レプリカを用いた索引分散管理システム

中山　達也† 　樋口　健† 　都司　達夫†

† 福井大学工学研究科　〒 910–8507 福井県福井市文京 3–9–1
E-mail: †{nakayama,higuchi,tsuji}@pear.fuis.fukui-u.ac.jp

あらまし　今日，データベースシステムは大規模なデータを管理し，高速に処理することが求められている．データベースシステムでは検索の高速化のために索引が広く用いられている．一方，分散システムにおいては，処理効率の向上のためにデータの複製（レプリカ）を用いることがある．レプリカを用いることで，検索を高並列に処理可能となり，応答時間の改善が予想される．さらに，データの一部が破損した場合，レプリカを用いることで停止することなく処理の継続が可能となる．しかし，レプリカを用いる分散システムでは，データとそのレプリカの同期が必要となる．本論文において，レプリカを用いた索引分散管理システムを提案する．レプリカを用いた索引分散管理システムでは，索引とそのレプリカの同期を保持することが可能である．並列計算機上に実装したレプリカを用いた索引分散管理システムでの実験により，レプリカを用いたことによる優位性の検証を行う．
キーワード　オブジェクト指向データベース，複合オブジェクト，索引分散管理システム

Distributed Index System using Replicas

Tatsuya NAKAYAMA†, Ken HIGUCHI†, and Tatsuo TSUJI†

† Graduate School of Engineering., University of Fukui　Bunkyo 3–9–1, Fukui-city, Fukui, 910–8507 Japan
E-mail: †{nakayama,higuchi,tsuji}@pear.fuis.fukui-u.ac.jp

Abstract　Today, database systems have to manage large scale data and more quick response is desired. For faster data retrieval, an index is generally used. Using replicas is one way to improve the performance of a distributed system. By using replicas, queries can be processed in highly parallel, and it is expected that the response time is improved. Moreover, a system using replicas can keep on processing without stop, even if some parts of the data are broken. But systems using replicas need the method for data synchronization of replicas with the corresponding original data. In this paper, a distributed index system using replicas is proposed. The distributed index system using replicas can preserve the data end. By experiment of the distributed index system using replicas implemented on paralell computer, the improvement of the performance of the system is proved.
Key words　object-oriented database, complex object, distributed index system

1. はじめに

今日，データベースシステムでは，大規模で複雑なデータを管理することが求められている．同時に，これらデータに対する高速な処理が要求されている．オブジェクト指向データベースでも同様であり，複雑な参照関係を持つ複合オブジェクト集合を効率よく管理し，それらを高速に検索する手法が要求されている．その 1 つの解決法として, [1] では複合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，その索引をメッセージ通信非共有メモリ型並列計算機上に分散配置し，並列処理を行うことで高速化を図る索引分散管理システムが提案され，検証されている．索引分散管理システムでは，オンライン更新 [1]，オンライン再編成 [2] [3] [4] が可能であり，各要求に対して高並列に処理可能となっている．しかし，データが大規模になり参照関係の数が大量となる場合は，索引に対する処理も非常に高負荷になることが予想される．また，索引はデータ本体から再構築可能ではあるが，非常に大規模な索引を再構築することは高コストであり，再構築中に索引を使用できないことによる不利益が非常に大きくなると予想される．したがって，データ本体のシステムと同様に，索引システムにおいても障害時に停止することなく稼動することが求められる．

そこで本論文では，データ本体のシステムがデータのレプリカを用いることでこれらの状況に対処するのと同様に，索引のレプリカを用いることで高速化と耐故障性の向上を目的とする，レプリカを用いた索引分散管理システムを提案する．

一般的に，レプリカを用いたシステムにおいては，データと

レプリカの同期が問題となる [5] [6]．したがって，データ更新にはレプリカを同期させるための何らかの処理が必要となり，それによって他の処理に影響が及ぶ場合がある．しかし，索引分散管理システムにおいては，オンライン更新機能が提案されており，データと索引の同期性と一貫性を保持するために，指定された順序で各要求を処理することが可能である．この更新手法を用いることにより，索引更新時に同期性を崩すことなくレプリカの更新を行うことが可能である．並列計算機上に実装したレプリカを用いた索引分散管理システム上での実験により，レプリカを用いることによる検索処理の効率化と，索引更新時の同期のための処理による他の処理への影響の検証を行う．

