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

2007－ARC－173 (11)
2007／6／1

高速かつ正確なキャッシュシミュレーション法とその評価

小野 貴継† 井上 弘士†† 村上 和彰††

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

あらまし 本稿では，高速かつ正確なキャッシュシミュレーション法について述べ，先行研究と定量的な比較を行い有効性を評価する．一般に，キャッシュメモリのシミュレーションにはトレース・ドリブン方式が用いられる．設計空間の拡大に伴い評価対象が増加しており，評価時間が長くなる傾向にある．トレース・サイズの削減によりシミュレーション時間を短縮できるが，精度が低下するという問題が生じる．そこで，本手法はメモリ・アクセスの特徴を利用し，精度を維持しつつ時間の短縮を実現する．先行研究と比較した結果，トレース・サイズは平均81.7%削減され，キャッシュ・ミス率の予測精度は平均34.6%向上した．

キーワード 性能評価，シミュレーション，メモリ・システム

Fast, Accurate Cache Simulation

Takatsugu ONO†, Koji INOUE††, and Kazuaki MURAKAMI††

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

Abstract This paper proposes a fast, accurate cache simulation technique for efficient design space exploration, and shows its efficiency by means of comparing with a related approach. Trace-driven simulation is a well known methodology to measure memory-system performance, e.g. cache hit rates. One of advantages of this method is the high-speed of simulations. Since the trend increases the complexity of microprocessor chips, e.g. CMPs, however, it is strongly required to achieve much faster simulations without sacrificing the accuracy of performance prediction. The proposed approach first attempts to characterize the memory-access patters, and then generates a small but well-constructed memory-access trace as a stimulus of cache simulators. In our evaluation, it is observed that the proposed technique reduces the trace size by 81.7% while the accuracy of cache miss rates is improved by 34.6%, compared with SimPoint approach.

Key words Performance evaluation, Simulation, Memory system

1. はじめに

メモリシステムが計算機性能に与える影響は極めて大きい．したがって，高性能な計算機システムを構築するためには設計制約を満足する適切なメモリアーキテクチャを決定する必要がある．多くの場合，設計空間の探索を目的としてソフトウェア・シミュレーションによる性能評価が行われる．しかしながら，実機での実行に比べてシミュレーション速度は数桁遅いため，広大な設計空間を短期間で効率的に探索するのは難しいのが現状である．

メモリアーキテクチャ・シミュレーションの代表的な手法として，プログラム実行において発生したメモリアクセス情報を事前に採取し，それを入力とするトレース・ドリブン・シミュレーション法がある．より詳細なシミュレーションを目的とした実行ドリブン方式と比較して，高い抽象度での実行が可能なため高速に動作することができる．しかしながら，今後，メニーコアに代表されるより複雑かつ大規模なシステムを対象とした場合，トレース・ドリブン方式における更なる高速シミュレーションが必要となる．一般に，トレース・ドリブン方式の速度と精度は入力となるトレース・サイズに大きく依存してお

り，これらの間にはトレードオフ関係が存在する．例えば，プログラム実行のある区間のみをトレース採取の対象とした場合，トレース・サイズの削減によりシミュレーション時間を短縮できる．しかしながら，プログラム実行に関するすべての振る舞いを反映できないため，シミュレーション結果の精度が低下するといった問題が生じる．

そこで著者らは，メモリシステムの中でも設計選択肢が多く計算機性能に大きな影響を与えるキャッシュを対象とし，高速かつ正確なシミュレーションを可能とするメモリアーキテクチャ・シミュレーション法を提案した [1]．本手法はまず，プログラムの実行開始から終了までを対象とした全アドレス・トレース（フル・トレース）を取得する．そして，それを小規模なアドレス・トレース（サブ・トレース）に分割し，メモリアクセスの特徴を抽出する．得られた特徴の類似性に基づき，代表となるサブ・トレースを選択する．この代表サブ・トレースをもとに小規模なトレースを生成しシミュレーションすることで，精度を維持しつつ時間の短縮を実現する．さらに，この小規模なトレースは，一度生成すると異なるメモリアーキテクチャのシミュレーションにも使用することができる．したがって，評価時間を大幅に削減することが可能である．本稿では，先行研究との比較を行い，本手法の有効性を検証する．

