C3-1

通信コストの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

福井大学大学院 工学研究科 中野究,滝澤淳,樋口健,都司達夫,宝珍輝尚

1. はじめに

1.1 研究の目的

複合オブジェクト集合に対する検索については多くの提案がなされているが、それらの多くは単一マシン上のものであり、複数マシンに分散して配置することにより大量索引の並列操作による効率向上を狙った研究は少ない。

本研究での分散管理システムでは、複合オブジェクト索引を分割し、それぞれ独立したプロセッサに分散管理させ、並列処理を行うことで検索処理の効率化を目指している。

しかし索引を分割して並列処理するためにはプロセッサ間での通信が必要となり、この通信コストにより処理効率が悪化することが考えられる。

また複数の検索要求を同時に処理する場合、処理の並列化により要求発生順に処理されない場合があることも考えられ、各検索要求のターンアラウンドタイムが悪化することもある。

そこで本研究では通信コストの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を提案する。

本研究では各プロセッサにおいて受信した検索要求を一時的に保留し複数の検索要求をまとめて処理することで効率化を図る。また、その際に保留中の検索要求を要求発生順にソートしてから処理することでターンアラウンドタイムを改善する。

また、インデキシングの手法としては、索引更新のオーバーヘッドが少なく、マシン間で部分的に索引を移動することが容易なマルチインデックス方式を採用した。

また、本研究が対象とする索引では複合オブジェクト内の属性値を複数想定し、さらに集合の属性をも認めたものとしている。その結果求められるであろう複数の検索経路について考慮したシステムとした。

そして、以上のシステムを並列計算機 IBM RS/6000 SP に実装する。

2. 複合オブジェクト索引の分割・並列化

本研究では、最適な分割方式と並列処理方式を採用するが、提案する方法では、複合オブジェクト属性の単一値検索の効率、及び一定時間に処理できる検索要求数としての検索スループットの向上を目的としている。また本研究では、索引更新のオーバーヘッドが少なく、マシン間で部分的に索引を移動することが容易なマルチインデックス方式を採った。

2.1 諸定義

複合オブジェクトのパスを、

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

とする。ここで、$A_j(1 \le j \le n)$ はクラス C_j の属性であり、かつクラス $C_{j-1}(2 \le j \le n)$ の属性の定義域が C_j 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 は C_1 のインスタンスであり、$o_j(j \ge 2)$ はオブジェクト o_{j-1} の属性の A_{j-1} の値である。o_{n+1} は単純値またはオブジェクト識別子(OID)であり、$o_1, o_2, \cdots, o_n$ は OID とする。P の値集合を O_p と表記し、オブジェクト o_j の属性 A_j の値が o_{j+1} である時、

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

を P の索引要素という。ここで、索引要素 $\langle o_j, o_{j+1} \rangle$ はオブジェクト o_j と o_{j+1} を保持することとする。ここで、o_{j+1} はキー値、o_j はデータ値である。さらに、パス P の索引 IP は、すべての索引要素の集合であり、

$$IP = \{ \langle o_j, o_{j+1} \rangle \mid o_1 o_2 \cdots o_n \in O_p, 1 \le j \le n \}$$

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。

2.2 マルチインデックス

マルチインデックス方式を用いた検索とは、「パス P が $C_1 A_1 A_2 \cdots A_n$ のとき、A_n の値が o_n であるようなクラス C_1 のインスタンスであるオブジェクトを求めよ。」という検索要求に対して、IP を用いて,

$$\langle o_n, o_{n+1} \rangle, \langle o_{n-1}, o_n \rangle, \cdots, \langle o_1, o_2 \rangle$$

の順に次々とオブジェクトの逆参照関係を求め(リバーストラバーサル)、最後の $\langle o_1, o_2 \rangle$ を満たす o_1 の集合を検索結果とする索引手法である。マルチインデックスにおけるオブジェクトの検索は,