2. 複合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，索引及び検索についての定義を行う．

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 をこの検索パスの長さと定義する．ここで，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．各 A_j は単一の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，A_j がクラス C_i の属性値として複数のオブジェクトを参照することもできる．また，P の値を次のように定義する．

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

$o_1 o_2 \cdots o_N$ はそれぞれ $C_1, C_2, \cdots, C_N$ のインスタンスである．$o_j (1 < j \leqq N)$ はオブジェクト o_{j-1} の属性 A_{j-1} についての値であり，o_{N+1} は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 $C_i (1 \leqq i \leqq n)$ はそれぞれ異なるクラスとする．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同士の参照関係であるか，内包関係であるかは，索引上において何ら違いはないため，オブジェクト間における関係の種類は特に限定しない．

2.2 マルチインデックス

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はデータ値である．ただし，実際の索引要素では，キー値，データ値共に OID を用いる．また，IP を全ての索引要素の集合とし，クラス c_i のインスタンスである OID をキー値とする索引要素の集合を IP_i，A_N の値をキー値とする索引要素の集合を IP_{N+1} とする．マルチインデックスにおける検索要求として「属性 A_N の値が要素 o_{N+1} であるようなオブジェクトを，検索パス式 P において参照している C_1 のインスタンスを求める」が与えられた場合，それは，

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

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

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

のようにオブジェクトの逆参照関係から求めることである．

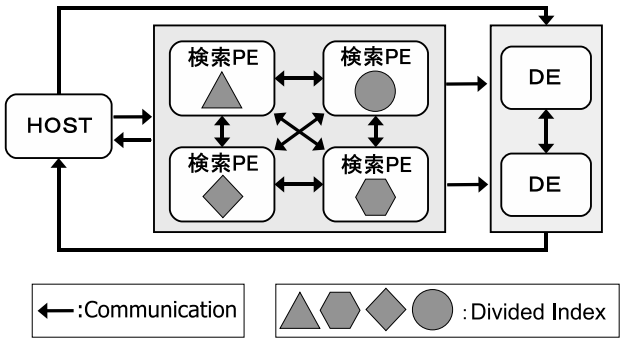

図 1　索引分散管理システム
Fig. 1　Distributed Index System

3. 索引分散管理システム

以下では，索引分散管理システムの構成と検索・更新処理について述べる．索引分散管理システムを構築する環境としては，メッセージ通信非共有メモリ型並列計算機を仮定する．対象オブジェクトのクラスとそのインスタンスには，それぞれクラス ID（以下，CID）とインスタンス ID（以下，IID）を割り当て，この 2 つを対にしたものを OID として構成する．また，各プロセッサエレメント（以下，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において，負荷分散並列処理を行うために IP を分割し，各 PE にそれぞれ格納して管理する．ただし，索引を分割・分散配置する際に「同じキー値の索引要素は同じ PE で管理する」という以外の制限は加えない．したがって，IP_i に対して様々な種類の格納を行えるため，自由な索引分割が可能となる．

3.1 システムの構成

索引分散管理システムにおいて，PE はその処理内容により HOST，検索 PE，DETECTOR の 3 種類に分類される．以下に，それぞれの PE における処理の概要を示す．また，図 1 は索引分散管理システムの処理の流れを表す．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，その索引の元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．

3.1.1 HOST

検索・更新要求など外部からの処理要求を検索 PE に送信し，処理結果を集計して最終的な終了判定を行うための 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（以下，

RID）を発行する．RID には自然数を用い，処理要求に対し昇順に付加する．また，処理要求として送信したメッセージの情報を DETECTOR に送信する．実験で使用するシステムでは，外部からの処理要求を受信してから検索 PE に処理要求を送信するまでの各処理と，処理結果を集計して最終的な終了判定を行う処理を，2 つの PE を用いてそれぞれに処理させることで HOST における処理負荷を分散させる．