以下，2. でこれまでに行われてきた関連研究について述べる．3. では提案手法について述べる．4. でその有効性を評価し，5. でまとめおよび今後の課題について述べる．

2. 関連研究

シミュレーション時間は，シミュレータの実行速度とその入力に依存する．シミュレータの高速化技術として，シミュレーション過程を時間軸方向に分割して並列化し，その精度を低下させることなく高速に実行する手法が提案されている [2]．シミュレータの入力，つまりシミュレーションの対象区間を削減し高速化する手法も多く提案されている [3]．シミュレーション区間を削減することで比較的高速に実行できる一方精度が低下するため，いかに精度を維持するかが重要な課題となる．本稿では，より高速なシミュレーションを実現するため，後者の技術に着目する．精度を維持しつつシミュレーション時間を削減する手法は以下の 3 つに大別できる．

（1） プログラムの入力データを削減

（2） プログラムの特徴を維持しつつ小規模なプログラムを生成

（3） シミュレーションの対象区間をサンプリング

手法 (1) に属するものとして MinneSPEC [4] が挙げられる．入力データの削減により実行時間は短縮するが，メモリアーキテクチャのシミュレーション精度が低いという問題がある．

(2) のアプローチをとっているものとして [5], [6] がある．これらの先行研究では，評価対象システムにおいて 1 度プログラムを実行し，あるメモリ構成を前提とした小規模なベンチマーク・プログラムを生成している．そのため，異なるメモリ構成での評価を行う場合は，新たに小規模ベンチマークを作り直す必要がある．一方，本稿で提案する手法はメモリアーキテクチャに依存しないため，一度小規模なトレースを生成することで異なるメモリ構成においてもシミュレーション可能である．

(3) に分類される代表的な手法として SMARTS [7] および SimPoint [8], [9] が挙げられる．本稿で提案する手法も本分類に属する．SMARTS は，ある固定インターバルによって命令ストリームをサンプリングする．プログラムの振る舞いに関係なく一定周期でサンプリングするため，フェーズの変化と周期が一致しない場合は精度が低下するといった問題が生じる．SimPoint はプログラム全体の特徴を解析してサンプリングするため，このような問題が生じることはない．SimPoint はプログラムの実行開始から終了までを一定命令数のインターバルで区切り，各インターバルにおける基本ブロックの実行回数によって，インターバルの特徴付けをしている．この特徴の類似性に基づいてインターバルをクラスタリングし，各クラスタから 1 つのインターバルをシミュレーションの対象として選出する．選出されたインターバルのみをシミュレーションし，結果に重み付けしている．

本稿における提案手法は，各インターバルの特徴を解析して代表を選出するため SMARTS のような問題が生じることはない．また，各インターバルをメモリアクセスによって特徴付けしており，この点が SimPoint と異なる．したがって，提案手法によるキャッシュシミュレーションの方がより高い精度を達成できる．

3. キャッシュシミュレーション法

3.1 提案手法の概要

本手法はまず，アプリケーション・プログラムを実行してフル・トレースを得る．これを一定クロック・サイクル間隔でサブ・トレースに分割する．次に，すべてのサブ・トレースを 3.3 で説明する特徴の類似性に基づきクラスタリングし，各クラスタからシミュレーションの対象となる代表サブ・トレースを 1 つ選出する．サブ・トレースの特徴が同じであれば，それらのシミュレーション結果は非常に近い結果になると考えられるからである．選出されたサブ・トレースのシミュレーション結果に，各クラスタに属するサブ・トレース数に基づいて重み付けする．このように，サブ・トレースの特徴に基づいてシミュレーション対象とするサブ・トレースを決定することで，高速かつ正確なシミュレーションを実現する．

