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

田邊 昇[†1] 安藤 宏[†2] 箱崎博孝[†2,☆]
土肥康孝[†2] 中條拓伯[†3] 天野英晴[†4]

パーソナルコンピュータ（PC）のメモリスロットに装着されるプリフェッチ機能を有するメモリモジュールを提案する．このデバイスは，Pentium4 などの COTS（Comercial Off-The-Shelf）型 MPU のキャッシュアーキテクチャの弱点を軽減することで，パーソナルスーパコンピュータに匹敵する実効性能を PC 上でも実現可能にすることを目指している．本論文では，プリフェッチ機構付きメモリモジュールの基本コンセプトとそのソフト的対応法について述べ，DIMMnet-2 プロトタイプにおける実装案を紹介する．NAS CG ベンチマークにおける適用を通し，間接参照の高速化に関する評価結果を示す．

Acceleration of Indirect Access on Personal Computer with a Memory Module with Prefetching Functions

Noboru Tanabe,[†1] Hiroshi Ando,[†2] Hirotaka Hakozaki,[†2,☆]
Yasunori Dohi,[†2] Hironori Nakajo[†3] and Hideharu Amano[†4]

A memory module with prefetching functions plugged into a memory slot of a PC is proposed. This device mitigates the weak points of cache architecture. In this way, it will realize personal supercomputer class performance with COTS (Comercial Off-The-Shelf) type MPU, such as Pentium4. In this paper, the concept and its software corresponding method of a memory module with prefetching functions, and introduction of its implementation plan on DIMMnet-2 network interface are presented. In addition, the evaluations for acceleration of indirect access with NAS CG benchmark are shown.

1. はじめに

半導体技術やアーキテクチャの進歩を背景にして，マイクロプロセッサ（MPU）は目覚ましい進歩をとげている．たとえば Pentium4 などはスーパコンピュータを凌駕する周波数で，スーパコンピュータの 1CPU に匹敵する演算性能をオフィスや一般家庭に提供している．量産効果に下支えされ，COTS（Comercial Off-The-Shelf）である MPU およびパーソナルコンピュータ（PC）の性能向上およびコストパフォーマンスの向上は目覚ましいものがある．

これらの COTS 部品である MPU はほぼ例外なくキャッシュアーキテクチャに基づいている．キャッシュアーキテクチャは主記憶の脆弱さを隠蔽できることが多いため，低コストな PC における主記憶は，ベクトル型スーパコンピュータのそれとは異なり，キャッシュが効かないアプリケーションに対しては演算能力にバランスしたものにはなっていない．

科学技術計算の分野では，SX-6i [1] のようにベクトル型スーパコンピュータのほぼ 1CPU 分を切り出したパーソナルスーパコンピュータ（PSC）製品上で，PC との性能差が数十倍に及ぶアプリケーションが多数存在することが報告されている．このようなアプリケーションにおいてはベクトル型スーパコンピュータは依然として存在意義がある．

キャッシュベースの MPU は記憶階層上で主記憶直前にあるキャッシュのラインサイズは伸びる傾向にあ

†1 株式会社東芝研究開発センター
Corporate Research and Development Center, Toshiba Corporation
†2 横浜国立大学
Yokohama National University
†3 東京農工大学
Tokyo University of Agriculture and Technology
†4 慶應義塾大学
Keio University
☆ 現在，日本電気株式会社
Presently with NEC Corporation

る．たとえばPentium3のL2キャッシュラインサイズは32バイトであるのに対し，Pentium4のL2キャッシュラインサイズは128バイト☆である．キャッシュベースのMPUはつねにライン単位で主記憶をアクセスする．

ところが，時間的に連続するメモリアクセス番地の距離の大きい不連続アクセスを主体とするアプリケーションでは，まさに必要なデータが128バイトのキャッシュラインの中に8バイトしか存在しないため，16倍のメモリバンド幅を消費してしまうというケースもある．このような状況が，たとえば科学技術計算においては間接参照を主体とするNAS CGベンチマークや2次元配列のカラム方向アクセスなどで見受けられる．それ以外でもデータベースのベンチマークであるWisconsinベンチマークの多くのクエリで同様の状況がある．それらで性質が代表される重要なアプリケーション群の存在を考慮すれば，その解決の工学的な価値は非常に高いと考えられる．

しかし，COTSであるPCとCOTSではないPSCの間には100倍程度の価格差があり，市場に劇的な変化がない限り後者を手軽に購入できる状態にはなり難い．ベクトル型の市場規模の衰退と寡占化にともない価格低下の望みは薄くなる方向に市場は進んでいる．よって，価格と性能の両面において，これらの両極端を埋めるコンピュータシステムの登場が望まれる．

さらに，COTSではないPSCの製品サイクルと，COTSであるPCの製品サイクルには歴然とした差がある．たとえば，2001年11月発表のSX6iの演算能力は8 GFLOPSであるのに対して，2005年1月現在のPentium4の性能は3.8 GHzの製品で7.6 GFLOPSにも達しており，少なくともピーク演算性能だけで比較すればPCは開発時から時間が経ち陳腐になってきた時点のPSCと遜色ないレベルまで進歩している．

以上のような観点から，キャッシュが効かないアプリケーションに対するアーキテクチャ面からの対応としては，COTSの目覚ましい性能向上を容易に取り込めるものでありつつ，キャッシュアーキテクチャの欠点を補う方式の研究開発が望ましい．

本研究は以上のような背景を鑑み，PCのメモリスロットに装着されるプリフェッチ機能を有するメモリモジュールを提案する．このデバイスは，Pentium4などのCOTS型MPUのキャッシュアーキテクチャの弱点を軽減することで，パーソナルスーパコンピュータ（PSC）に匹敵する性能をPC上でも実現可能にすることを目指している．**表1**に本研究の目標とする性能と価格のイメージを示す．

表1 目標とする性能と価格のイメージ
Table 1 Perspective of our goal of performance and price.

