ハイパフォーマンス
コンピューティング 55−15
（1995. 3. 9）

# 仮想共有メモリの性能評価

吉井 卓　　岩下茂信†　　村上和彰†

九州大学 工学部 情報工学科
†: 九州大学 大学院総合理工学研究科 情報システム学専攻
〒816 福岡県春日市春日公園6-1

*E-mail: {yoshii, iwashita, murakami}@is.kyushu-u.ac.jp*

マルチプロセッサ型の並列計算機の通信手段の一つに，仮想共有メモリと呼ばれる方法がある．これは，ハードウェア的には分散メモリ型のシステム上に，ソフトウェアにより仮想的な共有メモリ空間を実現するものである．

仮想共有メモリ型のシステムは，スケーラビリティに優れるメッセージ交換型システム上にアプリケーションの開発が容易である共有メモリ型の環境を実現する．

本稿では，仮想共有メモリの機構および動作について述べる．また，シミュレータを用い，さまざまな構成条件における仮想共有メモリの性能を測定し，評価を行う．

# Performance Evaluation of Shared Virtual Memory

Takashi YOSHII　　Shigenobu IWASHITA†
Kazuaki MURAKAMI†

Department of Computer Science and Communication Engineering
Kyushu University
†Department of Information Systems
Interdisciplinary Graduate School of Engineering Sciences
Kyushu University
6-1 Kasuga-koen, Kasuga-shi, Fukuoka 816 Japan

*E-mail: {yoshii, iwashita, murakami}@is.kyushu-u.ac.jp*

A shared virtual memory is a virtual address space that is shared among all processors in a message-passing multiprocessor system. The shared virtual memory provides a conventional shared-memory programming model on message-passing multiprocessor systems, while it keeps scalability of the message-passing multiprocessor systems. This paper evaluates the performance of Kai Li's algorithms for implementing the shared virtual memory with alternating several design parameters such as the number of processors, page size, communication speed, and so on.

# 1 はじめに

マルチプロセッサ・システムにおいて，効率的な並列処理を行うための重要な要因として，いかにプロセッサ間のデータ交換を効率的に行うかがある．この問題を解決する方法の一つに，各プロセッサ間で共有されるメモリを使用する「共有メモリ型」と呼ばれる方法がある．これは，各プロセッサが同一のアドレス空間を持ち，そのアドレス空間を通じてデータ交換を行うものである．共有メモリ型は逐次型の計算機の計算モデルとも馴染みやすく，ソフトウェアの移植が比較的容易なことから，多くの中小規模の計算機のがこの形式を採用している．これら計算機のほとんどは，プロセッサとメモリを高速のバスで結合している．これはバスによる結合は，実装が技術的に容易で安価であり，現実性に富んでいるためである．しかし，各プロセッサからのアクセスがすべてバスに集中するので，多数のプロセッサに対して性能を維持するのは困難である (スケーラビリティに乏しい)．物理的な共有メモリ (真の共有メモリ) を用いた並列計算機は，メモリアクセス時間も短く，性能は高い．しかし，特に大規模なシステムにおいては，真の共有メモリを作ることが物理的，技術的に困難である．

マルチプロセッサ・システムの通信方式には，共有メモリ型以外に「メッセージ交換型」がある．メッセージ交換型の場合，各プロセッサはそれぞれ独自のローカル・メモリを持つ．データ交換は，プロセッサ間でのメッセージ (データ) 送受信によって行われる．メッセージ交換型は共有メモリ型と比較してスケーラビリティに優れているが，メッセージ通信を陽に記述しなければならない，プログラムがハードウェアの構造に依存したものになる，などプログラム記述に関してて難があるとされる [5].

そこでメッセージ交換型のマルチプロセッサ上に仮想記憶機構などを用いて，仮想的にメモリ共有を実現する方法が考えられ，これまでさまざまなインプリメンテーションが発表されてきた [4]. [1]

本稿では以下，第2章で仮想共有メモリについて説明し，第3章で評価の目的・方法を述べ，第4章で実験の結果を示し,, その考察を行う.

[1]本稿ではこれらを一括して，このタイプのシステムの草分け的な存在である Kai Li らの Ivy システムに関する論文 [3] での命名に従って，仮想共有メモリ (Shared Virtual Memory) システムと呼ぶ.

# 2 仮想共有メモリ

## 2.1 仮想共有メモリとは

仮想共有メモリシステムとは，物理的な共有メモリを持たないマルチプロセッサ・システムにおいて，メモリ上のデータ・ブロックを通信で交換することによって，各プロセッサが単一の仮想アドレス空間を共有するシステムである.