3.1.2 検 索 PE

分割された複合オブジェクト索引を管理し，実際に検索・更新要求の処理を行う PE である．各要求の処理手順については，3.2 節，3.3 節で示す．

3.1.3 DETECTOR

DETECTOR（以下，DE）は，HOST から発行された要求に対する処理の終了判定を行うための PE である．DE は，HOST，検索 PE，DE より得られた 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 毎に IP_i の処理が終了したかどうかを判定する．これは，検索パス式の検索処理の方向が一定であることから，「IP_{i+1} に対し行われた検索によって得られる検索結果の OID 数は，その次に検索を行う IP_i に対する検索要求の OID 数と同じ」ということを利用する．つまり，IP_i に関する検索処理は，IP_{i+1} に関する処理が全て終了し，かつ，IP_i に対する検索要求のメッセージ総数と 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場合に限り終了したと判定される．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 への検索要求メッセージ」は HOST に送信する検索結果である．この性質を用いて検索の終了を判定する．なお，本論文においては，1 つの処理要求についての終了判定は 1 つの DE で全てを行う．このとき，ある RID を付加された処理を管理する DE は，ハッシュ関数を用いて決定する．

3.2 検索処理

索引分散管理システムにおいて分割された索引に対する検索処理は，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．また，各検索 PE に送信したメッセージの総数を DE に送信する．このメッセージの総数は，DE が各検索 PE より受信するメッセージの総受信予定数となる．

（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 を自分が管理する IP の部分索引で検索する．

（3） 各検索 PE において（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．また，各検索 PE は，到着した受信メッセージの総数と，他の検索 PE に送信したメッセージの総送信数を DE に送信する．DE は，HOST 及び検索 PE より受信したメッセージを元に IP_i 単位で終了判定を行う．

（4） 各検索 PE に対して検索要求がある限り（2）と（3）を繰り返す．

（5） 全ての検索 PE の処理が終わり，HOST に全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) 及び（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．本論文では，ハッシュ関数を用いて索引の分割を決定し，それと同じハッシュ関数を用いて送信先 PE を決定する方法を用いる．

3.3 更新処理

索引の更新に関しては以下の（1）～（6）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．更新としては，データの整合性を保つために「更新要求の RID より小さな RID を付加した要求全てにおいて，更新対象となる索引要素が属する IP_i に対する処理が終了すること」が求められる．

（1） HOST は，更新要求に対して検索要求と同じように RID を付加する．

（2） HOST は，更新要求を更新対象の索引を管理する検索 PE に直接送信する．また，更新要求に付加した RID を管理する DE に，更新要求が送信されたことを知らせるメッセージを送信する．

（3） HOST からメッセージを受信した DE は，他の DE に対して上述の更新条件を満たしているか確認のメッセージを送信する．他の DE は，この条件が満たされたときに，確認メッセージを送信してきた DE に対してメッセージを返す．全ての DE において更新条件が満たされたとき，更新要求を管理する DE は，更新対象の索引を管理する検索 PE に更新許可を送信する．

（4） 更新要求を受信した検索 PE は，DE からの更新許可を受信するまで更新をせず，更新許可を受信してから更新を行う．なお，更新許可を受信するまで更新要求を保持している検索 PE は，以下の条件の場合のみ検索を行う．

- 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．

（5） 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 HOST，DE に伝える．

（6） 更新終了のメッセージを受信した DE は，HOST に更新終了メッセージを送信する．

この終了判定及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，他の索引要素には全く無関係に処理が可能となる．また，検索 PE につい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，処理要求を保留として取り扱う．

