社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－ARC－170 (16)
2006／11／30

ハードウエアによるMPI派生データ型通信の支援

宮部保雄[†] 宮代具隆[†] 北村聡[†]
田邊昇[††] 中條拓伯[†††] 天野英晴[†]

MPIの派生データ型は、メモリ上に不連続に存在するデータを1つの型として定義することで、1回の関数呼び出しで不連続なデータの通信を可能にし、効率的な並列アプリーションを開発するのに役立つ。しかし、多くのMPIの実装における派生データ型による通信は、その処理に不連続なメモリアクセスを伴うなどの理由により、性能が悪いという問題があった。そこで我々は、ハードウエアの支援により派生データ型通信の性能を向上させる手法を提案してきた。本論文では、実際に派生データ型通信を支援する機構を、メモリスロット装着型ネットワークインタフェースであるDIMMnet-2の試作基板に追加実装した。そして、実装した機構を利用するMPIライブラリを開発し、その性能を派生データ型通信のバンド幅によって評価した。その結果、追加実装した機構を利用する場合、利用しない場合に比べて約2.6倍のバンド幅を観測することができた。

Hardware Support for MPI Derived Datatypes Communication

YASUO MIYABE,[†] TOMOTAKA MIYASHIRO,[†] AKIRA KITAMURA,[†]
NOBORU TANABE,[††] HIRONORI NAKAJO[†††] and HIDEHARU AMANO[†]

MPI derived datatypes, which allow users to communicate noncontiguous data with a single communication function, are useful for developing parallel applications efficiently. However, because of the overhead for noncontiguous memory accesses, the performance of derived datatypes communication in many MPI implementations tends to be insufficient. We have proposed to use hardware mechanisms for the improvement of the performance of derived datatypes communication. In this paper, we implemented the mechanisms in DIMMnet-2 prototype board, which forms a PC cluster by attaching into the memory slot of host PC, and developed the MPI library using them. The result of evaluation shows that the bandwidth of derived datatypes communication using the mechanisms is 2.6 times compared to that without it.

1. はじめに

PCクラスタに代表される分散メモリ型の並列システムにおけるアプリケーション開発には、MPI (Message Passing Interfece)[1]を利用するのが主流になっている。MPIはアプリケーションの開発を簡便化するために多くの機能を提供するが、その1つに派生データ型がある。

派生データ型は、ユーザがアプリケーションの実行時に自由に定義できる型であり、その型を用いてMPIの送受信関数を呼び出すことができる。例えば、派生データ型を使うとメモリ上に不連続に存在するデータを1回のMPI関数の呼び出しで送信することができる。このように便利な派生データ型であるが、多くのMPIの実装において派生データ型を用いた通信の性能は十分でなく[2)3)]、このため、従来、敬遠される場合が多かった。

しかし、近年、派生データ型を処理するソフトウエアの改良が行われ、その通信性能は若干改善されている[2)4)]。このため、その便利さや、若干の通信の高速化が期待できることなどから、派生データ型通信を利用したアプリケーションが登場し、今後増加することが期待される。とはいえ、派生データ型の処理には、本質的に不連続なメモリアクセスが伴い、キャッシュベースのCPUには向いておらず、ソフトウエアによる処理には限界があると考えられる。

そこで我々は、ハードウエアの支援により派生データ型通信の性能を向上させる手法を提案してきた[5)]。本論文では、派生データ型通信を支援するハードウエア機構を、我々が開発を続けているネットワークイン

† 慶應義塾大学
Keio University
†† (株) 東芝, 研究開発センター
Corporate Research and Development Center,Toshiba
††† 東京農工大学
Tokyo University of Agriculture and Technology

