計算機アーキテクチャ 123－6
（1997. 3. 7）

プログラム制御キャッシュメモリの性能評価

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

並列コンピュータのスケーラビリティを阻害する問題の1つに, リモートメモリアクセスのレイテンシがある. この問題を解決するため, キャッシュメモリにプロセッサの使用するデータをプリフェッチする方法が提案されてきた.

しかし, キャッシュメモリはブロック単位で管理され, プログラムから制御不可能であるため, 不要なデータがキャッシュメモリに置かれ, 必要なデータがキャッシュから追い出されることがあり, またキャッシュ整合プロトコルのための無駄なデータ転送が多くなり, 並列コンピュータの性能を最大限に引き出せなかった.

この問題に対し, 我々は並列コンピュータで用いられる従来のキャッシュメモリを置き換える新しい高速メモリを提案してきた. これは, キャッシュメモリと同等のスピードを持ち, ワード単位でアクセス可能なメモリ (キャッシュレベルメモリ) とこのメモリを制御する命令セットを実行するハードウエア機構からなる. 本文では, 我々が提案するシステムと通常のキャッシュを持つNUMA型並列コンピュータと比較して, 性能を評価した. 評価のために, 我々が提案するシステムのシミュレータを作成するとともに, 通常のキャッシュを持つNUMA型並列コンピュータのシミュレータを作成した. 評価は,Libermore Loop ベンチマークプログラムを用いて行なった. その結果, 各ベンチマークプログラムにおいて, 本方式は既存のキャッシュに比べ高い性能を示した.

A Performance Evaluation of Program Controlled Cache Memory on 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

A latency with remote memory access is one of the problem on the design of Parallel Computer. To solve this problem, Data prefecthing to data cache has already proposed. But Prefetching cannot handle inefficient cache usage and invalid traffic on cache coherence protocol.

To improve these problems, 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 evaluate performances between Our Proposed Memory System and Parallel Computer with Conventional Cache. We use Livermore loop benchmarks for these evaluation. As a result, it shows higher performance than conventional cache.

1 はじめに

並列コンピュータの性能を最大限に発揮するためには，それを構成するプロセッサを待たせないようにデータを供給することが必要である．これに対し，各プロセッサに接続されるキャッシュメモリにデータをプリフェッチする方法が提案されている [1]．しかし，いくつかのアプリケーションにおいてデータキャッシュへのプリフェッチが有効に働かないことがある．キャッシュが有効に働かない要因は，ライン単位の転送を行なうので不要なデータまで一緒に転送され，キャッシュに置かれることと，キャッシュ内のどこにデータをおくか，およびどのラインを追い出すかが固定アルゴリズムにより決定することである．

我々は，キャッシュが有効に働かないことによる性能低下を避けるために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を提案してきた [5]．本方式では，キャッシュレベルメモリへの転送をキャッシュメモリのようなライン転送ではなく，プログラムにより任意の位置に必要量だけ行うので，無駄な転送がなく，データの追い出しがないという利点を持つ．我々が提案する方式の有効性を示すため，通常のキャッシュを持つ NUMA 型並列コンピュータとの比較を行ない，性能を評価した．評価は，我々が提案するシステムのシミュレータと，通常のキャッシュを持つ NUMA 型並列コンピュータのシミュレータを用いた．これらのシミュレータで Libermore Loop ベンチマークプログラムを実行して 1) 実行時間 2)PU 待ち時間 3) ネットワーク待ち時間に関して評価を行なった．

本文では，本方式を適用する並列コンピュータアーキテクチャ，動作，特徴を述べ，次にシミュレーション方法とその結果を示す．最後に評価結果について考察を行なう．

2 対象となる並列コンピュータアーキテクチャ

本方式を適用する並列コンピュータアーキテクチャは，図 1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 [2] 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．tc は同期のための信号 (トークン) を PU-DU 間で送受する機構であり PU-DU 間の命令実行の同期を取る．s は同期ビットであり DU-DU 間のデータ参照の同期を取る．DUDM にワード毎につけられる同期ビット s への DU のデータ転送命令による set/reset 操作によってデータ参照の同期を取る．

