GigaE PM: Gigabit Ethernetを用いた高速通信機構の設計と評価

住元真司[†] 堀敦史[†] 手塚宏史[†]
原田浩[†] 高橋俊行[†] 石川裕[†]

我々は，Gigabit Ethernetを用いたクラスタシステム上で並列アプリケーションを実行することを目的とした高速通信機構（GigaE PM）を開発している．GigaE PMは，信頼性を確保した高バンド幅，低遅延通信を実現し，かつ，同時にTCP/IPなどの既存の標準プロトコルもサポートしている．GigaE PMでは，クラスタシステム向けプロトコル処理はNIC上のファームウェアで，既存の通信プロトコル処理は従来どおりカーネルで処理する．プロトタイプシステムをEssential社NICに実装し評価した結果，Pentium II 400 MHz PC上で48.3 μsのラウンドトリップ遅延（4バイトメッセージ）と，56.7 MBのバンド幅（1,468バイトメッセージ）を実現している．さらにGigaE PM上にMPI通信ライブラリを構築し，GigaE PMとTCP/IP上のMPI通信ライブラリを用いたベンチマーク評価を行っている．NAS並列ベンチマークIS（クラスS）の評価では，GigaE PMの結果はTCP/IPの結果に比べ，1.8倍の性能向上を得ている．

GigaE PM: The Design and Evaluation of High Performance Communication Facility Using a Gigabit Ethernet

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

A high performance communication facility, called the *GigaE PM*, has been designed and implemented for parallel applications on cluster systems using a Gigabit Ethernet. The GigaE PM provides not only a reliable high bandwidth and low latency communication, but also supports existing network protocols such as TCP/IP. In the design of the GigaE PM, a reliable communication protocol for a parallel application is implemented on the firmware of network interface card while existing network protocols are handled by an operating system kernel. A prototype system has been implemented using an Essential Communications Gigabit Ethernet card, and evaluated its performance. The performance results show that a 48.3 μs round trip time for a four byte message, and 56.7 MB/sec bandwidth for a 1,468 byte message have been achieved on Intel Pentium II 400 MHz PCs. We have implemented MPICH-PM on the top of GigaE PM, and evaluated the performance using NAS parallel benchmarks. The results show that the IS (class S) performance on GigaE PM is 1.8 times faster than that of on TCP/IP.

1. はじめに

多数のPC/WSをクラスタ結合した価格対性能比が高い並列システムが広まりつつある．このようなシステムを実現するには2つのアプローチがある．1つは，ギガビットクラスのシステム間ネットワークを用いた専用の高速通信機構を利用する方法があり，既存の並列計算機と比較して安価に同等性能が得られる[3]．この通信機構としてたとえば，Myrinet[10]を用いたAM[6]，FM[7]，PM[1),2)]などがある．

もう1つのアプローチは，MPI，TCP/IPなどのコモディティソフトウェアと100 BASE-TX，ATM-LANなどのコモディティLANを利用する方法がある．現状のコモディティLANのバンド幅はたかだか100 Mbpsくらいであり，並列計算機のネットワークと比べてはるかに遅いが，既存のLAN環境を利用し，簡単，かつ，安価に構築できる．

近年，Gigabit Ethernetが普及し始め，次世代のコモディティLANとなりつつある．Gigabit Ethernetを利用すれば，高性能なクラスタシステムがLAN環境上で安価に構築可能になる．しかし，Gigabit Ethernet

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

上で既存の通信プロトコルを用いて，並列アプリケーションで必要なバンド幅と通信遅延を実現することは難しい．たとえば，Gigabit Ethernet上でのTCP/IP性能は物理層のバンド幅が125 MB/sにもかかわらず，DEC社製Alphaプロセッサ（533 MHz）の計算機（OSはWindows NT）とPacket Engine社製のGigabit Ethernetを用いた評価結果では，29.6 MB/sのバンド幅しか出ないと報告されている[4]．TCP/IP処理が性能上のボトルネックとなっており，Gigabit Ethernetの物理層性能を最大限に引き出せる通信機構が必要である．

我々は，LAN環境で既存の通信プロトコルをサポートし，同時に，並列アプリケーションのための高バンド幅，低遅延通信を実現するGigaE PM通信機構を設計実装している．GigaE PMを使えば，並列アプリケーションと既存の分散アプリケーションが共存できる環境を構築できる．

Gigabit Ethernetは，ハードウェアレベルでメッセージ転送を保証しないため，これを保証するプロトコル処理が必須である．このため，GigaE PMの設計では，プログラム可能なプロセッサを持つネットワークインタフェースカード（以降，NIC）を想定し，ホスト計算機（以降，ホスト）とNIC間のデータ交換のオーバヘッドを最小にするため，GigaE PMの通信プロトコルをNIC上で実現している．この結果，GigaE PMのプロトタイプでは信頼性のある通信を実現しながら，48.3 μsのラウンドトリップ遅延（4バイトメッセージ）と，56.7 MB/sのバンド幅（1,468バイトメッセージ）性能を実現している．

