配列転置データ転送を高速化する10 Gb Ethernetインタフェースカードの設計

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

本論文では，配列転置とデータ転送を同時に行う 10 Gb Ethernet ネットワークインタフェースカード（NIC）の設計について述べる．並列化された FFT や行列積演算といった数値計算アプリケーションの基盤となる処理では，配列転置をともなうデータ転送が多用されるため，ハードウェア化により，アプリケーションの高速化が可能である．配列転置データ転送を備える NIC の設計では，転置用バッファの大きさを抑えながら，PCI のデータ転送能力を高めることが，転置データ転送性能向上のために大きな課題となる．この課題を解決するため，実機上での PCI バスの転送性能を解析し，この解析結果に基づいた設計手法により，転置用バッファの大きさと転送性能を最適化し，転送性能を見積もった．これを FPGA 搭載の 10 Gb Ethernet NIC UZURA 上に実装し，評価した．その結果，設計どおりの通信性能を実現し，かつ，ホスト上で実行する方式と比較して，最大 10.5 倍の転置データ転送性能を実現した．また，この機能を，FFT ライブラリの 1 つである FFTW に適用し，評価した結果，転置転送処理時間を 40.8%削減，FFT にかかる処理時間全体を 34.9%削減し，この設計手法により限られたリソースで高い実行性能を実現した．

The Design of 10 Gb Ethernet Interface Card for Accelerating Data Transfer with Matrix Transposition

Kohta Nakashima,[†] Mitsuru Sato,[†] Masanori Goto,[†]
Shinji Sumimoto,[†] Kouichi Kumon[†] and Yutaka Ishikawa[††]

This paper discusses the design of 10 Gb Ethernet network interface card (NIC) to accelerate data transfer with matrix transposition. In applications such as parallelized FFT, data transfer with matrix transposition is in heavy usage. Therefore, the applications are able to accelerate using dedicated hardware. The design issue of the NIC which accelerates data transfer with matrix transposition is to satisfy both maximizing PCI data transfer performance and minimizing size of matrix transposition buffer. To solve this design issue, we use the design method based on the data analysis of PCI data transfer, optimize the size of matrix transposition buffer, and estimate its data transfer performance. We have implemented and evaluated the NIC which accelerates data transfer with matrix transposition on UZURA with FPGA and 10 Gb Ethernet interface. The evaluation results show that its data transfer performance achieves as designed and up to 10.4 times faster than that processing on host processor. We apply it to FFTW, one of FFT library, on UZURA, and evaluate its performance. The evaluation results show that hardware based data transfer with matrix transposition reduces 40.8% of processing time of matrix data transposition and data transfer, and 34.9% of total FFT processing time. These results show that our design achieves higher data transfer performance with limited hardware resource.

1. はじめに

PC プロセッサ性能の向上，Ethernet や専用インターコネクトの通信性能の向上により PC クラスタが普及している．PC プロセッサの動作周波数は 3 GHz を超え，インターコネクトも 10 Gbps を超えるものが普及しつつある．一方，プロセッサ性能とネットワークの性能向上に比べ，メモリ性能の向上は小さく，メモリバンド幅性能が逼迫している．

ソケットや TCP/IP といった標準的な通信方式では，主記憶内でのコピー処理をともなう．このコピー処理は，メモリへの負荷が高い処理であり，メモリバ

† 株式会社富士通研究所
Fujitsu Laboratories
†† 東京大学
The University of Tokyo

ンド幅を消費する．このため，メモリバンド幅がボトルネックとなり，十分な通信性能が達成できない場合がある．そこで，これを削減する手段としてアプリケーション上の転送データを直接 NIC へと転送する RDMA（Remote Direct Memory Access）通信が実装され，用いられている．

しかし，並列化された FFT（Fast Fourier Transform）や行列積演算といった数値計算アプリケーションの基盤となる処理では，配列転置をともなうデータ転送が多用される．配列転置では，連続するデータが不連続なアドレス上に分散配置されるため，RDMA による高速化は困難である．

我々は，並列アプリケーションが行うデータ転送処理を高効率に動作させる機構を提供する NIC の研究開発を行っている．アプリケーションが必要とする転送機能を NIC 上のハードウェアによりオフロードすることで，ホストが不得意とする処理を排除し，アプリケーション性能を向上させることができると考えている．また，このような NIC を実際に PC クラスタのノード間接続に適用し，高性能クラスタが実現できると考えている．

今回，我々は NIC へのオフロード処理の 1 つとして，配列転置処理をともなうデータ転送を取り上げ，これを高速化する NIC を開発した．配列転置データ転送を備える NIC の設計では，転置用バッファの大きさを抑えながら，PCI のデータ転送能力を高めることが転置データ転送性能向上のために大きな課題となる．

この課題を解決するため，実機上での PCI-X バスの転送性能を解析し，この解析結果に基づいた設計手法により，転置用バッファの大きさを最適化し，転送性能を見積もった．これを FPGA 搭載の 10 Gb Ethernet（10 GbE）NIC UZURA [1] 上に実装し，評価した．その結果，設計どおりの通信性能を実現し，かつ，ホスト上で実行する方式と比較して，最大 10.5 倍の転置データ転送性能を実現した．また，この機能を，FFT ライブラリの 1 つである FFTW [2] に適用し，評価した結果，転置転送処理時間を 40.8%削減，FFT にかかる処理時間全体を 34.9%削減し，この設計手法により高い実行性能を実現した．

本論文では，2 章で関連研究について述べる．3 章で配列転置データ転送処理，4 章でホストでの転置データ転送処理の問題点を述べ，NIC での転置データ転送の高速化が重要であることを述べる．5 章で NIC での転置データ転送機能を実現する場合の設計手法，6 章で UZURA を用いた実装設計，7 章でその評価を行い，8 章でまとめる．

2. 関連研究と提案方式の概要

本章では，関連研究と提案方式の概要について述べる．

2.1 関 連 研 究

ハードウェアによる転置機能については，以下のような研究がなされている．

2.1.1 FFT 専用機 FX

FFT 専用機である FX [3] では，コーナーターナと呼ばれる機構を用いてデータ並べ替えを高速化している．コーナーターナは，FFT 専用機のプロセッサパイプラインに用いられているデータ並べ替え用のレジスタであり，縦方向に格納したデータを横方向に取り出す機能を提供している．これにより転置を高速化している．プロセッサパイプライン上に実装されているので，コーナーターナ上では，あらかじめ設計されたデータ量の転置処理を行う．容量以上の配列転置については，論文 3) では言及されていない．

