計算機アーキテクチャ 113−22
（1995. 8. 24）

ユーザプログラム制御階層メモリシステム

牧 晋広　　岡本 秀輔　　曽和 将容
makin@sowa.is.uec.ac.jp
電気通信大学大学院情報システム学研究科曽和研究室
〒182 東京都調布市調布ヵ丘 1-5-1

プロセッサの高速化に対応するため、メモリの高速大容量化が要求されている。現在では、コスト的原理的に難しいメモリの高速大容量化を実現するため、キャッシュメモリシステムが用いられている。しかし、キャッシュメモリシステムには、キャッシュミスが発生する。キャッシュミスが頻繁に発生すると、プロセッサの高速性をスポイルすることになる。

キャッシュミスが頻繁に発生する原因の１つにキャッシュメモリと主メモリの内容の入れ換えを固定アルゴリズムで行っていることがある。そこで、プログラマが、一般のプログラムとともにキャッシュメモリを操作するプログラムも記述することで、キャッシュミスの極限までの減少を狙う。本稿では、本研究システムの原理、問題点を明らかにした後、簡単な評価を行う。その結果、キャッシュミスに相当するデータフォルトの大幅な削減を示した。

User Program Controlled Hierarchical Memory System

Nobuhiro Maki　　Shusuke Okamoto　　Masahiro Sowa
Graduate School of Information Systems
University of Electro-Communications
1-5-1 Chofugaoka,Chofu,Tokyo,182,JAPAN

Execution ability of processors has been getting higher and higher. As a result,it needs memories which is big size and can access within slight time. It is difficult to realize in its construction and its principle. So recently cache memory system is used. But,cache miss sometimes occurs in cache memory system. If cache miss occurs frecuently,it spoils the processor's abilty.

One of the reasons why cache miss occurs frecuently is to replace the contains between cache memory and main memory by constant algorithm. Therefore the programmer who knows the data movement in the programs programs not only application program but also specified programs which replace the contains. That had better lead the slightest cache misses latency. In this paper,we propose the new memory system and argue the solution of the problem which the system have potentially. And then we show the simple performance review of this system. The result shows that this system can reduce the cache miss latency extremely.

1 はじめに

現在、あらゆる方向から、コンピュータシステムの高速化の研究・開発が行われている。その中でもプロセッサは、著しく高速化し発展している。これらの高速化されたプロセッサを最大限利用する為には、高速で大容量なメモリが必要とされる。しかし、そのようなメモリを製作することは、原理的、コスト的な理由で難しい。そこで、キャッシュメモリシステムとして高速大容量なメモリを疑似的に実現している。キャッシュメモリシステムには、キャッシュミスが発生する。プログラムによっては、キャッシュミスが頻繁に発生し、コンピュータの処理時間を大幅に増大させる。[1]

キャッシュミスが頻繁に発生する1つの原因は、キャッシュメモリと主メモリの内容の入れ換えを、固定のアルゴリズム (LRU法など) で行っていることにある。固定のアルゴリズムでは、プログラムごとに、キャッシュメモリと主メモリ間の最適な内容の入れ換えを行うことは、不可能である。

そこで、プロセッサ上で実行されるプログラムごとに、階層メモリ間の内容を操作する、メモリ操作プログラム (このプログラムと区別するために一般のプログラムを基本プログラムと呼ぶ) を作成し、メモリ操作プログラムを基本プログラムと並列に動作させる。これは、プログラム内でデータが何時何処で必要になるかを最も知っているプログラマが、基本プログラムの他に更にメモリ操作プログラムも記述することを意味する。メモリ操作プログラムは、プログラマが知っている基本プログラムのデータ参照動作にしたがって記述されるので、キャッシュミスを極限まで減少させることができるはずである。本研究は、この原理による大容量なデータを扱う高速な階層メモリに関するものである。

本研究では、階層メモリをプログラムで操作する為、プログラマにキャッシュメモリの内容が見えてしまう。そのため、キャッシュメモリのキャッシュ(隠れる) と言う用語が使用できない。そこで、キャッシュメモリに相当するメモリを高速メモリと呼ぶことにする。この研究システムをユーザプログラム制御階層メモリシステム:UPCHMS(User Program Controlled Hierarchical Memory System) と呼ぶ。本論文は、データメモリ中心に話を進める。

