2005－HPC－103（18）
2005／8／4

高性能通信処理オフロードエンジンUZURA実現に向けて

中島耕太[†]　佐藤充[†]　住元真司[†]
久門耕一[†]　石川裕[††]

クラスタシステムで使用される MPI 通信ライブラリや NFS 分散ファイルシステムなどが必要とする低レベル通信機構について議論した後、高性能通信処理オフロードエンジン実現に向けて、メッセージ通信と RDMA 通信の基本機能の実現方式について検討する。PCI-X と FPGA を搭載した 10Gb Ethernet ネットワークカード UZURA による予備評価では、ホストと NIC におけるデータ転送は、転送サイズによらず DMA 通信機能を用いた方が NIC のレジスタを CPU が読み書きするよりも高性能である。割り込み遅延が 4.99μs と大きいために、NIC からホストへの通知は割り込みを使わずにホストによるポーリング方式を採用する。UZURA 上の予備評価では、通信片道遅延は、4.03μs、スループットは、4KB 転送の場合で最大 1.02GB/s となる。

Toward a Implementation of High Performance Communication Offload Engine UZURA

Kohta NAKASHIMA,[†] Mitsuru SATO,[†] Shinji SUMIMOTO,[†]
Kouichi KUMON[†] and Yutaka ISHIKAWA[††]

In this paper, we discuss a low level communication facility to realize the MPI library and NFS distributed file system efficiently, and discuss the implementation of the message transfer and remote DMA mechanisms toward the implementation of a high performance transfer offload engine. In preliminary performance evaluation using UZURA which is a 10Gb Ethernet network adapter with PCI-X and FPGA, DMA data transfer is faster than CPU read and write registers on NIC regardless of data size and the latency of interrupt is very large(4.99μs). So we decide to implement the data transfer mechanism using DMA and the notification mechanism using polling. As a result, our preliminary performance evaluation shows the proposed communication mechanism can achieve 4.03μs latency and 1.02GB/s of throughput with 4KB transfer.

1. はじめに

近年、計算機や通信路の高性能化と低価格化が進んでいる。多数の計算機を高速なネットワークで接続したクラスタシステムがさかんに利用されるようになった。特にクラスタシステムは、科学技術計算やシミュレーション計算の分野において広く利用されており、今後ますます普及が進むと考えられる。

科学技術計算やシミュレーション計算といったHPCアプリケーションをクラスタ上で実行する際には、一般に、処理すべき問題をノード毎に分割し割り当てて計算し、計算途中のデータを交換しながら計算を行う。また、各ノードでは、並列分散ファイシステムを用いた入出力操作が伴う。したがって、全体性能を向上させるためには、通信時間を削減することが重要となる。クラスタ内のノード間通信をソケットや TCP/IP といった標準的な通信方式で実現すると、十分な性能を発揮できないため、低遅延を実現するメッセージ通信と高スループットを実現する RDMA 通信を実装する必要がある。しかし、MPI 通信ライブラリや NFS 分散ファイルシステムを効率良く実装する観点から既存通信ハードウェア機構が設計されていないために、RDMA 通信機構を効果的に利用できていない。

我々は、MPI 通信ライブラリや NFS 分散ファイルシステムなどクラスタシステムで使用される通信アプリケーションを効率良く実装する機構を提供するネットワークアダプタの研究開発を行なっている。このために、FPGA 搭載の 10Gb Ethernet ネットワークアダプタ UZURA を開発し、FPGA 上にプロトコルオフロード機能を実装している。

本稿では、まず、MPI 通信ライブラリや NFS 分散ファイルシステムが必要とする通信機構について議論した後、高性能通信処理オフロードエンジン実現に向

† (株) 富士通研究所
Fujitsu Laboratories
†† 東京大学
the University of Tokyo

けて、メッセージ通信と RDMA 通信の基本機能の実現方式について検討する。

2. 課　　題

2.1 MPI 通信ライブラリ

MPICH に代表される MPI 通信ライブラリの実装では、メッセージのサイズによって、Eager プロトコルや Rendezvous プロトコルを切替えることができる。Randevous プロトコルでは、送信側と受信側で同期が行なわれる。同期時に、受信バッファアドレスを送信側に送ることにより、RDMA 通信が実現される。RDMA 通信により、CPU の介在によるデータコピー操作なしに、送信側アプリケーションのデータを受信側アプリケーションのメモリ領域に対して直接転送される。