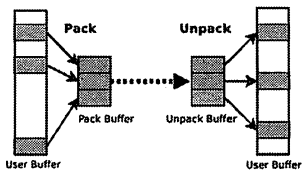

図 1 Pack/Unpack 処理

タフェースカード（NIC）である DIMMnet-2 の試作基板に追加実装した。そして、その機構を用いて派生データ型通信を行う MPI ライブラリを開発し、通信バンド幅を計測した。その結果、追加実装した機構を利用しない場合に比べ、本機構を利用する場合は、約 2.6 倍高いバンド幅が観測された。

以下、本論文では、2 章で関連研究と我々が提案しているハードウエアによる派生データ型通信の概要について述べる。3 章で DIMMnet-2 へのハードウエアの実装を、4 章で開発した MPI ライブラリについて述べ、5 章でその評価を行う。そして、6 章でまとめと今後の課題について述べる。

2. 関連研究と提案手法

2.1 一般的な派生データ型通信

多くの NIC では、メモリ上の連続したデータに対して、開始アドレスと長さを指定して転送する。このため、不連続なデータ転送が必要な MPI 派生データ型の実装では、通常、Pack 処理によりデータを連続化してから送信し、受信側での Unpack 処理により、再び不連続化している（図 1）。

この Pack/Unpack 処理に含まれる不連続なメモリアクセスを伴うデータコピーが、派生データ型通信の性能低下の要因となる。この性能低下を防ぐため、TLB の情報やメモリアクセスパターンなどから適切な Pack のアルゴリズムを選択する研究[2] が試みられているが、全てをホスト PC のソフトウエアで処理するためオーバーヘッドが大きい上、Pack 処理によりホスト PC のキャッシュを汚染するなどの問題がある。

2.2 NIC の機能の利用した派生データ型通信

一方で、MPI 派生データ型通信を InfiniBand の VAPI（Verbs Level API）が提供する Gather/Scatter 機能付 RDMA を用いて実装し、Pack/Unpack 処理を NIC 上の CPU 上で走るファームウエアにオフロードする手法が提案されている[3][6]。しかし、NIC 上の CPU はホストの CPU より周波数が大幅に低い上、連続するデータのサイズが小さい場合に NIC 上の CPU のキャッシュミスが多発するという問題がある。

2.3 提案手法の概要

我々は、NIC 上の CPU 上のソフトウェアではなく、ベクトル転送命令を実行するハードウエアを NIC 上の FPGA に搭載することによって派生データ型通信の性能向上する手法を、提案してきた[5]。

このハードウエアはメモリと密に結合されており、連続したデータへの単純なアクセスに加え、メモリ上に等間隔に存在するデータへのアクセス（ストライドアクセス）や、アドレスが格納されたリストを用いたデータへのアクセス（リストアクセス）をホストの CPU とは独立して、高速に行うことができる。これらのアクセスを組み合わせて利用することで、ホストの CPU 時間の浪費やキャッシュの汚染などをおこさずに、Pack/Unpack 処理が高速化できることが先行研究[5] で確認されている。

本論文では、このハードウエアに、ベクトルアクセス命令によりメモリから読み出したデータをそのままネットワークに送出する機能と、ネットワークより受信したデータをそのままベクトルアクセスでメモリに格納する機能の 2 つから成るリモートベクトルアクセス機構を追加実装する。この機構を用いると、Pack/Unpack 処理のどちらかを省略することができ、さらなる派生データ型通信の高速化を図ることができる。

3. DIMMnet-2 へのハードウエア実装

本章では、DIMMnet-2 試作基板へのリモートベクトルアクセス機構の実装について述べる。

3.1 DIMMnet-2 の概要

DIMMnet-2 は、PC クラスタ向けインターコネクトの NIC である。一般に PC クラスタ向けインターコネクトの NIC が PCI-X バスや PCI-Express バスに接続されるのに対し、DIMMnet-2 はホストの DDR-SDRAM バスに接続される。メモリバスに接続することにより、ホストから NIC へのアクセスレイテンシを低く抑えられ、結果、低レイテンシな通信を実現することが可能となる。DIMMnet-2 ではシステム構築のコスト削減のために、ネットワークスイッチとケーブルに InfiniBand を採用している。