3.2 メモリアクセスの特徴抽出

メモリアクセスには時間局所性および空間局所性があることが知られている．これらの特徴はメモリシステムのシミュレーションにおいて重要である．また，単位時間あたりのメモリアクセス数や局所性は，プログラム実行におけるフェーズと共に変化する．したがって，フル・トレースを一定の間隔でサブ・トレースに分割し，その区間において特徴を抽出すると効果的であると考えられる．サブ・トレースの分割方法の選択肢として，命令単位とプログラム開始時刻からの経過クロック・サイクル数（時間）単位が挙げられる．本稿では評価対象をキャッシュとしているが，他のメモリ・アーキテクチャの評価への適

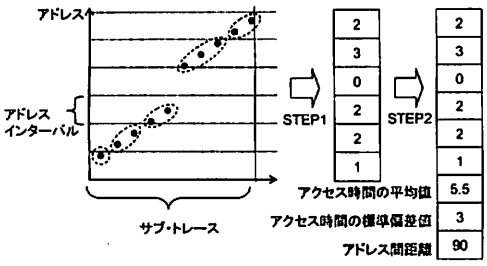


図 1　特徴抽出手順

用を考慮し，時間単位で分割する．さらに，空間局所性を抽出するために，サブ・トレースにおけるアドレス空間を図 1 のように一定の間隔で分割する．以後この間隔のことをアドレス・インターバルと呼ぶ．縦軸はメモリアクセスのアドレスであり，横軸はクロック・サイクルである．各点はメモリアクセスを表している．図 1 の STEP1 で，各アドレス・インターバルにおけるメモリアクセスの数をカウントし，その値を各アドレス・インターバルに対応する表に格納する．たとえば，最上位のアドレス・インターバルにおいてメモリアクセス数は点線の円で囲んだ 2 つであるため，表の最上位の要素に 2 を格納する．STEP2 では，STEP1 で作成した表に特徴抽出精度の向上を目的として以下の 3 つの要素を追加する．

（1） アクセス時間の平均値：各サブ・トレースの開始時刻を 0 として，各メモリアクセスの時間から平均値を算出する．サブ・トレースにおけるメモリアクセス数を N，各メモリアクセスの時間を x_i とすると，平均値 $\bar{x}$ は式 (1) によって求められる．

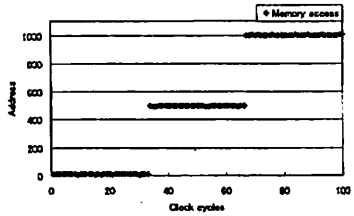
$$\bar{x} = \frac{\sum x_i}{N} \tag{1}$$

（2） アクセス時間の標準偏差値：各サブ・トレースの開始時刻を 0 として，各メモリアクセスの時間から標準偏差値 σ を算出する．これは式 (2) によって求められる．

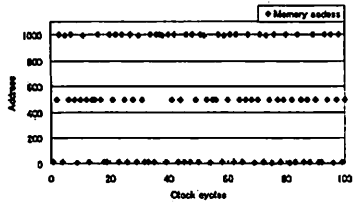
$$\sigma = \sqrt{\frac{\sum (x_i - \bar{x})}{N}} \tag{2}$$

（3） アドレス間距離：時間的に連続したメモリアクセスにおけるアドレスの絶対差分の総和によって定義される．

アクセス時間の平均値および標準偏差値を導入することで，サブ・トレースにおけるメモリアクセスの時間的な偏りがわかる．アドレス間距離の必要性について，図 2 の例を用いて説明する．図 2 の縦軸はアドレス，横軸は時間であり，各点はメモリアクセスの時刻とそのアドレスをもとにプロットしている．図 2(a) はあるアドレス付近に時間的に連続してアクセスが生じており，図 2(b) は離散的にアクセスが生じている．この違いがあるにも関わらず，図 2(a)，(b) ともにメモリアクセスの数，平均値ならびに標準偏差値は同じ値になる．そこで，これらの違いを表す指標としてアドレス間距離が必要となる．時間的に



