ハイパフォーマンス
コンピューティング 55−13
（1995. 3. 9）

MPI/DE – 並列計算機 Cenju-3 上の MPI ライブラリ – の性能評価

小西 弘一　神舘 淳†　加納 健　Christopher Howson　高野 陽介[1]

NEC C&C 研究所　†(株) ソフテック

MPI/DE は 並列計算機 Cenju-3 用の MPI ライブラリであり、Mach microkernel 上の並列実行環境 DenEn の上で動作する。MPI/DE の目標は、フロー制御、進行の保証などの MPI の仕様を完全に満たすこと、高い通信性能を実現すること、およびマルチタスク環境での使用に対応することである。本稿では、MPI/DE の基本的な通信性能、実装方式とその効果について述べる。一対一通信で、レイテンシ 90 μsec、スループット 19 Mbytes/sec の性能が得られた。シングルタスクを条件とし、新たに開発したハードウェアを使う場合では、レイテンシは 40 μsec になった。

Implementation and Performance of MPI on Cenju-3

Koichi Konishi　Jun Kohdatte†　Yasushi Kanoh

Christopher Howson　Yosuke Takano

NEC C&C Laboratories　†*SofTek*

We report on the development and preliminary measurements of MPI/DE, an implementation of MPI on Cenju-3. It has been implemented as a part of DenEn, a Mach microkernel-based environment for parallel processing on Cenju-3. The goals of MPI/DE are high communication performance, compliance with the MPI standard, and correct behaviour in DenEn's multitasking environment. We present basic performance numbers, along with implementation decisions and some of their consequences. We have achieved a short message latency of 90 μsec, and a large message throughput of 19 Mbytes/sec. We also experimented with a modified network interface and were able to reduce latency to 40 μsec when in singletasking mode.

[1]Email: {konishi,kohdatte,kanoh,howson,takano}@csl.cl.nec.co.jp

1 はじめに

MPI/DE は、Cenju-3[4] のためのメッセージ通信ライブラリであり、MPI[1] のフルセットをユーザーに提供する。目標は、以下の全てを同時に実現することである。

- 高い通信性能
- MPI としての正しい動作
- マルチタスク環境への対応

実行環境は DenEn[8] という Mach microkernel に基づく OS であり、タスクの起動など、プロセス管理の処理は DenEn に依存している。一方、通信機能についてはユーザーレベルのライブラリの中で実装した。Mach には NORMA IPC というプロセッサ間通信機能があるが、これは MPI とは用途が異なり、オーバーヘッドも大きいため、これを利用してその上位に MPI を作ると遅くなると予想した。そこで、Cenju-3 のネットワークインターフェース (NIF) をデバイスとしてユーザーアドレス空間にマップし、ユーザーレベルからこれを利用する方法を採用した。

本稿では MPI/DE の、通信に直接関わる部分の実装方式と、基本的な通信性能の評価結果について述べる。最初に MPI、Cenju-3 とそのネットワークインターフェース、DenEn のそれぞれについて簡単に説明する。次に MPI/DE の基本的な通信性能を示した上で、実装上の課題と、それぞれの解決策、およびそれらが性能に与える影響を示す。

2 MPI

MPI は、MPI forum が定めた、アプリケーションのためのメッセージ通信インターフェースの標準仕様である。従来のメッセージ通信ライブラリと比べると、通信モードが多様である、バッファレイアウトの自由度が高い、集団通信が豊富である、プロセスグループを自由に設定できる、システムが複数の通信コンテクストを提供する、などの特徴がある。

1 回の一対一通信は、送信側が MPI_Send を呼ぶことによりメッセージを送出し、受信側が MPI_Recv によってメッセージを受信することによって成立する。どの MPI_Send と MPI_Recv が対応するかは、発信者の ID、タグなどにより指定することができる。送信と受信の対応を決める処理はかなり重い処理であり、MPI/DE の最小レイテンシの大きな部分を占めている。

3 Cenju-3 および NIF ハードウェア

Cenju-3 は分散メモリ型並列計算機である。汎用マイクロプロセッサ (VR4400) とローカルメモリ (最大 64Mbytes) からなるプロセッサエレメント (PE) を 4x4 のスイッチによる多段接続網で接続する構成を持つ。最大構成での PE 数は 256 台、ネットワークのポート間のスループットは 40 M bytes/sec である。また、UNIX ワークステーション 1 台 (ホスト) を上記のネットワーク経由で接続する。