本論文では，コモディティLANであるGigabit Ethernetを用いたGigaE PMの設計と評価について述べる．2章でクラスタ通信における要件について述べ，Gigabit Ethernetを用いたクラスタ通信はどうあるべきかについて述べる．3章でGigaE PMの設計課題，4章でGigaE PMの設計と実装について述べる．5章ではGigaE PMの遅延，バンド幅性能，NAS並列ベンチマークについてTCP/IPと比較評価する．6章に高速通信機構に関する関連研究をまとめ，7章に結論を述べる．

2. クラスタ通信としてのGigabit Ethernet

本章では，並列計算向けの通信ネットワークの要件について述べ，Gigabit Ethernet上でクラスタ通信をどう実現すべきかについて述べる．

2.1 並列計算向け通信ネットワークの要件

本論文で想定している並列計算を実行するアプリケーションは，複数のプロセスで実現されており，それぞれのプロセスは1つのホスト上で実行され，他の多数のプロセスと通信する．

並列計算向け通信ネットワークの要件について以下に述べる．

- 信頼性のある通信：並列計算向け通信ネットワークは，メッセージ到着とメッセージの順序が必須である．たとえば，UDP/IPは，メッセージ転送を保証しないので，UDP/IPだけでは使えず，別途メッセージ転送を保証する機構が必要になる．
- 少ないメモリ使用量：並列計算向け通信ネットワークは，多数の計算機が結合されるため，通信ネットワーク維持に必要なメモリ使用量は最小であるべきである．たとえば，TCP/IPを用いた1000ノードのクラスタを考えた場合，他ノードとの通信のためノードあたり999ポートのコネクションを張る必要があり，かつ，それぞれのコネクションについて送受信バッファが必要となる．送受信バッファがそれぞれ128 KBの場合，約256 MBの送受信バッファが必要になり，ノードの搭載メモリ量で結合可能なノード数が制限される．
- 高い通信性能：上記2つの要件を満たしたうえで高い通信性能（高バンド幅，低遅延）を実現する必要がある．
- 低コスト：並列計算向け通信ネットワークは，多数のノードを結合するため，低コストで実現されるべきである．

2.2 Gigabit Ethernetのクラスタ通信への適用

Gigabit Ethernetをクラスタ通信へ適用しようとした場合，Gigabit Ethernetはハードウェアレベルでメッセージ転送を保証しないため，これを保証するプロトコル処理が必須である．このため，GigaE PMの設計では，プログラム可能なプロセッサを持つNICを採用し，GigaE PMの通信プロトコルをNIC上で実現している．NIC上にプロトコルを実現することで，プロトコル処理に必要な制御フレームの送受信を入出力バス（たとえば，PCIバス）を経由する必要がなくなり，ホストとNIC間のデータ交換のオーバヘッドをより少なくできると考えたからである．

また，実装すべきプロトコルとして，メッセージ転送を保証したデータグラム転送プロトコルを採用することにより，メモリ使用量の少ない信頼性のある通信を実現することとした．

3. 設計課題

本章では，GigaE PM の設計課題について述べる．

(1) **シンプルな並列計算向き通信プロトコルの実現：**
前章の議論より，通信プロトコルは，多数のプロセスと通信するため，巨大な送受信バッファを必要としてはならない．また，通信プロトコルは，NIC 上の限られたハード資源内で実現可能なくらいシンプルでなければならない．

(2) **メッセージの到着保証と順序保証の実現：**
Gigabit Ethernet はメッセージ転送を保証していないので，通信プロトコルはメッセージの到着保証と順序保証をサポートする必要がある．

(3) **NIC とホスト間の情報交換コストの最小化：**
高い通信性能を実現するため，NIC とホスト間の情報交換コストを最小化する必要がある．

(4) **並列計算向け通信プロトコルと TCP/IP など他プロトコルとの共存の実現：**
通信機構は TCP/IP など他のプロトコルが同時に利用される環境で利用されるため，通信機構は高性能な通信とともに既存の通信プロトコルも同時にサポートする必要がある．

4. 設計と実装

4.1 仮想ネットワークと API

GigaE PM 通信機構は並列計算アプリケーションに適応できるよう設計されている．GigaE PM は，PM[1] で採用されている仮想ネットワークをサポートするためのチャネルを提供している．このチャネルはメッセージ転送を保証したデータグラム通信を提供する．

並列アプリケーションのそれぞれのプロセスは，仮想ネットワーク上の同じ番号のチャネルを排他的に利用する．チャネルの数は NIC のハードウェア資源に依存する．現状の実装では，4 つのチャネルをサポートしている．チャネル数よりも多くのチャネルを必要とする並列プロセスを実現するために，我々は PM[1] のチャネルの多重化を実現する SCore-D システムを開発している．

GigaE PM の API は PM[1] の API に準拠しており，PM 上に開発された MPI を含む SCore システムとアプリケーションは，再コンパイルにより実行することができる．

4.2 信頼性のある通信

メッセージの到着と順序性を保証するために **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** 方式を採用した．

4.2.1 GO back N プロトコル

GO back N プロトコルでは，送信側は i 番目から $i+N$ 番目までのデータを受信側からの ACK メッセージを待つことなく送信できる．受信側は i 番目のメッセージの受信時に送信側に ACK メッセージを送信する．送信側は i 番目の ACK メッセージを受信後に $i+1$ 番目から $i+1+N$ 番目のデータを受信側に送信できる．