基本的なストラテジは，アクセスしたいデータがローカルなメモリになければ，それを所有している他のプロセッサと通信しローカル・メモリにコピーしてからアクセスする，というものである．いわば 仮想共有メモリシステムは，物理メモリを仮想共有メモリ空間のキャッシュとしているともみなせる．そのため仮想共有メモリでもマルチプロセッサのキャッシュと同様なコヒーレンス問題が発生し，その制御も必要となる．これらを実現するため，仮想共有メモリには，

- プロセッサのメモリ管理ユニット (MMU) 等の機能を用いた，仮想共有メモリ空間と物理メモリのマッピングの操作，
- プロセッサ間通信によるデータおよびコヒーレンス制御情報等のやりとり

といった機能が必要となる.

仮想共有メモリは，基本的には上記の通り必要に応じて他のノードからローカル・メモリにデータをコピーしてアクセスするのであるが，データの管理，コヒーレンス制御などについては，さまざまな方法が考えられ，多くの方式が提案・実装されている.

以下では仮想共有メモリの草分け的存在である Yale 大学の IVY(Integrated shared Virtual memory at Yale) について説明する [3][1].

## 2.2 IVY(Integrated shared Virtual memory at Yale)

IVY は，Yale 大学の Kai Li らによって，リングネットワークで結合された Apollo DOMAIN ワークステーション上に実装されたシステムである．IVY を構成するハードウェアは，一般的なワークステーションである．仮想共有メモリの機能は仮想記憶と同様にワークステーション上の MMU を用いてソフ

トウェアで実現されている．このソフトウェアをメモリ・マッピング・マネージャと呼ぶ．メモリ・マッピング・マネージャはページ・フォルトなどのMMUの発生する例外によって起動され，プロセッサの物理メモリと仮想共有アドレス空間とのマッピング，および，コヒーレンス制御を行う．以下，IVYの詳細について述べる．

**処理単位**

仮想共有メモリ空間は固定長(1Kbyte)の「ページ」に分割して管理される．各ページは「読み出し可」と「読み書き可」の2つ状態をとり得る．各プロセッサは仮想アドレスでのページング方式の仮想記憶と同様に仮想空間の全ページについて物理メモリとのマッピング状態，ページ属性その他の管理情報を保持する．

**コヒーレンス制御**

IVYで使われるコヒーレンス制御の手法はいわゆるキャッシュでいうところのディレクトリ方式であり，書込み時には**write invalidate**を用いる．つまりあるプロセッサが書き込みを行うと，書き込みの行われたページはそのプロセッサの物理メモリにのみマップされる．そしてディレクトリに記録されたコピーの所有者の情報にしたがって，コピーを持つ他のプロセッサの当該ページのマッピングを無効化するのである．

**write invalidate**に対し当然**write update**も考えられるが，すべての書込みごとに「例外の発生」と「メッセージの送信」が行われるので，高速なハードウェアの支援がなければ，処理にかかるコストが大きすぎて実用にならないと考えられる．

**アルゴリズム**

MMUで例外が発生する事項と，その時の処理内容は以下の通りである．

- マップされていないページへの書込み
  1. オーナ（と，おぼしきプロセッサ）に書込み権を要求する(受けとったプロセッサがオーナでなければ，そこが同じようにオーナに転送する(forwarding))．
  2. オーナからページを受けとりローカルメモリにコピーし「書込み可」でマップし，ページのオーナとなる．
  3. コピーを持つプロセッサのリストを受けとり，自分以外のプロセッサの持つコピーを無効化(Invalidate)する．
- マップされていないページの読み出し
  1. オーナに読み出し権を要求する．
  2. オーナはそのプロセッサをコピーのリストに追加し，ページの属性を「読み出し可」にする．
  3. オーナからページを受けとりをローカルメモリにコピーし「読み出し可」でマップする．
- 書込み禁止ページへの書込み
  1. 当該ページの自分以外のコピーを無効化する．
  2. 当該ページを「書込み可」とする．

このアルゴリズムを，「動的分散マネージャ方式」と呼ぶ．

仮想共有メモリのアルゴリズムには，この他にも，所有者・ページの管理の方法によって，以下の方式が考えられる[3]．

**集中マネージャ方式** システムを構成するプロセッサのうちどれかひとつを管理用と定める．各々のプロセッサは，アクセス権変更，ページのコピー等のリクエストを管理用プロセッサに送り，管理プロセッサがリクエストを処理，もしくはページの所有者の情報にしたがって該当するプロセッサに転送する．