4. レプリカを用いた索引分散管理システム

索引のレプリカ（以下，レプリカ索引）とは，索引分散管理

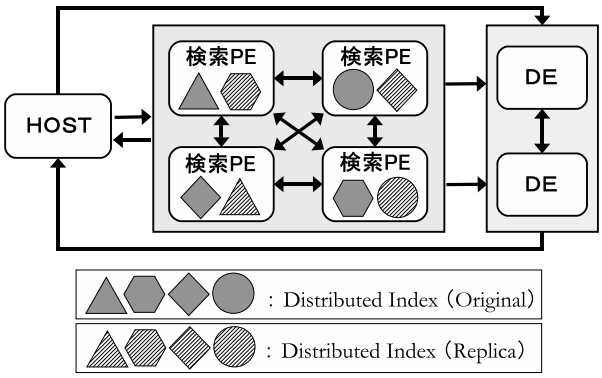

図 2　レプリカを用いた索引分散管理システム
Fig. 2　Distributed Index System using Replicas

システムで管理している索引 IP（以下，オリジナル索引）を複製したものである．説明のため，オリジナル索引，レプリカ索引と呼び分けを行うが，これらは全く同等なものであり処理方法や管理方法の違いはない．また，レプリカを用いた索引分散管理システム（図 2）では，オリジナル索引に対して複数のレプリカ索引を用いることもでき，オリジナル索引を含めた各索引にはそれぞれ異なる索引番号が付加されているものとする．また，オリジナル索引とレプリカ索引の総数を索引数と呼ぶことにする．以下では，レプリカを用いた索引分散管理システムにおける各索引の配置及び検索・更新処理について記述する．

4.1　索引の配置

索引分散管理システムにおいて，索引要素の配置条件は「同じキー値の索引要素は同じ PE で管理する」という制限のみであった．レプリカを用いた索引分散管理システムではこれを変更して,「同じキー値の索引要素は同じ PE で管理する．ただし，オリジナル索引の索引要素とその索引要素を複製したレプリカ索引の索引要素は，各々異なる PE で管理しなければならない」と定める．これにより，1 つの索引要素を複数の PE が管理することになるので，処理を分散させることができる．各索引の分割の決定には索引番号によって異なるハッシュ関数を用いて分割の決定を行うことで実現される．

4.2　レプリカを用いたシステムにおける検索処理

索引分散管理システムでは，1 つの索引要素は 1 つの検索 PE に管理されているので，検索要求の送信先は指定された検索要求中の OID と索引分割時に用いたハッシュ関数によって決定されていた．一方，レプリカを用いたシステムでは，複数の検索 PE が同じ索引要素を管理している．そのため，検索要求を送信する際に同じ索引要素を管理する複数の検索 PE から 1 つを選択（以下，経路選択）する必要がある．なお，検索処理の手順は経路選択を行うこと以外は，3.2 節で示した索引分散管理システムの検索処理手順と同じである．以下では，経路選択方式について示す．

4.2.1　経路選択方式

本論文では，経路選択として 2 種類の方式を提案する．1 つ目の方式は，HOST および各検索 PE において OID 毎に検索要求送信先を，対象となる索引要素を管理する検索 PE の中からランダムに決定する方式（以下，Random 方式）である．2 つ目の方式は，各検索要求においてそれぞれ指定された索引のみを用いて処理を行う方式（以下，Group 方式）である．各検索要求に用いられる索引は，HOST においてオリジナル索引，レプリカ索引に付加した索引番号を順に選択することで指定される．また，レプリカを用いたシステムでは次に使用する索引要素が同じ検索 PE 内に存在し，それが上述の方式で選択した索引要素とは異なる場合がある．このとき，同じ検索 PE 内にある索引要素を優先的に使用する場合は，他の検索 PE へ送信するメッセージサイズの縮小，あるいは，通信回数の削減が見込める．一方で，Random 方式，Group 方式で選択した通りの索引要素を使用する場合は，検索 PE に対する検索処理の負荷を分散させることが見込める．

実験では，Random 方式，Group 方式のそれぞれに対し，同じ検索 PE 内にある索引要素を優先的に使用する「通信コスト削減型」と，経路選択方式で選択した通りの索引要素を使用する「処理負荷分散型」を考慮して，以下の 4 種類の方法を検証する．

- 通信コスト削減型：Random 方式
- 処理負荷分散型　：Random 方式
- 通信コスト削減型：Group 方式
- 処理負荷分散型　：Group 方式

