Vol. 41 No. SIG 5(HPS 1) 情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム Aug. 2000

高速通信機構 PM2 の設計と評価

住元真司† 堀 敦史† 手塚宏史†
高橋俊行† 原田 浩† 石川 裕†

本論文では，Myrinet，Ethernet 等複数のネットワーク上で稼働する並列処理のための高速通信機構 PM2 を設計し評価する．PM2 は，既存ネットワークプロトコルに見られるプロトコルスタックの実現方法をとらず，各ネットワークデバイスごとに PM プロトコルを実現するアプローチをとる．PM2 は，実行時に呼び出すネットワークデバイスを決定するため，アプリケーションを再コンパイルすることなく異なるネットワーク環境上で実行可能である．Linux 上の PM ネットワークデバイスとして，Myrinet，Ethernet および SMP 用に OS の共有メモリ機構を用いたデバイスが実装されている．PM2 を Pentium III 500 MHz 2 CPU を搭載した 16 ノードの SMP クラスタで NAS 並列ベンチマークを用いて評価した結果，16 ノード時に IS では Gigabit Ethernet 上の TCP/IP の結果に比べ，PM2 の Myrinet では 2.5 倍，PM2 の Ethernet では 2.1 倍の性能向上が得られている．また，PM デバイスを実行時に切替えるオーバヘッドは，最も影響を受ける SMP 用の PM デバイスで 0.05 μs（3.6%）である．

The Design and Evaluation of High Performance Communication Facility: PM2

Shinji Sumimoto,† Atsushi Hori,† Hiroshi Tezuka,†
Toshiyuki Takahashi,† Hiroshi Harada† and Yutaka Ishikawa†

This paper describes the design and evaluation of a high performance communication facility, called *PM2*, for parallel processing on multiple network environments, such as Myrinet, Ethernet, and a facility using shared memory supported by the OS. Each PM device is realized independently using its optimal way. PM2 does not employ the protocol stack architecture used in existing network protocols, because of its overhead. PM2 switches network devices at runtime, and programs using PM2 are able to execute on differently configured of network environments without re-compiling. PM devices, Myrinet, Ethernet and a device using shared memory supported by an OS for SMP, have been implemented on Linux. The PM2 performance has been evaluated on a 16 node dual Pentium III 500 MHz PC SMP cluster. The results of the NAS parallel benchmark IS show that the IS performance on PM2 Myrinet is 2.5 times faster, and, that on PM2 Ethernet it is 2.1 times faster than that on TCP/IP on Gigabit Ethernet. The runtime device switching overhead is 0.05 μs (3.6%) on the shared memory PM device.

1. はじめに

PC クラスタはすでに研究対象だけではなく，本格的な普及段階に移行しつつある．PC クラスタの普及につれ，クラスタシステム向けネットワークも多く登場し，ユーザのクラスタシステム向けネットワークに対する要求も多様化していくと予想される．また，マルチ CPU 構成の PC も安価に手に入るようになり，これを用いた SMP クラスタが注目される等，クラスタの構成も多様化しつつある．

クラスタ構成の多様化の 1 つとして，複数クラスタを結合してより大きなクラスタとして利用する場合が想定される．これは，複数のクラスタが運用されている環境下で，より大きな計算能力が必要な場合に有効な手段である．このようなクラスタを *COC*（Cluster of Clusters）と呼ぶ．

クラスタ用高速ネットワークで結合されたクラスタを結合した *COC* では，必ずしもクラスタ用ネットワークが統一されているとは限らず，複数の異なるネットワークをどう扱うが問題となる．

本論文では，Myrinet[1]，Ethernet 等，複数の異な

† 新情報処理開発機構つくば研究センタ
Tsukuba Research Center, Real World Computing Partnership

るネットワーク上で稼働する並列処理のための高速通信機構 PM2 の設計と評価について述べる．PM2 は，既存ネットワークプロトコルに見られるプロトコルスタックの実現方法をとらず，各ネットワークデバイスごとに PM プロトコルを実現するアプローチをとる．PM2 では，実行時に呼び出すネットワークデバイスを決定するため，PM2 上のプログラムは再コンパイルすることなく異なるネットワーク環境上で実行可能である．また，複数ネットワークの支援機構により，複数のクラスタを結合してより大きなクラスタとして扱える環境（Cluster of Clusters）の提供が可能である．

PM2 は Linux 上に実装されており，現在，Myrinet，Ethernet および，SMP 用に OS の共有メモリ機構を用いた PM デバイスが実装されている．また，PM2 上のクラスタシステムソフトウェアである SCore3[2] と SCore3 上の MPI である MPICH/SCore も実現されている．Pentium III 500 MHz 2 CPU を搭載した 16 ノードの SMP クラスタで NAS 並列ベンチマークを用いて評価した結果，16 ノード時に IS では Gigabit Ethernet 上の TCP/IP の結果に比べ，PM2 の Myrinet では 2.5 倍，PM2 の Ethernet では 2.1 倍の性能向上が得られている．

本論文では，2 章で PM2 の設計の基になった PM について述べ，3 章で PM2 の設計について述べる．4 章で実装，5 章で評価，第 6 章で関連研究について述べ，7 章でまとめを行う．

2. PM

PM[3)~5)]は，チャネルと呼ばれる仮想ネットワークを提供している．TCP/IP のようにコネクション型通信ではなく，信頼性のあるデータグラム通信を実現している．並列アプリケーションの各プロセスは，チャネルを排他的に利用し，同じ番号のチャネルを用いて通信を行う．各ノードはノード番号によって管理され，他のノードへの通信はノード番号によって行われる．また，PM チャネルの状態を PM コンテキストと呼ぶ．PM はポーリングを用いたメッセージ通信のほかに，遠隔ノードのユーザメモリ間でデータ通信を実現する *Zero-Copy* 通信機構を採用し高い性能を実現している[5)]．

並列アプリケーションを書くユーザは，通常，PM 上の MPI である MPICH-PM/CLUMP を用いる．MPICH-PM/CLUMP は SMP 計算機内の MPI 通信において，複数プロセス間で直接コピーする *Direct memory copy* 機構の導入により共有メモリ経由のコピーオーバヘッドを削減している[6)]．

