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

2005－ARC－162 (24)
2005－HPC－101 (24)
2005／3／8

メモリモジュール上での等間隔アクセス連続化の効果

田邊　昇[†1]　箱崎博孝[†2]　安藤　宏[†2]
土肥康孝[†2]　中條拓伯[†3]　宮代具隆[†4]
北村　聡[†4]　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，Pentium®4 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．主記憶データベースに対する Wisconsin ベンチマークへの適用を通し，等間隔参照の高速化に関する評価結果を示す．その結果，タプルサイズがキャッシュラインサイズより大きい場合は、最大で 25.2 倍の加速率を並列処理なしで得られることが判った．

Effects of Conversion of Strided Access into Continuous Access on a Memory Module

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

Prefetching functions in a memory module plugged into a memory slot of a PC are proposed. The proposed device overcome the defect of cache architecture of a COTS (Commercial Off-The-Shelf) type MPU such as Pentium®4 in order to realize personal supercomputer class performance. In this paper, the concept and its software corresponding method of a memory module with prefetching functions and its implementation plan on a DIMMnet-2 prototype are presented. In addition, evaluations for improvement of a stridden access with Wisconsin benchmark programs are shown. As a result, at most 25.2 times performance improvement without parallel computation is gained, when tupple size is bigger than cache line size.

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たマイクロプロセッサ (MPU) の目覚しい進歩を遂げている．例えば Pentium®4 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 1 CPU 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．

一方，DB2® の父と称される IBM® フェローの Linsay 氏は，コンピュータ上の主記憶が増加することに伴って DBMS の重要な部分が主記憶に常駐するようになり，キャッシュミスの最小化の重要性が高まることを指摘[2] しており，キャッシュが効かないアプリケーションは科学技術計算にとどまるものではない．この点は将来，MRAM のような不揮発性大容量高速メモリが DRAM に置き換わるような状況になった場合，この分野におけるキャッシュミスの最小化の重要性は加速する．

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る．例えば Pentium®3 では L2 キャッシュのラインサイズが 32 バイトであるのに対し，Pentium®4 では 128 バイトである．キャッシュベースの MPU は常に最下位階層キャッシュライン単位で主記憶をアクセスする．

ところが，例えばデータを主記憶に配置した際のリレーショナルデータベース処理のように，ストライドの大きい不連続アクセスが必要なアプリケーションでは，真に必要なデータが 128 バイトのキャッシュラインの中に 1 バイトしか存在しないというケースもありうる．

このような状況が，間接指標配列を用いた圧縮型の配列や多次元配列を用いる科学技術計算のみならず，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ーで見受けられる．

ベクトル型スーパコンピュータはデータベース処理には使われてこなかったが，データ全体が乗ってしまうような大容量な主記憶が安価に提供できるのであれば，テーブルスキャンの高速化には向いている．

†1 (株) 東芝, 研究開発センター
Corporate Research and Developpment Center, Toshiba
†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

しかし，ベクトル型スーパコンピュータは高価である点，製品更新サイクルが長い点，ユーザーが PC クラスタ等に移行している点などから，今後も市場の拡大や低価格化は望めず，提供するベンダーも減少の一途にある．その入手困難性の増加ともあいまって，その低コストな解決の工学的価値は非常に高いと考えられる．

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してきた．このデバイスは，Pentium®4 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，パーソナルスーパー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．

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無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] 上で行われており，既に FPGA 版の DIMMnet-2 プロトタイプがメモリとしてホストからアクセス可能であることは確認できている．

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ーに対しても，PC の単体性能を向上でき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®[5] や QsNET[6] 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ー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等間隔アクセスを多用する応用である Wisconsin ベンチマークを紹介し、その本報告における評価の対象としての位置づけを示す．6 章では提案システム用に改造された Wisconsin 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．

例えば 128 バイトを超える構造体の配列に対して，整数型のあるメンバを全てのインデックスに対してアクセスするようなストライド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 Pentium®4 で実行する場合，真に必要なデータが 128 バイトのキャッシュラインの中に 4 バイトしか存在しない．キャッシュを用いた CPU を使って等間隔参照を行う際の問題点を**図 1** に示し，以下に列挙する．

