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

2007－ARC－173（4）
2007／5／31

チップマルチプロセッサにおけるデータ・プリフェッチ効果の分析

福本 尚人† 三原 智伸† 井上 弘士†† 村上 和彰††

† 九州大学大学院システム情報科学府 〒819-0395 福岡市西区元岡744番地
†† 九州大学大学院システム情報科学研究院 〒819-0395 福岡市西区元岡744番地
E-mail: †{fukumoto,mihara}@c.csce.kyushu-u.ac.jp, ††{inoue,murakami}@i.kyushu-u.ac.jp

あらまし 複数コアを1チップに搭載するチップマルチプロセッサ(CMP)が注目されている．CMPは，複数コアで並列処理することで高い演算性能を達成することができる．しかしながら，メモリバンド幅の制約や複数コア搭載によるメモリアクセス頻度の上昇により，メモリウォール問題が深刻化する．主記憶のアクセス時間を隠蔽する方法のひとつにデータ・プリフェッチがある．CMPにおいてデータ・プリフェッチを行う場合，コア間の相互作用があるため，シングルコアプロセッサとは異なる効果が現れる．そこで本稿では，CMPにおけるデータ・プリフェッチが性能へ与える影響を分析した．その結果，プリフェッチしたデータが無効化される割合は極めて小さく，プリフェッチを発行したコア以外のメモリアクセス時間を隠蔽するプリフェッチが約5%あることが明らかになった．
キーワード チップマルチプロセッサ，データ・プリフェッチ，キャッシュメモリ

Effect of Data Prefetching on Chip MultiProcessor

Naoto FUKUMOTO†, Tomonobu MIHARA†, Koji INOUE††, and Kazuaki MURAKAMI††

† Graduate School of Information Science and Electrical Engineering, Kyushu University
744 Motooka Nishi-ku Fukuoka 819-0395 JAPAN
†† Faculty of Information Science and Electrical Engineering, Kyushu University
744 Motooka Nishi-ku Fukuoka 819-0395 JAPAN
E-mail: †{fukumoto,mihara}@c.csce.kyushu-u.ac.jp, ††{inoue,murakami}@i.kyushu-u.ac.jp

Abstract Chip MultiProcessors (or CMPs) can achieve higher performance by means of exploiting thread level parallelism. Increasing the number of processor cores in a chip dramatically improves the peak performance. However, since the memory bandwidth does not scale with the number of cores, the negative impact of the memory-wall problem becomes more critical. Data prefetching is a well known approach to compensating for the poor memory performance, and has been employed in commercial processor chips. Although a number of prefetching techniques have so far been proposed, in many cases, they have assumed that the processor core in a chip is only one. In CMP chips, there are some shared resources such as L2 caches, buses, and so on. Therefore, the effect of prefetching on CMPs should be different from that on single-core processors. In this paper, we analyze the effect of prefetching on CMP performance. This paper first classifies the impact of prefetch operations issued during a program execution. Then, we discuss qualitatively and quantitatively the effect of prefetching to the memory performance. The experimental results show that the negative effect of invalidation of prefetched data is very small. In addition, it is observed that about 5% of prefetch operations improve the cache hit rates of other cores.
Key words CMP, deta prefetching, cache memory

1. はじめに

マイクロプロセッサの高性能化は主に動作周波数の向上や命令レベル並列性の利用によって達成されてきた．しかしながら，近年では命令レベル並列性の利用の限界や消費電力の増大により，これらの手法では性能を向上させることが難しくなっている．この問題を解決する方法として，1チップ上に複数のプロセッサコア（以降，コアと略す）を搭載するチップマルチプロセッサ(CMP:Chip MultiProcessor)が注目されている．CMPは各コアを低周波数で並列処理させることで，高性能かつ低消

費電力を実現することができる．

しかしながら，搭載コア数を増やしても必ずしもプロセッサ全体の性能は向上するわけではない．その主な理由として，プロセッサシステム全体の性能が主記憶の性能により抑制されるメモリウォール問題が挙げられる．CMP では複数のコアが搭載されるため，シングルコアプロセッサと比較して主記憶へのアクセス頻度が高くなる．しかしながら，プロセッサチップの I/O ピン数は物理的な制約により制限されるため，十分なメモリバンド幅を確保することが困難となる．その結果，メモリウォール問題は CMP においてより顕著に現れる．