各 PE は通信のための専用ハードウェア (ネットワークインターフェース、NIF) を持つ。NIF はメッセージ通信と DMA 通信の両方をサポートしている。NIF のメッセージ通信では、届いたデータが受信側であらかじめ設定された受信バッファに書き込まれる。DMA 通信では、届いたデータは送信側が指定した物理アドレスにあるメモリ領域に書き込まれる。NIF と CPU の間ではキャッシュコヒーレンシは保証されない。

メッセージ、DMA とも、ヘッダとデータが分離された構造を持っており、ヘッダにデータ位置を示すアドレスを持たせている。送信側にはメッセージ、DMA 兼用の送信ヘッダキューを 2 本、受信側にはメッセージ用の受信バッファ (ヘッダ用とデータ用に分かれている) を 2 本、DMA 用の受信ヘッダバッファを 1 本、メモリ上に設けることができる。

NIF は物理アドレス空間中にマップされたレジスタを持ち、このレジスタにヘッダキュー、受信バッファその他の領域の位置を示すアドレス、キューやバッファの次の読み出し位置、次の書き込み位置を示すアドレスなどの値を保持している。MPI/DE では NIF のレジスタの一部、キュー、バッファをさらにユーザータスクの仮想アドレス空間中にマップして、直接読み書きすることにより通信を行っている。

4 DenEn

DenEn は Cenju-3 上の並列プログラムの実行環境であり、Mach microkernel とその上のいくつかの小規模なサーバ、およびホスト側の UNIX 上で走るデーモンで構成されている。ユーザープログラムは、Mach タスクとしてマルチタスク環

境で実行される。スレッドライブラリ (cthreads) を含めた Mach カーネルインターフェースを直接利用できる他、ディスク I/O のための UNIX システムコールをホスト上のディスクに対して実行することができる。

5 MPI/DE の通信性能

以下に示す測定結果は、14PE 構成の Cenju-3、CPU クロック 75MHz という条件で行った。

5.1 一対一通信

一対一通信の性能の測定では、2PE 間で一定の大きさのメッセージを繰り返し往復させ、1 往復当たりの時間の半分を求めて片道の所要時間とみなした。メッセージの送信には MPI_Send, 受信には MPI_Recv を用いた。メッセージ長 8bytes から 800 Kbytes について測定した。

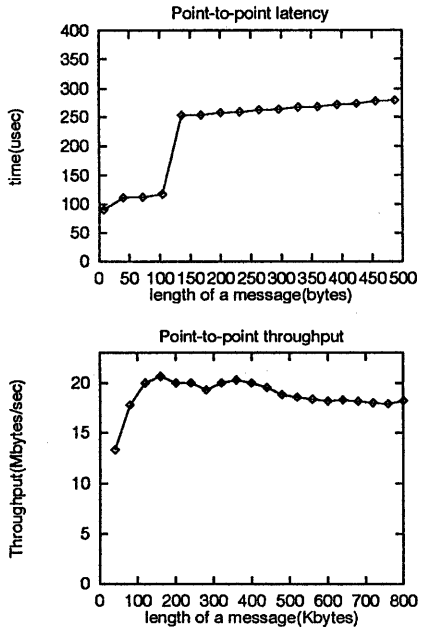

図 1: 一対一通信の性能

図1 に測定結果を示す。最小レイテンシは 90 μsec(メッセージ長 8bytes)、最大 スループットは 19 Mbytes/sec (メッセージ長 200-400Kbytes) であった。

レイテンシの不連続な変化はプロトコルの違いによる。メッセージ長 128 bytes 未満は NIF メッセージで受信側とは非同期に、128 bytes 以上は受信側と同期してから DMA 転送により送信している。

5.2 集団通信

図 2 に MPI_Barrier(バリア同期), MPI_Bcast (マルチキャスト、1-1000words), MPI_Scatter(データの分配、1280bytes) の性能の測定結果を示す。現在、集団通信は、すべて一対一通信の上に実装している。MPI_Barrier と MPI_Bcast は最大中継回数がプロセス数の log に比例する通信パターンを用いて、MPI_Scatter は分配元が他のプロセスと順に通信することによって実現している。