3.1.1 DIMMnet-2 試作基板

DIMMnet-2 試作基板はコントローラ部に Xilinx 社の Virtex-II Pro XC2VP70-7FF1517C を用いている。Virtex-II Pro に内蔵されている RocketIO トランシーバを用いて InfiniBand（4X:10Gbps）に接続する。また、ノート PC 用の DDR SO-DIMM を 2 枚搭載しており、通信用のバッファとして使用するほか、ホスト

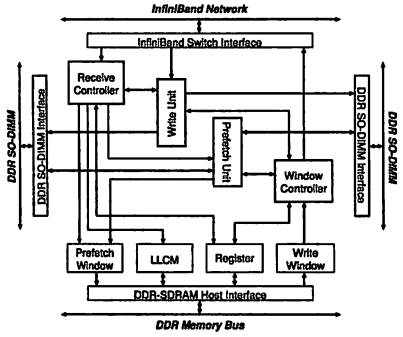

図 2 ネットワークコントローラのブロック図

のデータ記憶領域として使用する。

3.1.2 DIMMnet-2 ネットワークコントローラ

DIMMnet-2 ネットワークコントローラのブロック図を図 2 に示す。

各ブロックの機能は以下の通りである。

- LLCM: ホストからもコントローラからも読み書き可能なメモリ領域
- Prefetch Window: ネットワークインタフェース上の SO-DIMM から読み出したデータを格納する領域
- Register: ネットワークコントローラの設定、ステータス、要求の発行等に使用するレジスタ群
- Write Window: SO-DIMM に格納するデータを書き込む領域
- Window Controller: ホストからの各種要求やパケット送信処理部
- Receive Controller: ネットワークから受信したパケットの処理部
- Prefetch Unit: SO-DIMM のデータ読み出し制御部
- Write Unit: SO-DIMM へデータ書き込み制御部

これらのブロックのうち、LLCM、Prefetch Window、Register、Write Window の 4 つのブロックが DIMMnet-2 ドライバによりユーザプロセスの仮想アドレス空間にマップされる。SO-DIMM には、ホストから直接アクセスすることはできず、Prefetch Window と Write Window を介してアクセスする。このような形態を取ることで、ホストのチップセットによる制約等を受けずに、ホストに搭載可能なメモリ容量よりも大きな記憶領域を持つことが可能になっている。

SO-DIMM と Prefetch/Write Window 間に作用するベクトルアクセス命令は、現在までに表 1 に示す 6 命令が実装されている。

VL 系列の命令と VS 系列の命令を 2 つ組み合わせて実行すると、ネットワークコントローラ内部で 2 命令が FIFO によってチェインされ、SO-DIMM の 2 領域間でのデータコピーを行うことができる。この DIMM 間データコピーによる Pack/Unpack 処理の高速化については、文献[5] に示されている。

表 1 ローカルの SO-DIMM に対するベクトルアクセス命令

アクセスの種類	SO-DIMM に対して Read	SO-DIMM に対して Write
連続アクセス	VL	VS
ストライドアクセス	VLS	VSS
リストアクセス	VLI	VSI

表 2 リモートベクトルアクセス命令

	命令名	SO-DIMM へのアクセスの種類 ローカル	転送方向	リモート
RDAM Write	RVS	VL	→	VS
	RVSS	VL		VSS
	RVSI	VL		VSI
RDAM Read	RVL	VS	←	VL
	RVLS	VS		VLS
	RVLI	VS		VLI

3.2 リモートベクトルアクセス機能の実装

以前より DIMMnet-2 には、リモート☆ の SO-DIMM に対するアクセス命令として、リモートにデータを書き込む RDMA Write に相当する RVS、リモートのデータを読み出す RDMA Read に相当する RVL が実装されていた。これらの命令は、SO-DIMM 上の連続データの転送のみ行なうことができる。