2 UPCHMS の基本原理

図1は、UPCHMS のハードウェア構成である。UPCHMS のデータメモリは、超高速メモリ:VHM(Very High-speed Memory)、高速メモリ:HM(High-speed Memory)、主メモリ:MM(Main Memory) の3階層からなる。超高速メモリは、レジスタに相当し、高速メモリは、キャッシュメモリに相当する。データメモリには、3つのプロセッサユニット PU、HU、MU が接続される。MU は MM と HM 間のデータ移動を受け持ち、HU は HM と VHM 間のデータ移動を受け持つ。PU は基本プログラムから HM と VHM 間のデータ移動を除いた処理、すなわち計算処理を行う。PU が実行するプログラムのことを処理プログラムと呼ぶ。各プロセッサが実行するプログラムは図2のようになっており、それぞれ IM-PU、IM-HU、IM-MU と呼ばれる命令メモリに格納されている。各プログラムはスレッドと呼ばれており、スレッド間命令の従属関係は、命令間のアークに沿ってトークンを送受することによって高速になされる [3]。

このプログラムは (a1+a2)x(a3-a4) を計算するプログラムであるが、ここで SPPU はプロセッサ P U用のスレッド、MMPHU は HU 用の，MMPMU は MU 用のスレッドである。例えば、H1 の ldr H0,V0 は、高速メモリの H0 番地のデータを超高速メモリの V0 番地に1ワード転送することを表す。H3 の sth は、超高速メモリから高速メモリにデータを1ワード格納することを

表す。M1 の lhm Data,H0 は、主メモリの Data 番地のデータを高速メモリの H0 番地に 1 ワード転送することを表す。M5 の shm は、高速メモリから、主メモリにデータを 1 ワード格納することを表す。

このプログラムでは、SPPU スレッドの P2 命令　add v0,v1,v2 で使われる MM メモリ上のデータ Data+1 が、M2 命令によって HU メモリに持ってこられ、そのデータが H2 命令によって VHM メモリに持ってこられて使われている。また、P5 で作られたデータが H7、M5 命令によって MM メモリに格納されている。このように本システムでは、プログラムの振る舞いを一番良く知っているプログラマが、メモリ操作プログラム (MMP) を書き、それにしたがって SPPU で必要なデータが、MMPHM,MMPMM スレッドによって並列に前もって持ってこられることが大きな特徴である。

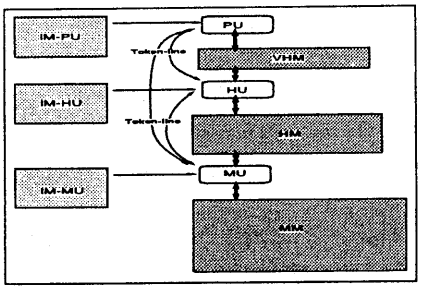

図 1: UPCHMS のハードウエア構成

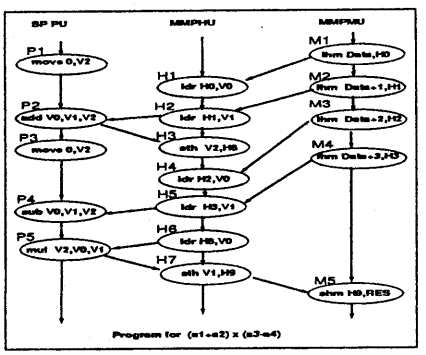

図 2: UPCHMS のプログラム

3　データ待ち

キャッシュメモリシステムのキャッシュミスに相当する現象が起こる。これをデータ待ちと呼ぶ。プログラムによっては、処理プログラムの命令が必要とするデータ (以後必要なデータと呼ぶ) が使用される直前に決定されることがある。この場合、主メモリから、そのデータを超高速メモリへ転送する必要があるが、主メモリからの転送が処理プログラムの動作に間に合わない時には、データを待つ状態が発生する。これが、データ待ちであり、データ待ちで必要とされる時間をデータ待ち時間と呼ぶ。データ待ちは、階層メモリシステムが、本質的に持つ性質であり、回避することは難しい。