6 通信機能の実装方式

実装上の課題は、物理アドレスを用いて転送を行う NIF ハードウェアを用いて、マルチタスクに対応すること、および、microkernel 上のライブラリであることを考慮して、特殊なカーネルコールの導入を避け、ほとんど全ての処理をユーザーレベルで処理を行うことであった。

これらを解決する際問題となったのは、データのコピー、割り込み処理、キャッシュ操作、およびメモリ保護の 4 点であった。以下、これらの処理について詳述する。

6.1 データのコピー

現在、MPI/DE は送信側と受信側の両方にメモリマップを固定したバッファ (MPI バッファ) を設け、両者の MPI バッファ間のデータ転送を NIF によって行っている。ユーザーのデータは、送信側のユーザーバッファから MPI バッファへ、また受信側の MPI バッファからユーザーバッファへ、計 2 回コピーされる。

コピーをする必要があるのは、Cenju-3 の NIF が remote DMA 書き込みの書き込み先アドレスとして物理アドレスを要求することに起因する。MPI/DE ではユーザーレベルから NIF を操作しており、書き込み先アドレスの指定もユーザーレベルから行う。しかし、仮想アドレスに対応する物理アドレスを知る方法は、通常ユーザーレベルには提供されていない。また、特別なカーネルコールを設けて対応を知ることができたとしても、物理アドレスと仮想アドレスの間のマップは、一般

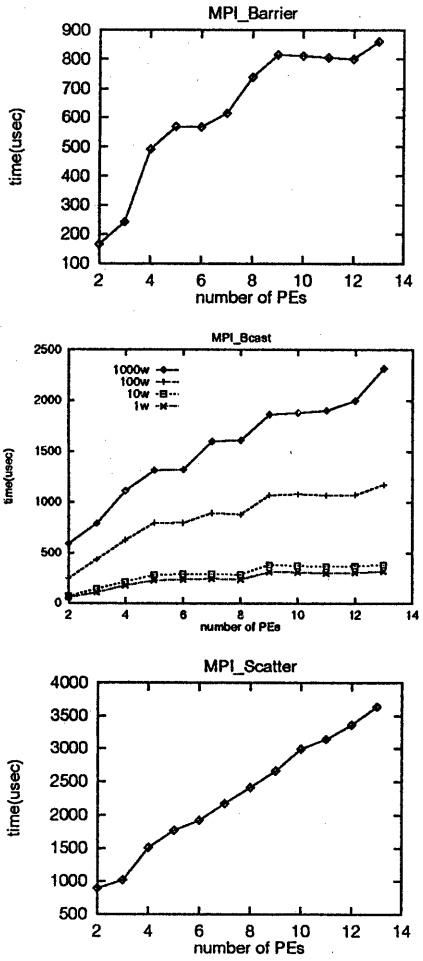

図 2: 集団通信の性能

にページングによって実行時に変化する。

これを防ぐため、今の版ではあらかじめ対応を固定した領域を設け、これをMPI バッファとして用いている。ユーザーはこの領域内の仮想アドレスをDMA の転送元、転送先アドレスとして指定したうえで、送信の起動のためにカーネルコールを行う。このコール内でカーネルが、上述の固定された対応に基づいて仮想アドレスを物理アドレスに変換する。

このコピーは、大きなメッセージについては転送を律速する。実験のために、このコピーを行わない版(当然この版では正しく通信はできない)を作って転送所要時間を測定した。図3にコピーを行

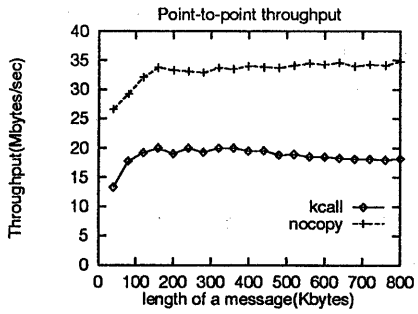

図 3: コピーによる転送の律速

う版(kcall)と行わない版(nocopy)の性能比較を示す。メッセージ長 800Kbytes の場合、通常の版のスループット 17 Mbytes / sec に対し、コピーを行わない版では 35 Mbytes/sec 相当の時間で通信が終了した。

コピーによる律速を回避する手段の検討も行っている。Machでは、1ノード内でのメモリコピーは、しばしばページマップの書換えによって遅延実行する。これにならって、MPI バッファとユーザーバッファ間のコピーをページマップの書換えで済ませることができる。