もし，送信側が一定時間内に i 番目の ACK メッセージを受信しない場合は，送信側は i 番目とそれに続くメッセージを再度送信する．これは，i 番目のデータが失われたことを意味する．送信側は i 番目から $i+N$ 番目までのメッセージのバッファ領域を確保する必要がある．TCP/IP と違い，受信側は N 個のメッセージ分バッファ領域を確保する必要がない．

もし，受信側がシーケンス番号 i より大きな番号のメッセージを受信した場合，受信データは廃棄され，送信側に **LOSE** メッセージを送信する．この点も TCP/IP の sliding window プロトコルと異なる．

もし，受信側のメッセージバッファがフルになった場合は，**STOP** メッセージが送信側に送信される．受信側は十分なバッファ領域が確保できた時点で **GO** メッセージを送信側に送信する．これは **STOP and GO** フロー制御と呼ばれている．

通信プロトコルの詳細は付録参照のこと．

4.3 ホストと NIC 間の情報交換

ホストと NIC 間の情報交換コストを定量的に見積もることは高バンド幅，低遅延の通信機構を設計するうえできわめて重要である．本節では，この情報交換のコストを解析し，このコストを基に通信機構を設計するうえで重要なディスクリプタの配置，アクセス方法，および，ホストと NIC 間の通知方法について議論する．

ホストと NIC 間の情報交換コストを Essential 社の PCI Gigabit Ethernet NIC を用い，Linux 2.1.131 を搭載した Pentium II 400 MHz の PC 上で解析した．**表 1** に，ホストと NIC 間のデータ転送，割込み，システムコール（ioctl）にかかるコストを示す．これらのコストは以下のようにして測定した．

プロセッサによるホストとホスト間のデータ転送：
10 MB のユーザメモリ空間に対して，ホストプロセッサ上でデータ転送プログラムを作成して測定し，得られた測定値を基に式を作成した．

プロセッサによるホストと NIC 間のデータ転送：
ユーザメモリ空間に NIC メモリを `mmap()` し，ホストプロセッサでこの領域に対するデータ転送プ

表1 ホストNIC間データ転送，割込み，システムコールコスト
Table 1 Data transfer cost between the host and NIC memories and interrupt/system call processing.

	コスト (μs)
N ワード転送コスト（1ワードは4バイト）	
プロセッサコピー：ホストからホスト	$0.03 \times N$
プロセッサコピー：ホストからNIC	$0.25 \times N$
プロセッサコピー：NICからホスト	$0.57 \times N$
NIC DMA機能：ホストからNIC	$2.30 + 0.03 \times N$
NIC DMA機能：NICからホスト	$2.00 + 0.03 \times N$
割込み処理コスト	6.5
システムコールコスト	1.0

ログラムを作成して測定し，得られた測定値を基に式を作成した．

NIC DMAによるホストとNIC間のデータ転送： NIC上でファームウェアを作成して測定を行った．測定はDMA起動のためのレジスタ設定とDMA完了までの時間をNICの持つタイマを用いて測定し，得られた測定値を基に式を作成した．

割込みのコスト： NIC上のファームウェア，および，ホストプロセッサ上で割込み処理を行うデバイスドライバと測定プログラムを作成し，NICファームウェアがプロセッサへの割込みレジスタに書き込んだ後，ホストプロセッサ上のデバイスドライバの割込み処理関数の先頭にたどり着くまでの時間をPCI Gigabit Ethernet NICの持つタイマを用いて測定した．

システムコールコスト： デバイスドライバと測定用プログラムを作成し，このデバイスドライバへの4引数の`ioctl()`が完了する時間をプロセッサのタイマを用いて測定した．

GigaE PMにおける情報交換はメッセージディスクリプタを使って行われる．それぞれのエントリはメッセージバッファのアドレス，メッセージサイズとメッセージタイプに依存した送信側あるいは受信側の識別子から構成される．

4.3.1 ディスクリプタの配置

メッセージディスクリプタの置き場所は，ホストメモリ上かNICメモリ上に置くことができる．このディスクリプタはホストとNICの両方がアクセスするため，両方のアクセスコストの総和が最小になるような場所に置くべきである．それぞれに必要なコストは，表1のデータを基に計算すると以下となる．

(1) ホストメモリ上：NIC DMAによるホストからNICへの3ワード転送コスト $= 2.39\,\mu$s
(2) NICメモリ上：プロセッサコピーによるホストからNICへの3ワード転送コスト $= 0.75\,\mu$s

Essential社のNICを用いた我々の実装では，アクセスコストを最小にするためメッセージディスクリプタはNIC上に置いている．

4.3.2 ディスクリプタへのアクセス

もし，ユーザプロセスが，直接送信メッセージディスクリプタを更新し，NICに新しいディスクリプタが準備できたことを知らせるのであれば，送信時においてシステムコールは必要ない．この場合安全な通信を保証するためには，以下の2つを行う必要がある．

(1) メッセージディスクリプタへのアクセスは，ユーザプロセスが適切な場所のみを参照可能とし，かつ，他のユーザプロセスからのアクセスを禁止する．
(2) ユーザプロセスが間違ったメッセージアドレスを書かないかどうか，NICが送信メッセージディスクリプタをチェックする．

Essential社のNICでは，NIC上メモリへのアクセスに制御レジスタの更新が必要である．このため，もし，ユーザプロセスが制御レジスタの更新が可能な場合，ユーザプロセスは他の制御レジスタも更新できるため危険である．

