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

2007－HPC－111 (37)
2007／8／3

PACS-CSにおける隣接通信性能の高速化

住元真司[†] 大江和一[†] 久門耕一[†]
高橋大介[††] 朴泰祐[††] 佐藤三久[††]
藏増嘉伸[††] 吉江友照[††] 宇川彰[††]

PACS-CSの主要アプリケーションであるLattice-QCDプログラムは、隣接通信主体のアプリケーションである。しかし、MPIで書かれた隣接通信について性能評価したところ、メモリコピー性能で通信性能が律速されていることがわかった。これを解決するために、一般のEthernetを用いて通信バッファへのコピーが不要な通信ライブラリを開発した。本論文ではPACS-CS上での3次元隣接通信バンド幅性能を高速化するライブラリの設計と評価について述べる。

Increasing Neighbour Communication Performance Techniques for the PACS-CS System

Shinji Sumimoto,[†] Kazuichi Ooe,[†] Kouichi Kumon,[†]
Daisuke Takahashi,[††] Taisuke Boku,[††] Mitsuhisa Sato,[††]
Kuramashi, Yoshinobu,[††] Tomoteru Yoshie[††]
and Akira Ukawa [††]

The lattice-QCD program, which is important to run on the PACS-CS system, mainly uses neighbour communication. However, the neighbour communication performance using MPI shows it is saturated by memory copy processing performance. To solve the problem, we have developed a communication library without copying between communication buffers and user buffers using Ethernet network. This paper discusses a design and evaluation of communication library to increase 3D neighbour communication performance on the PACS-CS system.

1. はじめに

2006年に稼働を開始したPACS-CS[1]システムはGigabit Ethernetを6系統用いた3次元Hyper Crossbarネットワークを採用した2560ノードのPCクラスタである。現在、我々は本システム上でアプリケーションの高速化に取り組んでおり、高速化を実現するためにpmBufferArrayを用いた通信ライブラリの開発[2]を進めている。

pmBufferArrayを用いた通信は、様々な利点があるが、本稿では、pmBufferArrayを格子QCDプログラムの主要通信である3次元隣接通信の高速化の観点から議論し実装評価する。第2章でPACS-CSとその主要アプリケーションについて説明し、通信高速化の課題を示す。第3章でこれらの課題を解決するpmBufferArray採用の通信ライブラリについて説明する。第4章で格子QCDプログラムにおける通信性能高速化の課題について述べる。第5章でpmBufferArrayを用いて課題を解決する通信方式の設計、第6章で実装、第7章で評価する。

† 富士通研究所
FUJITSU LABORATRIES
†† 筑波大学
University of Tsukuba

2. PACS-CSと主要アプリケーション

PACS-CS[1]は、筑波大学で運用中のPCクラスタシステムで、3次元Hyper Crossbarネットワークを持つ。システムの概要を表1に示す。

表1 PACS-CSシステムの概要

ノード計算機	Intel LV Xeon 2.8GHz(EM64T) (Intel E7520, 2GB DDR2 SDRAM 4 x 64bit 133MHz PCI-X Bus)
計算ネットワーク	3次元 Hyper Crossbar Gigabit Ethernet(E1000)x6 JUMBO FRAME 利用
管理IOネットワーク	Gigabit Ethernet x2
ホストOS クラスタOS	Linux Fedora Core 3 for x86_64 SCore5.8.3[3]
ノード数	2,560 (16x16x10)

PACS-CS の特徴は、計算処理と通信処理のメモリバンド幅バランスを考慮した単一プロセッサノードと 3 次元 Hyper Crossbar 結合 (16x16x10) を採用している点である。

PACS-CS システムでは、SCore クラスタシステムソフトウェア [3] が採用され、3 次元 Hyper Crossbar 結合に対応した通信機構である PM/Ethernet-HXB[4] が開発され、利用されている。

PACS-CS 上での主要アプリケーションとしては、素粒子物理学における格子 QCD 計算と物性物理学における RS-DFT 計算がある。これら以外にも一般の MPI アプリケーションが実行される。