図 3は、UPCHMS のデータ待ちの動作を示している。P1、H1、M1 などの四角の枠は、命令を表し、縦方向は、時間を表している。P1 で、3 命令後の命令 (P4) の必要とするデータが決定されるが、P1 で決定されたデータは、主メモリにしか存在しないため、そのデータの情報を、HU に伝える (P1)。PU は HU に情報を伝えた後、そのデータに依存しない命令の処理を行う (P2、P3)。HU は、PU から伝えられた情報を MU に伝える (実際は、高速メモリにその情報を転送する：H1)。MU は、HU から高速メモリ経由で、PU が必要とするデータの情報を得て、そのデータを高速メモリに転送する (M1)。HU は、M1 により転送されたデータを超高速メモリに転送し、PU に転送したことを伝える (H2)。PU は、HU からの情報を受け取り、処理を再開する (P4)。しかし、PU は、P3 の処理を終えた後、主メモリから転送されるデータを受け取るまで、しばらく、処理を中断している。(Data Waiting)

本システムでは使用データの決定時期とそのデータの使用時期との時間距離が大きければ大きいほどデータ待ちになる確率は減少する.

4 UPCHMSのコヒーレンス問題

UPCHMSがプログラムを実行する際、主メモリ上にある1つのデータが高速メモリ上に複数コピーされるように動作する場合がある。この状態に注意をしてプログラミングしないと、UPCHMSが正しく動作しなくなる。この現象をUPCHMSのコヒーレンス問題と呼ぶ。

4.1 原因

UPCHMSのプログラム実行中、PUの演算処理と並行して、MUは、あらかじめ高速メモリにPUが将来必要とするデータを転送して準備をしておく。この時、UPCHMSのプログラムを記述する時点で、PUが必要とするデータがあらかじめ特定できるものを、静的データと呼ぶ。また、UPCHMSのプログラムを記述する際に、必要なデータを特定することができない場合も存在する。その場合、UPCHMSのプログラム実行中に、その必要なデータを特定できた時点で、UPCHMSは、そのデータを主メモリから高速メモリに転送する。このようにプログラム記述の際に、PUが必要とするデータを特定することできないものを動的データと呼ぶ。

UPCHMSのプログラム実行中に、高速メモリ内に静的データが存在する状態を考える。この時、PUが1つの動的データを必要になり、その動的データが高速メモリ内の静的データと重複するかどうかの判定処理を行わない場合、MUが主メモリから高速メモリへ、その動的データを転送することになる。その後、HUが高速メモリから超高速メモリへその動的データを転送する。図4は、この状態を示している。あらかじめ静的データａａａが転送されている高速メモリ上に、新たに(静的データと同じデータである)動的データａａａが転送される。

もし、そのPUが必要とする動的データ(以下、動的データと呼ぶ)がPUの処理により更新され、高速メモリ上に書き込まれたとすると、同一であるべきデータが高速メモリ内で異なるデータになる。本来PUは、更新された後のデータを参照するべきであるが、更新される前の静的データを参照する可能性がある。この現象が起こると、UPCHMSのプログラムが正しく動作しなくなる。UPCHMSを正しく動作させるために重要な問題となる。図5は、この状態を示している。PUによりデータａａａが、更新されａａａ'になっている。そのため、以前から高速メモリに存在する静的データと動的データとが異なることになる。

4.2 回避法

基本的に、このUPCHMSのコヒーレンス問題を回避するためには、主メモリ上の1つのデータを高速メモリ内に重複してコピーをさせなければよい。

そのためには、UPCHMSのプログラムに高速メモリ内で重複する可能性のあるデータを何らかの方法で判定させる必要がある。

1 PUが使用するデータの総量が高速メモリの容量よりも少ない時、全てのデータを高速メモリに転送する

この方法は、高速メモリ上にPUが使用する全てのデータを転送することで、動的データのロードが主メモリから行なわれることを無くすことができる。すなわち、必ず高速メモリにPUが必要とするデータが存在するために、動的データの参照も高速メモリ上からロードすれば良く、更新したデータも同じ位置に書き戻せば、コヒーレンス問題は生じない。