オブジェクトの参照関係を逆順にたどることで実現される。

はじめに、対象となるデータ値 oid8 からそれを参照しているオブジェクト oid6,oid7 を特定し、以降、再起的に参照オブジェクトの検索を繰り返すことによって最終的な参照元オブジェクト oid1,oid2,oid3 の割り出しが可能となる(図 2.1 参照)。oid1～oid7 はオブジェクト ID、oid8 は単純値またはオブジェクト ID である。本システムでは、参照関係を検索するために B+木を採用した。B+木についての詳細は第４章で述べる。

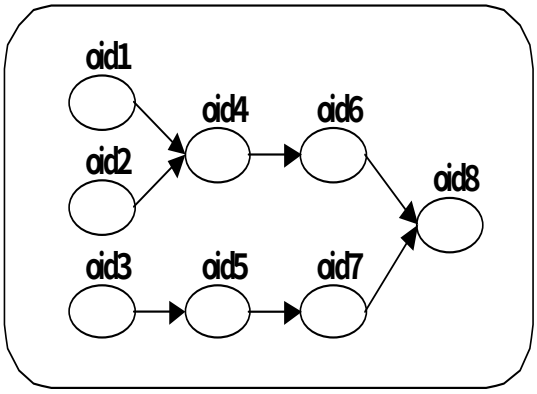

図 2. 1：オブジェクトの参照関係

3. 高並列計算機 IBM RS/6000 SP

IBM が開発した RS/6000 SP は、複数の CPU, ローカルメモリ, ローカルディスク成る PE ノードを, 高速スイッチによる相互結合網で結合した分散メモリ型の超並列計算機である。

64 ノードが高速スイッチにより結合され, 各ノードは, 332MHz の PPC604e 4CPU, 512M 共有メモリ, 4.2GB+9.1GB ディスクから構成されている。

プロセッサ間通信はシングルステージ SP スイッチ(150MB/s)により行なわれる。

なお、プログラミングは、標準化通信ライブラリ MPI を用い、C 言語で行う。

4. 複合オブジェクト索引検索システムの基本設計

4.1 オブジェクト ID の定義とインデックス登録

オブジェクト ID(OID)を次のように定義する。

(CID, IID)
CID：クラス ID
IID：インスタンス ID

また、PE の論理番号を PID と定義し、オブジェクト A がオブジェクト B を参照していることを次のように定義する。

((ACID, APID, AIID)(BCID, BPID, BIID))

これらの参照関係は、全て検索 PE 内の B+木に B のオブジェクト ID をキー値として A のオブジェクト ID が導出されるように登録される

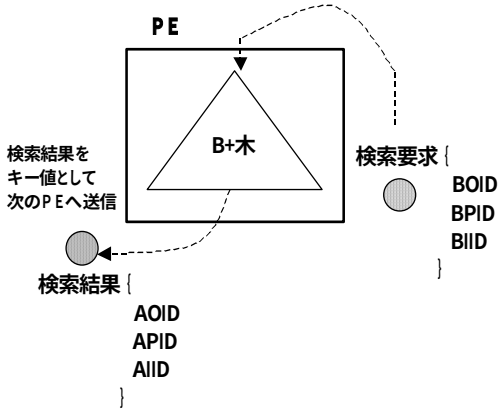

図 4. 1：PE の構成

4.2.1 索引の分割と高並列計算機への分散配置

複合オブジェクトに対する索引 IP_i を分割し、分割された索引をメッセージ通信非共有メモリ型の並列計算機の各プロセッサエレメント(PE)にそれぞれ管理させる。各 PE が同時に処理をすることにより、並列に処理を行うことができる。

ここで、検索要求を送信し、検索結果を集計するための特別な 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 から検索要求が出され、検索結果が HOST に到着するまでの時間である。

また、HOST から送られる検索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。実際に索引要素を管理し、検索要求を処理する PE を検索 PE とよぶ。

索引の分割は、前述のようにマルチインデックスの各索引ごとに行う。

4.3 検索方法

図 4.2 は分散管理システムにおけるの基本的な処理の流れを表す。

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検索要素が格納されている PE に対して検索要求を送信する。
2. 各 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(他のクラスのインスタンスの OID)ならばその OID をキー値とする索引要素を管理する PE に対し、その OID を検索条件とする検索要求を送信する。
   (ア) 各 PE に対して検索要求があるかぎり 2. と 3.を繰り返す。
   (イ) 全ての 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。