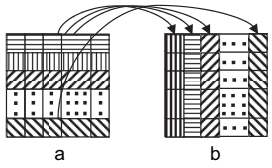
MPI では、MPI_Type_vector や MPI_Type_hvector などの機能を利用して、ユーザタイプを定義することができる。本機能を使った C 言語による配列の転置プログラム例を図 1 に示す。行方向にまとめたデータタイプ row を定義し、さらに row データタイプを用いて配列全体を表現するデータタイプ xpose を定義している。送信側は xpose データタイプを用いてデータを送信し、受信側は MPI_REAL タイプで受信する。C 言語における配列は列方向から配置されるため、配列転置が実現される。本例では、送信側において、断片化されたメモリ領域が転送され、受信側で連続メモリ領域にコピーされている。受信側もこのようなユーザデータタイプが使用されると、送受信において断片化されたメモリ領域の転送が必要となる。

PMv2[1] で提供されているような一つの連続したメモリ領域の RDMA 通信機能では、断片毎に NIC に命令を送る必要がありオーバヘッドが増す。このため、断片化されたメモリ領域を一回連続領域にコピーした上で、その領域を送信することになる。

さらに、MPI-2 の片側通信機能では、MPI_PUT、MPI_GET、MPI_Accumulate を呼び出したプロセスのデータタイプを相手ランクに送らなければならない。図 2 の例では、rank0 が MPI_Get 関数を呼び出すと、rank1 の配列 b から rank0 の a に転置されたデータがコピーされる。

ここで注意しなければならないのは、xpose データタイプが rank1 で組み立てられていない場合でも、rank1 側で xpose データタイプを処理しなければならない点である。このため、片側通信においてユーザデータタイプが使われる場合には、ユーザタイプをリモート側で構築できるだけの情報をリモートプロセス側に送信する必要がある。

このように、ユーザが定義したデータタイプのデータを高速に転送するためには、単純に断片化されたメモリから断片化されたリモートメモリへの転送機能だけでなく、データの再配置情報をリモート側で解釈実行する機能が必要となる。また、プロセッサヘテロ環境における MPI 通信を想定する場合には、基本データタイプの変換を行なう機能が必要となる。




```
MPI_Type_extent(MPI_REAL, &sz);
MPI_Type_vector(100, 1, 100, MPI_REAL, &row);
MPI_Type_hvector(100, 1, sz, row, &xpose);
MPI_Type_commit(&xpose);
MPI_Sendrecv(a, 1, xpose, rank0, 0,
             b, 100*100, MPI_REAL, rank1, 0,
             MPI_COMM_WORLD, &status);
```

図 1　ユーザデータタイプ機能による配列転置

```
MPI_Win_create(b, 100*100*sizeof(float), 0, 0,
               MPI_COMM_WOLD, &win);
...
MPI_Get(a, 1, MPI_REAL, rank1, 0, 1, xpose, win);
```

図 2　MPI_Get 使用例

2.2 NFS と iSCSI

NFS や iSCSI での Read/Write 処理では、クライアント/サーバ間で大量のデータ交換処理を行う。これらのデータをリモートへ転送する際は、ヘッダとデータ本体から構成されるパケットを転送する。このデータ本体は、クライアント/サーバのメモリ上に断片的に配置されているため、パケット生成/分解をホスト上で実行する場合、データコピーが生じる (図 3)。このコピー処理を削減するためには、送/受信の両方において Scatter/Gather(S/G) 処理を NIC がサポートする必要がある。

