計算機アーキテクチャ 119－10
（1996．8．27）

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

中 済 光 昭† 岡 本 秀 輔† 曽 和 将 容†

本文では，分散共有メモリ型並列コンピュータで，メモリからキャッシュメモリへのデータ移動を，プログラムによって行うプログラム制御キャッシュメモリ方式を提案する．データ移動用のプログラムは，演算処理用とは異なる転送専用のプロセッサにより並列に実行される．データ移動は命令により，バイトを基本単位として可変長で行なえるので，キャッシュに相当するメモリに，必要最小限のデータを不整合なく格納できる．また，リモートデータをプリフェッチ出来るので，従来のキャッシュメモリで問題となっていた，キャッシュミスやリモートデータ転送のオーバーヘッドを最少にすることができる．

Program Controled Cache Memory on Distributed Shared Memory based Parallel Computer

MITSUAKI NAKASUMI,† SHUSUKE OKAMOTO † and MASAHIRO SOWA†

In this paper,We propose Distributed Shared Memory based Multi-processor system which migrates data between high speed memory(cache like memory) and distributed shared memory by user program.

Data Migration Unit is performed by user program and is able to migrate variable size of data. It makes two better characteristics than conventional cache memory; as follows. 1)Minimum and consistent data is always available in high speed memory (cache like memory). 2) Data Migration Unit can hide cache miss and cache coherent protocol overhead because DU and PU can perform simultaneously.

1. はじめに

並列システムの性能を最大限に引き出すために，キャッシュの適用およびソフトウエアプリフェッチ[1]などデータ遅延を隠蔽する手法が用いられている．しかし,cache pollution や キャッシュ整合プロトコルのオーバーヘッドにより，性能の低下をみる場合がある．

この問題について，データ移動をプロセッサと独立に行うプログラム可能なデータ転送ユニット(Data Migration Unit;DU) を用いるデータ転送方式を提案する．この方式は，各プロセッサのキャッシュ相当のメモリ (Cache Level Memory; 後述) を，各プロセッサのプログラムとは別に存在するデータ転送用プログラムと，データ転送用プログラムを実行するデータ転送ユニットによって制御し，整合の保たれた必要最小限のデータが，各プロセッサのキャッシュ相当のメモリにあるようにする．

すなわち

- 最小限のデータのみを転送する
- アプリケーションに応じた置き換え制御/メモリ整合プロトコル記述を行なう

ことからリモートデータアクセスにおけるオーバーヘッドを最小にする．本論文では，プロセッサが，従

† 電気通信大学
University of Electro Communications
{nakasumi,okam,sowa}@sowa.is.uec.ac.jp

来のキャッシュを持つ分散共有メモリを持つマルチプロセッサシステム上での, リモートメモリアクセスのオーバーヘッドを解決する方式を評価する.

定　　義

以下では, 本文で用いる用語の定義を示す.

- プロセッシングユニット (PU) : 演算処理を行うユニット.
- データ転送ユニット (DU) : DSM 間およびローカルに持つ CLM と MM 間のデータ転送に関する処理を行うユニット.
- 分散共有メモリ (DSM) : 各 DU の持つメモリを任意の DU が線形の論理アドレスを用いてアクセスできるメモリ.DUDM の集合.
- DUDM:DSM を構成し, 各 DU に直接 1 つ接続されるメモリモジュール (データメモリ). 他の DU からは, ネットワーク経由でアクセスされる.
- DUIM:DSM を構成し, 各 DU に直接 1 つ接続されるメモリモジュール (命令メモリ).
- PUIM:各 PU に直接 1 つ接続されるメモリモジュール (命令メモリ).
- キャッシュレベルメモリ (CLM) : キャッシュレベルの高速アクセスが可能なメモリ. キャッシュメモリと異なり, 通常のメモリと同じ線形アドレスを持ち, 通常のメモリと同様のアドレッシングでアクセスする.

2. システムアーキテクチャ

2.1 システム構成

図1は, システム構成を表す. # 1 から # n は, 通常のプロセッサに相当する演算とデータアクセスを行なう機能部を示している. このシステムは, この機能部を複数, ネットワーク (N/W) に接続して構成する. PU は, 演算を行うユニットである. PU は,CLM と Reg. のみにアクセス可能である.CLM は線形な物理アドレスを持っているメモリで, データを格納するために用いる. 命令メモリは,PU に CLM とは別に接続されている. DU は,CLM と DSM 間のデータ転送を行うユニットで,CLM,DSM にアクセス可能である. DU は, リモートに接続されている DUDM には,N/W を介してアクセスをおこなう.

