計算機アーキテクチャ 125－14
（1997. 8. 21）

# キャッシュコヒーレンス制御が不要な並列計算機におけるプログラミング

中済 光昭, 岡本 秀輔, 曽和 将容

電気通信大学 大学院 情報システム学研究科

## 概要

並列計算機でのデータアクセス遅延を解決するため, キャッシュメモリを適用することがある. しかしキャッシュメモリがブロック単位で管理され, プログラムから制御不可能なものであるため, 並列計算機の性能を最大限に引き出せなかった. この問題を解決するため, 我々はプログラム制御キャッシュレベルメモリを提案してきた. 本文では, プログラム制御キャッシュレベルメモリにより, キャッシュコヒーレンス制御が不要な並列計算機を構築するためのプログラミング手法について述べる.

# *A Programming Method for Cache Cohenence Protocol Less Parallel Computer*

Mitsuaki NAKASUMI,Shusuke OKAMOTO and Masahiro SOWA

Graduate School of Information Systems
University of Electro-Communications
E-mail:{nakasumi,okam,sowa}@sowa.is.uec.ac.jp

**Abstract**

Applying cache to multiprocessor is one of the solution to improve latency with remote memory access.

But the line size of the cache is the unit of storage reference and the cache cannot control by user program, it cannot get maximum performance.

To solve this problem, we have already proposed Program Controlled Cache Memory on Parallel Computer. This memory system can migrate data between high speed memory as fast as cache memory and NUMA-type shared memory by the program for data migration. This memory system is composed by word-addressable high speed memory (Cache Level Memory) and hardware mechanism which executes instructions to migrate variable sized data.

In this paper, we describe a programming method for cache coherence protocol less parallel computer.

# 1 はじめに

分散共有メモリ型マルチプロセッサにおいて，リモートメモリアクセスレイテンシを削減するため，キャッシュメモリが用いられている．

キャッシュメモリを用いることにより，リモートメモリアクセスのオーバーヘッドの低減，プリフェッチ効果がみられ，性能が向上する．

しかし，キャッシュメモリを用いた並列計算機では，キャッシュメモリがブロック単位のメモリ管理を行なうため，他のプロセッサが使うデータまで一緒に転送され，キャッシュメモリに置かれることがある．このため，false sharing[10] を生じたり，キャッシュコヒーレンスプロトコルによりスラッシングが発生することがあるなどの問題があった．

我々は，キャッシュメモリが有効に働かないことによる性能低下を避けるためにプログラム制御キャッシュレベルメモリ [13][15][16] を持つ並列計算機を提案してきた．プログラム制御キャッシュレベルメモリは，ワード単位にアクセスできる高速なメモリ (キャッシュレベルメモリ) とそのメモリへのデータ転送をプログラムにより，可変長単位で行うハードウエア機構からなり，ユーザーによって記述されるデータ転送プログラムにより，プロセッサが必要とするデータをキャッシュレベルメモリに読み書きするものである．

この方式では，キャッシュレベルメモリへの転送をブロック単位ではなく，プログラムにより必要量だけ任意の位置に行えるので，悪影響を及ぼす false sharing を避けることができる．

またプログラムによってデータの位置を完全に制御できるのでキャッシュコヒーレンスプロトコルが不要である．という利点を持つ．

本文では，キャッシュメモリの問題を簡単に説明した後，プログラム制御キャッシュレベルメモリの構成とこれを制御するプログラムについて述べ，次にキャッシュコヒーレンスプロトコルによる問題を解決する手法を示す．

# 2 キャッシュメモリの問題

ここでは，キャッシュメモリを用いた並列計算機における false sharing とスラッシング問題について説明する．

## 2.1 false sharing

キャッシュメモリでブロック単位の管理が行なわれることによる問題として更新データの false sharing がある．false sharing 問題を図 1のデータアクセス例により説明する．

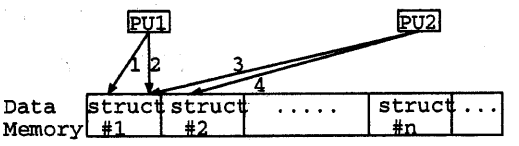


図 1: データアクセス例

図 1中,PU1,PU2 はプロセッサを,Data Memory はデータが割り当てられるメモリ領域を示す．struct は，プログラムで定義される構造体などのデータ構造である．

通常，データは構造体などのデータ構造を用いてメモリ内の連続した領域に確保される．この時，複数の PU で各々が使うデータを保存する 2 つのメモリ領域が，連続して共有メモリに置かれると，それらが同一キャッシュブロック中に割り当てられることがある．