(1) 有効データが少ないゆえのメモリバンド幅の浪費
(2) 有効データが少ないゆえのキャッシュエントリの浪費
(3) 有効データが少ないゆえの TLB エントリの浪費

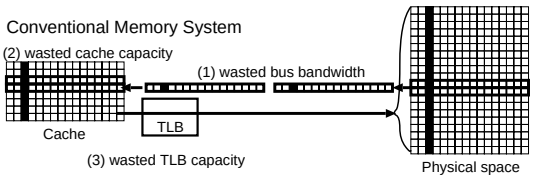

図 1 キャッシュを用いた CPU での等間隔参照の問題点

キャッシュアーキテクチャの上記のような問題点は，例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ー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伴い，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上，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ー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伴い，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，例えば 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. プリフェッチ機能付メモリモジュール

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等間隔ロード命令で，配列の等間隔ロード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．ホストはこれらの window のデータ格納状態をフラグを通じて確認できる．例えばホスト MPU の L2 キャッシュライン分のデータ毎にフラグが 1bit 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等間隔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 (2)〜(4) の問題点が全て解決される．

3.2 ソフトウェア的対応

本機構は，キャッシュメモリ等とは異なり，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 Pentium®4 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．

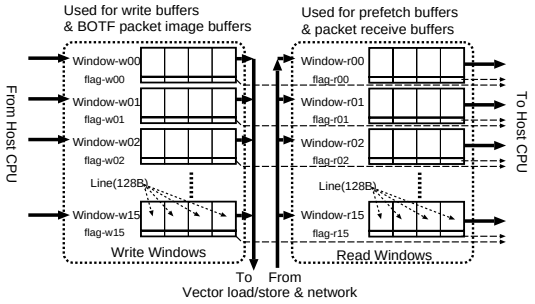

図 2 プリフェッチ機能付き Window メモリ

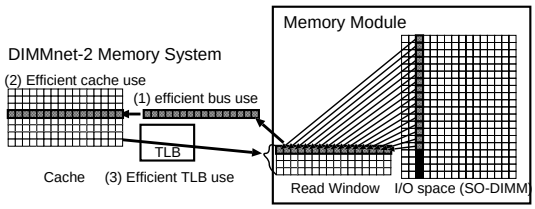

図 3 提案システムにおける等間隔配列参照時のキャッシュ挙動

(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頃に実行する．Pentium®4 の場合は時間的局所性を持たないデータ用のプリフェッチ命令である PREFETCHNTA 命令を適切なタイミングで実行しておくと良い．
(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 (Pentium®4 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 [7] 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 プロトタイプ上での実装

提案されたアーキテクチャの検証用として DIMMnet-2 プロトタイプ上に実装して実機検証を行なうことにしている。本報告では主に等間隔アクセスに関して評価を行なう。

4.1 資源モデル

ローカルな SO-DIMM 上にある配列への等間隔ベクトルロードやベクトル間接ロードを実行するのに必要な，DIMMnet-2 プロトタイプ上に実装されているプログラマから見える資源は以下のものがある。
(1) SO-DIMM 領域 (4GB×1〜16)
ベクトルアクセスされる対象の配列データを格納する。仕様上は最大 4GB のページを 16 枚まで持てるが，このサイズは搭載している SO-DIMM の容量や枚数に依存する．
(2) RW : Read Window(64bit 幅 64 語× N 本)
等間隔ベクトルロードコマンドを用いて (1) から必要なデータをロードする。ホストからはユーザー空間にマップされており、キャッシュ可能なメモリ領域として見えている。
(3) RWSR : RW Status Register(64bit 幅 1 語)
(2) 上へのデータのロードのライン単位での完了状況が反映される．ホストからはユーザー空間にマップされており、キャッシュ不可能なメモリ領域として見えている．
(4) CMR : Command Register(64bit 幅 16 語)
ベクトルコマンドをホストから書き込むことで等間隔ベクトルロードコマンドなどが起動される．ホストからはユーザー空間にマップされており、キャッシュ不可能なメモリ領域として見えている．
(5) CMSR : Command Status Register(16bit 幅 1 語)
(4) の完了状況が反映される．ホストからはユーザー空間にマップされており、キャッシュ不可能なメモリ領域として見えている．
(6) CPR : Current Page Register(16bit 幅 4 語)
(1) のどのページに対するベクトルコマンドなのかを指定する．ホストからはユーザー空間にマップされており、キャッシュ不可能なメモリ領域として見えている．

