PVMによる分散共有メモリアルゴリズムの評価

中嶋 卓雄*，西山晃司*，中村 良三*

*熊本大学 工学部

複数の異種計算機をネットワーク上で統合する分散処理技術が活発に研究されている．従来，分散アルゴリズムの研究は，特定のハードウェアで実現したり，基本的なアルゴリズムに限ったシミュレーションが多く，ネットワーク資源全体に関わるアルゴリズムやその評価はなされていない．本稿では，PVM(Parallel Virtual Machine) により，ネットワーク上でメモリを共有させ分散管理する分散共有メモリアルゴリズムを実現した．今回は，代表的な分散メモリ共有アルゴリズムである Read Replication アルゴリズムに注目し，マネージャサーバにおける状態を詳細化し，メモリブロックのロックによる制御を待ち行列によりモデル化した。アルゴリズムのモデル化からコストの評価式を導出し，さらに，待ち行列を考慮したシステムの応答時間を考察した。さらに，PVM により，シミュレートすることにより，動的な負荷および応答時間を評価した．

1　はじめに

近年，コンピュータの小型化・高性能化にともない，複数の異種計算機をネットワーク上で有機的に統合し，計算機資源を有効に活用し，より複雑な問題を解決する分散処理技術が活発に研究されている．特に，ハードウェアのダウンサイジング・高性能化によりマルチプロセッサ構成の計算機システムを製作することが可能となり，ハードウェア上で分散処理のシミュレーションが行なわれてきた．

分散アルゴリズムをソフトウェアから評価する研究は，基本的なアルゴリズム [1] に限定したシミュレーション [2] が多く，ネットワーク資源全体に関わるアルゴリズムやその評価はなされていない．また，ソフトウェアにより分散処理を実現する手法として分散 OS が研究されているが，個々の分散アルゴリズムを評価するのは困難である。各種の分散アルゴリズムの中で分散共有メモリアルゴリズムはユーザにとって自然にプログラミングできる環境を与えることができ，利用価値が高い．[5][6]

一方，容易にシミュレーションを実現する手法の中で，ネットワークで接続された異機種の逐次・並列コンピュータを単一の大きな並列計算機資源に統合するパッケージソフトウェアであるPVM(Parallel Virtual Machine) が並列アルゴリズムのシミュレータとして利用されている．[3][4]

本稿では，ネットワークで接続されたコンピュータのメモリを仮想的に共有させ分散管理する分散共有メモリモデルを PVM 上で実現し，各モデルのコストを評価するとともに，シミュレーションによりモデルの評価を行う．

PVM で分散共有メモリモデルを具現化することにより，異機種の計算機を統合した計算機環境で分散共有モデルを実現することができ，さらにアルゴリズムと PVM 内部の処理を分離することが可能となる．また，アプリケーションへ有効なインターフェースを提供することも可能である．

今回は代表的な分散共有メモリアルゴリズムで

Evaluation of Distributed Shared Memory Algorithm on PVM

Takuo Nakashima*,Koji Nishiyama*, Ryozo Nakamura*

*Faculty of Engineering, Kumamoto University

ある，Read Replication アルゴリズムについて，マネージャサーバにおける状態を詳細化し，メモリブロックのロックによる制御を待ち行列によりモデル化する。アルゴリズムのモデル化からコストの評価式を導出し，さらに，待ち行列を考慮したシステムの応答時間を考察する。さらに，PVM により，シミュレートすることにより，動的な負荷および応答時間を評価した．

2 基本モデル

2.1 分散共有メモリモデル

基本的なモデルとして，文献 [5] に紹介された分散共有メモリモデルを用いる．そのモデルは，表1に示すように，移動 (migration) および複製 (replication) の有無によって4つに分類される．

表1．モデルの分類

	データのコピーをしない	データのコピーをする
ブロックの移動をしない	Central-server モデル	Full-replication モデル
ブロックの移動をする	Migration モデル	Read-replication モデル