メモリアクセス時間を隠蔽する手法の一つにデータ・プリフェッチ（以降，プリフェッチと略す）がある．将来参照されるであろうデータを事前にキャッシュメモリへ読込むことにより，メモリアクセス時間を隠蔽できる．しかしながら，必ずしもプリフェッチによりメモリ性能が向上するわけではない．例えば，プリフェッチしたデータによってキャッシュメモリから有用なデータが追出される場合や，プリフェッチ発行によりバスの競合が発生する場合はメモリアクセス時間が増加する．

これまでに，プリフェッチに関する多くの手法が提案されてきた．これらの多くはシングルコアプロセッサを前提としており，実際に多くの商用プロセッサで実用化されている．CMP においても，シングルコア向けに考案されたプリフェッチ技術を用いることでメモリウォール問題を緩和できる．しかしながら，CMP ではオンチップキャッシュやメモリバスを複数コアが共有するため，シングルコアの場合と比較してプリフェッチ効果は異なる．したがって，メモリ性能を更に向上するためには，CMP の特性を考慮した効率的なプリフェッチ手法を考案する必要がある．

CMP において，より効率的なプリフェッチ手法を考案するには，CMP におけるプリフェッチ効果の詳細な分析が必要である．これまでに，マルチプロセッサに関してはプリフェッチ効果の分析が数多く行われている [3]〜[5]．CMP とマルチプロセッサは，共通のメモリ階層に接続され，コヒーレンスを維持するという共通点がある．しかしながら，マルチプロセッサは複数の CPU チップがオフチップ・ネットワークにより接続されるのに対し，CMP では各コアが高速なオンチップ・ネットワークで接続される．この相違点により，マルチプロセッサと CMP ではプリフェッチ効果が異なると予想される．そのため，マルチプロセッサにおけるプリフェッチ効果の分析結果を直接 CMP に適用することはできない．

そこで本稿では，CMP におけるプリフェッチが性能へ与える影響を解析する．まず，メモリ性能に与える影響に基づき，プログラム実行において発行される各プリフェッチを分類する．次に，性能モデル式を構築し，分類結果を基に CMP でのプリフェッチ効果を議論する．その後，分類したプリフェッチの割合を定量的に評価することで，プリフェッチがメモリアクセス時間へ与える影響について考察する．

本稿の構成は以下のとおりである．まず，第 2 節で対象とする CMP モデルについて説明する．次に，第 3 節で文献 [4] により提案されたマルチプロセッサにおけるプリフェッチ効果の分析方法を説明し，第 4 節でこの分析方法を CMP 向けに拡張する．その後，第 5 節で性能モデル式を構築し，プリフェッチがメモリアクセス時間へ与える影響を定性的に評価する．さらに，第 6 節でベンチマークプログラムを用いた定量的な評価を行い，第 7 節でまとめる．

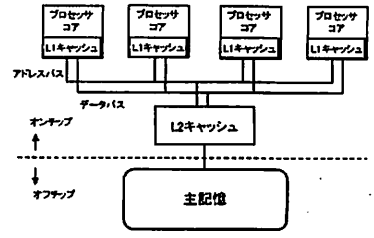

図 1　対象とする CMP モデル

2. 対象とするチップマルチプロセッサモデル

2.1 CMP モデル

図 1 に評価対象となる CMP の概略を示す．各構成要素の詳細は以下の通りである．

- コア：4 個のインオーダのコアが 1 チップ上に搭載される．
- オンチップ・キャッシュ
  - 各コアに命令/データ分離型の L1 非共有キャッシュが搭載される．L2 キャッシュは命令/データ統合型の共有キャッシュである．
  - ノンブロッキングアクセスが可能である．
  - ライトバック方式により下位のメモリ階層に書戻しを行う．
  - コヒーレンスプロトコルはライトインバリデート型の MOESI プロトコルを想定する．
- オンチップ・ネットワーク：アドレス/データ分離型の共有バスにより，L2 キャッシュと各コアの L1 キャッシュが接続される．
- プリフェッチ機構
  - L1 キャッシュへプリフェッチする．
  - 主記憶からプリフェッチしたデータは，L2 キャッシュ，L1 キャッシュ双方に書込まれる．
  - プリフェッチ手法は tagged プリフェッチ [6] またはストライドプリフェッチ [1] を使用する (詳細は 2.2 節にて説明する)．