6.2 割り込み処理

メッセージの到着の検出は普通、割り込みかポーリングのどちらかで行う。MPI/DE では、Mach のマルチスレッド、イベント通知機構、タイマー機構を用いて割り込み相当の処理を実現し、これをポーリングと併用している。

以下、割り込み処理が必要な理由、DenEn での割り込み処理と MPI/DE で用いた割り込み相当の処理、および MPI/DE で用いた方法のオーバーヘッドを順に説明する。

6.2.1 必要性

割り込みは、まず、通信処理の進行を保証するために必要である。割り込みなしでは、MPI のノンブロッキング通信の進行を保証することが難しい。ノンブロッキング通信では、通信を開始してからすぐにスレッドをユーザーのコードに返さなければならず、以降の通信処理には別のスレッドが必要である。ユーザーレベルで割り込み処理ができれば、メッセージの到着時に割り込みハン

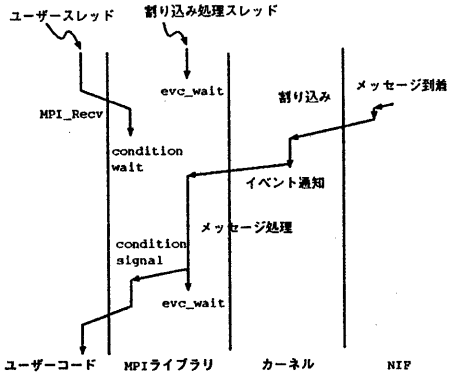

図 4: 割り込みスレッドによる受信処理

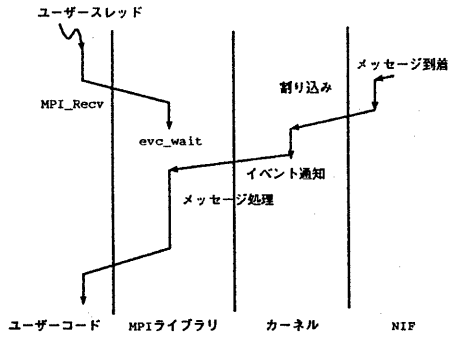

図 5: ユーザースレッドによる受信処理

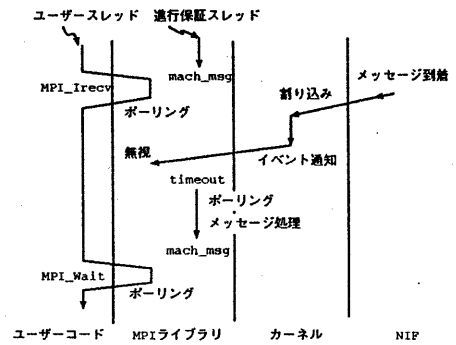

図 6: 進行の保証

ドラを起動して、そのスレッドにより通信処理を進めることができる。

割り込みが必要なもう一つの理由は、マルチタスク実行への対応である。マルチタスク実行時にはビジーウェイトは避けなければならないが、ポーリングのみの実装では、ブロッキング通信においてメッセージの到着を待つ際にはどうしてもビジーウェイトすることになってしまう。

6.2.2 DenEn における実現

ユーザーレベルの割り込みハンドラを実現するために Mach が用意しているカーネルコールは evc_wait() である。これにより、ユーザースレッドはカーネルからのイベント発生通知を待つことができる。カーネルはイベントが発生したとき、evc_wait() によってそのイベント発生を待っているスレッドがいれば、そのスレッドをスケジュールする。イベント発生時に実行中のスレッドの処理を止めてハンドラを起動するわけではないので、割り込み処理とは異なる。割り込みに相当する処理を実現するには、割り込み処理専用のスレッドを設けて、このスレッドに evc_wait() でイベント待ちをさせる必要がある (図 4)。

しかし、上述の方法で実現した割り込み処理はレイテンシを大きく悪化させる。一つの要因は、カーネル内での割り込みハンドラの起動に伴うオーバーヘッドである。もう一つは、同じユーザーレベルで複数のスレッドを用いることにより、それらの間のコンテクスト切り替えのオーバーヘッドが発生することである。

