分散問題解決による共有資源の効率的な実現

3T－2

濱地 弘樹*　楢崎 修二**　吉田 紀彦*　牛島 和夫*
＊　九州大学大学院 システム情報科学研究科
＊＊　九州工業大学 工学部

1. はじめに

分散環境下で共有する資源の参照・更新時にはプロセッサ間で通信が行なわれる。分散処理の効率を考えるうえで、通信コストは無視できない問題である。

従来の資源共有方式である中央サーバ方式[1]では資源の参照時に必ず通信が行なわれる。完全複製方式[1]では通信を行なわずに局所的に複製を参照できるが、更新時にプロセッサの数に比例した通信が行なわれる。分散共有メモリにおいて一般に使用される読み取り複製方式[1]では、前回の更新の後に１回でも参照したプロセッサの数に比例した通信が更新時に行なわれる。

複製を作ることによって更新時の通信回数は増加するが、それ以上に参照時の通信を削減できれば効率的になる。そのために各プロセッサにおける資源の参照頻度を基に最適な複製の配置を決定する必要がある。また各プロセッサにおける参照頻度の動的な変化に対応する必要もある。そのために本研究では分散問題解決の枠組を利用して共有資源の実現を行なう。

分散問題解決において問題プログラムと協調戦略とを分離して記述する方法が提案されている[2]。その研究では、メタオブジェクトプロトコルを用いて、エージェントの持つ局所情報の変化に付随するメタレベル計算として協調処理を実現している。本研究では、共有資源の複製の配置決定戦略を共有資源の参照・更新に付随するメタレベル計算として実現する。

2. 分散問題解決による資源共有

本研究では、規模や地理的な範囲に因われないように放送機能を持たない分散システムを前提としている。また資源の参照頻度は急激に変化しないものと仮定している。これらの前提の下で、共有資源の参照・更新時に行なわれる通信を削減するために、複製の配置を動的に変更する方法を提案する。

本研究で提案する共有資源の実現方法を図１に示す。各プロセッサにそれぞれ１つのエージェントを置いて、それらのエージェントの集合として共有資源を表現する。ここに出てくるエージェントを共有資源エージェントと呼ぶ。

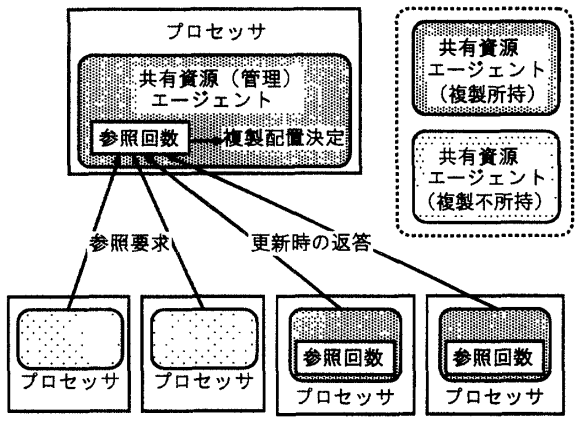

図 1: 分散問題解決による共有資源の実現

各プロセッサは共有資源エージェントを共有資源とみなして参照・更新の操作を行なう。その時に複製配置決定戦略を起動する。この部分はメタオブジェクトプロトコルを使用して実装している。詳細は３章で述べる。全体の通信回数を小さくするために、各共有資源エージェントがプロセッサから参照された回数の情報を基に複製配置を決定する。ここで特別な共有資源エージェントを設ける。これを管理エージェントと呼ぶ。管理エージェントは中央サーバのようなもので、常に複製を所持し、参照要求に対する返答や複製の配置決定を行なう。分散問題解決においては全体を管理する管理エージェントなるものは本来存在しないのだが、研究の第一歩として管理エージェントを設けることにする。今後の課題として管理エージェントの処理の分散化を考えている。

Efficient resource sharing using a framework for distributed problem solving.
Hiroki Hamachi*,Shuji Narazaki**,Norihiko Yoshida* and Kazuo Ushijima*.
*Graduate School of Information Science and Electrical Engineering,Kyushu University.
**Department of Computer Engineering, Faculty of Engineering, Kyushu Institute of Technology.