格子 QCD プログラムの動作： 格子 QCD プログラムにおける計算は、全体のデータ領域を 3 次元に分割し、QCD 計算と 3 次元 Hyper Crossbar 結合上での 6 方向の隣接双方向通信によりデータ交換を繰り返しながら、計算を進めていく。

3 次元 HyperCrossbar 結合上での 6 方向の隣接双方向通信において、転送されるデータは、データ配列上飛び飛びの位置に存在するため、メモリバンド幅で処理性能が律速される。

RS-DFT のプログラムの動作： RS-DFT プログラムの主要処理は、共役勾配法 (CG) もしくは最小残差法 (RMM) によるハミルトニアンの固有値・固有ベクトルの算出処理とグラムシュミット直交化処理を組み合わせた反復計算を行なう。この計算過程で、連続的にサイズが変化する MPI_Allreduce 処理が多用される。計算処理自体に占める通信処理の割合は大きくないが、大規模なデータ転送となるために通信オーバヘッドの削減が重要である。

以上主要 2 つのアプリケーションにおいては、隣接通信性能と集合通信性能は大きな比重を占めるため、これら通信の高速化は重要である。本稿では、特に格子 QCD の実行性能に大きく関わる隣接通信の高速化を中心に議論を進める。

3. pmBufferArray を用いた通信

本章では、PACS-CS 上の拡張通信ライブラリである pmBufferArray[2] を用いた通信の目的と特徴を述べる。

pmBufferArray を用いた通信は次の目的を満たすために開発されたものである。

- 通信処理におけるコピーオーバヘッドの削減
  - RDMA 通信のハードウェアサポートを持たないネットワークインターフェイスにおいてもユーザプログラム間での高い通信バンド幅性能を実現する。
  - 通信バッファの再利用が可能
    - 送信バッファ上に主要なデータを配置し修正しながら送信して再利用する。例えば送信バッファ自体をデータ領域として利用して、非同期 I/O のように送信の完了を確認した後に送信バッファを更新し、そのまま送信する。
    - 受信データのバッファをそのまま送信データとして送信する。例えば、MPI の broadcast などの collective 通信のように、同じデータをバイナリツリー形式などで転送する場合、あるいは受信データにそのまま何らかの処理を施して送信する場合に有効である。
- 通信手続きオーバヘッドの削減
  - PMv2 API[5] に見られる、通信デバイスの最大転送メッセージサイズを意識した通信処理ではなく、大きな通信バッファ単位で処理ができるため API 呼び出し回数を削減する。
- 扱いやすいプログラムインターフェイス
  - PMv2 API に見られる関数の利用制約の緩和
  - TAG によるマッチング受信の提供
  - ストライドコピーなど補助ライブラリの提供

表 2 pmBufferArray 利用のための主要 API の概要

関数名	関数の説明
pmGetSendBufferArray	送信 pmBufferArray を一つ割り当てる
pmSendBufferArray	送信 pmBufferArray を一つ送信する
pmSendDoneBufferArray	指定した pmBufferArray の送信完了確認
pmReleaseSend-BufferArray	指定した送信 pmBufferArray の解放
pmReceiveBufferArray	pmBufferArray の受信確認
pmReleaseReceive-BufferArray	受信 pmBufferArray の解放

表 2 に主要 API を示す。pmBufferArray を用いた通信は PMv2 に対する拡張機能として実装され、PMv2 を用いた通信と pmBufferArray を用いた通信は同時に実行することが可能である。

4. 格子 QCD プログラムにおける通信性能高速化の課題

本章では PACS-CS 上で pmBufferArray を活用しようとした場合のアプリケーション性能劣化の要因を整理して、その解決のアプローチについて述べる。

4.1 格子 QCD プログラムの通信パターンと pmBufferArray の隣接通信における適用方針

格子 QCD プログラムの典型的な通信パターンは、全ノードでの Allreduce 通信と、3 次元の隣接双方向通信の繰り返しが全体の大部分をしめる。本節では、格子 QCD プログラムへの pmBufferArray の適用を