**静的分散マネージャ方式** 各ページに対し，例えばプロセッサをサイクリックに割り振るなどして管理するプロセッサをあらかじめ決めておく．その他は集中マネージャ方式と同様である．

# 3 性能評価方法

## 3.1 評価の目的

今回の評価の目的は

- 仮想共有メモリ・システムがどれほどの能力を持っているか．

- 仮想共有メモリの性能が，相互結合網の構造・通信速度などのようなハードウェアの特性とどのような関係にあるか．

を，知ることにある．

## 3.2 評価尺度と評価項目

評価尺度として，

- プログラムの実行時間
- 実行効率

の2つを用いる．実行効率は，

- 発生した無駄なイベントの実行割合

に着目して観測することにする．

また，評価項目として，以下のものを用いる．

- プロセッサ数：1 ～ 32 台
- 相互結合網の通信速度：0.001 ～ ∞ Byte/単位時間
- ページサイズ：4 ～ 8192 Byte

## 3.3 評価対象

評価対象となるハードウェアおよびソフトウェアは，以下の通り．

仮想共有メモリマネージャのアルゴリズムはIVYの論文[3]を参考にして

- 静的分散マネージャ
- 動的分散マネージャ

の2つのアルゴリズムを実験する．

このため，ハードウェアもページングによるMMUを備えていることを前提とする．ただし相互結合網などのハードウェアは構造をあまり明確に限定しないで，ネットワークの特性をパラメータとして変化させることにより調べる．

## 3.4 シミュレータ

### 3.4.1 シミュレータ・プログラムの構造

シミュレータのプログラムはC言語を用いUNIX上で作成した．プログラミングおよび実験を容易にするために，

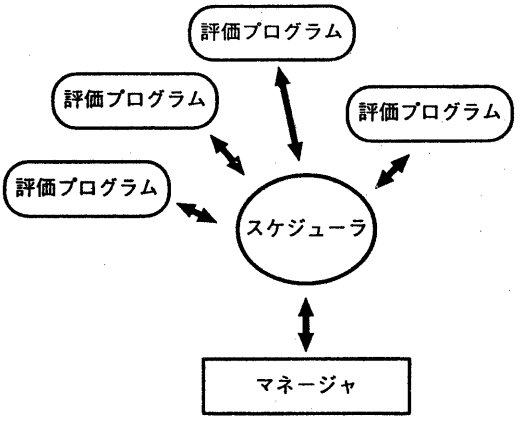


図 1: シミュレータの構造

- メモリ・マネージャのシミュレータ・プログラム (以下，マネージャ)
- 各PEで実行すべきベンチマーク・プログラム (以下，評価プログラム)

を別々のUNIXのプロセスとして作成する．しかし，UNIXのプロセスには時間が均等に割り振られるわけではないので，プロセス間で時間のズレが発生する．そのため，さらに，

- 上記の2種類のプログラムの同期をとるプログラム (以下，スケジューラ)

を設け，これらをパイプで結合し，メッセージの交換で時間軸の一貫性を保つ構造とした．シミュレーション時のプロセスの結合関係は，図1のようになる．

### 3.4.2 シミュレータの動作

シミュレータの動作の概略を以下に示す．

1. 各評価プログラムは独立したUNIXプロセスであるため，独立に実行される．
2. 評価プログラムがメモリ・アクセスのかわりにスケジューラにメッセージを発行する (予め，メモリ・アクセスの部分をメッセージ送信の関数に置き換えておく)．
3. スケジューラは一旦このメッセージを蓄え，スケジューラの管理する各プロセッサの時計を

基にこれらメッセージをソートし，早い順にマネージャへと送る．

4. マネージャは受けとったメッセージに従って，メモリ管理動作をシミュレートし，結果をスケジューラ経由で評価プログラムへ返す．

#### 3.4.3 時間単位

今回作成したシミュレータ内部での時間は，メモリ・アクセスを除く全命令の実行時間を等しいと仮定し，これを1単位時間とした，すなわち,「1単位時間=プロセッサが(メモリ・アクセスを除く)命令を1つ実行する時間」である．以下の説明および図表における実行時間の単位として単位時間を用いる．