S/G 処理を NIC がサポートする場合には、送信側においては、S/G リストをホストが生成し、これに基づいて NIC がヘッダと断片化されたデータ本体を DMA 転送し、NIC 上でパケット生成を行う。したがって、送信側処理においては、NIC が S/G 処理機能を提供すれば、ホスト上でのコピー処理のオーバヘッドを削減できる。

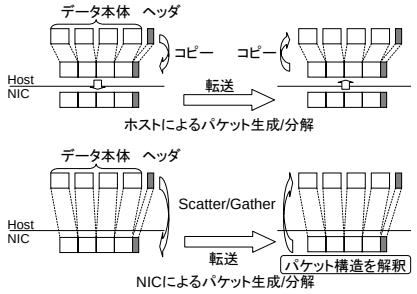


図 3　パケット生成分解処理

受信側においては、受信パケットの NFS や iSCSI のヘッダを NIC 上で解釈し、パケット構造に基づいてホストへデータ本体を DMA 転送する必要がある。このように、受信処理においては、単に S/G 処理を NIC がサポートするだけでなく、受信パケットのヘッダを NIC 上で解釈し、これに基づいた転送処理を実行する機能が必要になる。

このように、MPI、NFS、iSCSI での通信処理を高速化するためには、ホスト上の処理のオーバヘッドを削減する機能を持つ NIC が必要である。

3. UZURA

本稿で実現しようとしている高性能通信は、現在我々が開発中の実験用ハードウェア UZURA の上に実装する。そこで、本章では実装対象となるハードウェア UZURA の概要と基本性能値について述べる。

3.1 UZURA 概要

UZURA は 10 Gigabit Ethernet (10GbE) を用いたクラスタ内ノード間通信用の実験用 NIC である。UZURA の主な部分は、メインロジックである FPGA と 10GbE MAC 処理を行う LSI から構成される (図 4)。この FPGA 上に独自の回路を実装することにより、ハードウェア上で様々な機能を実現できる。

ホストと FPGA の間は PCI-X バスで接続されており、最大データ転送性能は 1066MB/s である。また、FPGA と 10GbE MAC LSI との間は POS-PHY Level 4 (PL4) インタフェースで接続されており、最大データ転送性能は 1360MB/s である。

また、UZURA には高速なアドレス変換などのために必要な SRAM を 4MB 搭載している。FPGA からこの SRAM を読み出す遅延時間は、35.3ns である。

3.2 基本性能値

高性能通信の設計を行う上で必要となる、パケット転送、PCI-X バス性能、割り込み性能について測定した。以降、これらの性能値について述べる。

3.2.1 パケット転送

UZURA 上のパケット転送時間を測定した (表 2)。

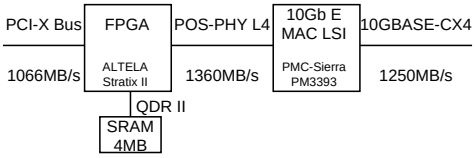


図 4 UZURA の構成図

表 1 測定環境

CPU	Xeon 3.4GHz
Chipset	Intel E7520
Memory	2GB(DDR2-400)
I/O Bus	PCI-X(133MHz)

表 2 パケット転送時間

処理	時間 (μs)
送信側 FPGA 処理 (PL4 以外)	0.09
送信側 FPGA 処理 (PL4)	0.16
10GbE MAC 処理 (送受信合計)	1.56
受信側 FPGA 処理 (PL4)	0.35
受信側 FPGA 処理 (PL4 以外)	0.15
合計	2.31

表 3 レジスタアクセス時の 1 word 分のバス上の転送時間

Write/Read	時間 (ns)
Write	97.5
Read	90.0

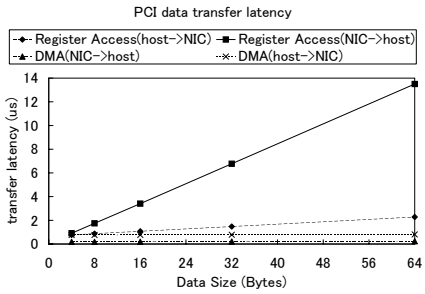


図 5 データ転送時間

送信側のホストから UZURA へのデータ転送完了直後から、受信側の UZURA からホストへのデータ転送開始直前までの転送時間を測定した。また、FPGA 処理の内訳は、シミュレーションにより算出した。