2 PUが参照するデータ列に範囲を定めて、その範囲全てを高速メモリに転送する

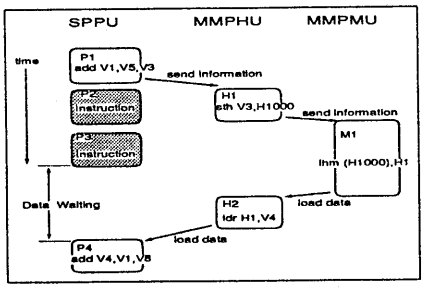

図 3: UPCHMS のデータ待ち

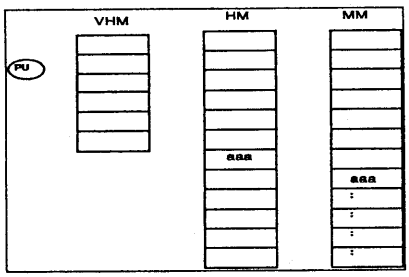

図 4: UPCHMS のコヒーレンス問題の発生 1

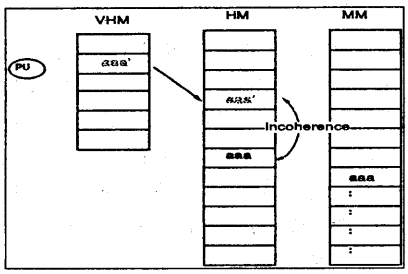

図 5: UPCHMS のコヒーレンス問題発生 2

この方法は、1 の方法で、PU が使用するデータの総量が、高速メモリの容量よりも大きい時に使用する方法である。PU が参照するデータの局初性を利用するもので、あらかじめ定めた範囲の主メモリ上のデータ列を全て高速メモリに転送し、それらのデータの範囲を HU が管理する。(先頭のデータと転送したデータ列のサイズを管理すれば十分である。) もし、動的データが必要になった場合、高速メモリに転送されたデータ列にその動的データが含まれているかどうかを判定させる。もし、高速メモリ上にあれば、高速メモリ上のデータを超高速メモリに転送する。そうでなければ、動的データは、高速メモリ上に無いことになるので、主メモリから転送する。この方法を実現するために必要なプログラムの処理は、高速メモリ上のデータ内に動的データが含まれているかどうかをチェックすれば良いので、それほど多くの処理を必要としない。

3 動的データが、高速メモリで重複するデータかどうかを高速メモリ内の特定のデータを管理することにより判定を行う

上の方法 2 では、PU の参照の局所性を利用し、主メモリ上にあるデータ列で範囲を定めて高速メモリに転送し、そのデータ列と動的データとを HU がチェックすることで、コヒーレンス問題を回避するのに対し、この方法は、PU が使用するデータの容量が、高速メモリの容量よりも大きく、更に PU のデータ参照に局所性がみられない時に用いる方法である。すなわち、高速メモリ内に存在するデータで、動的データと重複する可能性のあるデータを特定し、それらのデータと動的データとをそれぞれ比較判定することで、コヒーレンス問題を避ける方法である。

5 性能評価

UPCHMS でどの程度の性能が得られるかを評価する為に、シミュレーションによる性能評価を行った。

5.1 前提

UPCHMS のメモリの容量、アクセス速度などは、現在利用されているコンピュータシステムのそれに近いものに設定する。

シミュレーションの中で、UPCHMS の性能と比較評価する為に、キャッシュメモリシステムを備えたコンピュータシステム (キャッシュコンピュータと呼ぶ。) との比較も行う。キャッシュコンピュータのメモリ容量は、UPCHMS のそれと同じ値を用いるが、アクセス速度は、転送方式の違いを考慮して変更されている。ここで言う転送方式の違いとは、UPCHMS の高速メモリ、主メモリ間の内容の入れ換えは、1 ワードで行うのに対し、キャッシュメモリシステムのキャッシュメモリ、主メモリ間の内容の入れ換えは、ブロック (4 ワード) で行うことを意味する。評価における主メモリへのアクセス速度は、評価別で明示しない限り UPCHMS 3 クロック/ワード、キャッシュメモリシステム 2.25 クロック/ワードとする。

図 6は、UPCHMS、キャッシュコンピュータ (CCMP) のメモリ容量を示している。縦は、UPCHMS と CCMP の 2 つの項目に分かれている。横は、UPCHMS の場合は、順に超高速メモリ (VHM)、高速メモリ (HM)、主メモリ (MM)、高速メモリと主メモリ間の転送量を表し、キャッシュコンピュータの場合は、順に、レジスタ (Reg)、キャッシュメモリ (CM)、主メモリ (MM)、キャッシュメモリと主メモリ間の転送容量を表す。メモリ容量は、ワードである。