参照時に複製不所持エージェントは管理エージェントに参照要求を送信する。よって、管理エージェントはそのエージェントがプロセッサから参照された回数を知ることができる。読み取り複製方式と違って、参照時に複製は作らない。一方複製所持エージェントは局所的に複製を参照した回数を自ら数えておく。

更新時には管理エージェントに更新要求を送信して、各エージェントが参照された回数を基に管理エージェントが複製の配置を決定する。この時、管理エージェントは複製の新規作成・書き換え・破棄といった命令を必要に応じて各エージェントに送信する。複製所持エージェントは局所的に複製を参照した回数をその返答に添付する。

これまでに述べたように、各エージェントがプロセッサから参照された回数を更新完了時に管理エージェントが知ることができる。次回の更新時にはここで得た参照回数を基に複製の配置を決定する。

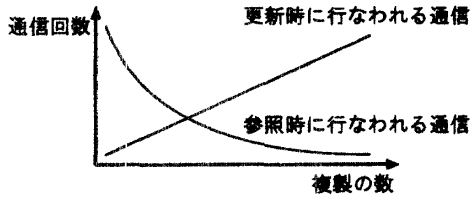

図 2: 複製の数と通信回数

参照された回数の多いエージェントから複製を所持させるわけだが、複製を所持するエージェント数は次のようにして決定する。前回の更新完了時から今回の更新完了までに行なわれる通信回数は、複製の数を変化させるに従って図2のように変化する。図2の2つのグラフを足し合わせたグラフが極小となる点が全体の通信回数を最小とする複製の数となる。

3. 実装方針

メタオブジェクトプロトコルを持つ Tiny CLOS[3] で提案した資源共有方法の実装を行なった。共有資源エージェントは分散処理を行なうユーザプログラムにおいて使用される。共有資源エージェントがユーザプログラムから参照・更新される時に、2章で述べた処理をメタレベル計算として起動する。そうすることによりユーザプログラムを書き換える必要なしに、共有資源の複製の配置変更をメタレベルで行なえる。

Tiny CLOS ではオブジェクトのスロットの参照・代入に `slot-ref`・`slot-set!` という関数を使用する。これらの関数はオブジェクトのクラスのスロット毎に設定された参照子・代入子を使用して実際に参照・代入を行なう。従って新たに関数を作成して特定のスロットへの参照子・代入子に設定してやれば、参照・代入時に通常と異なる処理を行なえる。

共有資源エージェントをオブジェクトとして実装し、それの持つ特定のスロット（body とする）に対する参照・代入を2章で述べた処理を行なうように再定義する。そしてユーザプログラムにおいて共有資源エージェントをスロットに持つオブジェクトを作成して、そのスロットに対する参照・代入を body に対する参照・代入と再定義すれば、2章で述べた処理を資源の参照・更新時にメタレベルで行なえる。

4. 実験・評価

従来の読み取り複製方式と本研究の提案方式とを実験により比較する。実験内容は、4つのクライアントが更新間で頻度を変化させて参照を行なうテストプログラムを作成して通信回数を計測するものである。その結果、読み取り複製方式では 1150 回の通信が行なわれたのに対し、提案方式で行なわれた通信は 884 回であった。両方式には参照頻度の低い時の複製の数に違いが見られた。読み取り複製方式では1回でも参照すると複製を作るので提案方式と比べて複製の数が多かった。この時の更新時に行なわれた通信回数の差が全体の通信回数に影響しているといえる。

5. おわりに

本研究では、分散問題解決の枠組を利用して共有資源の複製の配置を動的に決定することにより効率のよい資源共有を行なった。また実験により読み取り複製方式と比較して通信回数の削減を確認できた。今後は管理エージェントの処理を分散化する方向で研究を進めていく。

参考文献

[1] Michael Stumm and Songniam Zhou. Algorithms Implementing Distributed Shared Memory. *IEEE Computer*, Vol. 23, No. 5, pp. 54–64, May 1990.

[2] 山崎賢治, 楢崎修二, 牛島和夫. メタレベル計算を用いた協調処理の実現. 情報処理学会研究報告 PRG, Vol. 95, No. 82, pp. 145–152, August 1995.

[3] Gregor Kiczales. Tiny CLOS. Xerox, ftp://arisia.xerox.com/pub/openimplementations/, 1991.