計算機アーキテクチャ 130－1
（1998. 8. 5）

超並列計算機 JUMP-1 における分散共有メモリ管理の実装とその評価

秤谷雅史[†] 小西将人[†††] 山内聡[†††]
前川勉[†††] 津田健[†††] 山本考伸[††]
五島正裕[†] 森眞一郎[†] 富田眞治[†]

JUMP-1 では、分散共有メモリ管理を専用のプロセッサ MBP-light を用いてソフトウェアで柔軟に管理する方式を採用している。本稿では MBP-light 上で動作する共有メモリ管理プログラムのプロトタイプを実装し、それをもとにして MBP-light の性能の評価を行った。MBP-light はパケットの操作を柔軟に行える一方で、クラスタメモリ及びローカルメモリアクセスに時間がかかってしまう弱点がある。当初は読み出し要求は 127 サイクルで終了するとされていたが、実際にはその 6.6 倍の 845 サイクルもの時間がかかることがわかった。クラスタバスへのロック及びクラスタメモリへのアクセスが高速に行えるハードウェアサポートを用意した場合には約 150 サイクルほどの高速化が期待でき、MBP-light は DSM 管理プロセッサとして特殊命令を多数持つプロセッサとするべきである。

Evaluation for Implementation of Distributed Shared Memory Management of the Massively Parallel Processor JUMP-1

Masashi Hakariya,[†] Masahito Konishi,[†††] Satoshi Yamauchi,[†††]
Tsutomu Maekawa,[†††] Takesi Tsuda,[†††] Takanobu Yamamoto,[††]
Masahiro Goshima,[†] Shin-ichiro Mori[†] and Shinji Tomita[†]

MBP-light is distributed shared memory(DSM) management controller of the Massively Parallel Prosessor JUMP-1. The coherence of inter-clusters is managed flexibly by the management programs for MBP-light. In this paper, we describe about this programs and evaluate the performance of MBP-light. On the opposite side of handling packets flexibly, MBP-light has some weakpoints such as it takes a lot of cycles to access its local memory and the cluster memory. By our experiments, we have learned that a read request between clusters takes 845 cycles, 6.6 times more than expected cycles. By providing hardware to reduce cluster bus lock and cluster memory access time, about 150 cycles will decrease. Therefore MBP-light should have more special operations in order to function as DSM management controller.

1. はじめに

我々は現在 CC-NUMA 型超並列計算機 **JUMP-1** を開発している。JUMP-1 ではクラスタ間での分散共有メモリ (**DSM**: Distirbuted Shared Memory) 管理をディレクトリ方式に基づきプログラムベースで行う。これは処理の高速性もさることながら、コヒーレンス制御の柔軟性や処理の複雑さなどにも対応しなければならないからである。

このアプローチによれば DSM 管理方式の柔軟性は保たれるが、単に汎用のプロセッサを用いて実現しても必要とされる処理能力を得ることは難しい。そこで JUMP-1 では DSM 管理のための専用のプロセッサ **MBP-light** を開発し、柔軟性と高速性の両立を図っている。

本稿では MBP-light 上で動作する DSM 管理プログラムのプロトタイプ実装について報告し、その実行速度に基づいて DSM 管理プロセッサとしての MBP-light の評価を行う。

2章では対象となる計算機である JUMP-1 の概要としてそのクラスタ構成と MBP-light の構成について述べる。3章では DSM 管理方式と実装したプログラムについて述べ、最後に 4章では作成したプログラムによるシミュレーションの結果から性能の評価を行う。

† 京都大学大学院情報学研究科通信情報システム専攻
Division of Communications and Computer Engineering, Graduate School of Informatics, Kyoto Univ.
†† 京都大学大学院工学研究科情報工学専攻
Division of Information Science, Graduate School of Engineering, Kyoto Univ.
††† 京都大学工学部情報学科
The department of Information science, in Faculty of Engineering, Kyoto Univ.

