計算機アーキテクチャ 152－9
（２００３．３．１０）

高速通信インタフェースDIMMnet-1における通信プリミティブの実装と評価

三橋彰浩[†] 濱田芳博[†] 中條拓伯[†]
田邊昇[††] 天野英晴[†††]

我々は，PC クラスタ用ネットワークインタフェース (NIC)DIMMnet-1 によるクラスタを構築している．DIMMnet-1 は PC メモリバスに接続される NIC で，従来の PCI バスに接続される NIC においてボトルネックとなっていた，メモリ-NIC 間のダイレクトメモリアクセス (DMA) を排除し，ホストが Programed I/O（PIO）により，NIC へ効率良くアクセスすることが可能になった．このような通信機構は On-The-Fly（OTF）通信機構と呼ばれ，ハードウェアで実装された低レイテンシな送信機構である Atomic-On-The-Fly（AOTF）と連続ワードでデータ送信を行う Block-On-The-Fly（BOTF）がある．またこれ以外にもリモート DMA による PUSH/PULL と呼ばれる通信機構を備えており，本稿ではそれらの送信機構を用いて Send,Receive を実装し，基本通信性能の評価を行うとともに，実装上の問題点について考察する．

Implementation and Evaluation of Send/Receive Primitives of a Fast Communication Interface DIMMnet-1

Akihiro Mitsuhashi ,[†] Yoshihiro Hamada ,[†]
Hironori Nakajo ,[†] Noboru Tanabe [††] and Hideharu Amano[†††]

We have been configuring a PC cluster with DIMMnet-1 which is a network interface plugged into a memory interface. Bottleneck caused by direct memory access between memory and NIC which connected to a conventional PCI bus has been eliminated, and efficient accesses from the host to the NIC has become possible with Programmed I/O (PIO). This communication mechanism is called an OTF communication mechanism in which AOTF, which is a low latency transfer mechanism and BOTF, which transfers data with continuous words, are both implemented by hardware. Moreover, communication mechanisms called PUSH/PULL by remote DMA are also implemented. In this paper, we describe implementation of Send and Receive primitives with these transfer mechanisms and we show evaluations of the basic performance and also discuss about problems of implementation.

1. はじめに

近年，PC の性能は著しく向上し，価格も安価になってきているため，高性能な PC を相互接続したネットワークで構成されるクラスタコンピューティングが浸透しつつある．その通信インタフェースの代表的なものとして，Myrinet[1] などの低遅延なものや，GigabitEther の様な高バンド幅のものが使用される．しかし，これらはいずれも PCI バスをホストコンピュータとのインタフェースに用いており，この部分の遅延やバンド幅の制限が通信性能を限定している．そのため，CPU の性能向上にともない，ネットワークインタフェース（NIC）や入出力バスについて考慮しなければ PC クラスタの性能を最大限に引き出すことは困難である．そこでこれらの問題を解決するために，DIMM スロットに接続するタイプの NIC である DIMMnet-1[2] を開発した．DIMMnet-1 は DIMM スロットに接続するタイプの NIC であり，PCI バス等の入出力バスに接続するよりも，CPU と密接に結合されるため，バンド幅の拡大とレイテンシの大幅な削減が可能となる．

本稿では，DIMMnet-1 によるクラスタシステムを RHiNET2/SW2[3] によって構築し，Message Passing-Interface(MPI) のようなライブラリ実装の足掛かりとして，Send，Receive を実装し，その評価と考察を行う．また実装上の問題点についてふれ，DIMMnet-2

† 東京農工大学
Tokyo University of Agliculture and Technology
†† （株）東芝 研究開発センター
TOSHIBA Corporate Research & Development Center
††† 慶応義塾大学
Keio University

に向けての改善点を洗い出す.