ここでは，Read-Replication アルゴリズムにおけるデータブロックの読み込み操作と書き込み操作の概要を示す．

Read-replication algorithm

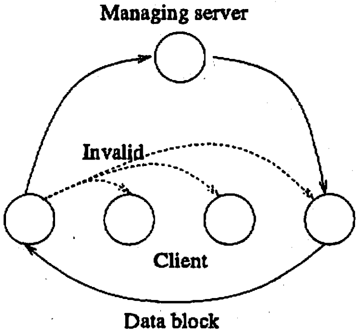

図1 Read-Replication アルゴリズム

このアルゴリズムでは，データブロックに所有者を設け，その複製を複数のクライアントが保持することを可能とする．所有者情報を管理する Managing server により，書き込み等の制御を行なう．

- Read 要求の場合，ローカルに存在するデータブロックはそのまま読み込み，ローカルにない場合には，Managing server を経由して，ブロックを所有しているクライアントからデータをもらう．Managing server は要求を出したホストを新しくそのデータの所有者とする．
- Write 要求の場合，書き込むホストが一つになるように Managing server が相互排除を行う．ローカルかリモートに関わらず，または所有者であるかないかに関わらず，Managing server に Write 操作を依頼する．要求を出したホストは，Managing server の要求により所有者からデータを受けとり，データブロックをコピーしている他のクライアントへ Invalid 命令を出してデータを無効化する．さらに，Managing server は要求を出したクライアントを所有者に変更する．

2.2 PVM

OakRidge 国立研究所において 1989 年から開発された PVM[7] はネットワークに接続された異機種 UNIX コンピュータ群を単一の並列コンピュータとして利用することを可能にするソフトウェアシステムであり，大規模な問題に対しても解決能力を持っている．

PVM では，複数のコンピュータの集合をユーザが定義し，一つの大きな分散メモリ型のバーチャルマシンとして表す．PVM における計算の単位はタスクであり，タスク間で通信および同期を実現する．

PVM は次のような特徴を持つ．

1. タスクを識別するためにホスト名とプロセス ID からなるタスク識別子 (tid) を用いる．
2. バーチャルマシンへの追加・登録が容易であり，UNIX 上のプロセスへのタスクの組み込みも簡単である．
3. 通信

   通信は，非同期ブロック送信として，タスク識別子 (tid) とメッセージタグ (tag) をパラメータとした pvm_send(tid,tag) を用いる，非同期受信には，pvm_nrecv(tid, tag) を用る．

2.3 待ち行列モデル

分散共有メモリアルゴリズムを実行するメモリ管理処理を1つのシステムと考える.

Read-Replication アルゴリズムでは, 同期制御のためのゲートとして待ち行列を利用する. また, 待ち行列の解析は, 動作解析法に基づく手法を用いる. このモデルでは, 以下の事項を要求条件とする.

- システムの流れは均衡している. すなわち, 観察中の待ち行列への到着数は出力数と同じである.
- すべての待ち行列は同質である. すなわち, 待ち行列の平均サービス時間は他の待ち行列の長さとは無関係である.

Read-Replication モデルについて, システムの構成を次の図のようにモデル化する.

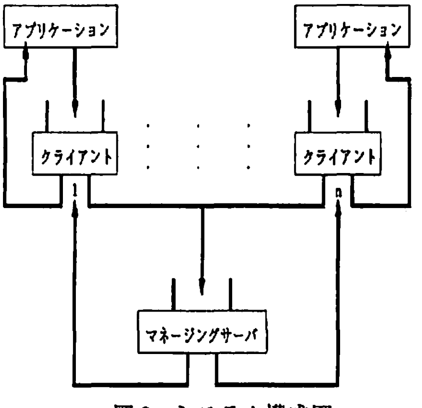

図2 システム構成図

ネットワークで接続されたコンピュータそれぞれに対して, アプリケーションが存在しメモリ管理システムのクライアントへ読み込み, 書き込み要求を出す. クライアントでは, 要求されたブロック単位に待ち行列を構成し, アクセスに対する同期制御を行う. クライアントから, Managing Server に対して, 共有メモリに対するリモート読み込み, および書き込み要求をブロック単位に待ち行列を構成し, アクセスに対する同期制御を行う. Managing server で処理された結果はクライアントに返され, クライアントでは, さらにアプリケーションに戻す.