4.2 ベクトル転送命令

ローカルな SO-DIMM 上にある配列への等間隔ベクトルロードやベクトル間接ロードを実行するのに必要な DIMMnet-2 プロトタイプ上に実装されたコマンドは以下のものがある。コマンドは 64bit 形式であり，ホストから CMR に書き込まれる．

以下，XW は WW と RW の総称，type はデータ型，iter は繰り返し回数，displace は XW 上での開始位置，top は SO-DIMM 領域を集めて構成される大域仮想空間上の 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) ベクトル等間隔ロード:
VLS(type,iter,displace,RWi,top,stride)
(2) ベクトル間接ロード:
VLI(type,iter,displace,RWi,top,indexXWj)

4.3 等間隔ベクトルロード時間

現在，DIMMnet-2 プロトタイプは FPGA 内部の論理設計途上にあり，一部の機能しか利用できず，使える場合でも性能チューニングが十分な状態ではない．本報告で評価するベクトル等間隔ロードコマンドについては性能チューニングが十分な状態ではないながら，動作が確認できており，Verilog レベルでの現状の所要サイクル数が判っている．現状の回路における等間隔ベクトルロードコマンドのデータ型 type が 4 バイト，要素間間隔 stride が 60 バイトという条件における所要時間を**表 1** に示す。ここでは FPGA 版を 100MHz 動作，ASIC 版を 200MHz 動作するものとしている．

表 1 等間隔ベクトルロードコマンドの所要時間

繰り返し数 iter	所要サイクル数	FPGA	ASIC
64	83	830ns	415ns
128	162	1620ns	810ns

5. 評価対象とするアプリケーション

本報告では提案システムの等間隔アクセス主体のアプリケーションの高速化の評価を目的として，主記憶上にデータを格納するデータベースを対象に検討する．半導体メモリの大容量化とビット単価の減少および検索処理への高速化のニーズを背景に，主記憶上にデータを格納する主記憶データベースへの期待感が高まってきており，本報告ではこれを扱う．

各属性データの格納形式はホスト CPU の基本アクセス単位である大きさ (Pentium®4 の場合は 32bitCPU なので 32bit) で格納し，それを超える大きさのオブジェクト (文字

列等) についてはポインタとして構成されるタプルの配列で格納されるものとする。4 バイトを超えるオブジェクトの内容で検索する必要がある場合のみ，上記のタプルに記載されたポインタをたどり，別途格納されたオブジェクトをアクセスするものとする．すなわち，基本的な検索ではデータのミスアラインの影響という問題はない．

データベースの性能評価には Wisconsin ベンチマークが用いられることがあり，文献 [9] でも主記憶データベース処理の SDT による高速化の評価を目的に Wisconsin ベンチマークが用いられている．本報告での評価でもこれをベースにした評価を行なう．文献 [9] の記述では本方式がデータベース処理には向かないということになっているが，それが正しいかどうかを実験的に証明するために，できるだけ文献 [9] に記載の評価対象に近いもので評価を行なうことを目指す．

Wisconsin ベンチマークでは 15 の属性からなるタプルが 10K 個または 1K 個から構成されるデータベース (10K 個の tenk，1K 個の onek) が検索対象になる．1 個のタプルは上記の主記憶に配置される形式で格納すると 15 個の属性は 4 バイトのデータまたは 4 バイトのポインタ (合計 60 バイト) からなる．そのうちの 2 つのポインタで指し示される 2 つの文字列データからなる．Wisconsin ベンチマークにおけるクエリー (問合せ) では上記の 60 バイトの部分が検索に用いられるので，本報告ではこの部分全体が SO-DIMM で構成される領域上に乗るようにする．