また、本システムでは HOST が検索要求を出す前に位置情報を管理している PE にその OID をキー値とする索引要素がどの検索 PE に格納されているかを問い合わせることによってどの PE に検索要求を送ればいいかわかる。

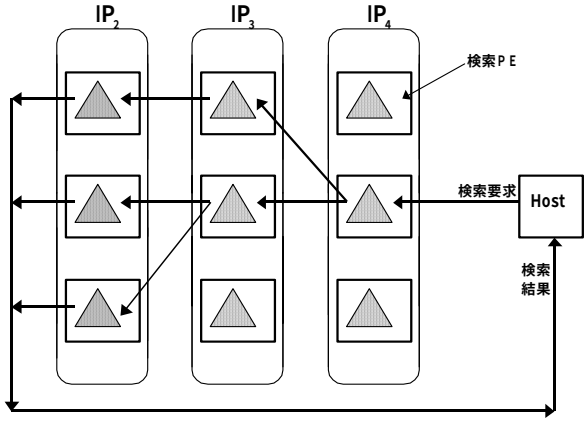

図 4. 2：索引の分割と検索

4.4 IP_i ごとの終了判定とそのアルゴリズム

本研究では各 IP_i ごとに終了の判定を行う方式を用いる。

この方式では、各段ごとに終了判定を行うことから、ハーフパスの処理などへの対応が容易であることや、検索要求内に分岐履歴のような余計な情報を盛り込む必要がないなどのメリットがある。

検索は $IP_{n+1} \rightarrow IP_n \rightarrow \cdots IP_3 \rightarrow IP_2$ の順に行われる。

ここで IP_i に関する処理の終了について考える。

IP_{i+1} の検索結果の OID 数の合計と IP_i において処理した OID 数が等しい時、IP_i の検索が全て終了したものと考えられる。この終了判定をディテクタ(判別器)と呼ぶ PE にまかせることにする。そして、ディテクタは各 IP_i に１つずつ配置する。

PE が検索したとき、検索した OID 数とその検索結果 OID 数をそのクラスの IP_i のディテクタへ送信する。各 PE から送信されてきた検索結果 OID 数の和を終了判定後に IP_{i-1} のディテクタに送信する。

IP_{i+1} の検索結果の OID 数の合計(IP_i で処理しないといけない OID 数)と IP_i において処理した OID 数を比較することで IP_i に関する処理が終了したかどうかの判定を行う。

これらの操作を HOST および各ディテクタで行うことで処理の終了を判定する。

図 4.3 はディテクタを用いた終了判定の流れを表す。

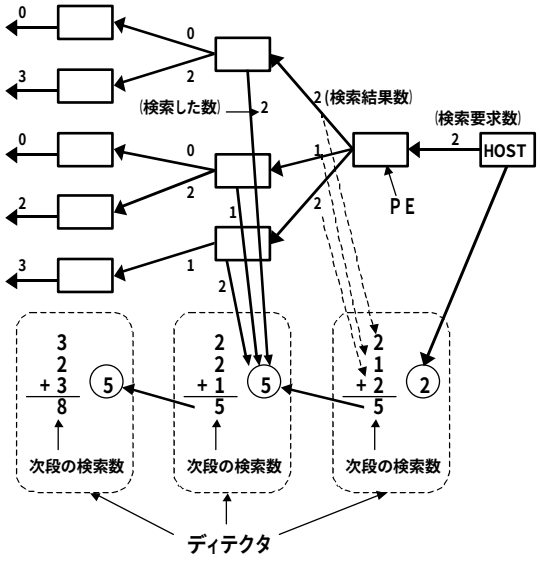

図 4. 3： IP_i ごとの終了判定アルゴリズム

5. B+木インデックス

本システムでは複合オブジェクト集合の参照関係を B+木に格納し、検索を行っている。この章では B+木の特徴と操作ライブラリについて述べる。

5.1 B+木

B+木とはB木の変形の一種であり、B木の順次処理コストが高くなるという欠点を補うために、B木を改良したものである。B+木は葉(Leaf)にデータを持ち、葉以外のノードのキーの値も含めて、葉をリンクしている。

