データベースシステム 121－8
情 報 学 基 礎 58－8
（2000. 5. 25）

検索と更新の並列処理可能な複合オブジェクトに対する索引の分散管理システム

樋口 健†, 山口誠司‡, 都司 達夫†, 宝珍 輝尚†

† 福井大学 工学部 情報工学科
〒 910-8507 福井市文京 3 丁目 9 － 1

‡ (株) アクトプレーン

複合オブジェクトに対する索引を分割並列処理することで性能の向上が見込まれる. 本研究は検索と更新が並列処理可能な複合オブジェクトに対する索引システムの構築を目的とする. 我々の提案する手法において索引が更新されたとき, 全ての索引がロックされるのではなく, 分割された索引の 1 つの部分のみをロックする. したがって, より処理の並列化による性能の向上が見込まれる. また, 検索シミュレーションにより我々の手法の性能を評価を行なう.

Parallel Index System for Complex Objects with On-Line Modification

Ken HIGUCHI†, Seiji YAMAGUCHI‡, Tatsuo TSUJI†, Teruhisa HOCHIN†

†Dept. of Information Science, Faculty of Eng., Fukui University
3-9-1, Bunkyo, Fukui-shi, Fukui 910-8507

‡Actplane Corporation

This paper concerns the parallel retrieval and modification of index for complex objects. An index is divided into sub-indexes each of which is placed on a separate machine in order to take advantage of parallelism. In this paper, we propose a new technique of retrieval and modification of split index for complex objects. Our technique requires locking to one partition of index in modification, but does not require that to whole index. Therefore our system would take further advantage of parallelism. From the retrieval simulation result, it will be also proved the performance of our technique.

1 はじめに

オブジェクト指向データーベースシステムやマルチメディアデータベースシステムでは, 実世界の情報の論理構造を直接反映させるために, 内部に構造を持つデータ型や他のオブジェクトを参照する複合オブジェクトを扱う必要がある. 複雑で大規模な複合オブジェクトの集合を取扱う必要が増加する現在では, 複合オブジェクトを効率的に操作することや管理する技法を確立することは重要な問題である.

複合オブジェクト集合に対する大容量索引については, 複合オブジェクトの実体集合をクラスタリングして分散配置し, 並列操作を行なうことにより効率向上が見込まれる. このような分割配置の一方式として, 大容量索引の並列操作による検索効率の向上を目的とした最適な水平垂直分割方式と並列処理方式が提案されている [1][4]. これら研究においては並列計算機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, 通信経路として2次元トーラス型を仮定し, 検索方式をマルチインデックス方式 [5, 6] として検索スループットに関する静的な近似評価式が提案されている. しかし, これら研究においては索引の更新は考慮されておらず, 検索処理の並列性を損なわない更新方法が求められる.

我々の提案する更新手法は, 索引全体をロックするのではなく, 更新の対象となる索引要素を格納する分割された索引部分のみをロックする. これにより, 更新中でも他の索引部分における処理は実行可能であり, 処理の並列性は損なわれない. 我々の手法と索引全体をロックする手法を検索シミュレーションを行なうでその有効性の考証を行なう.

2 対象オブジェクト

以下では対象とするオブジェクトおよび処理環境等の定義を行なう.

2.1 複合オブジェクト

複合オブジェクトのパスを

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

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. また, P の値を

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

とする. ここで, o_1 は C_1 のインスタンスであり, o_j $(j > 1)$ はオブジェクト o_{j-1} の属性 A_{j-1} の値であり集合値を許す. また, o_{N+1} は単純値またはオブジェクト ID(以下, OID), $o_1, o_2, \ldots, o_N$ は OID とする.

P の値の集合を O_P と表記する.

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なわれる.

オブジェクト o_j の属性 A_j の値が o_{j+1} であるとき,

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

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. さらに, IP を全ての索引要素の集合とする. また, オブジェクト o をキー値とする索引要素の集合を $IP(o)$ とし, その集合を IPO とする.