2.2 プリフェッチ手法

tagged プリフェッチ [6] は，next line(sequential) プリフェッチの一種である．あるブロックアドレス a へのアクセス時にキャッシュミスが発生した場合，もしくは，ヒットでありかつ当該データが過去にプリフェッチされたデータであった場合，ブロックアドレス $a+1, a+2, \cdots, a+d$ のデータのプリフェッチ要求を発行する．d は一度に発行するプリフェッチ要求の数を表し，本稿では 5 とする．

ストライドプリフェッチ [1] は，メモリアクセスパターンからメモリアクセスアドレスを予測しプリフェッチする手法であ

る．参照ブロックアドレスが $a-s$，前回と前々回の参照ブロックアドレスの差分が s のとき，次の同じ PC を持つ命令のメモリアクセスアドレスが a だった場合，$a+s, a+2s, \cdots a+ds$ のデータをプリフェッチする．tagged プリフェッチとは異なり，命令の PC，メモリアクセスアドレス，差分値を記録するテーブルが必要となる．本稿ではテーブルのエントリ数を 64，d の値を 5 とする．

3. マルチプロセッサにおけるプリフェッチ効果

Natalie ら [4] は，マルチプロセッサにおけるプリフェッチ効果の分析方法を考案した．この分析方法は，プログラム実行中に発行された各プリフェッチがメモリ性能に与える影響について考える．ここでは，各プリフェッチ処理においてアクセス対象となるデータ（以降，プリフェッチ・データと呼ぶ）に着目する．そしてプリフェッチ・データのキャッシュ滞在期間中に発生する各種イベント（プリフェッチ・データの参照など）に基づき，発行されるプリフェッチがメモリ性能へ与える影響を分類する．具体的には，プリフェッチ・データが取り得る 6 つの状態を定義し，当該データがキャッシュに読込まれてから追出されるまでの間に発生したイベントに基づき状態を遷移する．発行された各プリフェッチに対してキャッシュメモリから追出されるときのプリフェッチ・データの状態を調べることで，プリフェッチ効果を詳細に分析することができる．

マルチプロセッサにおいてプリフェッチを行う場合，以下のようなイベントによりメモリ性能は変化する．

- イベント 1：プリフェッチ・データの参照．メモリアクセス時間を削減することができる．
- イベント 2：プリフェッチ・データによって追出されたデータへの参照．プリフェッチデータがキャッシュメモリに読込まれることにより，後に参照される有用なデータがキャッシュメモリから追出される．そのため，プリフェッチを行わない場合には発生しないキャッシュミスが発生する．
- イベント 3：プリフェッチにより共有状態になったデータへの書込み．図 2 のような場合，プリフェッチにより本来なら発生しない無効化要求が発生する．時刻 2 に CPU0 が CPU1 の Modified のデータをプリフェッチすることによりデータが共有される．時刻 4 に CPU1 が共有データに書込みを行うことにより CPU0 のプリフェッチしたデータを無効化する要求が発行される．プリフェッチを行わない場合には，CPU1 のキャッシュブロックの状態は Modified から変化せず，無効化要求は発行されない．

時刻	CPU0 命令	CPU0 Cache state	CPU1 命令	CPU1 Cache state	Coherence Traffic
1			store A	Modified	CPU1 ライトヒット
2	prefetch A	Shared			CPU0 プリフェッチ発行
3				Owned	CPU1 Ownedへ
4			store A	Modified	CPU1 Modifiedへ
5		Invalid			CPU0 無効化 (broad cast)