GigaE PMでは安全な通信を保証するため，ディスクリプタの更新はシステムコールを用いてカーネルで行う．なお，カーネルでディスクリプタ更新を行うことによるオーバヘッドは，表1より $1.0\,\mu$s と見積もることができる．

4.3.3 通　　知

ホストからNIC，および，NICからホストへの通知方式とそのコストを表1のデータを基に計算した結果を以下に示す．

ホストからNIC： 2つの方式が選択可能

(1) **ホストがNICメモリのフラグを更新：** プロセッサコピーによる1ワード転送コスト $= 0.25\,\mu$s
(2) **NICがホストメモリのフラグをポーリング：** NIC DMAによる1ワード転送コスト $= 2.33\,\mu$s

以上の計算結果より，ホストがNICメモリに書く方式を採用する．

NICからホスト： 3つの方式が選択可能

(1) **NICがホストメモリのフラグを更新：** NIC DMAによる1ワード転送コスト $= 2.03\,\mu$s
(2) **ホストがNICメモリのフラグをポーリング：** プロセッサコピーによる1ワード転送コスト $=$

0.57 μs

(3) **NIC がホストに割込み：** 割込み+プロセッサコピーによる 1 ワード転送コスト = 7.05 μs

以上の計算結果より，ホストが NIC メモリのフラグをポーリングする方式を採用する．しかし，安全な通信を保証するためポーリングをシステムコールで行うのは，さらに 1.0 μs のコストがかかりオーバヘッドが大きい．

そこで，Essential 社の NIC では，ホストが制御レジスタを更新しないで直接アクセスできる少量のメモリ領域があるため，この領域にユーザプロセス用の受信メッセージディスクリプタを置くこととした．ユーザプロセスが，このディスクリプタを **Read-Only** で直接参照することにより，システムコールを使わなくてもディスクリプタを参照できる．

4.4 GigaE PM 実装の概要

本節では，GigaE PM 実装の概要について述べる．GigaE PM は，NIC 上のファームウェア，デバイスドライバ，および，ユーザライブラリから構成される．

- API は，PM[1),2)] と同じである．現状，Zero-Copy 機能は未実装．
- 送受信バッファは，ホストメモリ上にあらかじめ割り当てられ，pin-down される．ユーザ領域へは `mmap()` でマップして利用される．
- NIC のレジスタは Read-Only でユーザ空間にマップされる．
- GigaE PM の MTU は，1,468 バイトである．

4.5 送受信処理の概要

本節では，**図 1** を利用して GigaE PM の送受信処理の概要を説明し，ホストと NIC 間の情報交換がどのように行われるかを示す．

送信側：

(1) ユーザプログラムは `_PM_getsendBuf()` を発行してメッセージバッファを獲得する（図 1 中 sc1）．引数はチャネル番号とバッファサイズである．このバッファ上でメッセージを作成する．

(2) ユーザプログラムが `_PM_sendmsg()` を実行すると，GigaE PM ドライバが起動（図 1 中 sc2）され，このドライバは送信メッセージの情報（メッセージアドレス，サイズ，送信先識別子）を NIC のディスクリプタに書く（図 1 中 sc3）．

(3) NIC はユーザ空間から NIC メモリ領域にメッセージを転送する（図 1 中 sd1）．

受信側：

(1) NIC が新しいメッセージを受信した場合，ユーザプロセス上にマップされた受信メッセージ領域に受信メッセージを転送する（図 1 中 rd1）．そして，NIC は到着フラグを更新する．

(2) ユーザプログラムが `_PM_receive()` を実行すると，`_PM_receive()` はシステムコールを発行することなくメッセージ到着フラグを参照する（図 1 中 rc1）．もし，このフラグが立っていたら受信メッセージディスクリプタより，メッセージバッファアドレス，サイズ，送信者 ID を獲得し，ユーザプログラムはメッセージ処理を行う（図 1 中 rc2）．

(3) ユーザプログラムが `_PM_putreceiveBuf()` を実行すると，GigaE PM ドライバが起動され（図 1 中 rc3），このメッセージバッファが解放されたことを知らせる（図 1 中 rc4）．

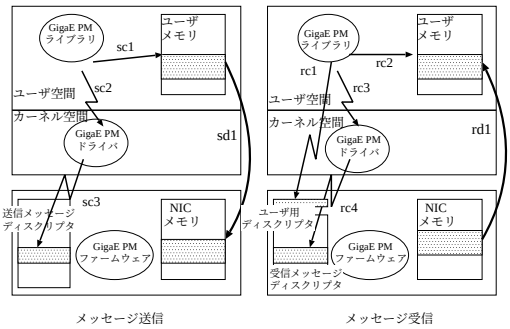


図 1 GigaE PM でのディスクリプタ，ホスト，NIC
Fig. 1 Descriptors, host, and NIC in the GigaE PM.