検討する。

3 次元 6 方向隣接双方向通信

従来の MPI を用いた格子 QCD の隣接通信は次のとおりである。

(1) 計算データ領域から 6 方向の通信バッファにコピー
(2) 6 方向へのデータ送信: MPI_send(MPI_isend)
(3) 6 方向からのデータ受信: MPI_recv(MPI_irecv)
(4) 6 方向の通信バッファデータを計算データ領域へコピー

これらの手順においては、予め送受信用の通信バッファ領域が割り当てられており、この通信バッファにデータを一旦コピー後、6 方向に対する隣接双方向転送を行う。データ処理の特徴としては、計算データと通信バッファ間のデータコピーがメモリへのストライドアクセスになるため、メモリバンド幅で処理が律速される性質がある。この場合、Ethernet を用いた MPI 通信においてもデータコピーが発生するため、コピーを減らすことが通信性能向上の大きな課題である。

なお、データを送受信の方式についてはバリエーションがあり、同期通信、非同期通信、3 次元 6 方向通信の組み合わせと各処理間に barrier 処理を挿入することにより通信性能が変わることが知られている。

pmBufferArray を用いた通信では、MPI プログラムに対して容易に書き直すことが可能である。

初期化時に pmGetSendBufferArray 関数により、送信バッファを獲得後、次のような処理になる。

(1) 計算データ領域から 6 方向への通信バッファにコピー
(2) 6 方向へのデータ送信： pmSendBufferArray
(3) 6 方向からのデータ受信： pmReceiveBufferArray
(4) 6 方向の通信バッファデータを計算データ領域へコピー
(5) 6 方向の受信バッファを開放： pmReleaseReceiveBufferArray

Allreduce 通信

従来の MPI を用いたプログラムでは MPI_Allreduce を用いて書かれているが、pmBufferArray を用いた通信では、MPI_Allreduce のプログラムを pmBufferArray を用いて書き直す必要がある。その場合、受信した pmBufferArray に対して計算を実行し、そのまま送信するプログラムにすることが可能である。

また、pmBufferArray を用いた通信は、従来の PMv2 上で実現された MPI と同時に利用することも可能であるため、pmBufferArray を用いず MPI_Allreduce を使うという選択も可能である。

4.2 隣接通信における通信性能問題

PACS-CS において、格子 QCD プログラムの主要通信パターンである双方向の 3 次元隣接通信において一定以上のメッセージ以上の場合に急激な性能劣化を観測している。

表 3 3 次元隣接通信性能 (MPI) とパケット廃棄率 L(%)

	単方向		双方向	
パケット長	バンド幅	L%	バンド幅	L%
65536	316.1	0	401.7	0
73728	343.4	0	417.3	0
81920	360.3	0	413.5	0
90112	373.4	0	425.0	0
98304	380.3	0	418.8	0
106496	397.9	0	425.6	0
114688	401.2	0	26.3	0.02
122880	418.4	0	23.4	0.06
131072	420.0	0	22.1	0.09
262144	484.3	0	26.6	0.67
524288	467.1	0	32.7	0.63

表 3 に 3 次元隣接通信性能 (MPI) とパケット廃棄率の PACS-CS 上での測定結果を示す。結果は、256 ノードにおいて MPI を用いた 3 次元隣接通信プログラムにおける通信バンド幅とパケットの廃棄率を示している。パケット廃棄率は、Ethernet デバイスのレベルで送受信パケットを集計して算出している。

