計算機アーキテクチャ 119－3
（1996．8．27）

超並列計算機CP-PACSにおけるPVMの実装

松原正純† 服部正樹† 板倉憲一†
朴泰祐† 中村宏†† 中澤喜三郎†††

本研究では，科学技術計算用の超並列計算機CP-PACS上にメッセージパッシング・ライブラリであるPVMの実装を行なう．その際，CP-PACSの高速通信機構を活かして効率良く通信を行なえるようにする．

実装したPVMの性能評価はNAS並列ベンチマークを用いて，CP-PACSに元々備わっているライブラリ関数を直接用いてプログラミングした場合と比較することにより行なった．CP-PACS上にPVMを実装することにより，複雑な並列プログラミングを容易にすると共に，既存のPVM用プログラムをCP-PACS上で動かすことが可能となった．

Implementation PVM on CP-PACS

MASAZUMI MATSUBARA,† MASAKI HATTORI,†
KEN'ICHI ITAKURA,† TAISUKE BOKU,† HIROSHI NAKAMURA††
and KISABURO NAKAZAWA†††

In this reseach, we implement PVM on CP-PACS. CP-PACS, a massively parallel processor, is aimed to resolve large scale scientific problems. In order to get high-performance, we efficiently use the high-speed interprocessor communication facility implemented on CP-PACS.

On NAS parallel benchmarks, we evaluated the performance of PVM implemented on CP-PACS comparing to the program written with native data communication library. This reseach introduces more easiness and portability to parallel programming on CP-PACS.

1. はじめに

CP-PACS (Computational Physics by Parallel Array Computer System)[1]は計算物理学などの大規模科学技術計算を目的とした超並列計算機であり，現在筑波大学の計算物理学研究センターに於いて稼働中である．このCP-PACS上での並列プログラミングは，通常の逐次型言語に並列プロセス生成・プロセス間通信用のライブラリを追加したC及びFortranを用いることにより行なわれる．この追加された関数群は，通信の際のOSの関与を極力減らし，プロセス間通信を高速に行なうことができるという特徴を持っている．しかしながらOSのメッセージ処理の一部をユーザが行なうようになるので，プログラミングが複雑になり，ユーザへの負担が大きくなる．

そこで本研究では，多くのマシンで使用されているメッセージパッシング・ライブラリであるPVM (Parallel Virtual Machine)[2]を高速通信機構を活かした形でCP-PACSに実装することによって，ユーザの負担を軽減すると共に，これまでにPVMを用いて作られたプログラムをCP-PACS上でも効率良く動作できるようにすることを目的とする．

2. CP-PACS

まず，CP-PACSのシステム構成について説明する．

CP-PACSは分散メモリ型MIMD方式の超並列計算機で，1024台の計算専用プロセッサを持つ．各ノードプロセッサとしてはHewllet Packerd社のPA-RISC1.1チップにPVP-SW機構[1]を付加したRISCプロセッサを用いている．このプロセッサはクロック周波数150MHzで動作し，また2命令並列実行可能なsuperscalar方式を採用している．キャッシュはライトスルー／ダイレクトマップ方式の2レベルキャッシュを持つ．1次キャッシュは命令用データ用共に16KB，また2

† 筑波大学 電子・情報工学系
Institute of Imformation Sciences and Electronics, University of Tsukuba
†† 東京大学 先端科学技術研究センター
Research Center for Advanced Science and Technology, University of Tokyo
††† 電気通信大学 情報工学科
Department of Computer Science, University of Electro-Communications