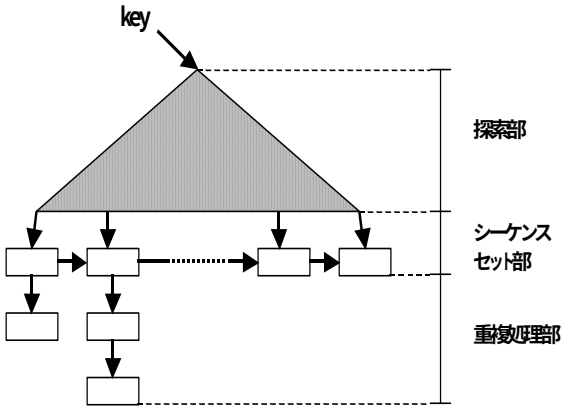

図 5. 1：B+tree の構造

B+木の特徴として、

- 一定時間以内にデータの検索が可能
- 検索時間は$O(\log n)$である(n：キー総数)
- データの逐次検索が容易(キーの範囲検索が高速)

などが挙げられる。これらの特徴は、B+tree をインデックスのデータ構造として使用することの有用性を示している。(図 5.1)に B+tree の構造を示す。ここで重複処理部とは、重複キーによるデータを格納する領域である。

6. 複合オブジェクト内の集合属性に対する処理

6.1 集合属性のある索引の検索

あるオブジェクトが、そのオブジェクトの持つ同一の属性により、異なる複数のオブジェクトを参照している場合、このようなオブジェクトを「集合属性を持つオブジェクト」と呼ぶ。

複合オブジェクト集合の検索は、参照関係の逆航路をたどることにより行われるので、集合属性を持つオブジェクトを持つような複合オブジェクト集合の検索を行う場合、集合属性を持つオブジェクト以降の検索を重複して行う可能性がある。例えば、検索要求パス$P=C_1A_0A_1A_3A_4$において、クラスC_2内に存在する集合属性を持つオブジェクト(O_1)の属性A_1のために検索が重複し、C_1－C_2間の検索を 1 つの検索要求に対して 2 回行うことになる(図 6.1 参照)。

重複検索は検索効率を下げ、検索スループットを低下させる。

このような検索の重複は、過去に検索を行ったオブジェクトを記憶しておき、同じ検索要求が到着した場合、再度同じ検索を行わないようにすることで回避できる。

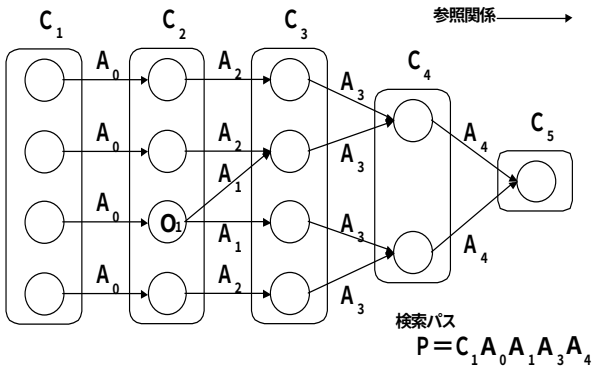

図 6.1：集合属性を持つ索引

6.2 重複検索の処理

各 PE において検索要求が到着するたびに過去に検索したオブジェクトが登録されているバッファを調べ、そのオブジェクトが登録されていなければ登録を行い検索を行う。登録済みであれば、検索が終了しているとして、再度の検索を行わないようにすることで検索の重複が防げる。また、本研究ではこの処理を行う時間を短縮させるためにハッシュを用いている。