このように表形式のデータベースではある属性に対する検索処理を行なう際には等間隔アクセスとなってしまう．上記の Wisconsin ベンチマークでは，ストライド値 60 バイトで不連続に格納されている 4 バイトデータに対する等間隔アクセスとなる．

例えば Pentium®4 の L2 キャッシュラインは 128 バイトなので 1 ラインに 2 個強の 4 バイトデータしか有効ではない．もし，検索対象の総データサイズがキャッシュ容量よりはるかに大きければ，このキャッシュラインリフィルに対する時間が実行時間の大半を占めることが予想される．この場合，データベースの先頭から順にスキャンするとしても，メモリバス上では 4 バイトのデータを取得するために平均 60 バイトを読み出すこととなり，必要なメモリバンド幅は 15 倍にも膨れ上がる．

文献 [9] 上での評価ではキャッシュ容量を 8KB として上記のベンチマークそのものをシミュレータ上で実行させている．しかし，近年の MPU は内蔵するキャッシュの容量が増加してきており，例えば Pentium®4 では L1 および L2 キャッシュの合計が 520KB にもなるため、本来の Wisconsin ベンチマークで規定されたタプル数 (10K 個) では大半がキャッシュに乗ってしまう．それでは主記憶データベースの評価としては不適切であるので，タプル数をキャッシュの増加分である 65 倍に増やして評価を行なうものとする．

6. 性能評価

6.1 評価環境

表 2 に性能評価を行ったマシンの仕様を示す．この環境は単にシミュレーションを行った機械という意味合いだけでなく，後述する評価方法を用いているために評価対象そのものを構成する．つまり，キャッシュラインサイズやキャッシュ容量はもちろんのこと，メモリバンド幅や命令の CPU 内部での実装の細部に至るまで評価対象そのものを構成し，提案メモリモジュールと組み合わせた際の性能を左右する．これらを仮想的に変更することができない代わりに，通常のシミュレータを用いた評価手法に比べて高速性と精度が高い．

6.2 評価方法

性能評価は Wisconsin ベンチマークにおけるデータの挿入、更新、削除以外の問合せ (クエリー) を利用して行なう．基本的には文献 [9] と同じクエリーを用いる．ただし Pentium®4 を用いた場合に On キャッシュデータベースではなく主記憶データベースの評価とするために，タプル数は文献 [9] における評価対象のキャッシュサイズと Pentium®4 のキャッシュサイズの比率である 65 倍に増やし，520KB ものオンチップヤッシュにデータの大半が収まってしまわないようにして評価を行なう．

表 2 評価環境

機種名	Dell^{TM} Precision^{TM}360
CPU	Pentium®4
FSB 周波数	800MHz
コア周波数	2.4GHz
L1 キャッシュ容量	8KB
L2 キャッシュ容量	512KB
L1 キャッシュラインサイズ	64B
L2 キャッシュラインサイズ	128B
メモリ種類	PC3200 (DDR SDRAM)
メモリバス本数	2
メモリ容量	2GB
OS	Linux 2.4.20-8
コンパイラ	gcc 3.2.2
最適化オプション	-O3

まず Wisconsin ベンチマークで規定された表形式のデータを C 言語で記述したプログラムで主記憶上に作成する．以下のクエリーを C 言語で記述したものを Original とし、この実行時間を 1 とした際の加速率を測定する．本来の Wisconsin ベンチマークでは tenk は 10K 個のタプルからなるが，本評価では 650K 個のタプルからなる．クエリーの番号は文献 [9] と合わせている．

(Q1) select * from tenk1 where (unique2 > 301) and (uniqu2 < 402)
(Q2) select * from tenk1 where (unique1 > 647) and (unique1 < 65648)
(Q3) select * from tenk1 where unique2 = 2001
(Q4) select * from tenk1 t1,tenk1 t2 where (t1.unique2 = t2.unique2) and (t2.unique2 < 1000)
(Q6) select t1.*,o.* from onek o,tenk1 t1,tenk1 t2 where (o.unique2 = t1.unique2) and (t1.unique2 = t2.unique2) and (t1.unique2 ¡ 1000) and (t2.unique2 ¡ 1000)
(Q7) select MIN(unique2) from tenk1