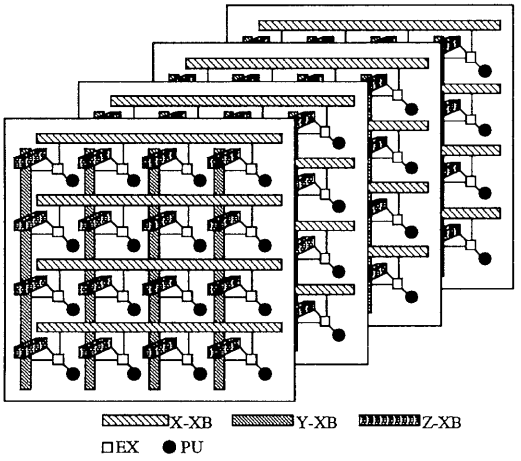

図1　3次元HXB（4×4×4）
Fig. 1　Three-dimensional HXB（4×4×4）

次キャッシュは命令用データ用共に512KBとなっている．各ノードには64MBの主記憶がある．

これらのノードを繋ぐネットワーク・トポロジとして3次元ハイパクロスバ網（HXB）が採用されている（図1）．CP-PACSでは，1024台の計算専用プロセッサ以外に，分散磁気ディスク記憶装置などを接続するためのアダプタが備わっている入出力プロセッサが64台あるので，8×17×8の3次元HXBとなる．

3次元HXBは，全ノードを3次元格子状に配置し，それらをクロスバ（以下，XBと略す）とエクスチェンジャ（以下，EXと略す）で相互結合したものである．XB，EXは共にクロスバ・スイッチによって構成される．EXはルータ機能を持っており，送受信ノードのアドレスが2次元方向以上で異なる場合は，各次元方向のXBの乗り換えを，EXを介して行なう．

HXBは隣接転送を無衝突で行なえるだけでなく，同一サイズ，もしくはサイズが異なる場合でも，総ノード数が等しいかより小さいメッシュ／トーラス・ネットワークの隣接転送も無衝突でシミュレートすることができる．さらに，ブロードキャスト転送が通常の1対1転送と同じ時間でできるといった利点もある[3]．

CP-PACSで用いているHXBの最大スループットは300MB/secである．このネットワーク上でメッセージパッシングによるノード間でのデータ交換が行なわれる．メッセージはwormholeルーティングによって転送され，この際固定ルーティング方式によりメッセージの通る経路を決定する．

CP-PACSでは高速なメッセージ転送立ち上げ及び転送スループットの向上のために，通常の転送モードの他に，ユーザ・アプリケーションから直接ネットワークへのメッセージ転送起動が行なえる高速転送モードが用意されている．この高速転送モードのことをリモートDMA転送と呼ぶが，次節で詳しく述べる．

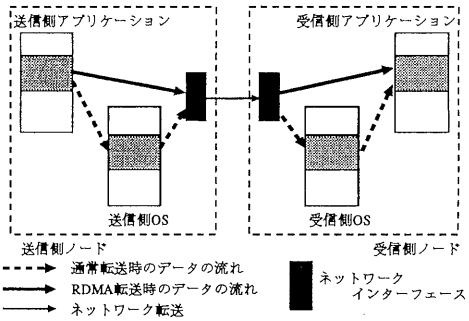

図2　リモートDMA転送による通信
Fig. 2　Communication using remote DMA

3. リモートDMA転送

リモートDMA転送（以下，RDMA転送と略す）は，送受信側とも通信データ領域となる物理メモリ領域をユーザプロセスの仮想アドレス空間に固定的に割り付けておき，その間で直接データ転送を行なう高速通信方式である（図2）．各ノードのユーザ仮想アドレス空間にマッピングされた物理メモリ同士で直接データ転送を行なうため，カーネル空間とユーザ空間との間でのデータコピーが発生せず，ネットワークの性能をそのままユーザに提供することが出来る．よって特に大きなメッセージを転送する場合に効率が良い．さらに，RDMA転送は軽いシステムコールによる起動が可能なので，転送立ち上げオーバヘッドが通常転送モードに比べてかなり小さい．このことはメッセージ長が短い場合の転送に非常に有利となる．

また，RDMA転送は，送信者主導のノード間メモリコピーとして考えられる．すなわち，RDMA転送は送信側が受信側のデータ格納領域を指定して直接データを書き込むことができる．よって，受信側は転送完了の確認だけを行ない，確認後は直ちにデータを利用できる．