3 PVM による Read-replication algorithm のシミュレーション

アルゴリズムを具現化するにあたって現実的な問題である相互排除, 同期などを考慮する. また, コピーブロックを有効としたまま保持する方法と無効にする方法についてシミュレートする. 以下, アルゴリズムの説明では, メッセージタグを明記することによって, pvm の送受信メッセージを表す.

3.1 Managing server

Managing server は各ブロックの配置情報とその所有者を常に把握する. また, 各データブロックに対するアクセスを監視し, ブロック単位にブロックのロック解放待ち行列を持つ.

次に, Managing server における状態遷移図を示す.

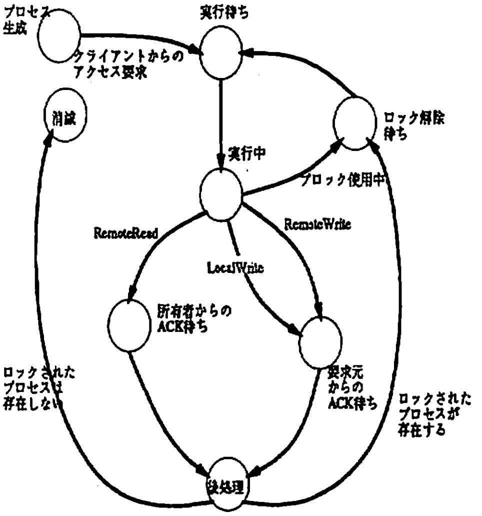

図3 Managing server における状態遷移

3.1.1 Read 要求

クライアントから Read_copy_REQ(読み込みコピー要求) を受信した場合

1. 要求されたブロックがロックされていなければロックし,
2. データブロック所有者にブロックの Copy_REQ(Copy 要求) を送る.

3. 所有者が要求を受け取ると，ACK を返すので，その ACK を Managing server が受け取ったときロックを解除し，待ちプロセスがあれば起動し，なければアイドル状態となる．

3.1.2 Write 要求

クライアントから Write_REQ(Write 要求) を受信した場合，

1. 要求されたデータブロックをロックする．
2. 所有者にブロックに対する Copy_REQ(コピー要求) を送る．
3. 要求元クライアントから ACK を待つ．
4. ACK を受け取ると，そのデータブロックのロックを解除し，所有者の配置情報を更新し，要求元を所有者とする．

クライアントから Local_Write_REQ(ローカル Write 要求) を受信した場合，

1. 要求されたデータブロックをロックする．
2. 要求元クライアントに ACK を送る．
3. 要求元クライアントから ACK を待つ．
4. ACK を受け取ると，ロックを解除する．所有者の配置情報を更新し，要求元を所有者とする．

3.2 クライアント

要求されたブロックがロックされているかどうかをチェックしロックされていれば待ち行列に入れる．

クライアントにおける状態遷移図を示す．

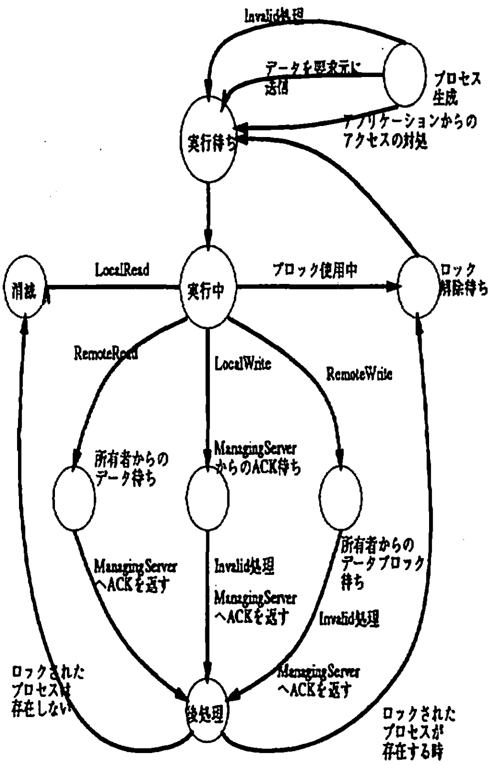

図4 クライアントにおける状態遷移

3.2.1 Read 要求

ローカル Read 要求

そのまま読み込む．アプリケーションを結果を返す．

リモート Read 要求

- Managing server へ読み込みたいブロックに対する Read_copy_REQ(読み込みコピー要求) を送る．
- 所有者から Copy_Data(コピーデータ) を待つ．
- ブロックを受信したら保持する．
- Managing server へ ACK を返す．
- アプリケーションを結果を返す．
- ロックを解除し，待ちプロセスがあれば起動し，なければアイドル状態となる．

