論　文

Maestro2クラスタネットワーク向けメッセージパッシングライブラリの開発と評価

青木　圭一[†a)]　山際　伸一[††]　和田　耕一[†]　小野　雅晃[†]

Development and Evaluation of Message Passing Library for Maestro2 Cluster Network

Keiichi AOKI[†a)], Shinichi YAMAGIWA[††], Koichi WADA[†], and Masaaki ONO[†]

あらまし　PCの価格性能比の向上から，PCを用いたクラスタによる並列計算が幅広く用いられるようになっている．クラスタの性能向上には，通信性能を高めるだけでなく，通信にかかわる処理をネットワークに移行し，計算と通信を重ね合わせることによって，全体の処理効率を向上させるアプローチも重要である．このようなアプローチの有効性を検証するため，我々は汎用プロセッサとそれに密に結合した通信用ハードウェアをもつMaestro2 cluster networkの開発を進めている．本論文では，Maestro2 cluster network上に構築したメッセージパッシング方式の通信ライブラリMMPの設計と実装について述べている．MMPでは，Maestro2 cluster network上の汎用プロセッサがネットワークハードウェアを制御し，通信や同期処理の大部分をホストプロセッサとは独立に行う，という特徴をもつ．評価結果から，MMPは，ホストプロセッサ制御による通信性能と比較して，高いスループットを有することを明らかにしている．

キーワード　クラスタコンピューティング，クラスタネットワーク，オフロード，通信ライブラリ

1. まえがき

PC（パーソナルコンピュータ）の価格性能比の向上から，多量の演算を必要とする処理に対して，PCのクラスタによる並列計算が幅広く用いられるようになっている．近年では，Myrinet [1]，QsNet [2]，RHiNet-2 [3]など，Ethernetに代表される広域向けネットワークと比べ，レイテンシとスループットに優れた高速ネットワークが開発されてきた．これらのネットワークは，高速な物理レイヤを利用し，通信プロトコルを最適化することで通信性能を高めることに主眼をおいたものといえる．

一方で，ネットワークの通信性能を高めるだけでなく，通信にかかわる処理をネットワークコンポーネントがホストプロセッサに代わって行うことで，クラスタの性能向上を図るアプローチも重要である．このようなアプローチの例として，通信プロトコルソフトウェアをネットワークに移行する方式があり，その有効性が示されている[4]～[6]．しかし，並列に動作するアプリケーションの間で，計算処理と通信処理を更にオーバラップさせながら高速に通信を行うためには，従来アプリケーションが行っている，アプリケーションの設計，実装に依存する通信処理やアルゴリズムの一部をネットワークハードウェアに移行する必要があると考えられる．このような通信処理の移行の例としては，分散型共有メモリシステムにおける遠隔ノードのメモリ操作や並列シミュレーションにおいて，各ノードで共有すべきデータの領域計算をネットワークインタフェースに移行したものがある[7]，[8]．これらの先行研究では，高性能プロセッサと十分な容量のメモリをもつネットワークインタフェースの必要性が指摘されている．

我々は，通信処理など並列処理の一部をネットワー

† 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻，つくば市
Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba, 1-1-1 Tennodai, Tsukuba-shi, 305-8573 Japan
†† INESC-ID Lisboa，ポルトガル
INESC-ID, Rua Alves Redol 9 Apartado 13069, Lisboa, Portugal
a) E-mail: k1@padc.cs.tsukuba.ac.jp

クコンポーネント自身で実行可能なクラスタ向けネットワーク Maestro2 cluster network を開発している．本ネットワークは，汎用プロセッサ，通信モジュールを実装する FPGA，大容量メモリを密に結合したアーキテクチャを採用しており，通信プロトコル処理のみならず，従来アプリケーションプログラムの一部として実装されてきた通信処理の一部を，ホストプロセッサに代わって実行することが可能である．また，Maestro2 cluster network では，自律性と同時に高い通信性能を達成するため，FPGA に実装されるリンクレイヤコントローラは，長いバースト転送を行う．この高速化技法により，物理レイヤの通信性能を引き出し，高い通信性能を達成できる [9], [10].

しかし，アプリケーションプログラムが通信を行う際，広域ネットワーク向けの通信プロトコルソフトウェアを用いた場合，リンクレイヤの通信性能を引き出せないばかりか，プロセッサが通信処理やネットワークハードウェアの制御にリソースを費やし，アプリケーションプログラムの性能が向上しないおそれがある [11], [12]．したがって，アプリケーションプログラムが，Maestro2 cluster network のリンクレイヤでの性能を引き出して通信するためには，アプリケーションプログラムの処理の一部をネットワークインタフェースへと移行するのと同時に，Maestro2 cluster network の高機能性を生かした通信ソフトウェアの開発が必須である．そのため，本研究では，Maestro2 cluster network の備える高速プロセッサを用いて，リンクレイヤの通信性能を引き出す高性能メッセージパッシングライブラリ MMP の開発を行っている．本論文では，MMP の開発と評価について述べる．

本研究で開発する高性能メッセージパッシングライブラリ MMP は，次のような特徴を備える．まず，ネットワークインタフェース上に搭載されている汎用プロセッサが，通信処理の大部分をホストプロセッサとは独立に行う．これにより，ネットワークインタフェースが自律的に転送を行うため，通信中にホストプロセッサがネットワークインタフェースを制御する必要がない．併せて，ライブラリのインタフェースはすべてノンブロッキングとし，通信処理と計算処理の重ね合わせを図る．また，オペレーティングシステムのカーネルに含まれる通信機能を用いず，アプリケーションプログラムがユーザ空間で確保したバッファとネットワークインタフェース間で直接データを転送するゼロコピー通信 [13]〜[15] を行う．MMP のゼロコピー通信においては，ゼロコピーに必要なアドレス変換をネットワークインタフェース上の汎用プロセッサが行うことで，ホストプロセッサによるアドレス変換，またはネットワークインタフェース上の専用ハードウェアの支援無しにゼロコピーを実現する．

また，ネットワークインタフェースが複雑な受信処理を高速に処理できることを生かし，事前に予想できないタイミングやサイズでネットワークインタフェースが受信したメッセージをゼロコピーでメインメモリに転送するための，active zero-copy 機構を備える．active zero-copy 機構を用いると，メッセージがネットワークインタフェースに到着した際，そのメッセージの受信先領域が登録されていない場合でも，ネットワークインタフェースがホストプロセッサに受信用領域の確保を依頼し，そのアドレス情報をホストプロセッサから受け取り，ゼロコピーでメッセージを受信することができる．

更に，転送要求のキューはネットワークインタフェース上のメモリに，転送状態をホストプロセッサに知らせるシグナルバッファは，ホストプロセッサのメモリ上に配置する．このようにそれぞれのローカルメモリにバッファを配置することにより，ネットワークインタフェースが転送開始を行う際や，ホストプロセッサが転送の進捗状況を確認する際に，バッファのポーリングを行っても，ネットワークインタフェースとホストプロセッサ間を接続するバスへの負荷を抑えることができる．

この論文では，上述した特徴を備える Maestro2 cluster network 向け通信ソフトウェア MMP の設計について詳述する．また，実験により通信性能を測定し，汎用のプロセッサを用いた MMP の性能が，ハードウェアの基本性能にどの程度迫れるかに注目し，評価を行う．

2. Maestro2 cluster network

2.1 背　　景

我々の開発している Maestro2 cluster network は，主にデータリンク層，物理層，通信ソフトウェア層の三つのレイヤから構成される．本節ではデータリンク層と物理層のそれぞれのレイヤにおいて発生し得るボトルネックと，Maestro2 cluster network におけるボトルネックを避ける技法について述べる．