3. PM2 の設計

PM2 は，2 章で述べた PM のプロトコルと機能を実現したうえで，*COC* と SMP クラスタ対応，および，複数ネットワーク対応の通信アーキテクチャの実現を目標として設計された通信機構である．なお，取り扱うネットワークは Myrinet と Ethernet である．

3.1 *COC* と SMP クラスタへの対応

図 1 に，*COC* の例として，クラスタ用ネットワークとして Myrinet を用いた SMP PC クラスタと，クラスタ用ネットワークとして Gigabit Ethernet を利用した PC クラスタで構成される *COC* を示す．すべての PC は，Fast Ethernet で結合されている．図 1 においてそれぞれの PC 上の PE におけるクラスタ通信を考えてみる．

通信に利用するネットワークとしては最も高速なものを選択するのが妥当である．このように考えると，クラスタ 1 の SMP PC 内の PE 間は共有メモリ（図 1 中，Shmem）を用いたプロセス間通信，クラスタ 1 内のノード間通信には Myrinet，クラスタ 1 とクラスタ 2 のノード間の通信は Fast Ethernet を用いて通信を行うのがよい．また，クラスタ 2 内のノード間の通信には Gigabit Ethernet を用いるのがよい．

図 1 において，SMP 内のプロセス間通信についてもネットワーク通信であると考えると SMP クラスタも一種の *COC* と考えることができるため，*COC* でのクラスタ通信においては，SMP PC 内での通信を含めて通信相手ごとに利用するネットワークを設定する枠組があればよい．

これを提供する枠組として，*PM/Composite* と呼ぶ機構を導入する[2)]．*PM/Composite* は，相手先のノード番号ごとに利用するネットワークの情報を格納するためのテーブルを持ち，メッセージ処理要求があると登録されたネットワークに処理を振り分ける．たとえ

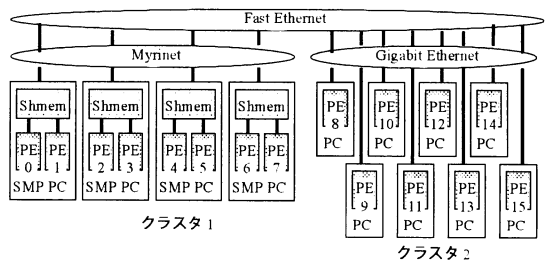

図 1 Myrinet クラスタと Gigabit Ethernet クラスタによる *COC*

Fig. 1 *COC* using Myrinet cluster and Gigabit Ethernet cluster.

表 1 PM/Compositeにおける宛先テーブルの例
Table 1 An example of destination table on PM/Composite.

PE#	dev	PE#	dev	PE#	dev	PE#	dev
0	–	1	SH	2	MY	3	MY
4	MY	5	MY	6	MY	7	MY
8	FE	9	FE	10	FE	11	FE
12	FE	13	FE	14	FE	15	FE

SH=Shmem, MY=Myrinet, FE=Fast Ethernet

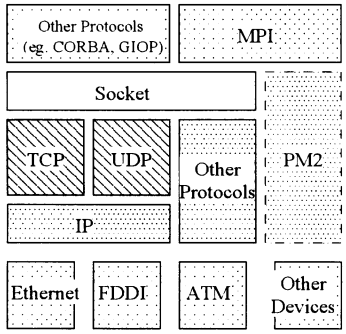

図 2 UNIX系OSのプロトコルスタック
Fig. 2 Protocol stacks on UNIX operating system.

ば，図1のPE0における *PM/Composite* のノードごとのPMデバイステーブルは**表1**のようになる．なお，1つのプログラムバイナリを異なるネットワーク環境下で動作させるには，プログラムの起動時に実行するネットワークに応じたPMデバイスを *PM/Composite* に登録すればよい．

3.2 複数ネットワーク対応の通信アーキテクチャ

複数のネットワークを扱うための代表的な通信アーキテクチャの例としてUNIX系OSのプロトコルスタックがある（**図2**）．プロトコルスタックは，ハードウェアを制御するデバイスドライバとユーザプログラムの間で通信プロトコルの制御を行う役割を持つ．

UNIX系OSのプロトコルスタックの利点は，デバイスとプロトコル間のインタフェースを規定することにより，複数のデバイスと複数のプロトコルの独立性を確保し，相互利用性を高めることができる点である．

ここで，UNIX系OSのプロトコルスタックを用いて，Ethernet，Myrinet，および，共有メモリ上でのプロセス間通信において，PM2で想定しているクラスタ向け通信で必要な信頼性のある通信を実現するためのプロトコルの実現について考える．

表2に，信頼性のある通信を実現するためのプロトコル処理への要求を各ネットワークごとにまとめる．Ethernetは，ハードウェアレベルでメッセージ転送を保証していないので，メッセージが失われる可能性がある．したがって，Ethernetで信頼性のある通信を実現するには，メッセージの到着保証と順序性を保証するためのプロトコル処理とバッファのフロー制御が必要である．しかし，Myrinetと共有メモリ上でのプロセス間通信は，ハードウェアレベルでメッセージ転送を保証しているため，メッセージの到着保証と順序性を保証するためのプロトコル処理は必要なく，バッファのフロー制御のみを行えばよい．

表 2 各ネットワークにおけるプロトコル処理への要求
Table 2 Protocol processing requirements on each network.