2. DIMMnet-1 の概要

2.1 ユーザレベル通信

ユーザレベル通信は，OS をバイパスしユーザプロセスが直接 NIC にアクセスして通信を行う方法で，ユーザ空間とカーネル空間の切替えのオーバヘッドを削減することができる．ユーザレベル通信では，ユーザプロセスから直接 NIC に渡された情報がどのプロセスからのもので，そのプロセスは NIC 上のリソースにアクセスする権利を持っているかどうかを NIC 側で判断する必要がある．NIC は，内部にユーザプロセスの仮想アドレスから物理アドレスへの変換を行うための Translation Look-aside Buffer（TLB）を持っている．この TLB に格納される属性によりプロセス間の保護を実現している．

2.2 Atomic on-the-fly（AOTF）

AOTF は，ホストプロセッサからの書き込みによってパケットを送信する機構である．ユーザプロセスにより AOTF 用の領域にデータの書き込みが行われると，あらかじめ設定しておいたヘッダシードを元にパケットが生成され，送信される．このため，CPU がそのデータを所定のアドレスに書き込むという一命令だけでパケット送信が可能である．AOTF では一度に最大 8 バイトのデータを送信することができ，低レイテンシな通信を実現している．AOTF 送信におけるパケット生成メカニズムを図 1 に示す.

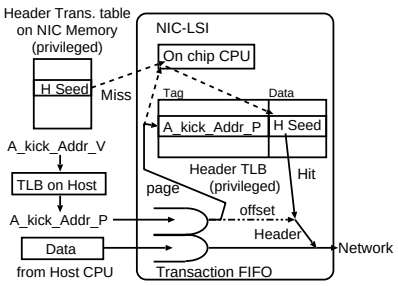

図 1 AOTF パケット生成メカニズム

ここで，HTLB は物理アドレスからヘッダシードを連想し，パケットを生成するハードウェアである．図 2 に DIMMnet-1 における HTLB の構成を示す.

2.3 Block on-the-fly（BOTF）

BOTF はユーザプロセスがパケットを作成し，送信する機構である．ユーザプロセスは，送信をするための領域である Window 上にヘッダを含めたパケットのすべてを書き込んだ後，Window 上のキックアドレスと呼ばれるプリミティブの種類を書き込むことで実行が開始されるアドレスに書き込みを行うことで，DIMMnet-1 の NIC コントローラである Martini[4] に送信開始を指示する．BOTF ではヘッダを除いて一

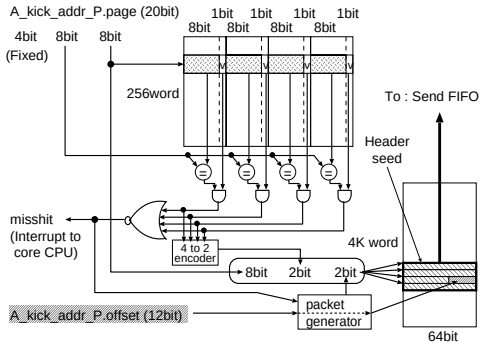

図 2 ヘッダ (HTLB) の構造)

度に最大 464 バイトのデータが送信可能であり，高バンド幅な通信を実現している．BOTF 送信におけるパケット生成メカニズムを図 3 に示す.

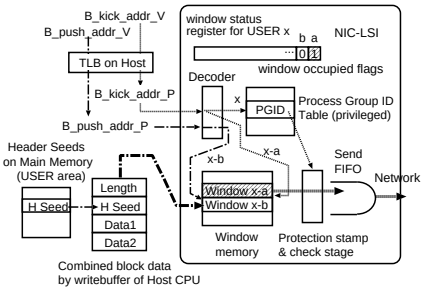

図 3 BOTF パケット生成メカニズム

2.4 OTF 受信機構

OTF 受信機構とは，アドレス変換や DMA コントローラの起動を行わず，パケットヘッダの情報から所定の長さのデータ部を直接メモリに書き込む機構である.

DIMMnet-1 では AOTF 送信に限り，リモートアドレスを物理アドレスで登録することができ，受信時のリモートにおけるアドレス変換のオーバヘッドを削除することが可能である．AOTF 送信に限り立てることができるヘッダー中のフラグを受信部が判定し，アドレス部と 1〜8 バイトのデータ部を書込みバッファに書き込んで行く．書込みバッファは Martini 上のオンチップメモリである低遅延共有メモリ (LLCM) に，書き込めるタイミングで書き込む.