4.3　レプリカを用いたシステムにおける更新処理

一般的な分散システムにおける更新処理は，更新を行うデータ全てに対しロックを取得するなどの同期処理を行い，同時に更新処理を行うといった方法が用いられる．しかし，この方法ではレプリカ数が増えた場合には，非常に大規模な索引部分のロックが必要になり，データ更新時の他の処理への影響が大きくなることが予想される．そこで，これらの同期処理時の問題を解決するために，更新対象となる各索引の索引要素を異なるものとして扱い，それらに対する更新を連続に行うことにより，更新対象の索引要素を一括してロックすることなく更新処理を行う方法を用いる．つまり，HOST において更新要求を各索引の更新対象となる索引要素数分コピーし，それら更新要求に対して連続する RID を付加する．この処理を，3.3 節で示した索引分散管理システムにおける更新処理の手順（1）に加える．索引分散管理システムにおいては，RID 順と等価な処理順で処理が行われるため，連続する RID の更新は連続して行われたのと等価な順で処理される．したがって，同じ参照関係から作成された索引要素集合は，同時に更新されたのと等価となる．また，索引分散管理システムはオンライン更新が可能であるため，これらの更新処理の連続要求は並列に処理され，他の処理への影響をほとんど及ぼさないことが予想される．

5.　評 価 実 験

ここでは，提案したレプリカを用いた索引分散管理システムを実装し，従来の索引分散管理システムとの比較による評価結果を示す．なお実験は，並列計算機 Sun Fire E4900（コア数：24，メモリ：48GB）を使用して評価を行う．

5.1　評 価 条 件

評価実験における条件を以下に示す．

5.1.1 対象オブジェクト

以下の複合オブジェクト集合を対象オブジェクトとして用意する．

- 索引対象となる複合オブジェクトは，クラス数を 6，各クラスのインスタンス数を 500000.
- 各オブジェクトは 0 から 2 個のオブジェクトを参照する．オブジェクトの参照数及び参照先オブジェクトはランダムに決定され，2 個の場合は異なるオブジェクトを参照する．

5.1.2 索引分散管理システム

実験で使用する索引分散管理システムは，以下のような構成からなる．

- HOST を 2 個（要求発行用と終了判定用），検索 PE を 12 個，DE を 3 個用意する．
- 検索要求は随時 HOST に送信されてくるものとする．
- 通信に MPI ライブラリ，索引要素の格納には 2 次記憶上の B^+ 木を使用する．

また，レプリカ索引は 0 から 5 個使用し，レプリカ索引数が 0 個の場合を従来の索引分散管理システムの処理結果とする．

5.1.3 評 価 項 目

以下のような項目について，4.2.1 項で記述した 4 種類の経路選択方式それぞれにおける，従来の索引分散管理システムとレプリカを使用した索引分散管理システムとの比較評価を行う．

（1） 検索要求のみの場合についての比較

・ 検索要求数：50

・ 検索要求 1 件に含まれる OID 数：2500(Case a-1), 5000(Case a-2)，7500(Case a-3)，10000(Case a-4) の 4 種類

（2） 検索要求と更新要求を発行する場合の比較

・ 検索要求数：50，更新要求数：5

・ 検索要求 1 件に含まれる OID 数：2500(Case b-1), 5000(Case b-2)，7500(Case b-3)，10000(Case b-4) の 4 種類

・ 検索要求に対して更新要求を等間隔に発行する．

（3） 一部の索引に対する検索要求が集中した場合の比較

・ 検索要求数：5000(Case c-1)，10000(Case c-2) の 2 種類

・ 検索要求 1 件に含まれる OID 数：5

・ 各検索要求に含まれる OID は，Case c-1，Case c-2 のどちらも全て同じ値とする．

5.2 実 験 結 果

Case a-1 から Case c-2 までの各実験の結果を，図 3 から図 12 にそれぞれ示す．ここで，図における「Random1」,「Random2」,「Group1」,「Group2」はそれぞれ順に,「通信コスト削減型：Random 方式」,「処理負荷分散型：Random 方式」,「通信コスト削減型：Group 方式」,「処理負荷分散型：Group 方式」のことを表す．