	メッセージ到着保証 順序保証プロトコル	バッファ フロー制御
共有メモリ	–	必須
Myrinet	–	必須
Ethernet	必須	必須

表2に示す要求を，UNIX系OSのプロトコルスタックの枠組で実現しようとした場合，Myrinetと共有メモリ上でのプロセス間通信においても，メッセージの到着保証と順序性を保証するためのプロトコル処理を行う必要があり，オーバヘッドが大きくハードウェアの性能を活かせない．クラスタ向け通信機構では，ネットワークハードウェアの性能を最大限に引き出すことが重要であるため，既存のOS上のプロトコルスタックの枠組は利用できない．

以上の理由により，ネットワークごとに最適な方式で実現することとした．

3.3 ハードウェアに特化した通信への対応

クラスタ向け通信では，ネットワークハードウェア性能を最大限に引き出すことが重要である．したがって，ハードウェアに特化した通信方式は，その方式の導入により顕著な通信性能の向上が見込めるのであれば，導入すべきである．

しかしながら，ハードウェアに特化した通信方式は，必ずしも他のすべてのネットワークで実現できるとは限らず，また，ハードウェアに特化した通信方式ごとに独立にAPIを作成するとアプリケーションの移植性が低下するという問題がある．

PM2は，ハードウェアに特化した通信方式の導入を可能とするため，オプション扱いのAPIを採用している．このAPIは，デバイスの属性によりプログラムが利用可能か判断して利用する．アプリケーションの移植性低下を最小限に抑えるため，オプション扱いのAPIの追加は最小限にし，可能な限り汎用に使えるものにすべきである．

PM2では，ハードウェアに特化した通信としてMyrinet上での *Zero-Copy* 通信機構[5]，および，共

表 3 PM2 の主要 API
Table 3 APIs on PM2.

PM デバイス操作（必須）	説明
pmOpenDevice()	PM デバイスのオープン
pmCloseDevice()	PM デバイスのクローズ
pmGetOptionBit()	PM デバイスのオプション獲得
PM コンテキスト操作（必須）	説明
pmOpenContext()	PM コンテキストのオープン
pmCloseContext()	PM コンテキストクローズ
pmAssociateNodes()	PM コンテキストへのノード登録
メッセージ送受信操作（必須）	説明
pmGetSendBuffer()	送信バッファの取得
pmSend()	メッセージ送信
pmReceive()	メッセージ受信
pmReleaseRecieveBuffer()	受信用バッファの解放
遠隔メモリ操作（オプション）	説明
pmMLock()	ユーザメモリの Pin-down
pmMUnlock()	ユーザメモリの Pin-down 解除
pmWrite()	遠隔メモリライトの実行
pmIsWriteDone()	遠隔メモリライト完了確認
pmRead()	遠隔メモリリードの実行
pmIsReadDone()	遠隔メモリリード完了確認

有メモリ上での *Direct memory copy* 機構[6]があるが，これらを汎用的に扱う枠組として遠隔メモリ操作の枠組を導入している．

3.4 PM2 API

表 3 に，PM2 の主要 API を示す．PM2 の API は，大きく分類して，PM デバイス，PM コンテキスト処理，メッセージ通信，遠隔メモリ操作に分類される．このうちすべての PM デバイスは，PM デバイス，PM コンテキスト処理，メッセージ通信の API を実装しなければならない．API の詳細については，PM Application Programmers' Interface Manual[7]を参照のこと．

PM デバイス，PM コンテキスト処理 API：（必須）
PM デバイスの Open，Close 処理，PM コンテキストの Open，Close 処理等，PM2 の初期化と通信環境設定を行う．

メッセージ通信 API：（必須）
ポーリングベースのメッセージ通信処理を行う．

- メッセージの送信は，`pmGetSendBuffer()` を用いて送信バッファを獲得し，この送信バッファ上にメッセージを構築し，`pmSend()` を用いて送信する．
- メッセージの受信は，`pmReceive()` によりメッセージを獲得する．受信したメッセージの処理後，`pmReleaseRecieveBuffer()` を用いて受信バッファを解放する．

遠隔メモリ操作 API：（オプション）
遠隔メモリ操作のための API として，ユーザメモリの pin-down 操作と遠隔メモリ操作の API を用意している．遠隔メモリ操作の利用手順は，以下のとおり．

- `pmMLock()` によりメモリ領域を pin-down する．
- `pmRead()`（Remote Memory Read），あるいは，`pmWrite()`（Remote Memory Write）を実行する．

3.5 PM デバイス

PM2 のドライバで提供すべき機能は，メッセージ送受信，および，オプションの遠隔メモリ操作，および，通信の初期化に関するプリミティブであり，どのように PM デバイスを設計すべきかはそれぞれのネットワークの性質と性能，および，ハードウェアの持つ機能により異なる．たとえば，以下の場合が考えられる．

- Network Interface Card（以下，NIC）上に CPU を持つ場合は，PM プロトコルとデバイス制御を NIC 上の CPU で実現する．
- 既存のデバイスドライバがある場合は，そのデバイスドライバを利用し，その上で PM プロトコルを実現する．

3.6 議　　論

3.2 節で，PM2 では各ネットワークデバイスごとに PM プロトコルを実現するアプローチをとると述べた．しかし，この方式を用いた場合，一般に以下の点が問題になる．

(1) プロトコルの実装コスト
(2) 各ネットワークごとにプロトコルを実装するコスト
(3) プロトコルの変更に対する修正にかかるコスト
(4) 性能チューニングコスト

しかし，以下に述べる理由により，これらの問題は高性能な通信を提供する目的からは，実装上問題にならない．

(1) PM 通信プロトコルは，表 2 で述べたメッセージ到着保証と順序保証プロトコル，および，バッファフロー制御とシンプルであるため，プロトコルの実装コストは小さく見積もることができる．
(2) PM 通信プロトコルは，シンプルであり，かつ，3.5 節に述べたようにネットワークハードウェアの機能や既存のデバイスドライバを用いることにより実装コストを削減できるため，実装コストはデバイスにより小さくできる．
(3) PM2 は，クラスタシステム上での並列処理という限られたアプリケーションを想定している

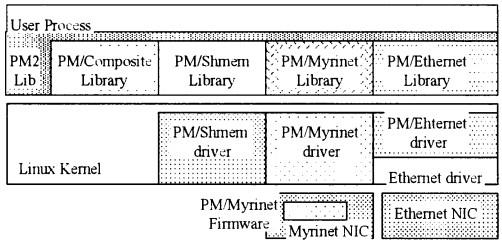

図3 PM2のアーキテクチャ
Fig. 3 PM2 architecture.