2.1.1 データリンク層

データリンク層で発生し得るボトルネックの原因の

一つは，冗長な転送単位である．したがってデータリンク層においては，クラスタ内通信の規模に適した，最小限のフレーム情報の付加が必要となる．

また，送信操作の繰返しに伴うオーバヘッドも，発生し得るボトルネックの原因として挙げられる．多くのネットワークでは，リンクレイヤにおいて，送信単位の最大量に満たなくとも送信単位を一つずつ送信している．このような方法では，複数の送信単位を送信する際，たとえそれが 1 度の送信機会で送信できる長さであったとしても，毎回送信操作を行い，送信操作に伴う時間が蓄積してしまう．したがって，通信性能を高めるためには，送信単位の大きさにかかわらず，連続して送出できるよう，設計する必要がある．Maestro2 cluster network では，送信操作の繰返しを避ける技法として，continuous network burst を提案している [10]．以下に，この技法の詳細と，continuous network burst を用いたリンクレイヤプロトコルについて述べる．

（1） Continuous network burst

この技法は，Maestro cluster network [16], [17] で提案した network burst を更に改良したものである．送信操作の繰返しによるオーバヘッドを避けるため，continuous network burst では，データリンク層においてメッセージを細かなパケットと呼ぶ単位に分割し，それらをまとめバースト転送する．また，転送開始後も新たにメッセージがデータリンク層に渡され，バースト転送を延長できる場合は，*continue command* と呼ぶ情報を挿入して，受信側にバースト転送を継続することを通知し，バースト転送を継続する．continuous network burst のバースト転送と，送信機会ごとに送信を繰り返す転送との比較を，図 1 に示す．このバースト転送により，物理層の利用率を高め，通信スループットを向上させることができる．

（2） MLX リンクレイヤプロトコル

Maestro2 cluster network では，データリンク層の通信プロトコルとして，MLX リンクレイヤプロトコルを用いる．MLX リンクレイヤプロトコルは，図 2 に示す入出力と通信路を想定している．プロトコルを実装するモジュールは，物理層とネットワークバッファの間に接続される．ネットワークバッファは 2ch の FIFO バッファとして構成され，ホストプロセッサ側から，書込み，読出しが行われる．対向するプロトコルモジュールは，ピアツーピアで接続し，全二重通信を行う．また，フロー制御は，対向するプロトコルモジュールが，お互いにバッファの空き容量情報 credit を交換することで行う．前述の continuous network burst は，この空き容量情報に基づいてバースト転送を行う．

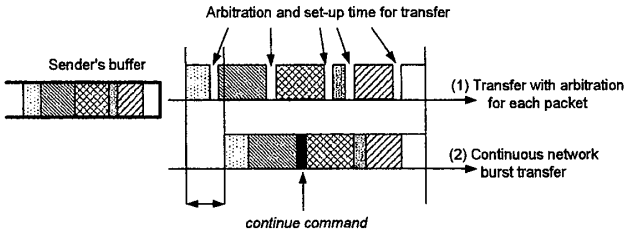


図 1 Continuous network burst と送信操作を繰り返す転送の比較

Fig. 1 Comparison between continuous network burst and non-burst transfer.

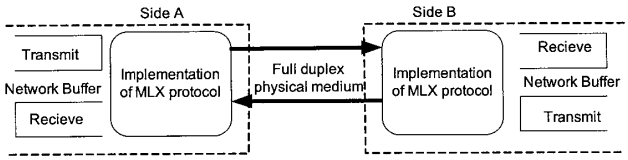


図 2 MLX リンクレイヤプロトコルの通信システムモデル

Fig. 2 Configuration of MLX protocol.

2.1.2 物 理 層

物理層は，その上位レイヤにおいて，マルチキャストに伴うオーバヘッドが発生しないよう，構成する必要がある．例えばイーサネットでは，マルチキャストを実現するために，全ノードにブロードキャスト通信を行い，転送対象外のノードでは受信したデータを破棄している．また，Myrinet においては，転送元から転送先にピアツーピアで 1 ノードずつ転送を行い，マルチキャスト通信を実現している [18], [19]．しかし，これらの方法では，メッセージをコピーする操作や必要のないメッセージを破棄する操作がオーバヘッドとなる．Maestro2 cluster network では，物理層において転送元から転送先のノードのみに同時に転送できるようにすることで，このオーバヘッドを回避する．

2.2 Maestro2 cluster network の全体構成

Maestro2 cluster network は，図 3 に示すように，ネットワークインタフェースとスイッチボックスで構成される．ネットワークインタフェースは 64 bit 66 MHz PCI バスでホストプロセッサと接続され，ホストプロセッサとメッセージ交換を行う．スイッチボックスは各ネットワークインタフェースと LVDS [20] ケーブルで接続され，ネットワークインタフェースとメッセージ交換を行う．以下に，ネットワークインタフェース，

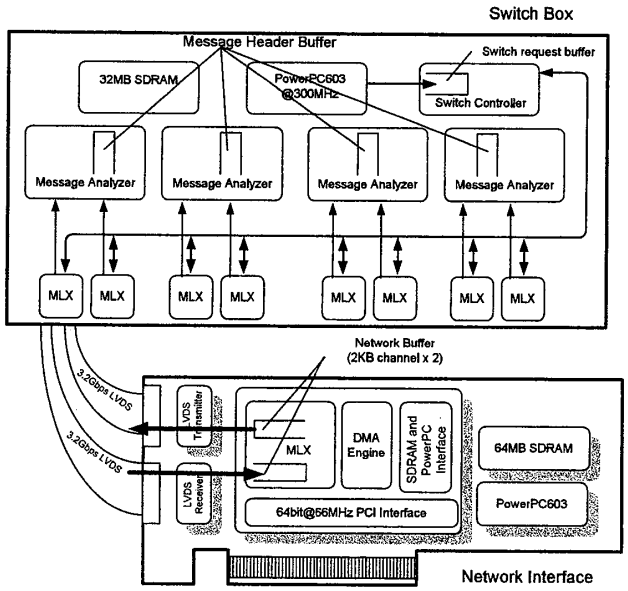


図 3 Maestro2 cluster network の構成
Fig. 3 Organization of Maestro2 cluster network.

スイッチボックスの構成を述べる．

（1） ネットワークインタフェース

ネットワークインタフェースは，LVDS transmitter/receiver チップ，Xilinx Virtex-II FPGA チップ [21]，PowerPC603e [22] 300 MHz と 64 MB SDRAM で構成される．8 kByte のネットワークバッファを含む MLX モジュール，DMA エンジン，PCI バスインタフェースは Virtex-II FPGA チップに実装される．

PCI バスインタフェースは PCI バス，SDRAM，MLX モジュール，PowerPC，DMA エンジンと接続され，ホストプロセッサと通信を行い，ネットワークバッファ，SDRAM とホストプロセッサとの通信を仲介する．また，ホストプロセッサとのメッセージ交換を容易にするため，ホストプロセッサのアドレス空間に SDRAM とネットワークバッファをマッピングする処理を行う．PowerPC・DMA エンジンは，PCI インタフェースを通じ，ホストプロセッサのメモリに直接アクセスすることが可能である．

MLX モジュールは LVDS インタフェースと，DMA エンジン・PowerPC・PCI インタフェースの間に接続されている．MLX モジュールには **2.1.1** で述べた MLX リンクレイヤプロトコルが実装されており，双方向通信を行い，LVDS transmitter/receiver を制御する．