3.2.2 PCI-X バス性能

ホスト-NIC 間のデータ転送時間とデータ転送時の PCI-X バス上の転送時間を、表 1 の測定環境を用いて測定した。

ホスト-NIC 間のデータ転送時間 ホストから NIC 上レジスタへの Read/Write 時間と NIC からホストメモリへの DMA 転送時間を測定した (図 5)。いずれも、データ転送を起動してから実際に転送先にデータが到達するまでの時間を測定した。

PCI-X バス上の転送時間 PCI-X バスアナライザを用いて、PCI-X バス上の転送時間を測定した。ホストから NIC 上レジスタへの Read/Write 時間を表 3 に、NIC からのホストメモリへの DMA 転送時間を図 6 に示す。

3.2.3 割り込み遅延

表 1 の測定環境を用いて、割り込み遅延を測定した。測定は、NIC 上のレジスタ Write を契機に割り込みを発生させるロジックを FPGA に実装し、割り込みを発生させてから割り込みハンドラが動作するまでの時間を計測した。NIC 上のレジスタ Write 時間を差し引いた割り込み遅延時間は、4.99μs である。

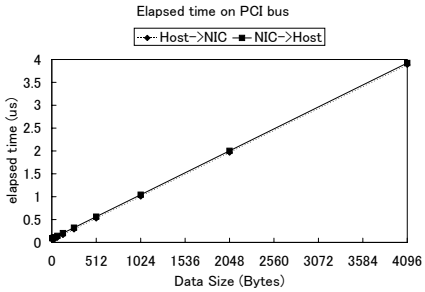


図 6 PCI バス上のデータ転送時間

4. メッセージ通信と RDMA 通信の設計

第 2 章において、MPI、NFS、iSCSI を高速に実現する通信機構が実現すべき機能について述べたが、本稿では、まず、MPI 通信のベースとなるメッセージ通信と RDMA 通信の設計を議論する。本章では、メッセージ通信と RDMA 通信の設計について述べる。

ハードウェアを含めた通信方式の設計を行う際には、ホスト上の処理と NIC 上の処理の機能分担が最も重要なポイントとなる。そこで、メッセージ通信と RDMA 通信の各処理要素について、基本性能に基づき、ホスト上の実装と NIC 上の実装の利害得失を定量的に検討し、実装方式を決定した。以降、メッセージ通信と RDMA 通信の各処理要素と実際のデータを元に検討した結果について述べる。

4.1 メッセージ通信と RDMA 通信の処理要素

メッセージ通信と RDMA 通信の処理フローを図 7 に示す。図 7 の各処理は、以下の処理に分類できる。

- NIC-ホスト間データ転送
- NIC-ホスト間の通知 (転送起動、到着通知)
- パケット生成/分解
- パケット転送
- 論理物理アドレス変換

このうち、パケット転送に関しては、NIC 上でのみ実装可能である。そこで、以降、パケット転送以外の各処理について、ホストと NIC の機能分担をメッセージ通信と RDMA 通信の各場合について検討する。

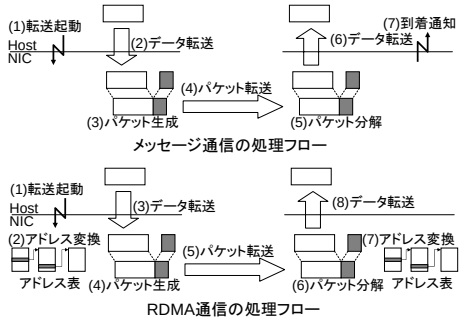


図 7 メッセージ通信と RDMA 通信

表 4 通知処理の比較

通知方向	方式	遅延 (μs)	バス使用
ホスト	レジスタ通知	0.78	通知時のみ
→ NIC	ホストメモリポーリング	0.76	連続使用
NIC	割り込み	4.99	無し
→ホスト	レジスタポーリング	0.91	連続使用
	ホストメモリ通知	0.18	通知時のみ