2.3 並列処理環境

索引検索を行なう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, 通信経路として $M \times N$ の2次元トーラスモデルを仮定する. ただし, N は複合オブジェクトのパスの長さである. i 行目の j 列目のプロセッサエレメント (以下, PE) を PE_i^j と表し,

$$H_i = \cup_{1 \leq j \leq N} PE_j^i,$$

$$V_j = \cup_{1 \leq i \leq M} PE_j^i$$

とする. これは並列計算機をそれぞれ水平方向, 垂直方向に分割するもので, それぞれ H 分割, V 分割と呼ぶ. また, 各 V_j に対して $no(V_j) = j$ とする. また, 上記 PE 群とは別に, 検索, 更新要求を出し, 検索結果を集計するためのす特別な PE として,HOST が存在すると仮定する. したがって検索時間は HOST

から検索要求が発せられ, 検索結果が HOST に送信されるまでの時間である. また,HOST から送られる検索, 更新要求には固有の要求識別子 (RID) がつけられているものとする.

2.4 索引分割

索引検索を上記並列処理環境で並列処理するには, IP を分割し, 各 PE にそれぞれ格納する必要がある. 各 PE は各索引要素ごとに検索結果のオブジェクトをキー値とする索引要素がどの PE に格納されているか知る必要がある. したがって, 各 PE は以下のような組を索引要素として格納していることになる.

$$\langle pe, o', o \rangle$$

ただし, $\langle o', o \rangle \in IP$ であり, pe は o' をキー値とする索引要素が格納されている PE である. しかし, IP をそのまま分割することは各オブジェクト o' に対して pe が一つに定まるとは限らない. これは, 実際に PE に格納される索引要素の総数が IP の要素数より多くなる, すなわち, 実質的に索引要素数が増える可能性があることを意味する. また, 通信の交差による干渉を極力押さえ, 検索結果を出力する PE を極力少なくする点を考慮し, 索引の分割において以下の制限をつける.

制限

1. 各オブジェクト o に対して全ての $p \in IP(o)$ は同じ PE に格納される.
2. o_j, o_{j+1} がそれぞれ C_j,C_{j+1} のインスタンスであり, かつ $\langle o_j, o_{j+1} \rangle \in IP$ ならば, $\langle o_j, o_{j+1} \rangle$ は V_j に格納される.

1. の制限より, 索引分割は IP の各要素を分割するのではなく IPO の各要素を分割することになり, PE に格納される索引要素の総数と IP の要素数が等しくなる. そこで, 各 $op \in IPO$ に対してそれを格納している PE を $PE(op)$, それを格納している V 分割を $V(op)$ で表す.

また, 2. の制限により, 索引検索を行なう PE の V 分割間の通信方向が一定で, しかも隣接する V 分割間でしか通信を行なわないため通信の交差が少なくなり相互干渉が少なくなることが予想される.

3 検索

以上のような索引分割において検索は以下のように行なわれる.

1. HOST から V_N の各 PE に対して検索要求が送られる.
2. 各 PE において到着した要求に対して検索を行う.
3. V_1 に属す PE ならば検索結果を出力. 他の PE ならば, 検索結果を検索要求としてそれを管理する PE に送信.
4. その検索に対する全ての処理が終了していないならば,2 へ.
5. 全ての PE において検索が終了したならば,V_1 からの出力された全ての結果を最終的な出力結果として出力.

これらの手順を HOST から送られる検索要求ごとに行なう.

ここで問題となるのは,4 の検索の終了をいかに判定するかである. 検索は分散並列処理されているため, ある検索要求に対する全ての PE の処理が全て終わったかどうかを知ることは難しい. そこで我々のシステムではメッセージの送受信数を利用して検索の終了判定を行なう.