表 3 の結果では、3 次元の 3 方向ではパケット廃棄の発生はなく、6 方向の場合にのみ発生し、かつ、メッセージ長が 114,688 バイト以上の場合に発生している。以上より、このパケットの廃棄は、Ethernet スイッチのバッファオーバフローや一つの Ethernet スイッチのポートに複数のポートからのパケット転送が集中する場合に発生していると考えられる。

pmBufferArray を用いた通信においても、パケット廃棄が発生しない仕組みを検討する必要がある。

4.3 格子 QCD プログラムの通信性能高速化課題のまとめ

格子 QCD プログラムの通信性能の高速化に必要な課題を以下にまとめる。

- 冗長なコピーを削減したデータ転送方式
- パケットロスを抑えたデータ転送方式

故にこれらの課題を解決する 3 次元双方向の隣接通信を開発する。

5. 隣接通信方式の設計

本章では、第 4 章で述べた格子 QCD プログラムにおける 3 次元 6 方向の隣接双方向通信の課題を解決する通信方式を設計する。

第 4.3 節で述べた課題のうち、冗長コピーを削減するデータ転送方式は、pmBufferArray を用いたデータ転送で解決する。従って、本章ではパケットロスを抑えたデータ転送方式について議論する。

5.1 パケットロスを抑えたデータ転送方式

第 4 章で述べた格子 QCD プログラムにおける 3 次

元 6 方向の双方向隣接通信について議論する。

第 4.2 節で述べた問題で、パケット廃棄が発生しやすい通信パターンとして、例えば、X のプラスマイナス、Y のプラスマイナス、Z のプラスマイナスのように通信する場合が考えられる。一方、より発生しにくい通信パターンとしては、例えば、XYZ のプラス方向にデータを転送し、その後 XYZ のマイナス方向にデータ転送するといった通信パターンが考えられる。

後者の通信パターンにおいても、長時間実行を繰り返すとノード間の実行にずれが発生し、そのずれによって、ある軸のプラスマイナスの通信が重なりパケットロスが発生する場合もあるため、ノード間で一旦同期して通信を行うことにより、発生する可能性を少なくすることができる。この場合、ノード数の増加による同期オーバヘッドが問題となる。

明示的な同期を用いずに、送受信ノード間でのランデブー通信により同期をとる方法がある。本方式では相手からの受信準備完了のメッセージを受信して初めて相手にデータを送るため、ping-pong 転送の性能が重要となる。特に、準備完了メッセージをいかに早く検出するかが重要となる。

ping-pong 転送を用い双方向データ転送を高速化するためには、相手からのデータ受信をできる限り早く検出する必要がある。しかしながら、pmBufferArray を用いたデータ転送は複数のパケットがすべて受信された後に受信がわかるため、全てのパケットの受信を待っていては、同期に時間がかかるのが問題である。

表 4 通信バンド幅とパケット到着時間 (μs)

	64B	1500B	4096B	8192B
125MB/s	0.5	12.0	32.8	65.5
250MB/s	0.25	6.0	16.4	32.8
500MB/s	0.13	3.0	8.2	16.4
750MB/s	0.09	2.0	5.5	10.9
1000MB/s	0.06	1.5	4.1	8.2
1250MB/s	0.05	1.2	3.3	6.6

具体的な待ち時間を考えるため、表 4 に通信バンド幅とパケット到着時間を示す。例えば、PACS-CS は Gigabit Ethernet を 2 系統もっているため、片方向 250MB/s の転送能力がある。PACS-CS は 9KB の JUMBO FRAME を使っているため、例えば、8KB のデータ転送には、パケットの先頭が届いてから、最低でも 32.8μs パケット受信に時間がかかることになる。実際には、pmBufferArray を用いた転送では、複数のパケットを送受信するため、パケット数に比例した時間がかかる。

この時間を短くするために、pmPeekBufferArray という関数を導入する。pmPeekBufferArray の導入により、pmBufferArray の先頭パケット受信の検出が可能になる。更にデータ長が長い場合には、最初に、peek パケットという事前通知ヘッダだけを送り pmBufferArray の到着を知らせることで、32.8μs の待ち時間を 64 バイトヘッダで 0.25μs にまで短くすることが可能になる。

pmPeekBufferArray と peek パケットの代わりに短いメッセージを用いてユーザプログラムで同期を行う方法も考えられるが、ユーザプログラムでの受信処理が必要であるためこの分時間がかかる。これに対して pmPeekBufferArray と peek パケットの処理はデバイス割り込み処理の延長で処理されるため、通知にかかる時間は最小限で済む。

