計算機アーキテクチャ　112－6
（１９９５．　６．　２３）

メッセージパッシング超並列システム アーキテクチャMESCAR

山田　茂樹、田中聡*、丸山勝己
NTTネットワークサービスシステム研究所、*NTTソフトウェア株式会社

概要　本論文は超並列システムを適用ターゲットとして、メッセージパッシングプログラミングモデルを前提に分散共有メモリを用いたメッセージ結合アーキテクチャMESCAR(Message-Coupled Architecture for Distributed Shared Memory Systems)の設計コンセプトを述べたものである。MESCARでは分散共有メモリ間コピー機構を用いて送受信プロセッサモジュール間でメッセージとメッセージ管理／制御情報の転送を行うことにより、メッセージバッファの捕捉から解放までのメッセージパッシングの一連の処理を効率化する。プログラミングと机上計算によりMESCARがメッセージレイテンシとカーネルオーバヘッドの削減に効果があることを示している。

MESCAR: A Message-Coupled Architecture for Massivelly Parallel Message-Passing Systems

Shigeki Yamada, Satoshi Tanaka*and Katsumi Maruyama
NTT Network Service Systems Laboratories　* NTT Software Corporation

Abstract　This paper describes the message-coupled architecture (MESCAR) for distributed shared memory (DSM) systems, which effectively supports the whole span of message passing processing from message buffer allocation to message buffer release. MESCAR utilizes an inter-DSM copying mechanism to transfer messages and message control and management data between a sending processor module and a receiving processor module. The programming experiments and latency calculations indicate that MESCAR is effective in reducing message latency and the kernel execution overhead for message passing.

1. まえがき

超並列システムアーキテクチャは、メモリモデルの観点から分類すると各プロセッサが独立したローカルメモリを持ち、メッセージパッシングにより処理を進める『ローカルメモリアーキテクチャ』（例えばAP1000[1]）と、全プロセッサ共通の共有メモリ空間を有し、共有メモリ空間上で共有メモリプログラミングモデルまたはメッセージパッシングモデルを実現する『共有メモリアーキテクチャ』とに分けられる。ローカルメモリアーキテクチャは、一般にメッセージパッシング開始から終了までにDMA転送や割り込み処理等、多くのソフトウェアの実行が必要で、メッセージレイテンシの増加やメッセージパッシングスループットの低下を招く場合が多い。

一方、共有メモリアーキテクチャについては超並列システムでは共有メモリへのトラヒック集中を回避するため、共有メモリを各プロセッサに分散配備する分散共有メモリ（DSM：Distributed Shared Memory）方式が用いられる。更にDSM方式はソフトウェアによって複数のDSM間でページ単位にコヒーレンシを維持する『ソフトウェアDSM型』（例えばIvy[2]）と、ハードウェアでコヒーレンシを維持する『ハードウェアDSM型』とに分けられる。ソフトウェアDSMでは、コヒーレンシ維持のソフトウェアオーバヘッドにより性能が向上しない欠点が指摘されている。ハードウェアDSMとしては共有メモリのコピーを記憶するキャッシュ間でコヒーレンシを維持するシステム（例えばDASH[3]、JUMP-1[4]、阿修羅[5]等）や、直接、DSM間でコヒーレンシを維持するシステム（Galactica Net[6]、SESAME[7]、MEMNET[8]、Merlin[9]）があるが、これらは共有メモリプログラミングモデルをサポートするもので、コヒーレンシ維持ハードウェア機構が複雑化しコストの増加を招いている場合が多い。

一方、メッセージパッシングモデル専用のハードウェアDSMとして、DSMの同一アドレス間で単純コピーをするDSMコピー機構により、ハードウェア構造を簡単化する方法（SHRIMP[10]）も提案されている。しかし、SHRIMPの主眼はメッセージパッシングの一部であるメッセージ転送の高速化のみを狙ったもので、メッセージバッファ捕捉から解放に至るメッセージパッシングの一連の処理を総合的にサポートしたものではない。