3.2.2 Write 要求

ローカル Write 要求

1. Managing server に Local_Write_REQ(ローカル Write 要求) を出す.
2. Managing server からの ACK を待ち，その後ブロックを更新する.
3. 各クライアントに Invalid(データ無効) またはデータ更新を送信する.
4. Managing server に ACK を返す.
5. アプリケーションに結果を返す.
6. ロックを解除し，待ちプロセスがあれば起動し，なければアイドル状態となる.

リモート Write 要求

1. Managing server に Write_REQ (Write 要求) を出す.
2. 所有者からブロックの受信を待ち，その後ブロックを更新する.
3. 各クライアントに Invalid(データ無効) またはデータ更新を送信する.
4. Managing server に ACK を返す.
5. アプリケーションに結果を返す.
6. ロックを解除し，待ちプロセスがあれば起動し，なければアイドル状態となる.

Managing server の Copy_REQ(コピー要求)

そのブロックと共に Copy_Data(コピーデータ) を要求元に送る.

他のクライアントからの Invalid 要求

対応するブロックを破棄する.

他のクライアントからのデータ更新要求

対応するデータブロックを更新する.

4 評価

4.1 コストの評価

Read-Replication アルゴリズムの評価式を導出する. 対象は確認のための ACK パケットを除くすべてのパケット処理について考慮する.

リモート Read/Write 要求ともデータブロックの移動に $(2P+4p)$ 要する. また，リモート Write 要求の際にはコピーを保持するクライアントに Invalid またはデータ更新メッセージをマルチキャストするので Sp が加わる. また，ローカル Write 要求について，要求に $4p$, Invalid またはデータ更新メッセージに Sp 要するので，コスト (C_{rr}) は次式のようになる.

$$C_{rr} = f'\left[(2P+4p) + S\frac{p}{r}\right] + (1-f')(4+S)\frac{p}{r}$$

ただし，

p	:	1 回のパケットイベントにかかるコスト. すなわち，1 回の送信または，1 回の受信にかかるコスト. (ただし，小さいサイズのパケットの場合).
P	:	データブロックの送信，または受信を 1 回するのにかかるコスト.
S	:	分散共有メモリシステムのクライアント数.
r	:	Read 要求と Write 要求の割合. r 回のアクセス (Read,Write) に対して 1 回の Write 要求が発生する.
f'	:	Read replication algorithm において，データがローカルに存在せずアクセスフォルトとなる確率.

4.2 応答時間の評価

Read-Replication アルゴリズムの待ち行列における応答時間の評価式を導出する.
システムは以下の構成とする.

- 1 個の Managing server と n 個のクライアントからなるシステムを考える.
- Managing server では，クライアントからの読み込み・書き込み要求をブロック単位に待

ち行列を構成する．

- クライアントでは，アプリケーションからの読み込み・書き込み要求をブロック単位に待ち行列を構成する．

時間間隔Tの間にサーバ・クライアントにおいて待ち行列が発生する．このとき，次のデータを測定する．

- A_i = ホスト i のアプリケーションからクライアントへ到着する要求数
- B_i = ホスト i のクライアントからサーバへ到着する要求数
- C =サーバの待ち行列から出ていく要求数
- D_i = ホスト i のクライアントの待ち行列から出ていく要求数
- E_i = ホスト i のクライアントの待ち行列のサイズ m_i が $m_i > 0$ である時間の長さ
- W = サーバの待ち行列での要求数と時間の乗算数の積算値