以上のように通信を高速に行なえるといった長所がある反面，OSの関与を極力減らしたことにより並列プログラミングが複雑になるといった欠点がある．

RDMA転送を行なうプロセスは，データ転送を行なう前に通信データ領域を確保しなければならない．さらに受信側では確保した通信データ領域の一部を受信の単位となる通信フィールドと呼ばれるものに割り当てる必要がある．この通信フィールドは複数作成することができ，個々の通信フィールドにはプロセス内で一意なディスクリプタが与えられる．受信側ではこのディスクリプタを用いて，受信待ちをする通信フィールドを指定する．

RDMA転送は送信者主導型であるため，誤ったデータ送信が行なわれないよう，送信側では，送信する通

信フィールドに対する送信権を予め獲得しなければならない．送信権を得るとその通信フィールドに対応したディスクリプタが返され，送信はこのディスクリプタを指定することによって行なう．これにより，プログラム上で規定されていない不当な通信による受信側通信データ領域の破壊などが生じないような保護が行なわれている．

以上のことを全てユーザ・アプリケーション内で行う必要がある．また，RDMA 転送を行なう際には次のことに気をつける必要がある．

RDMA 転送では，転送データを直接受信側のユーザプロセス空間に書き込む．このため，受信領域のオーバーライトが発生する可能性がある．これは，受信側プロセスが受信データを完全に処理し終わる前に，次のデータの送出が行なわれることが，原理的に可能だからである．

受信は通信フィールド上でしか行なえない．また，送信するデータは確保した通信データ領域上になければならない．ただし，受信したデータをそのまま送信することは可能である．この制約により，プロセスの持つあらゆるデータが自由に RDMA 転送できるわけではないため，プログラムは場合によっては繁雑になる．

ユーザは上記のような制約も考慮に入れたうえで，プログラムを組まなくてはならない．

4. PVM

PVM の概略について説明する．

PVM は，ネットワークに接続された異機種 UNIX マシン群を，単一の分散メモリ型並列マシン（バーチャルマシン）として利用することを可能にするソフトウェアシステムである[2]．この PVM ソフトウェアの構成は，デーモンプロセスと PVM インターフェースライブラリの 2 つに大きく分けられる．デーモンプロセスは，バーチャルマシンを構成する全てのマシン上に常駐し，プロセス間通信の仲介や，バーチャルマシンを構成しているホストの管理などを行なう．ライブラリは，メッセージパッシング，プロセスの生成，プロセス間の協調動作，及びバーチャルマシンの再構成のための各種サブルーチンを提供する．

PVM では，プロセス間のデータ交換を send/receive 型のメッセージパッシングによって行なう．この際，転送するデータのバッファリングを行なう．

送信側ではまず送信バッファを初期化し，そして送信するデータをその送信バッファ内にパックする．それから受信側のタスク ID とメッセージタグを指定し送信する．受信側では送信側のタスク ID とメッセージタグを指定して受信待ちを行ない，一致するメッセージが到着すれば受信する．受信したらパックされたデータを受信バッファより取り出すことによりそのデータが利用可能となる．

PVM では，デーモンプロセスがプロセス間通信の仲介をしてくれる．したがって，ユーザは通信相手がどのホスト上に存在するかなどといったことを意識する必要がないため，比較的容易に通信プログラムを記述することができる．

5. PVM/CP-PACS

本節では，本研究で CP-PACS に実装する PVM について述べる．以降では，本研究で CP-PACS に実装する PVM のことを PVM/CP-PACS と呼ぶことにする．

5.1 実装方針

ノード間のメッセージ転送は PVM デーモンプロセスを介さず直接ライブラリの中で実現する．よって，本研究では，ライブラリのみを実装し，PVM デーモンプロセスが行なっていた役割は，基本的に全てライブラリの方で実現する．PVM ライブラリは，CP-PACS の並列プロセス管理ライブラリ，及び RDMA 転送ライブラリを用いて作成する．特に通信関数の実装に関しては，RDMA 転送の有効性が保たれるように工夫をする．