2 つの PU が同じデータ構造を参照する，すなわち，構造体 struct # 1 に対して，PU1 が矢印 1 でポイントされるデータを読み込み，その値を処理して書き戻し，PU2 が矢印 3 でポイントされるデータを読み込み，その値を処理して書き戻す場合，各々の PU のキャッシュに同じデータのコピーが置かれ，false sharing が起こる (図 2(1))．

また,2 つの PU が連接した領域におかれるデータ構造を参照する，すなわち，PU1 が矢印 2 でポイントされるデータを読み込み，その値を処理して書き戻し，PU2 が矢印 4 でポイントされるデータを読み込み，その値を処理して書き戻す場合，キャッシュブロックサイズが大きいと，データ構造の境界を越えて，隣接するデータ構造の一部も同じキャッシュブロックに格納されてしまうので，各々の PU のキャッシュに同じデータのコピーが置かれ，false sharing が起こる (図 2(2))．

リモートメモリアクセスレイテンシが大きい並列計算機では，転送オーバーヘッドの削減やプリフェッチ効果を得るため，ブロックサイズを大きくするが，それにより上記の false sharing が発生しやすくなり，

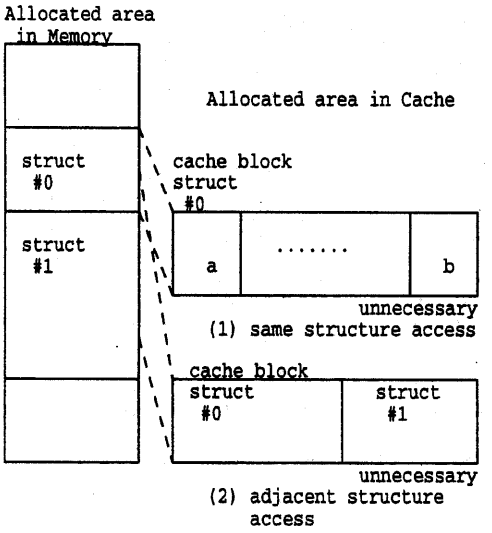


図 2: false sharing

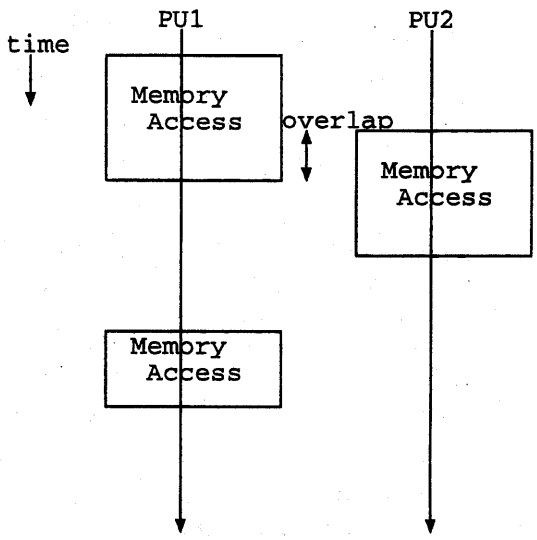


図 3: スラッシング

キャッシュコヒーレンスプロトコルのための通信が増加する.

## 2.2 スラッシング

図 3は,false sharing が発生する場合, 各 PU での命令の実行状況により発生するスラッシングを示す.PU1,PU2 は命令実行の流れである. Memory Access は ロード・ストア命令の実行, 他の部分は演算命令などの実行を示す. 図 3上部では,PU1 と PU2 の Memory Access に overlap が見られる. この部分のロード・ストア命令の対象データが, キャッシュメモリにおいて false sharing を起こしている時, スラッシングがおこる. この現象は, メモリアクセス以外の演算などの命令の実行時間が, メモリアクセスに比べ相対的に少ない場合起きやすい. スラッシングが起こると無駄なデータ転送とそれに伴うデータ待ちが起こるので, 性能が著しく低下する.