3 プログラムの動作

図 2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-ra3 は,PU_1の汎用レジスタ,rb1-rb3 は,PU_2の汎用レジスタ，aa1,aa2 は,DU_1のアドレス計算用レジスタ，ab1,ab2 は,DU_2のアドレス計算用レジスタを示す．p11-p16 は ThPU1,d11-d14 は ThDU1,p21-p22 は ThPU2,d21-d24 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

[1]添字 i は i 番目の構成要素を表す．

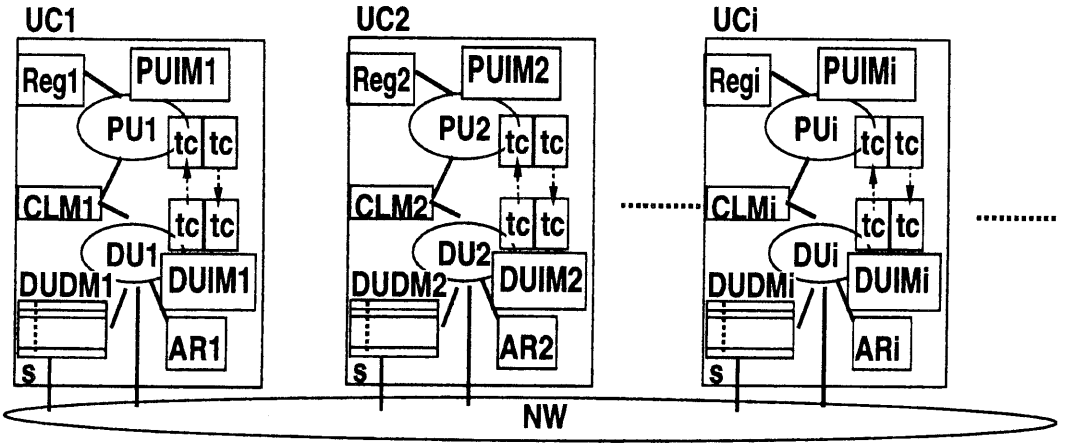

図 1: 本方式を適用する並列コンピュータの構成

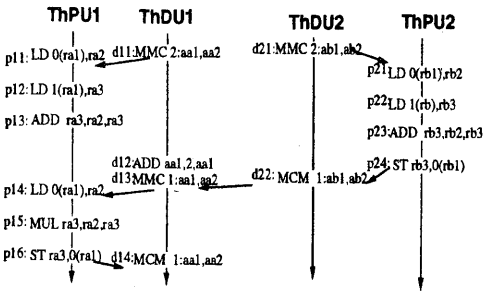

図 2: 動作例

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.d13 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.

4 シミュレーションによる性能評価

本方式の性能評価のため, クロックレベルでの各部の動作を再現する実行駆動型ソフトウエアシミュレータを作成し, ベンチマークプログラムを実行した. 以下では本シミュレータを用いた性能評価結果について述べる. 比較のため, 従来のキャッシュによる NUMA 型マルチプロセッサのソフトウエアシミュレータを作成し, ベンチマークプログラムを実行した.

4.1 シミュレーションパラメータ

本研究の評価のため, 今回提案する並列コンピュータと従来のデータ転送方式に基づく並列コンピュータのシミュレータを作成した. 提案する並列コンピュータのシミュレーションパラメータは, 次の通りである. システムは,1,2,4,8 台の UC をバス型ネットワークで接続し構成する. UC の構成は,DUDM を 64k バイト,CLM を 512 ワード, PU の処理単位は 1 ワード 32 ビットであり, レジスタは 32 個とする. 命令メモリ (DUIM,PUIM) は, 理想のものとし, 遅延なくアクセスできるものとする. PU の命令実行クロックは, 演算命令を 1 クロック,CLM のロードで 2 クロックとした. DU の命令実行クロックは, アドレス演算命令を 1 クロック, データ転送命令を 6 クロック, これに加え 1 ワードごとに 1 クロックを加算するものとした.

従来型並列コンピュータのシミュレーションを, 以下の条件で行なった. シミュレータは,PU, キャッシュコントローラ, 通常のキャッシュ, NUMA 型共有メモリを構成するメモリモジュールからなる要素プロセッサをネットワークで接続した構成とする. システムは,1,2,4,8 台の要素プロセッサをバス型ネットワークで接続し構成する. キャッシュは,1 ライン 32 バイト,64 ラインのダイレクトマップとし, ライトワンス方式の整合プロトコルを用いる. PU の命令実行クロックは, 演算命令を 1 クロック, キャッシュにデータがある場合のロードで 2 クロック, データがない場合のロードでキャッシュチェックの 2 クロックに加え, 6 クロック, これに加え 1 ワードごとに 1 クロックを加算するものとした.

