ハイパフォーマンス
コンピューティング 67－13
（1997. 8. 19）

Myrinet 上のユーザーレベル通信の比較報告

小西 弘一　Angelos Bilas†　荒木 壮一郎　Czarek Dubnicki†
Jan Edler‡　James Philbin‡[1]

NEC　†プリンストン大学　‡NEC 北米研究所

ハイエンド PC を Myrinet を用いて接続した PC クラスタ上で、三種類のユーザーレベル通信ライブラリ、Active Messages(AM)、Illinois Fast Messages(FM), および VMMC の、機能、実装方式、性能を評価した結果を報告する。また PM についても機能と実装方式を比較する。通信モデルは AM は RPC, FM と PM はメッセージ、VMMC は遠隔メモリ書き込み、とそれぞれ異なる。いずれも小メッセージに対して極めて小さい通信遅延を示すが、バンド幅では VMMC が、PIO とメモリコピーを排して DMA のみを用いたことにより、FM, AM に大きく勝っている。

User-Space Communication: A Comparative Study

Koichi Konishi　Angelos Bilas†　Soichiro Araki　Czarek Dubnicki†
Jan Edler‡　James Philbin‡

NEC　†Princeton University　‡NEC Research Institute

In this paper we investigate the functional, design, and performance aspects of three user-space communication subsystems, AM, FM, and VMMC on a cluster of high-end PCs connected with Myrinet network. In addition, PM is also put to the comparison of functional and design aspects. AM implements a request/reply model, FM and PM are more like the traditional message passing, and VMMC uses memory mappings between virtual address spaces for direct data deposit. All three communication systems have very low latency for small messages, but VMMC achieves higher bandwidth than AM and FM by avoiding PIO and data copying.

[1]Email: {konishi,soichiro}@ccm.cl.nec.co.jp, {bilas,dubnicki}@cs.princeton.edu, {edler,philbin}@research.nj.nec.com

1 はじめに

可塑性の高いインターフェースを持つ市販 Gigabit ネットワーク Myrinet を用いて、これまでに様々な通信ライブラリが実現されている。本稿では、Active Messages(AM)[1], Illinois Fast Messages(FM)[6], および Virtual Memory-Mapped Communication(VMMC)[5] の機能と性能の両面の比較を報告し、さらに PM[3] についても機能比較を行う。性能比較は、必要に応じて筆者らが移植を行い、すべてのライブラリを同一実行環境 (PC / AT 互換機と LINUX) で実行した結果による。

本稿の構成は以下の通りである。次節で Myrinet の概要を述べる。3節では、各ライブラリの特徴と実装方式の違いを述べる。4節では性能測定の詳細と結果を示す。そして最終節で結論を示す。

2 Myrinet

Myrinet はローカルエリアネットワーク (LAN) もしくはシステムエリアネットワーク (SAN) である。SAN はクラスタや一匡体に収容されるマルチプロセッサシステムなどのシステム構築に用いるネットワークを指す。全二重の双方向チャネルを提供し、チャネルの一方向のバンド幅は 1.28Gbit/s (160MB/s) である。固有のトポロジーはなく、4,8, ないし 16 ポートのスイッチを任意に接続して用いる。スイッチ内はクロスバである。各リンクはフロー制御されている。ルーティング情報は、発信元が各パケットにヘッダとして与える。

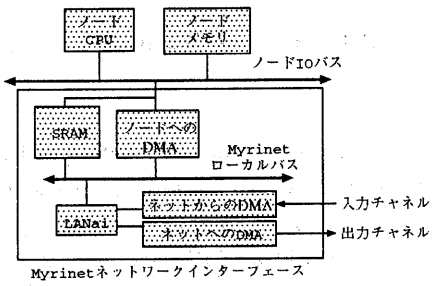

図 1: Myrinet ネットワークインターフェース

図 1にネットワークインターフェースの構成を示す。LANai と呼ばれるプロセッサと、SRAM、3 つの DMA エンジン、およびこれらを結ぶローカルバスからなる。LANai のレジスタの一部および SRAM の全体は、ノードのプロセッサの物理アドレス空間にマップされており、ノードのプロセッサが programmed IO (PIO) として直接読み書きすることができる。DMA エンジンは LANai によって起動され、うち一つは、ノードのメモリと LANai SRAM の間の双方向データ転送を、残りの二つはそれぞれ SRAM からネットワークへ、およびネットワークから SRAM へのデータ転送を行う。LANai はネットワークに対し DMA によらず直接データを送受信することもできる。