次に性能評価で用いるシミュレータについて説明する。

シミュレーション

性能評価で使用するシミュレータは、2 種類存在する。1 つは、UPCHMS 用であり、もう 1 つは、キャッシュコンピュータ用である。

	VHM Reg	HM CM	MM	BLK size
UPCHMS	32	512	131072	1
CMS	32	512	131072	4

図 6: シミュレーションで用いたメモリ容量

UPCHMS は、1 命令 4 ステージ構成で 4 クロックで処理する。キャッシュシミュレータは、UPCHMS 同様 1 命令 4 ステージ構成で、4 クロックで処理する。また、LRU 置換、write-through、フルアソシエイティブ方式を採用した。

評価で用いるプログラムは、以下の 2 つである。

- **add-sum**:主メモリ上にあるデータの単純な演算を行う。(逐次にデータを参照する)
- **bubble sort(bs)**：200 個のデータのソートを行う。(局所性のあるデータ参照をする)

5.2 評価の結果と考察

図 7は UPCHMS によるスピードアップを示すグラフである。縦軸は、キャッシュコンピュータと UPCHMS の実行時間の比を、横軸は、評価プログラムを示す。キャッシュコンピュータに比べて、add-sum では約 1.4 倍、bubble sort(以下 bs) では約 1.8 倍スピードアップされていることがわかる。

図 8はプログラムの実行中どの位のデータ待ちが起こるかを示す図である。縦軸は、データ待ち時間/実行時間を、横軸は、それぞれ、UPCHMS、キャッシュコンピュータを示す。また、グラフの色により評価プログラムを区別する。UPCHMS では add-sum、bs に対してデータ待ちはそれぞれ約 0.6 %、23.7 %であるのに対して、キャッシュメモ

リシステムでは 27 %、35.6 %になっている。UPCHMS のデータ待ちがキャッシュメモリシステムのキャッシュペナルティに対してどの程度減少するかを示すのが図 9である。この図から UPCHMS ではデータ待ちは，add-sum の場合には約 1/50 に、bs の場合には 1/3 になることがわかる。これらの２つの図 8、9は、UPCHMS が他のプログラムにおいても、データ待ちを十分に縮小する可能性があることを示している。

図 7～図??では主メモリのアクセス時間が、キャッシュメモリシステムの場合は２．２５クロック／ワード、UPCHMS の場合は３クロック／ワードと固定されていた。図 10は主メモリのアクセス時間を変化させた場合の評価プログラム add-sum の実行時間の変化を表す。縦軸は、実行時間 (クロック) を、横軸は UPCHMS の主メモリへのアクセス時間を示している。キャッシュメモリシステムの場合には主メモリのアクセス時間の増加に比例して実行時間も増加するが、UPCHMS ではアクセス時間１５クロックまでほぼ一定でその後増加に転じる。これは UPCHMS ではデータを高速メモリに移動してもそれを使用する時までには時間的余裕があることが多く、この余裕がある間主メモリアクセス時間の増加によるデータ転送の遅れが隠蔽されるからであると考えられる。１５クロックがこの余裕を使い果たす限界である。本評価プログラムでは特に転送プログラムの最適化をしてないが、この余裕はプログラムの振る舞いを知るプログラマが転送プログラム最適化することによって最大になる。この余裕の時間は、評価プログラムのデータ参照に局所性がある時、すなわち、データの再利用性が高い時増加する。しかし、バブルソートのように、必要なデータの決定とそのデータを使用するまでの時間間隔が短いような評価プログラムでは、データ待ち時間が増加する。これは、処理プログラムとメモリ操作プログラムに依存度が高くなり、これらのプログラムを並列に処理する時間が短くなるためである。