本研究では，RDMA 転送を容易に使用できるようにするためのインターフェースとして PVM を用いる．したがって，PVM のライブラリ全てを実装することは考えていない．PVM/CP-PACS は，CP-PACS のみで使用するので，中にはホストマシンの追加・削除を行なう関数のように必要のないものもある．また，エラー処理を行なう関数など実際にはあまり使用されないものもある．そこで今回はその中の必要最低限かつ通常のプログラムで多用される関数を実装する．

5.2 実装方法

CP-PACS 上では，転送するデータは通信データ領域内になければならないという制約がある．よって，RDMA 転送の高速性を利用するならば，プログラムのワーキングエリア全てを通信データ領域とし，そこから直接データの送信を行ないたいが，これはプログラミングの問題上，困難である．そこで，PVM/CP-PACS においてもバッファリングを行なうことにする．転送するデータのパックは通信データ領域とは別のメモリ領域に行ない，これを送信時には通信データ領域内の送信領域に移してから送信する．また，送信領域は固定サイズとしたので，この送信領域に入り切らないメッセージを転送する場合には送信領域にデータをコピーをしてから送信するという処理を，複数回行なう必要がある．以上のことは，RDMA 転送を用いているにも関わらず，通常転送機構における OS メモリ空間へのデータコピーと同じようなことを行なうことになる．しかしそれでも尚，OS への処理の移行によるオーバーヘッドは生じないため，RDMA 転送の高

速性をある程度維持できると考えられる．

送信されたデータは受信側のデータ受信用通信フィールドに格納される．この時，同時に複数のノードからデータが転送されてくるとオーバーライトが生じる可能性があるので，通信フィールドは各送信元ノード別に用意しておく．また，同一ノードから続けてデータが送られてきた場合も，オーバーライトが起こる可能性がある．そこで，次のデータ転送を行なう前に，前回の転送で送ったデータを通信フィールドから別のメモリ上へ移した（つまりバッファとなる通信フィールドが空になった）という ack が受信側から送信確認用フィールドに返ってこない限り，データを送れないように制御する．このハンドシェイクを行なうための送信確認用フィールドは別に設ける．

基本的には上記のようにして実装を行なうが，さらに出来る限り RDMA 転送の高速性を損なうことなく PVM を実装することを考える．

まず，データのパックは最初は送信領域に直接書き込む．そして送信領域が溢れたら通信データ領域とは別の領域を確保し，そこに書き込むようにする．こうすることで送信領域サイズよりも小さいメッセージは余計なデータコピーをともなわずに送信することが可能となる．

また，ダブルバッファリングを用いることにする．つまり，送受信側とも送信（受信）領域を 2 つ用意することにより，通信の高速化を図る．ダブルバッファを持つことにより，送信の際に受信側ノードより前回の送信に対する ack が返ってくるのを待たずに次の送信を行なうことができる．ただし，前々回の送信に対する ack は待つ必要がある．さらに，一方の送信領域を送信中に他方の送信領域へのデータコピーをオーバーラップすることができるため，処理の高速化が図れる．

以上の方法で PVM/CP-PACS の実装を行なった．なお，RDMA 転送用のライブラリ関数と PVM の関数は両方ともプログラミング言語として C 及び Fortran がサポートされている．PVM/CP-PACS は基本的には C を用いて作成されており，Fortran 版の PVM 関数は内部で C 版の PVM 関数を呼び出すことで実現できる．

PVM/CP-PACS には次のような制約がある．まず，1 ノードにつき 1 プロセスしか生成できない．子プロセスの生成は親プロセスによる 1 回のみしか許されない．そして，今回実装を見合わせた関数はプログラム中で使用することはできない．

6. 性能評価

本節では，CP-PACS でサポートされている RDMA 転送用の関数群を直接使用したプログラム（以下，RDMA/CP-PACS と呼ぶことにする）と PVM 用プログラムの実行結果を比較することにより，PVM/CP-PACS の性能の評価及び考察を行なう．