本論文ではメッセージパッシングモデルがネットワークも含めた今後の大規模分散処理と親和性があるとの認識に立ち、メッセージパッシングの一連の処理をソフトウェアとハードウェアでバランス良くサポートし、プロセッサ間通信のメッセージレイテンシとソフトウェアオーバヘッドを削減するメッセージパッシング超並列システムアーキテクチャMESCAR(Message-Coupled Architecture for Distributed Shared Memory Systems)を提案する。MESCARがサポートするプログラミングモデルはメッセージパッシングであるが、ハードウェア構造上は分散共有メモリを用いている点に注意されたい。

2. MESCARの設計アプローチ

2．1 メッセージパッシング処理内容

本論文で扱う非同期型メッセージパッシングの処理内容は以下のステップに分解される。

[A]送信側の処理

[1.1]送信メッセージバッファ（MB）の捕捉

[1.2]送信オブジェクトによる送信MBへのメッセージ書き込み

[1.3]送信側から受信側へのメッセージ転送

[1.4]送信MBの解放

[B]受信側の処理

[2.1]受信MBの捕捉

[2.2]受信MBへのメッセージのストア

[2.3]受信オブジェクトへのメッセージ到着通知

[2.4]受信オブジェクトによる受信MBからのメッセージ読み出し

[2.5]受信MBの解放

送信オブジェクトと受信オブジェクトが、異なるプロセッサで実行される場合の上記各ステップにおける問題点を分析し、MESCARでのアプローチをステップワイズに説明する。

2．2 MESCARにおけるアプローチ

（1）**共有メモリへのMB配置**：上記[2.1]の受信MB捕捉処理を、送信MBと受信MBを一致させることにより削除する。具体的には共有メモリアーキテクチャを用いて、送信プロセッサが捕捉した送信MBを受信プロセッサに引き継ぎ、受信MBとして使用する。

（2）共有メモリへの制御データエリアの配置と双方向通信：送信MBと受信MBの一致により、上記[2.3]のメッセージ到着通知ステップで送信MBアドレスを送信プロセッサから受信プロセッサに通知する必要がある。また、[2.5]の受信MB解放ステップで受信MBの解放時期を受信プロセッサからMB管理権を持つ送信プロセッサに通知して送信／受信MBを同時に解放する必要がある。そこで共有メモリ上にこれらの制御情報を記憶する制御データエリアを配置し、送信プロセッサ、受信プロセッサの双方から読み書きを行うことで制御情報を交換する。

（3）アクセス競合対策：MBとその制御データエリアを共有メモリ上に配置した場合、複数の送信プロセッサが同一エリアを捕捉して書き込みを行うと論理矛盾を起こす。これを回避するために共有メモリエリアを（送信プロセッサ、受信プロセッサ）のペア対応に論理分割し、各送信プロセッサは自分専用のエリアに書き込むことにより、送信プロセッサ間競合を回避する。また、3.4節で述べるように共有メモリエリアの一部をFIFOメモリ構造にすることにより受信側での競合をハードウェアで解決する。これによって[2.3]の受信オブジェクトへのメッセージ到着通知を効率化する。

（4）DSM間コピー：論理分割された共有メモリエリアを送信プロセッサ側と受信プロセッサ側の分散共有メモリ（DSM）に重複実装した重複型DSM構成を用いる。重複によって送信DSM、受信DSM間のコヒーレンシ維持が必要となる。しかしメッセージパッシングモデルではMBへのメッセージの書き込み後にMBからの読み出しを行うと言う実行順序を保証しているので、共有メモリプログラミングモデルのような複雑なコヒーレンシ制御は不要で、3.2節で述べる分散メモリカップラによる単純なDSM間コピーで実現できる。

（5）DSMの保護：共有メモリ空間が多数のプロセッサで共有されるために不正アクセスに対する保護手段が必要となる。そこで3.3節で述べる分散メモリプロテクタによって重複型DSMの保護を実現する。

なお、MESCARの共有空間は論理アドレス＝物理アドレスとして単純化しているが、仮想記憶を用いる場合には送信側と受信側に論理⇔物理アドレス変換テーブルを追加すれば物理アドレスの場合と同様の操作が実現できる。また、以下では簡単のためにライトスルーキャッシュを仮定しているが、適切なタイミングでダーティデータをキャッシュにプッシュする命令をコンパイラが生成することにより、ライトバックキャッシュも適用可能である。