3 Myrinet によるユーザーレベル通信

各通信ライブラリはそれぞれ独特の仕様を持つ。このうち、ユーザープログラムから見える、インターフェースの仕様は、ライブラリ固有のものであるが、内部の実装方式は、ライブラリの特定の版の属性であり、同じライブラリの別の版では全く違っていてもおかしくない。

本節では各ライブラリの仕様の違いを述べる。表1に各ライブラリの仕様の主な違いを示す。最初に、ユーザーから直接見える機能の違いを説明し、その後今回使った版における実装方式の違いを述べる。以下、AM, FM, VMMC について機能仕様と実装方式における違いを述べる。本節の記述は、AM は、LANai Active Messages release 0.99 pl 14、FM は、FM2.0 に基づく。また、本節に限り、もう一つの通信ライブラリ PM[3] についても言及する。

3.1 機能仕様

以下の点は全ライブラリに共通している。送信順に従って配送が行われること、Myrinet が正しく動作する限り、メッセージの到着が保証されていること。それ以外では、大きな違いがある。以下、各ライブラリの API、複数のチャネルのサポート、フロー制御の有無について述べる。

3.1.1 API

AM の通信は、基本的に遠隔手続き呼び出し (RPC) である。要求と応答の二つのメッセージが必ず対を成す。各メッセージは、常にメッセージハンドラのアドレスを持つ。受信側はポーリングを行い、そこで到着したメッセージのハンドラの実行を行う。

FM はストリームインターフェースという独特

表 1: 通信ライブラリの機能および実装の比較

	API	フロー制御	複数チャネル	構成検出	フロー制御	Myrinet ノード間転送
AM	RPC	あり	不可	ルーティング自動記述	トークン	両側 PIO または両側 DMA とコピー
FM	メッセージ	あり	不可	スイッチ単位記述	トークン	送信側コピーと PIO 受信側 DMA とコピー
VMMC	RMW	不要	可	ノード単位記述	なし	両側 DMA(送信側小メッセージ PIO)
PM	メッセージ	可	あり	リンク単位記述	改良 ACK/NACK	両側 DMA

の API により、一つのメッセージを任意の個数の任意の長さのデータ片から構成することができる。受信側でも、一つのメッセージを任意の長さの任意の個数の断片に分けて受けとることができる。受信側はポーリングを行い、メッセージの先頭の到着が発見された時点でメッセージハンドラを起動する。メッセージハンドラはメッセージをユーザーバッファに取り出す処理を行うが、必要なデータがまだ届いていなければ、ブロックし、次回以降のポーリングの際に再起動され、処理を継続する。ハンドラの中では、送信ができない場合があるため、確実に返答を返すためには、ハンドラの外に処理を渡す必要がある。

VMMC による通信は、メッセージ通信ではなく、遠隔メモリ書き込み (RMW) である。あるプロセスが公開した仮想アドレス空間中の領域に、別のプロセスがデータを転送する。後者のプロセスは転送以前に公開領域を「輸入」する必要がある。データ受信の際には、受信側による明示的な受信操作は不要である。公開領域ごとに通知 (notification) ハンドラを指定することができ、送信側は、データ転送時にハンドラを起動するかどうかを選択できる。

PM の API の特徴は、メッセージが入ったシステムバッファをユーザーに公開し、アクセスを許すことである。データをシステムバッファから改めてユーザーバッファにコピーするかどうかの判断は、ユーザープログラムが行う。送信側は、送信用システムバッファの割り当てを受け、このバッファにデータを書き込んで送信する。受信側はシステムバッファに入ったメッセージを受けとり、使用済のシステムバッファを返却する。

複数チャネル

AM, FM は、同一ノード上の複数のプロセスが同時に利用することはできない。VMMC では、同一ノード上の複数のプロセスが、それぞれ異なる公開領域を用いれば、並行して通信を行うことができる。PM は複数のチャネルをサポートし、またチャネルのコンテクストを退避復元する手段を提供する。