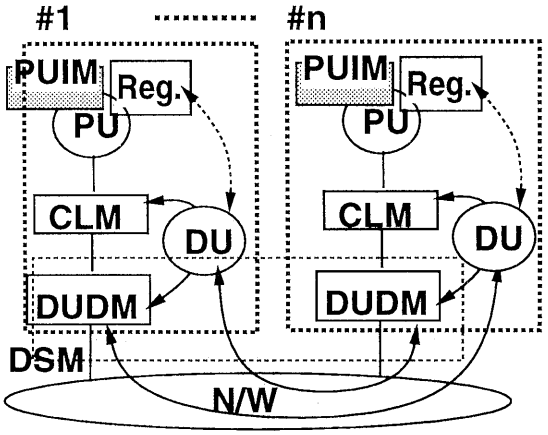

図1　システム構成

2.2 DU の動作

図2に,DU の構成を示す.

PU は, PUIM から命令を取り出し実行する. データ移送命令をフェッチすると,CLM に有効なデータがあれば, データを移動する.DU は,PU とは別にデータ移送命令を前もって実行し, プログラムの要求するデータが, 他の DU の管理化にあるときには, DU は MRU を通して,DSM へデータを要求し, 到着したデータを CLM に入れる.

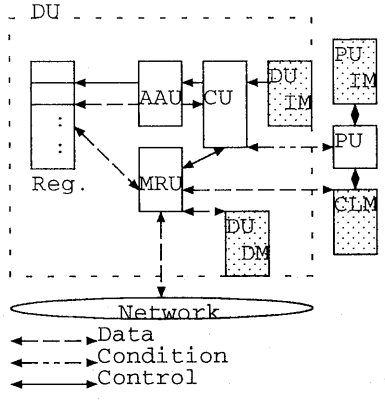

図2　DU の構成

DU の基本動作は以下の通りである.

1) プログラム実行開始時,CU に命令開始アドレスをセットする.
2) DU は DUIM の内容をフェッチし, アドレス計算であれば AAU, 制御命令であれば CU で実行する.
3) DU はフェッチした命令が, データ移送命令がローカルメモリに関するものなら,DUDM からデータをフェッチして CLM に格納し, リモートメモリに関するものなら, リモートの DUDM にデータ送出要求を出して, 返って来たデータを CLM に格納する.

3') データ移送命令のオペランドに同期オプションがあれば, CLM については,PU とのトークン同期,DUDM については, 同期フラグによる同期機構を有しており, 同期チェックや処理を行なった後. データを転送する.

4) PU は,PUIM から命令をフェッチし, プログラムの実行を開始する. ロード/ストアがあり, 同期オプションがセットされていれば,DU からの同期を待ち, もしくはトークン送信を行なう. (DU,PU は並行に実行).

以上の手順により,PU がリモートデータを必要とするときには, そのデータのほとんどが CLM に格納されるようになるので, データアクセスの遅延がさけられる.

2.3 アドレス空間

図 3は, アドレス空間を表す.PUIMx☆,CLMx は,PU に接続されるデータメモリ (CLM) と命令メモリ (PUIM) であり, DSM は,DU に接続されるデータメモリ (DUDMx) の総称である. DU の命令メモリは,DUIM である. DUIM から DU1 にプログラムが送り込まれる. 図の例では,5000 バイトのサイズのメモリモジュールにより,DSM を構成していく. アドレス 0 から 4999 番地は,DU1 にとってローカルの DUDM であり, それ以降のアドレスは, リモートとなる. アクセスされたデータは, これらのアドレス体系とは異なるキャッシュレベルメモリ (CLM1) に格納される. これは, 他のメモリとは独立のアドレスを持つ. よって PU1 は,PU2 の CLM2 へアクセスすることはできない.

☆ 当節では,{x:1,2}.