この Original プログラムをベースに，提案メモリモジュール用の改造を行ったもので行った．ハードウェアで実行する部分を記述したエミュレーション版を作成し，その実行結果の一致による改造の妥当性をチェックする．

その後、ハード部の記述をしている関数の中身をコメントアウトすることで，ハード部の遅延がゼロになった状態の実行時間を再現して，ベクトルコマンド起動やフラグチェックなどのソフトウェアオーバーヘッドを含んだ性能を再現する．その際，本来ハードによって設定される Read Window 上のデータが更新されないため実行結果自体は Original とは異なったものになるが，その選択率や実行時間が狂わないように初期値を設定した Read Window 上のダミーデータを用いて処理をすることで実行時間を再現する．

その上で，フラグのポーリング部の遅延時間としてホスト側に見えるハード部の遅延時間を Pentium®4 に内蔵されるパフォーマンスカウンタを用いて挿入し，その変動により全体の実行時間がどのように変化するかを観測する．

さらに，実際の現場で用いられているデータベースのようにもっと大きなタプルサイズを有するテーブルに対する性能を評価するために，140 バイトのタプルサイズを有するテーブルを作成し，同様の評価を行なう．

6.3 実験結果

6.3.1 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

エミュレーション版のプログラムのハード部の記述をしている関数の中身をコメントアウトした際に観測される実行時間が，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に対応する．その測定結果を**図 4** に示す．各クエリーの Original での実行時間との相対値である加速率を示している．ここで，Read Window のサイズは 512 バイト (4 バイトデータ 128 個分) に固定し，Simple は Read Window を単純に 1 枚用いた場合の加速率，Simple unroll は Simple にループアンローリングを施した場合の加速率，Pipelining は Read Window を 2 枚用いてソフトウェアパイプライニングを行なった場合の加速率，Pipelining unroll は Pipelining にループアンローリングを施した場合の加速率，NOCLFLUSH はキャッシュラインをフラッシュする CLFLUSH 命令の影響を排除できた場合の加速率である．

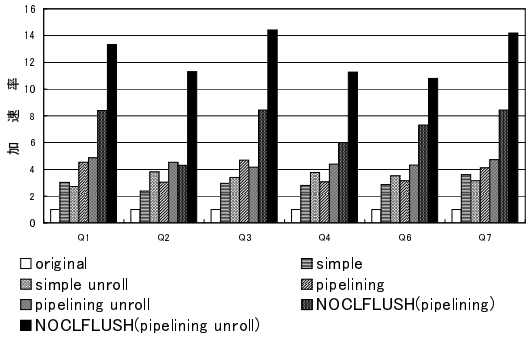

図 4 Wisconsin ベンチマークにおける提案システムの効果

この結果 Pipelining unroll の利用形態において 4.16 倍〜4.87 倍の加速率が得られることがわかった．この結果は逐次性能向上における加速率としては高い加速率ではあるが，NOCLFLUSH(Pipelining unroll) が 11.27 倍〜14.42 倍という高い加速率を示しているように，提案メモリモジュールの潜在能力的にはもっと高い加速率を期待できる．Wisconsin ベンチマークの場合は 4 バイトの有効なデータを取得するのに 60 バイトという 15 倍のメモリバンド幅を浪費することから，15 倍が提案メモリモジュールの加速率の限界であるが，今回の結果は 15 倍に迫る加速率が NOCLFLUSH(Pipelining unroll) で得られることは理に適っている．

このように今回の評価で加速率を制限した主な原因は CLFLUSH 命令の影響と考えられる．文献 [4] における評価でも測定に用いた Pentium®4 の CLFLUSH 命令の実行時間やその副作用はかなり大きく，実効性能を半減させてしまうことが判っており，今回の実験結果でも同様の傾向が観測された．CLFLUSH 命令がこのような影響を与えることは，Intel® 社内以外では詳細不明なブラックボックス状態にある Pentium®4 を Intel® 外部の人間がシミュレーションによって発見することは事実上困難であるが，本評価手法はこの問題点の発見を可能とした．