LVDS transmitter/receiver はスイッチボックスと LVDS ケーブルで接続され，6.4 Gbps（3.2 Gbit/s + 3.2 Gbit/s）の双方向物理媒体の性能でデータの転送を行う．MLX モジュールに含まれるネットワークバッファは，PowerPC と DMA エンジンから読み書きできる．メッセージヘッダの作成を PowerPC が行い，ネットワークバッファに書き込むことで，ホストプロセッサとは独立したリンクレイヤの管理が可能である．また，DMA エンジンがネットワークバッファとホストプロセッサのメモリとの間でメッセージの転送を行うことにより，ゼロコピー通信が行える．

（2） スイッチボックス

スイッチボックスは，8 組の LVDS transmitter/receiver チップと PowerPC603e 300 MHz，32 MByte SDRAM，Xilinx Virtex-II FPGA チップで構成される．FPGA チップには，八つの MLX モジュール，四つのメッセージアナライザ，スイッチバス，スイッチコントローラが実装される．

8 組の LVDS transmitter/receiver は，それぞれネットワークインタフェースと LVDS ケーブルで接続され，メッセージの転送を行う．MLX モジュールは，ネットワークインタフェースと同じものを搭載している．メッセージアナライザは MLX モジュールに接続され，ネットワークインタフェースから受信したメッセージの先頭のパケットに書かれたメッセージヘッダを取り出し，PowerPC へと転送する．

PowerPC は四つのメッセージアナライザとスイッチコントローラに接続されており，メッセージアナライザを巡回してヘッダが到着しているかどうかを調べ，到着していればヘッダを受信し，スイッチコントローラへコマンドを送信する．

スイッチコントローラは，PowerPC とバスで接続されており，PowerPC からのコマンドを受け取り，コマンドに書かれた送信元から送信先へとメッセージを転送する．スイッチバスは八つの MLX モジュールを結んでおり，スイッチコントローラからの指示により MLX モジュール間でのデータ転送を行う．このバスは同時に複数の MLX モジュール上のネットワークバッファにメッセージを書き込むことができる．これにより，**2.1.2** で述べた，メッセージのコピー操作や破棄操作によるオーバヘッドを伴わないマルチキャストを実現する．

3. 高性能メッセージパッシングライブラリ MMP

通信ソフトウェア層においては，ホストプロセッサ

内での複数回のコピー操作が問題となる．広域ネットワーク向けの高度な通信制御を行う通信プロトコルソフトウェアでは，メッセージを送受信する際に，ユーザ空間に設けられたバッファと，カーネル空間内に設けられたバッファ，通信ハードウェア上のバッファの間でコピーを行い，通信を行っており [23]，大きなオーバヘッドとなって通信性能を低下させる [24]．

また，これらの通信ソフトウェアでは，ホストプロセッサが，通信ハードウェアの設定，通信フローの制御や通信エラー処理などの通信処理を行っている．特に通信ハードウェアの設定では，最新のプロセッサにおいても，デバイスからの応答を待つ際には後続の命令をブロックしなければならず，システムの性能を低下させる原因となる．更に，ホストプロセッサは，通信が発生するたびにコンテクストスイッチを伴って通信処理を行うため，通信を大量に行う場合，大きなオーバヘッドとなる．これらのオーバヘッドは，プログラム実行時間全体の 30%に及ぶ場合があり [12]，クラスタ全体の性能低下を引き起こす要因となっている．

本研究では，これらのオーバヘッドを避けるため，Maestro2 cluster network に適した通信ソフトウェア MMP を開発する．まず，MMP では，先行研究で提案されている次の技法を用いることにより，通信性能の向上を図る．

（1） メッセージパッシング方式のノンブロッキング通信インタフェース

MMP の送受信インタフェースはノンブロッキングとし，転送要求発行後直ちにアプリケーションプログラムに制御を戻すことにより，通信と計算のオーバラップを可能とする [25], [26]．

（2） ユーザモードオペレーションとゼロコピー通信

アプリケーションプログラムからの転送要求は，オペレーティングシステムのカーネルやデバイスドライバを介さず，デバイスドライバによってユーザ空間にマップされたネットワークインタフェースのメモリ領域を，ユーザ空間で動作する MMP のインタフェースライブラリが直接書き換えることで行う [4]．また，ユーザ空間で確保したバッファとネットワークインタフェースとの間で直接メッセージの転送を行う，ゼロコピー通信を実装する [13], [14]．

更に，MMP は，Maestro2 cluster network の通信性能を引き出すため，以下に述べる設計とする．

（3） ネットワークインタフェースによる転送処理

Maestro2 cluster network のネットワークインタフェース上に搭載されるプロセッサにより，ホストプロセッサとの通信要求の交換をはじめ，メッセージヘッダの付加と解析，転送対象バッファの仮想–物理アドレス変換，DMA ハードウェアの制御など，通信処理の大部分を行う．

特に，アドレス変換については，他のネットワークと通信ソフトウェアでは，そのネットワークインタフェース上に変換を支援するハードウェアを搭載し，それを用いるか [2], [3]，またはホストプロセッサで変換処理を行っている [4]．Maestro2 cluster network では，ゼロコピー通信に必要なアドレス変換機構を支援するハードウェアを搭載していないが，ネットワークインタフェース上のプロセッサが高速であるため，MMP では，通信バッファのユーザ仮想アドレスと物理アドレスをネットワークインタフェースのファームウェアに登録し，転送時にファームウェアがアドレス変換を行うことでゼロコピー通信を実現する．

また，ホストプロセッサでは，ネットワークインタフェースへの転送要求発行と，転送要求完了通知の確認のみを行う．これにより，ホストプロセッサは通信中も他の処理を行うことができる．

（4） Active zero-copy

メッセージパッシング方式の通信では，ゼロコピー通信を行うためには，転送が発生する以前にホストプロセッサで物理アドレスの取得を行い，転送要求とともにネットワークインタフェースに登録する必要がある．そのため，受信側アプリケーションプログラムが，メッセージの受信タイミングやサイズを事前に予測できない場合，送受信側がメッセージ本体を交換する前に，メッセージのサイズの情報などを交換し，その後メッセージ本体を転送する rendezvous での通信を行うか，プログラム開始時にメモリページをピンダウンし，そこに受信する eager での通信を行う必要がある．

しかし，前者の方法では，事前にメッセージサイズの情報を交換するためのレイテンシが発生し，通信性能を低下させる．後者の方法では，(1) 事前に確保したメモリページを超えるサイズのメッセージや複数のメッセージを同時に受信した場合に，受信操作が間に合わず，受信用メモリページが不足することにより，通信性能を低下させる可能性がある．(2) 想定される受信ペンディングメッセージ数に応じた容量のバッファを確保する必要があるため，メモリを大量にロックしなければならない．その結果，メモリページのスワッ

プ回数が増加し，システム全体の性能低下を招く．

MMP では，これらの性能低下の可能性に対して，ネットワークインタフェースが主体となってゼロコピー通信を行う，active zero-copy で対処する．

active zero-copy では，ネットワークインタフェースは，アプリケーションプログラムからの転送要求がない場合や，受信用メモリページが登録されていない場合，メッセージを受信すると，ホストプロセッサへの割込みを発生させ，ホストプロセッサに受信用メモリ領域の確保を促す．割込みを受けたホストプロセッサでは，受信用メモリ領域を確保したのち，そのアドレスをネットワークインタフェースに登録する．その後，ネットワークインタフェースは，受け取ったアドレスに従って，ホストプロセッサのメインメモリへのメッセージの転送を行う．このように，受信側ネットワークインタフェースが，受信先メモリ領域の確保から，確保したメモリ領域への DMA による書込みまでの一連の受信処理を，ホストプロセッサと協調して実行することで，メッセージのサイズ情報などの交換や，事前にメモリ領域を確保しておくことなしに，ゼロコピーでメッセージの受信を行うことができる．

