4F−8

ローカルメモリを持つ共有メモリ型並列計算機のための仮想記憶の実装と評価*

服部 大　山本 淳二　天野 英晴†

慶應義塾大学‡

1 はじめに

共有メモリ型並列計算機は規模が大きくなるほど、ネットワークやバスのトラフィックが増え、共有メモリのアクセス時間が大きくなり、ボトルネックとなる。それを補うためにキャッシュやローカルメモリが付加されている。ローカルメモリはネットワークやバスを介さずにプロセッサに接続したメモリであり、ネットワークやバスとは無関係なので、当然、共有メモリよりアクセス時間は小さい。

ユーザはプログラムの処理速度を上げるためにローカルメモリを有効に使うプログラミングを行う。しかし、プログラムが大規模になると物理メモリに入りきらなくなることがあり、物理メモリの大きさを意識して、プログラミングすることはユーザにとって大きな負担となる。そこで、仮想記憶システムを導入することで、ユーザに物理メモリ以上のアドレス空間を提供し、プログラミングしやすい環境を与える。

今回は、ローカルメモリを持つ共有メモリ型並列計算機を対象とした仮想記憶管理システムを実装し、その評価を行った。

2 設計と実装

2.1 並列計算機 ATTEMPT-0

今回実装を行った並列計算機 ATTEMPT について説明する。

ATTEMPT は慶應義塾大学理工学部と Allumer 株式会社が共同開発したバス結合型並列計算機である [1]。プロトタイプ ATTEMPT-0 は PU(Processing Unit) に MC68030、FPU(Floating Point Unit) に MC68882 を用い、4MB のローカルメモリを持っている。そして最大 20 枚のプロセッサボードが非同期バス Futurebus(IEEE P896.1) によって結合されており、バスを介して最大 16MB の共有メモリをアクセスできる。

共有メモリに対してはスヌープキャッシュが存在し、ライトスルー方式で 64KB の 2 ウェイセットアソシアティブマッピング制御である。プロセッサ間の同期機構には、一種の共有メモリである Synchronizer というハードウェアが提供されていて、不可分命令 Fetch&Dec を提供している。メモリのアクセス時間を表 1に示す。

2.2 仮想記憶

仮想記憶システムの目的は、ユーザが物理ローカルメモリの大きさを意識せずに、プログラミングが行えるような環境を提供することである。

メモリシステム	アクセスの種類	時間
ローカルメモリ		250 ns
共有メモリ	Read hit	150 ns
	Read miss	18200 ns
	Write	750 ns

表 1: メモリシステムのアクセス時間

今回は、一つのプロセッサには一つの静的なタスクしか実行されないとして、単一アドレス空間をページングにより管理するように実装した。仮想記憶管理は、MMU(Memory Management Unit) からの bus error を受けて、それに応じた処理を行う。

一般的な計算機では仮想記憶を行う際に二次記憶としてハードディスクのような補助記憶装置を用いる。しかし、並列計算機において、全てのプロセッサに補助記憶装置を接続することは計算機システム自体が高価になり難しく、補助記憶装置を設ける場合は、共有バスに接続するか、いくつかのプロセッサに対して一つの補助記憶装置を想定する方が妥当である。そこで、本稿の実装では、ローカルメモリのスワップ領域を共有メモリ上に設け、共有メモリのスワップ領域として、補助記憶装置を用いることにする。

3 評価

3.1 測定

評価には行列同士の乗算を行うプログラムをアプリケーションとして用い、ページサイズを 1KB、2KB、4KB、8KB と変えて測定した。また、ページ置換えアルゴリズムとしては FIFO と疑似 LRU の一種であるクロックアルゴリズムの 2 種類について測定を行った。

本実装の対象である ATTEMPT-0 には 4MB の物理ローカルメモリが存在するが、今回は物理ローカルメモリが不足する場合の評価を行うので、仮想記憶時に使用する物理ローカルメモリを 128KB に制限した。また、評価の対象をローカルメモリの仮想化に焦点を絞り、共有メモリはスワップをさせず、ローカルメモリのスワップだけを行うようにした。

実行時間の測定結果を図 1に示す。normal は仮想記憶を行わずに物理ローカルメモリを制限しない場合で、shared はデータを全て共有メモリに配置した場合である。括弧内の数字はページサイズを表す。

また、使用するプロセッサ数を変えて台数効果を測定した。結果を図 2に示す。

3.2 検討

図 1の normal は純粋にアプリケーションの実行時間を表しており、これは最速実行時間である。仮想記憶システ

*An Implementation and Evaluation of Virtual Memory System on a Shared Memory Multiprocessor with Local Memory

†Dai HATTORI, Junji YAMAMOTO, Hideharu AMANO

‡Keio University

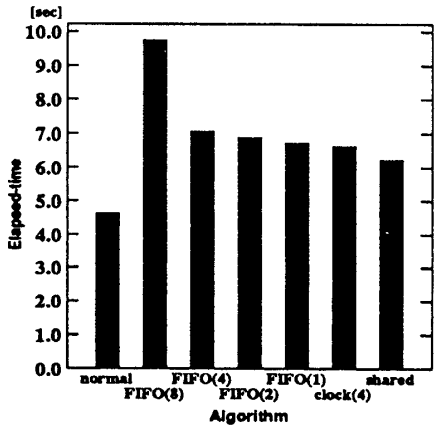

図 1: 実行時間 (4PU)