図 2　プリフェッチによる無効化要求増加の例

これらのイベントを遷移条件とするマルチプロセッサでのプリフェッチ・データの状態遷移は図 3 のようになる．各状態は過去に起こったイベントの履歴を表し，以下のように定義される．

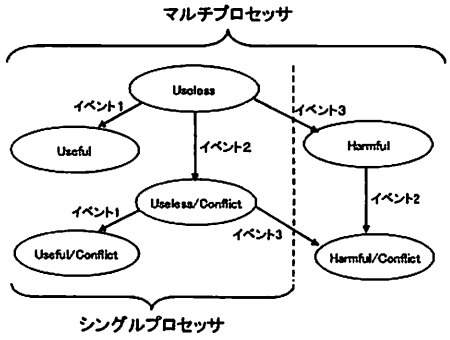

図 3　プリフェッチ・データの状態遷移

- Useless：プリフェッチ・データに対してどのイベントも発生していない状態．プリフェッチしたデータがキャッシュに読込まれた直後は必ずこの状態になる（初期状態）．
- Useful：過去に，イベント 1 が発生した状態．
- Useless/Conflict：過去に，イベント 2 が発生した状態．
- Useful/Conflict：過去に，イベント 2，イベント 1 が順に発生した状態．
- Harmful：過去に，イベント 3 が発生した状態．プリフェッチ・データは無効化されているため，参照されることはない．
- Harmful/Conflict：過去に，イベント 2，イベント 3 が発生した状態．

これらの状態の中で，マルチプロセッサに固有の状態は Harmful，Harmful/Conflict で，その他の状態はシングルプロセッサでも存在する．

Natalie らがマルチプロセッサにおいてこれらの状態の割合を定量的に評価した結果，Harmful と Harmful/Conflict の合計が全体の約 23%あり，トラフィックの増加に大きく寄与していることが分かった．また，プリフェッチによるトラフィックの増加が性能へ大きな影響を与えることも明らかにされた．

4. CMP におけるプリフェッチ効果

CMP は，複数チップで構成されるマルチプロセッサとは異なり，他のコアがオンチップに読込んだデータに対して高速にアクセスできる．したがって，マルチプロセッサの場合に加えて，以下のようなイベントが追加される．

- イベント 4：プリフェッチ・データの他コアからの参照．プリフェッチを発行したコア（以降，プリフェッチ発行コアと呼ぶ）以外のコアから当該プリフェッチデータを参照される．このイベントはプリフェッチ・データの取得先が L2 キャッシュもしくは主記憶の場合のみ発生する．なぜならば，これらの場合のみメモリアクセス時間が変化するためである．

CMP におけるプリフェッチ・データの状態は，図 4 のよう

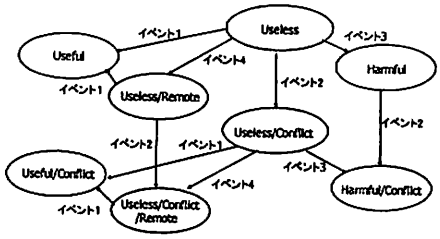

図 4 プリフェッチ・データの状態遷移

な 8 状態となる．マルチプロセッサの場合と比較して追加された状態は，以下の 2 つである．

- Useless/Remote : 過去に，イベント 4 が発生した状態．
- Useless/Conflict/Remote : 過去に，イベント 2，イベント 4 が発生した状態．

これらの状態はそれぞれ Useless，Useless/Conflict から，イベント 4 が発生した場合に遷移する．その後，Useless/Remote，Useless/Conflict/Remote においてイベント 1 が発生した場合，それぞれ Useful，Useful/Conflict に遷移する．

次に，プリフェッチ・データが追出された時の状態（最終状態）に基づき，プリフェッチによるキャッシュミス回数とメモリアクセス回数の変化を考える．プリフェッチ・データが各状態に至るまでに与えたキャッシュアクセス回数や共有バスへのアクセス回数の変化を表 1 に示す．表の「他のコア」はプリフェッチ発行コアがプリフェッチすることで影響を受ける周りのコアを指す．また，共有バスアクセス回数の Data/Address はデータバス/アドレスバスに対するアクセス回数の変化を表す．