ユーザースレッド間のコンテクスト切替えのオーバーヘッドを避けるため、MPI/DE ではユーザーのスレッドが直接 evc_wait() によりメッセージ到着を待つようにしている (図 5)。evc_wait() はこれを呼び出したスレッドをブロックすることがあるので、ノンブロッキング通信の場合には、ポーリングを用いる。しかし、ユーザースレッドだけでは処理の進行を保証できないので、別のスレッドを設け、これを一定時間間隔で起動して、到着メッセージのポーリングと処理を行わせている (図 6)。一定時間間隔での起動は、Mach IPC のタイムアウトを利用して実現した。

6.2.3 オーバーヘッド

受信処理を、割り込みスレッドで行う版 (intr)、ユーザースレッドが evc_wait で行う版 (evc_wait)、

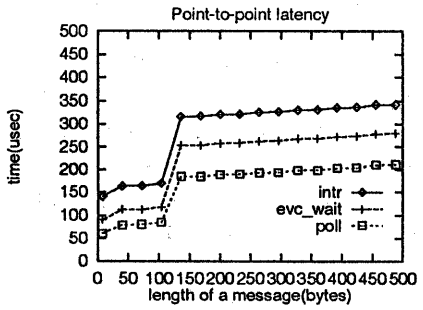

図 7: 割り込みのオーバーヘッド

ユーザースレッドがポーリングで行う版 (poll) のレイテンシの比較を図 7に示す。ポーリング版はカーネルにも割り込みを掛けていない。最小レイテンシはそれぞれ 140, 90, 60 μsec であった。メッセージが大きい場合のスループットには違いが見られなかった。

マルチタスク環境でのポーリング版の利用には問題があるが、明示的にシングルタスク実行を保証する機能があれば、ポーリング版の高性能を生かすことができる。

進行保証スレッドの有無は、起動間隔を 1 秒とした場合、レイテンシには影響を与えなかった。

6.3 ユーザーレベルでのキャッシュ操作

Cenju-3 の NIF ハードウェアによる DMA 書き込みの際には、キャッシュの整合性が保証されない。正しいデータ転送を保証するには、NIF ハードウェアがアクセスするメモリ領域のキャッシュをプログラムで操作して整合性を保つ必要がある。

CPU のキャッシュ操作用命令は特権命令なので、ユーザーレベルでは利用できず、これを使うにはカーネルコールが必要になる。送信側ではもともと NIF ハードウェアに送信を開始させるためのカーネルコールが必要なので、その際同時にキャッシュフラッシュも行うことができる。しかし受信側では、キャッシュフラッシュのためだけにカーネルコールを新たに行わねばならず、レイテンシが著しく悪くなる。これを避けるため、受信側では以下に述べる方法でキャッシュの操作を行った。

Cenju-3 の CPU VR4400 のキャッシュはダイレクトマップであり、2 次キャッシュサイズは 1 Mbytes である。したがって、あるアドレスから物理アドレス上で 1Mbytes 離れたアドレスを読み出すと、最初のアドレスに対するキャッシュは後者のアドレスのキャッシュによって入れ換えられる。キャッシュの入れ換えはキャッシュライン長 (16 ワード) を単位として起きる。したがって、連続した大きな領域のキャッシュを入れ換えるには、全域に渡って 16 ワードおきに 1 ワードを読み出してやればよい。

この方法では、キャッシュの整合性を保つためには必要のないメモリアクセスが起きることがあり、大きなメモリ領域のキャッシュ操作を行うのには向かないが、小さなメッセージの受信時にはカーネルコールによる操作より高速である。MPI/DE では 8Kbytes を単位として転送とキャッシュ操作を行っている。このサイズでは、ユーザーレベルからのキャッシュ操作の方が高速である。

6.4 DMA からのメモリ保護

Cenju-3 の NIF ハードウェアに送信を開始させるために、カーネルコールを行っていることは既に述べた (6.1節)。この際起きるユーザーモードからカーネルモードへのコンテクストスイッチのオーバーヘッドは、レイテンシの大きな部分を占めている。これを取り除くことを狙って、ユーザーレベルから直接 NIF ハードウェアに送信を開始させる実験を行った。

カーネルコールによらず NIF を起動する際の問題点としては、DMA 書き込みからのメモリ保護、仮想アドレスと物理アドレスの変換、およびキャッシュ操作が挙げられる。