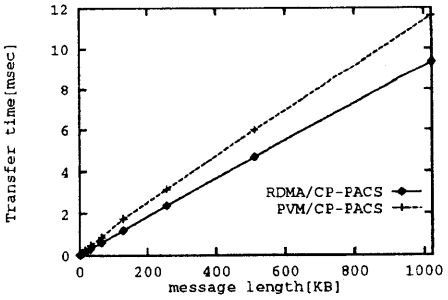

図3 データ転送時間
Fig. 3 Data transfer time

表1 PVM/CP-PACS の転送性能
Table 1 Throughput of PVM/CP-PACS

メッセージ長(KB)	Throughput(MB/sec)
1	19.1
4	37.9
16	88.5
64	108.9
256	108.5
1024	115.1

6.1 基本通信性能評価

まず，PVM/CP-PACS の通信性能を調べるために，単純な転送実験により転送レーテンシ及びオーバーヘッドを求める．この実験ではデータのパック／アンパックも時間測定の範囲に入っている．

ping-pong 転送に基づくデータ転送時間を測定した結果を**図3** に示す．転送するメッセージ長が短いときには PVM は RDMA の約 3 倍の転送時間を要するが，メッセージ長が長くなると約 1.3 倍の時間で転送できる．RDMA よりも性能が落ちる主な原因は，バッファリングを行なうことにより生ずる余分なデータコピーによるものと考えられる．また，RDMA/CP-PACS の転送オーバーヘッドは約 12μsec なのに対し，PVM/CP-PACS では最大で約 54μsec となっている．よって，メッセージ長が短いほどこの転送オーバーヘッドの差が効いてきて性能差が大きくなる．

表1 にパック／アンパック処理を含めた PVM/CP-PACS の転送性能を示す．メッセージ長が 64KB 以上のデータを転送しようとすると，PVM/CP-PACS では 2 回以上に分けて転送を行なう．しかし，ダブルバッファリングを用いているため，複数回に分けて転送を行なうことによる性能低下を防ぐことができている．

6.2 アプリケーションによる性能評価

次に，NAS 並列ベンチマークを用いて，PVM/CP-PACS の全体性能評価を行なう．

NAS 並列ベンチマーク（以下，NSAPB と略す）[4]

は，科学技術計算を中心とした並列計算機用のベンチマークである．NASPBには，5つのカーネルプログラムがあるがそのうちの次の4つのカーネルを用いて評価を行なう（表2）．

表2　NASPBのカーネルプログラム
Table 2　Kernel programs of NASPB

EP	合同乗算法による乱数の生成及びモンテカルロ法
CG	CG法による大規模疎行列の最小固有値の求解
FT	多次元FFTによる3次元偏微分方程式の求解
MG	マルチグリッド法によるポアソン方程式の求解

各カーネルには，問題規模がtiny，Class A，Class Bの3種類が用意されている．今回の測定ではClass Aを用いることにする．

NASPBでは，各マシンに適した並列化手法などに改良して良いことになっているが，RDMA/CP-PACS用プログラムとPVM用プログラムの計算量，転送量及び転送回数が異なると正しい比較評価が行なえないので，これらが等しくなるように注意を払った．また，測定できた全てのケースに関し，計算結果をチェックし，PVMプログラムが正常に実行できていることを確認した．

今回，最大128ノードを用いて実験を行なったが，残念ながらすべてのノード数において結果を得るということはできなかった（以降の表中の“-”となっているところは結果が得られなかった）．Class Aの問題は比較的問題サイズが大きいため，1ノード当たりのワーキングセットサイズが大き過ぎると主記憶が不足し，RDMA/CP-PACS，PVM/CP-PACSとも測定結果が得られないケースがあった．さらに，今回用いたプログラムはなるべく高速になるように最適化されているため，RDMA/CP-PACSでは通信データ領域を必要なだけ確保している．それに対して，PVM/CP-PACSでは予め決められたサイズしか通信データ領域を確保しないため，PVM/CP-PACSでは測定結果が得られているが，RDMA/CP-PACSでは得られてない箇所がある．