(a) 連続したメモリアクセス



(b) 離散的なメモリアクセス

図 2　アドレス間距離

連続したメモリアクセスが生じた場合，アドレス間距離は小さい．一方，アクセスが分散した場合は大きな値を示す．つまり，図 2(a) におけるアドレス間距離は図 3(b) のそれよりも小さい．

このようにして得られたベクトルを以後特徴ベクトルと呼ぶ．すべてのサブ・トレースに対して特徴ベクトルを求める．

3.3　小規模ベンチマーク・トレースの生成

サブ・トレースの特徴が同じであれば，そのシミュレーション結果は近いと考えられる．そこで，3.2 で述べた手法によって抽出された特徴ベクトルの類似性に基づいて，サブ・トレースをクラスタリグする．本手法では，ベクトルの数およびベクトルの次元が高いことから，たとえば K-平均法 [10] といった比較的短時間で実行可能なアルゴリズムが適している．各クラスタには同じ特徴のサブ・トレースが属しているため，これらの中からどれをシミュレーションの対象としても結果は同じ，もしくは近くなることが予想される．したがって，各クラスタから 1 つのサブ・トレースをランダムに選出し，シミュレーションの対象として選出する．

代表サブ・トレースのみを用いたキャッシュ・シミュレーションでは，それ以前のトレースを実行していないためコールド・スタートの影響を受けるという問題が生じる．この問題を解決するために，各代表サブ・トレースの直前のトレースをウォームアップ・トレースとして用いる．以後，各クラスタの代表サブ・トレースと，それに対応するウォームアップ・トレースの集合を小規模ベンチマーク・トレースと呼ぶ．

各クラスタに属するサブ・トレースの数が多い程，フル・トレースによるシミュレーション結果に大きな影響を与えていると考えられる．したがって，各代表サブ・トレースのシミュレーション結果に，クラスタに属するサブ・トレースの数を重みとして掛ける．重み付けした結果を合計することで，フル・トレースのシミュレーション結果を予測する．小規模ベンチマーク・トレースを用いた場合のキャッシュ・ミス率 $miss\_rate$ は式 (3) で求めることができる．

表 1 パラメータ等

サブ・トレースサイズ (クロック・サイクル)		10,000
アドレス・インターバル	命令キャッシュ	1,024
	データキャッシュ	8,192
クラスタリング・アルゴリズム		K-平均法
クラスタ数		500

$$miss\_rate = \frac{\sum_{i=1}^{k}(miss_i \times weight_i)}{access} \times 100 \qquad (3)$$

ここで，クラスタ i における代表サブ・トレースの総メモリアクセスを $access$，ミス数を $miss_i$，クラスタに属するサブ・トレース数を $weight_i$ とする．k はクラスタ数である．

4. 評　価

4.1 評価環境

3. で生成した小規模ベンチマーク・トレースの有効性を評価する．ここでは命令およびデータの L1 キャッシュを対象とし，ミス率を測定する．フル・トレースと小規模ベンチマーク・トレースのシミュレーション結果を比較し，トレース・サイズ削減率およびキャッシュ・ミス率の予測精度を求める．予測精度を表す指標は，それぞれのキャッシュ・ミス率の差（パーセンテージ・ポイント）とする．先行研究の SimPoint との定量的比較を行い，本手法の有効性を明らかにする．

マイクロプロセッサ・シミュレータである SimpleScalar3.0 [11] で MPEG2 のデコードプログラム [12] と SPEC2000 ベンチマーク [13] を実行し，フル・トレースを採取した．さらに SimPoint と比較するため，そのアルゴリズムを実装したソフトウエア SimPoint3.1 [14], [15] を使用してトレースを採取した．SimPoint3.1 において，インターバルの値は高速かつ高精度な結果を示す 10M 命令とした [15]．このときの命令キャッシュの構成は，キャッシュ・サイズ 16KB，ブロック・サイズ 32B，ウエイ数 1 である．また，データキャッシュの構成は，キャッシュ・サイズ 16KB，ブロック・サイズ 32B，ウエイ数 4 とした．MPEG2 デコードプログラムでは carphone および foreman を入力データとして使用した．それぞれの画像サイズは QCIF で，フレーム数は 150 である．SPEC2000 ベンチマークは 175.vpr および 176.gcc の 2 つの整数プログラムと，浮動小数点プログラムである 183.equake を使用した．また，SPEC2000 の入力はいずれも test を用いた．