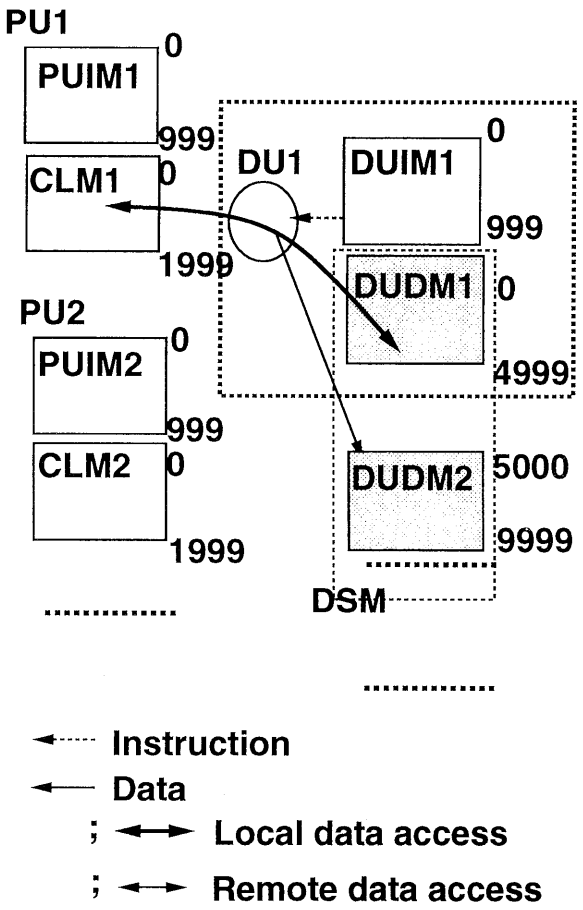

図3 アドレス構成

2.4 DU 命令

特に DU 独特の命令について説明する. 他の命令は通常のプロセッサと同様である.

この方式では, データ移送命令を有し,CLM と DSM 間のデータ移動と同期が記述できる. デー

タ移動は, 第 1 オペランドで移動元のアドレスを示すレジスタと移動すべきデータ数 (byte), 第 2 オペランドに移動先のアドレスを示すレジスタを記述することで行なう. 同期は,DSM 間では第 3 パラメータとしてコロンを挟んで記述される DSM の同期 bit 操作のための操作オプション {E(xist),N(oExist),S(et),R(eset)} を記述して行なう. 操作オプションは, 順に,DSM の同期フラグをチェックし 1 なら当命令実行, DSM の同期フラグをチェックし 0 なら当命令実行, 当命令実行後,DSM の同期フラグを 1 に, 当命令実行後,DSM の同期フラグを 0 にする. を表現する.

PU-DU 間は、命令末尾の token 操作のための操作オプション {S(end),R(eceive)} を記述し、PU-DU 間の同期をとる. 操作オプションは, 順に, 当命令実行後, 同期トークンを PU に送る, 当命令実行後, 同期トークンを PU から受けとる. を表現する.

例を示す.

(1) MMC 10:r1,r2,C:R,S

(2) MCM 10:r1,r2,N:S,R

この例で 1) は DSM の同期ビットをチェックし, リセットした後, DSM の r1 の内容で示される番地から 10 バイト分の内容を CLM の r2 番地から格納し, PU にトークンを送る. 2) は PU からトークンを受け取ったら, CLM の r1 の内容で示される番地から 10 バイト分の内容を DSM の r2 番地から格納し, DSM の同期ビットをセットする.

2.5 PU 命令

特に PU 独特の命令について説明する. 他の命令は通常のプロセッサと同様である.

この方式では, 通常のロード/ストアに同期オペランドが付与される. DU との同期は、命令末尾の同期ビット操作オプション {S(end), R(eceive)} で行なう。操作オプションは, 順に, 当命令実行後, 同期トークンを PU に送る, 当命令実行後, 同期トークンを PU から受けとる. を表現する.

例を示す.

(1) ST r1,10(r2),S

(2) LD 10(r1),r2,R

この例で 1) はレジスタ r1 の内容を CLM の r2 の内容+10 番地へ格納し、DU にトークンを送る。2) は DU からのトークンを待って、CLM の r1 の内容+10 番地の内容をレジスタ r2 へ格納する。

3. シミュレータ

本方式の評価のために, 2 つのシミュレータを作成した. 1 つは, 通常のキャッシュを用いた分散共有メモリ型マルチプロセッサ, 1 つは今回提案するデータ転送ユニットを用いたマルチプロセッサである.