NIF ハードウェアの現在の版はカーネル空間にも、別のアプリケーションのプロセスのメモリ空間にも自由に書き込みができてしまう。このためカーネルコールは、不正な領域への書き込みが起きないように、書き込み先アドレスなど、ユーザーが書き込んだ NIF ハードウェアへの指示内容をチェックしている。

このようなメモリ保護を保ったまま、ユーザーレベルから NIF ハードウェアを利用できるようにするため、新しい NIF ハードウェアの開発を行った。Cenju-3 の NIF ハードウェアは FPGA でできているので、ROM の交換だけで異なる機能を容易に実現することができる。

新しい NIF ハードウェアでは、2 本の送信側ヘッダキューにカーネル用とユーザー用の区別を設ける。ユーザー用のヘッダキューの中の宛先 PE

番号と DMA 書き込み先アドレスを示すフィールドは、あらかじめ OS が設定したマスクによって修正されてから効果を持つ。これにより、OS はユーザーレベルからの DMA 書き込み先範囲を特定の PE 群の特定の物理アドレス領域に限ることができる。ユーザーレベルから指定できる DMA 書き込み先アドレスは、下位 18 ビットであるため、カーネルを介在させずにユーザーレベルから DMA 書き込みができる領域の大きさは、256Kbytes である。このマスク操作によって、実質的に仮想アドレスから物理アドレスへの変換も同時に行うことができる。

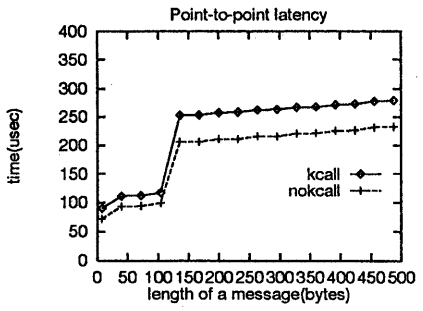

図 8: 新版 NIF によるレイテンシ

従来カーネルコールが行っていた、送信側 MPI バッファのキャッシュフラッシュは、受信側バッファのキャッシュ操作と同じく、1Mbytes 離れたアドレスをアクセスしてキャッシュラインを入れ換えることにより行うようにした (6.3節参照)。

新版 NIF を用いる版での一対一通信の性能を図8に示した。kcall がカーネルコールによって、nokcall がユーザーレベルから新版 NIF を起動した場合の結果である。最小レイテンシが 20μsec 短縮され、70μsec となった。さらに、新版 NIF を使うポーリング版の処理系 (6.2.3節) では、40 μsec であった。大きなメッセージのスループットは変わらなかった。

7 マルチタスク環境への対応

マルチタスク環境に対応するためには、ビジーウェイトの回避、複数のタスクによる NIF ハードウェアの共有、およびページングが問題となる。ビジーウェイトにはすでに対処している (6.2節) ので、ここでは他の 2 点について簡単に述べる。

NIF ハードウェアが用いるキューおよびバッファを物理アドレス空間中のどこに置くかは OS が随時設定することにする。すなわち、マルチタスク実行時には、タスクスイッチの際に設定し直せば、タスクごとに異なるキューとバッファを設けることができる。

NIF ハードウェアがヘッダキューをアクセスしている最中に、ヘッダキューの先頭アドレスを示すレジスタを書き換えると、正しい動作は期待できない。NIF ハードウェア動作中のレジスタ書換えを避けるため、タスク間で 1 本のヘッダキューを共有する。ヘッダキューへの書き込み操作はカーネルコールで行っているので、複数のタスクからの書き込みの競合によって問題が起きることはない。

また、DenEn がページングを行う場合、ページアウトされているページに DMA による書き込みを行う可能性が生じる。MPI/DE では、特定の領域を MPI バッファと定めて、MPI バッファ間で DMA 転送を行っている。したがって、ページアウトを防ぐ必要があるのは、MPI バッファだけであり、これをメモリ中にロックすればページングは問題にならない。

ページマップの書き換えによりメモリコピーを回避する場合 (6.1節) には、NIF をカーネルから利用するので、任意の物理アドレスが DMA 書き込み先となる可能性がある。この場合は処理単位がページであり、また、カーネルが行う処理なので、その都度ロックしてもあまり重い処理にはならないと予想している。