### 3.5 評価プログラム

**行列の積計算**

正方行列 A,B,C においてCにAとBの積を求めるプログラムである．行列の要素は int 型 (4byte) とし，A,B,C それぞれメモリの固定番地よりC言語の2次元配列と同様の配列で割り付ける．また，A,B の内容ににについてはシミュレーションを開始する前にあらかじめメモリにデータを書き込んでおく．特徴としては以下の2点があげられる．

- 計算コストが小さい (メモリアクセスの占める割合が多い)
- 一つのプロセッサから連続したアドレスでの読込み/書込みが行われる

## 4 評価結果および考察

### 4.1 実験結果

#### 4.1.1 プロセッサ数 vs 処理速度向上

図2に，2つの行列 (64 × 64) の積の計算時間とプロセッサ数の関係を示す．結合網の通信速度を1(ワード/単位時間) として，シミュレーションを行った．

- ページ・サイズが128ワードの場合，ほぼ理想に近い速度向上を示している．行列の積計算という問題の並列実行性が高いこと，および，通信速度が1ワード/単位時間と高めであることが原因であろう．
- ページ・サイズが512ワードの場合は，プロセッサ数8までは理想的だが，そこから急に速度向上が悪くなっているのがわかる．図3に，この場合にやりとりされたメッセージ数の関係を示す．プロセッサ数16以上で書込み要求およびそれにともなう無効化要求が爆発的に増加していることがわかる．これは行列のサイズが $64 \times 64 = 4096$ ワードであるのに対し，ページ・サイズが512ワードと大きいため，行と列の積和をとって結果を行列に書き込むところで，プロセッサ数16以上では同一ページへの書込みが発生して，ページの移動が頻発するいわゆる「ページのピンポン状態」になっていることが原因であると考えられる．

#### 4.1.2 通信速度 vs 実行時間

図4に結合網の通信速度と実行時間の関係を示す．実行したプログラムはページサイズを大きくして，先に述べた「ピンポン状態」を起こさせたものである．この図から，この状態頻発すると，ローカルメモリアクセスと同等の通信速度をもってしても，実用に耐え得る性能が出せないことがわかる．この状態を回避する工夫が必要だと思われる．

**ページサイズ vs 効率** 図5にページサイズ (4byte ～4096byte) と効率 ((全時間 − マネージャの処理にかかった時間)/全時間 × 100) との関係を示す．

### 4.2 考察

今回の実験から，さきに述べた「ピンポン状態」にならなければ実用に値する性能が得られそうなことがわかった．

「ピンポン状態」への対処法としては

- ページを考慮してデータの配置を行う
- バーストアクセスの際にページを明示的にロックする機能 [7]
- ページの入れ換え時間間隔に制限を設ける [4]

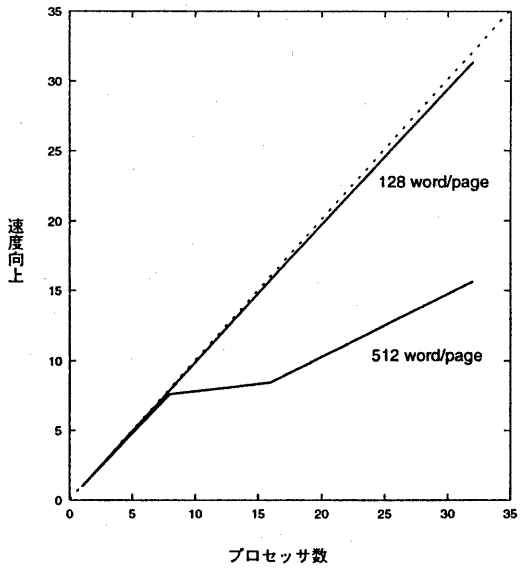


図 2: プロセッサ数 vs 処理速度

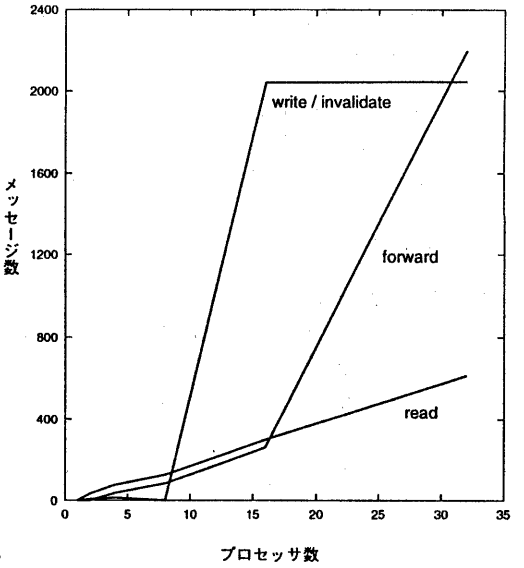


図 3: プロセッサ数 vs メッセージ数

などの方法が考えられる.