3.1 終了判定

検索を各 V_i ごとに考えてみると, 検索が全て終了した時点においては, HOST が送信した検索要求メッセージ数と V_N が受信したメッセージ数は同じでなければならない. 同様に $V_i(1 \leq i < M)$ が受信した検索要求メッセージ数は V_{i+1} 送信したメッセージ数と同じでなければならない. つまり, 検索要求に対する V_{i+1} の処理が終了し, V_i へ送信したメッセージの合計数がわかれば, その検索要求に対する V_i の処理が終了したかどうかを判定することが可能である. また,V_N に関しては HOST が送信したメッセージの数が分かれば処理が終了したかどうかの判定を行なうことが出来る. 従って, 各 PE において送受信したメッセージ数を記録し, 集計することで各 PE がある検索要求に対して処理が終了したかどうかを判定可能となる.

上記終了判定を行なう特別なPEとして,本システムにおいてはdetectorを導入する．detectorは各V分割ごとに1PE用意し,それぞれ対応するV分割に関する終了判定を行なう．ここで,V_iを管理するdetectorをD_iと呼ぶことにする．各PEはそれを管理しているdetectorに対し,自分が受け取ったメッセージ数とそれを処理して送信したメッセージ数を送り,各detectorでは検索要求ごとにその送受信数を集計しておく．そして,D_iにおいてV_iが受信したメッセージとD_{i+1}から送られて来たV_{i+1}が送信したメッセージ数が一致した場合,V_iにおけるその検索要求に対する処理は終了したことになり,D_{i-1}に対して検索の終了信号と送信したメッセージ数を送信する．また,HOSTは検索要求メッセージ数をD_Nに送信し,D_1はHOSTに対して送信メッセージ数を送り,HOSTではD_1からの情報と受信したメッセージ数を比較することでHOSTおよび各V分割において検索が終了したかどうかの判定が可能となる．

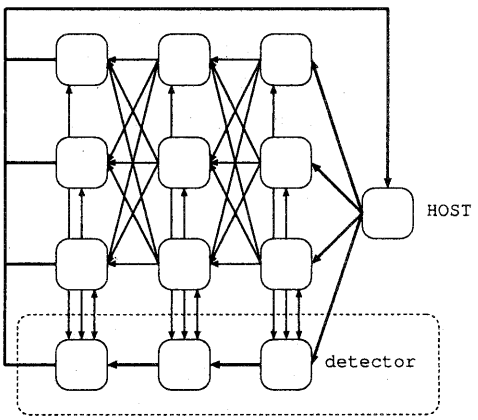

図 1: Index Retrieval System

3.2 重複検索の削除

複合オブジェクトにおいて属性値に集合を許す場合,同じ検索要求内で同じOIDに対する検索が行なわれる場合があり,検索が重複して行なわれる可能性がある．例えば,オブジェクトoがオブジェクトo'とオブジェクトo''の2つのオブジェクトを参照している場合,索引要素としては

$$\langle o, o' \rangle, \qquad \langle o, o'' \rangle$$

の2つが存在する事になり,o'とo''の両方が検索要求に含まれる場合,検索結果にoが重複して2回現れることなる．また,そのまま検索を進めるとo以降の検索処理は全て2回行なわれてしまう．しかし,これら重複して実行される処理は冗長な処理であり,処理を行なわなくても最終的な検索結果には何ら影響は無い．したがって,これらの重複した処理を行なわないために各PEにおいてバッファを用意し,それまでに検索したOIDを保管し,今までに検索していないOIDのみ検索を続行するようにする．そして,検索が終了した場合にそのバッファを解放して検索処理の終了となる．PEではHOSTからの検索要求ごとにバッファを生成し,それぞれの検索ごとに以上の処理を行なう．したがって,各PEには複数のバッファが生成されることになり,検索が終了した場合にはすみやかにバッファが解放されることが望ましい．そこで,detectorでの終了判定によってそのV分割における検索が終了した場合,detectorからPEに検索が終了したこと送信し,各PEはその通信を受信した場合にバッファを解放することにする．これらの処理により,各PEは重複した無駄な処理を行なうこと無く,また使用するバッファもより早く解放できる．