3．　MESCARシステム構成法

3．1　システム全体構成

MESCARのシステム構成を図1に示す。各プロセッサモジュール（PM）は、プロセッサ、ローカルメモリ、分散共有メモリ（DSM）、分散メモリカップラ、分散メモリプロテクタを持ち、PM間はプロセッサ間通信路(Processor Interconnect)で相互接続される。プロセッサ間通信路は送信PMから送出したデータを順序逆転せずに受信PMに送り届ける『順序保証機能』を有する。各プロセッサは自PMに属するDSMをアクセスすることができるが、他PMのDSMはアクセスできない。

3．2　分散メモリカップラの構成

分散メモリカップラは、DSM間のコピーを実行するハードウェア機構で、図2に示すようには自PMのDSMに割り当てられたページアドレスと同じアドレスを持つDSMを保有するPMのID（PM-ID）を共有ページディレクトリに記憶している。分散メモリカップラは、プロセッサからDSMへの書き込みバスサイクルを常時監視しており、書き込みを検出すると書き込みアドレスを共有ページディレクトリに供給し、一致したページアドレスに対応するPM-IDを取り出す。これを書き込みアドレス、書き込みデータ、書き込み幅情報に付加

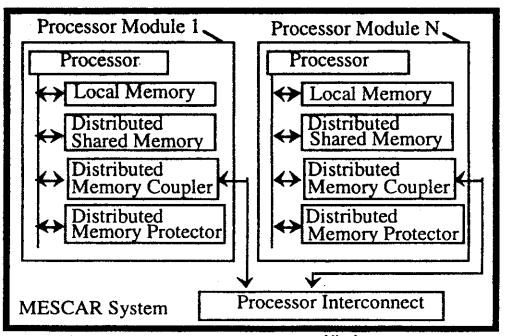

図1 MESCARのシステム構成

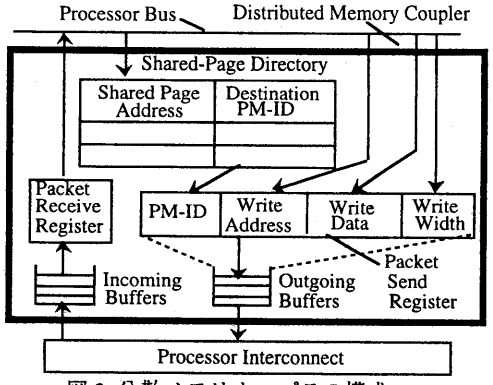

図2 分散メモリカップラの構成

してコピーパケットを作成し、プロセッサ間通信路に送り出す。プロセッサ間通信路は、指定された受信PMにコピーパケットを送り届ける。受信PMの分散メモリカップラは、受信DSMの、指定されたアドレスエリアにコピーデータを書き込むことでコピーが完了する。

DSM上の制御データエリアに関しては送信側の分散メモリカップラに受信PM-IDを登録するとともに、受信側の分散メモリカップラに送信PM-IDを登録しておく。これによって送信DSM、受信DSM間の双方向コピーが実現される。

3．3　分散メモリプロテクタの構成

分散メモリプロテクタは、軽量ケーパビリティプロテクション（LCP）機構[11]に基づいて各PMのDSMに配置されたMBを不正なアクセスから保護するハードウェア機構である。分散メモリプロテクタは図3に示すようにCCR（Current Capability Register）、MCR（Memory Capability Register）用メモリ、比較器から構成される。CCRは、プロセッサで実行中のアプリケーションオブジェクト（プロセッサのユーザモードで実行される送信あるいは受信オブジェクト）のIDを含む。MCR用メモリは複数のMCRから構成され、各MCRはDSM上のMBに1対1に対応し、そのMBへのアクセス権を持つアプリケーションオブジェクトのIDを含む。MBへのメモリアクセスが生じると、分散メモリプロテクタはMBアドレスを基に、対応するMCRをMCR用メモリの中から選択しCCRと比較する。両者が一致すればMBへの正しいアクセス、不一致であれば不正アクセスと判定する。不正アクセスを検出した場合は、実