表 1 各状態に至るまでに与えるメモリ性能への影響

状態名	プリフェッチ発行コアの L1 キャッシュミス回数	他のコアの L2 アクセス回数	共有バスアクセス回数 (Data/Address)
Useless	±0	±0	+1/+1
Useless/Remote	±0	−1	+1/+1
Useful	−1	±0	±0/±0
Harmful	±0	±0	+1/+2
Useless/Conflict	+1	±0	+2/+2
Useless/Conflict/Remote	+1	−1	+2/+2
Useful/Conflict	±0	±0	+1/+1
Harmful/Conflict	+1	±0	+2/+3

表 1 より，プリフェッチによるメモリアクセス時間の変化について考える．メモリアクセス時間の増減は，表のプリフェッチ発行コアのキャッシュミス回数，他のコアの L2 アクセス回数の増減に対応している．ただし，Useful/Conflict のメモリアクセス時間は，主記憶または L2 キャッシュからプリフェッチした場合には改善し，他のコアの L1 キャッシュからプリフェッチした場合には変化しない．また，Useful 以外はプリフェッチを行わない場合と比較して，共有バスアクセス回数が増加している．これらのアクセスによって競合が発生する場合はメモリアクセス時間が増加する．

5. 性能モデル式による評価

本節では，性能モデル式を構築し，前節を基にプリフェッチが性能モデル式の項に与える影響について考察する．

1 つのスレッドについての実行クロックサイクル数から性能モデル式を構築する．あるスレッドの実行クロックサイクル数 CC は，演算実行に要するクロックサイクル数 CC_{exe}，メモリ参照によりストールしたクロックサイクル数 CC_{mem}，オーバーラップ実行したクロックサイクル数 $CC_{overlap}$ を用いて，

$$CC = CC_{exe} + CC_{mem} - CC_{overlap} \quad (1)$$

と表すことができる．

CC_{mem} は式 (2) のように表すことができる．

$$\begin{aligned} CC_{mem} = AC \times \{ & HCC_{L1} + MR_{L1} \times \\ & ((1 - MR_{L1R}) \times (SBCC + HCC_{L1}) + \\ & MR_{L1R} \times \quad ((HCC_{L2} + SBCC) + \\ & MR_{L2} \times (MBCC + MC_{L2}))) \} \end{aligned} \quad (2)$$

各項の定義は以下の通りである．

- AC:メモリ参照回数
- HCC_{L1}:L1 キャッシュアクセスに要する時間
- MR_{L1}:L1 キャッシュミス率
- $SBCC$:L1-L2 キャッシュ間のバスアクセスに要する時間
- MR_{L1R}:他のコアの L1 キャッシュに対するミス率
- HCC_{L2}:L2 キャッシュアクセスに要する時間
- MR_{L2}:L2 キャッシュミス率
- $MBCC$:L2-主記憶間のバスアクセスに要する時間
- MC_{L2}:主記憶アクセスに要する時間

次に，式 (2) においてプリフェッチを行うことによるメモリアクセス時間の変化について考える．プリフェッチを行わない場合と比較すると，式 (2) の以下の項が変化する．

- MR_{L1} : Useless/Conflict に遷移するプリフェッチの発行回数と Useful に遷移するプリフェッチの発行回数の割合により増減が決まる．Useless/Conflict より Useful が多くなる場合，MR_{L1} は減少し，逆の場合は MR_{L1} は増加する．
- $SBCC$:共有バスへのアクセス回数により変化する．Useful 以外に遷移するプリフェッチが発行される場合，共有バスへのアクセス回数は増加する．アクセス回数の増加により，共有バスで競合が発生する場合 $SBCC$ は増加する．
- MR_{L1R} : 注目したスレッドを実行しているコア以外の Useless/Remote, Useless/Conflict/Remote に遷移するプリフェッチにより MR_{L1R} は減少する
- MR_{L2} : プリフェッチにより L2 キャッシュに読込まれた有効に使用されるデータの個数と L2 キャッシュから追出された有用なデータの個数の割合で増減が決まる．
- $MBCC$: 主記憶へのアクセス回数により変化する．Useful 以外に遷移するプリフェッチにより主記憶アクセスが増え，これらのアクセスにより競合が発生する場合，MC_{L2} は大きくなる．