これらの値から，応答時間を含むシステムの性能尺度は次のように表される．

- $R_i = \frac{E_i}{T}$ ホスト i のクライアントのプロセッサ使用率
- $S = \frac{W}{T}$ サーバの待ち行列での平均待ち行列長
- $U = \frac{W}{C}$ サーバの待ち行列での平均応答時間

5 シミュレーション

5.1 シミュレーション実行環境

Read-Replication アルゴリズムを PVM で実現し，シミュレーションを行った．シミュレーションには，ネットワークでつながれた5台の Sun ワークステーションを用いた．メモリブロックに対する Read および Write の総要求回数を一定にし，Write 要求発生確率を 10%〜90%まで変化させた時の，ネットワークで交換されるメッセージ数およびコストを計測した．

5.2 シミュレーション結果

シミュレーションは評価式に現れない輻輳の問題，およびデータコピーの手法の相違によるコストの評価に注目した．

5.2.1 データコピーがコストに及ぼす影響

Read-replication アルゴリズムにおいて，データを更新するモデルのシミュレーション結果を図5に，破棄するモデルの結果を図6に示す．どちらも，Write 要求発生確率に対するコストおよびコストに含まれるリモート/ローカルおよび Read/Write の割合を示す．ここでは，$P = 10, p = 1$ とし，f' については，各クライアントにおいて共有メモリ全体に均等にアクセスが発生すると仮定した．

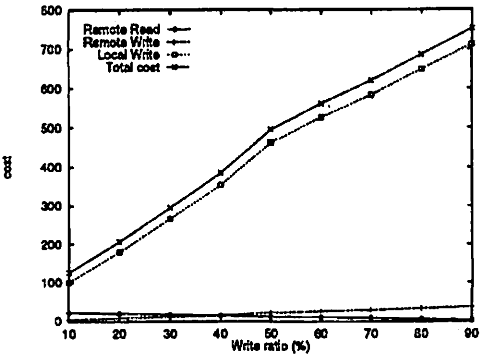

図5 更新するモデルのコスト

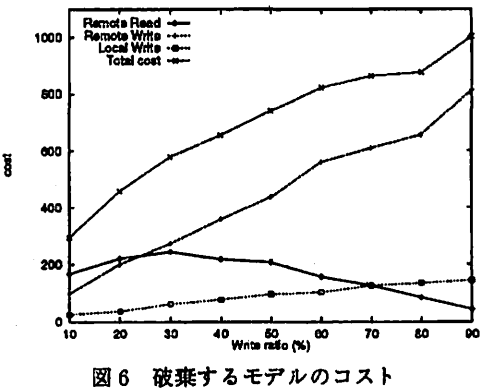

図6 破棄するモデルのコスト

更新するモデルでは，データブロックのコピーをローカルに保持する結果となり，コストの大部分がローカル Write で占められる．一方，破棄するモデルではローカルに保持するブロックが減るため，リモート Write がコストの大部分を占める結果となる．

また，図7に2つのモデルにおいて，要求数の増加に伴う，アクセス要求から応答までの時間の総和の変化をを示す．

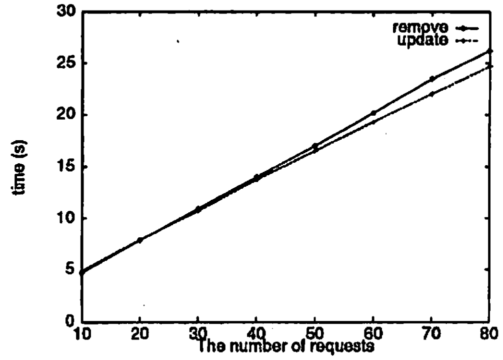

図7　2つのモデルの応答時間

図7に示すように，破棄するモデルの方が更新するモデルよりも応答に時間がかかる．これは，更新するモデルではローカルにコピーブロックが存在するため，ブロックの転送による処理時間が必要となり応答時間が短くなる．

5.2.2　応答時間の評価

ここでは，平均応答時間を次のように定義する．

[平均応答時間]

$= \gamma$[行列待ちの平均時間] + [平均サービス時間]