4.6 GigaE PM ファームウェア

GigaE PM ファームウェアは，他の多くの PCI Ethernet NIC がハードウェアの機能として備えている PCI DMA とネットワークインタフェースの DMA を制御し，メッセージ転送を行う．クラスタ通信については GigaE PM 通信プロトコル処理を行う．ホストプロセッサとの情報交換は，NIC メモリを介して行う．

他の多くの NIC の持つ機能を用いて実装しているため，NIC 上にプログラム可能なプロセッサとホストプロセッサがアクセス可能なメモリを持つ NIC であれば，GigaE PM を実装することが可能である．

4.7 GigaE PM と他プロトコル

GigaE PM の通信プロトコルは NIC で処理されるが，他の通信プロトコルは，他の Ethernet NIC と同様ホストで処理される．このプロトコルの切替えは Ethernet フレームのタイプで行っている．GigaE PM の場合には特殊なタイプを用い，タイプが GigaE PM のものでないパケットが届いた場合，そのパケットはそのままホストに転送され，ホストのデバイスドライバの割込みハンドラが起動される．

表2 測定環境
Table 2 Machine environment.

ハードウェア	Pentium II 400 MHz，440 BX チップセット，256 MB SDRAM メモリ
NIC	Essential 社 PCI Gigabit Ethernet NIC model EC-440-SF (33 MHz clock, PCI DMA, MAC SEEQ 8100 MAC), 1 MB MEM
スイッチ	Extreme 社 Summit 1
ホスト OS	Redhat 5.1 Linux (2.1.131 kernel)

5. 評　　価

この章では，基本性能として遅延とバンド幅の測定結果，および，NAS 並列ベンチマークの性能評価をTCP/IP と比較する．**表2** に測定環境を示す．

GigaE PM のバンド幅と遅延の測定では，次の2つの場合について評価を行った．なお，TCP/IP の結果はすべてスイッチを用いて接続した場合である．

(1) 2台のホストを Extreme 社スイッチを用いて接続した場合
(2) 2台のホストをスイッチなしで直接接続した場合

5.1 バンド幅

図2 に GigaE PM，GigaE PM 上の TCP/IP，Essential 社ファームウェア上での TCP/IP のバンド幅の測定結果を示す．GigaE PM は，1,468 バイトメッセージ時に 56.7 MB/s のバンド幅性能である．これに対して TCP/IP の場合は，Essential 社製ファームウェア利用，1,280 バイトメッセージのときに 33.0 MB/s のバンド幅性能である．

GigaE PM 上の TCP/IP と Essential 社ファームウェア上の TCP/IP 性能の比較では，4 バイトから 256 バイトメッセージまでは，GigaE PM を用いた場合が性能が良く，512 バイトから 1,280 バイトメッセージまでは，Essential 社ファームウェアを用いた場合が性能が良い．

5.2 遅　　延

図3 にラウンドトリップ遅延の結果を示す．GigaE PM は，4 バイトメッセージ時にスイッチを介さない場合は 48.3 μs のラウンドトリップ遅延であり，Summit 1 スイッチ経由の場合に 58.3 μs となる．これは，スイッチの遅延が単方向で 5.0 μs であることを意味する．

一方，TCP/IP のラウンドトリップ遅延は Essential 社製ファームウェア利用，4 バイトメッセージ時に 414.0 μs であるのに対して，GigaE PM を用いた場合には，131.4 μs であった．GigaE PM は，高速通信プロトコルとともに，既存のネットワークプロトコルにおいても高い性能を実現している．

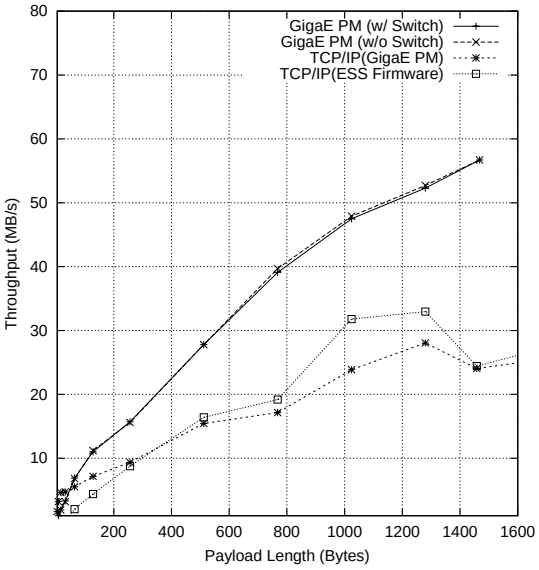


図2 バンド幅性能
Fig. 2 Bandwidth.

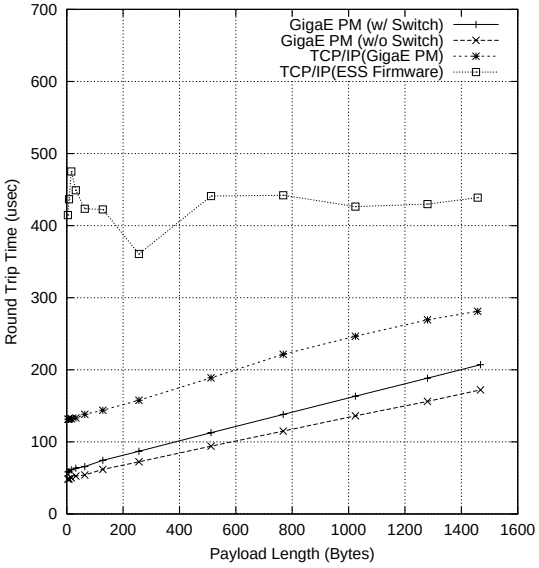


図3 ラウンドトリップ遅延
Fig. 3 Round trip time.