ため，プロトコル変更の容易性を重視する必要性は少ない．

(4) 性能のチューニングに関しては，デバイスに依存する場合が多く，チューニングコストの増加につながることは少ない．

4. PM2実装

4.1 実装の概要

PM2をLinux上に実装した．**図3**にPM2のアーキテクチャを示す．PM2は，全体をまとめるPM2ライブラリ（図3中，PM2 Lib）のほか，PMデバイスとして，PM/Myrinet，PM/Ethernet，PM/Shmem，およびPM/Compositeから構成される．

3.2節で述べたように，PM2はシンプルなクラスタ向け通信プロトコルを，Myrinet，共有メモリ，Ethernetのそれぞれについて，ハードウェアの性能を最大限に引き出すプロトコル実装方式を採用している．以下に各PMデバイスの実装の概要について述べる．

PM/Myrinet： 160 MB/sの帯域を持つ高速ネットワークであるMyrinet[1]上のPMである．

- 効率的なバッファフロー制御を行うためのプロトコルとして，Modified ACK/NACK方式[3]を採用しており，Myrinet NIC上のPM/Myrinetファームウェアで実行される．
- PM/Myrinetライブラリは，Myrinet NIC上のプロトコルとのインタフェースを行う．ユーザプロセス空間に`mmap()`関数を用いてMyrinet NICのレジスタとメモリをユーザメモリ空間上にマップしてインタフェース処理を行う．
- ユーザメモリのPin-down機構は，PM/Myrinetドライバで実装している．

PM/Ethernet： Ethernet上での通信機構である[8]．

- メッセージの到着保証と順序保証，およびバッファフロー制御を行う通信プロトコルとして，**STOP and GO**方式による受信バッファフロー制御を備えた**GO back N**方式[9]を採用しており，Linuxカーネル内のPM/Ethernetドライバで実行される．
- PM/Ethernetライブラリは，カーネル内のドライバへのインタフェースを行う．

PM/Shmem： 共有メモリを介したプロセス間通信をPM通信として扱う通信機構である．

- PMコンテキストとして共有メモリを割り当て，この共有メモリを介して通信を行う．
- 通信プロトコルはPM/Shmemライブラリで処理される．プロトコル処理は単純なリングバッファ処理である．
- *Direct memory copy*機構は，PM/Shmemドライバ上で実現されている．

PM/Composite： 上記のPMデバイスを相手ノードごとに切り替えるPMデバイスである．

- 相手先のノード番号ごとに利用するネットワーク指定のためのAPIを追加し，PM/Compositeライブラリで実現している．
- 相手先ノードのテーブルを引く機構とSMPのための排他制御機構を備えている．
- メッセージ送信時には，宛先テーブルにより送信先のノードに対応するPMデバイスを用いてメッセージが送信される．また，メッセージ受信時には，登録されているすべてのPMデバイスにポーリングする．
- PM/Compositeのオーバヘッドは，SMPのための排他制御と相手先のPMデバイスのテーブル引きと間接関数呼び出し程度である．

3.4節で述べたAPIのうち，PMデバイス処理は，全体をまとめるPM2ライブラリ部で，デバイス固有のPMデバイス処理，PMコンテキスト処理，メッセージ通信は各PMデバイスで実装されている．遠隔メモリ操作については，PM/MyrinetとPM/Shmemで実現されている．

各PMデバイス用のAPIはコンテキスト構造体上にAPIに対応する関数へのポインタが格納されており，間接関数呼び出しで実行される．

PM2は，以上述べたネットワーク以外にも，たとえば，NIC上にプログラムが変更可能なCPUを持つNIC，カーネル上のデバイスドライバで制御可能なNICであれば，PM2用のデバイスを開発することが可能である．

4.2 SCore3とMPICH/SCore

PM2上のクラスタシステムソフトウェアとして

SCore3[2]がある．SCore3 の実行環境では，コマンドオプションで利用するネットワークと構成を指定する．SCore3 は，PM デバイスを組み合わせて *PM/Composite* を作成し実行するプログラムに渡す．SMP クラスタとして利用する場合には，複数プロセスを起動し，それぞれのプロセスに対して *PM/Composite* を作成する．複数のプロセスが 1 つの PM デバイスを多重化して利用することになる．

以上のように，SCore3 上のプログラムは，*PM/Composite* を用いることにより，SCore3 から渡された PM デバイスの種類を意識することなく実行される．

SCore3 上の MPI である MPICH/SCore は，MPICH-1.1.2[10]を SCore3 に移植したもので，MPICH の持つ Channel インタフェースを用いて実現されている．

5. 評 価

本章では，PM2 が複数ネットワークに対応しながら，高い通信性能と実アプリケーション性能を実現しているかという点について評価する．

評価項目として，基本通信性能と実アプリケーションレベルの性能として NAS 並列ベンチマーク[11]を用いて評価する．PM/Ethernet，PM/Myinet については比較対象として TCP/IP を用いる．PM2 上のベンチマークプログラムと実アプリケーションが 1 つの実行バイナリで複数ネットワーク上で動作することを評価するため，各々のベンチマークプログラムは同一バイナリを用い，実行時のコマンドオプションで利用ネットワークを切り替えて測定を行う．

測定は 16 ノードの Pentium III の SMP クラスタで行った．測定環境を**表 4** に示す．すべての測定において表 4 に示すスイッチを利用している．PM/Ethernet と TCP/IP については，Gigabit Ethernet（以下，GigaEther），および，Fast Ethernet（以下，100-BaseT）上でも測定を行った．なお，PM2 の測定は，ポーリングを用いたメッセージ通信を用いている．

5.1 PM/Myrinet, PM/Ethernet の通信性能

基本通信性能として PM レベルのバンド幅とラウンドトリップ時間を TCP/IP と比較する．TCP/IP の通信性能は netperf-2.1pl3[12]を NODELAY オプションをつけて測定した．測定は，2 ノードを用いて行った．バンド幅は 100,000 メッセージを連続的に転送するのに要した時間より算出し，ラウンドトリップ時間は ping-pong プログラムを 100,000 回実行した時間より算出した平均値である．すべての測定結果はスイッチを利用したものである．

表 4 測定環境
Table 4 Measurement environments.