本論文では、ローカルの SO-DIMM に対するベクトルアクセス命令を拡張し、リモートの SO-DIMM に対してもベクトルアクセスを可能とするリモートベクトルアクセス機構を DIMMnet-2 試作基板に追加実装した。利用可能なリモートベクトルアクセス命令を表 2 に示す。

リモートの SO-DIMM に対するベクトルアクセスを可能にするにあたって必要となった主なネットワークコントローラの改造点を以下に示す。

- Receive Controller による Prefetch Unit の制御

  リストアクセス命令が参照するリストは、SO-DIMM に格納され、Prefetch Unit によって読み出される。ローカルの SO-DIMM に対するリストアクセスでは、Window Controller が適宜、Prefetch Unit を制御し、リストの読み出しを行なっていた。この機能を Receive Controller にも実装した。
- 512Byte の制限の解除

  Prefetch Window と Write Window は、それぞ

☆ 通信を起動した側をローカル、ローカルの通信相手をリモートとする。

れ内部が 512Byte 毎に分割されており、ユーザプログラムの誤った操作によって、意図しないデータの書きつぶしなどがが起きないような機構が Prefetch Unit や Write Unit には備わっている。この機構によりベクトルアクセス命令によっては、SO-DIMM からネットワークに送出するデータや、ネットワークから SO-DIMM へ書き込むデータのサイズが、512Byte に制限されてしまっていたが、これを解消した。

4. MPI ライブラリの実装

4.1 MPICH2

MPICH2[7] は、アルゴンヌ国立研究所が中心となり開発されている MPI の模範実装である。この MPICH2 をベースに東京農工大学で開発された DIMMnet-2 上における MPI の実装がある[8]。本論文では、この MPI の実装を改造して、DIMMnet-2 のリモートベクトルアクセス機構を利用した派生データ型通信を実現した。ただし、現状、DIMMnet-2 の SO-DIMM と通常のメモリを同じようにユーザプログラムからアクセスするためのドライバの整備が完了していないため、データの転送は各 DIMMnet-2 の SO-DIMM 間でのみ行う。

MPICH2 は、様々なシステムへの移植性を考慮し、MPI 層と通信層を切り分けている。この通信層を Abstruct Device Interface（ADI）と呼び、この ADI 層で Eager 及び Rendezvous の 2 つのプロトコルを実装することを前提として MPICH2 全体が設計されている。

Eager プロトコルでは、送信側はデータを受信側の所定のバッファに転送し、受信側はそのバッファから受信領域にデータをコピーする。低遅延な転送が可能であるが、バッファ間のコピーが必要であり、大きなデータの転送には向かない。

一方、Rendezvous プロトコルでは、データの送受信に先立って、送信側、受信側で同期を取り、データを転送する。同期の遅延がかかるものの、ゼロコピー転送をサポートしたシステムでは、データを直接、送信領域から受信領域に転送できる利点がある。

4.2 派生データ型通信のプロトコル

本論文では、Rendezvous プロトコルによる派生データ型通信を実装する。RDMA 機能によって Rendezvous プロトコルを実装する場合、RDMA Write（RVS 系列の命令）を使う手法と、RDMA Read（RVL 系列の命令）を使う手法が考えられるが、RDMA Read による実装は、RDMA Write による実装に比べ次の 2 つの利点がある[9]。そのため、今回は RDMA Read による実装を行なった。

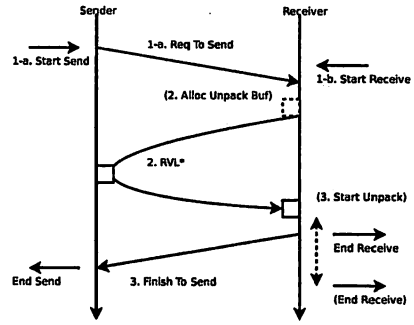

図 3 DIMMnet-2 の派生データ型通信

- プロトコルを制御するパケットの数が少ない
- 受信側が Rendezvous を開始する要求を送信側から受け取りさえすれば、あとは受信側が送信側から独立して、データ転送処理を行なうことが可能