2. JUMP-1 の概要

2.1 JUMP-1 の構成

JUMP-1 はクラスタを 1 つの単位とし、クラスタ間を **RDT**(Recursive Diagonal Torus) を用いて相互に結合された超並列計算機である。その結合網である RDT は 2 次元トーラスが Fat-tree 状に階層化された構造を持つ。

JUMP-1 のクラスタの構成を図 1に示す。クラスタ内では 4 つのプロセッサカードと 1 つのメモリユニットがクラスタバスで結合されている。

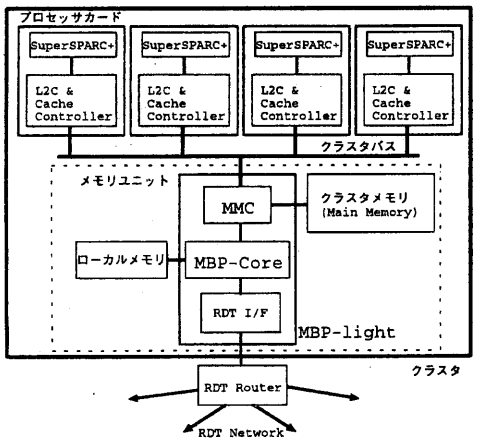

図 1　JUMP-1 のクラスタ構成

- プロセッサカード…1 次キャッシュを内蔵する SuperSPARC+及び外部 2 次キャッシュ(L2C) とそのキャッシュを制御する Cache Controller(以下 CC) を主体とする。
- メモリユニット…クラスタメモリ (64MByte) とその制御行う専用プロセッサ **MBP-light**[2),5)] とに分割される。クラスタメモリは主記憶として使用するだけでなく、クラスタ間でのメモリアクセスに対する 3 次キャッシュとしても利用する。MBP-light については次節で詳細を説明する。

JUMP-1 ではパケット[6)]を交換することによってクラスタメモリアクセスを行なう。1 つのパケットは 64+2bit を 1 つのフリットとし、1 つのヘッダ部と 1 ブロック (32Byte) に対応した 4 つのデータ部との合計 5 フリット☆から構成されている。

2.2 MBP-light

2.2.1 MBP-light のハードウェア構成

DSM 管理プロセッサの行うべき処理は、パケットの種別の判別やキャッシュラインの転送、ラインの状態の変更などのクラスタ間でのコヒーレンス制御である。したがってパケット操作の特殊な命令を必要とし、汎用のプロセッサでは十分な性能が得られない。そこで JUMP-1 では DSM 管理を行うための専用プロセッサ **MBP-light** を開発した。

MBP-light はクラスタメモリのコントローラ (MMC) とコアプロセッサである **MBP-Core** 及び外部クラスタとのインターフェース (RDT I/F) から構成される (図 1)。外部には 21bit×64K エントリのアドレス空間を持つローカルメモリを備えている。

MMC はクラスタバス及び MBP-Core からのメモリアクセス要求パケットの処理を行う。クラスタメモリはメモリデータの有効性及び共有状態を示すタグを所有しており、MMC はタグの値を見て共有状態がクラスタ内で閉じていなければ MBP-Core に処理の依頼を行う。同様に RDT 側からのパケットの到着があった場合にも処理の依頼を行う。

更新要求パケットは double word 単位での更新を指定することができるが、MMC にはこの機能がサポートされておらずライン単位での書き込みしかできない。したがってこの機能をサポートするために MBP-Core 上のプログラムによって更新処理を行っている。

MBP-Core は命令長 21bit データ長 16bit の RISC プロセッサで以下のレジスタを所有している。

- 16bit 汎用レジスタ (GPR: Genenal Purpose Register)
- 内部メモリ (16bit×256)
- パケット格納用レジスタ (**PBR**: Packet Buffer Register)

GPR は GPR 間の演算だけでなく PBR-GPR 間の演算を行うことができる。PBR については次項で詳しく説明を行う。

また、内部メモリには MMC の送受信バッファの状態やバスへのロックレジスタなどがマッピング[4)]されており、MBP-Core からこれらのレジスタに対する読み書きを行うことができる。