小規模ベンチマーク・トレースの生成において，各種パラメータを表 1 のように設定した．サブ・トレースサイズは，小さい程特徴抽出精度の向上が見込まれる．しかしながら，代表サブ・トレース数が増加することから，クラスタリングに長時間を要することになる．これらのトレードオフを考慮して，サブ・トレースサイズを決定した．アドレス・インターバルの値をキャッシュのブロック・サイズに合わせることが考えられる．しかしながら，小規模ベンチマーク・トレースは異なるメモリアーキテクチャでも使用するため，それに依存する値に設定することはできない．アドレス・インターバルの値を変更してすべてのプログラムで実験を行った結果，精度に影響を与えることは明らかになったが，その差は小さいことがわかった．アドレス・インターバルの値を小さくすると，特徴ベクトルの次元が高くなりクラスタリング時間が長くなる．したがって，大幅な精度の向上は得られないにも関わらず，クラスタリングに長時間を要することになることから，アドレス・インターバルを必要以上に小さな値に設定しても利点が少ないと判断した．シミュレーション精度と，現実的な時間で実験を行うという点を考慮して，本評価においては表 1 の値を用いて議論する．

表 2 SimPoint と同サイズとした際のクラスタ数

ベンチマーク		命令キャッシュ	データキャッシュ
MPEG2	carphone	1,740	360
	foreman	1,200	250
SPEC2000	175.vpr	4,260	1,270
	176.gcc	2,350	850
	183.equake	3,600	1,360

クラスタリング・アルゴリズムは，クラスタリング精度も高く比較的短時間で実行できる K-平均法 [10] を用いた．K-平均法をサポートしているソフトウエア Cluster3.0 [16] を使用して実験を行った．クラスタ数が少ない場合，特徴の異なるサブ・トレースが同じクラスタに属する可能性が高くなる．逆にクラスタ数を多くするとその可能性は低くなるが，シミュレーション対象となるトレース・サイズが増加し，削減率が低下する．小規模ベンチマーク・トレースのメモリアクセス数 は以下の式で見積もることができる．

$$S = \sum_{i=1}^{k}(N_i \times W) \qquad (4)$$

ここで k はクラスタ数，N_i はクラスタ i における代表サブ・トレースのメモリアクセス数，W はウォームアップ・トレースのメモリアクセス数である．N_i はプログラムを実行した時点で決定するため，変更できるパラメータは k ならびに W である．k および W が大きい程シミュレーション精度が高くなると予想される．しかしながら，小規模ベンチマーク・トレースのメモリアクセス数が増加するというトレードオフの関係にある．最適なクラスタ数はプログラムによって異なることから一意に決定できない．本稿における評価では，様々なクラスタ数を用いてすべてのプログラムに対して実験を行った結果，すべてのプログラムにおいて 97%以上の削減率の達成できるクラスタ数 500 を用いて議論をすすめる．クラスタ数と同様に，ウォームアップ・トレースサイズもトレードオフの関係にある．目標のトレース・サイズ削減率を達成するためにウォームアップ・トレースサイズは 16,000 とした．

代表サブ・トレースをランダムに選出するため，シミュレーション結果がばらつくことが予想される．したがって，1 つのベンチマークに対し，各クラスタから代表サブ・トレースをランダムに選出する試行を 10 回行い，10 個の異なる小規模ベンチマーク・トレースを生成した．4.2 および 4.3 で示す結果はすべて 10 回シミュレーションした結果の平均値である．