なお，以降の表中の#PUはノード数，RDMAはRDMA/CP-PACS用プログラムの実行に要した全実行時間（sec），PVMはPVM用プログラムをPVM/CP-PACSを用いて実行するのに要した実行時間（sec）である．また，RATIOはPVM/CP-PACSのRDMA/CP-PACSに対する実行時間の比率を示している．

(1)Kernel EP

EPの測定結果を表3に示す．

EPでは処理のほとんどが内部処理に費やされ，通信は最後に一回だけ行なわれる．したがって，全実行時間に占める通信時間の割合はかなり小さいため，RDMA/CP-PACSとPVM/CP-PACSではほとんど性能差が見られない．

表3　EPの測定結果
Table 3　The result of EP

#PU	RDMA	PVM	RATIO
2	219.255	222.904	1.017
4	110.226	111.908	1.015
8	55.041	55.351	1.006
16	27.708	27.711	1.000
32	13.834	13.861	1.002
64	6.953	6.966	1.002
128	3.462	3.471	1.003

(2)Kernel CG

CGの測定結果を表4に示す．

表4　CGの測定結果
Table 4　The result of CG

#PU	RDMA	PVM	RATIO
4	11.702	12.435	1.063
8	6.319	7.309	1.157
16	4.029	5.010	1.243
32	3.848	5.038	1.309
64	5.368	7.505	1.398

測定結果を見ると，ノード数が増えるに従ってPVM/CP-PACSのRDMA/CP-PACSに対する性能比が悪くなっている．今回測定に用いたCGのプログラムでは，通信のボトルネックは全ノードによるベクトルデータの一斉交換にあり，ここではノード数が増えても総転送量は変わらない．しかし一回当たりの転送量が減り，代わりに転送回数が増加する．よって転送オーバーヘッドの大きいPVM/CP-PACSの方が不利となり，この差が生じたものと考えられる．

(3)Kernel FT

FTの測定結果を表5に示す．

表5　FTの測定結果
Table 5　The result of FT

#PU	RDMA	PVM	RATIO
64	-	5.364	-
128	1.677	2.812	1.677

FTは128ノードの場合しか比較評価が取れていないが，そのRATIOは1.677と今回測定した中で最も高い．実験を行なったFTのプログラムの内部ではバタフライ転送を行なっている[5]．バタフライ転送の転送回数はノード数Pに対して$O(\log(P))$で増加し，一ノード当たりの転送量は$O(1/P)$で減少する．よってFTもノード数が増えるに従って，転送オーバーヘッドがネックになってくるものと考えられる．

(4)Kernel MG

MGの測定結果を表6に示す．

MGではノード数が増えると転送回数は増加するも

表 6　MG の測定結果
Table 6　The result of MG

#PU	RDMA	PVM	RATIO
16	-	6.243	-
32	-	3.478	-
64	2.605	3.343	1.283

のの，一回当たりの転送量も増加する．よってノード数が増加しても，CG や FT ほど RDMA/CP-PACS と PVM/CP-PACS の性能差が広がることはないものと思われる．残念ながら，今回の測定では 64 ノードの場合のみしか比較評価が取れていないため，確かめることはできなかった．

6.3 総合評価

PVM/CP-PACS では転送するメッセージ長が長い場合は，送信領域へのデータコピーなどのオーバーヘッドはあるものの，ある程度の性能を維持することができる．しかしながら，メッセージ長が短くなるほど大きな転送オーバーヘッドがネックになり，RDMA/CP-PACS の性能の 1/3 ほどに落ちる．CG の実行結果を見てみると，一回の転送量が少なく，そして転送回数が多くなってくると，転送オーバーヘッドの大きい PVM/CP-PACS では不利となってくるのが分かる．だが，CP-PACS が目的としている科学技術計算アプリケーションにおいては，ping-pong 転送のように実行の最初から最後まで頻繁に通信が起こるということは少ないと思われるので，悲観するほど性能が落ちるわけではない．今回測定した中では，最大でも 128 ノードで FT を行なった時の 1.677 倍で納まっている．