2.1.2 FPGA 搭載 NIC

NIC 上で転置機能を実現している事例として，論文 4) に述べられる手法がある．論文 4) では，FFTW における転置転送処理を，FPGA を搭載した NIC 上で実行している．この方式では，転置配列をいったんすべて NIC 上の SRAM に格納し，その後転送処理を行っている．このため，配列サイズが NIC 上のメモリ量で制限される．また，ホストから NIC への転送と NIC からネットワークへの転送がオーバラップしないため，転送時間が 2 倍になる．さらに，ネットワークは Gigabit Ethernet を用いているため，転送性能は高くない．

2.1.3 DIMMnet-2

DIMMnet-2 [5)~8)] は，高速ベクトルアクセス機能を提供する高機能メモリモジュールである．DIMMnet-2 は大容量 SO-DIMM を搭載し，SO-DIMM 上の非連続的なデータをベクトル型として定義し，ホストから連続した形でアクセスすることができる．用途は転置に限定されないが，この機能を用いると，本論文で対象としている配列転置も高速に処理することが可能となる [8]．さらに，DIMMnet-2 にはネットワークとのインタフェースが用意されており，ネットワーク越しの処理も可能となっている．

しかし，DIMMnet-2 では高速ベクトルアクセス機能を適用できる範囲が搭載 SO-DIMM 上のデータに限られており，ホストからは自由にアクセスできない領域となってしまう [7]．そのため，配列データを DIMMnet-

2 上の SO-DIMM 上に配置すると，配列の大きさが DIMMnet-2 に搭載される SO-DIMM の容量に制限される，配列にアクセスするたびに特殊な操作が必要となるなどの問題が生じる．逆に，ホストメモリ上に転置配列を配置すると，CPU によるコピーによって DIMMnet-2 へデータを転送する必要があり，CPU 負荷が高くなる．

また，現在の DIMMnet-2 の実装[6] では，基板サイズが大きく 1U サーバへの格納は物理的に困難である．

2.2 提案方式の概要

本論文では，以下の特徴を持つ I/O バス接続の転置機能付き NIC により，既存の研究の問題点を解決し，転置処理の高速化をはかる．

(1) アドレス連続方向のメモリアクセス
(2) 転置領域を小さな領域に分割
(3) 転置用領域の最適化

DIMMnet-2[7] では高速なストライドアクセスを実現できているが，これはベクトルアクセス機構とこの機構がアクセスするメモリとが密に結合されるためである．I/O バスを経由してメモリへアクセスする場合は，ストライドアクセスは非常に低速である．そこで，連続アドレス方向にメモリへアクセスし，データを取得し，取得したデータを NIC 上で並べ替える．これにより高い性能を達成する．

既存の研究では，転置配列全体を転置用メモリに格納しているため，転置用メモリサイズを超える領域は転置できない．そこで，転置領域を小さな領域に分割し，分割された領域ごとに転置処理を行う．これにより転置用メモリサイズを超える領域の転送を実現する．

NIC 上の転置用領域はリソースに限りがあるため，これを抑えつつ最大限の性能を達成するよう転置用バッファの大きさを最適化する．これによりさらに高い性能を達成する．

3. 配列転置データ転送処理

並列化された FFT 処理や行列積演算では，転置をともなうデータ転送が多発する．たとえば，2 次元 FFT では，2 次元の配列データを縦方向に分割し，各ノードに割り当て横方向に計算させ，その後，ノード全体が持つ配列を転置する．**図 1** では，node 0 上の A_1，A_2，A_3 はそれぞれ転置されて，node 1，2，3 へ転送される．転送される 2 次元配列 A_1，A_2，A_3 上では，アドレス連続方向は横方向であるのに対し，転送データの連続方向は縦方向である．このため，各転送データはアドレスが不連続な位置に配置される．

このような配列の転置をともなうデータを直接転送しようとすると，各断片ごとに転送起動する必要がある．各断片の大きさが非常に小さい場合は，転送起動の遅延時間が非常に大きくなり，非効率である．このため，実際には，転送データの各断片をコピーにより連続データに並べ替えてから転送する．なお，FFT では，各ノードが持つデータの大部分が転送される．このため，転置をともなって転送されるデータ量が非常に多い．

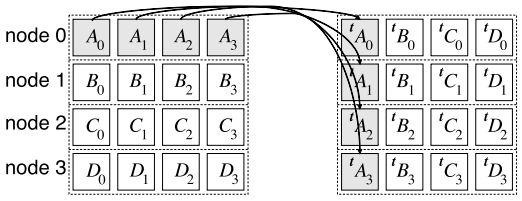


図 1 2 次元 FFT の転置をともなう転送
Fig. 1 Data transfer with matrix transposition in 2-D FFT.

4. NIC による転置データ転送処理の優位性

本章では，データ転送をともなう配列転置の処理方式として，ホストによる転置データ転送方式と NIC による転置データ転送方式を比較し，後者の優位性を示す．

4.1 ホストによる転置方式とその性能限界

ホストによる転置方式は，CPU がホストメモリ上でコピーによる転置処理（以降，転置コピー処理）を行い，その後，この領域を転送する方式である．この方式は，特殊なハードウェアを必要としない点が利点である．

一方，この方式はキャッシュの利用効率が悪く，大きな配列サイズで性能劣化する点が問題である．

転置コピー処理では，転置元領域か転送先領域のいずれかで必ず不連続なアドレスに配置される領域へアクセスする必要がある．このため，ストライドアクセスが多発する．CPU のメモリに対するアクセス単位はキャッシュライン単位であるが，ストライドアクセスの場合は，キャッシュライン中の有効なデータの割合が低い．たとえば，配列要素が 8 バイトでキャッシュラインサイズが 64 バイトの場合では，有効なデータの割合は，1/8 である．このように，キャッシュの利用効率が低いため，メモリバンド幅負荷が高くなり，性能は低下する．

転置コピー処理のチューニングにより，キャッシュ利用効率は改善されるが，これには限界がある．**表 1** の環境における以下 3 つの場合の転置コピー処理性能を**図 2** に示す．

表 1 評価環境
Table 1 Measurement environment.

CPU	Xeon 2.8 GHz
L2 キャッシュサイズ	512 KB（8-way）
チップセット	ServerWorks GC LE
メモリ	1 GB（DDR-266）
PCI バス	64 bit 133 MHz PCI-X × 2
ホスト OS	Fedora Core 3（Linux 2.6.15）

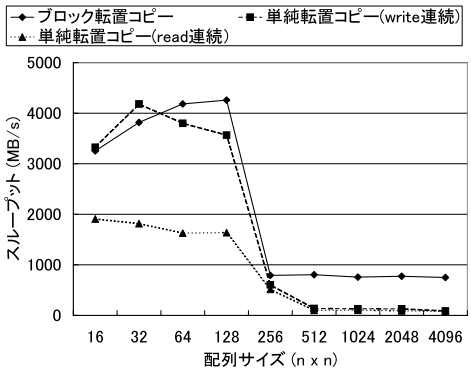


図 2 転置コピー処理性能
Fig. 2 Performance of copy with transposition.