4 索引の更新

本システムにおいては索引の更新要求は以下のように与えられるものとする．

- 更新対象の索引要素が格納されているPEの位置情報が付加されている．
- 索引の更新は索引要素の削除と挿入の2種類とする．
- 更新要求は1つの索引要素の削除または挿入とする．

索引の更新が起こるのはオブジェクトが生成,削除された場合,オブジェクトの参照関係が変更された場合のいずれかである．いずれの操作も索引上では索引要素の削除,挿入に帰着し,これら操作が可能な

らばオブジェクト上の更新を索引に反映することは可能である. また, 更新対象索引要素の格納 PE の位置に関しては, 索引の分割をハッシュ関数を用いるなどの方法や, 位置情報の管理システムを設けることにより知ることが可能となり, 更新要求メッセージ内に付加することは可能である. 以上の更新要求に対して更新を行なうこととする.

本システムにおいては更新要求や検索要求が連続して HOST から送信される状況を想定している. したがってシステム全体としては以下の条件を満たす必要がある.

- 検索要求はそれ以前に出された更新要求が終了後に処理を開始.
- 検索要求はそれ以後に出された更新要求が処理前に処理を終了.

これらの事項が満たされない場合は索引としての一貫性を保つことができず, 現実のオブジェクトの参照関係を反映しないような検索結果になる可能性がある. これらの条件を満たすために, 索引システム全体をロックする方法が考えられる. つまり, 更新要求があった場合,HOST 側で要求の送信を中断し, それ以前の処理要求が全て終了するのを待ち, 終了が確認された時点で更新要求を送信する方法である. また,HOST は更新処理が終了後に次の要求を送信を行なう. 上記更新要求では更新処理の対象となる PE は 1 つである. したがって, 全体をロックする方法では更新要求により対象外の PE もロックされてしまい, 処理効率を下げることになってしまう. そこで我々のシステムでは detector の終了判定の情報を用いて更新対象 PE のみをロックする方法を用いる.

本システムのような分散並列処理を行なう場合, ある PE に到着する検索要求は,HOST が送った検索要求順に届く保証はない. したがって, 更新要求を受け取った PE はその更新をどの時点で行なえば良いか判定をしなければならない. そこで, 本システムでは以下の方法を用いる.

(1) HOST は各要求に対して,RID として自然数を用い, 小さい順に連続してつける.

(2) HOST は更新対象 PE に直接通信を行なう.

(3) detector は今まで終了していた処理の RID を保持し, 更新の条件を満たし, 更新可能になった場合に PE に更新許可のメッセージを送信する.

(4) 更新要求を受け取った PE は detector からの更新許可が到着するまで更新は行なずに保留しておき, 更新許可が到着した時点で更新を行なう.

(5) 更新要求を受け取った PE ではその RID より小さい RID をもつ要求に対しては処理を行ない, それより大きな RID をもつ要求に対しては処理をせずに保留しておく.

(1) を用いることで,2 つの要求に対し, どちらが先に出された要求であるかを判定することが出来る. また,RID をみることにより, それ以前に出された要求の RID を知ることが出来るので, detector の保持する終了した要求の RID と比較することにより, 更新処理をすることが可能かどうかを判定でき,(3) の処理が可能となる. (2) を用いることで, 更新対象 PE に更新要求がそれ以降の要求より先に到着することが保証される. したがって,(5) の処理をすることにより, 更新要求をそれより後の要求より先に処理できることが保証される. 以上にの処理によって, 前述の更新処理の条件を満足することが可能である.

5 計算機実験

以下の条件で索引分割を行ない, 出力結果の索引に対して検索時間のシミュレーションを行なった.

5.1 対象複合オブジェクト集合

以下に索引の対象となる対象となる複合オブジェクトを示す.

- 各クラスのインスタンス数は 10000 である.
- 一つのオブジェクトは 2 つの違うオブジェクトを参照する. つまり, 各属性値 $A_j(1 \leq j \leq N)$ は集合属性である.
- パスの長さは 8 である. つまり $N = 8$ である.
- 索引の格納 PE は一様乱数に基づいてランダムに決定される.