2.2.2 パケットの取り扱い

前述のように MBP-light の行うべき処理はキャッシュラインの転送及び状態の更新などである。したがってパケットの判別及び操作を柔軟にかつ高速に行わなくてはならない。そこで MBP-light にはパケット格納用の専用のレジスタが用意されおり、PBR(Packet Buffer Register) と呼ぶ。

PBR は 64+2bit でパケット 1 フリットに 1 本割り当てられている。PBR は汎用が 64 本と RDT の送受信用にそれぞれ 8 本×3 組ずつ用意されている。

RDT 送受信用 PBR は FIFO 構造をとり、MMC 及び MBP-Core からは送信用 PBR の末尾と受信用 PBR の先頭のみが可視となっている (図 2)。プログラムによって FIFO がデキュー及びエンキューされ受

☆ 正確にはヘッダ部のみまたはヘッダ部とデータ部 1 フリットのパケットも存在する。

信待ち状態及び送信状態となる。

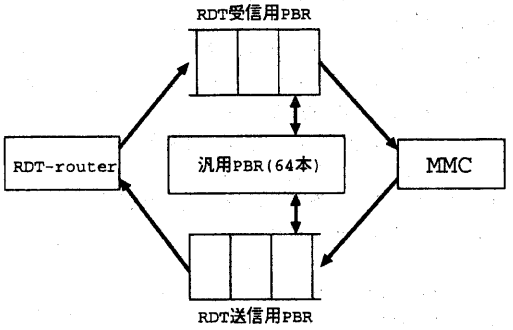

図2 PBR

MBP-light にはPBR操作の専用の命令セットが用意されており、PBR-GPR及びPBR-PBR間の演算やデータ転送及びパケットのヘッダ作成などが容易に行える。

3. DSM管理

クラスタ間で共有されるデータの管理は全てMBP-lightが行う。本章ではJUMP-1でのDSM管理方式とMBP-light上に実装したDSM管理プログラムについて述べる。

3.1 JUMP-1のコヒーレンスプロトコル

JUMP-1ではクラスタ内に有効なデータを所有するクラスタを**Renterクラスタ**と呼ぶ。

Renterクラスタは自クラスタメモリを前述のように3次キャッシュとして用い、**Homeクラスタ**と呼ばれる特別なクラスタのクラスタメモリを主記憶としてキャッシングを行う。Homeクラスタは分散配置されており通信の集中化を防いでいる。

JUMP-1ではオーナーシッププロトコルを採用している。有効なデータを所有するクラスタのうち、キャッシュのミスヒット時に応答する責任を持つクラスタを**Ownerクラスタ**と呼ぶ。Ownerクラスタは最新のデータを所有するクラスタとし、Homeクラスタ上に有効なデータが存在する場合にはHomeクラスタが、有効なデータが存在しない場合には無効化要求を発行したクラスタがOwnerクラスタとなる。

- Homeクラスタ…あるアドレスについて、他のクラスタメモリの主記憶として機能するクラスタメモリを所有するクラスタ。
- Ownerクラスタ…読み出し要求に答える義務を持つ最新のデータを所有するクラスタ。
- Renterクラスタ…有効なデータを持つHomeクラスタ以外のクラスタ。

3.2 JUMP-1のコヒーレンス制御

クラスタ内でのコヒーレンス制御は共有バスベースのスヌープ方式を採用している。頻繁に制御要求が起るので、高速処理が可能なハードウェアで構成されている。クラスタ内のキャッシュコヒーレンスはDSM管理プログラムから不可視であるので以下そのことについての説明は行わない。

以下クラスタ間でのコヒーレンス制御について述べる。

クラスタ間でのキャッシュコヒーレンスはディレクトリ方式を採用している。ディレクトリはページ(4KByte)単位で持ち、Homeクラスタに置かれる。

ある共有されたデータに対するクラスタ間のパケットの転送及び一連の状態遷移を**トランザクション**という。トランザクションを開始するクラスタを**Initiatorクラスタ**と呼ぶ。