	PSC	目標	PC
cacheが効くアプリ性能	2	1	1
cacheが効かないアプリ性能	20	10	1
価格	200	2	1

本研究では，MPU，チップセット，マザーボードをはじめとするメモリモジュール以外のあらゆるハードウェアを価格性能比や製品サイクルに優れたCOTS部品を使いながら，着脱可能な独自ハードウェアを援用してキャッシュアーキテクチャの欠点を軽減することでCOTSのMPUの高い潜在的演算能力を引き出すという，独創性と経済性の高い手法を提案している．

さらに，本研究は現在のPCの主流であるDDR型のDIMMスロットにメモリ以外のデバイスを装着するという先例がない実装形態をとるため，電気的観点から実現可能であることを実証する必要がある．そこで，本研究の実機検証はDIMMスロットに装着されるネットワークインタフェース（NIC）であるDIMMnet-2[3]上で行われている．

本方式をDIMMnet-2上に適用することで，実質的に高速化された安価なノードPCが高速なネットワークで結合され，全体として巨大かつ高速アクセス可能な共有メモリを有するスーパコンピュータに見えるPCクラスタが構築される．

ただし，本論文における提案技術は必ずしもNICと共存して並列処理しなければ意味がないものではない．並列化に抵抗感を感じるより多くのユーザに対しても，PCの単体性能を向上させられる高付加価値メモリモジュールという提供形態もありうる．つまり，Myrinet[4]やQsNET[5]などのPCクラスタ用ネットワークインタフェースとは対象とするユーザ層や市場規模が異なる．このため，本手法はこれらにはない量産効果向上による低価格化の実現が期待できる．この点でも独創性と経済性を有する．

本報告では，2章でキャッシュアーキテクチャの問題点を指摘する．3章ではプリフェッチ機構付きメモリモジュールの基本コンセプトとそのソフト的対応法について述べる．4章ではそのDIMMnet-2における

☆ 厳密には，サーバ機で用いられるPentium4 Xeonのマルチプロセッサ環境に限定すればL2キャッシュラインサイズを64バイトに設定することも可能[2]であるが，本論文で取り扱うメモリバンド幅ネックとなる応用ではシングルプロセッサ環境での利用が中心であり，実現例であるDIMMnet-2もPCI-Xなどを持たない安価な一般ユーザ向けPC上で用いることを志向したものであるため，本論文ではL2キャッシュラインを128バイトとして扱うこととする．

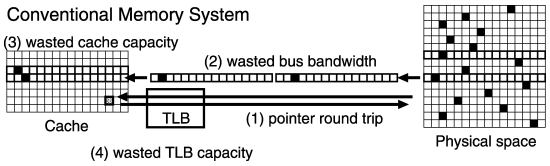

図 1 キャッシュを用いた CPU での間接配列参照の問題点
Fig. 1 Problems of indirect array accesses with cache-based CPU.

実装例を紹介する．5 章ではキャッシュアーキテクチャが苦手とする配列への間接参照を多用する応用として NAS CG ベンチマークを取りあげ，プリフェッチ機能を有するメモリモジュールの使用における，ソフトウェアも含めたその処理の過程を示す．6 章では提案システム用に改造された NAS CG ベンチマークプログラムが PC 上で示す性能から，実機上での処理性能を推定する．7 章で関連研究について述べ，8 章でまとめる．

2. キャッシュアーキテクチャの問題点

キャッシュアーキテクチャでは間接参照などの不連続アクセス時にキャッシュラインの利用効率が低下する．

たとえば NAS CG ベンチマーク，なかでも大きな配列サイズを有する Class C などのように，時間的に連続するメモリアクセス番地の距離の大きい不連続アクセスが必要なアプリケーションを 128 バイトのキャッシュラインを有する Pentium4 で実行する場合，必要なデータが 128 バイトのキャッシュラインの中に 8 バイトしか存在しない．キャッシュを用いた CPU を使って間接配列参照を行う際の問題点を**図 1** に示し，以下に列挙する．

(1) メモリと CPU 間のポインタまたはインデックスの往復にともなうメモリバンド幅やキャッシュエントリの浪費
(2) 有効データが少ないゆえのメモリバンド幅の浪費
(3) 有効データが少ないゆえのキャッシュエントリの浪費
(4) 有効データが少ないゆえの TLB エントリの浪費

一方，ベクトル型スーパコンピュータのリストベクトルアクセス命令によれば，ベクトルレジスタ上に必要なデータのみ圧縮格納されるため，上記の(2)〜(4)のような問題に対する適応性が高い．

キャッシュアーキテクチャの上記のような問題点は，間接参照のときだけでなく，ストライドがキャッシュラインサイズと比べて大きい等間隔アクセスを行うアプリケーション，たとえば主記憶上にデータを保持する主記憶データベースにおける Wisconsin ベンチマークの多くのクエリにおいても顕著に発生する．半導体メモリの容量拡大と価格低下の着実な進歩にともない，処理ネックがハードディスクから主記憶アクセスに移動する傾向が強まるため，主記憶上での等間隔アクセスの高速化のニーズが高まってきている．

一方，ベクトル型スーパコンピュータの等間隔ベクトルアクセス命令によれば，ベクトルレジスタ上に必要なデータのみ圧縮格納されるため，本来，このような問題に対する適応性が高い．

しかし，ベクトル型スーパコンピュータは大変高価なハードウェアであるうえ，HPC 市場の主流がベクトル型から PC クラスタなどの並列型の計算機に移行し，ソフトサポートもユーザ数の多いそれらのプラットフォーム側に移行してきていることにともない，今後の安定した入手性を疑問視する意見もある．

上記のような問題が，たとえば NAS CG ベンチマークや Wisconsin ベンチマークで代表される重要なアプリケーション群で見受けられ，それらがベクトル型スーパコンピュータという受け皿を失いつつある現状と合わさって，安価な PC 上でのそれらの問題の解決は工学的な価値が高い．

3. プリフェッチ機構付きメモリモジュール

3.1 基本コンセプト

メモリ空間にマップされたメモリモジュール側にあるバッファ（プリフェッチバッファ）へのプリフェッチコマンドを発行し，ホスト CPU から利用確率が高い状態に整えられたデータ群に対してブロックアクセスを行う．

その結果，キャッシュ・TLB・FSB・メモリバスの利用効率が向上する．メモリモジュールは着脱可能なので，MPU やチップセットを改造することなく，高性能かつ低価格な COTS を HPC 向けコンピュータとして有効に活用できる．

なお，プリフェッチコマンドには種々の実装法がありうるが，本研究ではベクトル転送命令について検討する．そのうち本論文で検討するのはベクトル間接ロード命令で，配列間接参照をベクトルレジスタに対して行う命令である．

図 2 は提案メモリモジュールにおける書き込み用（Write window）と読み出し用（Read Window）のバッファを有するプリフェッチ機能付き Window メモリ（Prefetch window）であり，これらがベクトルレジスタとして用いられる．これらは通信用にも用いることができる．ホストはこの Prefetch window のデー

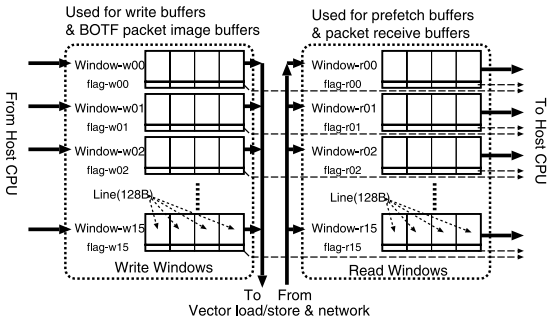

図 2 プリフェッチ機能付き Window メモリ
Fig. 2 Window memory with prefetching functions.