フロー制御

AM, FM, PM はいずれもフロー制御を行う。VMMC は RMW であり、送信側は受信側が特定の処理を行うのを待つ必要がないため、フロー制御も不要である。

3.2 実装方式

実装方式に関しては、いずれもユーザーレベル通信を行う点は共通している。すなわち、ユーザープロセスがシステムコールを使わず直接 Myrinet にデータの送信を依頼する。また、割り込みによるメッセージ到着の通知を行わず、ユーザープロセスがポーリングでメッセージの到着を検出する。これにより、コンテクスト切り替えのオーバーヘッドを回避し、主プロセッサをデータ転送処理から解放している。

どのライブラリも LANai SRAM とネットワークの間の転送を主に DMA エンジンで行う。これは、LANai は多くのイベントに同時に対処する必要があり、特定のイベントの発生を待ったり、長くかかる処理に専念する余地がないためである。

以下、ライブラリごとに大きく異なる、ネットワーク構成検出、フロー制御、ユーザーバッファ・LANai SRAM 間のデータ転送について述べる。

ネットワーク構成検出

どのライブラリも、ネットワーク構成を記述したファイルをユーザーが与える必要があるが、記述形式にはライブラリにより大きな違いがある。AM では、ルーティングをすべて記述する必要があるが、mapper というツールが用意されており、これが自動的に構成を検出して上記の記述を生成する。FM では、スイッチを単位とする構成記述を与える必要がある。これは比較的容易に書けるが、自動検出機能はサポートされていない。VMMC は、ネットワーク中のノード名リストをユーザーが与えると、ネットワーク構成の詳細を自動的に検出する。PM では、リンクを単位とする構成記述ファイルをユーザーが指定する。自動検出機能はサポートされていない。

フロー制御

AM と FM は、各プロセスが通信に参加する全プロセス数に比例した (したがって全システムではプロセス数の自乗に比例する) 量のバッファを用いる単純なトークン制御を用いる。PM は、改良版 ACK/NACK 制御 [3] によって、プロセス数に比例するスケーラブルなフロー制御を行う。

ノード・Myrinet 間データ転送

前述のように、Myrinet は、ノードのメモリとネットワークインターフェース上の SRAM の間の転送手段として、**PIO** と **DMA** の二種類を用意している。

ノードのメモリに対する DMA では、物理アドレスを指定する必要がある。一般にアドレス変換はシステムコールを必要とする。多くのユーザーレベル通信ライブラリでは、データ転送時に毎回システムコールを行うことを避けるため、次のような方式を用いる。ライブラリの初期化時に、システムコールによりページングを禁止したシステムバッファを設け、そのバッファの固定された物理アドレスをユーザーレベルに記録しておく。この固定されたシステムバッファを**コピーブロック**と呼ぶ。送信側は、ユーザーバッファからコピーブロックにデータを一旦コピーし、コピーブロックの物理アドレスを指定して、LANai に DMA を行わせる。受信側の LANai も届いたデータをコピーブロックに DMA 転送し、それをさらにノードの CPU がコピーブロックからユーザーバッファにコピーする。

AM は、今回の評価では、実装方式の異なる二種類の通信関数を用いた。`am_request_4()` は、4 ワードのデータを送信するもので、送信側受信側ともに PIO を用いる。アプリケーションは送信すべきデータを `am_request_4()` に引数として渡す。したがって、多数のレジスタを持つ RISC マシンでは、多くの場合引数がレジスタから LANai SRAM に直接転送される。`am_store()` は、最大 4KB までのバルクデータを転送するもので、送信側受信側ともに、ユーザーバッファとコピーブロックの間のコピーと、コピーブロックに対する DMA を用いるが、メッセージのヘッダは送信側受信側両方で PIO で読み書きする。

FM は、送信側では、ノードの CPU がユーザーバッファから LANai SRAM に PIO 転送を行う。受信側では、コピーブロックへの DMA と、コピーブロックからユーザーバッファへのコピーを行う。

VMMC では、小メッセージのみ送信側で PIO を使い、それ以外の場合は、送信側受信側ともユーザーバッファに対する DMA を用いる。送信側のユーザーバッファは初回使用時、受信側は公開時に OS を介してページングを禁止して物理アドレスを固定し、物理仮想アドレスの対応をユーザー空間にキャッシュする。送信側ユーザーバッファに多量の物理メモリが占有されないように、固定するページ数には上限を設け、必要に応じ、先に固定したページから解放する。

PM は、送信側受信側とも、システムバッファに対する DMA を用いる。PM の API では、ユーザーバッファとシステムバッファ間の転送は、必要な場合にのみユーザーが行う。