このように DIMMnet-1 では送信側の AOTF と受信側の OTFR が共同して極めて低遅延な通信を実現している.

2.5 Remote DMA（RDMA）

RHiNET[5] 等においては，PCI バスを経由して RDMA がホストの主記憶をアクセスする．これに対し，DIMMnet-1 では RDMA は NIC 基板上のバンク構成になった DIMM や通信制御 ASIC 内部の低遅延共有メモリ LLCM にアクセスする．DIMMnet-1 上

の DIMM ホストから書き込んだ送信データをネットワークに送信する時には，RDMA による送信を起動する前に，送信データを保持している DIMM のバンクをホスト側から Martini 側に切替える必要があるので，オーバヘッドが大きい．

一方，現在の DIMMnet-1 の実装では RHiNET 用に最適化された受信部を流用しているため，受信側におけるパケット毎に必要なインターバルが大きく，BOTF による短いパケットに伴うバンド幅が十分に出ない．このため，このような DIMMnet-1 に最適化されていない実装においては，パケット長に関する制約が緩い RDMA は，大量データを送信する際に高バンド幅を実現する手段として意味がある．

RDMA には PUSH プリミティブと PULL プリミティブが存在し，PUSH プリミティブはローカルノードのメモリブロックをリモートノードのメモリブロックへ転送し，PULL プリミティブはリモートノードのメモリブロックをローカルノードのメモリブロックへ転送する．

2.6 バンクの切替えについて

SO-DIMM は Martini 上に 2 枚搭載されている．これら 1 枚の容量は 256M バイトであり，図 4 のようにバンク構成になっており，各々のメモリがホスト側と Martini 側に対応している．このようにそれぞれのバンクをホスト側と Martini 側に対応させることで，PUSH プリミティブ使用時などにおいてデータを送信する場合，データの入った Martini 側の SO-DIMM からデータを送信する間にもう一方の SO-DIMM に次の送信データを書き込んでおき，データの送信が終了したあと，タイミングを図り，バンクを切替えることでバッファリングが可能となっている．またデータ受信時においても，Martini 側から SO-DIMM に対して受信が行われ，データをホストメモリに退避する必要がある場合においても，同じくタイミングを図り，バンクを切替えることによって効率の良いデータの退避が可能となる．

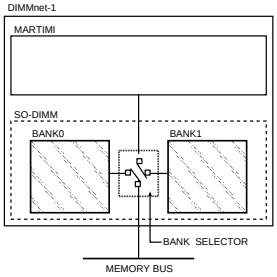

図 4 SO-DIMM とバンク構成

2.7 BOTF と PUSH プリミティブのバンド幅における交差点

BOTF によるデータ送信は，送信データ量が 464 バイト以下の場合は 1 枚の Window でのデータ送信を行い，464 バイトを越えるデータを送信する場合においては，送信データ量を 464 バイト毎に分割し，2 枚の Window を交互に使用することによってデータの転送を行う．ここで，送信 Window のキャッシュ属性は WriteCombining にし，データ受信領域には LLCM を用いた．また送信時のデータは CPU キャッシュ上に乗っているものとしている．送受信にかかる時間はデータを送信し始めてから受信領域に全てのデータが受信されるまでとし，この時間を 200 回測定した値の平均値を求めることでバンド幅を算出した．

PUSH プリミティブ発行用の送信 Window はキャッシュ属性を UnCashed にし，データ送受信領域には LLCM を用いた．送信側において，送信完了の ACK 受信先には LLCM を用い，PUSH プリミティブ発行後の ACK を受信するまでの時間を 200 回測定した値の平均値を求めることでバンド幅を求めた．

図 5 のグラフより，464 バイトまでは BOTF の性能が高く，800 バイトまではほぼ同じ性能だが，それ以上のデータ量を送信する場合には PUSH プリミティブの性能が高くなっているのがわかる．BOTF の性能が 464 バイト付近で落ち込んでいるのは，1 回に送信可能な容量（464 バイト）を越えたために，2 つ目のパケットを新たに生成してデータを送信しているためである．