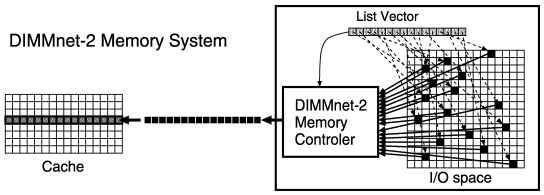

図 3 提案システムにおける間接配列参照時のキャッシュ挙動
Fig. 3 Behavior of cache on indirect array accesses with proposed system.

タ格納状態をフラグを通じて確認できる．たとえばホスト MPU の L2 キャッシュライン分のデータごとにフラグが 1 bit ついていて，ステータスレジスタを読むとこのフラグを確認できる．

これらのハード的な仕組みを使って間接配列参照を行った場合の動作と高速化原理を**図 3** に示す．前章で列挙した問題点がすべて解決される．

3.2 ソフトウェア的対応

本機構は，キャッシュメモリなどとは異なり，ソフト的に透過なハードではない．利用するためにはプログラムの変更またはコンパイラによる対応が必要である．

本機構を Pentium4 などのキャッシュベースの MPU から利用する際のリード時における指針を以下に示す．

(1) 主記憶の WriteBack 属性の領域にスクラッチパッド領域 A を確保する．

(2) 本機構を利用するためのプリフェッチコマンドを実際にデータを利用するより前に発行する．

(3) Read window から領域 A にブロックコピーを行うコードをプリフェッチが完了するころに実行する．

(4) Read window のアドレスに対応するキャッシュラインを再利用前にフラッシュする命令（Pentium4 の場合は CLFLUSH 命令）を実行する．

(5) 領域 A（実体はキャッシュ上）にあるデータを用いて，所望の処理を行う．プリフェッチと処理の時間差が少なければキャッシュへのアクセスだけで必要なデータが得られる．

一方，ライト時における高速化の指針は DIMMnet-1 における BOTF における Window メモリへの CPU からの書き込みの高速化指針[6] とほぼ同様であり，Window メモリにコピーし終えたら，送信を指示するキック操作の代わりにベクトルストアコマンドを起動する．

4. DIMMnet-2 における実装

4.1 資源モデル

プログラマから見える 1 個の DIMMnet-2 上の資源を以下に列挙する．(12)，(13) は特権領域にマップされる．

(1) SO-DIMM 領域（4 GB×1〜16）
(2) LLCM 領域（数百 KB）
(3) WW：Write Window（64 bit 幅 64 語 ×N 本）
(4) RW：Read Window（64 bit 幅 64 語 ×N 本）
(5) SR：Scalar Register（7 bit 幅 8 語）
(6) WWSR：WW Status Register（64 bit 幅 1 語）
(7) RWSR：RW Status Register（64 bit 幅 1 語）
(8) CMR：Command Register（64 bit 幅 16 語）
(9) CMSR：Command Status Register（16 bit 幅 1 語）
(10) CCR：Command Count Register（8 bit 幅 16 語）
(11) CPR：Current Page Register（16 bit 幅 4 語）
(12) CPMR：Current Page Mask Register（16 bit 幅 4 語）
(13) PGIR：Process Group ID Register（16 bit 幅 4 語）

(1) は CMR に書き込まれたコマンドにより WW または RW を介してアクセスする．つまり，ホストのメモリ空間には基本的にはマップされておらず，DIMM スロットに割り当てられた最大物理メモリ容量の壁を越える一種の I/O 空間（大域仮想アドレス空間）に存在する．大域仮想アドレス空間は rank（12 bit），page（4 bit），offset（32 bit）の合計 48 bit で指定されるノード間にまたがる共有空間である．この領域は大域仮想アドレス空間にマップされているので，後述の転送命令でリモートノードからもアクセスが可能である．ただし，各ユーザは CPMR により許可された page にしかアクセスできない．

(3)，(4)，(6)，(7) は DIMMnet-2 における書き込み用と読み出し用のバッファとして考案された図 2 に示されるプリフェッチ機能付き Window メモリの構成要素であり，これらがベクトルレジスタとして用いられる．

(2)〜(13) はホストの DIMM スロットに割り当てられた物理アドレス空間にマップされ，通常のメモリアクセスによりアクセス可能とする．さらに，これらもプロテクション上問題のないものは大域仮想アドレス空間にマップされているので，後述の転送命令でアクセスが可能である．つまり RW もリモートストアの書き込み対象となる．これは，超並列計算機 TS/1 におけるリモートのベクトルレジスタ間のチェイニングを行うプロセッサ間チェイニング[7] の変形である．

4.2 ベクトル転送命令

DIMMnet-2 におけるベクトル転送命令を列挙する．命令は 64 bit 形式であり，ホストから CMR に書き込まれる．Packed と通常の間接アクセスの違いは Packed は命令中で指定する 1〜4 個の要素に対する命令であるが，通常の方は Window 上のリストベクトルで要素を指定する点である．移動と複製の違いは，ステータスレジスタのフラグがクリアされるか否かの違いである．VT 命令のみ直後に書き込まれた命令と組み合せてリモートアクセスを行う命令と解釈される．ベクトルクリア命令は指定された Window 上のデータを破棄し，対応するフラグをクリアする．ベクトル同期命令はその命令が起動されるまでに起動済みの命令が実行完了するまで新たな命令の実行を待たせる．以下，XW は WW と RW の総称，type はデータ型，iter は繰返し回数，displace は XW 上での開始位置，top は大域仮想空間上の offset，stride は要素間間隔（byte 数），indexXW は index ベクトルを保持する window である．

(1) ベクトル連続ロード：
VL(type,iter,displace,RWi,top)
(2) ベクトル等間隔ロード：
VLS(type,iter,displace,RWi,top,stride)
(3) ベクトル間接ロード：
VLI(type,iter,displace,RWi,top,indexXWj)
(4) Packed ベクトル間接ロード：
VLPI(type,iter,displace,RWi,top,index1,index2,index3)
(5) ベクトル連続ストア：
VS(type,iter,displace,WWi,top)
(6) ベクトル等間隔ストア：
VSS(type,iter,displace,WWi,top,stride)
(7) ベクトル間接ストア：
VSI(type,iter,displace,WWi,top,indexXWj)
(8) Packed ベクトル間接ストア：
VSPI(type,iter,displace,WWi,top,index1,index2,index3)
(9) ベクトル移動：
VM(type,iter,displace,sourceXWi,destinationXWj)
(10) ベクトル複製：
VC(type,iter,displace,sourceXWi,destinationXWj)
(11) ベクトル転送拡張：VT(rank,page)
(12) ベクトルクリア：VCLR(XWj)
(13) ベクトル同期：VSYN()