# 3 キャッシュコヒーレンス制御が不要な並列計算機

キャッシュコヒーレンス制御が不要な並列計算機の構成は, 図 4の通り要素プロセッサ上に他要素プロセッサからリードライト可能な NUMA 型ローカルメモリを持ち, 各要素プロセッサ上のデータ転送ユニットにより相互接続網を介してデータ転送を行なうものである. システムは, 以下の機能ユニットから構成される. $UC_i$[1]は要素コンピュータである. NW は要素コンピュータを相互接続するネットワークである. $PU_i$は $Reg_i$と $CLM_i$を使って演算を行なうユニットであり,DLX [5] と同等である. $Reg_i$は $PU_i$の汎用レジスタファイルで, $PUIM_i$は $PU_i$の命令メモリである. $DUDM_i$は NUMA 型データメモリの一部である. $DUIM_i$は $DU_i$の命令メモリである. $AR_i$は $DU_i$のアドレス計算用レジスタファイルである. $CLM_i$はキャッシュレベルメモリでありキャッシュと同じように高速アクセスが可能なメモリであるが, 通常のメモリと同様のアドレスを持ち, ワード (4バイト) 単位でデータを読み書きできる. $DU_i$は $CLM_i$と全ての DUDM 間のデータ移動を行うデータ転送ユニットであり, DLX の命令セットのうちアドレス計算に必要な命令および新たに定義するデータ転送命令を実行するものである.$DU_i$は, $DUIM_i$に格納されたデータ転送プログラムを実行する. PU-DU 間の同期は,PN コンピュータで用いられた同期のための信号 (トークン)[7] を PU-DU 間で送受する機構により実現する. この信号の送受は PU,DU で実行される命令の同期オプションにより行なわれる. DU-DU 間の同期

[1]添字 i は i 番目の構成要素を表す.

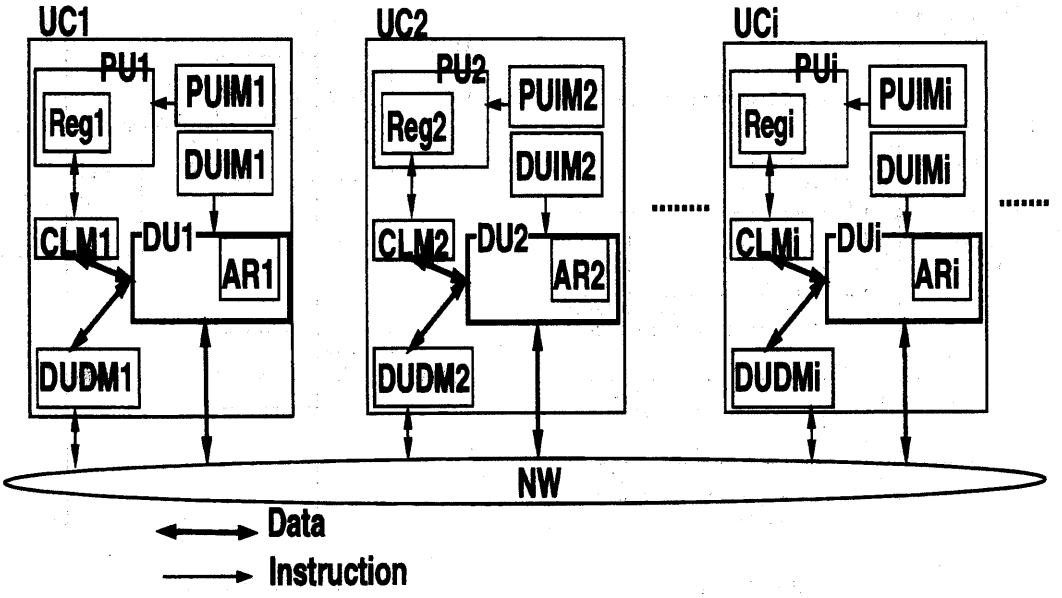


図 4: キャッシュコヒーレンス制御が不要な並列計算機の構成

は,I-structure[2] に基づく同期ビットの set/reset で実現する. DUDM にワード毎につけられる同期ビットを,DU のデータ転送命令の同期オプションにより set/reset することによって同期を取る.

## 4 プログラム制御キャッシュレベルメモリを持つ並列計算機のプログラム

プログラム制御キャッシュレベルメモリを持つ並列計算機のプログラムを以下に示す.

図 5は,DUDM に順に格納されているデータ f1,f2,f3,f4 から $(f1 + f2) * (f3 + f4)$ を計算する処理の流れである. ThPU1,ThPU2,ThDU1,ThDU2 は, それぞれ PU1,PU2,DU1,DU2 の命令実行の流れであり, 各命令間に付けられたアークは, 命令の依存関係をあらわす.

ra1-ra4 は,$PU_1$の汎用レジスタ,rb1-rb3 は,$PU_2$の汎用レジスタ, aa1-aa5 は,$DU_1$のアドレス計算用レジスタ, ab1-ab3 は,$DU_2$のアドレス計算用レジスタを示す. p11-p16 は ThPU1,d11-d13 は ThDU1,p21-p24 は ThPU2,d21-d22 は ThDU2 の命令アドレスを示す. また,(Reg) は Reg で示されたレジスタの内容を示す.