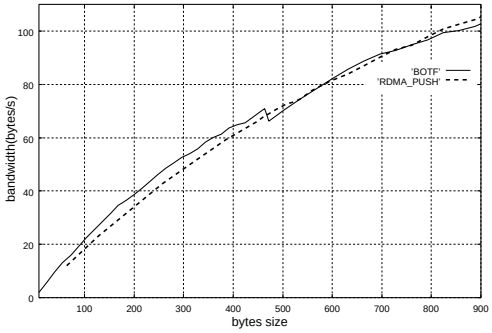

図 5 BOTF と PUSH プリミティブの交差点

2.8 受信側のデータ受信完了のタイミングについて

データ通信時において，SO-DIMM にデータを送信した場合，受信側はデータがいつ完全に受信したかを知るのが難しい．それは，データが届く SO-DIMM 領域は，その時点ではバンク構成上 Martini 側を向いているため，ホスト側からはそのデータを確認することができないからである．そのため，送信データの最後にマーキングをつけてデータを送信を行ったとしても，バンク切替えを行ってしまうと，2 つのバンクにデータが混在してしまう可能性が出てくるため，ポーリングなどの手段は用いることができない．解決手段の 1 つとしてパケットフォーマットに定義されている受信側でのステータスを書き込むアドレスを指定できる機

能を使用し，SO-DIMM への受信が終了したというフラグを LLCM 上に書き込むことで，受信側がデータ受信完了のタイミングを知る方法が考えられる．しかし，この機能はパケットフォーマットでは定義されているが，DIMMnet-1 では省略されてしまったため，使用することができない．そこで他の方法として，はじめに送信側が BOTF，PUSH プリミティブでデータを送信し，受信側からの Ack が返った後で，改めて AOTF で受信完了通知を受信側の LLCM 領域に送信する．この方法では，データ送信後に再び AOTF 送信を行わなければならないため，通信時間のロスになるが，現状の DIMMnet-1 で確実にデータ受信完了のタイミングを知るために，今回の測定ではこの方式を採用した．

3. DIMM_Send,DIMM_Recv の実装

はじめにプログラムに参加するノード数だけ，LLCM 領域にランクごとのフラグ領域を設ける．フラグ領域は図６のようになっており，フラグ領域が有効かどうかを示す flag，受信側の残りバッファ数，送信側が使用するバッファ数，プロセスグループ ID（pgid），ランク番号に対応したアドレス番地からのオフセットを計算するためのバイトフラグが存在する．バイトフラグは 8bit で構成されており，最大で 255 個のバッファの管理が可能である．また，SO-DIMM 上の受信領域は図７のようになっており，各ランク毎の領域の先頭にはアドレスを連想するためのグループ ID（gid）が割り付けられている．１つのランクが確保できる受信領域の大きさ，バッファサイズは SO-DIMM のメモリ領域をどのように使用するか，またはプログラムに参加するランク数によって変化する．今回の実装ではランク毎に受信領域 1M バイト（バッファサイズ 4K バイト）を確保した．データの送信に関して，DIMM_Send，または DIMM_Recv が先行した場合は AOTF でリモートランク上のフラグ領域の更新を行う．後発の DIMM_Send，DIMM_Recv はフラグ領域が有効かどうかを確認し，有効であるなら，バイトフラグより 1 バイトで指定される相手側のバッファの準備が整っていることを知り，後発 DIMM_Send なら 464 バイトまで BOTF，それ以降は PUSH プリミティブを実行し，後発 DIMM_Recv なら PULL プリミティブを実行する．また，送信側のメッセージの使用バッファ数が受信側の残りバッファ数を越えていた場合，もしくはメッセージサイズが受信領域の大きさを越えていた場合は再送処理に入り，メッセージの大きさを複数回に分けて送受信を行う．

