優位性が認められる．以上の結果より，小規模な再編成においても提案手法が優位であるといえる．

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，オンラインで再編成を行うシステムを提案し，その優位性について検証を行った．我々の提案する再編成手法は従来の索引分散管理システムの更新手法を基本とし，大きな変更を行わずに実現可能である．また，索引分散管理システムの最大の利点であるデータ本体との分離も保証している．したがって，データ本体と索引の両方にまたがるような要求に対しても処理の並列性は高まる．また，シミュレーションにより，本システムの特徴であるオンライン処理の優位性と再編成において処理を分割することの優位性について検証した．

しかし，いくつかの点において課題が残る．(a) 再編成をどの状況で行うかの決定．(b) 最適なハッシュ関数の決定．(c) 再編成における最適な処理分割単位の決定．つまり，並列計算機の特性や検索要求の傾向などからどのハッシュ関数に変更すべきかや，どのタイミングで再編成命令を発行するかなど実装上の問題は今後の課題となる．

参 考 文 献

1) Bertino, E. and Kim, E.: Indexing techniques for queries on nested object, *IEEE Trans. Knowledge and Data Eng.*, Vol.1, No.2, pp.196–214 (1989).
2) Salzberg, B. and Dimock, A.: Principles of Transaction-Based On-Line Reorganization, *Proc. 18th International Conference on Very Large Data Bases*, pp.511–520 (1992).
3) Yong, V., Naughton, J.M. and Yu, J.: Storage Reclamation and Reorganization in Client-Server Persistent Object Stores, *Proc. 10th International Conference on Data Engineering*, pp.120–131 (1994).
4) 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).
5) Amsaleg, L., Franklin, M.J. and Gruber, O.: Efficient Incremental Garbage Collection for Client-Server Object Database Systems, *Proc. 21st International Conference on Very Large Data Bases*, pp.42–53 (1995).
6) 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).
7) Omiecinski, E.: Concurrent File Reorganization: Clustering, Conversion and Maintenance, *Data Engineering Bulletin*, Vol.19, No.2, pp.25–32 (1996).
8) 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).
9) Zou, C. and Salzberg, B.: Towards Efficient Online Database Reorganization, *IEEE Data Engineering Bulletin*, Vol.19, No.2, pp.33–40 (1996).
10) Ashwin, S., Roy, P, Seshadri, S., Silberschatz, A. and Sudarshan, S.: Garbage Collection in Object Oriented Databases Using Transactional Cyclic Reference Counting, *Proc. 23rd International Conference on Very Large Data Bases*, pp.366–375 (1997).
11) 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).
12) 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).
13) 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).
14) Lakhamraju, M.K., Rastogi, R., Seshari, S. and Sudarshan, S.: On-line Reorganization in Object databases, *Proc. 2000 ACM SIGMOD International Conference on Management of Data*, pp.58–69 (2000).
15) 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).
16) Lee, M., Kitsuregawa, M., Ooi, B., Tan, K. and Mondal, A.: Towards Self-Tuning Data Placement in Parallel Database Systems, *Proc. 2000 ACM SIGMOD International Conference on Management of Data*, pp.225–236 (2000).
17) 渡辺明嗣, 横田治夫：分散ディレクトリ探索コストを考慮した並列データアクセス偏り制御, 電子情報通信学会論文誌, Vol.J85-D-I, No.9, pp.877–886 (2002).
18) 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).

(平成 16 年 3 月 20 日受付)
(平成 16 年 7 月 6 日採録)

(担当編集委員 有次 正義)

樋口 健（正会員）

平成 4 年電気通信大学電子情報学科卒業．平成 9 年同大学大学院博士後期課程修了．平成 9 年 4 月より福井大学工学部情報工学科助手，現在，同情報・メディア工学科助教授．博士（工学）．この間，オートマトン，分散 OODBMS の研究に従事．電子情報通信学会会員．

都司 達夫（正会員）

昭和 48 年大阪大学基礎工学部電気工学科卒業．昭和 53 年同大学大学院博士課程修了．同年福井大学工学部情報工学科講師．現在，同情報・メディア工学科教授．工学博士．データベースシステム，プログラミング言語の研究に従事．著書 “Optimizing Schemes for Structured Programming Language Processors”（Ellis Horwood）．電子情報通信学会，IEEE，日本情報考古学会各会員．