この度実装した DIMMnet-2 における RVL 系列の命令を用いた Rendezvous プロトコルによる派生データ型通信の流れを、次に示す（図 3）。

1-a 送信関数の呼び出し

送信を行なう関数が呼ばれると、送信側は直ちに送信側の派生データ型と、送信するデータの先頭アドレスの情報が含まれる送信要求パケットを受信側に送る。

1-b 受信関数の呼び出し

受信を行なう関数が呼ばれると、受信側はそれ以前に送られて来た送信要求パケットに Tag や Rank などの条件が一致するものがないか検索する。なければ、条件が一致する送信要求パケットが送られて来るまで待つ。

2 Unpack 用バッファ確保と RVL 系列の命令実行

条件に一致する送信要求パケットが見つかると、受信側は、受信側で Unpack が必要な場合、SO-DIMM に Unpack 用の領域を確保する。そして、送信要求パケット内の送信側の情報と、受信側の SO-DIMM のアドレスを指定して RVL 系列の命令を実行する。この受信側のアドレスには、Unpack を行なう場合は確保した領域のアドレスが、そうでない場合は、受信関数に受信先として渡されたアドレスが設定される。なお、RVL 系列の命令は必要に応じて、複数回実行される。

3 通信終了通知の送信と Unpack の実行

受信側は、RVL 系列の命令によるデータの受信完了を検出すると、直ちに送信終了パケットを送信側に送る。このパケットを受信した送信側は、送信操作を終了する。

表 3　評価環境

CPU	Pentium4 2.6GHz
Chipset	VIA VT8751A
Memory	PC-1600 DDR SDRAM 512MByte ×1
	DIMMnet-2 ×1
OS	RedHat8.0 (Kernel 2.4.27)
Switch	Voltaire ISR6000 (Cable: 2m ×2)
Compiler	gcc-3.3.7 (-O3 -march=pentium4 -msse2)

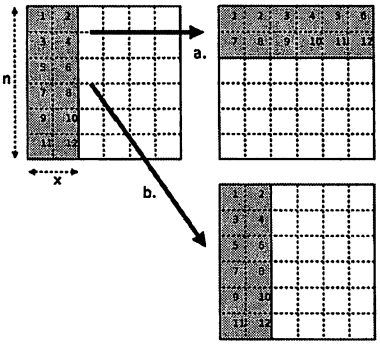

図 4　データの転送パターン

送信終了パケットを送信した受信側は、Unpack が不必要ならば受信操作を終了する。必要ならば、受信側の派生データ型の情報を元に、VL 命令と VS 系列の命令の組合せによる DIMM 間コピーによる Unpack 開始し、完了を待つ。Unpack が完了したら、Unpack 用バッファを開放し、受信操作を終了させる。

以上のプロトコルにおいて、送信側から受信側に送られる派生データ型の情報や、受信側が Unpack 処理を行なう際に参照する派生データ型の情報は、DIMMnet-2 のベクトルアクセス命令のパラメータのリストである。

現時点で実装が完了しているデータ型コンストラクタは、配列に等間隔に並んだエントリを要素とする派生データ型を構築するときに使われる `MPI_Type_vector` 関数のみである。この関数が呼ばれると、ライブラリは指定された情報からストライドアクセス命令のパラメータのリストを作成し、派生データ型として保持する。

5. 評　　価

本章では、本論文で実装した MPI 派生データ型通信の評価を示す。

5.1 評価手法

表 3 に評価環境を示す。DIMMnet-2 を搭載した PC を 2 台を InfiniBand スイッチを介して接続した。この 2 台の PC 間で通信を行なう。