以上に述べた実装方式をまとめると，DIMMnet-1 上で実装される同期標準モードの DIMM_Send，DIMM_Recv，は以下のように実装される．

＜ DIMM_Recv 先行型＞

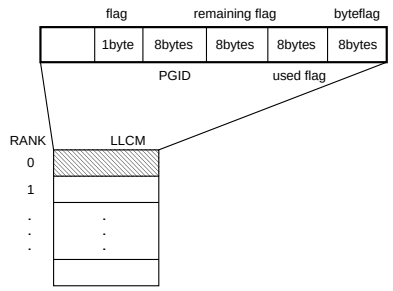

図 6 LLCM 上のフラグ領域

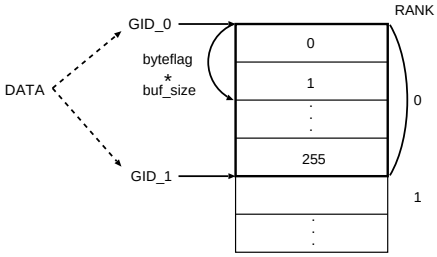

図 7 SO-DIMM 上の受信領域

- DIMM_Send 処理

（1） リモートランクに対応するフラグ領域を調べ，flag が有効で pgid が一致したなら２へ
（2） データサイズを調べ，464 バイト以下なら BOTF 送信，それ以上ならば PUSH プリミティブをデータ送信に使用
（3） フラグ領域から受信側の残りバッファ数を調べ，１度に送信可能なら９へ
（4） メッセージの使用バッファ数を再送処理通知として AOTF で送信
（5） データ送信
（6） データ送信通知を AOTF で送信
（7） ６に対する返信を受信したなら８へ
（8） 使用バッファ数が１でなければデクリメントし，５へ
（9） データ送信
（10） データ送信通知を AOTF で送信
（11） １０に対する返信を受信したなら終了

- DIMM_Recv 処理

（1） 送信側の自ランクに対応するフラグ領域を AOTF で更新
（2） 送信側からの AOTF に対する受信を行い，データ送信通知なら７へ，再送処理通知ならばデータ受信回数として３へ
（3） データ送信通知を受信したなら４へ
（4） ３に対する返信を AOTF で送信
（5） データ受信回数をデクリメントし，１でなければ３へ

(6) データ送信通知を受信したなら7へ
(7) 6に対する返信を AOTF で送信し，終了

＜ DIMM_Send 先行型＞

- DIMM_Send 処理

(1) 受信側の自ランクに対応するフラグ領域を AOTF で更新
(2) 受信側からの AOTF に対する受信を行い，データ送信通知なら7へ．再送処理通知なら（受信側からの）PULL 実行回数として3へ
(3) PULL 実行通知を受信したなら4へ
(4) 3に対する返信を AOTF で送信
(5) PULL 実行回数が1でなければデクリメントし，3へ
(6) データ送信通知を受信したなら7へ
(7) 6に対する返信を AOTF で送信し，終了

- DIMM_Recv 処理

(1) リモートランクに対応するフラグ領域を調べ，flag が有効で pgid が一致したなら2へ
(2) フラグ領域から送信側の残りバッファ数を調べ，1度の PULL プリミティブでデータの移動が終るなら7へ
(3) PULL 実行回数を再送処理通知として AOTF で返信
(4) PULL プリミティブの実行
(5) PULL 実行実行通知を AOTF で送信
(6) 5に対する返信を受信したなら7へ
(7) 使用バッファ数が1でなければデクリメントし，4へ
(8) PULL プリミティブの実行
(9) PULL 実行通知を AOTF で送信
(10) 8に対する返信を受信したなら終了

4. 評　　価

上記の実装を行った DIMM_Send，DIMM_Recv を用いて，DIMM_Recv が先行し，後発の DIMM_Send がデータ送信完了通知に対する返信を受信し，関数を抜けるまでの時間を 200 回計測し，その平均値よりバンド幅を算出した．また，ここで1つの目安として現状の実装に対する Myrinet を用いた MPI_Send, MPI_Recv との比較を行う．MPI_Send における通信性能については2つのランク間で，1方が

(1) MPI_Irecv()（ノンブロッキング受信）
(2) MPI_Barrier()（バリア同期）
(3) MPI_Send()（ブロッキング送信）
(4) MPI_Wait()（受信完了待ち）