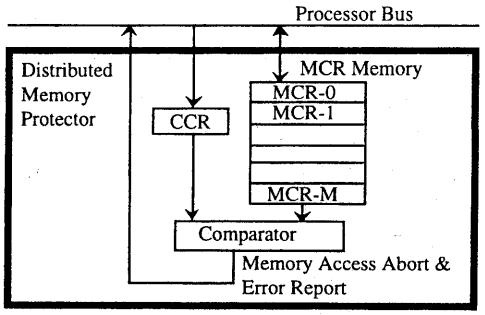

図3 分散メモリプロテクタの構成

行中のMBアクセスを中断し、エラーをプロセッサに緊急通知する。

3．4　DSM上のデータ配置と構造

共有空間エリアを図4に示すようにPM間通信エリアIPC(Inter-PM Communication Area)と処理要求エリアRQ(Processing Request Area)に分割する。

PM間通信エリアIPCは、（送信PM、受信PM）のペア対応に論理分割され、送信PMのDSMと受信PMのDSMに物理的に重複して実装される。図4のハッチ部分が実装エリアである。各IPCは、複数のMBエリアと制御データエリア(CTLD)を含む。各MBは固定長で、ユーザモードの送受信オブジェクトから直接アクセスできる。各MBは送信PMから受信PMへの一方向通信領域として使用される。一方、制御データエリアCTLDは送信PMと受信PMの両方から読み書きを行う双方向通信領域で、以下の2種類のエリアから構成される。

（1）MBステータスエリア（MS）：対応するMBの捕捉状態／解放状態を表示する。送信PMによって『捕捉状態』が、受信PMによって『解放状態』が書き込まれる。

（2）ディスクリプタエリア(MD)：複数のディスクリプタを含み、各ディスクリプタはメッセージの制御情報を含む。ディスクリプタエリア内の複数のディスクリプタは、アドレス順にサイクリックに使用される。

処理要求エリアRQは、MBやCTLDと異なって受信PM対応に論理分割されており、送信PMが異なっても受信PMが同じならば同一アドレスエリアが使用される。RQは送信PMから受信PMへの処理要求（＝メッセージ到着通知）を記憶し、送信PMから受信PMへの1方向通信領域として使う。送信DSM上のRQは通常のRAMで構成される

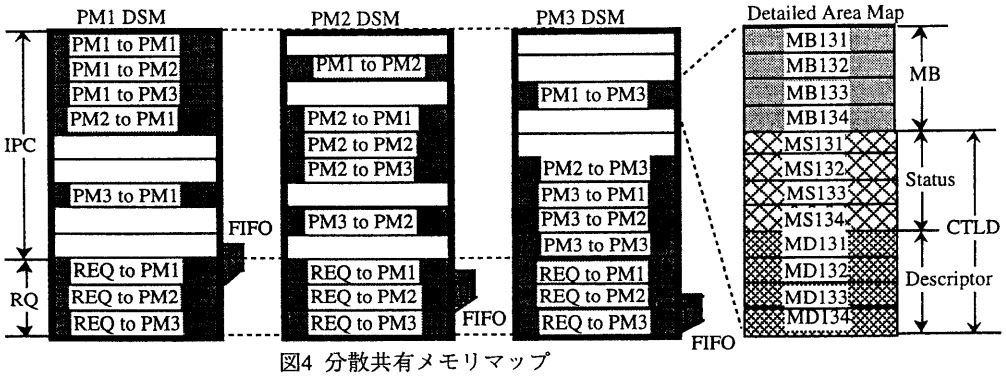

図4 分散共有メモリマップ

が、受信DSM上のRQはFIFOメモリで構成される。

送信PMから受信PMに処理要求を伝える場合、送信PMが送信DSMのRQに処理要求を書き込むと分散メモリカップラによって受信DSMの同一アドレスエリアのRQにコピーされる。その際、受信DSMのRQはFIFO構成なので古い処理要求データを保存したまま到着順に次々と新しい処理要求をFIFO内に蓄積する。このように複数の送信PMがそれぞれのDSMのRQに処理要求を同時に書き込んでも受信DSMのRQのFIFOで自動的にアクセス競合整理が行われる。受信PMでは受信DSMのRQをFIFOメモリから読み出すだけで全送信PMからの受信処理要求を検出できるので、従来必要であった送信PM対応の多数の監視点のポーリング、あるいは頻繁な割り込みによるオーバヘッドを削減できる。