4 性能評価

以下に、評価項目を示す。

片道通信遅延 メッセージの往復所要時間の半分。

ソフトウェアオーバーヘッド ノードの CPU が送信および受信のために行う処理の所要時間合計。受信側オーバーヘッドは [4] にある手法で測定した。

単方向通信バンド幅 一対一で、片方がもう一方に繰り返し、相手からの返事を待たずにメッセージを送る際のバンド幅。

4.1 測定環境

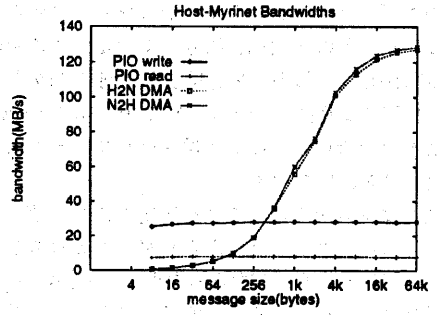

図 2: Myrinet・ノード間転送のバンド幅

測定には、Myrinet による、16 台構成の PC クラスタを用いた。各ノードは、200MHz Pentium Pro、512KB 二次キャッシュ、440FX(Natoma) チップセット、主メモリ 128MB を備え、LINUX2.0 を実行する。Myrinet のネットワークインターフェースには, PCI 用で、256KB の SRAM と LANai 4.1 プロセッサを持つ版を用いた。16 ノードを接続するため、8 ポートのスイッチ 4 個を、それらのスイッチを頂点とする正四面体を成すように相互に接続し、各スイッチに 4 ノードを接続した。

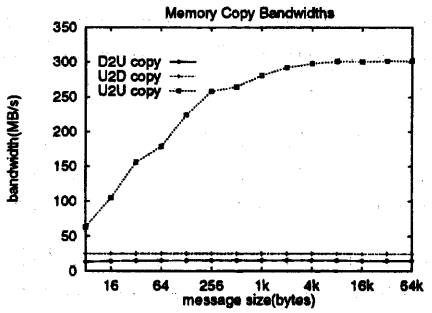

図 3: メモリコピーのバンド幅

図 2に、今回用いた環境における、PIO および DMA による LANai SRAM とノードのメモリの間の転送バンド幅を示す。H2N DMA がノードメモリから SRAM へ、N2H がその逆の DMA である。DMA のバンド幅は最大で PCI バスの転送速度の上限 133MB/s(クロック 33MHz * 32 bits) を達成しており、一ページ相当 (4KB) を単位とする転送でも 100MB/s を示す。PIO では、読み出し速度は 8MB/s しかない。しかし 256byte 以下では PIO による書き込みは DMA より速い。

図 3に、今回用いた環境での、コピーブロックとユーザーバッファ間のコピーのバンド幅を示す。U2U は通常のメモリ間コピーで、D2U、U2D はそれぞれコピーブロックから通常領域へ、通常領域からコピーブロックへのコピーを示す。コピーブロックはキャッシュを止めているため、書き込みで 25MB/s、読み出しでは 15MB/s 程度のバンド幅しかない。このため、コピーブロックを用いる方式では、コピーブロックのメモリバンド幅が通信性能の制約となる。

4.2 測定結果

図 4に片道遅延時間を、図 5に送信側と受信側のオーバーヘッドの合計を、図 6に単方向送信時のバンド幅を示す。

FM は送信側 PIO 受信側 DMA という使い分けにより小メッセージ通信遅延を短縮している。

AM は小メッセージ転送における通信遅延およびオーバーヘッドが高い。これは、AM が受信側 PIO を使っており、そのバンド幅が今回の測定環境で極めて悪いため。

VMMC は他に大きな差をつけて 100MB/s 超のバンド幅を示した。これはメモリコピーがないことの効果である。またオーバーヘッドが他に比べて小さいが、これはほとんどの処理を LANai に任せていること、および受信側の処理が存在しないためである。

5 おわりに

Myrinet を用いた特色ある 3 つの通信ライブラリについて、その機能と性能を比較した。AM は RPC 風のインターフェースを持ち、ネットワーク構成の自動検出ができるが、その実装方式が今回の測定環境に不向きなため、性能は他に比べ劣った。FM はストリームインターフェースを特徴とし、小メッセージ転送時には通信遅延、オーバーヘッドともに小さいが、ハンドラの処理の条件が複雑である。VMMC は、ユーザーバッファに対する直接 DMA 転送により、バンド幅で他に大きく勝り、また複数のチャネルを構成できるが、API が低レベルなため、他に比べプログラマの負担は大きい。