また，EP のように全体処理時間に対する通信時間の割合が少ないアプリケーションなどでは，RDMA/CP-PACS とほぼ同じ性能を保つことができる．このように通信の割合の低いアプリケーションでは，プログラミングの容易さと性能低下のトレードオフ，プログラムのポータビリティ等を考慮すると PVM/CP-PACS は非常に有効であると考えられる．

参考までに，PVM 用プログラムと RDMA/CP-PACS の通信部分に関するプログラム記述量を比較してみると，FT では RDMA が 97 行に対し PVM は 34 行と約 1/3，CG に至っては RDMA が 219 行に対し PVM は 32 行と約 1/7 の量で記述できる．

7. おわりに

本研究では，超並列計算機 CP-PACS に並列メッセージパッシング・ライブラリである PVM の実装を行なった．その際，通信には CP-PACS の高速通信機構であるリモート DMA 転送をできる限り活用した．

PVM/CP-PACS の性能を評価するために，まず単純な転送実験によって RDMA/CP-PACS と PVM/CP-PACS の通信性能の比較評価を行なった．さらに，実アプリケーションとして NAS 並列ベンチマークを用いて RDMA/CP-PACS と PVM/CP-PACS の全体性能の比較も行なった．

実験した結果，PVM/CP-PACS は RDMA/CP-PACS に比べて転送オーバーヘッドが大きいため，転送するメッセージ長が短いと約 1/3 程度の性能であることが分かった．しかしながら，メッセージ長が長くなると全転送時間に占める転送オーバーヘッドの割合が低くなり，またダブルバッファリングにより複数回に分けて転送することの性能低下を防いでいるので，より高い性能が得られた．アプリケーションによる性能評価では，一回の転送量が少なく，また転送回数が多いアプリケーションにおいてはあまり良い性能は発揮できなかった．しかし，全体の処理に対する通信処理の割合が少ないアプリケーションにおいては，RDMA/CP-PACS と比べてもそれほど劣ってはおらず，プログラミングの容易さを考慮すれば，十分に有効であることが分かった．

今回の性能測定実験では，主記憶の不足などで満足のいく測定を行なうことができなかった．今後の課題として，さらに多種多用な PVM 用プログラムを PVM/CP-PACS を用いて実行し，より綿密な性能測定を行なう必要がある．また，今回は実装を見合わせた PVM 関数のうち，実装できるものは実装する予定である．

さらに，本研究ではリモート DMA 転送を容易に使用できるようにするためのインターフェースとして PVM を用いたが，現在メッセージパッシングのインターフェースとして主流になりつつある MPI などを用いるのも興味深い研究である．

謝辞　本研究に関して貴重な御意見を頂いた，筑波大学西川博昭助教授ならびにアーキテクチャ研究室の諸氏に深く感謝します．なお，本研究の一部は創成的基礎研究費（08NP0401）の補助によるものである．

参考文献

1) 岩崎洋一ほか: 専用並列計算機による「場の物理」の研究, 新プログラムによる研究研究進捗状況報告, 筑波大学計算物理学研究センター (1994).
2) Geist, A. et al.: *PVM3 USER'S GUIDE AND REFERENCE MANUAL* (1994). ORNL/TM-12187.
3) 朴泰祐ほか: ハイパクロスバ・ネットワークの性能評価, 電子情報通信学会技術研究報告, pp. 41–48 (1993). CPSY93-40.
4) Bailey, D. et al.: THE NAS PARALLEL BENCHMARKS, RNR Technical Report RNR-94-007 (1994).
5) 服部正樹ほか: CP-PACS パイロットモデルにおける NAS 並列ベンチマークの評価, 情報処理学会研究報告, pp. 43–48 (1995). HPC95-57.