5. NAS CG における適用例

5.1 NAS CG ベンチマーク

NAS CG は NASA 提供のハイパフォーマンスコンピュータ評価用ベンチマークの 1 つで，共役勾配法による疎行列の最小固有値を求めるプログラムである．問題のサイズが小さい方から順にクラス S，W，A，B，C の 5 種類がある．

NAS CG（シリアル版）のカーネル部分は以下のとおりであり，ほとんどの処理時間がこの部分で消費されるため，この部分の高速化が重要である．その部分にはリストベクトル colidx[k] による間接参照がある．

```
for (j = 1; j <= lastrow-firstrow+1; j++) {
   sum = 0.0;
   for (k = rowstr[j]; k < rowstr[j+1]; k++) {
      sum = sum + a[k]*p[colidx[k]];
   }
   w[j] = sum;
}
```

上記の内側 for ループでは，j 行に含まれる非零要素数 i=rowstr[j+1]-rowstr[j] 回の繰返しがなされるので，これがベクトル処理を行う場合のベクトル長を与える．行列の次数と i の平均を各クラスに関して測定した結果と配列 p[ ] のサイズを**表 2** に示す．

CG では p と同程度の大きさの配列が 8 個，a（p の 73〜273 倍）の大きさの配列が 5 個必要となる．たとえばクラス S や W ではキャッシュに大半の配列要素が載るのでメモリバンド幅の問題は生じないが，クラス B では二次キャッシュからもあふれ出て多くが主記憶へのアクセスとなる．Pentium4 の二次キャッシュのラインサイズ 128 バイトの中には 8 バイトのデー

表 2 CG における行列サイズと行あたり非零要素数の平均
Table 2 Matrix size and average number of non-zero-elements per row in CG.

クラス	行列次数	非零要素数/行の平均	p[ ] のサイズ
S	1,400	55	11 KB
W	7,000	72	54 KB
A	14,000	132	109 KB
B	75,000	182	586 KB
C	150,000	240	1.2 MB

タがキャッシュラインの中に 16 個入るが，クラス C では非零要素が十分に疎らであるため，p[ ] の間接参照を行う際に 1 ラインの中の他の非零要素はほとんどの場合入っていないと考えられる．よって，ラインサイズ 128 バイトのキャッシュを内蔵する Pentium4 においては p[ ] の間接参照は 16 倍のメモリバンド幅を消費してしまい，性能はメモリバンド幅がボトルネックとなると予想される．

5.2 プログラムの改造方法

カーネルループは NAS CG ベンチマークの実行時間の大半を占める．そのため，このループを高速化できれば全体の性能が高速化する．基本的には VLI() で表現される提案ハードウェアによる間接ベクトルロード，すなわち，gather 操作により，RW0 の連続領域に圧縮されたデータ列を，ホスト上のベクトルレジスタシャドー領域にコピーすることで，ホストでのアクセスは連続化されることにより，メモリバス上のデータ転送の効率化やキャッシュラインの効率的利用による高速化がなされる．**図 4** および以下にプログラムの改造後の処理の流れを示す．

(1) colidx[ ] を主記憶から提案ハードウェア上のメモリ領域に VS() 命令を使用して colidx-DIMM[ ] にコピー．
(2) p[ ] を主記憶から提案ハードウェア上のメモリ領域に VS() 命令を使用して p-DIMM[ ] にコピー．
(3) 以下カーネルループで，colidx-DIMM[ ] を RW0 に VL() 命令を使用してロード．
(4) VLI() 命令を起動する前にホスト側の RW0 に対応するキャッシュラインを CLFLUSH 命令で消去．
(5) VLI() により RW1 をインデックスとして p-DIMM[ ] を RW0 にロードする．
(6) VLI() によるロードの完了をホストからポーリング後，RW0 をソフトウェアプリフェッチまたはベクトルレジスタシャドー配列に RW0[ ] を代入することで，ホスト側キャッシュにブロック転送でコピー．
(7) キャッシュに取り込んだデータを用いて a[ ] と内積を計算．その際，a[ ] は連続アクセスなので MPU のハードウェアプリフェッチが起動される．
(8) 上記と同様の処理を繰り返す（3 へループ）．
(9) p[ ] を更新し，繰り返す（2 へループ）．

なお，図 4 において下線を付した VS()，VL()，VLI() の中身は，実機上ではハードウェアが実行すべき部分なので，その機能を模擬（エミュレーション）する関数を記述したバージョンのプログラムでは，実行速度は遅くなるものの，改造が論理的に正しいことを同一の実行結果が得られることで確認することができる．

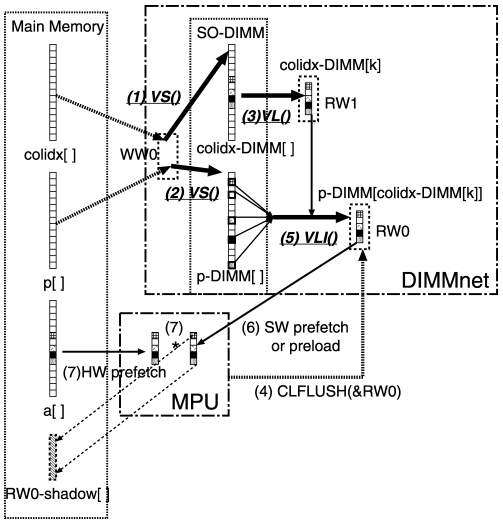

図 4 NAS CG における提案システム用プログラムの流れ
Fig. 4 Flow of NAS CG program modified for proposed system.

6. 性能評価

6.1 評価環境

性能評価は RWCP 版 OpenMP ディレクティブ付きの C 版 NPB CG[8] と，それをベースに前章で説明した提案メモリモジュール用の改造を行ったもので行った．Class C については RWCP 版のオリジナルのコードでは Linux 上ではメモリを確保できず実行できないので，一部の配列の確保を static ではなくプログラムの最初の部分で動的に確保することで OS の制限を回避して実行させている．**表 3** に性能評価を行ったマシンの仕様を示す．

6.2 メモリバンド幅と CG の性能の関係

表 4 に Pentium4 PC 上で CG（クラス C）実行に要求されるバンド幅と演算性能を示す．メモリバンド幅が性能を限定していることが予測されていたが，メモリバスを 1 本だけ動作する状態に主記憶を構成する DIMM の配置を変更して測定したものの上での CG の MFLOPS 値を示している．無対策の PC 上ではバンド幅から導かれる予測性能とかなり一致した実測性能が観測された．これは，無対策の PC 上では CPU の性能ではなくメモリバンド幅が CG の性能を決定していることが実験的に示されていることになる．

表 3 評価環境
Table 3 Evaluation environment.