8 関連研究

ユーザーレベル IPC をサポートするネットワークインターフェースハードウェアに関する研究では、通信コプロセッサ Elan[6]、SHRIMP[7]、AP1000+[3] がある。Elan を利用している CS-2 上のメッセージ通信ライブラリ PVM および PARMACS では、UNIX プロセス間の通信を レイテンシ 78 μsec, スループット 44Mbytes で実現している。

ANU による AP1000 上の MPI[2] は、一対一通信の性能は 332μsec, 2.85 Mbytes / sec であるが、集団通信に関しては専用のネットワークの利用と OS の修正により、任意のサイズのグループについて集団通信の所要時間を一定以下に押えている。

IBM SP1/SP2 上の MPI[5] は SP1 でレイテンシ 30μsec(メッセージ長 0bytes), スループット約 9Mbyte/sec、 SP2 では 40μsec, 35Mbytes/sec を実現している。実行環境はマルチタスクであるが、MPI ライブラリを利用できるのは 1PE あたり 1 プロセスだけである。このレイテンシはポーリングによるものであり、ユーザーレベルの割り込み処理を行う場合のレイテンシは 200 μsec まで増大している。 MPI/DE での割り込み処理に比べて遅い理由としては、 UNIX と Mach microkernel でコンテクスト切り替えのコストが違うことが考えられる。

9 まとめ

MPI/DE は、高い通信性能と MPI としての正しい動作を、ほとんどの処理をユーザーレベルで行って実現している。メッセージ到着イベントの処理は、evc_wait とポーリングを併用して検出し、原則としてユーザースレッドによって行う。通信の進行は、定期的に走ってポーリングする別スレッドにより保証する。キャッシュの操作は、メモリアクセスによるキャッシュラインの入れ換えにより行う。新版 NIF ハードウェアによる、まったくカーネルコールを行わない実装についてもその動作と効果を確認した。以上の実装方式は、マルチタスク環境でも用いることができる。シングルタスク環境を想定すれば、より小さなレイテンシで済ませることができる。

MPI/DE は、一対一通信と集団通信のみのサブセットが、既に DenEn システムとともに国内外のいくつかの大学に対して、フリーウェアとして配布中である。フルセットについても、同様に配布を行う予定である。

今後の課題としては、6.1節に述べたコピー回避方式の検証、マルチタスク化の実現、Cenju-3 の外部との通信の実現がある。

謝辞

MPI/DE の開発に当たっては、NEC 情報システムズの川戸勝史氏、鷲谷彰一氏にご尽力いただきました。ありがとうございました。また、MPI/DE 実装方式に関しては、同僚である DenEn 開発グループ、Cenju-3 ハードウェアグループから多数の有益な意見を頂きました。改めてお礼申し上げます。

参考文献

[1] Message Passing Interface forum, "MPI: A Message-Passing Interface Standard," May 1994

[2] David Sitsky, David Walsh, Chris Johnson, "Implementation and Performance of the MPI Message Passing Interface on the Fujitsu AP1000 Multicomputer," August 1994, Accepted to ACSC'95

[3] 石畑宏明、堀江健志、清水俊幸、林憲一、小柳洋一、今村信貴、白木長武、 "AP1000+: デザインコンセプト"、情処研究会報告 ARC 107-20、 1994 年 7 月

[4] 広瀬哲也、加納健、丸山勉、中田登志之、 "並列コンピュータ Cenju-3 のアーキテクチャ"、情処研究会報告 ARC 106-16、 1994 年 7 月

[5] Hubertus Franke, Peter Hochschild, Pratap Pattnaik, Jean-Pierre Prost, Marc Snir, "MPI on IBM SP1/SP2: Current Status and Future Directions," Proceedings of the 1994 Scalable Parallel Libraries Conference, IEEE Computer Society Press, October 1994

[6] Jon Beecroft, Mark Homewood, Moray McLaren, "Meiko CS-2 interconnect Elan-Elite design", Parallel Computing, Vol. 20, pp1627-1638, 1994

[7] Matthias A. Blumrich, Cezary Dubnicki, Edward W. Felten, Kai Li, Malena R. Mesarina, "Virtual-Memory-Mapped Network Interfaces", IEEE Micro, pp21-28, February 1995

[8] 高野陽介、Christpher Howson, 荒木宏之、菅原智義、小西弘一、小長谷明彦、 "Mach マイクロカーネルをベースとした並列 OS DenEn の実現"、情処 50 回全国大会 (予定)、 1995 年 3 月