ノード PC	DUAL Pentium III 500 MHz 搭載 PC (440BX chipset, 512 MB SDRAM 32 bit 33 MHz PCI bus)
Ethernet NIC	Packet Engines 社 G-NIC II (GigaEther) Intel 社 EEPRO/100 (100BaseT)
Ethernet Switch	3Com 社 SuperStackII 9300 (GigaEther) 3Com 社 SuperStackII 3900 (100BaseT)
Myrinet	Myricom 社 M2M-PCI32
Myrinet Switch	Myricom 社 M2M-OCT-SW8
ホスト OS	Redhat 6.1 Linux (2.2.12 kernel)
G-NIC II デバイスドライバ	hamachi.c:v0.11 8/21/99 Written by Donald Becker
EEPRO/100 デバイスドライバ	eepro100.c:v1.09j-t 9/29/99 Written by Donald Becker

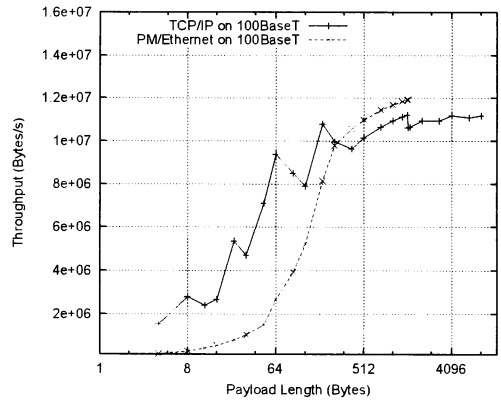

図 4 PM2 の通信バンド幅性能（100BaseT）
Fig. 4 Communication bandwidth on PM2 (100BaseT).

図 4 に 100BaseT 上での PM/Ethernet と TCP/IP の通信バンド幅性能を示す．最大バンド幅性能は，PM/Ethernet は 11.9 MB/s，TCP/IP は 11.1 MB/s と 100BaseT のほぼバンド幅一杯の性能である．メッセージ長が 256 バイト以下で TCP/IP の方がバンド幅が PM/Ethernet に比べ高いのは，TCP/IP が複数のメッセージを 1 つの TCP/IP パケットで送信しているからである[8]．

図 5 に，PM/Myrinet，GigaEther 上での PM/Ethernet，TCP/IP の通信バンド幅性能を示す．最大バンド幅性能は，PM/Myrinet は 116.1 MB/s，PM/Ethernet は 78.4 MB/s，TCP/IP は 38.8 MB/s であった．PM/Ethernet は TCP/IP に比べ 2.0 倍性能が高い．

なお，メッセージ長が 256 バイト以下で TCP/IP の方がバンド幅が PM/Ethernet に比べ高いのは，

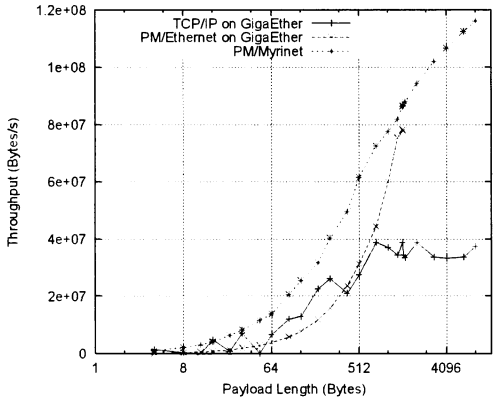

図 5 PM2 の通信バンド幅性能（GigaEther，Myrinet）
Fig. 5 Communication bandwidth on PM2 (GigaEther, Myrinet).

表 5 PM2 のラウンドトリップ時間
Table 5 Communication Round Trip Time on PM2.

	RTT
PM/Myrinet	16.4 μs
PM/Ethernet（GigaEther）	69.4 μs
TCP/IP（GigaEther）	125.0 μs
PM/Ethernet（100BaseT）	115.2 μs
TCP/IP（100BaseT）	178.4 μs

100BaseT のときと同じ理由による．

表 5 に PM/Myrinet および 100BaseT と GigaEther 上での PM/Ethernet，TCP/IP の 4 バイトメッセージにおける通信ラウンドトリップ時間を示す．表 5 の結果より，PM/Ethernet の 100BaseT の値の方が，GigaEther 上の TCP/IP より 7.2%良い性能であった．

5.2 PM/Shmem のラウンドトリップ時間と PM/Composite のオーバヘッド

本節では，PM/Shmem のラウンドトリップ時間と PM/Composite のオーバヘッドを示す．PM/Composite のオーバヘッドは，どの PM デバイスを用いても同じであるため，最も影響を受けやすい PM/Shmem を用いて測定した．

表 6 に *PM/Composite* を利用しない場合と利用した場合の PM/Shmem の PM レベルでのラウンドトリップ時間（RTT）を示す．PM/Shmem のラウンドトリップ時間は 2.75 μs，*PM/Composite* 挿入によるオーバヘッドはラウンドトリップ時間で 0.1 μs（片道，0.05 μs，3.6%）である．

5.3 実アプリケーションによる性能測定

実アプリケーション性能の比較のため，NAS 並列ベンチマーク ver 2.3[11] クラス A を用いて性能を測定する．測定に用いた NAS 並列ベンチマーク ver 2.3 の中の 8 つのベンチマークより，IS（大規模整数ソート），CG（大規模疎行列の最小固有値を用いるための共有勾配法），LU（Symmetric SOR iteration による CFD アプリケーション），および，BT（5x5 block size ADI iteration による CFD アプリケーション）である．

表 6 PM/Shmem のラウンドトリップ時間
Table 6 Communication Round Trip Time on PM/Shmem.

	w/o Composite	w/ Composite
MIN RTT	2.75 μs	2.85 μs

利用した MPI は，PM2 は MPICH/SCore，TCP/IP は MPI/LAM[13] である．以下の測定結果は，各ベンチマークプログラムごとに同一バイナリを用い，実行時のコマンドオプションで利用ネットワークを切り替えて測定した結果である．