- 単純転置コピー（write 連続）
- 単純転置コピー（read 連続）
- ブロック転置コピー

単純転置コピー（write 連続）の場合は，各要素ごとに転置元領域をアドレス不連続の縦方向に read し，転置先領域へアドレス連続方向の横方向に write するようにコピーする．単純転置コピー（read 連続）の場合は，逆に横方向に read し縦方向に write する．ブロック転置コピーの場合は，キャッシュ利用効率向上のためにブロック単位でコピー処理を行い，ラインコンフリクトを避けるためにカラーリングを行い，転送データのアドレス配置をずらしてコピー処理を行っている．図 2 では，カラーリング単位をパラメータとして変化させた場合での最も良い値を示している．なお，ブロック単位は，4×4 である．

この結果から，いずれの場合も配列サイズが 128×128 以下の場合は，十分高速に転置コピー処理を実行できることが分かる．これは，転送元/先領域がともにすべて L2 キャッシュに収まるためである．一方，いずれの場合も，256×256 以上の配列サイズの場合は，性能が低く転置転送処理のボトルネックとなることが分かる．配列サイズが 4096×4096 の場合では，転置コピー処理性能は，単純転置コピー（write 連続）の場合は 87.1 MB/s，単純転置コピー（read 連続）の場合は 79.0 MB/s，ブロック転置コピーの場合は 750 MB/s である．このように，チューニングによりある程度は性能改善できるものの，10 GbE（1.25 GB/s）のバンド幅と比較すると，チューニングした場合でもスループットは十分ではない．転置転送処理では，さらに，転置先領域から NIC への DMA 転送によるメモリ負荷がかかるため，実際の性能はさらに低下すると予想される．

4.2 NIC による転置方式

NIC による転置方式では，NIC が直接ホストメモリより配列データを取得し，NIC 上で転置処理を行う．このため，NIC 上に配列転置を実現するための機構を実装する必要がある．

一方で，ホストメモリ上でのコピー処理は不要となる．したがって，コピー処理分のメモリバンド幅の消費が削減される．

この方式では，配列転置機能を持つ特殊な NIC を必要とするが，メモリバンド幅の消費が少ない．このため，メモリバンド幅がボトルネックとなって性能が低下することを回避できる．これゆえに，我々は NIC 上での転置転送機能を実現する方式を選択した．

5. NIC 上での転置方式の設計手法

本章では転置データ転送を実現する NIC の設計について述べる．まず設計方針について議論し，設計の際に重要となるハードウェアリソースと通信性能の関係を整理し，どのように設計すべきかを議論する．

5.1 設 計 方 針

5.1.1 ホスト–NIC 間インタフェース

NIC の接続方式として，DIMMnet-2[5] のようにメモリバスに接続し，高機能メモリモジュールとして実装する方式と，通常の NIC のように I/O バスに接続する方式が考えられる．

高機能メモリモジュールとして実装する方式では，メモリモジュール上にメモリ領域を用意し，この領域に転置配列を配置する．この場合，メモリ領域と転置機能モジュールはともにメモリモジュール上にあり密に結合されるため，DIMMnet-2 のような汎用的なベクトルアクセス機構を高速に実現できる．しかし，高周波数のメモリバスに対応するためには，少なくとも読み出しに関しては，メモリモジュール上のメモリ領域をホストへ直接マップすることは実装上難しく[5]，ホストが転置配列にアクセスする際には特殊な操作を必要とする．また，転置配列はすべて高機能メモリモジュール上に配置する必要があるため，これを超える大きさの配列は取り扱えない．さらに，メモリスロットは，通常のメモリ接続を前提に設計されているため，物理的サイズや形状の制限が厳しく，様々なサーバへ

接続可能なモジュールの実装は困難である．

I/O バスに接続する方式では，転置配列は通常のホストメモリ上に配置される．メモリ領域と転置機能モジュールは I/O バスを経由して結合されるため，レイテンシが大きく，汎用的なベクトルアクセス機構は高速に実現できない．このため，高速に実行できる連続アドレスのバースト転送を用いる必要がある．しかし，転置配列へのアクセス時に特殊な操作を必要としない．また，転置配列はホストメモリ上に配置されるため，NIC のメモリ量によって転置配列の大きさは制限されない．さらに，I/O バススロットは，拡張カードの接続を前提に設計されているため，規格を満たすよう実装することは比較的容易である．また，接続するバススロットを持つサーバであれば，1U サーバのような高密度サーバに対しても接続可能である．

ソフトウェア側からの観点から転置配列アクセスの容易性は重要であると考える．また，より大規模な転置配列を取り扱えることが好ましい．さらに，我々は，PC クラスタのノード間接続に本 NIC を適用することを目標としている．現時点では，ラックマウントされた 1U サーバによって PC クラスタを構成する事例が多いため，I/O バス接続方式における接続の容易性は重要な点である．このため，I/O バス接続方式を採用する．

5.1.2 転置機能の実現方式

NIC の接続方式として I/O バス接続方式を用いる場合は，連続アドレスのバースト転送によりデータを取得する必要がある．

そこで，アドレスが連続する横方向に走査し DMA 転送を行い，複数の FIFO で転送データが到着するたびに転置処理を行う（**図 3**）．この転置用 FIFO は，FX のコーナーターナのような横方向のデータを縦方向に取り出す機能を実現している．この機構を用いることで，連続アドレスのバースト転送によりホストメモリ上のデータを取得でき，I/O バスの転送性能を高めることができる．なお，転送処理は DMA によって行われるため，転送処理中の CPU 負荷は低い．

5.2 設 計 目 標

前節で述べたように，NIC での転置データ転送には複数本の FIFO を転置用バッファとして用い，転置を行いながらデータ転送を行う．この FIFO バッファの本数とそれぞれの深さの積が NIC に搭載すべきメモリ量となる．

NIC の物理的制約（カードサイズ，消費電力など）から，NIC 上に搭載できるメモリ量には限界がある．そのため，転置用バッファとして転置対象となるデータすべてを保存できる領域を用意することができない．そのため，転置対象となるデータを小さなブロックに分割して，ブロックごとの転置を行いながら全体を転置するという方式をとる（**図 4**）．

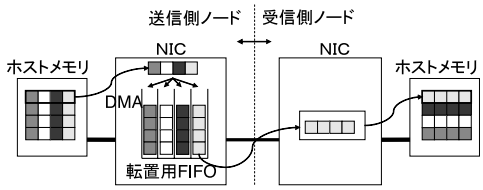


図 3 NIC による転置方式
Fig. 3 Transpose method by NIC.