送信するデータは、DIMMnet-2 の SO-DIMM に置

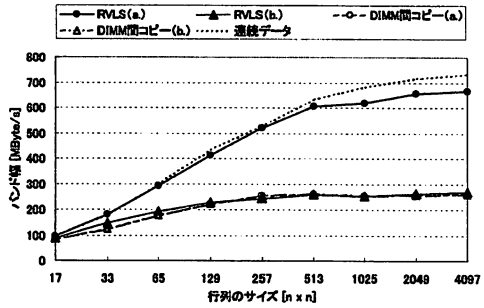

図 5　列数 $x = 16$ の時の通信バンド幅

かれた MPI_DOUBLE 型の $n \times n$ 行列の列とした。この行列の列を図 4 の a. と b. のような 2 パターンで受信側の DIMMnet-2 の SO-DIMM に転送した際のバンド幅を、次の 2 種類の通信方法を用いた場合のそれぞれで計測した。ここで図中、x は一度に転送する列数を表している。

- RVLS を使い Pack 処理無しに不連続データを転送
- DIMM 間コピー（VLS と VS のチェイン）によって Pack 処理を行ない連続データを転送

なお、本論文の MPI の実装では、全てのデータ転送は Rendezvous プロトコルによって行なわれる。

`MPI_Send` と `MPI_Recv` に渡す派生データ型は、`MPI_Type_vector(n,x,n,MPI_DOUBLE,&colType)` で構築した。構築した新しいデータ型 colType を a. の転送では `MPI_Send` に、b. の転送では `MPI_Send` と `MPI_Recv` の双方に渡した。

現在の DIMMnet-2 の設計では SO-DIMM へアクセスする際、データが連続しているブロックのサイズが 8Byte 以下で、その間隔の Byte 数が 16 の倍数であると、バンクコンフリクトを起こし、性能が低下することが分かっている。そのため、行列サイズ n は、このバンクコンフリクトを起こさない値を使った。

5.2 評価結果

列数 $x = 16$ の場合のバンド幅の計測結果を、図 5 に示す。参考として掲載した図中の連続データという項目は、実際に転送した行列の列データのサイズ分、SO-DIMM より連続したデータを送受信した場合のバンド幅である。

RVLS を使わない場合、受信側で Unpack を行なわない a. の $n = 4097$ の通信におけるバンド幅は、260.8MByte/s であった。一方、RVLS を使った場合のバンド幅は、667.4MByte/s であり、約 2.6 倍の性能向上が見られる。

ここで、ホストの CPU によって a. の通信に含ま

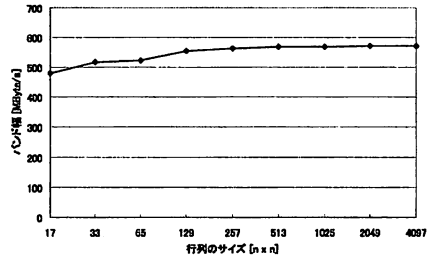

図 6 列数 $x = 16$ の時の RVSS のバンド幅

れる Pack 処理を行なう場合を考えると、処理を開始する前、ホスト CPU のキャッシュは今までの計算処理などのために全て使われており、Pack 用のバッファなどはキャッシュから追い出されていることが予想できる。そこで、この状況を再現した上で Pack 処理を行なうプログラムを作成し、表 3 に示すホスト PC で実行した所、$n = 4097$ において、そのバンド幅は約 165MByte/s であった。これにより、受信側で Unpack を行なわない派生データ型通信においては、RVLS を実行可能な DIMMnet-2 を使うと、ホスト PC のみで処理を行なう場合に比べ、約 4.0 倍の高いバンド幅を得られる可能性があることがいえる。

一方、受信側で Unpack を行なう b. の通信において RVLS を使った場合のバンド幅は、RVLS を使わない場合にほぼ一致する。この結果は、通信の性能が、受信側での DIMM 間コピーによる Unpack 処理に律速されていることを示している。