これらの工夫で ping-pong 通信での双方向データ通信の高速化を実現する。

6. 実　　装

pmBufferArray を PM/Ethernet-HXB[4] 上に実装した。図 1 に、PM/Ethernet-HXB のソフトウェア構成を示す。PM/Ethernet-HXB は、SCore の通信機構である PMv2[6),7)] の一つの通信デバイスとして実装されている。OS は Linux(2.6.14.4) である。pmBufferArray は PM/Ethernet-HXB に対する拡張として実装されている。以降、pmBufferArray を実装した PM/Ethernet-HXB を PM/Ethernet-HXB BAEX(Buffer Array EXtension) と呼ぶこととする。

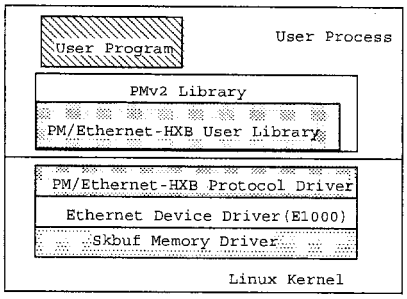

図 1 PM/Ethernet-HXB BAEX の構成

PM/Ethernet-HXB BAEX の実装では、PM/Ethernet-HXB[4] の送受信の枠組を利用して、別のチャネルで通信を行う。このため、従来の PMv2 通信と並行して pmBufferArray を用いた通信が実行可能である。

従来の PMv2 通信との違いは、同時並行可能な通信が PMv2 の場合は最大 31 であるのが、pmBufferArray を用いた通信の場合は、制御用バッファ領域の節約から 8 としているため、メッセージ長が短い場合のバースト転送性能が遅くなる。しかし、pmBufferArray は長いデータ長での利用を前提としているため影響は少ないと考えている。pmBufferArray に関する通信の詳細は論文[2)] を参考のこと。

7. 評　　価

本章では、PM/Ethernet-HXB BAEX の通信性能を評価する。表 5 に評価環境を示す。プロセッサ以外は PACS-CS と同等のものである。

表 5　性能評価クラスタ環境

ノード計算機 (8 ノード)	Intel Xeon 3.6GHz(1MB cache) (Intel 7520, 2GB DDR2 SDRAM, 4 x 64bit 133MHz PCI-X Bus)
Ethernet (1Gbps)	Intel Dual E1000 NIC x 4 Summit 7i, AXEL 144 port GigE
OS	Fedora Core 3 for x86_64, (2.6.14.4 Uni-Processor kernel)SCore5.8.3

性能評価としては、pmBufferArray を用いた基本通信性能、ping-pong を用いた隣接通信性能における pmPeekBufferArray と peek パケットの効果について検証する。

7.1 pmBufferArray を用いた基本通信性能

本節では、pmBufferArray を用いた基本通信バンド幅性能を評価する。

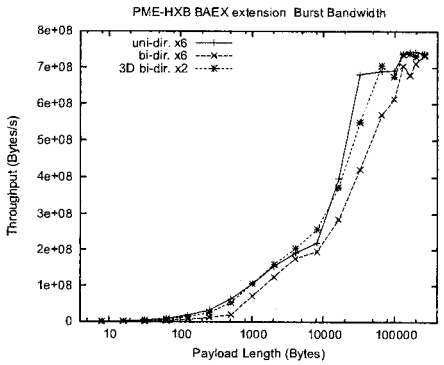

図 2　PM/Ethernet-HXB BAEX の転送バンド幅性能

表 6　PM/Ethernet-HXB BAEX の最大転送バンド幅性能

	通信バンド幅 MAX
1 次元 単方向 x6	737 MB/s
1 次元 双方向 x6	732 MB/s
3 次元 双方向 x2	736 MB/s

図 2 に PM/Ethernet-HXB BAEX の転送バンド幅性能を示す。理想的な場合は 750MB/s となる。図 2 で、1 次元で Gigabit Ethernet を 6 系統を用いた場合の単方向 (uni-dir x6) と双方向 (bi-dir x6) と 3 次元でそれぞれ、Gigabit Ethernet を 2 系統用いた場合の双方向 (3D bi-dir x2) の結果を示している。最大のバンド幅 (表 6) は、片方向のバンド幅で 730MB/s 以上の結果であり、物理リンク性能に比べ 97%以上の転送率を実現している。