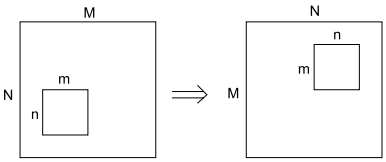


図 4 ブロック単位の転置
Fig. 4 Transposition in block unit.

このとき，小さなブロックのアドレスが連続する横方向の大きさ m が，NIC の転置用バッファとして用いる FIFO の数に相当し，アドレスが不連続な縦方向の大きさ n が各 FIFO の深さに相当する．

したがって，NIC に搭載すべきメモリ量を S とすると，

$$S = m \times n \tag{1}$$

となる．この S をある一定量以下に保ちつつ転送性能を十分に引き出すことが，本 NIC を設計するうえでの目標となる．

5.3 バースト転送長と I/O バス上の転送性能

NIC に搭載する FIFO の数 m とそれぞれの深さ n は，データの転送速度と深く関係する．

図 4 のように配列転置を行うと，横方向がアドレス連続方向であるので，転置元で NIC がメモリを読み出す際は，m ワードのバースト転送を n 回実行する．一方転置先で NIC がメモリに書き出す際は，n ワードのバースト転送を m 回実行する．したがって，これらの NIC が行うデータ転送を，限られたサイズで十分に性能を発揮できるようバッファ構成および NIC の I/O インタフェースを設計しなくてはならない．

一般に，I/O バス上でのバースト転送（DMA Read/Write など）は，実際のデータ転送に加えて，データ転送実行のためのオーバヘッド（リクエストの発行，バス利用権の確保など）が含まれる．いま，このオーバヘッドを X サイクル，1 ワードあたりのサイクル数を c サイクル/ワード，バースト転送量を w ワード

とすると，I/O バス上の実行データ転送性能 B_{io} は以下の式で表される．

$$B_{io} = B_{ideal} \times \frac{cw}{cw + X} \quad (2)$$

ただし，B_{ideal} は I/O バスの理論最大転送速度である．

式 (2) より，c，X が一定であれば，バースト転送量 w が大きいほど，実行転送速度 B_{io} は高くなる．一方で，搭載するメモリ量 S を一定量以下にする必要がある．式 (1) より，転送元の転送性能を向上させるために m を増加させると，逆に n は単調に減少し，転送先の転送性能は低下する．全体の転送性能は，転送元/先の転送性能のうち低い方に律速されるので，全体の転送性能を最大にするためには，転送元/先双方の転送性能を等しくする必要がある．したがって，転送元/先のオーバヘッドサイクル数を X_m，X_n とすると，式 (2) より，

$$\frac{cm}{cm + X_m} = \frac{cn}{cn + X_n}$$

すなわち，

$$m = \frac{X_m}{X_n} n \quad (3)$$

を満たすように決定すればよい．このとき，転置用配列の大きさは，

$$S_{trans} = m \times n = \frac{X_m}{X_n} n^2 \quad (4)$$

になる．

オーバヘッド X は，以下のような要因で変化する．

- 送信か受信か
- 転送要求だけを先行発行するパイプライン処理が可能か，またその先行発行できる要求の数
- I/O コントローラの種類

したがって，NIC 上のメモリ構成と I/O インタフェースの設計は，このような様々な要因で変化する X の値を実際に調査し，送信側の転送速度と受信側の転送性能が一致するよう，m や n の値を調整しなくてはならない．さらに，NIC の I/O インタフェースは，m や n に対してオーバヘッド X が十分小さくなるように設計しなくてはならない．

6. NIC 上での転置転送方式の実装

NIC 上での転置転送方式を実現する機能を，現在我々が開発中の実験用 NIC である UZURA 上に実装した．そこで，本章では実装対象となる UZURA の概要と UZURA 上への実装方式について述べる．

6.1 UZURA の概要

UZURA は FPGA 搭載の 10 GbE を用いた実験用 NIC である．UZURA の主な部分は，メインロジックである FPGA と 10 GbE MAC 処理を行う LSI から構成される（**図 5**）．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．また，ハーフサイズの PCI-X カードとして実装されており，PCI-X スロットを持つサーバであれば，1U サーバのような高密度サーバにも接続可能である．

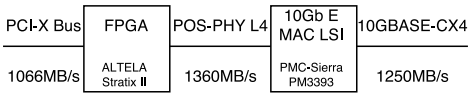


図 5 UZURA の構成図
Fig. 5 Structure of UZURA.

ホストとのインタフェースは PCI-X を採用している．これは FPGA でインタフェース処理を行うことができるようにしたためである．FPGA でホストインタフェース処理を行うことによって，NIC 上で処理のボトルネックとなりやすい，ホストとの通信部分に独自のハードウェアロジックを組み込むことが可能となる．

FPGA と 10 GbE MAC LSI との間は，POS-PHY Level 4（PL4）インタフェースで接続されている．PL4 のデータ転送は 1,360 MB/s で行われ，PL4 のプロトコル処理のオーバヘッドを考慮しても 10 Gbps の実効データ転送レートを確保できるようになっている．

10 GbE の媒体としては，できるだけ低レイテンシな通信を実現するために電気インタフェースを採用している．UZURA では 10 GbE MAC の XAUI インタフェースを利用して，10 GBASE-CX4 として動作するようにしている．これにより，UZURA 側および接続先のスイッチ側の両方で光–電気変換を省くことができ，レイテンシ低減の一助となっている．

6.2 UZURA に実装される機能

現在，UZURA 上には，以下の機能が実装されている．

- 通常 NIC 機能
- 通常 RDMA 機能
- 転置 RDMA 機能

通常 NIC 機能は，通常の Ethernet-NIC としての機能を提供している．すなわち，ホスト OS が提供する TCP/IP といったプロトコルを利用した通信処理を行うことができる．

通常 RDMA 機能は，ホストメモリ上の連続領域を別ホストメモリ上の連続領域へ転送する機能を提供している．この機能を用いることで，ホスト上でのコ

ピー処理を行わずに連続領域間でのデータ転送を行うことができる．

転置 RDMA 機能は，配列の転置処理を行いながら，RDMA 転送する機能を提供している．この機能を用いることで，ホスト上での転置のためのコピー処理を行わずに RDMA 転送を行うことができる．6.3 節では，この転置 RDMA 機能の実装について述べる．