以上の条件で3つ索引を生成, 分割して対象索引とする.

5.2 並列処理環境

並列処理環境の特性値として以下のことを仮定した.

- 検索用 PE は 8×8 であり, 各 V 分割ごとに detector を用意する.
- 通信パケットは最大 100 の OID を格納可能である.
- 各 PE,HOST,detector 間の通信時間は同じものとする.
- 検索要求は HOST から送信され, 検索結果は HOST に送信される.
- PE は複数の PE に対して同時に通信は出来ない.
- PE は送信中には検索処理を行うことは出来ない.
- PE は検索処理中でも受信は可能である.
- C_s を通信初期化時間,C_t を一つの OID を送信するのに必要な時間, T_r を一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 $C_s : C_t : T_r = 100 : 1 : 1000$ とする.
- 重複検索の削除のためのバッファ内の検査時間は検索時間と同じとする.

5.3 検索シミュレーション

以下の条件の検索, 更新要求を HOST から送信し, その処理時間を計算する.

- HOST は 0 から 2000000 まで 100000 間隔で要求を送信する場合 (case1) と HOST が 0 から 10000000 まで 500000 間隔で要求を送信する場合 (case2) の 2 種類の要求系列を用いる.
- 要求が更新である確率を 0 から 1 まで 0.1 きざみで変更して行なう.
- V_N の各オブジェクトが検索要求に含まれる確率は 0.01 とする.

以上の要求の系列をそれぞれ3つずつ用意し, 各索引に対してシミュレーションを行なう. case1 と case2 では要求の間隔が異なり,case1 の方がより負荷が高い要求系列となっている. また, 比較対象として全ての索引をロックしてから更新を行なうシステムのシミュレーションも行なった.

5.4 検索シミュレーション結果

図 2,3 はそれぞれ case1 と case2 における処理時間の平均である. これにより, 本システムと全ロックのシステムとの差はほとんどないことがわかる. これは V 分割の方法により処理時間のの大半が V_1 や V_2 における処理になり, 全ロックしたとしても,V_1

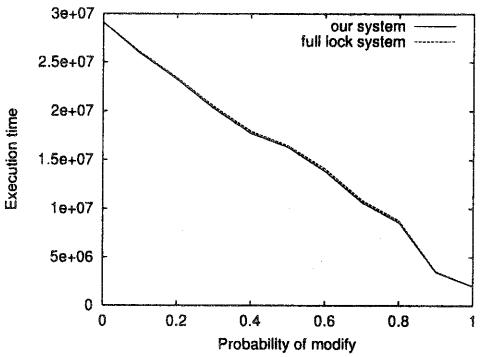

図 2: Execution times for request case1)

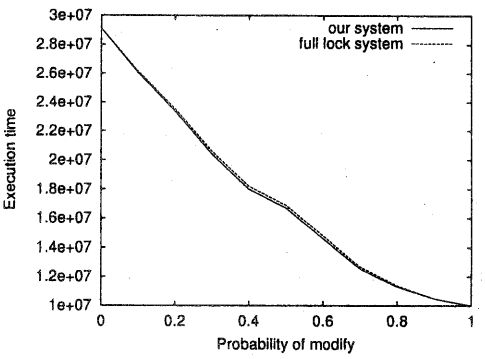

図 3: Execution times for request case12)

や V_2 はほとんど休むことなく処理する状況になるためと思われる．つまり,HOST が連続的に検索要求を送る方法であっても，1 つの要求の終了を待ってから次の要求を PE に送信する方法であっても，最終的に処理が終了する時間はほぼかわらなくなってしまい，部分ロックでも，全ロックでも処理時間がかわらなくなってしまったと思われる．