Essential 社製ファームウェアを利用した場合のラウンドトリップ遅延がメッセージサイズに関係なく 400 μs 付近の値を示しているのは，ホストプロセッサへの割込み負荷を抑えて通信バンド幅性能を向上させるため，割込みを NIC の持つタイマにより一定間隔遅らせているからである．図2 における GigaE PM 上の TCP/IP よりも Essential 社製ファームウェアを用いた場合の方が通信バンド幅性能が良いのはこの理

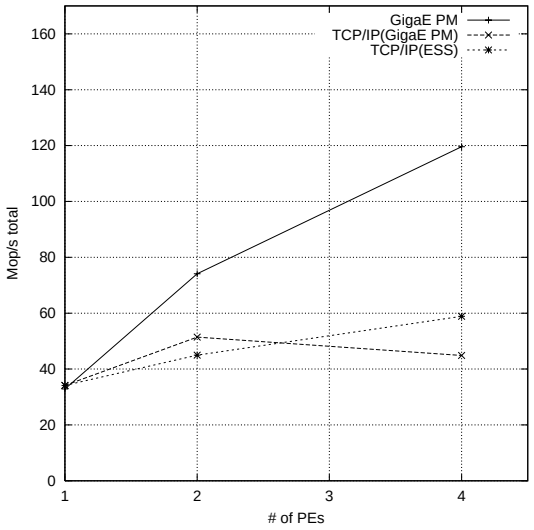


図 4 CG クラス S
Fig. 4 CG Class S.

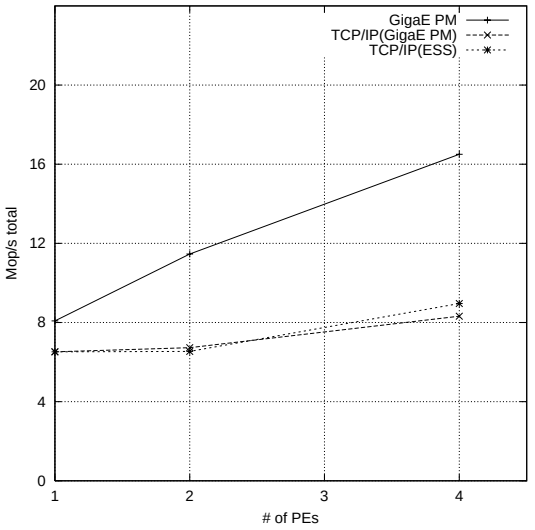


図 5 IS クラス S
Fig. 5 IS Class S.

由による．

5.3 NAS 並列ベンチマーク

GigaE PM 上に MPICH-PM（MPICH 1.0.11 ベース）を移植し，NAS 並列ベンチマーク 2.3[5]の性能を測定した．NAS 並列ベンチマーク 2.3 には 8 つのベンチマークがあるが，この中で低い通信遅延と高いバンド幅が必要な CG（Conjugate Gradient），IS（Integer Sort）の性能を測定した．結果を**図 4**，**図 5** に示す．クラスは S である．比較のため，GigaE PM 上の TCP/IP，Essential 社製ファームウェアを用いた結果を載せる．TCP/IP に用いた MPI は MPICH 1.0.11 の ch_p4 である．

結果より，GigaE PM では 4 ノード時に 3.63 倍（CG），2.0 倍（IS）の性能向上が見られるのに対し，TCP/IP では，4 ノード時に最大，1.72 倍（CG），1.37 倍（IS）の性能向上であった．4 ノード時の GigaE PM の結果は，TCP/IP の結果より，2.0 倍（CG），1.84 倍（IS）高速であり，GigaE PM の通信性能の高さを反映した結果である．

5.4 メモリ使用量比較

表 3 に N ノード，および，1000 ノードのクラスタでの TCP/IP と GigaE PM における送受信バッファと制御に必要なメモリ使用量を示す．TCP/IP の送受信バッファのメモリ量は Linux 2.1.131 のデフォールト値，制御構造体は tcp_opt 構造体のサイズである．

表 3 の結果より，GigaE PM の送受信バッファに必要なメモリ量はノード数に依存しないこと，制御構造体も 1000 ノードのクラスタにおいても 23 KB と，NIC 上メモリに収まることが分かる．反面，コネクションベースの TCP/IP はノード数に比例した送受信バッファが必要になるため，ノードの搭載メモリ量で結合可能なノード数が制限される．

表 3 N ノードクラスタにおけるメモリ使用量の比較
Table 3 Comparison of memory usage in an N node cluster.