なお，SDT[9] ではほとんど加速できなかったマルチウェイの結合演算 (クエリー Q6) も，提案メモリモジュールでは高速化が達成されている．その理由は間接ベクトルロードが高速化されるためである．

6.3.2 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 までの等間隔ベクトルロード実行にかかる時間を，1 回のポーリング時間を越える範囲で変化させた時の性能の変化を測定した．遅延時間の生成には Pentium®4 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 5** に示す．本測定における対象のクエリーは Q7 である．

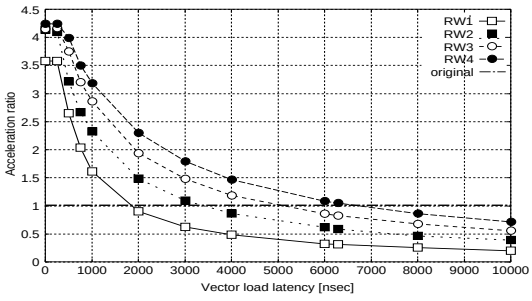

図 5 プリフェッチにかかる遅延を変動させた場合の Q7 の処理時間

この測定は 512B の Read window が 1 本の場合 (Simple unroll) と 2〜4 本の場合 (Pipelining unroll) で上限性能が分離する．本数は多いほど等間隔ベクトルロードコマンド実行にかかる時間に対する耐性が強くなる．複数の本数を用いれば概ね 1μs 程度の遅延を隠ぺいできることがわかる．現状の DIMMnet-2 プロトタイプ上で SO-DIMM から Read window までの等間隔ベクトルロードコマンド実行にかかる時間は，512 バイトの Read window を一杯にする 128 ワードに対して，前述のように FPGA 版で 1.62μs，ASIC 版で 810ns である．つまり半分のクロック周波数で動作する FPGA 版では Pipelining unroll の利用形態をとっても性能低下が発生することがわかる．一方，ASIC 化して 200MHz 動作をさせれば Pipelining unroll の利用形態でほぼ隠ぺいできる．

6.3.3 キャッシュラインより大きなタプルサイズにおける性能

実際の現場で用いられている多くのデータベースのように Wisconsin ベンチマークより大きなタプルサイズを有するテーブルに対する問合せ性能を評価するために，キャッシュラインより大きな 140 バイトのタプルサイズを有するテーブルを作成し，同様の評価を行なった．その結果を**図 6** に示す．

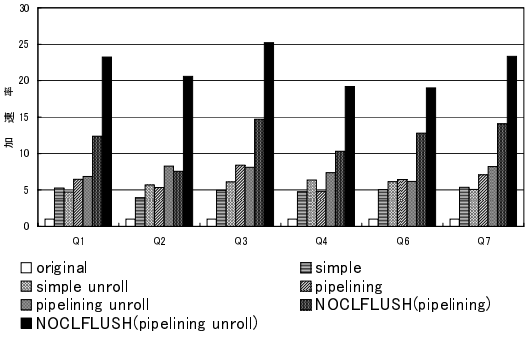

図 6 キャッシュラインより大きなタプルサイズにおける提案システムの効果

この結果から明らかなように，クエリー Q3 で 8.13 倍，CLFLUSH 命令の影響を排除すると 25.2 倍と，極めて高い加速率が得られることもわかった．その理由は，Pentium®4 を用いる場合は L2 キャッシュライン 128 バイト中に有効なデータが 4 バイトしか無くなり，Wisconsin ベンチマーク (60 バイト中 4 バイトが有効) よりもメモリバンド幅の浪費の程度が高まるため，Original の性能低下が起こるのに対し，提案メモリモジュールを用いた場合はそのような問題が発生せず，実行時間が Wisconsin ベンチマークの場合と比べてほとんど増加しないためである．