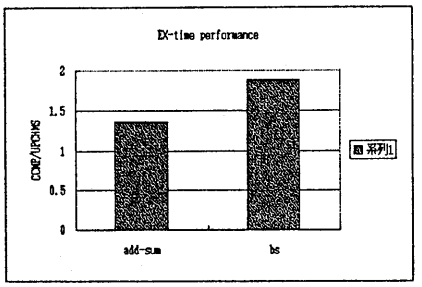

図 7: UPCHMS の実行時間比の結果

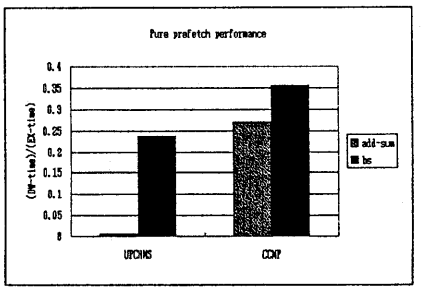

図 8: UPCHMS のデータ待ち時間比の結果

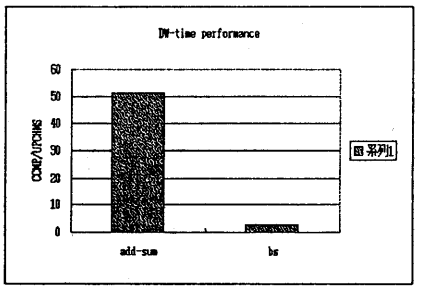

図 9: UPCHMS とキャッシュコンピュータとのデータ待ち時間の比較

6 さいごに

本論文は、従来のキャッシュメモリシステムの問題点を明らかにし、その問題を解決する方法として、ユーザプログラム制御階層メモリシステム：UPCHMS の検討を行った。

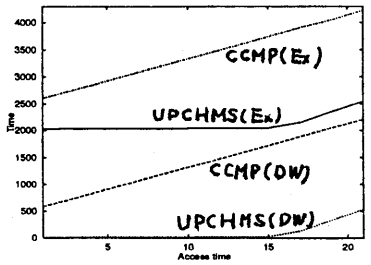

図 10: 主メモリのアクセス速度変化による UPCHMS の実行時間

最初に UPCHMS の簡単な動作を説明し、ソフトウェアを視点とした時の動作とプログラム表現を示した。UPCHMS のもつ問題点とその解決法について示した後、実際にシミュレーションをすることで、UPCHMS の評価を行った。この結果、UPCHMS と同容量のキャッシュメモリを持つコンピュータシステム (キャッシュコンピュータ) に比べ最大約 1.8 倍高速に処理を行うことができ、データ待ち時間をほぼ 0 にすることができた。

評価の結果としては、UPCHMS は、全ての場合に、キャッシュコンピュータ以上に高速に処理を行うことができることを示し、また、ソートなどのデータ転送がプログラムの大半を占めるようなの特別な場合を除いて、プロセッサユニット PU を待たせる時間は、非常に短いことを示した。

本論文では、命令メモリを理想的なものとした。というのは命令メモリは線形的にアクセスされることが多くこのように考えても現段階では問題が少ないと考えたからである。命令メモリに本方式を導入することも考えられており、その成果の一部はすでに発表されている [2] し、その改良型の研究も精力的に行われている。

参考文献

[1] J.Hennessy and D.Patterson, "COMPUTER ARCHITECTURE:A QUANTITATIVE APPROACH" MORGUN KAUFMAN PUBLISERS INC,1990

[2] 佐藤 正樹, "並列処理によるキャッシュ操作の明示化", 並列シンポジウム JSPP'90,pp 1-7,June (1990)

[3] 高木 浩光, 河村 忠明, 有田 隆也, 曽和 将容, "問題が持つ先行関係のみを保証する高速な静的実行順序制御機構の構成法", 並列処理シンポジウム JSPP'90 論文集 (1990)

[4] Fredrik Dahlgren,Michel Dubois,and Per Strenstrom, 'Fixed and Adaptive Sequential Prefetching in Shared Memory Multiprocessors' International Conference on Parallel Processing,(1993)

[5] Alan L.Cox,Robert J.Fowler 'Adaptive Cache Coherency for Detecting Migratory Shared Data' IEEE,Advances in Parallel and Distributed Systems pp98 -pp108,(1993)

[6] Kieran Harty and David R.Cheriton 'Application-Controlled Physical Memory using External Page-Cache Management' ACM,Architectural Support for Programming Languages and Operating Systems, p187 - 197, October 12-15,(1992)

[7] Callahan, D.,Kennedy, K. and Porterfield,A."Software Prefetching",Proc.the 4th International Conference on Architectural Support for Programming Languages and Operating Systems,pp.40-52(Apr.1991)