受信PMでは自分宛の処理要求エリアRQを読み出してディスクリプタアドレスを知りディスクリプタMDを読み出す。MDは、送受信間でディスクリプタ読み書きの同期をとるためのOwnershipフィールドや、対応するMBアドレス等を含む。なお、MBに対応するMSのアドレスはMD内のMBアドレスから機械的な計算で求められる。MBのヘッダ部には受信オブジェクトのID（RID)が記憶されている。RIDは受信オブジェクトが所属するMESCARシステムのID（SYS-ID）、PMのID（PM-ID）、PM内のローカルIDの３階層で構成されているのでRIDを参照することにより受信オブジェクトにメッセージを送り届けることができる。

３．５　PM間メッセージパッシングの実現例

図５はPM間メッセージパッシングのタイムチャートを示したもので、送信PMはPM1、受信PMはPM3、送信PMが捕捉するMBはPM1からPM3宛のMB群（MB13X）の第4番目のMB（MB134）、対応するMBステータスはMS134、使用するディスクリプタはPM1からPM3宛のディスクリプタ群（MD13X）の第2番目のディスクリプタ（MD132）、処理要求エリアはPM3宛のエリアRQ3を使用したと仮定する。なお、送信DSMと受信DSMのエリア名を区別するため、受信側エリア名の最後にCを付加する。（例えば送信DSMのMBはMB134、受信DSMのMBはMB134C）

図５の送信オブジェクト起動(Sender Obj. Invoke)ステップでは、送信カーネルが送信オブジェクトの実行開始前に送信オブジェクトID（SID)を送信側分散メモリプロテクタのCCRに設定する。次にMB捕捉(MB Alloc.)ステップで送信カーネルは、受信オブジェクトID（RID）に対応する受信PM宛メッセージバッファMB134を捕捉し、そのMBステータスMS134に『捕捉状態』を書き込む。その結果、分散メモリカップラ経由で受信DSMのMS134Cにも『捕捉状態』がコピーされる。更に送信カーネルはMB134用のMCRにSIDを書き込み、MB134を送信オブジェクト以外のアプリケーションオブジェクトの不正アクセスから保護する。次のメッセージ書き込み（Write to MB)ステップで送信オブジェクトがメッセージを４バイト単位で順次MB134に書き込むと分散メモリカップラによって受信DSMのMB134Cに４バイトずつコピーが行われる。SENDステップでは送信カーネルはMCRの値をクリアし、以後、送信オブジェクトがMB134にアクセスできないように保護する。続いて送信カーネルは送信DSM上の

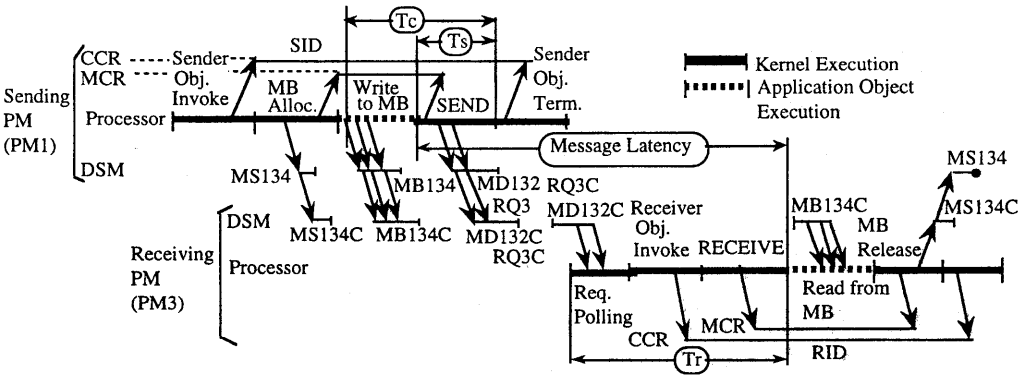

図5 PM間メッセージパッシングのタイムチャート

ディスクリプタ（MD132）にメッセージ制御情報を、受信PM対応の処理要求エリア（RQ3）に処理要求を書き込む。それらは分散メモリカップラによって、それぞれ受信DSM上のMD132CとRQ3Cにコピーされる。送信カーネルは最後に送信オブジェクト終了（Sender Obj. Term.）ステップでCCRをクリアする。