以上より，プリフェッチによる MR_{L1}，MR_{L1R}，MR_{L2} の改善によるメモリアクセス時間の減少を $SBCC$，$MBCC$ によるメモリアクセス時間の増加が上回る場合，CC_{mem} が増加することが分かる．また，プリフェッチにより MR_{L1}，MR_{L2} が悪化する場合は，他の項の変化にかかわらずメモリアクセス時間は増加する．

表 2　評価対象の主な構成

キャッシュメモリ	
L1 命令キャッシュ	64KB 2-way, 64B lines, 1 clock cycle
L1 データキャッシュ	64KB 2-way, 64B lines, 1 clock cycle
L2 共有キャッシュ	4MB 8-way, 64B lines, 12 clock cycles
ネットワーク	
オンチップ共有バス幅	64B
L2-主記憶間バス幅	16B
DRAM レイテンシ	300 clock cycles

6. 評　　価

6.1 評価環境

Michigan 大学で開発された CMP シミュレータ M5 [2] 用いてプリフェッチ分析のための評価環境を構築した．評価対象の CMP の構成は第 2 節の CMP モデルに従っている．CMP の主な構成のパラメータを表 2 に示す．ベンチマークプログラムは，並列計算用の科学技術計算のベンチマークプログラムである SPLASH2 [7] を用いた．各ベンチマークプログラムの入力パラメータを表 3 に示す．ただし，表 3 に記述されていない入力パラメータはデフォルトの値を使用した．以上の条件で，プログラムの実行開始から終了までシミュレーションを行った．また，1000 命令あたりの平均プリフェッチ発行回数を表 3 に記載している．

表 3　SPLASH2 の入力，プリフェッチの発行回数

ベンチマーク	入力パラメータ	ストライド発行数	tagged 発行数
barnes	8k particles	0.05	12
fmm	16k particles	0.34	7.2
lu(contig)	512 × 512matrix	1.2	2.2
radix	256K keys	1.4	26
raytrace	teapot.env	0.54	10
water(spatial)	512 molocules	0.02	1.4

6.2 評価結果

図 5 に第 4 節で定義したプリフェッチ・データの最終状態の割合を示す．ストライドプリフェッチはすべてのベンチマークプログラムで Useful が全体の 80%以上を占め，他の状態の割合が相対的に少ない．したがって，プリフェッチによるトラフィックの増加などの悪影響は小さいと予想される．一方，tagged プリフェッチは，Useful の割合は 30%程度で，それ以外の状態が多いため，メモリアクセス回数を増加させるプリフェッチが大部分を占める．また，Useless/Conflict も全体の約 20%を占め，Useful と同程度の割合で存在する．特に Radix では Useful の割合よりも Useless/Conflict の割合のほうが大きいため L1 キャッシュミス率が悪化している．Harmful，Harmful/Conflict の状態に遷移するプリフェッチは最大 1%以下で性能へ与える影響は小さい．プリフェッチ発行コアにとって有益ではないが，それ以外のコアが参照してメモリアクセス時間を削減できる Useless/Remote，Useless/Conflict/Remote は全体の約 5%を占める．

図 6 にプリフェッチによる平均メモリアクセス時間の変化を示す．縦軸は平均メモリアクセス時間で，横軸はベンチマークプログラムである．左から順にプリフェッチを行わない場合，ストライドプリフェッチ，tagged プリフェッチ適用する場合を表す．なお，平均メモリアクセス時間は各コアの平均をとったものである．凡例の各項目の HCCL1，HCCRL1，HCCL2，SBCC，MBSS，MCL2 は，それぞれ L1 キャッシュ，他のコアの L1 キャッシュ，L2 キャッシュ，共有バス，メモリバス，主記憶の平均アクセス時間を表す．これらの項目は，シミュレーション結果を利用して以下の計算式により求める．