γ :待ち行列に登録されロックされる確率

Read-Replication アルゴリズムにおける Managing Server の待ち行列に注目したときの応答時間のシミュレーション結果を検討する．なお，以下のシミュレーションでは，ブロック分割を固定させた上で，クライアントの数を変化させている．

図8に，サーバにおいて待ち行列に登録されロックされる確率を示す．

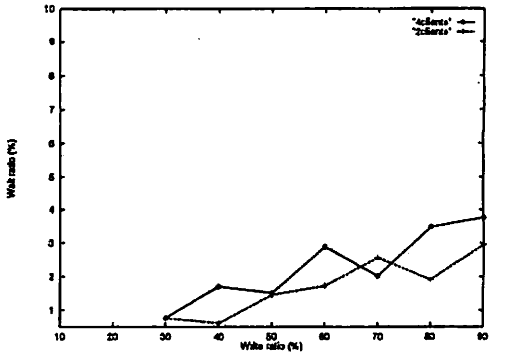

図8　待ち行列にロックされる確率

図8から明らかなように，クライアント数が増加するにしたがって，ロックされる確率も増加する．また書き込み要求が増加するにしたがって，要求数も増えるため，ロックされる確率も増加する．

また，図9に，サーバにおける平均応答時間を示す．

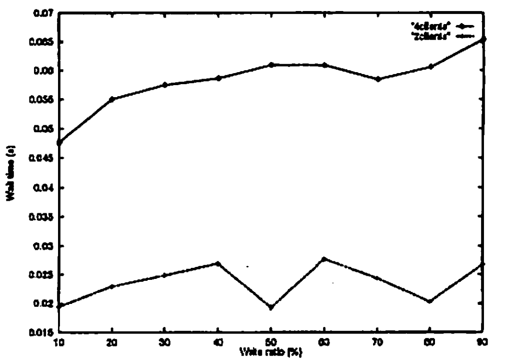

図9　サーバにおける平均応答時間

図9から明らかなように，クライアント数が増加するにしたがって，サーバの待ち行列に入る要求数も増加するため，全体の平均応答時間も増加する．また，書き込み割合が増加するにしたがって，Managing server への要求も増加し，さらにブロック転送などの時間もかかるため，全体の平均応答時間も増加していると思われる．図8の結果も考慮すると，応答時間の大部分は，サービス時間で占められていると考えられる．このように，シミュレーションにより，応答時間を構成する成分を分析することが可能である．

6 おわりに

今回，分散共有メモリアルゴリズムをPVMでシミュレートすることにより，具現化したアルゴリズムのコストおよび応答時間の評価が可能になった．また，データブロックのコピーがおよぼす影響についても考察することができた．実行した環境は，ネットワークで接続された実際の分散環境でありアルゴリズムの有効性なども適切に評価できると思われる．

今後は，他の分散アルゴリズムについても評価していきたい．また，PVMの機能を利用したPVM上のアプリケーションなどについても検討していきたい．

参考文献

[1] Arthur E. Oldehoeft, Rodney R. Oldehoeft:オペレーティングシステムの先進的概念，丸善株式会社 (1990).

[2] 角川裕次，藤田聡，山下雅史，阿江忠:分散アルゴリズムの実験的評価について，情報処理，Vol.34，No.4，pp.1629-1637(1993).

[3] 梶崎浩嗣，黒川恭一:ニューラルネットワークの並列シミュレータについて，マルチメディア通信と分散処理，71-1(1995).

[4] 東田学:ワークステーション上の並列処理 1，UNIXマガジン，ASCII，1995年4月号，pp39-51(1995).

[5] Micahel Stumm, Songnian Zhou: Algorithms implementing distributed shared memory, *IEEE Comput. Mag.*, vol.23, no.5, May(1990).

[6] S. Zhu, M. Stumm, K. Li, and D. Wortman:Heterogeneous Distributed Shared Memory, *IEEE Trans. on Para. and Distri. Sys.*, vol.3, No.5, September(1992).

[7] Al Geist, et al.,"PVM: Parallel Virtual Machine – A Users' Guide and Tutorial for Networked Parallel Computing",*MIT Press*(1994).