7.2 Ping-pong 通信における pmPeekBufferArray と peek メッセージの効果

本節では、ping-pong 通信における、pmPeekBufferArray と peek メッセージの効果を評価する。この評価では 1 次元で Gigabit Ethernet を 6 本用いた場合の ping-pong 通信バンド性能を測定する。

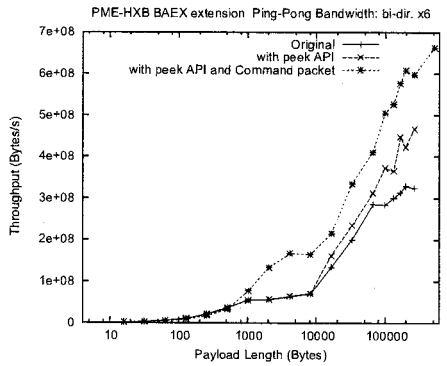

図 3　pmPeekBufferArray と peek メッセージの効果：双方向通信バンド幅性能

表 7　Ping-pong 転送の最大転送バンド幅性能

	最大通信バンド幅
オリジナル	324 MB/s
peek API あり	467 MB/s
peek API+peek パケットあり	662 MB/s

図 3 に片方向のバンド幅性能の測定結果、この時の最大転送バンド幅性能を表 7 を示す。理想的な場合は 750MB/s である。図 3 の結果より、pmPeekBufferArray の導入により 10KB 以上のデータ転送性能が改善されること、更に peek メッセージの導入により、512 バイト以上でのデータ転送性能が改善されることがわかる。また、表 7 の結果より、その最大通信バンド幅は、pmPeekBufferArray の導入により 1.44 倍、更に peek メッセージの導入により 2.0 倍に向上することがわかる。

7.3 Ping-pong 通信における 3 次元 6 方向隣接双方向通信性能の評価

本節では、ping-pong 通信における、3 次元 6 方向隣接双方向の通信性能を評価する。この測定では、3 方向の場合は 4 ノードを用い、6 方向の場合は 7 ノードを用いた結果である。また、通信パターンは 3 軸のプラス方向、次に 3 軸のマイナス方向を繰り返すものである。なお、本測定では、pmPeekBufferArray と peek メッセージを導入したプログラムを用いている。

図 4 は 3 方向と 6 方向の 3 次元隣接双方向通信の性能測定結果である。片方向の結果のため、理想的な場合の最大値は 750MB/s である。図 4 の結果より、3 方向の場合はメッセージ長 512KB まで性能劣化が発生していない。しかし、6 方向の場合は 110KB 以

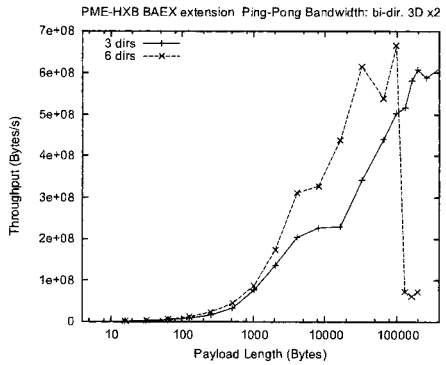

図 4 PM/Ethernet-HXB BAEX の隣接6方向バンド幅性能

上で 70MB/s 程度まで性能劣化が発生した。

これは Ethernet スイッチでのパケット廃棄の影響で、この時のパケット廃棄率は 0.26%であった。6 方向通信の場合、先にプラス方向に転送の後、マイナス方向に転送する。この時、プラス方向の転送が終わりきらないうちに、マイナス方向の転送が始まってしまうためにパケット廃棄が発生していると考えられる。