また，特に，送信側がサイズの一定しないメッセージを連続して送る場合，事前のメッセージ情報の交換なしに転送が開始できるため，送信側の送出処理，受信側での受信メモリ領域の確保と受信処理が，シリアライズされることなく，それぞれがオーバラップして実行される．そのため，受信側でのメモリ領域の確保に伴うオーバヘッドを隠ぺいでき，高速にメッセージの交換ができると考えられる．

このように，active zero-copy ではネットワークインタフェースが主体となって受信操作を行うため，ネットワークインタフェース上で実行される受信処理が複雑となる．MMP では，ネットワークインタフェース上の高速プロセッサを利用することで，active zero-copy の効率良い実装を図っている．

（5） ネットワークベースの同期処理

並列アプリケーションにおける同期処理に伴う，通信量と通信処理のステップ数を低減するために，MMP では同期処理についても，ネットワークインタフェースとスイッチボックスがホストプロセッサと独立して行う．すなわち，ネットワークインタフェースとスイッチボックスに搭載されたプロセッサが，自律的にメッセージを交換し，ホスト間の同期を行う．したがって，ホストプロセッサ自身は同期処理をネットワークインタフェースに依頼するのみでよい．また，メッセージ交換を行うスイッチボックスが同期のメッセージを処理することにより，少ないメッセージの転送で高速に同期を行うことが可能である．

4. MMP の実装

MMP は，ホストプロセッサで動作するインタフェースライブラリ，ネットワークインタフェース上で動作するファームウェア，スイッチボックス上で動作するファームウェアから構成される．

インタフェースライブラリとネットワークインタフェース上のファームウェアは，アプリケーションプログラムからの要求をノンブロッキングで処理するために，図 4 に示すように，ネットワークインタフェース上のメモリに request queue，ホストプロセッサのメモリに request status buffer を作成し，これらを用いて非同期に情報を交換する．

4.1 インタフェースライブラリ

インタフェースライブラリは，アプリケーションプログラムからの要求を受け，転送用バッファのピンダウン，ファームウェアへの転送要求の発行を行う．インタフェースライブラリが提供するインタフェース関数のうち，代表的なものを以下に示す．

- MMP_Pindown_memory 関数

アプリケーションがゼロコピー通信を行うためには，送信するメッセージを含むメモリ領域やメッセージを受信するメモリ領域をピンダウンし，ネットワークインタフェースにその領域の物理アドレスを知らせなければならない．`MMP_Pindown_memory` 関数は，指定されたユーザ仮想アドレスが含まれるページのピンダウンをデバイスドライバに依頼し，物理アドレスを得る．そして，取得した物理アドレスを，ユーザ仮想アドレ

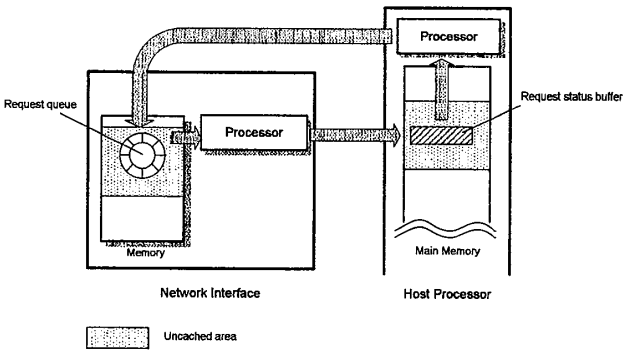


図 4 MMP での要求処理
Fig. 4 Request processing on MMP.

スと組にして，ネットワークインタフェースのファームウェアに登録する．これにより，転送要求時には，ネットワークインタフェースにユーザ仮想アドレスのみ通知すれば，転送が可能となる．

- MMP_Send, MMP_Recv 関数

MMP_Send, MMP_Recv 関数は，アプリケーションプログラムから呼び出されると，送信・受信要求をネットワークインタフェース上の request queue に書き込み，ファームウェアにメッセージの転送を指示する．送信・受信要求には，送受信先ネットワークアドレス・送受信に用いるメモリ領域のアドレス情報が含まれる．受信を行う場合，MMP を用いたアプリケーションプログラムは，メッセージを受信する前に MMP_pindown_memory 関数を用いてメモリ領域をピンダウンし，MMP_Recv 関数を用いて受信要求を発行しなければならない．しかし，メッセージが到着するタイミングや，メッセージサイズが不定な場合，“ANY” キーワードを MMP_Recv 関数の引数に渡すことで，ファームウェアに active zero-copy での受信を指示することができる．

- MMP_Wait_send, MMP_Wait_recv 関数

3. で述べたとおり，MMP_Send, MMP_Recv 関数はノンブロッキングでの送受信操作を提供する関数であるので，これらの関数の終了時には通信の完了は保証されない．MMP では，通信の完了を確認するための関数として，MMP_Wait_send, MMP_Wait_recv 関数を提供する．これらの関数は，アプリケーションプログラムから呼び出されると，request status buffer に書き込まれている転送完了情報を確認し，通信が完了していれば，アプリケーションプログラムに処理を戻す．

- MMP_Sync 関数

MMP_Sync 関数は，同期に参加するホスト情報を引数とし，複数ホスト間での同期を行う．アプリケーションプログラムから呼び出されると，同期要求をネットワークインタフェース上の request queue に書き込み，ファームウェアに同期処理を指示する．

4.2 ネットワークインタフェース用ファームウェア

ネットワークインタフェース用のファームウェアは，転送要求バッファから要求情報を読み出し，ネットワークインタフェース上の DMA エンジン，ネットワークバッファを操作して，転送を行う．

メッセージを送信する場合は，送信要求に従って，メッセージヘッダを作成し，ネットワークに送出する．その後，送信要求に書き込まれた，アプリケーションプログラムの用意したバッファのユーザ仮想アドレスを，対応する物理アドレスに変換し，DMA エンジンに設定して転送を行う．

また，受信要求をホストプロセッサから受けた場合は，要求をいったんプロセッサのキャッシュが有効なメモリ領域にコピーする．そして，ネットワークからメッセージを受信した際，メッセージヘッダを読み出し，コピーした要求から適合する受信要求を検索する．プロセッサのキャッシュ領域にコピーすることで，受信する可能性のあるメッセージが複数あり，いくつかの受信要求が発行されている場合においても，高速に受信要求を検索することができる．検索の結果，受信要求が存在した場合は，送信の場合と同様，受信要求に書き込まれたユーザ仮想アドレスを，物理アドレスに変換し，DMA エンジンを制御して転送を行う．

更に，受信要求に “ANY” キーワードが含まれている場合には，active zero-copy を行うため，ホストプロセッサへの割込みを発生させ，ホストプロセッサに受信バッファの確保を要求する．その後，ホストプロセッサから確保されたバッファの物理アドレスを受け取り，そのアドレスに転送する．

4.3 スイッチボックス用ファームウェア

スイッチボックス用ファームウェアは，ネットワークインタフェースから受信したメッセージのヘッダを解析し，メッセージの転送をスイッチハードウェアに指示する．同期メッセージを受信した場合は，同期に参加する他のネットワークインタフェースからも同様の同期メッセージが届くのを待ち，すべての同期ホストからのメッセージが届いたら，同期に参加したすべてのネットワークインタフェースに同期完了メッセージをマルチキャストし，同期の完了を通知する．