典型的なトランザクションの流れを図3に示す。

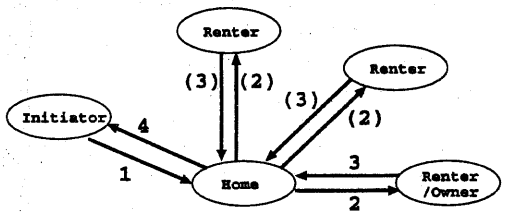

図3 クラスタ間でのパケットの流れ

(1) 読み出し要求が発行されたInitiatorクラスタはHomeクラスタに対して要求パケットを送信する。
(2) Homeクラスタはその要求に対してOwnerクラスタに要求を送信する。
(3) Ownerクラスタは応答パケットをHomeクラスタに送信する。
(4) Homeクラスタはその応答パケットを Initiatorクラスタに送信する。

更新要求の場合にはInitiatorクラスタからHomeクラスタに要求パケットが到着すると、Homeクラスタは Ownerクラスタだけでなく、Renterクラスタにもパケットを送信し(図3の()の送信)データが更新される。

3.3 DSM管理プログラム

3.3.1 プログラムの基本構造

MBP-lightでの処理は、基本的に、パケットの到着による割り込みによって起動される。作成したプログラムはその実行方法に関して以下のことを考慮した:

- DSM管理のための通常の処理は、普通長くとも数百サイクル程度で終了する。したがってそれらの処理は、割り込みなどがかからないようにして、一気に実行してしまう。
- MBP-lightは、ページ単位の処理など、通常のDSM管理の処理に比べて非常に長い時間のかかる処理を同時に扱わなければならない。そのよう

な処理に対しては割り込みをかけ、通常の処理を優先的に行う。

これらのことを実現するために今回は、到着したパケットの処理をプロセスとして実現することで対応[1]した。したがってプログラムはパケットの処理を行うプロセスとそのプロセスを管理するカーネルの2階層のシステムとして実現される。

3.3.2 プロセス管理

前述したようにMBP-lightは、通常のDSM管理に関する処理と、長い時間を要する処理とを扱う必要がある。したがってプロセスの優先順位としては、基本的には、以下の2レベルを用意する：

(1) 高位…通常のDSM管理に関する処理は、割り込み/横取り不可とする。

(2) 低位…その他の処理は割り込み/横取り可とし、処理中に新たなパケットが到着したときには、割り込まれ横取りされる。

カーネルは要求パケットが到着すると、RDT送受信用PBRへのロード(図2)及びスタックの確保等を行いプロセスを生成し、即座にそのプロセスの実行を開始する。プロセスはその処理中に低位の処理であると判断すると割り込みを許可する。

カーネルは横取りされたプロセスのための実行可能キューを用意し、高位のレベルのプロセスが存在しない場合にのみディスパッチする。このレベル内の各プロセスの間のスケジューリング方式には任意性があるが、今回は単純にFIFOとしている。

プロセスの解放は要求パケットの処理が終了しても行なわず、その対応する応答パケットの処理が終了した際に行い、プロセスを生成する手間を削減している。すなわち、要求パケットの処理が終了し応答パケット待ちとなったプロセスは、その解放を行なわずにプロセステーブルに登録する。このテーブルはアドレスをキーとする連想テーブルである。

上述したテーブルはペンディング・トランザクション・テーブル(以下PTT)[1]と呼び、プロセス間の競合の検出にも用いている。要求パケットが到着した場合にはPTTを検索し当該アドレスに対して処理がなされていなければ、PTTに登録を行いパケットに対する処理を実行する。このときすでに登録があれば待ちプロセスとしてフックする。待ちプロセスはフックしているプロセスの応答パケットが到着し、その処理が終了すると実行が開始される。同一のアドレスに対する要求が到着した場合にも処理の順序が入れ替わることはない。