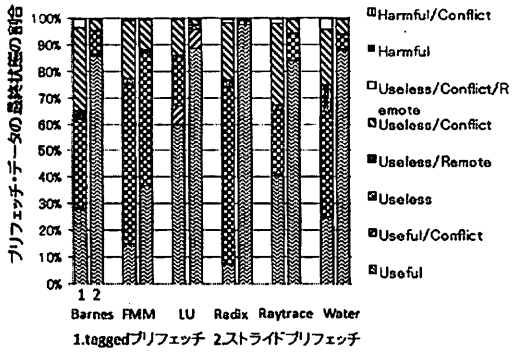

図 5　プリフェッチ・データの最終状態の割合

- HCCRL1=$MR_{L1} \times (1 - MR_{L1R}) \times HCC_{L1}$
- HCCL2=$MR_{L1} \times MR_{L1R} \times HCC_{L2}$
- SBCC=$MR_{L1} \times SBCC$
- MBSS=$MR_{L1} \times MR_{L1R} \times MR_{L2} \times MBCC$
- MCL2=$MR_{L1} \times MR_{L1R} \times MR_{L2} \times MC_{L2}$

ただし，各項目のミス率にプリフェッチによるアクセス回数は含まれない．また，L1 キャッシュミス率と他のコアに対する L1 キャッシュミス率は，各コアの平均値を使用している．

プリフェッチを行うことにより，全てのベンチマークプログラムで主記憶アクセスの時間が減少し，平均メモリアクセス時間が減少している．Useful の割合が 80%を超えるストライドプリフェッチより，tagged プリフェッチの方がすべてのベンチマークプログラムにおいて平均メモリアクセス時間が短い．これは，ストライドプリフェッチの発行数が tagged プリフェッチと比較して少なく（表 3），キャッシュミス回数の削減数が tagged プリフェッチより小さいこと，トラフィック増加による

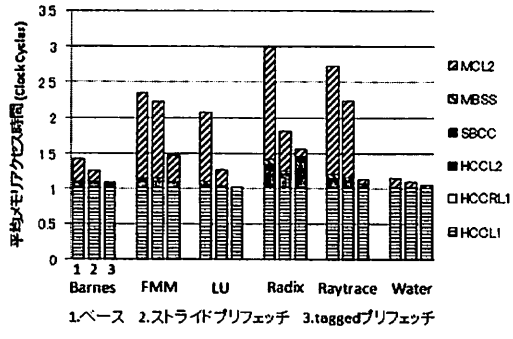

図 6　平均メモリアクセス時間

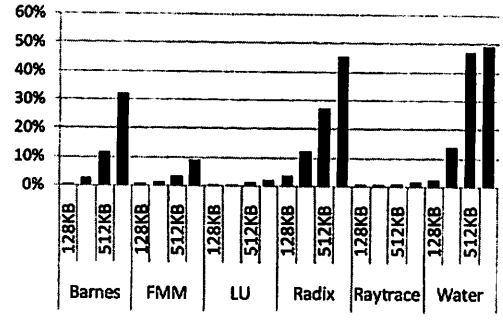

図 7 プリフェッチ・データが無効化される割合
(L1 Dcache=128,256,512,1024 KB)

メモリアクセス時間の増加量が小さいことによる.

6.3 プリフェッチが性能へ与える影響

評価結果より，プリフェッチ・データの最終状態の割合がメモリアクセス時間へ与える影響について考察する．ストライドプリフェッチは Useful 以外の割合が小さく，プリフェッチの発行数が少ないため，性能へ与える影響が小さい．そこで tagged プリフェッチの結果より，プリフェッチの性能へ与える影響を考える．Useful の割合は全体の 30%程度で，プリフェッチ発行数の 70%がメモリアクセス回数を増加させる．しかしながら，L1-L2 間のバスのアクセス時間の増加量は小さい．これはオンチップのメモリバンド幅が十分にあり，アクセス競合の発生が抑制されたことが要因である．また，Useless/Conflict に遷移するプリフェッチが全体の約 20%を占めている．このプリフェッチにより，キャッシュミス回数が増え，プリフェッチによるメモリアクセス時間削減量が小さくなる．Useful，Useless/Conflict が同程度存在するが，全てのベンチマークで平均メモリアクセス時間は減少している．これは，プリフェッチ・データの読込みによる L2 キャッシュからの有用なデータの追出しが発生した回数と比較して，プリフェッチ・データの参照数が多いためである.