しかしページなどのハードウェア構造をユーザが意識してプログラミングをしなければならないとなると, メッセージ交換型のプログラミングに対してのプログラミングの優位性を充分に享受できない.

この問題を解決するには, コンパイラ等の支援が必要となる.

ハードウェアから, システム・ソフトウェア, コンパイラまで, システム全体を一貫して設計することによって, 仮想共有メモリは十分に高い性能を発揮できると考えられる.

## 5 おわりに

本稿では仮想共有メモリの評価を行った. 今後は, 現在我々が開発を進めているオンチップ・マルチプロセッサ **PPRAM**[2] への適用可能性を検討していく計画である.

## 謝辞

日頃より御討論頂き, 貴重なご意見を数多く頂きます九州大学 大学院総合理工学研究科 安浦寛人教授, 岩井原瑞穂助手, 安浦研究室の諸氏, ならびに, Hokke-Club の諸氏に深く感謝します.

## 参考文献

[1] 並列アーキテクチャ技術動向委員会, "並列アーキテクチャに関する技術開発動向調査研究報告書," 第 5.1 節, pp.77–81, 1993 年 3 月.

[2] 村上, 吉井, 岩下, "21 世紀に向けた新しい汎用機能部品 PPRAM の提案," 情処研報, vol.94, no.91, pp.49–56, 1994 年 10 月.

[3] Li, K, Hudak, P, "Memory Coherence in Shared Virtual Memory Systems," *ACM Transactions on Computer Systems*, vol.7, no.4, pp.321–359, November 1989.

[4] Nitzberg, B, Lo, V, "Distributed Shared Memory: A Survey of Issues and Algorithms," *COMPUTER*, August 1991.

[5] 高橋義造編, "並列処理機構," 丸善株式会社, 1989 年.

[6] Culler, D. et al., "LogP: Towards a Realistic Model of Parallel Computation," *Proc. 4th*

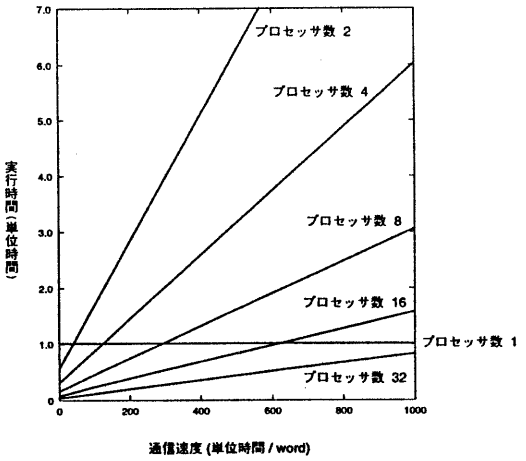


図 4: 通信速度 vs 実行時間

*Symp. Principles & Practice of Parallel Programming (PPOPP)*, pp.1–12, May 1993.

[7] 中條拓伯, Miura, N, 和田耕一, 金田悠紀夫, “ネットワーク結合型並列計算機上の仮想共有メモリシステムにおける無矛盾化プロトコルの性能評価とハードウェアによる実現,” *JSPP '91*, pp.45–52, 1991 年 5 月.

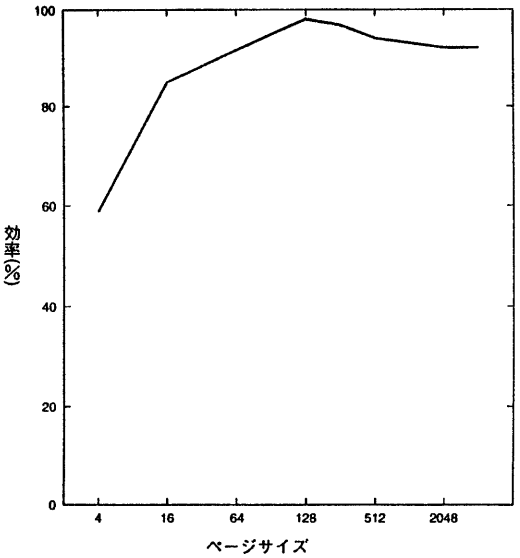


図 5: ページサイズ vs 効率