機種名	Dell プレシジョン 360
CPU	Pentium4
FSB 周波数	800 MHz
コア周波数	2.4 GHz
L1 キャッシュ容量	8 KB
L2 キャッシュ容量	512 KB
L1 キャッシュラインサイズ	64 B
L2 キャッシュラインサイズ	128 B
メモリ種類	PC3200（DDR SDRAM）
メモリバス本数	1 および 2
メモリ容量	2 GB
OS	Linux 2.4.26
コンパイラ	gcc 3.2.2
最適化オプション	-O3

表 4 Pentium4 PC 上で CG（クラス C）実行に要求されるバンド幅と PC3200 1 本に対する演算性能
Table 4 Required bandwidth and MFLOPS per PC3200 based DIMM on Pentium4 PC for CG (class C).

		無対策の PC	提案システム
バンド幅	p[colidx[k]]	8/2×16=64 B/flop	8/2=4 B/flop
	a[k]	8/2=4 B/flop	8/2=4 B/flop
	colidx[k]	4/2=2 B/flop	0
	total	70 B/flop	8 B/flop
MFLOPS	PC3200×1（予測）	45.7 MFLOPS	400 MFLOPS
	PC3200×1（実測）	41.8 MFLOPS	378 MFLOPS（clflush 無）
			188 MFLOPS（clflush 有）

なお，提案システムでの机上予測性能は，バンド幅比率から導かれる上限値を示しており，ソフトウェアオーバヘッドやメモリシステムのハードウェア動作効率は反映されていない．それらを反映した評価については現状で得られている最善の測定結果のみ表 4 中に示し，詳細は後述する．

6.3 エミュレーション性能

現状では提案メモリモジュールは存在しないので，そのハードウェアが実行する予定になっている部分の動作は本評価ではソフトウェアによってエミュレーションしている．そのデータ構造および関数は前章で記載したとおりである．

本プログラムは NAS CG の処理を提案メモリモジュールを用いて実行できるように NAS CG プログラムそのものを書き換えたものであり，実機を使用する際にも残る記述と，実機を使用するときには削除されるべき記述の両方を含む．本プログラムは NAS CG の処理内容をまったく変えずに，同じ結果が得られるように等価変換がなされていることを実行結果から認識することができる．ただし，実行時間はハードウェアをエミュレーションしている時間も加算されるうえ，ハードウェアをエミュレーションするために追加された配列によってキャッシュの利用効率が悪化するなど，アルゴリズム的にもハードの実機がなければ遠回りしているだけにすぎず高速化する要素はなく，オリジナルの CG よりは確実に遅くなる．

表 5 NAS CG のオリジナルおよびエミュレーション性能（MFLOPS）
Table 5 MFLOPS for original NAS CG and emulation program on Pentium4 PC.

Class	S	W	A	B	C
original	167.61	166.54	167.71	90.22	55.57
emulation	94.31	106.91	111.66	54.62	38.20

表 5 にオリジナルの NAS CG ベンチマークおよび提案メモリモジュールを含むシステム上での動作をエミュレーションしているプログラムに関して，前述の評価環境において得られた実行時間および MFLOPS 値を示す．すべてのクラスについてオリジナルに対して 2 倍の実行時間で実行できた．実行結果はオリジナルと完全に一致しており，前述の改造がプログラムの意味を変えていないことが確認できた．

6.4 ハードによる SO-DIMM からの読み出しを理想化した場合の性能

提案システムを使用するために追加されるソフトウェアオーバヘッドを測定するために，ハードウェアで実現される機能をソフトウェアでエミュレーションしている箇所についてコメントアウトを行い，演算途中でけっして NaN（非数）割込みが発生しないように事前に RW などの配列を初期化したプログラムを用いて実行時間を測定した．この際，VLI() などの関数起動オーバヘッドは折り込まれるように，関数の中身をコメントアウトしている．

このバージョンのプログラムでは，VLI() などのハードで更新されるべき値がまったく更新されないために，CG 法の繰返しがなされてもまったく収束が進まないなど，実行結果の妥当性チェックでは失敗と表示されるが，NAS CG ではクラスごとに所定の繰返し回数で終了してくれるので，実行時間についてはほぼ実機を用いた場合を忠実に再現している．ただし，実機よりも若干多くの配列をホストの主記憶上に作って実行しているので，キャッシュの効率は若干今回の評価の方が実機よりも悪くなる．

ハードによる SO-DIMM から Read window までの読み出し時間が 1 回のポーリング時間以内で実行できる場合の提案メモリモジュールによる NAS CG 実行速度確認プログラムによる実行結果を**図 5** に示

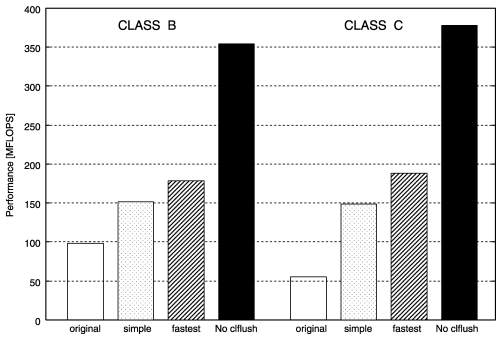

図 5 NAS CG における提案システムの効果
Fig. 5 Effect on MFLOPS of NAS CG with proposed system.

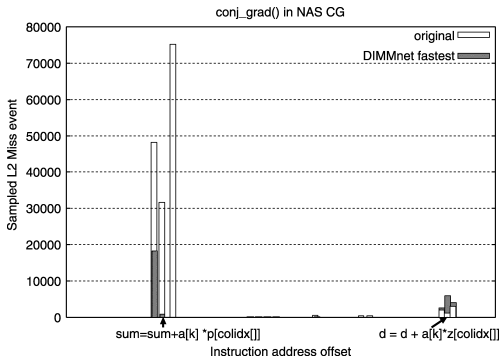

図 6 キャッシュミス発生の抑制効果
Fig. 6 Effect on reduction of cache-miss-hit.