なお，Ethernet フレーム上に独自の軽量 RDMA プロトコルを実装しており，通常 RDMA 機能と転置 RDMA 機能は，これを用いている．これにより，TCP/IP に代表される標準的なプロトコルにおける処理オーバヘッドを削減し，高速なデータ転送を実現できる．

6.3 転置 RDMA 機能の実装

実装した転置転送機能は，以下のブロックから構成される（**図 6**）．

- PCI 転送制御
  ホスト–NIC 間の DMA 転送を制御する．また，ホストに対するインタフェースとなる制御用レジスタが実装されている．
- パケットデータ生成制御
  ホストから転送されたデータを各パケットごとに転置用バッファに振り分け，配置する．
- 転置用バッファ
  ホストから転送されたパケットデータを保持する．
- 送信制御
  転置用バッファからパケットデータを取り出し，ヘッダを付加してネットワークへ送信する．
- 受信制御
  受信したパケットのヘッダを解析し，ホストへ転送する．

各ブロックは，以下のように動作し，転置転送機能を実現する．まず，ホストから転送起動されると，PCI 転送制御が，ホスト上の転送データをアドレスが連続する横方向に走査し，DMA 転送によりパケットデータ制御へ受け渡す．パケットデータ生成制御は，受け取った転送データを転置用バッファへ配置し，パケットのデータ部を生成する．このとき，受け取った転送データの各ワードは別々のパケットとなるので，それぞれ別々に各パケット用の領域に格納する．転置用バッファにパケットデータがすべて格納されたら，送信制御は，各パケット用の領域からパケットデータを取り出し，ヘッダを付加してパケットを生成する．パケットは，PL4 を経由して 10 GbE MAC へ転送する．

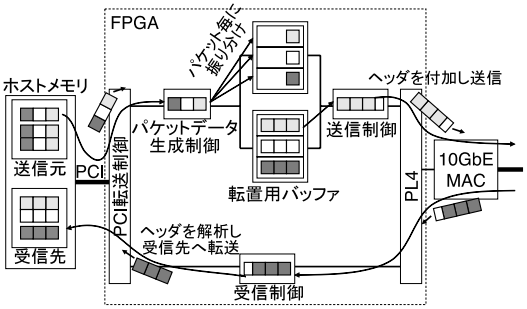


図 6 転置転送機能
Fig. 6 Structure of transposition module.

10 GbE MAC が受信したパケットは，PL4 を経由して受信制御へ転送される．受信制御では，ヘッダを解析し，パケットデータを指定されたアドレスへ転送する．

転置用バッファは，パケットデータ生成制御よりパケットデータがすべて格納された後に，送信制御により取り出される．したがって，連続的にデータ転送を行うために，2 つのバッファを用意する．そして，片方のバッファから送信制御がデータを取り出す間に，もう片方のバッファへパケットデータ生成制御がパケットデータを格納するよう処理する．

6.4 最適な転置用バッファサイズの設計

式 (2) に示すように，I/O バス上の転送性能は，バースト転送のサイクル数と，オーバヘッドサイクル数 X により決定される．実装に用いる PCI-X では，サーバや NIC の PCI コントローラの実装により X の値は，異なる．**表 2** に，UZURA を Xeon サーバと Opteron サーバに接続したときの，ホストから NIC，NIC からホスト方向におけるオーバヘッドサイクル数の値を示す．

送信側と受信側の転送性能を等しくするよう転置用バッファの縦と横の長さ m，n を決定すると，実装に用いる Xeon サーバでの環境では，$(X_m, X_n) = (12, 12)$ であるので，式 (3) より

$$S_{trans} = \frac{12}{12} n^2 = n^2$$

となる．

転置用バッファとして利用できる UZURA 上のメモリ領域は，64 KB である．また，ダブルバッファリングする必要があるため，1 個あたりは 32 KB 利用できる．1 ワードを 8 バイトとすると，最適な転置用バッファの縦と横のサイズは，式 (4) より $(m, n) = (64, 64)$

表 2 サーバによるオーバヘッドサイクル数の違い
Table 2 Difference of overhead cycle in each server.

サーバ	ホスト→ NIC（X_m）	NIC →ホスト（X_n）
Xeon	12	12
Opteron	15	17

表 3 Xeon サーバでの実効転送性能（算出値）
Table 3 Transfer performance in Xeon (estimate).

領域の大きさ（$m \times n$）	転送性能（MB/s）
64 KB（64×64）	898
16 KB（32×32）	776
4 KB（16×16）	610

表 4 IB-HCA を用いた場合のオーバヘッドサイクル数
Table 4 Overhead cycle of IB-HCA.

サーバ	ホスト→ NIC（X_m）	NIC →ホスト（X_n）
Xeon	30	14
Opteron	34	18

表 5 IB-HCA のパラメータを用いた場合の性能
Table 5 Performance estimation using IB-HCA parameter.

メモリ領域	最適形	対称形	性能差（%）
64 KB	807	726	11.2
16 KB	646	551	17.2
4 KB	463	371	24.8

と決定される．

また，仮に，メモリ領域が 16 KB，4 KB と限られている場合について，同様に算出すると，それぞれ，$(m, n) = (32, 32)$，$(16, 16)$ となる．これらの場合の実効転送性能を式 (2) より算出すると，**表 3** のようになる．

なお，opteron の場合も表 2 を用いて同様に算出可能であり，$(m, n) = (68, 60)$ のときに最適となり，853 MB/s である．対称形である $(m, n) = (64, 64)$ とした場合の性能は 843 MB/s であり，最適な場合の方が，1.3%高速である．

これらの UZURA での事例では，ホストから NIC 方向と NIC からホスト方向のオーバヘッドサイクル数の差が小さいため，対称形との大きな性能差はないが，実際には，サーバや NIC の PCI コントローラによっては，オーバヘッドサイクル数の差が大きくなる場合がある．

たとえば同じ Xeon サーバや Opteron サーバに富士通製 InfiniBand ホストチャネルアダプタ（IB-HCA）を接続した場合は，オーバヘッドサイクル数は転送方向で大きく異なる（**表 4**）．仮に，表 4 の Xeon の場合のようなオーバヘッドサイクル数である場合について性能を算出すると（**表 5**），最適な縦と横のサイズは，メモリ領域が 64 KB，16 KB，4 KB の場合，それぞれ，$(m, n) = (93, 44)$，$(46, 22)$，$(23, 11)$ であり，対称形よりも 11.2〜24.8%性能が高くなる．

7. 性能評価

本章では，UZURA 上に実装した転置転送機能の基本性能と転置転送機能を FFTW に適用した場合のアプリケーション性能評価について述べる．なお，いずれの評価にも表 1 に示す 2 台のサーバを用いて実機上で評価を行っている．

表 6 転置 RDMA の分割単位
Table 6 Unit of division for transpose RDMA.