プロセスが横取りされると、カーネルはそのコンテクストとして各種レジスタ及びPBRの内容を保存する。RDT送受信用のPBRはFIFO構造(図2)をとっているので常に可視である汎用のPBRにパケットを移動する。移動先の汎用のPBRが不足している場合にはパケットをプロセススタック上に保存する。

3.3.3 DSM管理プロセス

上位層であるDSM管理プロセスは主に次のような動作を行う。

(1) パケットのクラスタバス及びRDTへの転送

(2) クラスタ間でのコヒーレンスを保持するためのパケットの作成及び送信

(3) 共有状態の更新が行われた場合のクラスタメモリのタグの変更

(4) クラスタメモリへの書き込み及び更新

(5) クラスタ間の共有状態を示すディレクトリ及びページテーブルの管理

2.2.1項で述べたようにMMCは共有状態がクラスタ内に閉じている場合にはMBP-Coreへの割り込みをかけずに、ハードウェアによる処理のみで応答パケットをクラスタバスに送信する。この時MBP-Coreはそのことに関してはまったく不可視である。したがって3の共有状態の更新が行われる場合には、その間にクラスタメモリへのアクセスを防止するためにクラスタバスへのロックを行ってから実行しなければならない。

また4の更新処理はプログラムによるdouble word単位での更新を実装するために、プログラムによってデータを読み出し内容を更新した後に書き込むという操作を行う。

4. 性能評価

4.1 評価環境

VHDLで論理合成可能なように記述された設計データを用い、Mentor Graphics社のVHDLシミュレーターqhsim v8.5_4.6cを用いてシミュレーションを行った。したがってサイクル数は実際のものと同じである。発行される要求パケット数は1つのみ☆でクラスタ数は2、各クラスタにはプロセッサを1つとした。プログラムはアセンブリ・コードで記述している。

クラスタ数が高々2つであるのでHomeクラスタには共有状態を示すディレクトリは作成せず、共有状態の判断はクラスタメモリのタグをMBP-lightが読み出すことで判断している。また、仮想メモリは実装されておらず、ページテーブルは所有していない。したがって実際のJUMP-1ではさらに処理時間が増大することを考慮されたい。

4.2 処理に要する時間

表1に上位層であるパケットの処理に要する時間を示す。主要なパケットである読み出し要求と更新要求の2つをとりあげた。読み出し要求はHomeクラスタ内のCC上のデータを他方のクラスタ(Initiatorクラスタ)から読み出す場合としている。

JUMP-1ではInitiatorクラスタでデータの更新を

☆ つまりパケットの到着時にMBP-lightでの処理が直ちに開始される。

表1　各パケットの処理時間

要求	時間(サイクル)
読み出し要求(at Initiator to Home)	47
読み出し要求(at Home from Initiator)	70
読み出し応答(at Home to Initiator)	162
読み出し応答(at Initiator from Home)	145
更新要求(Initiator from CBus)	270
更新要求(Home from RDT)	324
更新要求(Renter from RDT)	281

投機的[3]に行っている。したがって表のサイクル数はHomeクラスタへのパケットの送信だけでなく、クラスタメモリの更新処理のサイクル数も加算されている。

Homeクラスタでの更新要求の処理に時間がかかっているのは、自クラスタ内の更新だけでなく、Renterクラスタへの更新パケットの生成及び送信を行っているからである。

詳しい処理内容とそのサイクル数については次節で述べる。

次に下位層のプロセス管理の処理時間を表2に示す。

表2　プロセス管理における処理時間

処理	時間(サイクル)
プロセス生成(Cbus Req)	57
プロセス生成(RDT Req)	101
プロセス復帰(Cbus Rpy)	34
プロセス復帰(RDT Rpy)	71
プロセスの終了	15
プロセスの中断	12
コンテクストの保存	211
コンテクストの復帰	156

プロセスの生成及び終了とは、プロセス用のスタックの確保及び解放などである。今回は他のパケットが発行されないのでコンテクストの切り替えは実際にはおこらない。したがってコンテクストの保存及び復帰は手でサイクル数を求めた。コンテキストはローカルメモリ上に保存している。