受信側では、ポーリング(Req. Polling)ステップで受信カーネルがRQ3Cを定期的に読み出す。処理要求を検出するとディスクリプタMD132C、メッセージバッファMB134C、MB134C内のRIDを順に読み出して受信オブジェクトのメッセージキューにMB134Cのアドレスを登録する。その後、受信オブジェクト起動(Receiver Obj. Invoke)ステップとRECEIVEステッにおいて送信側と同じ手順で受信側のCCRとMCRにRIDを設定し、MB134Cの保護を行う。

メッセージ読み出し(Read from MB)ステップでは受信オブジェクトはMB134Cからメッセージを読み出し、必要な処理を行う。次のMB解放（MB Release)ステップで受信カーネルはMCRをクリアし、対応するMBステータスMS134Cに『解放状態』表示を書き込むと分散メモリカップラ経由で送信側DSMのMS134にも『解放状態』がコピーされる。このコピー完了後は、いつでも送信PMでMB134を再使用できる。

3．6　MESCARの利点

（1）**カーネルのオーバヘッドの削減**：分散メモリカップラによるコピー動作はカーネルの介在なしに自律的に行われ、従来のDMA方式のようにカーネルがDMAコントローラを起動／制御する必要がない。また、分散メモリカップラの双方向コピー機能により送受間同期やMB状態通知もメモリ書き込みだけで実現でき、オーバヘッドが少ない。更に受信DSMの処理要求エリアのFIFOメモリ化により、ソフトによる競合整理を不要とし、ポーリング監視点を1箇所に集中できポーリングオーバヘッドを削減できる。

（2）**メッセージレイテンシの削減**：DSMへの書き込みと同時に分散メモリカップラでコピーを開始するので、送信オブジェクトから受信オブジェクトにメッセージを引き継ぐまでのレイテンシを短縮できる。

（3）**ハードウェア構造の簡単化**：分散メモリカップラは時間的制約の少ない簡単なコピー機能のみを持てば良く、ハードウェアDSMのキャッシュコヒーレンシ機構のような、時間的制約が厳しく複雑な制御機構を必要としない。また、分散共有メモリへのアクセスもローカルメモリと同じ速度で高速に実現できる。

（4）**高い信頼性**：分散メモリプロテクタにより、コピーデータを配置した複数のDSMを少ないソフトオーバヘッドで不正アクセスから同時に保護できる。

4．　性能評価

提案したアーキテクチャの有効性を評価するため、カーネルオーバヘッドとメッセージレイテンシを評価項目として従来方式の代表である（分散メモリプロテクタを有しない）ローカルメモリアーキテクチャのDMA方式と（分散メモリプロテクタも含んだ）MESCAR方式との間で比較を行う。

表1 メッセージパッシングのカーネルオーバヘッド

Comm. Type	Functions	Relative Overhead	
		DMA-based	MESCAR
Intra-PM	MB Allocation	0.16	0.25
	SEND	0.25	0.27
	RECEIVE	0.20	0.23
	MB Release	0.16	0.14
	APL Obj. Invoke & Term.	0.23	0.28
	Total	1.00	1.17
Inter-PM	MB Allocation	0.16	0.25
	SEND	0.25	0.27
	Sender DMA control	0.35	-
	Request Polling	-	0.23
	Receiver DMA control	0.50	-
	RECEIVE	0.20	0.23
	MB Release	0.16	0.14
	APL Obj. Invoke & Term.	0.23	0.28
	Total	1.85(1.00)	1.40(0.76)