配列サイズ（$n \times n$）	分割単位
16×16，32×32	16×16
64×64，128×128	32×32
256×256 以上	64×64

7.1 ホスト転置方式との性能比較

NIC 上での転置転送方式の有効性を検証するため，ホスト上での転置転送方式との性能比較を行う．このため，以下の 3 つの場合について評価を行った．

- 単純転置コピー + 通常 RDMA
- ブロック転置コピー + 通常 RDMA
- 転置 RDMA

ホストによる転置方式の性能として，単純転置コピー + 通常 RDMA とブロック転置コピー + 通常 RDMA の場合について評価する．どちらの処理も，ホスト上でコピーにより転置し，その後通常 RDMA 機能を用いて転送する．

単純転置コピー + 通常 RDMA の場合は，転置コピー処理を，3 章での単純転置コピー（write 連続）と同様に行う．転置コピー処理が完了した後に転送処理を行う．

ブロック転置コピー + 通常 RDMA の場合は，転置コピー処理を，3 章でのブロック転置コピーと同様に行う．さらに，転置コピー処理と転送処理をオーバラップさせる．なお，ブロック転置コピーのカラーリング単位やオーバラップさせるデータ量を変化させた場合の最も高い性能を評価値とする．

NIC による転置方式の性能として，転置 RDMA の場合を評価する．この場合は，転送元領域のデータを UZURA 上に実装した転置 RDMA 機能を用いて直接転送先領域へ転送する．なお，配列の分割単位は，配列のサイズごとに，**表 6** のようにした場合を評価値とする．

`double` 型（8 バイト）の 2 次元配列を転送データとし，配列サイズを 16×16 から 4096×4096 まで変化させたときのスループットを測定した．また，ホストによる転置方式の場合において，配列サイズがキャッシュに収まらない 256×256 以上の場合において，通常 RDMA 性能がボトルネックとなっていないことを確認するため，上記 3 つの場合に加えて，通常 RDMA によるデータ転送性能を測定した．これらの

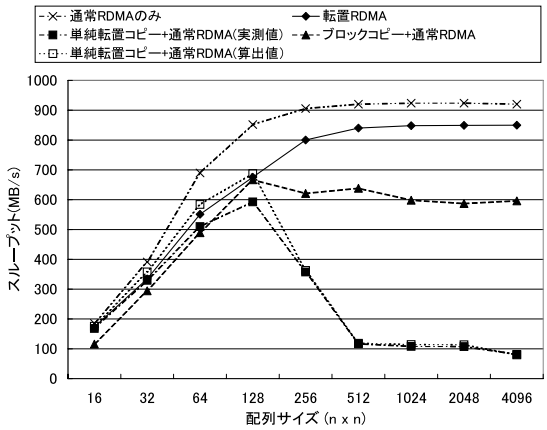


図 7 各転置転送処理の性能
Fig. 7 Data transfer performance with matrix transposition.

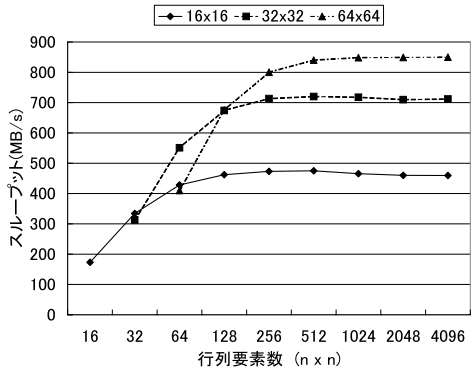


図 8 転置用バッファサイズごとの性能
Fig. 8 Performance in each transfer buffer size.

場合のスループットを**図 7** に示す．

配列サイズが 256 × 256 以上の場合において，通常 RDMA の性能は，単純転置コピー + 通常 RDMA とブロック転置コピー + 通常 RDMA の性能より十分高い．したがって，通常 RDMA 性能はボトルネックとなっていないことが確認でき，ボトルネックはコピーをともなう転置処理性能であるといえる．

また，転置 RDMA の性能は，単純転置コピー + 通常 RDMA とブロック転置コピー + 通常 RDMA の性能と比較して高い．このことから，転置 RDMA の場合は，ホスト上でのコピー処理によるボトルネックを解消できたといえる．この結果，単純転置コピー + 通常 RDMA と比較して，最大 10.5 倍に性能向上した．

なお，転置コピー+通常 RDMA の実測値と算出値の比較については，付録 A.1 を参照のこと．

7.2 転置 RDMA 性能予測と実測値の比較

転置用バッファサイズによる性能の変化を評価するため，転置用バッファサイズを 16 × 16，32 × 32，64 × 64 とした場合のスループットを測定した．

スループットを**図 8** に示す．16 × 16，32 × 32，64 × 64 の場合の最大性能は，それぞれ，475，720，850（MB/s）である．この値と表 3 での性能予想値を比較すると，いずれの場合も予想値より性能が低い．これは，ホストが転置 RDMA を起動する際のコストが含まれていないためである．

現在の実装では，配列を分割して転送する際に，分割された配列ごとに起動パラメータを NIC に設定する必要がある☆．PCI バス上のレジスタアクセスに必要な時間を計測すると，1 回あたり平均約 1 μs であった．このコストには，ソフトウェアのコスト，チップセットを経由するコスト，PCI バストランザクションのコストが含まれる．このコストを加えて算出すると☆☆，16 × 16，32 × 32，64 × 64 の場合の性能は，それぞれ，470，708，874（MB/s）となり，実測値とほぼ一致する．

また，配列サイズが 32 × 32 の場合は，転置用バッファサイズが 32 × 32 の場合よりも 16 × 16 の場合の方が性能が良い．これは，転置用バッファサイズが 16 × 16 の場合は，配列が 4 分割されるため，ホストから NIC へ転送する処理と NIC から通信路へ転送する処理とがオーバラップし転送時間が短くなるのに対し，32 × 32 の場合は，いったん，配列データをホストから NIC へすべて転送してから通信路へ転送するためこの 2 つの処理がオーバラップしないため，転送時間が大きくなるためである．同様のことが，配列サイズが 64 × 64 の場合にもいえる．

7.3 FFTW によるアプリケーション評価

UZURA 上に実装した転置 RDMA 機能を FFTW ライブラリの内部で使用される通信処理に適用した場合の性能評価を行った．本節では，まず FFTW の動作について説明し，性能評価について述べる．

7.3.1 FFTW の動作

FFTW ライブラリは，広く知られている FFT 演算機能を提供するライブラリである．FFTW ライブラリを用いて 2 次元の FFT 計算を行う場合，以下のように動作する．

(1) 各列ごとに 1 次元 FFT 計算を実行
(2) 計算結果を転置
(3) 各列ごとに 1 次元 FFT 計算を実行

☆ これは現在の実装による制限でありホストメモリ上にディスクリプタ領域を設けることによりこのコストを削減できる．

☆☆ 算出方法は，付録 A.2 を参照のこと．

表 7 処理されるデータ量
Table 7 Processing data amount.