6.4 プリフェッチによる無効化要求の増加が少ない理由

マルチプロセッサの場合と違い，Harmful，Harmful/Conflict の割合が極めて少ない理由について考察する．これらの状態が極めて少ない理由は，プリフェッチの発行される回数に対して，共有データへの書込み回数が極めて少ないことが要因である（stride プリフェッチで平均 77 倍の差がある）．共有データへの書込みが少ない理由は L1 キャッシュサイズが小さいため，データが共有状態になる前に L1 キャッシュから追出されているためだと考えられる．図 7 にプリフェッチ・データが無効化される割合のグラフを示す．L1 キャッシュサイズは左から，128KB，256KB，512KB，1MB で，プリフェッチ手法は tagged プリフェッチを使用している．確かに，L1 キャッシュサイズが大きくなると共にプリフェッチ・データが無効化される割合が増加している.

7. おわりに

本稿では，CMP におけるプリフェッチが性能へ与える影響を解析し考察した．その結果，共有キャッシュ/共有バス型の CMP とマルチプロセッサでは，プリフェッチ効果が大きく異なることが明らかになった．CMP では，プリフェッチによるトラフィックの増加がメモリアクセス時間に与える影響が小さく，プリフェッチしたデータが無効化される割合は 1%以下であることが明らかになった．また，プリフェッチを発行したコア以外のメモリアクセス時間を隠蔽するプリフェッチが約 5% (tagged プリフェッチ) あり，性能向上に寄与していることが明らかになった．以上のことから，CMP 専用のプリフェッチ手法を考える場合，トラフィックの増加やプリフェッチしたデータの無効化による性能低下より，有用なデータの追出しや周りのコアによる参照などを考慮する必要性が高いことが分かった.

今後は，プリフェッチが実行時間やバスの競合などに与える影響を詳細に解析する予定である．また，プリフェッチを発行したコアだけではなく，その他のコアのメモリアクセス時間を効果的に削減できるプリフェッチ手法の実現を目指す.

謝辞 日頃からご議論頂く安浦・村上・松永・井上研究室ならびに九州大学システム LSI 研究センターの皆様に感謝します．なお，本研究は一部，科学研究費補助金 (若手研究 A:課題番号 17680005) による.

文 献

[1] J. L. Baer and T. F. Chen. "An Effective On-Chip Preloading Scheme to Reduce Data Access Penalty". In Proceedings of the 1991 Conference on Supercomputing, pp. 176-186, 1991.

[2] N. L. Binkert, E. G. Hallnor, and S. K. Reinhardt. "Network-oriented full-system simulation using m5". In Sixth Workshop on Computer Architecture Evaluation using Commercial Workloads, February 2003.

[3] F. Dahlgren and P. Stenström. "Evaluation of Hardware-Based Stride and Sequential Prefetching in Shared-Memory Multiprocessors". IEEE Transactions on Parallel and Distributed Systems, pp.385-398, 1996.

[4] N. D.Enright Jerger, E. L. Hill, and M. H. Lipasti. "Friendly Fire: Understanding the Effects of Multiprocessor Prefetching". In Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, March 2006.

[5] M.J. Garzaran, J.L. Briz, P.E. Ibanez, and V. Vinals. "Hardware prefetching in bas-based multiprocessors: Pattern characterization and cost-effective hardware". In Proceedings of Parallel and Distributed Processing 2001, pp. 345-354, February 2001.

[6] A. J. Smith. "Cache Memories," Computing Surveys, Vol.14, No.3, pp. 473-530, September 1982.

[7] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. "The SPLASH-2 programs: Characterization and methodological considerations". In Proceedings of the 22th International Symposium on Computer Architecture, June 1995.