3.3.2項で述べたように応答パケットの受信待ちの際にプロセスの解放は行わない。プロセスの中断とはPTTにコンテキストを保存する処理を、同様にプロセスの復帰とはコンテキストをPTTから復帰する処理を指している。

RDT側からの到着パケットのプロセスの生成及び復帰がクラスタバス側からのものと比較して値が大きくなっている。これはRDT側からのパケットの到着割り込みは要求パケットと応答パケットの区別がなくその判断を行ったり、デッドロックの防止するためにMMCからクラスタバスへの送信バッファの確保及び受信バッファのsweep outを行っているためである。

その理由は次の通りである。前述したようにMMCはクラスタメモリのタグを見て、MBP-Coreへの割り込みの判断を行う。したがってRDT側から到着したパケットが共有状態を変更するものであればその間にメモリアクセスのないよう、クラスタバスをロックしなければならない。しかし、共有状態の変更を要するパケットであるかは実際にそのパケットの処理を開始しなければ判断できない。すなわちその間に他の要求パケットによるメモリアクセスが行われてしまう可能性がある。

したがって、今回のプログラムではパケットの処理を行う前にカーネルが全てのパケットに対してクラスタバスをロックし、バスからの受信バッファをsweep outすることによってコヒーレンスを保持している。

4.3　所要サイクル数の詳細

読み出し要求

MBP-lightが開発された当初は読み出し要求は127サイクルで完了するとされていた[2]が、実際はその6.6倍の845サイクル☆もの処理時間がかかってしまう結果となった。

図4に読み出し要求の所要サイクル数の内訳を示す。各値は1つの読み出し要求の処理に対するInitiatorクラスタとHomeクラスタでの合計サイクル数である。ハードウェアによる処理とはクラスタ間の通信などMBP-Coreによる処理以外の処理を指している。

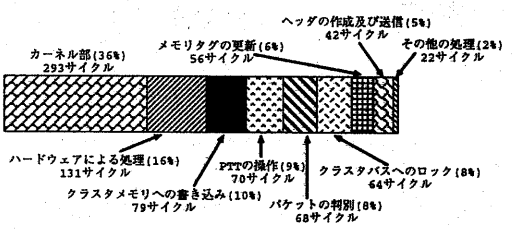

図4　読み出し要求の各処理の所要サイクル数

クラスタバスへのロックに多くの時間がかかっているが、これは2.2.1項で述べようにMMCの内部レジスタに値を書き込むことによってロックを行わなければならないからである。この方法では即座にロックをかけることはできず、Atomictyの確保から考えるとこれは望ましいことではない。

今回のプログラムでは、前述したようにカーネルが全てのパケットに対してクラスタバスをロックを行う。したがって正確には表中のクラスタバスへのロックという項目は、マスクの変更とロックの解除を指している。

バスへのロックとその解除を1サイクル、さらにヘッダ部の作成をすることなくクラスタメモリ及びそのタグへのアクセスを実現できるハードウェアを用意すれば、約150サイクル程度の高速化が期待できる。

☆ この値はInitiatorクラスタのクラスタバスに要求パケットが流れてから、Homeクラスタからの応答パケットがInitiatorクラスタのクラスタバスに送信されるまでのサイクル数である。

また、クラスタメモリへの書き込み及びそのタグの更新はそれぞれ書き込みパケットヘッダを用意してアクセスしなければならない。表中の値はその作成に必要なサイクル数も含んだ値である。

PTT への操作はローカルメモリ上にその領域を確保しており、そのアクセスに 1 サイクルのストールを伴なう。したがってサイクル数がその分増大している。命令キャッシュを実装するなどして 1 サイクルでアクセスが可能であれば、約 20 サイクル程度の高速化が図れる。また、管理が難しくなるが PTT をハードウェアでサポートし、その登録・検索等が 1-2 サイクル程度で可能となれば 60 サイクル程度の高速化が図れる。

更新要求

図 5に Home クラスタに RDT 側から更新要求パケットが到着した場合の所要サイクル数の内訳を示す。4.1節で述べたがメモリタグの読み出しは共有状態の判断のためのアクセスである。