4.4 通信の流れ

MMP を用いた場合の通信の流れを図 5 に示す．

最初に，送信側ホストプロセッサ・受信側ホストプロセッサで，送受信を行うバッファをピンダウンし，物理アドレスとユーザ仮想アドレスをネットワークインタフェースに登録する（図 5 (a), (a′)）．次に，送信側ホストプロセッサがネットワークインタフェースに，送信先アドレス情報と，送信バッファのアドレス情報を含んだ送信要求を書き込む（図 5 (b)）．受信側ホストプロセッサは，送信元アドレス情報と受信先バッファのアドレス情報を含む，受信要求をネットワークインタフェース上の要求バッファに書き込む（図 5 (c)）．送信元ネットワークインタフェース上のプロセッサは，

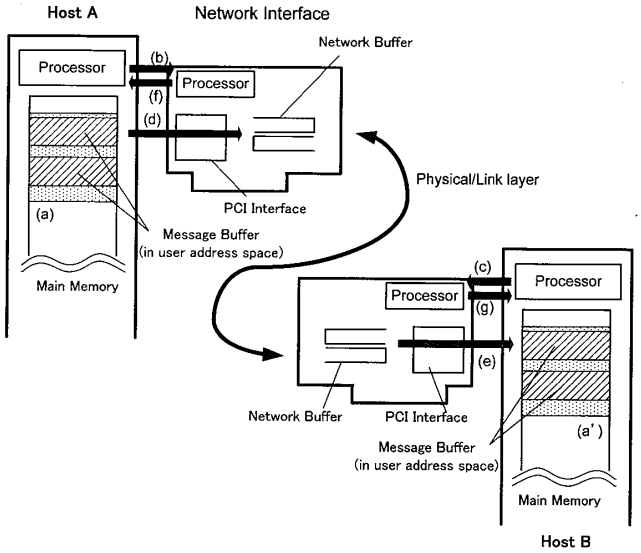


図 5 MMP での通信の流れ
Fig. 5 Message flow with MMP.

書き込まれた送信要求を読み出し，ネットワークバッファにメッセージヘッダを書き込む．次に，送信要求に書き込まれたユーザ仮想アドレスから，それに対応する物理アドレスを求め，DMA エンジンを制御して，ホストプロセッサのメインメモリからネットワークバッファにメッセージ本体を転送する（図 5 (d)）．

受信側ネットワークインタフェースでは，メッセージの到着を認識すると，まずプロセッサがメッセージヘッダを読み出し，受信要求と照合する．そして，適合する受信要求が存在した場合は，受信要求に含まれる転送先バッファのユーザ仮想アドレスから，対応する物理アドレスに変換し，DMA エンジンを制御して，ネットワークバッファからホストプロセッサのメインメモリに転送する（図 5 (e)）．最後に，送信側，受信側とも，すべてのメッセージ本体を転送し終わると，ネットワークインタフェース上のプロセッサが，メインメモリ上の送受信ステータスバッファを転送完了に変更し，ホストプロセッサに転送完了を通知する．(図 5 (f),(g))

5. 性能評価

Maestro2 cluster network と MMP の性能を測定するため，表 1 に示す実験環境を構築した．この実験環境上で通信を行い，通信時間とファームウェアの処理時間を測定する．

評価は，(1) MMP の実効性能，(2) ホストプロセッサがネットワークインタフェースの制御を行い，二つのホスト間で通信した場合と，MMP を用いて二つのホスト間で通信した場合の性能の比較，(3) MMP を用いて同期を行った場合の所要時間，(4) active zero-copy の性能，(5) MMP で通信を行った場合のファームウェアでの通信処理に要する時間，について行う．

表 1 実験環境
Table 1 Experimental environment.

ホスト PC	Intel Xeon 2.8 GHz Intel E7520 PC3200 ECC DDR2 SDRAM 1 GB
OS	Debian GNU/Linux 3.1

(1) 及び (2) の評価は，(a) 二つのネットワークインタフェース上のメモリ間で通信した場合，(b) ホストプロセッサがネットワークインタフェースの制御を行い，二つのホスト間で通信した場合，(c) MMP を用いて二つのホスト間で通信した場合，の通信性能を測定し，(a) 及び (b) での性能と，(c) の性能を比較する．

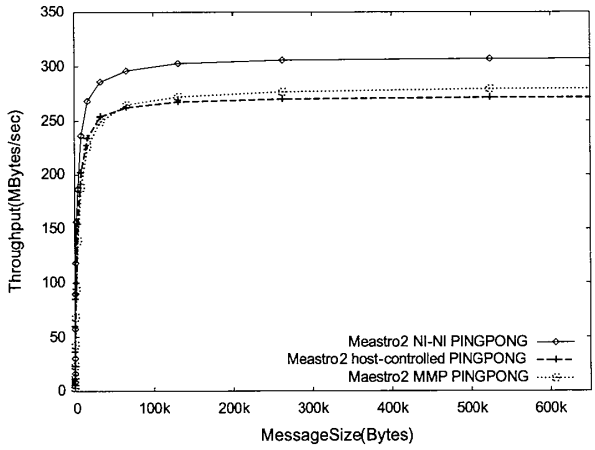
(4) の評価では，メッセージパッシング方式の通信で広く用いられている，eager と rendezvous の組合せによる通信と，active zero-copy との比較のため，それぞれランダムなサイズのメッセージを転送し，単位時間当りに転送できるメッセージ数を比較する．

5.1 ネットワークインタフェース間通信性能との比較

まず，Maestro2 cluster network の，ネットワークインタフェース間ピーク性能に対する MMP の性能を確認するため，二つのネットワークインタフェース上のメモリ間で単方向通信と双方向通信を行い，MMP を用いて二つのホストプロセッサのメインメモリ間で同様の転送を行った場合と性能を比較した．ネットワークインタフェース間の通信性能の測定では，ネットワークインタフェース上のプロセッサを用いて，DMA エンジンの設定，通信完了の確認と時間計測を行った．MMP を用いた通信性能の測定では，ホストプロセッサの `gettimeofday` 関数を用いて時間計測を行った．

単方向通信の評価では，一方のメモリから，他方のメモリにメッセージを送信し，メッセージを受信したネットワークインタフェースまたはホストプロセッサは，すべてのメッセージを受信し終わるまで待ってから，送信元のメモリに受信したメッセージをそのまま送り返す通信を行った．双方向通信の評価では，最初に 2 者間で 32 Byte のメッセージを送り合って同期をとった後，お互いに同時に同サイズのメッセージを送信する通信を行った．

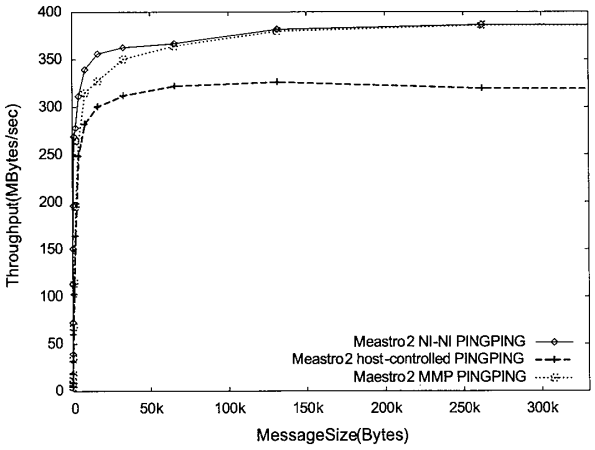
メッセージのサイズを 32 Byte から 2 MByte まで



Message size(bytes)	32	64	128	256	512	1024
NI-NI latency(us)	3.5	3.8	4.0	4.6	5.6	7.7
host-controlled Maestro2 latency(us)	5.5	5.5	5.5	7.0	8.5	12.0
MMP latency(us)	9.0	11.0	11.0	11.0	12.0	15.0

図 6 Maestro2 cluster network の単方向通信での性能比較

Fig. 6 Performance comparison of uni-directional communication with Maestro2 cluster network.