ここでは, 簡単のため必要なレジスタにアドレスがセットされるものとする. このプログラムにおいて,ThPU1 の p11 と p12 の命令が必要とするデータ f1,f2 は, d11 の命令により,f1 が置かれる $DUDM_1$の (aa1) 番地から 2 ワード分,$CLM_1$の (aa2) 番地から 2 ワード分の領域へ転送され,p11 と p12 の命令によりそれぞれ $PU_1$ の ra2 と ra3 に転送される.

また,ThPU2 で計算された f3+f4 の結果は p24 の命令によって,f3 が置かれていた $CLM_2$の (rb1) 番地に上書きされる.d22 の命令は DUDM の (ab2) 番地にデータを転送する.d12 の命令は d22 の命令と同期を取り $CLM_1$にデータを転送し.p14 の命令により $PU_1$の ra2 に転送する.

## 5 キャッシュコヒーレンスプロトコルの問題を回避する手法

プログラム制御キャッシュレベルメモリを持つ並列計算機では, 図 5に示すプログラムにより, 各メモリの整合性が保証されるため, キャッシュコヒーレン

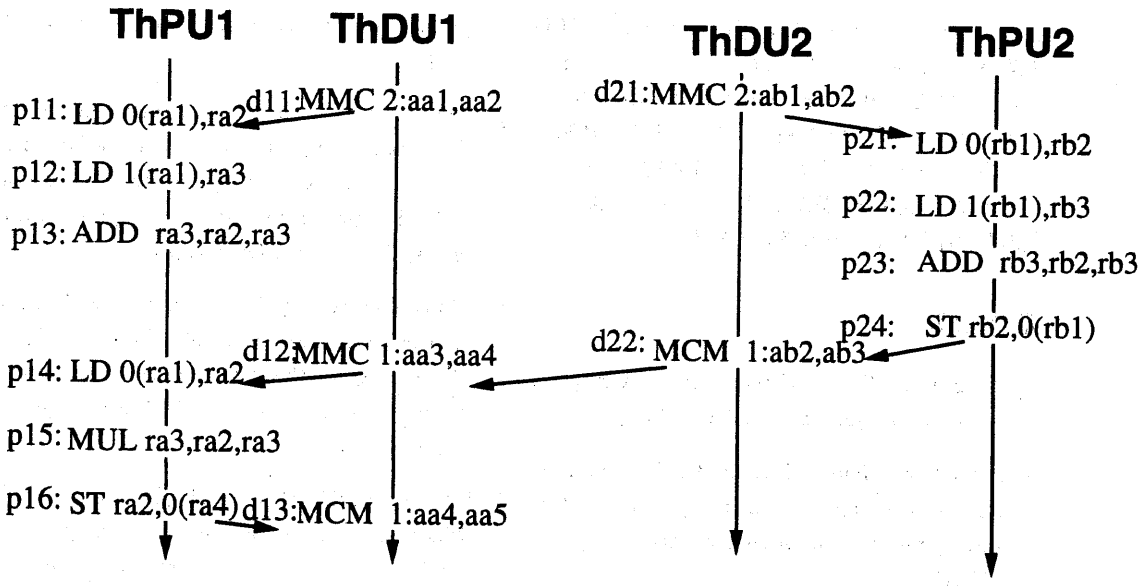


図 5: プログラム例

スプロトコルが不要であり, キャッシュメモリにおける false sharing やスラッシング問題を回避できる.

すなわち, キャッシュレベルメモリにワード単位の可変長サイズのデータを読み書きできるので, 固定サイズのキャッシュブロックでは, 同じブロックに置きたくないデータが同じブロックに入ることがあるが, プログラム制御キャッシュレベルメモリでは, プログラムによってデータ構造ごとにメモリを管理できるため,false sharing がない.

また, スラッシングについても, 同期機構によってデータの移動のタイミングを制御できるため, データ転送命令と演算命令のタイミングなどの不確定要素に左右されず, データの移動が行なわれる. よってスラッシングが起きない.

## 6 考察

複数の PU からのメモリアクセスが時間的にオーバーラップすることが予想されるメモリ領域に対して, 一般のキャッシュメモリでは, 注意を払わなければならない.