を実行し，他方で

(1) MPI_Irecv()（ノンブロッキング受信）
(2) MPI_Barrier()（バリア同期）
(3) MPI_Wait()（受信完了待ち）
(4) MPI_Send()（ブロッキング送信）

を実行することで（round-trip time）/ 2 を 200 回測定し，バンド幅を算出した．評価環境を**表 1** にし，評価結果を図8に示す．

表 1　測定環境

測定環境	(a)	(b)
CPU/FSB	PentiumIII 850MHz/100MHz	
マザーボード	D6VAA	
OS	PentiumIII Linux (Kernel 2.4.2)	PentiumIII Linux (Kernel 2.4.18) SCORE 5.2/MPICH 1.2.4
NIC	DIMMnet-1(Martini2nd) 光モジュール	Myrinet M2M-PC132C-10450
スイッチ	RHiNET2/SW2 通過遅延 240ns	Myrinet M2M-DUAL-SW8

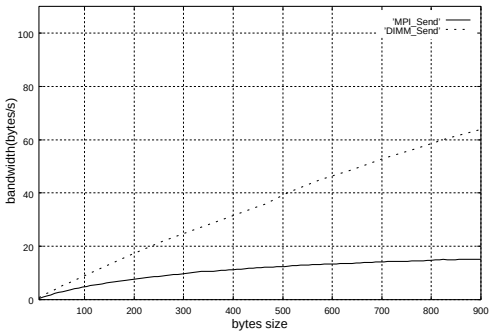

図 8　評価結果

4.1 評価結果について

現状の DIMM_Send，DIMM_Recv では tag などが実装されておらず，MPI_Send，MPI_Recv と全く同じ機能を持った対等な立場での評価ではないが，メッセージを他のノードに送信することを目的とした今回の実装に関しては，全体的にバンド幅が高いことが確認できた．今後は機能を追加し，グラフがどのように変化していくかを調べる必要がある．

5. 考　　察

5.1 フラグ領域の静的確保と動的確保

今回の実装では LLCM 領域に決められたランクごとのフラグ領域を設けていたが，この方式では，メモリ領域を無駄なく使用することは難しい．静的に割り当てを行っているため，1つのランクに対し，他の全てのランクからメッセージが届くという状態を除けば，必ず領域内に空きができてしまうからである．そこで動的にメモリを管理するために，はじめの先行した DIMM_Send，または DIMM_Recv が行うリモートランクへの登録を通常の AOTF 送信ではなく，未定義コマンドを用いた AOTF 送信で行う．そうすることにより，コア PU に処理が移り，あらかじめ書き込まれていた割り込み用プログラムを実行させることで，より柔軟なメモリ管理が可能となる．その時の

LLCM 領域の使用方法は，先頭が登録されたランク数の総数となっており，2line 以降にフラグデータが書き込まれる．起動したコア PU は，はじめに先頭のランク数をインクリメントした後，その総数から書き込むフラグデータの場所を判断し，データをパックして書き込む．この方法では動的なメモリ領域の管理は可能となるが，領域の先頭から複数のランク番号が混在することになり，登録数が多くなった場合，検索に時間が費されることが予想される．この回避方法として，はじめの静的な割り当てを行った上で動的な割り当てを行えばよいが，利点を得た分，欠点が増えてしまうことは避けられない．また，通常の AOTF を用いた round-trip time は 2.91μs なのに対し，はじめの 1 回に割り込み AOTF を用いた場合の round-trip time は 11.01μs となる．通常 AOTF の round-trip time より，片道にかかる AOTF の時間を 1.46μs としたとき，割り込み AOTF にかかる時間は 9.55μs と 6.5 倍になるため，小量のデータを送信する際には大きなボトルネックとなることが考えられる．そのため，FIFO の登録には割り込み AOTF を用い，動的なメモリの管理を行う場合，コア PU の性能向上を考えるか，全く別の方法で FIFO を実装する必要がある．