図 6〜9 にそれぞれのベンチマーク結果を示す．これらの測定は，各ノードで 1 つのプロセスを実行させて行った．

図 6 の IS の結果より，16 PE 時に 100BaseT 上の TCP/IP が 2.5 倍の性能向上に留まっているのに対し，同じ 100BaseT 上の PM/Ethernet は 5.6 倍の性能向上を示している．同様に GigaEther 上の TCP/IP は 5.2 倍，PM/Ethernet は 11.0 倍，PM/Myrinet は 12.8 倍の性能向上を示しており，TCP/IP の性能に比べ PM2 の Myrinet では 2.5 倍，PM2 の Ethernet では 2.1 倍の性能を実現している．IS ベンチマークでは，TCP/IP より PM/Ethernet，PM/Myrinet を用いた方が高い性能が得られる．

図 7 の CG の結果は，GigaEther 上では TCP/IP より PM/Ethernet を用いた方が高い性能が得られるが，100BaseT 上では，PM/Ethernet でも TCP/IP でも性能が変わらないことを示している．

図 8，図 9 の結果より，これら 2 つのベンチマークについては，100BaseT 上の TCP/IP を除けば，どのネットワークを用いても性能差は少ないといえる．

5.4 SMP クラスタ上での実アプリケーション性能

本節では，SMP クラスタのアプリケーション性能を評価する．1 ノード内に 2 プロセスでベンチマークプログラムを実行した場合を測定し，1 ノード内に 1 プロセスでベンチマークプログラムを実行した場合と比較する．**図 10** と**図 11** にそれぞれ IS と LU ベンチマークの結果を示す．図中 SMP と表記したものは，1 PE の時を除き 1 ノード内に 2 PE でベンチマークプログラムを実行した場合の結果である．たとえば，SMP の 32 PE は 16 ノード × 2 PE での結果である．

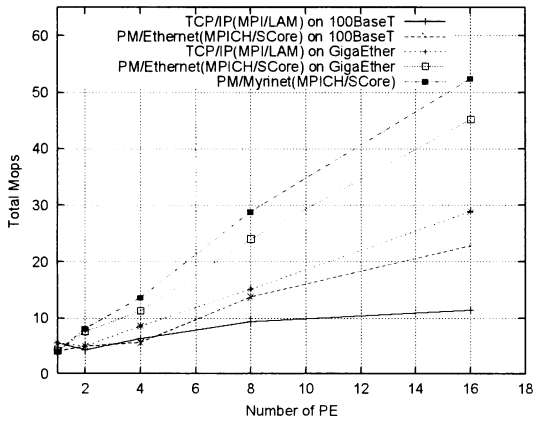

図 6 IS クラス A
Fig. 6 IS CLASS A.

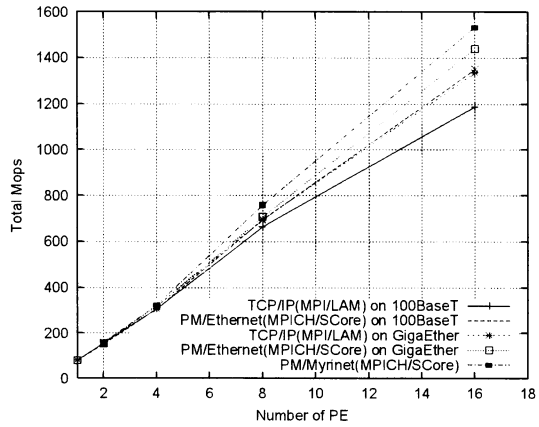

図 8 LU クラス A
Fig. 8 LU CLASS A.

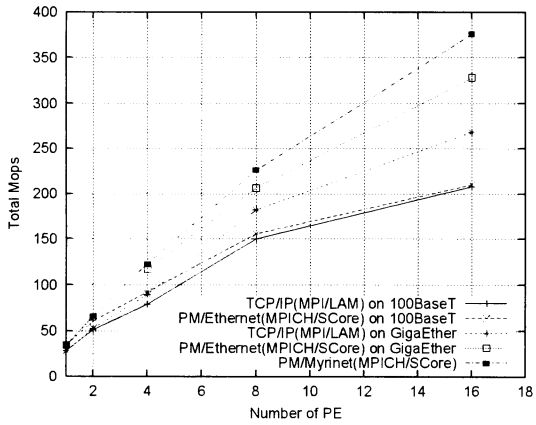

図 7 CG クラス A
Fig. 7 CG CLASS A.

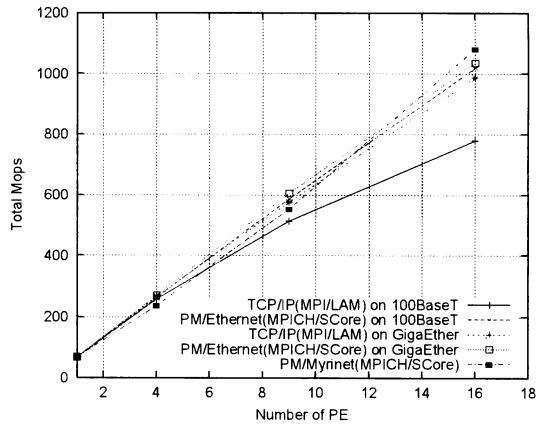

図 9 BT クラス A
Fig. 9 BT CLASS A.