す．各クラスに対して，オリジナル（original），単純に提案システムを用いた場合（simple），Read window を 2 本用いたうえでループアンローリングを行い PREFETCHNTA 命令や CLFLUSH 命令実行時期の調整を行った場合（fastest），CLFLUSH 命令の影響を完全に取り去った場合（No clflush）の 4 種類の測定結果が示されている．このグラフより明らかなように Original は Class B から C になると性能が半減してしまうのに対して，提案システムを用いる場合はむしろ性能が向上する．このため，Class C のように計算時間が長く高速化が強く望まれる処理において，CLFLUSH 命令を使用する場合においては 3 倍の高速化を達成することができることが分かった．さらに Pentium4 の CLFLUSH 命令自体の改善や，その影響の回避方法が発見されれば 388 MFLOPS 程度まで高速化がされることが分かった．これは Single チャネルの PC 上で提案システムを 1 ボード用いた場合の性能上限 400 MFLOPS に近いため，残された課題は CLFLUSH 命令の影響排除が主体であることが分かる．

図 5 において Original のプログラムの測定値は Dual チャネルを用いた場合の結果である．Original のプログラムの ClassC において，表 4 に示されているように Single ではバンド幅が有効に働き，Dual ではそれほどではない．ハードディスクへのアクセスがほとんどないことを確認しているものの，この測定条件においてはバンド幅に比例しないその他の要因（TLB ミスなど）の影響も大きいことが示唆されている結果であると考えられる．

6.5 キャッシュミス発生の抑制効果

CG におけるキャッシュミス発生状況を Pentium4 のパフォーマンスカウンタを用いた測定ツールである Brink および Abyss [9] の組合せによって測定した．**図 6** に RWCP 版 CG（original）と前節の fastest の Class B の conj_grad() 関数における結果を示す．横軸は命令コードのオフセット値であり，縦軸は L2 キャッシュミスのサンプリング回数である．両方ともプログラム実行中の L2 キャッシュミスヒットの大半はカーネル部分の間接参照 p[colidx[k]] によって発生している．もう 1 つ 1/20 ほど小さな山が後半に存在するが，こちらも間接参照 z[colidx[k]] によって発生していることが分かる．キャッシュミス数の回数比から考えて，カーネル部分のみ提案ハードウェアを用いた高速化を施すことでほぼ十分と考えられ，本論文における評価はすべてカーネル部のみの改造を行う際の評価である．また，Original に比べて，提案ハードウェアを用いた場合の Fastest のキャッシュミスヒット回数が大幅に減少していることが分かる．これは提案ハードウェアがキャッシュライン浪費を抑制する効果と，改造したプログラムが PREFETCHNTA 命令をおおむね適切なタイミングで挿入できていることによると考えられる．

6.5.1 Pentium4 のハードウェアプリフェッチの効果

今回実験に用いた Pentium4 にはハードウェアプリフェッチ機能がついており，その影響が懸念されたため，ハードウェアプリフェッチ機能を有効にした場合と無効にした場合の実行時間を測定した．その結果を**図 7** に示す．この結果，ハードウェアプリフェッチ機能はオリジナルのプログラムではほとんど影響がなく，提案システムを用いる場合には Class B および C の両方で若干メリットがあることが分かった．

6.6 Read window の語数を変動させた場合の性能

1 本の配列の間接参照のために Read window を 1 本のみ用いた場合，Read window の語数を変動させた場合の性能変化を測定した．その結果を**図 8** に示

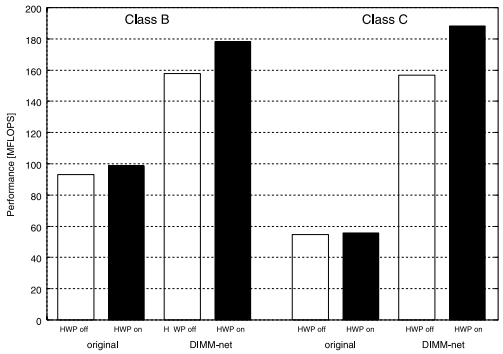

図 7 ハードウェアプリフェッチの演算性能への効果
Fig. 7 Effect on MFLOPS with hardware-prefetching.

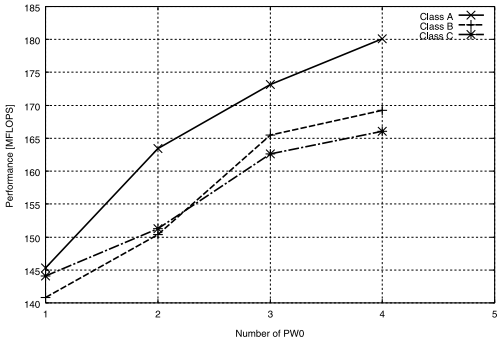

図 9 Read window の個数を変動させた場合の演算性能
Fig. 9 Effect on MFLOPS with the number of Read window.

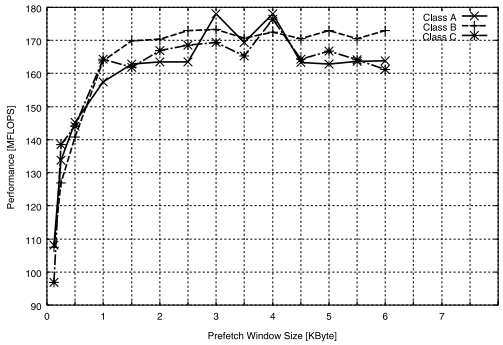

図 8 Read window の語数を変動させた場合の演算性能
Fig. 8 Effect on MFLOPS with the word-number of Read window.

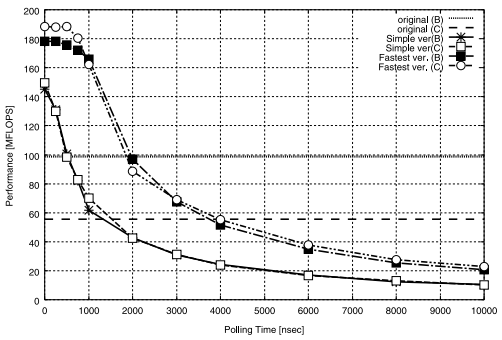

図 10 プリフェッチにかかる遅延を変動させた場合の性能
Fig. 10 Effect on MFLOPS with the latency of prefetching.