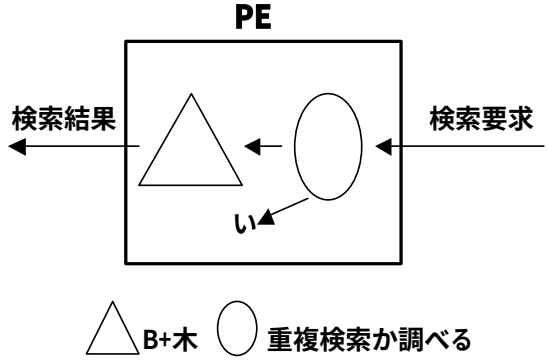

図 6.2：重複検索の処理方法

7. 索引の更新処理

オブジェクトの参照関係が変更された場合、それと対応する索引要素を更新しなければならない。

そこで、検索要求と更新要求が連続的に複数到着し、それらに対する処理を同時実行するための方式を考える。

7.1 更新要求

索引分割管理システムにおいて、検索要求、更新要求は HOST に順に与えられるものとする。

図 7.1 に検索、更新要求の例を示す。ｒは検索要求、mは更新要求である。

検索要求ｒ1～ｒ3 については、更新要求m1 が処理されない状態で実行されなければならず、検索要求ｒ4 は更新要求m1 が処理された後で、且つ更新要求m2 が処理されない状態で実行されなければならない。

なお、本システムにおいて、更新要求についてはその索引が存在する PE に対し、HOST が直接通信を行うことにする。これにより更新要求以降の検索要求よりも、更新要求を更新対象 PE に確実に先に到着させることができる。

図 7. 1：検索、更新要求の例

7.2 システム設計

分散管理システムにおいては、HOST から送信される要求の順序と、PE に届く要求の順序が同じである保証は全くない。すなわち、図 7.1 の例で言えば、ある PE では、ｒ3 がm1 よりも後に到着する可能性もある。よって、各 PE においては、到着した更新要求を処理してもよいのか確認する必要がある。

具体的な処理の流れとしては、

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 に更新要求を直接送信する。また、各ディテクタ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. ディテクタは既に終了した処理の RID を保持する。更新対象 PE に対応するディテクタにおいて、更新要求より RID の小さな要求がすべて終了したとき、更新条件が満たされたとして更新対象 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 はディテクタからの更新許可が到着するまで更新を行わずに保留しておき、更新許可が到着した時点で更新を行う。
5. 更新要求を受け取った PE では以下の条件の検索要求のみ処理し、その他の検索要求に関しては処理を保留しておく。
   (ア).保留中の更新要求の RID より小さな RID をもつ検索要求。
   (イ).保留中の更新要求の更新対象キー値が属するクラスに属さない検索要求。ただし、保留中の更新要求の処理が終了した場合は保留中の検索要求に対し、保留するべきか再度検討を行う。

8. 通信コストの軽減

並列処理をおこなう上で PE 間の通信は多くおこなわれる。そして、この通信時間は無視することができない。

これまで作成してきたシステムでは検索 PE が検索処理をおこない、検索結果を次の検索 PE に送信するという処理において、受信したメッセージごとに検索処理をおこない、検索結果を次クラスの検索 PE に送信していた。この場合、検索要求数が増えれば増えるほど処理全体の通信にかかる割合が大きくなる。

そこで検索 PE ごとにバッファを用意して、そこに受信した検索要求を一時的に保留しておく。

複数の検索要求をまとめて処理し、検索結果を宛先 PE ごとにまとめて送信することで通信回数を減らし、検索処理全体の処理時間の向上を計る。

8.1 アルゴリズム

検索 PE は定められた検索要求数を受信するまで、受信した検索要求をバッファに保留しておき、定められた検索要求数に達した場合、保留しておいた検索要求をまとめて処理する。

定められた検索要求数を受信していない場合でも、受信するデータがない場合、検索 PE の稼働率を上げるために検索処理を行う。これにより検索で検索要求の到着を待ち、なにもしない状態になるのを防ぐ。