4.2 各処理の実装方式の検討

4.2.1 NIC-ホスト間データ転送

NIC-ホスト間でのデータ転送方式には、NIC 上の DMA コントローラがデータ転送する DMA 方式と、ホストが NIC 上のレジスタを読み書きする方式がある。図 5 によると、NIC-ホスト間のデータ転送は、DMA 方式が高速であるため、メッセージ通信と RDMA 通信の両方で DMA 方式を採用する。

4.2.2 NIC-ホスト間の通知

ホスト→ NIC 通知には以下の方式がある。

- レジスタ通知方式
  ホストが NIC 上レジスタを Write し通知
- ホストメモリポーリング方式
  NIC がホストメモリをポーリングし通知

NIC →ホスト通知には以下の方式がある。

- 割り込み方式
- レジスタポーリング方式
  ホストが NIC 上レジスタをポーリングし通知
- ホストメモリ通知方式
  NIC がホストメモリに Write し、この領域をホストがポーリングし通知

各方式の特徴を表 4 に示す。このうち、ホストメモリポーリング方式とレジスタポーリング方式は、PCI バス経由でポーリングするため、ポーリングの間、PCI バスを連続的に使用し、データ転送のスループットに大きく影響を与える。また、割り込み方式の遅延時間は、4.99μs と大きい。そこで、メッセージ通信と RDMA 通信の両方において、ホスト→ NIC 通知にはレジスタ通知方式を、NIC →ホスト通知にはホストメモリ通知方式を採用する。また、メッセージ通信の到着通知処理時間をさらに削減するため、転送データを直接ポーリングする方式を採用する。

4.2.3 パケット生成/分解

RDMA 通信では、ホストメモリと NIC 間で直接 DMA 転送するため、NIC 上でパケット生成/分解処理を行う。

メッセージ通信では、小さいメッセージであれば、パケット生成/分解処理の実行は、ホスト上でも NIC 上でも大きな性能差はない。しかし、NIC 上に実装する RDMA 用のパケット生成/分解処理の大部分を共用できるため、メッセージ通信のパケット生成/分解処理を実現できるため、NIC 上に実装する。

4.2.4 論理物理アドレス変換

論理物理アドレス変換については、変換表と変換表の管理処理を NIC 上に実装する場合とホスト上に実装する場合について検討する。

変換表の実装箇所 変換表を参照するのは、NIC であるので、NIC 上に実装すると、1 回の参照レイテンシは 0.04μs と小さい。しかし、NIC 上に実装できる変換表の総量は、最大 4MB であるため、ページサイズ 4KB、物理空間 64bit の場合、最大 2GB までしか取り扱えない。ホスト上に実装すると、取り扱うメモリ量に制限はない。しかし、PCI 経由で参照するため、1 回の参照レイテンシは 0.32μs と大きい。そこで、使用頻度が高い領域は NIC 上に、使用頻度が低い領域はホスト上に変換表を配置し、両方式を併用する。

変換表の管理 変換表の登録、変更、削除、変換表の空き領域管理等の管理処理を、NIC 上の FPGA で実装するのは、ホストで実装する場合と比較して複雑である。したがって、ホスト側に実装する。

5. 高性能通信の UZURA 上への実装

本章では、前章で設計したメッセージ通信と RDMA 通信を実現するハードウェア構成について述べる。

前章の結果から、UZURA の FPGA に搭載すべき機能は以下のものである。

- DMA 転送機能
- ホストからの要求受付
- 論理物理アドレス変換
- ヘッダ生成/分解

これらの機能を FPGA 上に実現すると、FPGA 内部の構成は図 8 のようになる。

DMA 転送機能は、PCI 転送制御により実現する。PCI 転送制御を FPGA 上に実装するため、転送起動要求が発生してから、24ns でデータ転送を開始することができる。また、バストランザクションを直接制御するため、ホストの特性に合わせた最適なバストランザクションを実装する。