図 3，図 4，図 5，図 6 より，Random 方式，Group 方式のどちらも通信コスト削減型では処理速度の向上が見られ，処理負荷分散型では従来の索引分散管理システムとほぼ同じ処理時間となることが分かる．これは，検索要求に含まれる OID 及び索引対象となる複合オブジェクトの参照関係をランダムに決定していることや，各検索要求に含まれる OID 数が同じであることから，どの方式においても検索処理の負荷が十分に分散されていることが考えられる．そのため，処理負荷分散型の特性が処理速度に反映されなかったものと考えられる．通信コスト削減型においては，上述にある処理負荷が十分に分散していることに加え，使用する索引を決定する際の計算処理が少ないため，Group 方式の方が Random 方式より多少良い結果が得られた．また，要求に含まれる OID 数が大きくなると，通信コスト削減型の Random 方式，Group 方式においても，処理速度向上の割合が減少することが分かる．これは，全ての検索

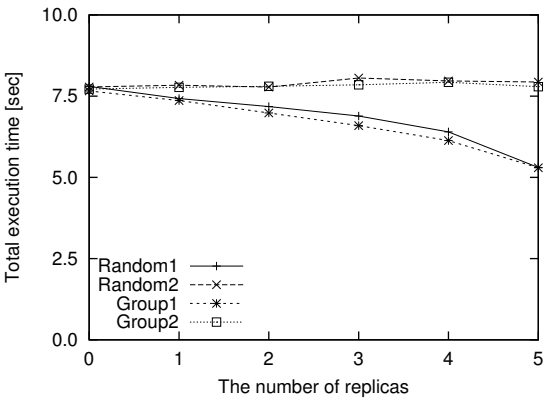

図 3 実験結果 (Case a-1)

Fig. 3 Experimental Result (Case a-1)

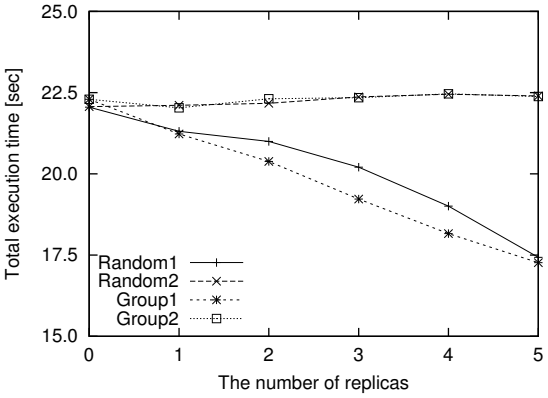

図 4 実験結果 (Case a-2)

Fig. 4 Experimental Result (Case a-2)

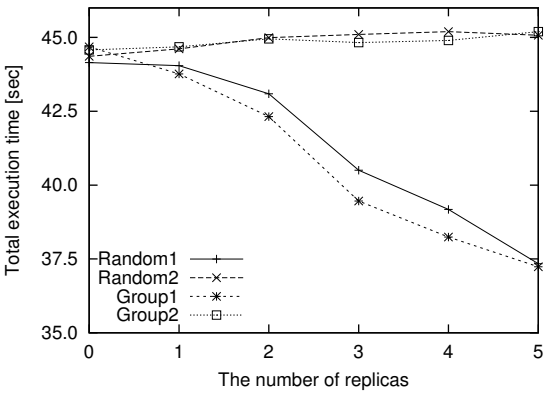

図 5 実験結果 (Case a-3)

Fig. 5 Experimental Result (Case a-3)

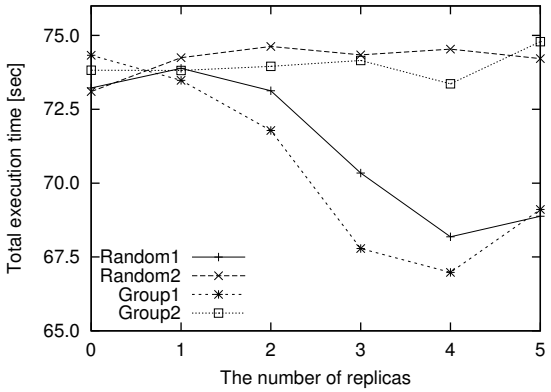

図 6　実験結果 (Case a-4)
Fig. 6　Experimental Result (Case a-4)