具体的な処理の流れとしては、

1. 規定数の OID を受信するまで保留しておく。
2. 受信するデータがない場合か規定数受信し

た場合は検索をおこなう。

3. 保留しておいた検索要求を処理する際に一番小さい RID のキーについてのみ検索する。
4. 検索結果を宛先 PE ごとにまとめて送信する。この時、2 つのタイプを用意した。
   (ア).検索処理と送信処理を 1 セットと考え、検索結果が出たあとにすぐに送信する。
   (イ).検索結果を送信する前に受信できるものがあれば受信し、検索処理をおこなう。

(3)の処理では RID ごとに要求がソートされるため、検索の追い越しを防ぐことができ、古い検索要求を優先的に処理することで検索要求のターンアラウンドタイムを速めることができる。

(4)の処理では、一般的には(ア)の処理タイプで通信コストの軽減が期待できるが、検索処理をおこなっている間に次の要求が来ていることも考えられるので、少しでも早く受信し、まとめて送信すればより多くの通信コストを軽減できるのではないかという考えのもとに(イ)の処理タイプも用意した。

9. 評価・考察

9.1 通信コストを軽減させた検索

8 章で述べたアルゴリズムで検索を行い、その効果を検証する。

尚、以下の 2 つのアルゴリズムについて検証する。8.1 の 4.(ア)をタイプ 1、4.(イ)をタイプ 2 とする

タイプ１：検索要求を受信した際に要求を保留しておき、規定数貯まったら処理する。また、次の要求が来るまでの空いた時間でも検索処理を行う。

タイプ２：タイプ 1 の処理にくわえて検索結果を次の検索 PE に送信する前に受信できる検索要求が来ていないか調べ、来ていたら受信して処理し、来ていなければ送信処理を行う。

9.1.1 評価環境

- オブジェクトはｎ=8 で各クラスのインスタンス数は 10000、1 個のオブジェクトがランダムで１個または 2 個のオブジェクトを参照、または参照なし。
- PE を 36 個使い、その内訳として HOST1 個、検索 PE24 個、ディテクタ 9 個、検索要求を出す PE1 個、結果集計用 PE1 個。
- 索引は一様乱数にもとづいて分割。
- 分割された索引はランダムで検索 PE に分散配置する。
- 各クラスで検索 PE を固定しない。(1 つの検索 PE が複数のクラスの索引を持つ)
- 検索要求 10 個を連続に出す。
- 各検索要求には 100～1000 のキー(OID)が含まれる。(各計測で固定値、100,300,500,700,1000 を測定)

検索 PE で検索要求を受信した時にバッファに貯めておく OID の最大格納数を変化させて調べる。

9.1.2 測定結果

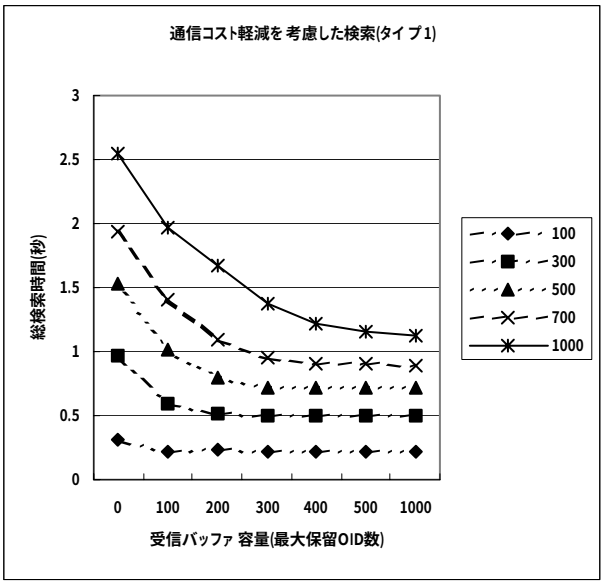

図 9. 1：通信コストを軽減させた検索(タイプ 1)