Message size(bytes)	32	64	128	256	512	1024
NI-NI latency(us)	3.8	3.9	4.1	4.6	5.8	8.2
host-controlled Maestro2 latency(us)	14.0	14.0	14.0	16.0	17.0	20.0
MMP latency(us)	14.0	14.0	15.0	15.0	15.0	18.0

図 7 Maestro2 cluster network の双方向通信での性能比較

Fig. 7 Performance comparison of bi-directional communication with Maestro2 cluster network.

変えながら，単方向通信と双方向通信を行ったときのレイテンシとスループットを，それぞれ図 6 と図 7 に示す．

ネットワークインタフェース間の単方向通信では，二つのネットワークインタフェース間での最小レイテンシは，32 Byte 転送時に約 3.5 μs であった．また，最大スループットは約 305 MByte/s であった．双方向通信では，二つのネットワークインタフェース間での最小レイテンシは，約 3.8 μs であり，最大スループットは約 387 MByte/s であった．

一方，MMP を用いた二つのホストプロセッサのメインメモリ間の通信では，単方向通信時において，最小レイテンシは約 9 μs，最大スループットは約 281 MByte/s であった．また，双方向通信では，最小レイテンシは約 14 μs であり，最大スループットは約 386 MByte/s であった．

この結果から，MMP は単方向通信では二つのネットワークインタフェース間の 92%のスループット，双方向通信では二つのネットワークインタフェース間のスループットと同等のスループットを達成しており，Maestro2 cluster network の基本性能を維持した通信を行えることが確認できた．

5.2 ホスト制御による通信との比較

次に，MMP における，ネットワークインタフェース上のプロセッサが通信制御を行う方式の有効性を確認するため，二つのホストプロセッサのメインメモリ間で，MMP を用いた方式と，ホストプロセッサがネットワークインタフェースを制御する方式のそれぞれで単方向通信と双方向通信を行い，性能を測定した．ホストプロセッサによる制御では，ネットワークインタフェース上のプロセッサは全く用いず，ネットワークインタフェースの DMA エンジンをホストプロセッサから設定し，通信の完了はホストプロセッサからネットワークインタフェースのレジスタをポーリングして確認することで行った．

メッセージのサイズを 32 Byte から 2 MByte まで変えながら，単方向通信と双方向通信を行ったときのレイテンシとスループットを，それぞれ図 6 と図 7 に示す．単方向通信では，ホスト制御による通信での最大スループットが約 273 MByte/s であったのに対し，MMP の最大スループットは，約 8 MByte/s 高い値を達成している．しかし，最小レイテンシは，MMP による通信の 9 μs に対して，ホストプロセッサの制御による通信では 5.5 μs と MMP による転送が大きくなっている．これは MMP ではホストプロセッサからネットワークインタフェースに転送要求を書き込み，それをネットワークインタフェースが読み出して処理を行うためと考えられる．

一方，双方向通信では，ホストプロセッサ制御による最大スループットは，約 326 MByte/s であった．これは，MMP による通信の最大スループットよりも約 60 MByte/s 低い値である．

単方向通信と双方向通信のどちらにおいても，MMPの最大スループットは，ホスト制御による最大スループットを上回っている．特に，双方向通信においては，MMPは大きなメッセージサイズに対しても最大スループットを維持し，ネットワークインタフェース間のスループットとほぼ同等のスループットを達成しているのに対し，ホストプロセッサによる制御では，32 kByteを超えると性能が頭打ちになっている．これは，メインメモリ上に分散している多数のメモリページを転送するために，ホストプロセッサからネットワークインタフェースのハードウェアを頻繁に制御することにより，PCIバス上に制御のためのトランザクションが多く発生し，メッセージの転送を妨げるためと考えられる．

このことを確認するために，ネットワークインタフェースのPCIインタフェースに，組込みロジックアナライザを接続し，双方向転送中のPCIトランザクションの一部を観察した．その結果，MMPにおいては，取得できた2048サイクル分の波形のうち，転送を行っていないサイクル数は497のみであったのに対して，ホスト制御の場合では，CPUからネットワークインタフェースに対する制御のためのターゲットアクセスが発生し，頻繁にバースト転送が中断されていた．その結果，転送を行っていないサイクル数は781サイクルに増加しており，PCIバスが隘路となって，性能が低下していることが確認できた．

このように，MMPではネットワークインタフェース上のプロセッサがネットワークインタフェース上のハードウェアを制御することにより，ホストプロセッサのPCIバス上に制御のためのトランザクションが発生せず，特に双方向通信において高いスループットを得られることが確認できた．この性能特性から，MMPは集合通信を用いて大量のデータを転送するアプリケーションに対して有用であると考えられる．

図8は，四つのホストプロセッサ間で，MMPを用いた場合と，ホストプロセッサからネットワークインタフェースを制御した場合で，all to all通信を行い，all to all通信に要した時間とデータサイズから算出したスループットを比較したものである．この結果から，2ホスト間での双方向通信性能と同様，集合通信においても，MMPでは，ホストプロセッサから制御した場合よりも20%以上高いスループットを得られることが分かる．このような通信を行うアプリケーションの例としては，並列FFTや，流体力学，分子動力学などに代表される科学技術分野の並列アプリケーションが挙げられる．並列FFTではデータを転置する際にall to allの通信が発生し，科学技術分野の並列アプリケーションでは，問題領域を各ホストプロセッサに分割し，計算結果をお互いに交換するという手順を繰り返すため，周期的に，かつ一度に大量のデータが集合通信で授受される．したがって，これらのアプリケーションにおいてMMPを用いた場合，高い通信性能が得られると期待できる．

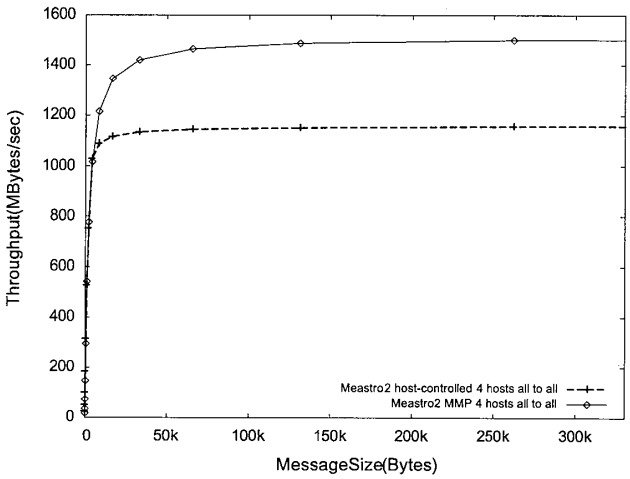


図8 All to All通信での性能比較
Fig. 8 Performance comparison of all to all communication.