4．1　カーネルオーバヘッドの評価

カーネルオーバヘッドを『送受信オブジェクト起動から終了までに実行する送信カーネルと受信カーネルのダイナミックステップ数の合計』と定義する。カーネルダイナミックステップについては通信網ワイド分散処理プラットフォーム用分散OSとして開発されたPLATINAカーネル[12]をもとに、両方式におけるPM内とPM間のメッセージパッシングをプログラミング／コンパイルし、マシン命令（プロセッサは68040）のステップ数をカウントした。次に、それらをDMA方式のPM内通信のステップ数で正規化してカーネルオーバヘッドを求めた。結果を表1に示す。

PM内通信ではMESCAR方式のオーバヘッドがDMA方式より17%多い。これはPM内通信ではメッセージの物理転送が不要なのでDMA方式は

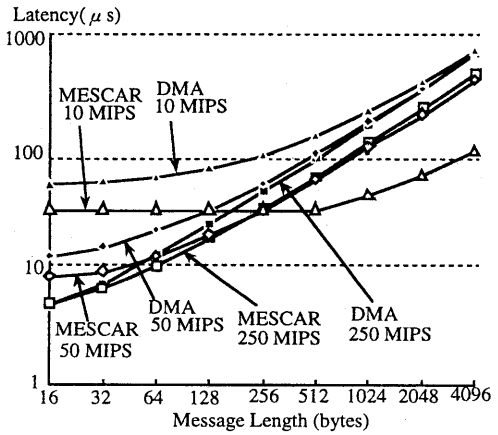

図6 プロセッサ性能を変えた場合のメッセージレイテンシの比較(156-Mb/sプロセッサ間通信路使用)

MESCAR方式は主に宛先PM対応のMB捕捉処理(MB Allocation)のオーバヘッドが少し大きいためである。

逆に、PM間通信ではMESCAR方式はDMA方式の0.76倍（=1.40／1.85）に低減されている。これは、DMA方式のDMA制御オーバヘッド（Sender & Receiver DMA Control）がMESCAR方式のMB捕捉処理（MB Allocation)等のオーバヘッドを大きく上回るためである。

4．2　PM間メッセージレイテンシの算出

PM間通信のメッセージレイテンシを図5に示すように『送信側のSEND処理開始から受信側のRECEIVE処理終了までの最短遅延時間』と定義する。DMA方式及びMESCAR方式の性能評価モデルの詳細は紙面の都合上省略し、レイテンシの計算結果（理想モデルに基づくレイテンシの下限値）の一例を図6と表2に示す。

図6はATMのAAL5プロトコルを想定し、1PMあたり156Mb/sのバンド幅を持つプロセッサ間通信路でプロセッサ性能を10、50、250MIPSと変えた場合のレイテンシを比較したものである。DMA方式ではメッセージ長の増加とともにレイテンシも増加する。これは、SEND処理によってDMA転送が開始され、転送量にほぼ比例してDMA転送遅延も増加するためである。一方、MESCAR方式では、メッセージ長に無関係にレイテンシが一定な区間（10MIPSの512バイト以下の区間）と、メッセージ長の増加とともにレイテンシが増加する区間とが存在する。MESCARでは図5に示すように送信オブジェクトによるMBへの書き込み(Write to MB)時点からDSMコピー転送が先行開始される。そのためDSMコピーがSEND終了直前までに完了すればレイテンシは、プロセッサの実行時間によって決まる一定値（図5のTs+Tr）となる。逆にDSMコピーがSEND終了前に完了しなければ、完了しきれなかった残存メッセージの転送時間がTs+Trに上乗せされるので、メッセージ長の増加とともにレイテンシも増大する。

図6でMESCAR方式の256バイト以上の領域では、プロセッサ速度が上がるほどレイテンシは逆に増加する現象が見られる。これについては、プ

表2　256バイトメッセージのメッセージレイテンシ

Processor Interconnect Bandwidth (Mb/s)	Processor Performance (MIPS)	Latency in DMA-based System (μs) (a)	Latency in MESCAR System (μs) (b)	Reduction Ratio ((b)-(a))/(a)
15.6	10	484	437	0.10
	50	444	441	0.01
	250	437	442	-0.01
156	10	93	32	0.66
	50	53	32	0.40
	250	45	33	0.27
1560	10	62	31	0.50
	50	23	7	0.70
	250	15	2	0.87