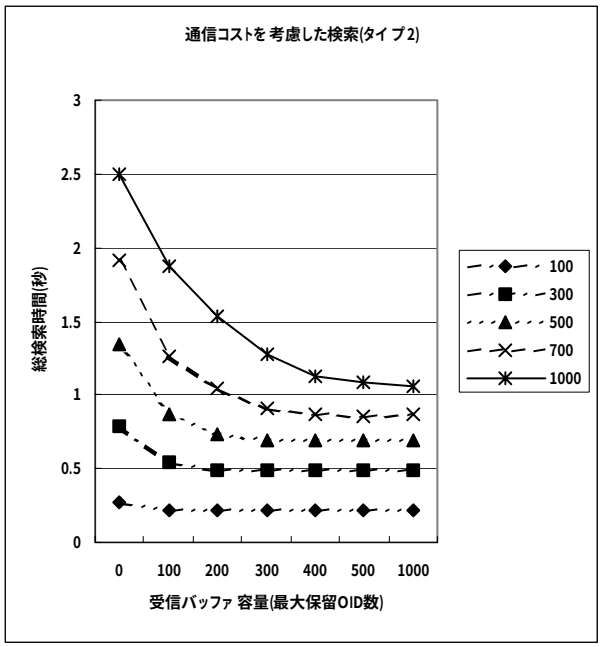

図 9. 2：通信コストを軽減させた検索(タイプ 2)

従来型、タイプ 1、タイプ 2 を比較した。尚、タイプ 1,タイプ 2 については受信バッファ容量が 1000 の時の数値で、従来型とは 1 つの要求ごとに受信したら処理して結果を送信するというタイプである。

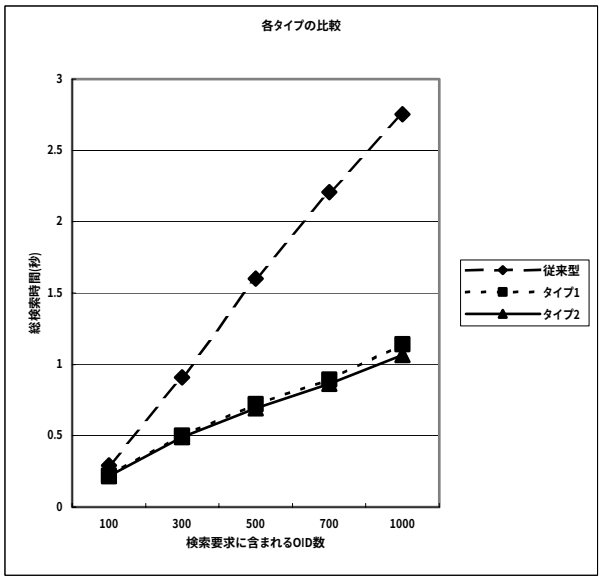

図 9. 3：各タイプの比較

9.1.3 考察

タイプ 1 の受信バッファ容量が 0 の時の検索は通信コストを考慮しない従来型の検索と同じである。

受信バッファに要求を保留しておくことにより、検索時間が速くなっているということがはっきりとわかる。検索処理をまとめておこなうことにより、検索結果を送信する際とディテクタに検索数、検索結果数を知らせる際の通信回数が減少し、検索時間が速くなったと考えられる。

また、検索要求に含まれる OID 数が多いほどその効果は大きく、要求を保留しておくことの有効性がうかがえる。これは、検索要求に含まれる OID 数が少ないと次クラスに送信される検索結果 OID 数がすぐに少なくなり、その結果、受信した OID が規定数保留される前に検索処理をおこなってしまうことが多くなる。よって、従来型との差が縮まり、検索時間が従来型と変わらなくなりやすいと考えられる。

また、受信バッファ容量を増やしていくとそれ以上増やしても検索時間は変わらないという状態になる。そして、検索要求に含まれる OID 数が少ないほど受信バッファ容量が小さい時にこの状態になる傾向であるとグラフから読みとれる。これは、受信バッファ容量が大きくなると通信コストは減少するが、保留している OID が受信バッファ容量いっぱいになる前に受信する要求が来なくなり、検索処理をおこなうことになるため、受信バッファ容量を増やしても全体の処理の流れは変化することがなく、結果、検索時間が同じになると考えられる。