方式	転置コピー	RDMA 転送
転置 RDMA 適用時	$1/n^2$	$(n-1)/n^2$
転置 RDMA 非適用時	$1/n$	$(n-1)/n^2$

(4) 計算結果を転置

並列環境でこの処理を行う場合は，入力データを分割して各ノードに割り当て，計算する．(2)，(4) の転置処理では，図 1 に示すように全ノードにまたがるデータ全体を転置する．この際，転置をともなうデータ転送が発生する．そこで，この転置転送処理に転置 RDMA を適用し，転置 RDMA 適用時と非適用時について評価した．n ノードでこの転置処理を行う場合，それぞれ以下のように実行する．

転置 RDMA 適用時： 各ノードは $1/n$ のデータを保持している．この保持しているデータをさらに n 分割し，各ノードへ転置 RDMA により転置転送する．相手が自ノードの場合は，ホスト上で転置コピーする．

転置 RDMA 非適用時： 各ノードは $1/n$ のデータを保持している．この保持しているデータ全体を，まず別バッファに転置コピーする．そして，転置コピーされたバッファを n 分割し，それぞれ各ノードへ通常 RDMA により転送する．転送先が自ノードである場合は，単なるコピーを行う．

以上のように処理するため，転置コピーされるデータ量，転置/通常 RDMA されるデータ量はそれぞれ**表 7** のようになる．

なお，転送処理では，PCI-X バスの帯域が片方向分しかないため，node0 → 1 の転送完了後 node1 → 0 への転送を行っている．

7.3.2 性 能 評 価

転置 RDMA 適用時と非適用時について，評価した．評価として，それぞれの場合について，FFTW 2.1.5 を用いた 2 次元配列 FFT 計算処理時間を測定した．評価は，`float` 型（4 バイト）の変数 2 つから構成する複素数を配列の要素とし，全体の配列サイズを 128×128 から 4096×4096 まで変化させた場合について行った．なお，評価には 2 ノードの環境を用いている．

配列サイズを変化させた場合における FFTW の実行時間を**図 9** に示す．各サイズにおける左側のグラフは転置 RDMA 非適用時の実行時間を，右側のグラフは転置 RDMA 適用時の実行時間を示す．

グラフ中での，「計算」は，(1)，(3) の 1 次元 FFT 計算処理時間を示す．「転置」は，転置コピー処理時間

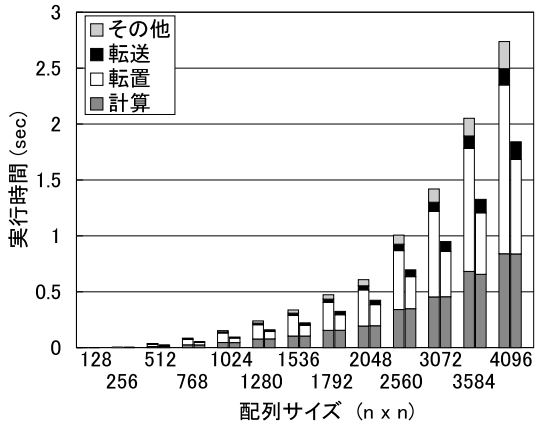


図 9 FFTW の実行時間
Fig. 9 Elapsed time of FFTW.

を示す．「転送」は，転置/通常 RDMA による転送処理時間を示す．処理時間には往復分が含まれる．「その他」は，(2)，(4) の転置処理に含まれる転置コピー処理と転送処理を除いた処理である．

いずれの場合も，コピーによる転置を約 1/2 に削減できていることが分かる．これは，ノード数が 2 であるため，転置 RDMA 非適用時は，転置コピーするデータ量は全体の 1/2 であるのに対し，転置 RDMA 適用時は，全体の 1/4 であるためである．また，ノード数を n 台に増やした場合を考えると，自ノード内で転置コピーされるデータ量は，転置 RDMA 非適用時は $1/n$ になるのに対し，転置 RDMA 適用時は $1/n^2$ になる．したがって，大規模なクラスタでは，転置 RDMA 適用の効果がより高くなると期待できる．

このように，ホスト上のコピーによる転置時間が削減できた結果，4096×4096 の場合は，転置 RDMA 処理時間を 40.8%削減し，全体として 34.9%性能が向上した．このように，アプリケーションに転置 RDMA 機能を適用した場合でも性能が向上することが分かる．

8. おわりに

本論文では，配列転置とデータ転送を同時に行う 10 Gb Ethernet ネットワークインタフェースカード（NIC）の設計について述べた．配列転置データ転送を備える NIC の設計では，転置用バッファの大きさを抑えながら，PCI のデータ転送能力を高めることが転置データ転送性能向上のために大きな課題となる．

この課題を解決するため，実機上での PCI バスの転送性能を解析し，この解析結果に基づいた設計手法により，転置用バッファの大きさと転送性能を定式化したうえで，転置用バッファの大きさを最適化し，

転送性能を見積もった．これを FPGA 搭載の 10 Gb Ethernet NIC UZURA 上に，実装し，実機上で評価した．その結果，設計どおりの通信性能を実現し，かつ，ホスト上で実行する方式と比較して，最大 10.5 倍の転置データ転送性能を実現した．また，この機能を，FFT ライブラリの 1 つである FFTW に適用し，評価した結果，転置転送処理時間を 40.8%削減，FFT にかかる処理時間全体を 34.9%削減し，この設計手法により限られたリソースで高い実行性能を実現した．

本論文の貢献は，限られたハードウェアリソースで最大限の転送性能を実現するためのネットワークインタフェースの設計である．転送性能とハードウェアリソース消費量を定式化し，式を解くことにより性能とハードウェアリソース量を見積もり，実装した性能結果がほぼ一致することを示したことである．