す．その結果 1 KB 程度あればすべての Class の CG に対しては十分な性能が得られることが分かった．他のアプリケーションについても調べる必要があるが，少なくともこの結果のみからいえば，CG に対しては DIMMnet-2 プロトタイプの当初の設計値である 512 B よりも，ハードウェア資源が足りるならば 1 KB 以上に増やした方が若干性能が向上するといえる．本評価では SO-DIMM から Read window までのプリフェッチにかかる時間が 1 回のポーリング時間以内で終わるという状況下での評価になっているが，実際には Read window の語数をあまり大きくするとこの仮定が非現実的なものとなり実機と本評価との間に誤差が増大する．この点を考慮すると必ずしも語数が多ければよいというものでもないと考える．

6.7 Read window の個数を変動させた場合の性能

1 本の配列の間接参照のために Read window を複数用いた場合，Read window の語数を 512 B に固定し，本数を変動させた場合の性能変化を測定した．その結果を**図 9** に示す．その結果，1〜4 本までの範囲では本数が多いほど性能が向上することが分かった．つまり，同じ 2 KB の資源を使うならば，2 KB を 1 本の構成で使うよりは 512 B を 4 本の構成で使った方が，性能向上が得られるといえる．このことは SO-DIMM から Read window までのプリフェッチにかかる時間の隠ぺいの観点からも小さめの Read window を複数設けて，大きな語数が効果があるような場面ではその使用本数を増やすことで同等以上の効果をあげるという方針が良いと考える．

6.8 プリフェッチにかかる遅延を変動させた場合の性能

SO-DIMM から Read window までのプリフェッチにかかる時間は DIMMnet-2 プロトタイプの間接参照ベクトルアクセスに関する論理設計が完了していない現時点では明確ではないため，この時間が 1 回のポーリング時間を超える範囲で変化させたときの性能の変化を測定した．遅延時間の生成には Pentium4 の性能カウンタを用いて正確な遅延時間を生成してフラグポーリングの部分に挿入した．その結果を**図 10** に示す．この測定は 512 B の Read window が 1 本の場合

（simple）と 2 本の場合（fastest）で明瞭に性能グラフが分離する．1 本の場合は 500 ns〜1 μs 程度で性能向上が得られなくなるほど遅延増加にともなう性能低下が激しくなるが，2 本の場合は 1 μs 程度の遅延を十分に隠ぺいできることが分かる．この結果より，Read window を 512 B を 2 本用いるとした場合は 1 μs 以下で SO-DIMM から Read window までのプリフェッチを完了させることを目標に論理回路の設計を進めるべきであるという指針が得られる．

間接ベクトルロードの DIMMnet-2 プロトタイプでの実装は未完了であるため正確な遅延は述べられないが，間接ベクトルロードと多くの部分を共通に用いる等間隔ベクトルロードの実装が，最適化できた状態ではないものの動作が可能な状態まで DIMMnet-2 プロトタイプ上で実装されており，512 バイトの Read window1 本を double 型データで満たす繰返し数 64 では本来の半分の周波数（100 MHz）で動作した際の遅延が 830 ns である．ただしこの測定結果は 32 bit データを不連続アクセスした場合なので，64 bit データを扱う CG の場合よりも SO-DIMM アクセスなどの効率が悪い状況での遅延である．ASIC 化がなされて本来の周波数（200 MHz）で動作すればさらに半分（400 ns）程度の遅延で間接ベクトルロードが行われるものと考えられる．これに先立つインデックス配列の連続ベクトルロードはデータ量が半分で，かつ転送効率も 2 倍程度見込めるので上記の 1/4 程度の遅延と考えられる．よって ASIC 化がなされるならば，500 ns 程度のハード遅延を期待できるため，Read window2 本で十分に高い効率での動作が期待できることを図 10 より読み取ることができる．

7. 従来研究

通常のキャッシュアーキテクチャベースの MPU のメモリアクセス部分の改良として，SCIMA [10] や，SDT および TPDT [11] が提案されている．しかしこれらは MPU そのものを改造する必要があるので，Intel や AMD などの MPU ベンダが採用しない限り PC クラスタに利用できる COTS とはならない．さらに SDT や TPDT では書き込みには対応していない．SCIMA では等間隔アクセスのみ命令でサポートし，間接参照には対応していない．これに対し本方式は間接参照や書き込みにも対応しており，MPU やマザーボードに手を加える必要がないので，COTS のメリットを最大限享受できる．

メモリコントローラ（ノースブリッジ）の改良として Impulse [12] が提案されている．Impulse はエイリアス上の連続アクセスを等間隔ベクトルアクセスや間接ベクトルアクセスに変換する点で機能が類似している．しかし，ホスト CPU 上の仮想空間上に設けられる実体と同サイズのエイリアスへのアクセスとなるために，メモリコントローラ側でのアドレス変換が多段階でオーバヘッドが増加し，かつ 32 bit CPU ではその空間の狭さに縛られる．さらに，ホスト CPU の FSB（Front Side Bus）に接続されるためホスト側のメモリコントローラを変更しない限り実装できない．このため，COTS のマザーボードを利用することはできず，PC クラスタ用技術としては適用できない．これに対し本方式はホストの仮想空間とは独立した巨大なページサイズ（DIMMnet-2 では 4 GB）を有する一種の I/O 空間に対するアクセスであるため，自ノードの SO-DIMM へのアクセスについては実質的にアドレス変換そのものが排除されているためベクトル型メモリアクセス部と相まって高速化が可能である．さらにノースブリッジやマザーボードに手を加える必要がないので，COTS のメリットを最大限享受できる．

8. まとめ

本論文では PC のメモリスロットに装着されるプリフェッチ機能を有するメモリモジュールを提案した．このデバイスは，Pentium4 などの COTS 型 MPU のキャッシュアーキテクチャの弱点を軽減することで，パーソナルスーパコンピュータに匹敵する性能を PC 上でも実現可能にすることを目指している．

本論文ではさらに DIMM スロットに装着される NIC である DIMMnet-2 におけるベクトル転送命令による本方式の実装例について示した．本実装例では，プロテクションのためのオーバヘッドを減らしつつ，ローカルとリモートの差を極力排除した統一的な命令により高速アクセス可能な，最も COTS の恩恵を享受できる 32 bit CPU のアドレス空間の不足を超越した巨大な共有メモリ構築のための 1 つの方法論が示されている．このようにして形成される大域的仮想空間は，Partitioned global address space と同様のセマンティックスをハードウェアが直接実現しており，それを基盤とする UPC や Co-array FORTRAN の効率的な実装が期待できる．

さらに，NAS CG ベンチマークによる間接参照の性能評価を行い，本方式により Pentium4 などの COTS 型 MPU を単純に用いる場合と比べ，メモリバンド幅の有効利用がはかられ，大幅な高速化が得られることや，CLFLUSH 命令の影響を排除できればより高速化することや，いくつかのハード的パラメータを振っ