	TCP/IP	GigaE PM
N ノードクラスタ		
送信バッファ（KB）	$128 \times (N-1)$	120
受信バッファ（KB）	$128 \times (N-1)$	128
制御構造体（KB）	$0.274 \times (N-1)$	$0.023 \times N$
1000 ノードクラスタ		
送信バッファ（KB）	127,987	120
受信バッファ（KB）	127,987	128
制御構造体（KB）	274	23

6. 関連研究

数多くの高性能通信機構が開発されている．AM[6]，AM-II[8]，FM[7]，VMMC-2[9]，と PM[1] は Myrinet[10] をベースとしている．Myrinet はギガビットクラスのネットワークでハードウェアレベルでメッセージ転送を保証している．それゆえにネットワーク上でメッセージが失われることは考える必要はなく，受信バッファのフロー制御を行えばよい．

システムコールのオーバヘッドを減らすために，PM と U-Net[11]ではシステムコールを使っていない．PM は Myrinet 上で 15 μs のラウンドトリップ遅延を実現しているため，システムコールのオーバヘッドは大き

い．しかしながら，GigaE PMの場合，システムコールのオーバヘッドは，48.3 μsのラウンドトリップ遅延の中で2.0 μs（4.1%）であるため，それほど大きな影響はない．

GigaE PMと同様，AM-II[8]も，信頼性のある通信をNIC上で実現し，かつ，安全な通信をサポートしている．AM-IIはシステムコールを使わずに安全な通信を実現しているが，GigaE PMはシステムコールを使い，安全，かつ，低遅延，高バンド幅通信を実現している．

VIA[12]（Virtual Interface Architecture）はマイクロソフトの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．VIAはその上にsocketやMPIのような通信ライブラリが実装されることを想定している．VIAはコネクションベースの通信をサポートしており，信頼性のある通信はVIA specification Version 1.0ではオプションである．

もし，コネクションベースの通信をGigabit Ethernetのような信頼性のないネットワーク上で実現する場合，通信プロトコル処理は，より大きな通信バッファを必要とする．よって，我々は，コネクションベースで並列アプリケーション（特にデータ並列アプリケーション）向きの高速通信のサポートは難しいと考える．GigaE PMのプロトコルはシンプルで他のコネクションベースの通信機構と比べ大きなメッセージバッファを必要としないため，NIC上で実現でき，かつGigabit Ethernet上で信頼性のある通信を実現することができる．

7. おわりに

本論文では，Gigabit Ethernet上で並列計算向きの高速通信を実現するためには，NICとホスト間の情報交換を考慮した設計が重要であることを示した．高速通信機構GigaE PMはこれを満たすよう設計されている．

プロトタイプをEssential社のGigabit Ethernet NIC上に実装，評価した結果，4バイトメッセージ時に48.3 μsのラウンドトリップタイム，1,468バイトメッセージ時に56.7 MB/sのバンド幅が得られている．同時に，GigaE PMは，性能を落とすことなく，TCP/IPプロトコルもサポートしている．また，MPICH-PM上のNAS並列ベンチマーク（クラスS）で評価した結果，4ノード時のGigaE PMの結果は，TCP/IPの結果より，2.0倍（CG），1.84倍（IS）高速である．

本論文の寄与するところは，Gigabit Ethernetを用いた高速通信機構の設計である．GigaE PMは，他の多くのNICが持つハードウェア機能を用いて実装しているため，NIC上にプログラム可能なプロセッサとホストプロセッサがアクセス可能なメモリを持つNICであれば実装することができる．

GigaE PMは，信頼性のある，高バンド幅，低遅延通信を提供するだけでなく，同時にTCP/IPなどの既存の通信プロトコルもサポートしている．この機能を用いることにより，並列アプリケーションと分散アプリケーションが共存した高性能なクラスタシステムが構築できる．

参考文献

1) Tezuka, H., Hori, A., Ishikawa, Y. and Sato, M.: PM: An Operating System Coordinated High Performance Communication Library, Sloot, P. and Hertzberger, B. (Eds.), *High-Performance Computing and Networking*, Vol.1225, Lecture Notes in Computer Science, pp.708–717, Springer-Verlag (Apr. 1997).
2) 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 (Apr. 1998).
3) http://www.rwcp.or.jp/lab/pdslab/benchmarks/.
4) http://www.packetengines.com/products/performance/gnicntperf.htm.
5) http://www.nas.nasa.gov/Software/NPB/
6) http://now.cs.berkeley.edu/AM/lam_release.html.
7) Pakin, S., Lauria, M. and Chein, A.: High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet, *Proc. Supercomputing '95*, San Diego, California (1995).
8) Chun, B.N., Mainwaring, A.M. and Culler, D.E.: Virtual Network Transport Protocols for Myrinet, *Hot Interconnect'97* (Aug. 1997).
9) Dubnicki, C., Bilas, A., Chen, Y., Damianakis, S. and Li, K.: VMMC-2: Efficient Support for Reliable, Connection-Oriented Communication, *Hot Interconnect'97* (Aug. 1997).
10) Boden, N.J., Cohen, D., Felderman, R.E., Kulawik, A.E., Seitz, C.L., Seizovic, J.N. and Su, W.-K.: Myrinet – A Gigabit-per-Second Local-Area Network, *IEEE MICRO*, Vol.15, No.1, pp.29–36 (Feb. 1995).
11) Basu, A., Buch, V., Vogels, W. and von Eicken, T.: U-Net: A User-Level Network Interface for Parallel and Distributed Computing, *Proc.3rd International Symposium on High*

Performance Computer Architecture HPCA (Feb. 1997).
12) http://www.viarch.org/.