以上の結果から，MMPは，ネットワークインタフェース上のプロセッサが通信制御を行うことにより，最小レイテンシはホストプロセッサが制御を行う場合よりも大きくなるものの，特に双方向通信や，双方向通信を伴う集合通信において，ネットワークインタフェース間の通信性能に極めて近い，高いスループットを得られることが確認できた．

5.3 MMPの同期性能

3. で述べたように，MMPはネットワークハードウェア上のプロセッサにより，ホスト間同期をとることが可能である．この同期性能を確認するため，複数台で同期を行い，レイテンシを測定した．このレイテンシとは，同期を1000回行い，各同期要求発行から同期完了通知までの時間の平均値である．

同期に参加するホストプロセッサの台数が，2，4，8台でのレイテンシを表2に示す．2台でのレイテンシは，MMPを用いて単方向通信を行った場合のホスト間レイテンシとほぼ同じ値となっている．また，4台，8台と同期に参加する台数を2倍，4倍に増やした際は，スイッチボックスのプロセッサの処理するメッ

表 2　MMP の同期性能
Table 2　Synchronization performance of MMP.

2 台	9.7 μs
4 台	10.7 μs
8 台	13.8 μs

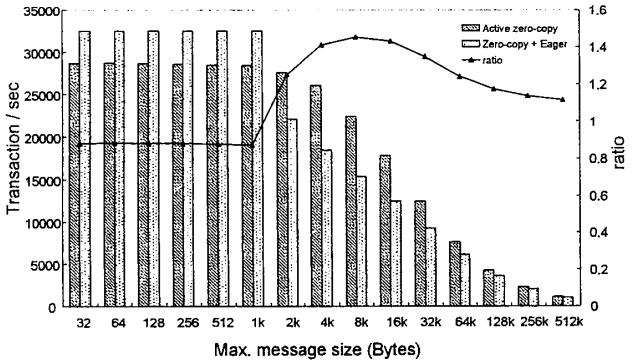


図 9　ランダムなメッセージサイズでのトランザクション数の比較
Fig. 9　Comparison of random transaction performance.

セージ数が増加することによってレイテンシが増加するが，それぞれ 1.1 倍，1.4 倍のみのレイテンシ増加で同期を行っている．

これらの結果から，ネットワーク上のプロセッサにおいて同期処理の管理を行い，同期完了をスイッチボックスからマルチキャストすることにより，短時間での複数ホスト間同期が実現できていることを確認した．

5.4　Active zero-copy の性能

active zero-copy の性能を確認するため，メッセージパッシング方式の通信で広く用いられている，サイズの小さなメッセージはコピーを伴う eager で通信を行い，大きなメッセージは rendezvous でのゼロコピー通信を行う方式と，active zero-copy を用いる方式のそれぞれで，ランダムなサイズのメッセージを 2 台のホストプロセッサの一方から他方へ連続的に転送し，単位時間当りに転送できるメッセージ数の比較を行った．

この評価においては，前者の方式では，eager と rendezvous を切り換えるしきい値を 1 kByte とし，1 kByte 未満のメッセージは eager で送受信を行い，1 kByte 以上のメッセージは，送信側がメッセージ本体を送信する前に，メッセージサイズを含んだメッセージを送信する rendezvous での通信とした．後者の方式では，メッセージのサイズにかかわらず，すべてのメッセージを active zero-copy により送受信した．この比較の結果を，図 9 に示す．

表 3　ファームウェアによるオーバヘッド
Table 3　Overheads of MMP firmware.

要求取得～初回 DMA 起動完了	1.6 μs
DMA 完了～次回 DMA 起動完了	0.5 μs
メッセージ到着確認～初回 DMA 起動完了	1.3 μs

図 9 の棒グラフは，それぞれの通信方式で 1 秒当りに受信側で受信完了したたメッセージ数，線グラフは，active zero-copy で受信側が受信完了できたメッセージ数に対する，通常のゼロコピー通信で受信側が受信完了できたメッセージ数の比を示している．また，横軸は送信側で生成する最大のメッセージサイズを示している．例えば，1 kByte のときは，1 kByte までのランダムなサイズのメッセージを，連続的に送信することを示している．

eager と rendezvous を組み合わせた方式で eager のみが用いられる 1 kByte までのメッセージサイズにおいては，active zero-copy は，eager の 0.9 倍程度のメッセージしか転送できなかったが，rendezvous での通信も併用される 1 kByte 以上のメッセージを含む通信では，active zero-copy が，最大で 1.4 倍程度のメッセージ数を通信できている．この結果から，active zero-copy は，数 kByte 以上の最大メッセージサイズで，そのサイズを予測できない通信において，必要最小限のメモリ容量で良好な通信性能を達成できるといえる．

5.5　ファームウェアによるオーバヘッド

3. で述べたとおり，MMP はネットワークインタフェース上の汎用プロセッサを用いて，通信処理の大部分を行うことを特徴の一つとしている．汎用プロセッサでソフトウェアを実行することで，ハードウェアで行う場合と比較して，容易に複雑な処理を実現できる．しかし，専用ハードウェアで処理する場合に比べ，動作速度では劣ると考えられる．そこで，MMP で通信を行った際の，ファームウェアでの処理に要した時間を測定し，ファームウェアのオーバヘッドを評価した．

表 3 は，基本性能との比較で用いた単方向通信を行った際の，ファームウェアでの通信処理に要した時間を示したものである．送信時のホストからの送信要求取得から，アドレス変換を行い，DMA エンジンを起動するまでに要した時間は，約 1.6 μs であった．これは，MMP での最小レイテンシの 18%程度のオーバヘッドである．一方，受信時のメッセージ到着から

DMA エンジンを起動するまでに要した時間は，約 1.3 μs であった．送信時に比べ，処理に要する時間が短いのは，DMA 転送中などファームウェアがアイドルになった際に，非キャッシュ領域に書き込まれた受信要求を，キャッシュ可能領域に転送して保存し直し，メッセージ到着の際にはそのキャッシュ領域から受信要求を取得しているためと考えられる．また，送信，受信時とも，複数回の DMA エンジン起動を伴う際の，前回 DMA 転送完了から，DMA エンジン再起動までに要する時間は，約 0.5 μs であった．これは，二つのネットワークインタフェース間の最小レイテンシの 14%の時間で，リンクレイヤプロトコルがメッセージを転送するのに要する時間と比べて短時間で DMA エンジンの再起動ができているといえる．これらの結果から，通信ハードウェアと密に結合したプロセッサで制御することにより，ソフトウェアでの制御でも短時間で連続して DMA 転送を行えていることが確認できた．

6. むすび

本論文では，Maestro2 cluster network の高機能性を生かし，通信性能を引き出す，メッセージパッシングライブラリ MMP の設計について述べ，性能評価を行った．

MMP では，ネットワークインタフェース上の汎用プロセッサが，ホストプロセッサに代わって通信処理の大部分を行う設計とした．

実験結果から，MMP は，ホストプロセッサがネットワークインタフェースを直接制御した場合と比較し，最小レイテンシは大きくなるものの，大幅に高いスループットを達成していることを確認した．また，同期性能の測定では，ホストプロセッサ間の単方向のレイテンシとほぼ同時間での同期を実現できていることを確認した．更に active zero-copy の評価においては，広く用いられている eager と rendezvous を組み合わせた通信方式と比較して，最大で 1.4 倍程度のメッセージ数を転送できることを確認した．これらの実験結果から，MMP は，Maestro2 cluster network に搭載した汎用の高速プロセッサを用いて複雑な送受信処理をネットワークインタフェース上で高速に実行することで Maestro2 cluster network の基本性能と同等の通信性能を達成しており，Maestro2 cluster network の通信性能を十分に引き出しているといえる．更に，ホストプロセッサに代わってネットワークコンポーネント上で通信処理を実行することにより，ホストプロセッサが通信処理を行う場合よりも高いスループットを達成し，また，通信メッセージ数や通信の手順を簡略化した通信を実現できているといえる．

今後は，転送すべきデータ領域の決定など，通常アプリケーションプログラム側で行う処理の一部を，ネットワークインタフェース上に実行委託するなど，Maestro2 cluster network の自律性を生かした並列実行環境を構築する予定である．

謝辞 本研究の一部は，科学研究費補助金基盤研究 (B)，課題番号 16300012 の助成を受けて実施された．

文献

[1] N.J. Boden, D. Cohen, R.E. Felderman, A.E. Kulawik, C.L. Seitz, J.N. Seizovic, and W.K. Su, "Myrinet — A gigabit-per-second local-area network," IEEE Micro, vol.15, no.1, pp.29–35, 1995.