PU/DU の命令仕様は, 前節で説明した命令仕様に示されるとおりである. 今回の構成では, 分散共有メモリを構成するメモリモジュール (DUDM) は, どちらの方式でも 64kbytes とする.

以下に各々のシステム構成を示す.

通常のキャッシュを用いたマルチプロセッサでは,PU から出されたロード/ストアは, キャッシュにあるかがテストされ, なければハードウエアにより, データを所有する DUDM にアクセスが行なわれる. キャッシュは,1 ライン 32bytes,512 ラインのダイレクトマップ, キャッシュ間のデータ整合は,write-once プロトコルによる.

DU を用いたマルチプロセッサシステムでは,DU が実行するデータ転送命令により CLM へハードウエアにより, データを所有する DUDM からデータ転送が行なわれる. データ整合は DU プログラムによる.CLM のサイズは,16kbytes とする.

4. 評　　価

本方式と通常のキャッシュ方式について, 前節のシミュレータを用い, 以下のプログラムを実行する.

このプログラムは, 2 つの PU が同じアドレスのデータを参照し, 演算を行なう. 通常のキャッシュ方式では,LD と ST 命令のアドレスを共有メモリのアドレスに変更し, 同期オペランドを除いたコードを実行する.

図 5は, サンプルプログラムのシミュレーション結果である. 横軸は, サイクルを示している. 縦軸上部はキャッシュ方式, 下部は, 提案方式の各システム構成要素の実行サイクルを示す. システムとしての実行サイクルは, 要素間の依存を踏まえ, これらのサイクルを加算したものとしてとらえられる.

キャッシュ方式との比較で以下の 2 点が示される.

- 提案方式では, 待ち要因が同期であり, キャッシュ方式に比べ短時間である
- 提案方式では, ネットワークの待ちがない

```
    PU1             DU1
                  | IMM r1,0
                  | IMM r2,0
IMM r1,0          | IMM r3,65536
LD 0(r1),r2,R <- MMC 2:r1,r2,C:R,S
LD 1(r1),r3 .     | INC r2
ADD r2,r3,r4      | INC r2
ST r4,2(r1),S   -> MCM 1:r2,r3,N:S,R
HALT              | HALT

    PU2             DU2
                  | IMM r1,65536
IMM r1,0          | IMM r2,0
LD 0(r1),r2,R <- MMC 2:r1,r2,C:R,S
                  | INC r1
LD 1(r1),r3       | INC r1
ADD r2,r3,r4      | INC r1
ST r4,2(r1),S -> MCM 1:r2,r1,N:S,R
HALT              | HALT
```

図4　サンプルプログラム

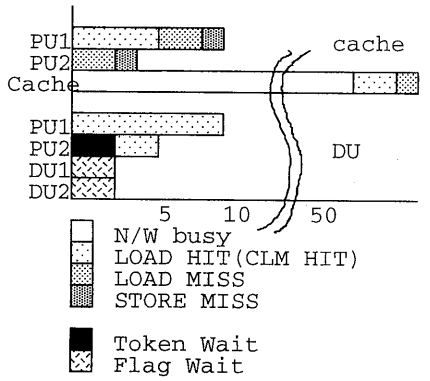

図5　評価結果

前者は,PU と DU の同時実行による待ち時間の隠蔽効果と考えられる. ただし, これはソフトウエアプリフェッチ方式でも, これに近い結果を得るが, ハードウエアを分割しているため, 一般にはソフトウエアプリフェッチ方式以上の性能を得る. 後者は, ライン転送による, 余計なデータ転送のオーバーヘッドと, 整合プロトコルの通信による差であるが, 顕著な差をみることができた.

この方式では

(1)　データ転送用プログラムが必要
(2)　データ転送用プログラムを格納する命令メモリとそれを実行するハードウエアが必要
(3)　同期機構に関するハードウエア資源の追加が必要

といった問題が考えられるが, 1) は, アプリケーションの蓄積により多くのアプリケーションについて自動化される予測がある 2) 3) はデータ転送のオーバーヘッド削減の効果が顕著であり, 十分意味がある と考える.

5. ま と め

メモリからキャッシュメモリへのデータ移動を, プログラムによって行う分散共有メモリ方式マルチプロセッサシステムを提案し, 従来のキャッシュシステムとの比較により, 評価した.

データ移動は命令により, バイトを基本単位として可変長で行なえるので, キャッシュに相当するメモリに, 必要最小限のデータを不整合なく移動することにより, 従来のキャッシュ方式より, 良い性能を示す.