付　　録

GigaE PM プロトコル

ここでは GigaE PM の通信プロトコルの詳細について述べる．それぞれのチャネルに送信バッファと受信バッファがあり，すべての送信メッセージは送信バッファ，すべての受信メッセージは受信バッファに格納される．バッファは，TCP/IP と違い peer to peer 通信単位ではなくチャネル単位に割り当てられる．

これ以降利用する用語を定義する．メッセージは，Msg(送信側 ID，受信側 ID, メッセージ順序番号) で表現する．N は送信側が ACK メッセージを待つことなく非同期に送信できるメッセージ数を示す．T はタイムアウト時間，$SBuf(r,i)$ は i 番目の送信メッセージを格納する送信バッファ，$STime(r,i)$ は i 番目のメッセージが受信側 r に送信された時刻，$MsgIdSent(r)$ は受信側 r に送信されたメッセージ順序番号の最大値，$MsgIdAcked(r)$ は受信側 r から受信した ACK メッセージの順序番号の最大値，$MsgIdRecv(s)$ は受信側が送信側 s から受信したメッセージ順序番号の最大値とする．初期化時点において $MsgIdSent(r)$, $MsgIdAcked(r)$ と $MsgIdRecv(s)$ は 0 である．

送信側と受信側における GigaE PM プロトコルを以下に示す：

送信側において：

S1 送信側 s は，次の条件を満たすメッセージ $Msg(s,r,i)$ を受信側 r に送信する．送信時には，以下の処理を行う．

$\forall\, Msg(s,r,j)$ where $MsgIdSent(r) < j < MsgIdAcked(r) + N$.

(1) $Msg(s,r,i)$ を受信側に送信する．
(2) 送信メッセージ $Msg(s,r,i)$ のバッファ $SBuf(r,i)$ を作成し，維持する．
(3) 現在の時刻を $STime(r,i)$ に格納する．
(4) $MsgIdSent(r) \leftarrow MsgIdSent(r) + 1$.

S2 現在時刻と $STime(s,r,i)$ の差が T より大きい場合，以下を実行する．

(1) $MsgIdSent(r) \leftarrow i$.
(2) S1 を実行する．

- もし次の条件を満たすLOSEメッセージ$LOSE(r,k)$を受信した場合，以下を実行する．

  $LOSE(r,k)$ where $MsgIdAcked(r) < k < MsgIdAcked(r) + N$.

  (1) 次の条件を満たす送信メッセージバッファ $SBuf(r,i)$ を解放する．
  $\forall SBuf(r,i)$ where $MsgIdAcked(r) < k$.
  (2) $MsgIdAcked(r) \leftarrow k$.
  (3) S1 を実行する．
- もし次の条件を満たすSTOPメッセージ$STOP(r,k)$を受信した場合，以下を実行する．

  $STOP(r,k)$ where $MsgIdAcked(r) < k < MsgIdAcked(r) + N$.

  (1) 次の条件を満たす送信メッセージバッファ $SBuf(r,i)$ を解放する．
  $\forall SBuf(r,i)$ where $MsgIdAcked(r) < k$.
  (2) $MsgIdAcked(r) \leftarrow k$.
  (3) 送信を停止する．
- もし次の条件を満たす GO メッセージ $GO(r,k)$ を受信した場合，S1 を実行する．

  $GO(r,k)$ where $MsgIdAcked(r) \leq k < MsgIdAcked(r) + N$.
- もし次の条件を満たす ACK メッセージ $ACK(r,k)$ を受信した場合，以下を実行する．

  $ACK(r,k)$ where $MsgIdAcked(r) \leq k < MsgIdAcked(r) + N$.

  (1) 次の条件を満たす送信メッセージバッファ $SBuf(r,i)$ を解放する．
  $\forall SBuf(r,i)$ where $MsgIdAcked(r) \leq k$.
  (2) $MsgIdAcked(r) \leftarrow k$.
  (3) S1 を実行する．

受信側において：

- 受信側 r がメッセージ $Msg(s,r,i)$ を受信したとき，
  - $MsgIdRecv(s) + 1 = i$ を満たし，かつ，受信側 r の受信バッファがフルでない場合，以下の処理を行う．
    (1) ACK メッセージ $ACK(r,i)$ を送信側 s に送信する．
    (2) 受信メッセージをホストに転送する．
    (3) $MsgIdRecv(s) \leftarrow MsgIdRecv(s) + 1$.
  - $MsgIdRecv(s) + 1 = i$ を満たし，かつ，受信側 r の受信バッファがフルの場合，以下の処理を行う．
    (1) STOP メッセージ $STOP(r,i)$ を送信側 s に送信する．
    (2) これ以降，到着するすべての受信メッセージを廃棄する．

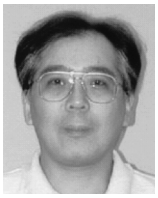
- $MsgIdRecv(s)+1<i$ を満たす場合，以下を処理を行う．これは，メッセージが失われたことを意味する．
  (1) LOSE メッセージ
  $LOSE(r, MsgIdRecv(s))$
  を送信側 s に送信する．

- もし，ホストの受信バッファに十分な空きがあり，かつ，STOP メッセージが送信済みの場合，以下を実行する．
  (1) GO メッセージ $GO(r, MsgIdRecv(s)+1)$ を送信側 s に送信する．
  (2) 受信側 r は，受信処理を再開する．

(平成 11 年 8 月 31 日受付)
(平成 11 年 12 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．

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．

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 年〜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．

石川 裕（正会員）

1987 年慶應義塾大学大学院理工学科電気工学科博士課程修了．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．