[2] F. Petrini, W. chun Feng, A. Hoisie, S. Coll, and E. Frachtenberg, "The quadrics network: High-performance clustering technology," IEEE Micro, vol.22, no.1, pp.46–57, 2002.

[3] 大塚智宏，渡邊幸之介，北村 聡，原田 浩，山本淳二，西 宏章，工藤知宏，天野英晴，"分散並列処理用ネットワーク RHiNET-2 の性能評価，" 先進的計算基盤システムシンポジウム SACSIS 論文集，pp.45–52, 2003.

[4] P. Shivam, P. Wyckoff, and D. Panda, "EMP: Zero-copy os-bypass NIC-driven gigabit ethernet message passing," Proc. ACM/IEEE SC 2001 Conference (SC'01), pp.49–57, Sept. 2001.

[5] A.B. Maccabe, W. Zhu, J. Otto, and R. Riesen, "Experience in offloading protocol processing to a programmable NIC," Proc. IEEE International Conference on Cluster Computing (CLUSTER'02), pp.67–75, Sept. 2002.

[6] J.C. Mogul, "TCP offload is a dumb idea whose time has come," Proc. 9th Workshop on Hot Topics in Operating Systems (HotOS IX), pp.25–30, May 2003.

[7] A. Santoro and R.M. Fujimoto, "Off-loading data distribution mananagement to network processors in HLA-based distributed simulations," Proc. Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT ' 04), pp.12–19, Oct. 2004.

[8] R. Brightwell and K.D. Underwood, "An analysis of NIC resource usage for offloading MPI," Proc. 18th International Parallel and Distributed Processing Symposium (IPDPS'04), pp.183–191, April 2004.

[9] K. Aoki, S. Yamagiwa, M. Ono, K. Wada, and L.M. Campos, "An architecture of high performance cluster network: Maestro2," Proc. IEEE Pacific Rim Conference on Communications Computers and Signal Processing (PacRim03), pp.784–787, Aug. 2003.

[10] K. Aoki, S. Yamagiwa, K. Ferreira, L.M. Campos, M.

Ono, K. Wada, and L. Sousa, "Maestro2: High speed network technology for high performance computing," Proc. 2004 IEEE International Conference on Communication (ICC2004), no.HS01-8, June 2004.

[11] N. Bierbaum, "MPI and embedded TCP/IP gigabit ethernet cluster computing," Proc. 27th Annual IEEE Conference on Local Computer Networks (LCN'02), pp.733–734, Nov. 2002.

[12] G.F. Pfister, "An introduction to the infiniband architecture," in High Performance Mass Storage and Parallel I/O, ed. J. Hai, C. Toni, and R. Buyya, chapter 42, pp.617–632, John Wiley & Sons, 2001.

[13] Myricom, Inc, The GM Message Passing System, 1999.

[14] T. Takahashi, S. Sumimoto, A. Hori, H. Harada, and Y. Ishikawa, "PM2: High performance communication middleware for heterogeneous network environment," Proc. IEEE/ACM SC2000 Conference, pp.52–53, Nov. 2000.

[15] Infiniband Trade Association, InfiniBand Architecture Specification, Release1.0, 2000.

[16] K. Wada, S. Yamagiwa, and M. Fukuda, "High performance network of PC cluster Maestro," Cluster Computing, vol.5, no.1, pp.33–42, Jan. 2002.

[17] 山際伸一, 福田宗弘, 和田耕一, "クラスタ向けネットワークアーキテクチャとプロトコルの提案 — Maestro ネットワークの開発と性能評価," 情処学論：ハイパフォーマンスコンピューティングシステム, vol.41, no.SIG 5(HPS 1), pp.91–103, 2000.

[18] W. Yu, D. Buntinas, and D.K. Panda, "High performance and reliable NIC-based multicast over Myrinet/GM-2," Proc. 2003 International Conference on Parallel Processing (ICPP'03), pp.197–205, Oct. 2003.

[19] Myricom, Inc., Myrinet FAQ, 2003, http://www.myri.com/fom-serve/cache/109.html

[20] IEEE Standard Department, IEEE Standard for Low-Voltage Differential Signals (LVDS) for Scalable Coherent Interface (SCI), 1996.

[21] Xilinx Inc., Virtex-II Platform FPGA Data Sheet, 2002, http://www.xilinx.com

[22] Motorola, MPC603e & EC603e Microprocessor User's Manual, 1997.

[23] W.R. Stevens, B. Fenner, and A.M. Rudoff, Unix Network Programming, vol.1, 3rd ed., Addison-Wesley, 2003.

[24] V. Karamcheti and A. Chien, "Software overhead in messaging layers: Where does the time go?," Proc. Sixth Symposium on Architectural Support of Programming Languages and Operating Systems (ASPLOS-VI), pp.51–60, Oct. 1994.

[25] S.B. Baden and S.J. Fink, "Communication overlap in multi-tier parallel algorithms," Proc. ACM/IEEE SC 1998 Conference (SC'98), Nov. 1998.

[26] D.E. Culler, A.C. Arpaci-Dusseau, S.C. Goldstein, A. Krishnamurthy, S. Lumetta, T. von Eicken, and K.A. Yelick, "Parallel programming in Split-C," Proc. supercomputing'93, pp.262–273, Nov. 1993.

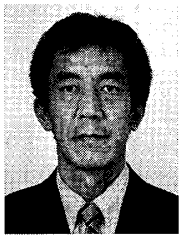
（平成 17 年 7 月 5 日受付，11 月 6 日再受付）

青木　圭一

2002 筑波大・第三学群情報学類卒．現在，同大大学院システム情報工学研究科コンピュータサイエンス専攻在学中．修士（工学）．並列・分散処理に関する研究に従事．情報処理学会学生員．

山際　伸一

2002 筑波大大学院博士課程工学研究科了．博士（工学）．2002 よりポルトガル PDM&FC 社シニア・コンサルタント．ポルトガル共和国立産業研究所 INESC-ID 客員研究員を兼務．



和田　耕一　（正員）

1978 神戸大・工・電気卒．1984 同大大学院システム科学専攻博士課程了．学博．同年同大大学院自然科学研究科助手．1987 筑波大学電子・情報工学系講師．助教授を経て 1999 教授，現在に至る．1992～1993 カナダ，ビクトリア大学客員研究員．並列・分散処理とコンピュータアーキテクチャに関する研究に従事．IEEE，ACM，情報処理学会各会員．

小野　雅晃

1981 芝浦工大・通信工卒．現在，筑波大学システム情報工学等支援室（装置開発班）技術専門職員．