図 10 の IS の結果より，複数プロセスで 1 つのネットワークを共有することによる遅延の増加により 16 PE 時に PM/Myrinet において 29%，GigaEther 上の PM/Ethernet で 33%の性能劣化が見られる．100BaseT 上の PM/Ethernet では性能向上が得られていない．

これとは対照的に図 11 の LU の結果は，16 PE 時に PM/Myrinet において 7%，GigaEther 上の PM/Ethernet で 9%，100BaseT 上の PM/Ethernet で 10%程度の性能劣化となっており，100BaseT を用いた SMP クラスタでも性能を引き出せている．

6. 関連研究

クラスタシステム向けの低レベル通信機構としては AM[14),15)]，AM-II[16)]，FM[17)]，GM[18)]，BIP[19),20)]，VMMC-2[21)]，U-Net[22)]がある．また，ユーザレベル通信の標準インタフェースとして，VIA[23)]（Virtual Interface Architecture）があり，マイクロソフトの Windows 上のギガビットクラスのネットワークに広く実装されている．以上述べたすべての低レベル通信機構は，複数のネットワークを扱うことを想定していない．

SMP を含むネットワーク上で動作するクラスタシステム向けの通信機構としては，BIP-SMP[20)]，Multiprotocol AM[24)]，GM[18)]がある．これらすべては現状，SMP と Myrinet のみで動作し，他のネットワークを同時に利用するための API は提供されていない．

BIP-SMP[20)]，Multiprotocol AM[24)]は，共有メモリと Myrinet への振り分けを行う機構を導入することにより SMP と Myrinet に対応しており，ノード内とノード外への通信の切り替えを考慮すればよいため，実装はモノシリックな構造となっている．また，ユー

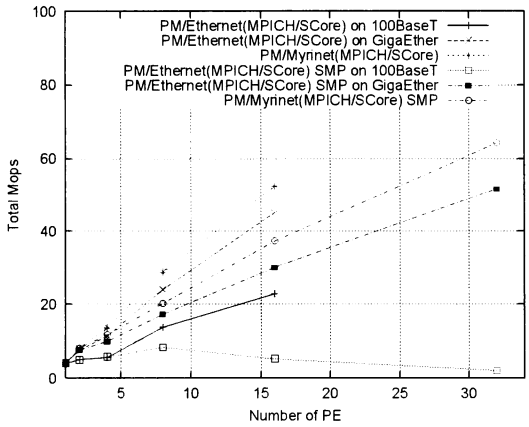

図10 IS クラスA on SMP
Fig. 10 IS CLASS A on SMP.

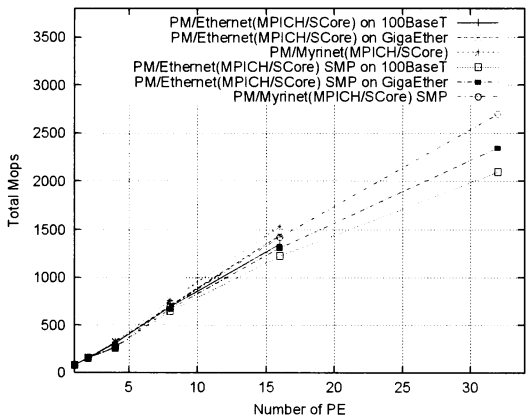

図11 LU クラスA on SMP
Fig. 11 LU CLASS A on SMP.

ザプログラムが通信先によりネットワークを選択するAPIは用意されていない．

GM[18]はSMPを含めたシステムで動作するが，共有メモリを用いて実装されているわけではなく，同じノード内の通信についてもMyrinet経由で通信する．

これに対してPM2は，SMP，Myrinet，共有メモリ，Ethernetのすべてについて PM2のAPIを提供し実装している．また，複数ネットワークを利用するためにはCompositeデバイスを用いる点がBIP-SMP[20]，Multiprotocol AM[24]と異なる．Compositeデバイスの導入により，ユーザプログラムにより利用できるネットワークを選択できる．たとえば，PM2のAPIを用いることで，複数のネットワークデバイス（ex. Myrinet×2，Ethernet×2，Myrinet＋Ethernet）を用いたクラスタ通信を実現できる．

7. まとめ

本論文では，複数のネットワーク上で稼働する並列処理のための高速通信機構PM2の設計と評価について述べた．PM2は，既存ネットワークプロトコルに見られるプロトコルスタックの実現方法をとらず，各ネットワークデバイスごとにPMプロトコルを実現するアプローチをとる．PM2ライブラリは，実行時に呼び出すネットワークデバイスを決定するため，PM2上のプログラムは再コンパイルすることなく異なるネットワーク環境上で実行可能である．また，複数ネットワークの支援機構により，複数のクラスタをLANで結合してより大きなクラスタとして扱える環境（Cluster of Clusters）の提供が可能となる．

PM2はLinux上に実装されており，Myrinet，Ethernetおよび，SMP用にOSの共有メモリ機構を用いたPMデバイスが実装されている．Pentium III 500 MHz 2 CPUを搭載した16ノードのSMPクラスタでNAS並列ベンチマークを用いて評価した結果，16ノード時にISではGigabit Ethernet上のTCP/IPの結果に比べ，PM2のMyrinetでは2.5倍，PM2のEthernetでは2.1倍の性能向上が得られている．また，PMデバイスを実行時に切り替えるオーバヘッドは，最も影響を受けるSMP用のPMデバイスで0.05 μs（3.6%）である．

PM2は現在，Myrinet，Ethernet，およびSMP上で実装されているが，他のネットワークにもPMデバイスを開発することにより実現可能である．

また，本論文ではクラスタ通信機構の性能の違いによる実アプリケーション性能の差を定量的に示した．

参考文献

1) Boden, N.J., Cohen, D., Felderman, R.E., Kulawik, A.E., Seitz, C.L., Seizovic, J.N. and Su. Wen-King: Myrinet - A Gigabit-per-Second Local-Area Network, *IEEE MICRO*, Vol.15. No.1, pp.29–36 (1995).
2) 堀，手塚，高橋，住元，曽田，原田，石川：クラスタ上のプログラミング開発環境—SCoreクラスタシステムソフトウェア，情報処理学会研究報告（*SWoPP'99*），99-HPC-77, pp.83–88 (1999).
3) 手塚，堀，石川：ワークステーションクラスタ用通信ライブラリPMの設計と実装，並列処理シンポジウム *JSPP'96*, pp.41–48. 情報処理学会 (1996).
4) Tezuka, H., Hori, A., Ishikawa, Y. and Sato, M. PM: An Operating System Coordinated High Performance Communication Library,