一方，図 4,5 はそれぞれ case1 と case2 におけるターンアラウンド時間の平均である． 処理が全く

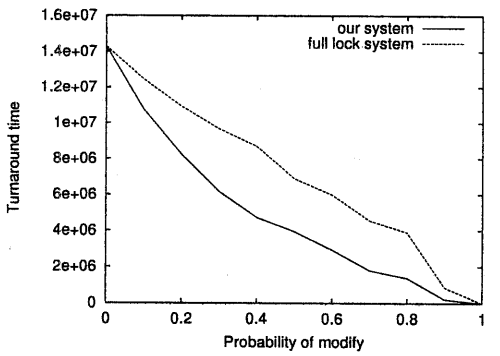

図 4: Execution times for request case1)

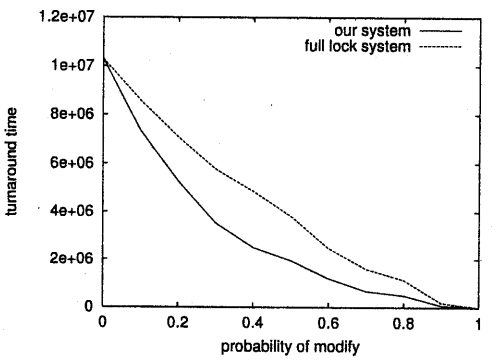

図 5: Execution times for request case12)

同じである更新の確率が 0 である場合をのぞき，全ての場合で本システムの方がターンアラウンドタイムが改善されている．これは，2 つのシステムにおいて検索要求に対するターンアラウンドタイムはほとんど差はないが，更新要求に対するターンアラウンドタイムが改善されているためである．全ロックのシステムの場合，更新要求はそれ以前の要求の終了を待つ必要がある．しかし，更新対象の PE が V_1 や V_2 などの処理が先に終わる PE であった場合，実際には更新対象 PE が更新処理の条件を満足していてもシステム全体がそれ以前の処理を終了するまで待たなければならない．一方，本システムでは更新対象の PE ごとに更新処理の条件を満たすかどうかの判定が行なわれるため，全体の終了を待たずに更新処理を実行することができる．この違いによりターンアラウンドタイムに大きな差が生じることとなったと考えられる．また,case1 と case2 を比較してみると,case1 の場合の方がより差が出ている．これは,case1 の方が要求間隔が短く，高負荷となり，更新要求発生時にそれ以前に出された未処理の要求がより多いため，その全ての終了を待つ全ロックのシステムの方がより処理待ちの時間が長くなることによるものと考えられる．

6 まとめ

複合オブジェクトに対する索引を分散並列処理するシステムにおける，更新処理の同時実行方法の提案を行なった．本方式では索引全体をロックすることなく，更新対象の索引要素を管理する PE における索引をロックすることにより，他の PE の検索処理を同時に行なうことが可能である．シミュレーション実験では，処理の終了時間にはほとんど差がみられなかった．これは，索引分割の方法により，各 PE の実際の処理負荷が均等でないために処理時間の変化がみられなかったものと考察される．したがって，処理負荷がより均一なシステムにおいては処理時間に改善がみられるものと予想される．一方ターンアラウンドタイムに関しては検索と更新の同時実行により改善がみられ，本システムが更新処理の効率化に関して有効であることが認められる．

PE の負荷がより均一な索引分割における本システムの有効性の検証と，detector の数を減らすためのシステムの改良が今後の課題である．

参考文献

[1] 小倉 一泰，都司 達夫，ブレト アルベルト，宝珍 輝尚，“複合オブジェクトの索引に対する水平垂直分割の一方式”，信学論，vol. J80-D-I, pp. 486–494, 1997.

[2] 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”, 信学技報, DE97-85, 1997.

[3] 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.

[4] 樋口 健, 都司 達夫, 宝珍 輝尚, “複 “複合オブジェクトに対する索引の分割に関する近似評価式” 第 10 回データ工学ワークショップ (DEWS'99) 論文集, pp.676–681, 1999.

[5] Bertino, E. and Foscoli,P., “Index organizasion for object-oriented database system”, *IEEE Trans. Knowledge and Data Eng.*, vol. 7, pp. 193–209, 1995.

[6] 加藤和彦, “オブジェクト指向データベースの記憶構造”, 情報処理, vol. 7, pp. 532–539, 1991.