5.2 バンクの切替えのオーバヘッド

SO-DIMM を用いたデータの送信においては，ホスト側からの SO-DIMM への書き込みと書き込みを行ったデータを送信する場合，データの受信に関しては，受信したデータにホストがアクセスする場合にバンクの切替えが必要となる．DIMM_Send，DIMM_Recv では完全同期でデータの受信中に他のランクからデータが受信されることはないため，バンクを切替えるタイミングはそれほど難しくないが，非同期でのメッセージ交換の実装を考えた場合，バンクを切替えるタイミングは複雑になる．それは DIMM_Recv がデータの受信を完了したとしても，バンクを単純に切替えた場合，非同期で送信中のデータが両方の SO-DIMM に混在してしまうため，なんらかの制御が必要となるからである．現状での解決策の 1 つとして，各ランク毎にバンク切替え用のテーブルを用意し，非同期でデータを送信する際には，テーブルへの書き込みが可能で，データ送信中のフラグを書き込んでからデータの送信を行い，終了した時点でフラグの初期化を行うようにする．そして受信側がバンクを切替える際には，データを送信中のランクが存在するかを調べ，存在した場合はデータ送信中のランクに Nack を送信し，テーブルのデータ送信中のフラグに対して強制的に初期化を行った後に，バンクを切替えるようにするという方法を用いることで，データが両バンクに混在することは避けられるが，これとは別に非同期で書き込んだメッセージがどちらのバンクに存在するかを示すテーブルも必要になる．この他にもいくつか方法は考えられるが，現状では，バンクの切替えによるオーバヘッドは大きくなることが予想される．

6. ま と め

DIMMnet-1 は新情報処理開発機構（RWCP）で開発されたが，総務省の「戦略的情報通信研究開発推進制度」に採択され，2002 年度から総務省の研究補助金により DIMMnet-2 が開発されることになっている．本報告では MPI ライブラリの足掛かりとして DIMMnet-1 の送信機構を用いて DIMM_Send，DIMM_Recv の実装を行った．その結果，現状の DIMMnet-1 におけるいくつかの問題点が得られた．DIMMnet-2 への改善点としては以下のようなことがあげられる．

- データ受信が完了したというフラグを書き込むための，受信側へステータスを書き込むアドレスを指定する機能の追加
- バンク切替えのタイミングに対する一貫性のある管理

今後はこれらの改善点も含めて，DIMMnet-2 の設計，開発を今後行う予定である．

謝辞

本研究は総務省戦略的情報通信研究開発制度の一環として行われたものである．DIMMnet-1 に関しては新情報処理開発機構が推進してきた RWC（Real World Computing）プロジェクトの並列分散コンピューティング技術研究の一環として研究開発されたものである．

産業技術総合研究所の工藤氏，(株）日立製作所の山本氏，西氏，慶応義塾大学の渡部氏，元・慶応義塾大学の土屋氏，元・東京農工大学の須田氏，(株）日立 IT の今城氏，柏原氏，大杉氏をはじめ Martini LSI および DIMMnet-1 の開発に携わった全ての方々に感謝いたします．

また，DIMMnet-2 の開発に関する議論にご参加いただいている和歌山大学の国枝教授，上原講師，齋藤助手，慶応義塾大学の大塚氏，伊豆氏，北村氏に感謝いたします．

参 考 文 献

1) Myricom Corp.http://www.myri.com/
2) 田邊，山本，工藤：”メモリスロットに搭載されるネットワークインタフェース MEMnet ”，情報処理学会計算機アーキテクチャ研究会，Vol99, No67，pp.73-78(1999)
3) 西，多昌，西村，山本，工藤，天野：”LASN 用 8Gbps/port 8x8 One-chip スイッチ:RHiNET-2/SW ”，JSPP2000 pp173-180，(May 2000)
4) 山本，田邊，西，他：”高速性と柔軟性を併せ持つネットワークインタフェース用チップ Martini ”，情報処理学会研究報告 2000-ARC-140, pp,19-24(2000)
5) 山本，渡邊，土屋，他：”高性能計算をサポートするネットワークインタフェース用コントローラチップ Martini ”，JSPP2002，pp.35-42（2002）