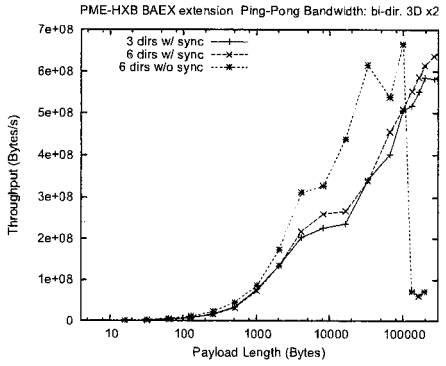

図 5 PM/Ethernet-HXB BAEX の隣接6方向バンド幅性能:sync 版

図 5 は、この問題を回避するため、マイナス方向の転送を実行する前に、プラス方向の転送が完了しているのを待つようにした結果である。図 5 の 6 dirs w/ sync の結果より、性能劣化が無くなったが、512 バイトから 96KB までの通信性能が悪化している。改善のための検討が必要であると考えている。

8. ま と め

本論文では PACS-CS 上での隣接通信性能を高速化するライブラリの設計と実装評価について述べた。3 次元隣接通信バンド幅性能を高速化するには、冗長なコピーを削減し、Ethernet でのパケットロスを抑えた通信を実現することが重要であることを述べた。

この課題を解決するために、pmBufferArray を用いた ping-pong 転送を用いて高速化する pmPeekBufferArray 関数の導入と peek パケットによる高速化手法を提案し、これを実現する通信ライブラリ PM/Ethernet-HXB BAEX を開発した。

PM/Ethernet-HXB BAEX の評価の結果、pmPeekBufferArray 関数の導入と peek パケットによる高速化手法は従来の転送方式に比べ、ping-pong バンド幅性能で 2.0 倍 (片方向 662MB/s, 双方向 1324MB/s) の効果を実現している。

一方、3 次元 6 方向転送においては、ある軸に対してプラス方向とマイナス方向の双方向通信が行われることにより、Ethernet ネットワークにおいてパケット廃棄が発生することがわかり、プラスとマイナス方向の同時転送を排除することにより、性能劣化が回避されることを示した。しかしながら、同時転送を排除すると性能の立ち上がりが悪いことがわかっている。

今後は、この性能劣化問題を調査するとともに、より大規模クラスタでの性能評価を進める。

謝辞 PACS-CS の MPI 通信性能問題調査に多大な協力を頂いた日立製作所の皆様に感謝致します。

参 考 文 献

1) 朴泰祐, 佐藤三久, 宇川彰. 計算科学のための超並列クラスタ PACS-CS の概要. 情報処理学会研究報告 05-HPC-103 (SWoPP'2005). 情報処理学会, August 2005.
2) 住元真司, 大江和一, 久門耕一, 朴泰祐, 佐藤三久, 宇川彰. PACS-CS のための高性能通信ライブラリインターフェイスの設計. 情報処理学会研究報告 07-HPC-107 (SWoPP'2007). 情報処理学会, July 2007.
3) SCore Cluster System Software: http://www.pccluster.org/.
4) 住元真司, 大江和一, 久門耕一, 朴泰祐, 佐藤三久, 宇川彰. 複数 Gigabit Ethernet を用いた PACS-CS のための高性能通信機構の設計と評価. SACSIS 2006 - 先進的計算基盤システムシンポジウム. 情報処理学会, May 2006.
5) PM 2.1 API :http://www.pccluster.org/score/dist/score/html/en/man/man3/PM.html.
6) 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.
7) Toshiyuki Takahashi, Shinji Sumimoto, Atsushi Hori, Hiroshi Harada, and Yutaka Ishikawa. PM2: A High Performance Communication Middleware for Heterogeneous Network Environments. In *Supercomputing 2000, IEEE and ACM SIGARCH, November, 2000, (Published by CD-ROM).*, November 2000.