た測定によるハード設計の最適化に関するいくつかの重要な指針を示した．本評価結果から，類似したデータ構造へのアクセスがネックとなっている有限要素法における効果も有望であると考えられる．

本評価手法は Simple Scalar Tool Set などの命令レベルシミュレータを使う従来の多くの研究と異なり，内部詳細構造が非公開な Pentium4 を対象として精密に結果に反映できている点，NAS CG ベンチマークの Class B や C といった実行時間の観点から命令レベルシミュレータでは現実的には取り扱えない対象における実行時間を，高速かつ精密に測定できている点で画期的な手法である．

本方式は NIC 上に必ずしも実装される必要はなく，1CPU のシステムにおいて有効である．つまり，並列処理とは別の原理により，本方式を内在する DIMMnet-2 は装着した安価なノード PC の実行性能を高速化する可能性を持っているという点で，従来にない画期的な特質を有する NIC であることが示された．これが，通信のみの高速化を実現する DIMMnet-1 との根本的な違いである．

現時点で DDR スロット上で FPGA がメモリとしてホスト PC から認識されてアクセスできることは確認済みで，電気的には現在の方向性での実現可能性は確認できている．今後は，DIMMnet-2 上の外部メモリの構成や，その他の部分の設計を進め，本方式の有効性をシミュレーションやハードウェアプロトタイプ作成により評価を行う予定である．主記憶データベースにおける Wisconsin ベンチマークによる等間隔アクセスの評価を実施中である．さらに専用コンパイラについても開発中である．DIMMnet-2 を 2 枚装着するなどにより Dual チャネルを有するマザーボードへの対応は可能と考えているが，その詳細の検討については今後の課題である．

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．間接参照評価用に有限要素法プログラムをご提供いただきました理化学研究所の姫野情報基盤センター長，黒川氏に感謝いたします．DIMMnet-2 の開発に関する議論にご参加いただいている慶應義塾大学の西講師，渡辺氏，大塚氏，伊豆氏，北村氏，伊沢氏，宮代氏，宮部氏，東京農工大学の並木助教授，浜田氏，荒木氏，木立氏，森氏，羅氏，立命館大学の国枝教授，和歌山大学の齋藤講師，日立 IT 社の上嶋氏，今城氏，岩田氏に感謝いたします．

参 考 文 献

1) 萩原，梅澤：パーソナルスーパーコンピュータ SX-6i，情報処理，Vol.44, No.3, pp.277–282 (2003).
2) Intel Corp.：IA-32 インテルアーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル，下巻：システム・プログラミング・ガイド，資料番号 253668-013J, B14 (2004).
3) 田邊，濱田，中條，天野：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，情報処理学会計算機アーキテクチャ研究会，2003-ARC-152, pp.61–66 (2003).
4) Myricom, Inc. http://www.myri.com/
5) Beecroft, J., Addison, D., Petrini, F. and McLaren, M.: Quadrics whitepaper, QsNETII: An Interconnect for Supercomputing Applications (Nov. 2003). http://www.quadrics.com/
6) 田邊，山本，濱田，中條，工藤，天野：DIMM スロット搭載型ネットワークインタフェース DIMMnet-1 とその高バンド幅通信機構 BOTF，情報処理学会論文誌，Vol.43, No.4, pp.866–878 (2002).
7) 田邊：超並列テラフロップスマシン TS/1 における並列処理—プロセッサ間チェイニングとその応用，情報処理学会論文誌，Vol.36, No.3, pp.658–668 (1995).
8) RWCP: NAS Parallel Benchmarks in OpenMP. http://phase.hpcc.jp/Omni/benchmarks/NPB/
9) Sprunt, B.: Brink and Abyss: Pentium 4 Performance Counter Tools For Linux (Sep. 2002). http://www.eg.bucknell.edu/~bsprunt/emon/brink_abyss/brink_abyss.shtm
10) 中村，近藤，大河原，朴：ハイパフォーマンスコンピューティング向けアーキテクチャSCIMA，情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム，Vol.41 No.SIG5, pp.150–27 (2000).
11) 府川，田中，宮崎：主記憶データベース向け高機能メモリコントローラの性能評価，情報処理学会計算機アーキテクチャ研究会，2003-ARC-152, pp.85–90 (2003).
12) Carter, Hsieh, Stoller, Swanson, Zhang, Brunvand, Davis, Kuo, Kuramkote, Parker, Schaelicke and Tateyama: Impulse: Building a Smarter Memory Controller, *International Symposium on High Performance Computer Architecture* (*HPCA-5*), pp.70–79 (1999).

(平成 17 年 1 月 24 日受付)
(平成 17 年 5 月 6 日採録)

田邊　昇（正会員）

1985 年横浜国立大学工学部卒業．1987 年横浜国立大学大学院工学研究科修了．同年（株）東芝に入社．1998 年より 2001 年まで新情報処理開発機構つくば研究センターに出向．計算機アーキテクチャ，並列処理，PC クラスタ向けネットワークインタフェース，高機能メモリモジュールに関する研究に従事．現在，(株) 東芝・研究開発センター勤務．工学博士．電子情報通信学会会員．

安藤　宏

2005 年横浜国立大学工学部卒業．現在，横浜国立大学大学院環境情報学府在学中．

箱崎　博孝

2003 年横浜国立大学工学部卒業．2005 年横浜国立大学大学院物理情報工学府修了．同年日本電気株式会社に入社．現在，システムソフトウェア事業本部グリッド推進センター勤務．

土肥　康孝（正会員）

1967 年東京工業大学大学院博士課程修了，工学博士．1967 年 4 月横浜国立大学工学部講師，助教授，教授を経て，計算機アーキテクチャ，並列処理等の研究に従事．2005 年 3 月停年退職．現在，横浜国立大学名誉教授．

中條　拓伯（正会員）

1961 年生．1985 年神戸大学工学部電気工学科卒業．1987 年神戸大学大学院工学研究科修了．1989 年神戸大学工学部助手の後，1998 年より 1 年間 Illinois 大学 Urbana-Champaign 校 Center for Supercomputing Research and Development（CSRD）にて Visiting Research Assistant Professor を経て，現在東京農工大学大学院共生科学技術研究部助教授．プロセッサアーキテクチャ，並列処理，クラスタコンピューティング，高速ネットワークインタフェースに関する研究に従事．電子情報通信学会，IEEE CS 各会員．博士（工学）．

天野　英晴（正会員）

昭和 56 年慶應義塾大学理工学部電気工学科卒業．昭和 61 年慶應義塾大学大学院理工学研究科電気工学専攻博士課程修了．現在，慶應義塾大学理工学部情報工学科教授．工学博士．計算機アーキテクチャの研究に従事．