High-Performance Computing and Networking, Lecture Notes in Computer Science, Vol.1225, pp.708–717, Springer-Verlag (1997).
5) Tezuka, H., O'Carroll, F., Hori, A. and Ishikawa, Y.: Pin-down Cache: A Virtual Memory Management Technique for Zero-copy Communication, *IPPS/SPDP'98*, pp.308–314, IEEE (1998).
6) Takahashi, T., O'Carroll, F., Tezuka, H., Hori, A., Sumimoto, S., Harada, H., Ishikawa, Y. and Beckman, P.H.: Implementation and Evaluation of MPI on an SMP Cluster, *Parallel and Distributed Processing – IPPS/SPDP'99 Workshops*, Lecture Notes in Computer Science, Vol.1586, pp.1178–1192, Springer-Verlag (1999).
7) http://pdswww.rwcp.or.jp/dist/score/reference/man/man3/PM.html.
8) 住元, 堀, 手塚, 原田, 高橋, 石川：GigaE PM II：Gigabit Ethernet による高速通信ライブラリの設計, 情報処理学会研究報告 (*SWoPP'99*), 99-ARC-134, pp.61–66 (1999).
9) 住元, 堀, 手塚, 原田, 高橋, 石川：Gigabit Ethernet を用いた高速通信ライブラリの設計と評価, 並列処理シンポジウム *JSPP'99*, pp.63–70, 情報処理学会 (1999).
10) http://www-unix.mcs.anl.gov/mpi/mpich/.
11) http://www.nas.nasa.gov/Software/NPB/.
12) http://www.netperf.org/.
13) http://www.mpi.nd.edu/lam/.
14) von Eicken, T., Culler, D.E., Goldstein, S.C. and Schauser, K.E.: Active messages: A Mechanism for Integrated Communication and Computation, *Proc. of the 19th ISCA*, pp.256–266 (1992).
15) von Eicken, T., Avula, V., Basu, A. and Buch, V.: Low-Latency Communication over ATM Networks using Active Messages, *Proceedings of Hot Interconnects II* (1994).
16) Chun, B.N., Mainwaring, A.M. and Culler, D.E.: Virtual Network Transport Protocols for Myrinet, *Hot Interconnect'97* (1997).
17) Pakin, S., Lauria, M. and Chien, A.: High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet, *Proceedings of Supercomputing '95, San Diego, California* (1995).
18) http://www.myri.com/GM/doc/gm_toc.html.
19) Prylli, L. and Tourancheau, B.: BIP: A new protocol designed for high performance, *PC-NOW Workshop* (held in parallel with IPPS/SPDP98), Orlando, USA (1998).
20) Geoffray, P., Prylli, L. and Tourancheau, B.: BIP-SMP: High Performance Message Passing over a Cluster of Commodity SMPs, *Super Computing 99*, Portland, USA (1999).
21) Dubnicki, C., Bilas, A., Chen, Y., Damianakis, S. and Li, K.: VMMC-2: Efficient Support for Reliable, Connection-Oriented Communication, *Hot Interconnect'97* (1997).
22) Basu, A., Buch, V., Vogels, W. and von Eicken, T.: U-Net: A User-Level Network Interface for Parallel and Distributed Computing, *Proc. Third International Symposium on High Performance Computer Architecture (HPCA)*, (1997).
23) http://www.viarch.org/.
24) Lumetta, S.S., Mainwaring, A.M. and Culler, D.E.: Multi-protocol Active messages on a cluster of smp's, *Super Computing (SC'97)* (1997).

(平成 12 年 2 月 11 日受付)
(平成 12 年 6 月 2 日採録)

住元 真司（正会員）

1986 年同志社大学工学部電子工学科卒業. 同年（株）富士通入社.（株）富士通研究所にて並列オペレーティングシステム, 並列分散システムソフトウェアの研究開発に従事. 1997 年より新情報処理開発機構に出向. コモディティネットワークを用いた高速通信機構の研究開発に従事. 並列分散システムのアーキテクチャ, システムソフトウェア等に興味を持つ.

堀 敦史（正会員）

1979 年早稲田大学理工学部電気工学科卒業. 1981 年同大学院理工学研究科計測制御工学専攻修士課程修了. 同年（株）三菱総合研究所入社. 1992 年より技術研究組合新情報処理開発機構に出向. JSPP'98 最優秀論文賞受賞. 並列オペレーティングシステムの研究に従事. 並列プログラミング言語, 並列アーキテクチャ等に興味を持つ. 工学博士（東京大学工学部）.

手塚　宏史（正会員）

1980 年東京大学教養課程中退．1981 年（株）生活構造研究所入社．1985 年ソニー（株）入社．1988 年（株）ソニーコンピュータサイエンス研究所入社．1990 年ソニー（株）入社．1993 年北陸先端科学技術大学院大学研究生．1995 年より技術研究組合新情報処理開発機構研究員．現在に至る．オペレーティングシステム，リアルタイム処理，マルチメディア処理等に興味を持つ．日本ソフトウェア科学会会員．

高橋　俊行（正会員）

1993 年東京理科大学理工学部情報科学科卒業，1995 年同大学院修士課程修了，1995～1998 年東京大学理学系研究科情報科学科博士課程，1998 年より新情報処理開発機構研究員．現在に至る．プログラミング言語におけるメタレベルアーキテクチャと並列計算ソフトウェア技術に興味を持つ．理学修士．

原田　　浩（正会員）

1988 年東京理科大学理学部物理学科卒業．同年（株）ソフトウェア・リサーチ・アソシエイツ入社．1997 年より技術研究組合新情報処理開発機構研究員．現在に至る．オペレーティングシステム，並列・分散システム等に興味を持つ．ACM 会員．

石川　　裕（正会員）

1987 年慶応義塾大学大学院理工学研究科電気工学専攻博士課程修了．同年電子技術総合研究所入所．1988～1989 年カーネギー・メロン大学客員研究員．1990 年日本ソフトウェア科学会高橋奨励賞を受賞．1993 年から新情報処理開発機構に出向．並列・分散システム，適応可能並列プログラミング言語/環境/処理系，リアルタイム処理等に興味を持つ．日本ソフトウェア科学会，ACM，IEEE 各会員．工学博士．