7. 関連研究

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[8] や，SDT[9] 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 ベンダーが採用しない限り PC クラスタに利用できる COTS とはならない．さらに SDT 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．データベース応用を想定した場合は，SDT だけではマルチウェイの結合演算は加速率が上がらないことがわかっているが，本方式は間接参照もベクトル化されるためにそのようなクエリーの高速化にも対応できる．

メモリコントローラ（ノースブリッジ）の改良として Impulse[10] 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ーヘッドが増加し，かつ 32bit 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 4GB) 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. おわりに

本報告では，キャッシュアーキテクチャの問題点を指摘し，その解決に向けて考案されたプリフェッチ機構付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た．さらにその DIMMnet-2 における実装例を紹介した．

キャッシュアーキテクチャが苦手とする等間隔アクセスを多用する応用として主記憶データベースをとりあげ，プリフェッチ機能を有するメモリモジュール用に改造された Wisconsin ベンチマークが PC 上で示す性能から，実機上での処理性能を推定した．

その結果，4.16 倍〜4.87 倍の加速率が得られることがわかった．この結果は逐次性能向上における加速率としては高い加速率ではあるが，CLFLUSH 命令の影響を排除することで提案メモリモジュールは例えばクエリー Q3 で 14.42 倍もの高い加速率が期待できることがわかった．さらに，SDT[9] ではほとんど加速できなかったマルチウェイの結合演算 (クエリー Q6) も等間隔ベクトルロードと間接ベクトルロードの組み合わせにより高速化が達成された．

Simple Scalar Tool Set などの命令レベルシミュレータを使う従来手法では内部詳細構造が非公開な Pentium®4 をホストとして用いる場合の CLFLUSH 命令のような特殊な命令のアプリケーション性能への影響を測定することは事実上困難であるが，本評価手法は Pentium®4 を対象として精密に結果に反映できている点で画期的な手法である．

また，DIMMnet-2 プロトタイプ上で実現されている等間隔ベクトルロードコマンド実行時間を考慮しても，その遅延時間は 2 枚の Read Window を用いることで ASIC 版の DIMMnet-2 では十分に隠ぺいできることもわかった．

さらに，実際の現場で用いられている多くのデータベースのように Wisconsin ベンチマークより大きなタプルサイズを有するテーブルに対する問合せに対してはクエリー Q3 で 8.13 倍，CLFLUSH 命令の影響を排除すると 25.2 倍と，極めて高い加速率が得られることもわかった．

このような高い加速率が COTS ベースの PC に提案メモリモジュールを装着するだけで得られるということは，本方式は CLFLUSH 命令問題の克服というさらなる改良の余地を残しつつも，文献 [9] における本手法はデータベースに向かないという指摘は適切ではなかったと考える．

DIMMnet-2 の実機での評価および他の PC 向け MPU における CLFLUSH 命令問題の確認とその克服，およびコンパイラによる対応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．SDT に関しご教授いただいた北陸先端科学技術大学院大学の田中助教授に感謝いたします．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，伊沢氏，宮部氏，東京農工大学の並木助教授，浜田氏，三橋氏，荒木氏，木立氏，森氏，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 (Mar. 2003)
2) B.Lindsay : “DB2 の父ブルース・リンゼイ博士が語る，データベース・テクノロジーの現在と未来”, http://db2.jp/interview/special/lindsay.html
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 (Mar. 2003)
4) 田邊, 中武, 箱崎, 土肥, 中條, 天野 : “プリフェッチ機能付きメモリモジュールによる不連続アクセスの連続化”, 情報処理学会計算機アーキテクチャ研究会, 2004-ARC-157, pp.139-144 (Mar. 2004)
5) Myricom®,Inc., *http://www.myri.com/*
6) Quadrics®,Ltd., *http://www.quadrics.com/*
7) 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 (Apr. 2002)
8) 近藤, 中村, 朴 : “SCIMA における性能最適化手法の検討”, 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.42 No.SIG12, pp.37-48 (Nov. 2001)
9) 宮崎, 府川, 田中 : “ストライドデータアクセスによる主記憶データベースの問合せ処理の評価”, 情報処理学会データベースシステム研究会, 2004-DBS-134(II), pp.361-367 (Jul. 2004)
10) 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 (Jan. 1999)