本論文では、MPI の実装では使わなかったが、RVLS の逆、ネットワークから受信したデータを SO-DIMM に VSS する RVSS 命令の実装も行なった。MPI の処理を含まない生の RVSS のバンド幅を計測した結果を**図 6** に示す。図 5 と図 6 により、今後、RVL 系列の命令で SO-DIMM から読み出したデータを、RVS 系列の命令と同様にそのまま VS 系列の命令で SO-DIMM に書き込むリモートベクトルアクセスを実装した場合、b. の通信パターンにおいても $x = 16, n \geq 513$ において 550MByte/s 前後の高バンド幅が期待できることが分かる。

6. まとめと今後の課題

本論文では、MPI 派生データ型通信を支援するハードウエアとしてリモートベクトルアクセス機構を DIMMnet-2 に実装し、この機構を利用して派生データ型通信を行なう MPI ライブラリの開発を行なった。評価の結果、派生データ型通信において、この機構を利用すると、利用しない場合に比べ最大で約 2.6 倍のバンド幅を観測することができた。

今後は、5 章で述べた RVL 系列の命令における受信データをそのまま VS 系列の命令で SO-DIMM に書き込む機構の実装や、複雑な構造をした派生データ型の通信を複数のリモートベクトルアクセス命令で実現する際の命令の組み合わせ方を決定するアルゴリズムの検討を行ないたい。

謝辞 本研究は総務省戦略的情報通信研究開発推進制度（SCOPE）の一環として行われたものである。DIMMnet-2 の開発に関する議論、開発にご参加頂いている日立情報通信エンジニアリング株式会社の今城氏、岩田氏、上嶋氏、東京農工大学の濱田氏、荒木氏、慶應義塾大学の渡邊氏、大塚氏に感謝致します。

参 考 文 献

1) The Message Passing Interface standard: http://www-unix.mcs.anl.gov /mpi/.
2) Surendra Byna, William Gropp, Xian-He Sun and Rajeev Thakur: Improving the Performance of MPI Derived Datatypes by Optimizing Memory-Access Cost, *CLUSTER*, Vol. 00, p. 412 (2003).
3) Jiesheng Wu, Pete Wyckoff and Dhabaleswar Panda: High Performance Implementation of MPI Derived Datatype Communication over InfiniBand, *IPDPS*, Vol. 01, p. 14a (2004).
4) Robert B. Ross, Neill Miller and William Gropp: Implementing Fast and Reusable Datatype Processing, *PVM/MPI*, pp. 404–413 (2003).
5) 田邊 昇, 北村 聡, 宮部 保雄, 宮代 具隆, 天野 英晴, 羅 徹哲, 中條拓伯: DIMMnet-3 ネットワークインターフェースにおける MPI 支援機能, 情報処理学会アーキテクチャ研究会, Vol. 2006-ARC-169, pp. 103–108 (2006).
6) Gopalakrishnan Santhanaraman, Dhabaleswar Wu and Dhabaleswar K. Panda: Zero-Copy MPI Derived Datatype Communication over InfiniBand, *PVM/MPI*, pp. 47–56 (2004).
7) MPICH2 home page: http:// www-unix.mcs.anl.gov /mpi/mpich2/.
8) 荒木 健志, 森 拓郎, 金井 遵, 田邊 昇, 天野 英晴, 並木 美太郎, 中條拓伯: DIMMnet-2 における通信ライブラリ MPI-2 の実現, 情報処理学会ハイパフォーマンスコンピューティング研究会, Vol.2006-HPC-105, pp. 49–54 (2006).
9) Sayantan Sur, Hyun-Wook Jin, Lei Chai and Dhabaleswar K. Panda: RDMA read based rendezvous protocol for MPI over InfiniBand: design alternatives and benefits, *PPOPP*, pp. 32–39 (2006).