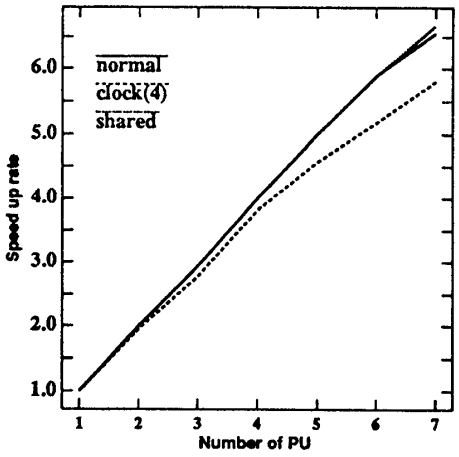

図 2: 台数効果

ムを用いた場合の実行時間のうち、normal に比べ遅くなった分は、仮想記憶システムが使用した時間 (主にスワップ時間) である。実行時間で見ると、仮想記憶システムを導入した時、最も効率の良いアルゴリズム、ページサイズの場合でも、normal に比べ 37%遅くなった。しかし、ユーザが物理ローカルメモリ以上のアドレス空間を使えるようになり、ユーザから物理メモリの大きさを意識するという負担を削減したことを考慮すると、さほど速度低下として問題ではないと考えられる。

並列計算機の主眼である台数効果 (図 2) という点でも、アプリケーションの並列性能をそれほど損なわずに、ユーザの期待通りの性能向上が得られている。

ページサイズを変えた場合、8KB の場合だけがかなり効率が悪いが、これは測定に用いたアプリケーションのメモリ参照パターンに依存するもので、ページ中に無駄な領域が多いためだと考えられる。また、ページ置換えアルゴリズムを変えた場合は、ほとんど実行時間は変わらなかった。一般に、最適のアルゴリズムやサイズはアーキテクチャやアプリケーションに依存するところが多いので、効率を向上させることは今後の課題である。

3.3 共有メモリ vs ローカルメモリ

他の並列計算機のアクセス時間を調べてみると、例えば IBM ACE では共有メモリのアクセス時間はローカルメモリの 2 倍程度 [2] であり、BBN GP1000 で 12.5 倍、TC2000 で 3.5〜29 倍程度 [3] である。このような計算機環境では、共有メモリと比較したときのローカルメモリのアクセス時間での利点がはっきりし、ローカルメモリの利用効率が大きく処理能力に影響する。よって、データを全て共有メモリにおいた場合よりも実行時間の短縮が可能になると考えられる。実際に、仮想記憶を用いて共有メモリのキャッシュを行った場合、実行時間が速くなったという研究がなされている [2]。

また、ハードウェアによる共有メモリキャッシュは、バス結合型並列計算機では比較的容易に実装可能だが、ネットワーク結合型の並列計算機では難しい。よって、ローカルメモリと共有メモリのアクセス時間の差が確実に生じるので、ローカルメモリの仮想記憶システムは十分有効であると考えられる。

今回の実験では、ローカルメモリを使用せずデータを全て共有メモリに置いた場合が、仮想記憶システムの最適な場合よりも良い性能であったが、この要因は、共有メモリキャッシュにヒットした場合、ローカルメモリより共有メモリキャッシュのアクセス時間の方が速い点にあると考えられる (表 1)。

4 おわりに

仮想記憶システムによって、ユーザに物理ローカルメモリの大きさを意識させないプログラミング環境を提供して、ユーザの負担を削減ができるようになった。ローカルメモリのスワップ領域を共有メモリにとることによる性能低下はほとんどなかった。ハードウェアのキャッシュのついていないバス結合型並列計算機やキャッシュの実装の難しいネットワーク型並列計算機においても、ローカルメモリの使用量を増やすことで、性能の向上が十分期待できると考えられる。さらに、マルチタスク環境では多重仮想記憶が必要不可欠であり、物理ローカルメモリの使用量は増えるので、有用性は大きくなると考えられる。今後の課題としては、他の並列計算機での実装および測定を行い、他の計算機環境との統一的な環境を提供できるようにし、ローカルメモリの有用性や利用効率の向上を示す必要がある。

参考文献

[1] H. Amano, T. Terasawa, and T.Kudoh. Cache with synchronization mechanism. In *Proceedings of IFIP Congress 89*, August 1989.

[2] William J. Bolosky, Robert P. Fitzgerald, and Michael L. Scott. Simple but effective techniques for NUMA memory management. In *Proceedings of the Twelfth ACM Symposium on Operating Systems Principles*, December 1989.

[3] Richard P.LaRowe Jr., Carla Schlatter Ellis, and Laurence S. Kaplan. The robustness of NUMA memory management. In *Proceedings of the Thirteenth ACM Symposium on Operating Systems Principles*, October 1991.