例えばスピンロックで参照される変数は, 特定のタイミングに集中してアクセスされるため,false sharing しないようメモリ領域の割り当てを工夫しなければならない. そのために 1 つのキャッシュブロックに 1 つのロック変数を割り当てることになり, メモリの無駄使いとなる. このためキャッシュ容量ミスが起こる.

メモリアクセスレイテンシが大きい場合, 性能向上のためキャッシュブロックサイズを大きくするが, これにより, 1 回のメモリアクセス実行時間が長くなる. このため, 演算などの命令の実行時間が相対的に小さくなり,false sharing が発生するキャッシュブロックに対するメモリアクセス期間の重複が起き, スラッシングの発生が多くなる.

これに対し, プログラム制御キャッシュレベルメモリでは, ワード単位でアクセス可能なキャッシュレベルメモリにより false sharing をなくすとともに, キャッシュレベルメモリと分散共有メモリ間のデータ転送をユーザープログラムにより制御することで, スラッシングをなくすことができる.

## 7 おわりに

本文では, プログラム制御キャッシュレベルメモリの構成と, これを制御するプログラムについて述べ,

次にキャッシュコヒーレンスプロトコルによる問題を解決する手法を示した. この方式では, キャッシュブロックに依らないメモリ管理を行なうので, キャッシュメモリの無駄使いを避け, コヒーレンス維持のためのデータ転送を削減することが可能である.

今後は, 代表的な並列アプリケーションに関し, 定量的な性能評価を行なう予定である.

## 参考文献


[1] A.J.Smith: Cache Memories,ACM Computer survey,vol.14,no.3, pp.473-530, 1982

[2] Arvind and Iannucci,R.A.: Critique of Multiprocessing von Neumann Style, Proc. 10th Int. Symp. on Computer Architecture,pp.426-436 ,1983.

[3] James R. Goodman : Using Cache Memory to Reduce Processor-Memory Traffic, Proc. 10th Ann. Int. Symp. on Comp. Arch., pp.124–131, June 1983

[4] S. Eggers : Simulation Analysis of Data Sharing in Shared-Memory Multiprocessors, Report No. UCB/CSD 89/501, Ph. D. Thesis, Computer Science Division, University of California, Berkeley, April 1989

[5] John L. Hennessy, David A. Patterson: COMPUTER ARCHITECTURE:A QUANTITATIVE APPROACH,Morgan Kaufmann Publishers Inc. ,1990

[6] Jouppi,N.P.: Improving Direct-mapped Cache performance by the Addition of a small Fully-Associative Cache and Prefetch Buffers, Proc. of 17th Int'l Symp. on Computer Architecture,pp.364-373,1990

[7] 高木 , 河村, 曽和: 問題の持つ先行関係だけを保証する高速な静的実行順序制御機構, JSPP,Vol.1,pp. 57-64, 1990.

[8] A.C.Klaiber and H.M.Levy: An architecture for software-controlled data prefetching, In proceedings of the 18th Annual International Symposium on Computer Architecture ,pp.43-53, 1991

[9] John B. Carter,John K. Bennett, and Willy Zwanepoel.: Implementation and Performance of Munin, In proceedings of the Thirthhenth ACM Symposium on Operating System Principles, pp. 152–164, October 1991

[10] Michel Dubois: "The Detection and Elimination of Useless Misses in Multiprocessors", In Proceedings of the 20th Annual International Symposium on Computer Architecture,pp.88-97,May 1993

[11] Jeffrey Kuskin et al.: "The Stanford FLASH Multiprocessor", In proceedings of the 21st Annual International Symposium on Computer Architecture, pp. 302-313, April 1994

[12] L.I.kontothanassis,M.L.Scott: Software cache coherence for large scale multiprocessors., In Proceedings of first IEEE Symposium on High-Performance Computer Architecture, pp. 286-295 ,1995

[13] 中済 光昭, 岡本 秀輔, 曽和 将容: 物理分散共有メモリ型並列コンピュータにおけるプログラム制御キャッシュメモリ, 情処学会研報, 96-ARC-119,pp. 55-60, 1996

[14] 高橋 雅史, 大庭 信之, 小林 広明, 中村 維男: データの更新をバイト単位で管理するキャッシュメモリ, 信学論 (D-I), vol.J79-D-I,no.7, pp. 425-436 ,1996

[15] 中済 光昭, 岡本 秀輔, 曽和 将容: プログラム制御キャッシュメモリの性能評価, 情処学会研報, 96-ARC-123,pp. 31-36 ,1997

[16] M.Nakasumi, S.Okamoto and M.Sowa: Program Controlled Cache Memory on Parallel Computer, In Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications, vol.III,pp.1423-1428,June 1997