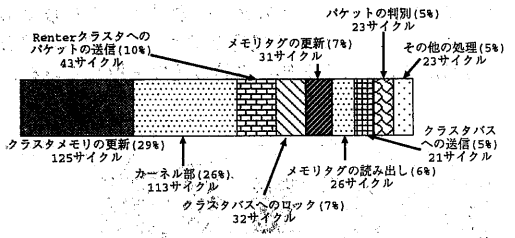

図 5　Home クラスタでの更新要求の各処理の所要サイクル数

表 1の値と比較するとクラスタメモリの更新がカーネル部を除く処理の約 4 割を占めており、所要サイクル数の増大に大きく影響していることがわかる。これは 2.2.1項で述べたが double word 単位での更新をプログラムによってサポートしているためである。クラスタメモリ及びタグの更新がハードウェアで高速に処理が可能であれば 37%の高速化が図れる。

また、読み出し要求と同様にローカルメモリアクセス等を高速に実行できるハードウェアサポートした場合には、プロセススタック管理などが高速化され、さらに 5〜7%程度の性能向上が図れる。

5. おわりに

本稿では JUMP-1 における MBP-light 上で動作する DSM 管理プログラムの実装と、それを用いた MBP-light の性能評価の結果について述べた。

MBP-light は柔軟なパケットの処理が可能であるがメモリアクセスなどにヘッダの作成を行う必要や、テーブルなどの検索のためのローカルメモリアクセスに時間がかかってしまうという弱点がある。

実在するアプリケーションを動かすことができればさらに正しい評価ができるであろうが JUMP-1 がまだ完成していないために、本稿では要求パケットが 1 つのみ発行されるという条件での評価を行った。この評価では読み出し要求パケットが発行されてから応答パケットが到着するまでにかかる時間が当初の予想よりも 6.6 倍もの時間がかかってしまう結果となった。

コンテキストの切り替えについてはサイクル数を参考程度に挙げることしかできなかった。しかし複数のレジスタセットを用意して、その切り替えによってコンテキストを切り替えを行うことができればかなりの高速化が期待できる。

DSM 管理プロセッサはキャッシュラインの転送や状態の更新という特殊な処理を管理するプロセッサである。したがってクラスタバスへのロック及びクラスタメモリアクセスなどの特殊命令をサポートするべきである。これらの命令がサポートされ、さらに命令キャッシュなどが実装されてローカルメモリアクセスの高速化がなされた場合、読み出し要求では約 170 サイクル (20%) 程度の高速化が図れるという結果となった。

結論としては DSM 管理プロセッサは、DSM 管理に特化した命令体系である方がより高いパケット処理能力が得ることができると言える。

謝　辞

日頃から貴重な御意見を頂いている文部省重点領域研究共同研究者各位に感謝します。

メンター・グラフィックス・ジャパン株式会社の Higher Education Program の一環として製品とサービスをご提供頂いたことに感謝します。

なお本研究の一部は、文部省科学研究費補助金（基盤研究 (B) 課題番号 10480059、10558045、 基盤研究 (C) 課題番号 09680334、奨励研究 (A) 課題番号 09780268）による。

参考文献

1) 山本 考伸 他:「超並列計算機 JUMP-0.5 における分散共有メモリ管理手法」情処研報 97-ARC-125, pp. 85-90.
2) 佐藤 充 他:「超並列マシン JUMP-1 のための分散共有メモリ管理プロセッサ」、並列処理シンポジウム JSPP '97, pp. 265-272.
3) 福島 直人:「遷移的矛盾を許容するメモリ・モデルに基づく一貫性制御方式」、修士論文, 京都大学 (1996)
4) 井上 浩明：MBP Core から見る MMC の内部 Register(1997 9/5)
5) JUMP-1 設計グループ：MBP-light マニュアル (1997 12/5)
6) 五島 正裕、他：JUMP-1 CBus 仕様書 (1994 7/29)