今後は，他の I/O バスチップセットのパラメータの調査結果をもとにし，PCI-Express への対応など，さらなる最適化を進める予定である．

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われた．

参 考 文 献

1) 中島耕太，佐藤　充，住元真司，久門耕一，石川裕：高性能通信処理オフロードエンジン UZURA 実現に向けて，情報処理学会研究報告，Vol.2003, No.27 (2003-ARC-152), pp.103–108 (2005).
2) Frigo, M. and Johnson, S.G.: FFTW: An Adaptive Software Architecture for the FFT, *Proc. 23rd International Conference on Acoustics, Speech and Signal Processing*, pp.1381–1384 (1998).
3) 近田義広：天体観測用の信号解析スーパープロセッサ，科学，Vol.54, No.10, pp.619–628 (1984).
4) Underwood, K., Sass, R. and Ligon, W.: Acceleration of a 2D-FFT on an Adaptable Computing Cluster, *Proc. 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machine* (*FCCM'01*) (2001).
5) 田邊　昇，濱田芳博，三橋彰浩，中條拓伯，天野英晴：メモリスロット装着型ネットワークインタフェース DIMMnet-2 の構想，情報処理学会研究報告，Vol.2003, No.27 (2003-ARC-152) (2003).
6) 北村　聡，濱田芳博，宮部保雄，伊澤　徹，宮代具隆，田邊　昇，中條拓伯，天野英晴：DIMMnet-2 ネットワークインタフェースコントローラの設計と実装，情報処理学会論文誌：コンピューティングシステム，Vol.46, No.SIG 12(ACS 11), pp.13–24 (2005).
7) 田邊　昇，安藤　宏，箱崎博孝，土肥康孝，中條拓伯，天野英晴：プリフェッチ機能を有するメモリモジュールによる PC 上での間接参照の高速化，情報処理学会論文誌：コンピューティングシステム，Vol.46, No.SIG 12(ACS 11), pp.1–12 (2005).
8) 田邊　昇，羅　徴哲，中條拓伯，箱崎博孝，安藤宏，土肥康孝，宮代具隆，北村　聡，天野英晴：プリフェッチ機能を有するメモリモジュールによる等間隔アクセスの高速化，ハイパフォーマンスコンピューティングと計算科学シンポジウム HPCS2006，pp.55–62 (2006).

付　　録

A.1 ホスト転置方式の性能検証

ホスト転置方式のうち，単純転置コピー ＋ 通常 RDMA については，転置コピー処理が完了した後に転送処理を行い，2 つの処理はオーバラップしない．このため，図 2 に示した単純コピー処理性能と図 7 に示した通常 RDMA 処理性能を用いて，性能を算出することができる．

図 7 における実測値と算出値を比較すると，配列サイズがキャッシュに収まらない 256×256 以上の場合は，ほぼ実測値と算出値が一致する．一方，配列サイズがキャッシュに収まる 128×128 以下の場合は，算出値に対して実測値は 4.1〜13.8%低下する．これは，算出に用いた単純転置コピー性能は，キャッシュ内部での転置処理性能であり，実際には，転置された領域をキャッシュ上から NIC へ転送する際に発生するオーバヘッドがあり，この分性能が低下するためである．

なお，ブロック転置コピー ＋ 通常 RDMA の場合については，転置コピー処理と転送処理がオーバラップするため，この際のメモリバスの挙動はチップセットに依存し，把握が困難である．このため，この場合は，転置コピー処理性能と通常 RDMA 性能から性能を算出することは困難である．

A.2 起動コストを含めた性能算出

現在の実装では，配列を分割して転送する際に，分割された配列ごとに起動パラメータを NIC に設定する必要がある．起動時のオーバヘッド時間を k，分割された行列のサイズを M とすると実際の性能 B_{actual} は，

$$B_{actual} = \frac{M}{\frac{M}{B_{io}} + k}$$

より算出できる．

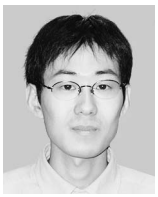
(平成 18 年 1 月 27 日受付)
(平成 18 年 5 月 24 日採録)

中島 耕太（正会員）

2000 年九州大学工学部電気情報工学科卒業．2002 年同大学大学院システム情報科学府情報工学専攻修士課程修了．同年富士通（株）入社．現在，（株）富士通研究所勤務．高速通信機構に関する研究開発に従事．

佐藤 充（正会員）

1969 年生．1992 年東京大学工学部電気工学科卒業．1997 年同大学大学院工学系研究科情報工学専攻博士課程修了．博士（工学）．同年富士通（株）入社．現在，（株）富士通研究所勤務．並列システムアーキテクチャの研究に従事．IEEE，ACM 各会員．

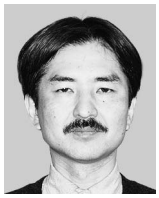


後藤 正徳

1999 年東京工業大学工学部情報工学科卒業．2001 年同大学大学院総合理工学研究科知能システム科学専攻修士課程修了．同年富士通（株）入社．現在，（株）富士通研究所勤務．アーキテクチャ，適応制御，オープンソースソフトウェアに興味を持つ．

住元 真司（正会員）

1986 年同志社大学工学部電子工学科卒業．同年富士通（株）入社．（株）富士通研究所にて並列オペレーティングシステム，並列分散システムソフトウェアの研究開発に従事．1997 年より新情報処理開発機構に出向．コモディティネットワークを用いた高速通信機構の研究開発，RWCP SCore2，SCore3 クラスタ等大規模 PC クラスタ開発に従事．2002 年より（株）富士通研究所にて高速通信機構の研究開発，理研スーパーコンバインドクラスタ等大規模 PC クラスタ，UHPC システムの開発等に従事，並列分散システムのアーキテクチャ，システムソフトウェア等に興味を持つ．平成 12 年度情報処理学会論文賞受賞，工学博士（慶應義塾大学大学院理工学研究科）．



久門 耕一（正会員）

1979 年東京大学工学部電気工学科卒業．1981 年同大学大学院電子工学専門課程修士課程修了．1984 年同大学院博士課程中退．同年（株）富士通研究所入社．現在，同社 IT コア研究所に所属．CPU，メモリ，並列計算機アーキテクチャに関する研究に従事．GCC，Linux カーネル等の改良にも興味を持つ．

石川 裕（正会員）

1987 年慶應義塾大学大学院工学研究科電気工学専攻博士課程修了．工学博士．同年電子技術総合研究所入所．1993 年技術研究組合新情報処理開発機構出向．2002 年より東京大学大学院情報理工学系研究科コンピュータ科学専攻．教授．クラスタ・グリッドシステムソフトウェア，高信頼システムソフトウェア開発技術，次世代高性能コンピュータシステム等に興味を持つ．