ホストからの要求受付を実現するため、転送要求管理が持つ起動レジスタを、ホスト上のメモリ空間にマップする。起動レジスタへの書き込み処理が発生すると、PCI 転送制御に対し、即座にデータ転送起動を要求する。

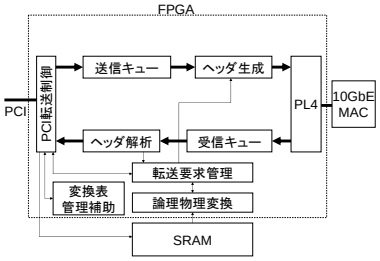


図 8 FPGA の実装

表 5 レイテンシ評価

処理	時間 (μs)
転送起動	0.78
DMA 転送 (ホスト→ NIC)	0.76
送信側 FPGA 内処理 (PL4 以外)	0.09
送信側 FPGA 内処理 (PL4)	0.16
10GbE MAC 処理	1.56
受信側 FPGA 内処理 (PL4)	0.35
受信側 FPGA 内処理 (PL4 以外)	0.15
DMA 転送 (NIC →ホスト)	0.18
合計	4.03

論理物理アドレス変換を実現するため、論理物理アドレス変換表を格納する SRAM に対して変換表を参照する論理物理変換機構を FPGA 上に実装する。変換表の管理はホストが行うため、SRAM はホスト上のメモリ空間にマップする。また、ホストによる SRAM 上の連続領域更新処理を高速化するため、ホスト-SRAM 間の DMA 転送機能を提供する。この機能を変換表管理補助に実装する。

ヘッダ生成/分解機能は、ヘッダ生成とヘッダ解析により実現する。ヘッダ生成処理は、起動レジスタへの書き込み直後より開始できるため、データ転送処理と並行して実行可能である。

6. 基本性能に基づく性能評価

本章では、設計した高性能通信の性能評価について述べる。現在のところ、本稿で述べた高性能通信の機能全ての実装が完了していないため、UZURA の基本性能と実装方式から、メッセージ通信のレイテンシ性能と RDMA 通信のスループット性能を予測する。

6.1 メッセージ通信のレイテンシ

設計した通信制御を用いた場合のメッセージ通信について、64 バイトのメッセージ (ヘッダ含) を転送した場合のレイテンシを算出した。

メッセージ転送に必要な各処理のレイテンシを表 5 に示す。これに基づき片道のレイテンシを算出すると、4.03μs になることが分かる。

このうち、送受信の双方で生じる 10GbE MAC 内部の処理に 1.56μs かかっている。また、送受信双方の FPGA 内処理のうち、0.51μs は、10GbE MAC へのインタフェースである PL4 マクロの処理である。したがって、10GbE MAC を FPGA 内に実装することで、これらの遅延時間は削減可能である。

6.2 RDMA 通信のスループット

RDMA 通信のスループットの算出のために、図 9 のようなデータ転送パイプラインを考える。図 9 は、転送ブロックが 1 パケットに収まる場合を示しており、上段は転送元 PCI-X バスを、中段は通信路を、下段

表 6 4KB 転送時の各ステージの転送時間

ステージ	転送時間 (μs)
転送元 PCI-X(t_1)	4.02
通信路 (t_2)	0.83
転送先 PCI-X(t_3)	3.89

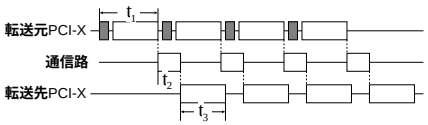


図 9 データ転送パイプライン

は転送先 PCI-X バスを示している。

PCI-X バス上の転送時間 (図 6、表 3) に基づき、MTU=4KB の場合の 4KB ブロック転送の各ステージの転送時間を算出した (表 6)。この結果から、スループットを算出すると、ボトルネックとなる転送ステージは転送元 PCI-X バスであり、4KB ブロックの RDMA 転送のスループットは 1.02GB/s となる。