ロセッサが遅いとDSMコピー開始からSEND完了までのコピー先行期間（図5のTc）が長いためにTcの時間内に大量のコピーが行われる。その結果、SEND終了までに転送しきれなかった残存転送量が少ないのでレイテンシも小さく抑えられる。逆にプロセッサが速いとコピー先行期間Tcが短いために、SEND終了までに少量のコピーしか行われず、大量残存分の転送時間がレイテンシに加わってレイテンシを増加させるためである。

表2は我々が応用を想定している256バイト長のメッセージに対して10, 50, 250MIPSのプロセッサと15.6, 156, 1560Mb/sのプロセッサ間通信路を組み合わせた場合のレイテンシを求めた結果である。15.6Mb/sのプロセッサ間通信路ではバンド幅が小さすぎでMESCAR方式の効果が発揮されないが、156Mb/s以上ではプロセッサ速度とプロセッサ間通信路バンド幅のバランスがとれてメッセージレイテンシが大きく削減されている。

5．　あとがき

短い大量のメッセージをリアルタイムで処理するメッセージ結合アーキテクチャMESCARを提案した。今後の課題としてプロセッサ間通信路やメモリアクセスでの競合待ち合わせを考慮した性能評価モデルの構築、メッセージ送受信スループットの分析、多数のプロセッサを含む超並列システム全体の性能評価、アプリケーションプログラムによる現実的な性能評価等も必要である。

参考文献

[1]石畑宏明, 稲野聡, 堀江健志, 清水俊幸, 池坂守夫：高並列計算機AP1000のアーキテクチャ, 電子情報通信学会論文誌D-I, Vol. J75-D-I, No. 8, pp. 637-645 (1992)

[2] Li, K. and Hudak, P. : Memory Coherence in Shared Virtual Memory Systems, ACM Trans. Computer Systems, Vol. 7, No. 4, pp. 229-359 (1989).

[3] Lenoski, D. et al. : The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor, Proc. 17th Int. Symp. on Computer Architecture, pp. 148-159 (1990).

[4] 平木敬, 天野英晴, 久我守弘, 末吉敏則, 工藤知宏, 中島浩, 中條拓伯, 松田秀雄, 松本尚, 森眞一郎：超並列プロトタイプ計算機JUMP-1の構想, 情処学会研究報告, 93-ARC-102, pp. 73-84 (1993)

[5] 森眞一郎, 斎藤秀樹, 五島正裕, 富田眞治, 田中高士, Frazer, D., 城和貴, 新田博之：分散共有メモリ型マルチプロセッサ「阿修羅」の概要、情処学会研究報告、92-ARC-94, pp. 41-48 (1992)

[6] Wilson Jr, A. W., LaRowe Jr., R. P., and Teller M. J. : Hardware Assit for Distributed Shared Memory, Proc. 13th Int. Conf. on Distributed Computing Systems, pp.246-255 (1993).

[7] Wittie, L. D., Hermannsson, G., and Li, A. : Eager Sharing for Efficient Massive Parallelism, Proc. 21st Int. Conf. on Parallel Processing, Vol. 2, pp. 251-255 (1992).

[8] Delp, G. S., Farber, D. J., Minnich, R. G., Smith, J. M. and Tam, M. : Memory as a Network Abstraction, IEEE Network Magazine, Vol. 5, No. 4, pp. 34-41 (1991).

[9] Wittie, L., and Maples, C. : Merlin : Massevely Parallel Heterogeneous Computing, Proc. 18th Int. Conf. on Parallel Processing, Vol. 1, pp. 142-150 (1989)

[10] Blumrich, M. A., Li, K., Alpert, R., Dubnicki, C., Felten, E. W., and Sandberg J. : Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer, Proc. 21st Int. Symp. on Computer Architecture, pp. 142-152 (1994).

[11] 山田茂樹, 丸山勝己：オブジェクト指向システム用軽量ケーパビリティプロテクション方式, 情処学論, Vol. 34, No. 9, pp. 2037-2047 (1993).

[12] Kubota, M., Maruyama, K., Tanaka, S., Osaki, K., and Yamada, S.：Distributed Processing Platform for Switching Systems : PLATINA, Proc. 14th International Switching Symposium, Vol. 1, pp. 415-419 (1992).