タイプ 1 と比べてタイプ 2 は検索時間が速くなっている。これは、検索結果を送信する前に受信できる要求があれば受信し、処理することでさらに通信コストを抑えることができた結果だと思われる。

二次記憶版 B+TREE を使用しているため、検索処理の際にディスクアクセスを要し、検索処理自体でのコストも無視できない。そのため、検索処理をおこなっている最中に要求が多数届いている可能性が高い。このことから処理全体の流れをスムーズにするために検索結果を送信する前に受信処理ができるようにしたことは有効であったと思われる。

9.1.4 更新要求を含んだ検索

9.1.1 の評価環境では更新要求は含んでいなかった。そこで、更新要求を 5 つ含んだ検索に関してもデータをとった。尚、その他の条件は 9.1.1 と同じである。

更新要求なしの時との比較グラフを図 9.4 に示す。

9.1.5 測定結果

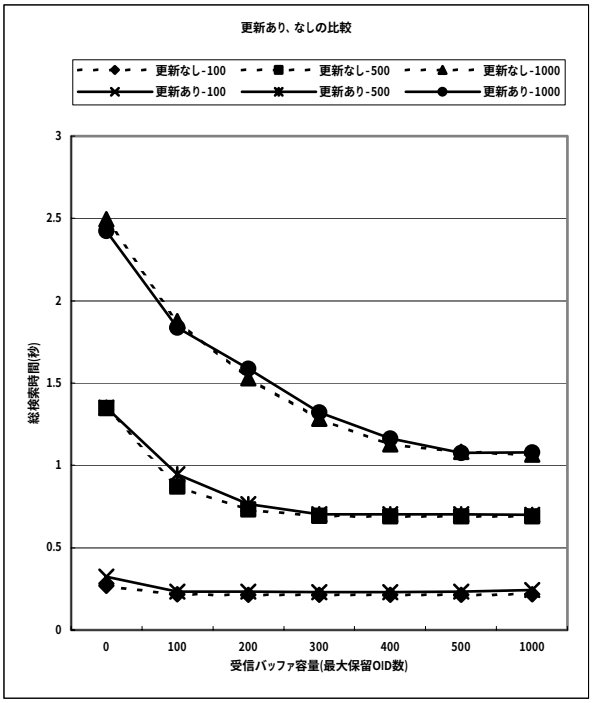

図 9. 4：更新要求を含んだ検索との比較

9.1.6 考察

検索時間に大して変化はなく、更新要求が入ったことによる影響が少ないことがわかる。本研究では更新要求の処理において PE 単位且つ RID 単位でのロックをおこなうことにより他の検索要求に対しての影響を最小限に抑えている。くわえて、受信した検索要求を RID ごとにソートして処理することにより検索の追い越しを極力抑えるシステムになっている。そのため、更新要求が検索 PE に到着した時にはそれ以前に出された検索要求の処理が終わっている可能性が高く、すぐに更新処理をおこなえるため、ロックしておく時間が短くなり、更新要求を入れたことによる影響が少ないと考えられる。

10. 結論

本研究では、マルチインデックス方式によって索引付けした複合オブジェクト集合を分割し、分散配置した分散管理システムにおいて通信コストを軽減させることにより検索時間を速めることを目指した。

その結果、通信コストを軽減させ、検索時間を速めることに成功した。

また、更新要求を含む検索についても効率の良い処理システムを提案し、その成果を示すことができた。

参考文献

[1] 樋口 健、山口 誠司、都司 達夫、宝珍 輝尚
「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
情報処理学会論文誌、2000.

[2] 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.

[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」
信学技報、DE97-85、1997.

[4] 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

[5] 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

[6] 山口 誠司
「複合オブジェクト索引の並列検索システムにおける索引の更新」
平成 11 年度電気関係学会北陸支部連合大会

[7] 丸山 幹夫
「高並列計算機システムのおける複合オブジェクト索引の並列計算機システム」
平成 10 年度福井大学工学部情報工学科卒業論文