今回の測定に用いた版の FM 以降に、イリノイ大が FM の新版を開発した。この版では送信側での PIO のキャッシュモードを Uncached から Write Combining (WC) [2] に変え、また、受信

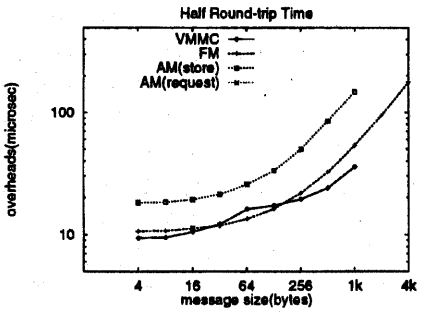

図 4: 片道遅延時間

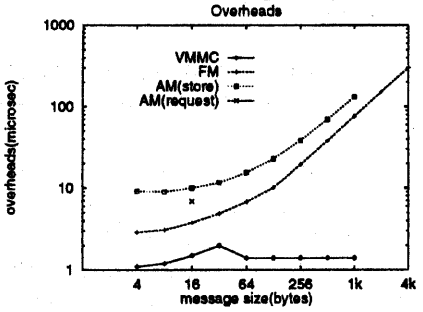

図 5: オーバーヘッド

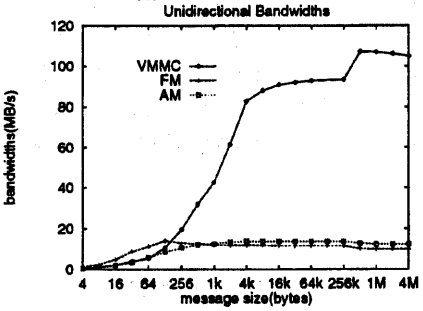

図 6: 単方向バンド幅

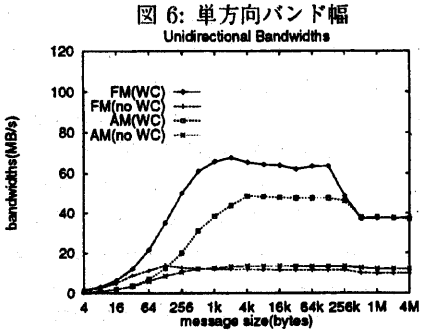

図 7: WC 版 AM, FM のバンド幅

側のコピーブロックのキャッシュを有効にすることで、バンド幅を大幅に改善している。この版の測定結果を、同じ手法を適用した AM の結果と合わせて図 7に示す。しかし、VMMC のバンド幅には及んでいない。このことはメモリコピーの回避の重要性を改めて示す。

PM はシステムバッファを限定的にユーザーに使わせることと多チャネルのサポートを特徴とする。今回性能を測定しなかったが、高い性能 [8] が報告されている。今後、PM についても同一プラットフォームによる評価を行いたい。

参考文献

[1] T.Eicken, D. Culler, S. Goldstein, and K. Schauser, *Active Messages: a mechanism for integrated communication and computation*, Proc. 19th Annual Symposium on Computer Architecture, pp 256-266, May 1992.

[2] Intel, *Pentium(R) Pro Family Developer's Manual, Vol. 3: Operating System Writer's Guide*, December 1995.

[3] H. Tezuka, A. Hori, and Y. Ishikawa. *PM: a high-performance communication library for multi-user parallel environments*, RWC Technical Report TR-96015, 1996.

[4] D. Culler, L. Liu, R. P. Martin, and C. Yoshikawa, *LogP performance assesment of fast network interfaces*, IEEE Micro, 1996.

[5] C. Dubnicki, L. Iftode, E. Felten, and K. Li, *Software support for virtual memory-mapped communication*, Proc. IPPS '96, 1996

[6] Scott Pakin, Vijay Karamcheti, and Andrew A. Chien, *Fast Messages: Efficient, Portable Communication for Workstation Clusters and MPPs*, IEEE Concurrency, Vol. 5, No. 2, 1997.

[7] *Myricom Home Page*, `http://www.myri.com`

[8] *PM: High-Performance Communication Library*, `http://www.rwcp.or.jp/lab/pdslab/pm/home.html`