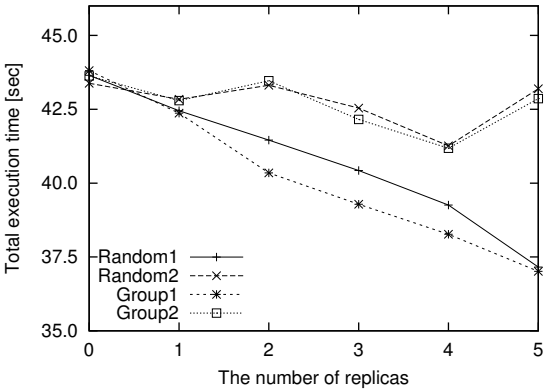

図 9　実験結果 (Case b-3)
Fig. 9　Experimental Result (Case b-3)

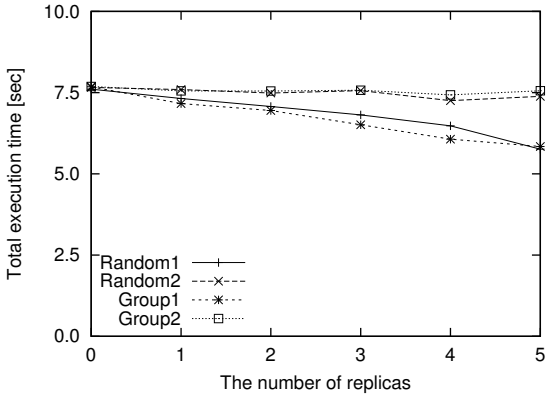

図 7　実験結果 (Case b-1)
Fig. 7　Experimental Result (Case b-1)

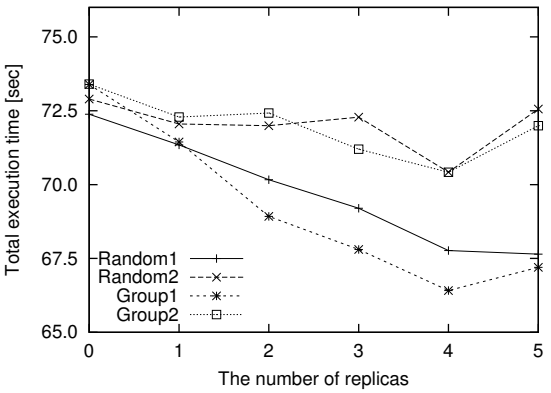

図 10　実験結果 (Case b-4)
Fig. 10　Experimental Result (Case b-4)

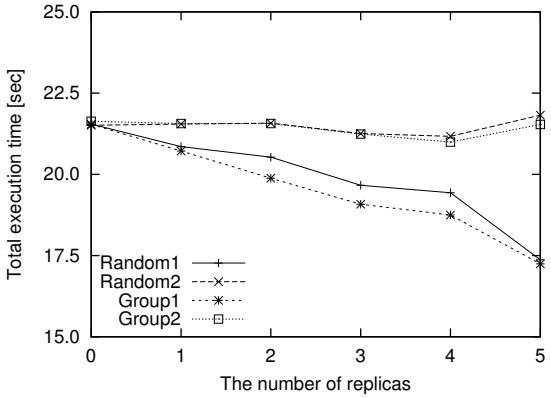

図 8　実験結果 (Case b-2)
Fig. 8　Experimental Result (Case b-2)