命令メモリは理想的なものとし分岐命令でのフェッチミスによるストールがないものとする.

評価は,Livermore Loop Kernel # 3(LM3),# 4(LM4)[2]の主要部から構成するプログラムにより行なった.

LM3 は,2 つの配列をストライド 1 でアクセスするため, 通常のキャッシュが最も有効に使われる. よってデータプリフェッチ効果が最大となる.

LM4 は, 1 つの配列をストライド 1 でアクセスするが, もう 1 つの配列はストライド 5 でアクセスするため, 通常のキャッシュでは使われないデータがキャッシュに存在する. データプリフェッチ効果はストライド 5 でアクセスする配列については発揮しにくい.

4.2 結果

図 3から図 6がシミュレーション結果である. これらの図中,CACHE は通常のキャッシュによるシステム,DU は本方式を適用したシステムのデータを示す. 図 3,4は, 各々LM3,LM4 についての本方式および通常のキャッシュによるシステムでの実行時間比である. データは PU=1 の時のキャッシュでの実行時間を 1 とした実行時間比である. 図 5,6は, 各々LM3,LM4 における本方式および通常のキャッシュによるシステムについての PU 待ち時間のグラフである. 棒グラフは各 PU のデータ待ち時間[3]のうち最も長いもの, 折れ線グラフはネットワークの使用権を取得するまでの待ち時間の総和を示す.

[2]以降 () 内の略号を用いる

[3]ネットワークの使用権を取得するまでの待ち時間と通信時間の総和

5 考察

図 3の通り, 本方式は PU 数の増加に対しキャッシュより良いか, 同等の性能を示している. LM3 についてみると,PU=1 の時, 本方式の性能が他の台数の場合に比べ良いが PU が増えるごとに本方式との性能差が縮まっている. これは図 5で PU=1 での PU 待ち時間の差が大きいことによる. PU=1 でのキャッシュにおける PU 待ち時間はキャッシュミスによる. キャッシュミス数は PU 数の増加に従って 131,33,18,9 と推移する.PU=1 の時ミス数が他のミス数に比べ大きく, これが性能低下の要因となっている. このミス数の推移は, 各 PU で処理されるデータ数が減少するためミス回数も同様に減ったためである. これに比べ本方式の PU 待ち時間が少ないのは, 本方式では PU が参照するデータは,DU によって CLM に転送されるので各 PU で処理されるデータ数に関わらずミスがない. ただし PU との処理との同期による待ちがあるだけであるが, DU による並行転送がローカルメモリと高速メモリ間の転送のため高速に行なわれ CLM へ PU がアクセスする時キャッシュミスに相当する状態が少なくなるため待ちが少なくなっている. PU 待ち時間の中で図 5に示すネットワーク待ち時間が大きな割合を占める. 図の通り PU=2 でネットワーク待ち時間の差が大きく PU=4 で低下し,PU=8 まで 同じ程度の待ち時間の増加となる. これはネットワーク待ち時間がネットワークに接続される通信ユニットの数と通信トランザクション量により決まるが PU=2 の時, キャッシュでは PU1 の分散共有メモリへのアクセスが短い間隔で頻繁に発生することによりネットワーク待ちが多くなる. 本方式では, メモリのアクセスを同期を取って行なえるので, 同じ状況でもネットワーク競合が少ない. PU が増えるに従って ネットワークトランザクションが発生するタイミングがずれるため, ネットワーク待ちが本方式と同じ程度で推移する.

図 4の通り, LM4 では,PU 数が増加しても キャッシュと本方式の差は大きく狭まらない. キャッシュミス数は PU 数の増加に従って 625,78,42,26 と推移する. ストライド 5 の配列では 4 ワードのキャッシュラインを持つキャッシュにおいて 1 ワードしか使用しないため, このようなミス数となる. これに比べ本方式の PU 待ち時間が少ないのは, LM3 と同様の DU によるデータ転送の並列実行の他, 必要なデータのみ

を転送することから通信トランザクション数を削減でき,図6の通りネットワーク待ち時間をキャッシュに比べ大きく削減できるからである. すなわちキャッシュのPU待ち時間はPU数が増加してもキャッシュミス数の減少度合いが減るが本方式のPU待ち時間は低い値で推移することから性能差が縮まらない.