7. 関 連 研 究

高性能通信機構について数多くの研究がなされている。AM-II[2] や FM[3] は、通信路に Myrinet を用いた高速通信機構である。データサイズの小さい short message は、レジスタ経由で転送し、それ以外は DMA 方式で転送することにより低レイテンシ化をはかっている。本研究で使用する NIC は、FPGA により DMA 転送制御を行っており DMA 起動遅延が小さいため、short message に関しても DMA 転送を用いている。

PM[4)~6)] は、Myrinet、Ethernet、InfiniBand 等の様々な通信路に対応した高性能通信機構である。PM では NIC 上の処理をファームウェアにより実現するのに対し、本研究では FPGA を用いるため、より細かい NIC 制御が可能である。例えば、PCI バスに関しては、バストランザクションレベルの細かい制御が可能である。

Myricom 社 Myrinet の低レベル通信ライブラリである MX 通信ライブラリ[7] は、MPI を効率良く実行することを念頭においたインタフェースを提供している。MPI 通信ライブラリの特徴である、メッセージのタグマッチングによる受信機能を提供している。

8. お わ り に

本稿では、MPI 通信ライブラリや NFS 分散ファイルシステムが必要とする通信機構について議論し、高性能通信処理オフロードエンジン実現に向けて、メッセージ通信と RDMA 通信の基本機能の実現方式について検討した。

メッセージ通信と RDMA 通信の実現方式をホストによる実行と NIC による実行との両面から検討し、定量的な観点から、両者の機能分担を決定し、通信制御を設計した。そして、 UZURA 上の FPGA に設計した通信制御を実装について述べ、通信制御を実装した場合の性能について評価した。

評価の結果、レイテンシについては片道 4.03μs、スループットについては、4KB ブロック転送の場合で 1.02GB/s を達成できる見込みであることが分かった。

今後の課題として、レイテンシを削減するための FPGA への MAC 処理の実装や、設計した通信制御を実際に実装した場合の実環境での性能評価を行なう。また、MPI、NFS、iSCSI を高速に実現するための NIC 支援機構を設計していく。

謝辞 本研究の一部は、文部科学省「eSociety 基盤ソフトウェアの総合開発」の委託を受けた東京大学石川研究室および東京大学石川研究室と富士通研究所との共同研究契約に基づいて行なわれた。

参 考 文 献

1) 住元真司, 堀敦史, 手塚宏史, 原田浩, 高橋俊行, 石川裕. 高速通信機構 PM2 の設計と評価. 情報処理学会論文誌, Vol.41 No. SIG 5 (HPS-1), pp.80–90, August 2000.
2) B.Chung, A. Mainwaring, and D. Culler. “Virtual Network Transport Protocols for Myrinet”. In Host Interconnects’97, Stanford, CA, April 1997.
3) Scott Pakin, Mario Lauria, and Andrew Chien. “High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet” In Proceedings of the 1995 ACM/IEEE Supercomputing Conference, San Diego, California, December 3–8 1995.
4) 手塚宏史, 堀 敦史, 石川 裕:ワークステーションクラスタ通信ライブラリ PM の設計と実装, 並列処理シンポジウム JSPP’96 , pp.41–48(1996).
5) 住元真司, 堀 敦史, 手塚宏史, 原田 浩, 高橋 俊行, 石川 裕 GigaE PM: Gigabit Ethernet を用いた高速通信機構の設計と評価 情報処理学会論文誌, Vol 41, No 5, pp.1390–1399(2002).
6) Shinji Sumimoto, Akira Naruse, Kouichi Kumon, Kouji Hosoe, and Toshiyuki Shimizu “PM/InfiniBand-FJ: A High Performance Communication Facility Using InfiniBand for Large Scale PC Clusters”, HPC Asia 2004 7th International Conference on High Performance Computing and Grid in Asia Pacific Region, pp. 104–113. IPSJ HPC SIG and IEEE, July, 2004.
7) Myrinet Express(MX): A High Performance, Low-Level, Message-Passing Interface for Myrinet (DRAFT)
http://www.myri.com/scs/MX/doc/mx.pdf