今後は, さらに広範囲のアプリケーションでの評価を行ないたい.

参考文献

(1)　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

(2)　E,Gornish,E.Granston and A.Veidenbaum , Compiler-directed data prefetching in multiprocessors with memory hierarchies , Proc. 1990 Int'l Conf. on supercomputing, pp.354-368,1990

(3)　A.C.Klaiber and H.M.Levy, An architecture for software-controlled data prefetching, Proc. of the 18th Annual Intl. Symp. on Computer Architecture ,pp.43-53,1991

(4) T.Mowry and A.Gupta, Tolerating latency through software-controlled prefetching in shared memory multiprocessors, Journal of par. and dist. Comp. ,12(2):87-106,June1991

(5) T.Mowry , M.S.Lam and A.Gupta, Design and evaluation of a compiler algoritm for prefetching,Proc. of the 5th Intl. Conf. on Architectural Support for programming Languages and Operating Systems,pp.62-73,1992

付録:命令仕様

DU 命令仕様

書式	概要
MMC imm:rs,rd,s,t	DSM から CLM へ転送
MCM imm:rs,rd,s,t	CLM から DSM へ転送
IMM imm,rd	$imm \rightarrow rd$
ADD rs,rt,rd	$[rs]+[rt] \rightarrow rd$
SUB rs,rt,rd	$[rs]-[rt] \rightarrow rd$
INC rs	$[rs]+1 \rightarrow rs$
DEC rs	$[rs]-1 \rightarrow rs$
SRA rs,imm,rd	$[rs] >> imm \rightarrow rd$
SLA rs,imm,rd	$[rs] << imm \rightarrow rd$
B imm	$PC \leftarrow PC + imm$
BEQ imm	$ifZ = 1 PC+ = imm$
BNE imm	$ifZ = 0 PC+ = imm$
BMI imm	$ifN = 1 PC+ = imm$
BPL imm	$ifN = 0 PC+ = imm$
BLT imm	$ifNxorV = 1$ PC += imm
BLE imm	$ifZor(NxorV) = 1$ PC += imm
BGE imm	$ifNxorV = 0$
	PC += imm
BGT imm	$ifZor(NxorV) = 0$ PC += imm
NOP	NOP
HALT	$HALT$

PU 命令仕様

書式	概要
ST rs,offset(reg),t	$[rs] \rightarrow offset + [reg]$
LD offset(reg),rd,t	$offset + [reg] \rightarrow rd$
IMM imm,rd	$imm \rightarrow rd$
ADD rs,rt,rd	$[rs]+[rt] \rightarrow rd$
SUB rs,rt,rd	$[rs]-[rt] \rightarrow rd$
MUL rs,rt,rd	$[rs]*[rt] \rightarrow rd$
DIV rs,rt,rd	$[rs]/[rt] \rightarrow rd$
INC rs	$[rs]+1 \rightarrow rs$
DEC rs	$[rs]-1 \rightarrow rs$
AND rs,rt,rd	$[rs]and[rt] \rightarrow td$
OR rs,rt,rd	$[rs]or[rt] \rightarrow rd$
XOR rs,rt,rd	$[rs]xor[rt] \rightarrow td$
SLL rs,imm,rd	$[rs] << imm \rightarrow rd$
SRL rs,imm,rd	$[rs] >> imm \rightarrow rd$
SLLV rs,rt,rd	$[rs] << [rt] \rightarrow rd$
SRLV rs,rt,rd	$[rs] >> [rt] \rightarrow rd$
SRA rs,imm,rd	$[rs] >> imm \rightarrow rd$
SLA rs,imm,rd	$[rs] << imm \rightarrow rd$
B imm	$PC \leftarrow PC + imm$
BEQ imm	$ifZ = 1 PC+ = imm$
BNE imm	$ifZ = 0 PC+ = imm$
BMI imm	$ifN = 1 PC+ = imm$
BPL imm	$ifN = 0 PC+ = imm$
BLT imm	$ifNxorV = 1$ PC += imm
BLE imm	$ifZor(NxorV) = 1$ PC += imm
BGE imm	$ifNxorV = 0 PC+ = imm$
BGT imm	$ifZor(NxorV) = 0$ PC += imm
NOP	NOP
HALT	$HALT$