6 おわりに

本文では, バイト単位でアクセスできるキャッシュレベルメモリとそれを制御するプログラムを実行するハードウエアにより可変長のデータをキャッシュレベルメモリの任意の位置に転送する方式について述べ, その評価を行った.

この方式では, 全てのメモリをプログラムで制御できるため, 不必要なデータ転送をしない, 不必要なデータを早期に追い出すことから, 少量のキャッシュレベルメモリを有効に利用することが可能となる. そしてシミュレーションの結果より, メモリ間のデータ転送すべてをプログラムで可変長単位で行う本方式の有効性が明らかになった.

参考文献

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

[2] J.L.Hennessy,D.A.Patterson: Computer architecture, Morgan Kaufmann Publishers Inc.(1990)

[3] 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)

[4] W.Y.Chen, S.A.Mahlke,P.P.Chang and W.-M.Hwe : Data access microarchitectures for superscalar processors with compiler-assisted data prefetching, Proc. of the 24th Int'l Symp.on Microarchitecture(1991).

[5] 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).

[6] 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).

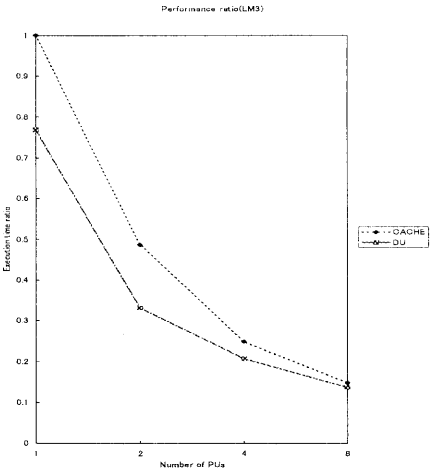

図 3: 実行時間比 (LM3)

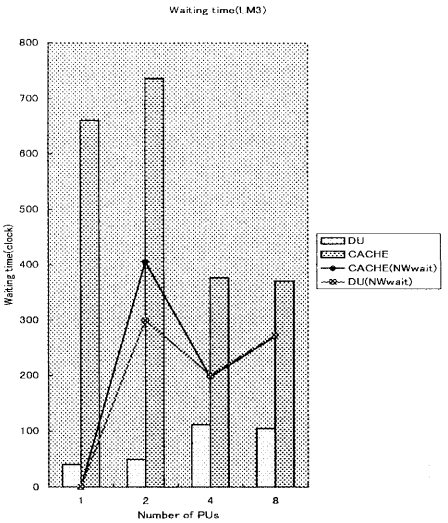

図 5: 待ち時間 (LM3)

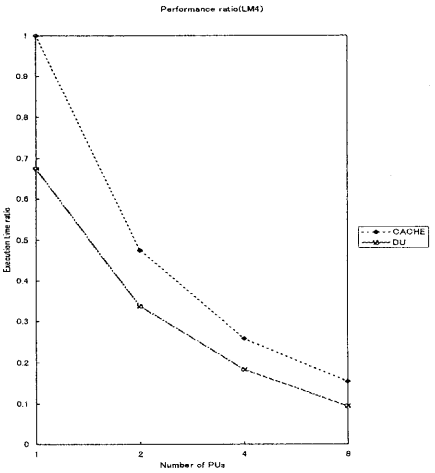

図 4: 実行時間比 (LM4)

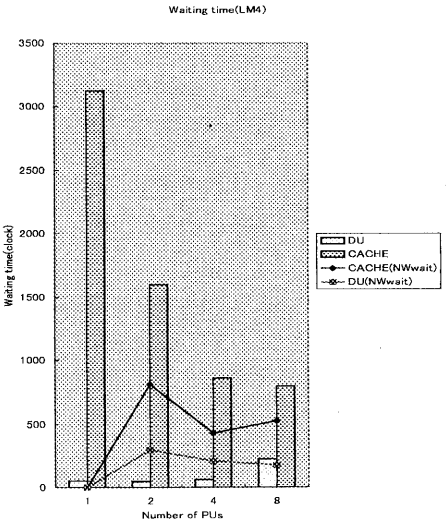

図 6: 待ち時間 (LM4)