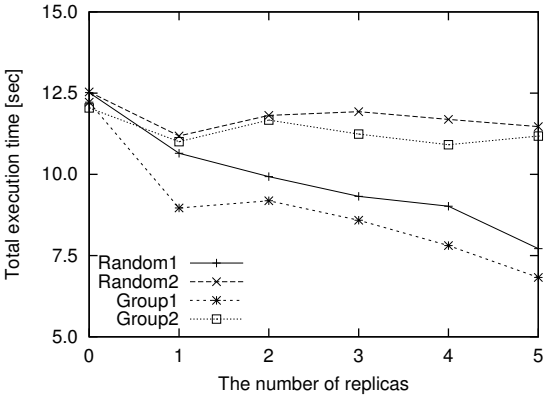

図 11　実験結果 (Case c-1)
Fig. 11　Experimental Result (Case c-1)

PE に対して常に検索要求が送られてきている状態となることから，従来の索引分散管理システムにおいても要求待ちによるアイドル状態がほとんど起こらないためと考えられる．

更新要求を含む場合についての処理結果である図 7 から図 10 は，それぞれ図 3 から図 6 とほぼ同じような結果となった．ただし，比較的負荷が低い Case a-1 と Case b-1 との比較では，索引数が多い場合は総処理時間にやや悪化が見られる．これは，索引数が増えたことにより，多くの検索 PE が複数の索引を管理することになり，複数の更新要求を連続的に行わなければならなくなるため，負荷の低いような状況ではその差が現れたものと思われる．したがって，全体的な傾向としては，レプリカを用いたシステムにおける更新処理のコストは，従来のシステムと同様に非常に小さなものとなり，レプリカ索引を用いることで起こる同期処理などは処理時間にあまり影響しないことが分かる．

負荷に偏りが生じるような検索要求を与えた Case c-1，Case c-2 の場合，図 11，図 12 より，全ての経路選択方式においてレプリカ索引を用いたシステムは，従来のシステムに比べて処

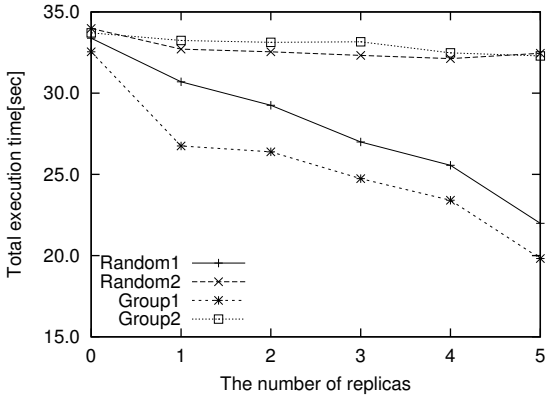

図 12　実験結果 (Case c-2)
Fig. 12　Experimental Result (Case c-2)

理時間が短縮していることが分かる．これは，レプリカ索引を用いることで検索処理の負荷が分散し，システム全体の処理効率が向上したものと考えられる．

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，レプリカ索引を各処理に用いることを提案し，従来の索引分散管理システムとの比較による評価・検証を行った．レプリカ索引を索引分散管理システムに用いることで，通信回数の削減や負荷分散といった要因により，検索にかかる処理時間の短縮を図ることができた．また，レプリカ索引を用いることにより考えられる更新処理時の同期などの問題に対しては，従来の索引分散管理システムの特徴であるオンライン更新機能を利用することで，新たな同期処理を加えることなく索引の一貫性を保つことができ，他の処理への影響も大きくないものとなった．

今後の課題として，レプリカ数が多くなった場合の更新時の効率化と，再編成処理時における応答時間の向上について検証する．また，検索 PE 数を変化させた場合における処理時間の影響についても検証する．

文　献

[1] 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）．
[2] 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）．
[3] 田村弘行，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，信学技報，2005-DBS-135（5），pp.31-38（2005）．
[4] 野村英憲，樋口　健，都司達夫，索引分散管理システムにおける段階的再編成，電子情報通信学会：DEWS2006，3C-i11（2006）．
[5] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, G. Alonso, Understanding Replication in Databases and Distributed Systems, Proceedings of the The 20th International Conference on Distributed Computing Systems (ICDCS 2000), pp.464-474 (2000).
[6] Takahiro Hara, Location Management of Replicas Considering Data Update in Ad Hoc Networks, 20th International Conference on Advanced Information Networking and Applications (AINA 2006), pp.753-758 (2006).