評価の対象とするキャッシュ構成は，命令キャッシュが，キャッ

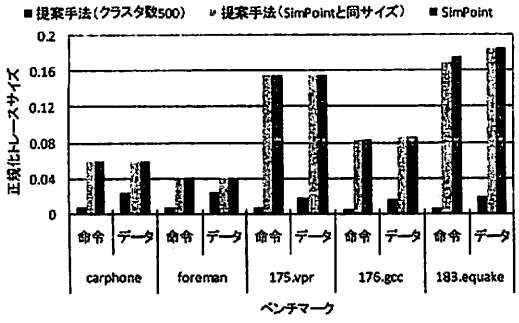


図 3 SimPoint とのトレース・サイズの比較

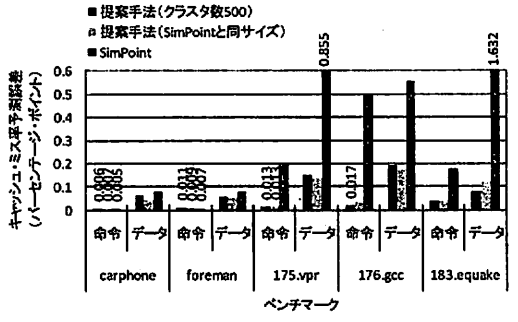


図 4 SimPoint とのキャッシュ・ミス率予測誤差の比較

シュ・サイズ 4KB，ブロック・サイズ 64B，ウエイ数 1 とした．また，データキャッシュの構成は，キャッシュ・サイズ 4KB，ブロック・サイズ 64B，ウエイ数 4 である．クラスタ数 500，ウォームアップ・トレースサイズ 16,000 とした提案手法に加えて，提案手法による小規模ベンチマーク・トレースのトレース・サイズを SimPoint によって得られるトレース・サイズと同一にした際の予測誤差についても比較した．すべてのベンチマークにおいてウォームアップ・トレースサイズを 64,000 とし，クラスタ数によって SimPoint のトレースサイズと同一になるよう調整した．このときの本手法のクラスタ数を表 2 に示す．

4.2 評価結果（SimPoint との比較）

図 3 に，フル・トレースのトレース・サイズを 1 として正規化した場合の本手法のトレース・サイズを示す．左の棒グラフから順に，クラスタ数 500 の提案手法，SimPoint と同一トレース・サイズの提案手法，SimPoint の結果である．本手法によるトレース・サイズは SimPoint の平均 18.3%であった．この結果から，SimPoint よりも高速に実行できることがわかる．

図 4 にキャッシュ・ミス率の予測誤差の比較結果を示す．縦軸はフル・トレースによって得られたキャッシュ・ミス率と本手法によって得られたそれとのポイント差であり，横軸はベンチマークである．carphone および foreman の命令キャッシュにおいては SimPoin よりも精度が低いが，その差はそれぞれ 0.001 および 0.004 パーセンテージ・ポイントと極めて小さい．それ以外では命令，データキャッシュ共に SimPoint よりもシミュレーション精度が高いことがわかる．トレースサイズを SimPoint と同一にした提案手法ではクラスタ数 500 と比べて，carphone や 175.vpr および 176.gcc のデータキャッシュにおいて精度の向上がみられる．carphone および foreman のデータキャッシュはクラスタ数が減少しているが，精度が向上している．これは，ウォームアップ・トレースが増加によるものと考えられる．しかしながら，183.equake のデータキャッシュや 176.gcc の命令キャッシュのように精度の低下が生じる場合があることがわかった．この原因については明らかになっていないが，メモリアクセスの特徴抽出の精度に問題があることなどが考えられる．

表 3 キャッシュ構成 (キャッシュサイズ KB/ウエイ数 W)

命令	データ		
32KB/1W	32KB/1W	32KB/2W	32KB/4W
16KB/1W	16KB/1W	16KB/2W	16KB/4W
8KB/1W	8KB/1W	8KB/2W	8KB/4W
4KB/1W	4KB/1W	4KB/2W	4KB/4W

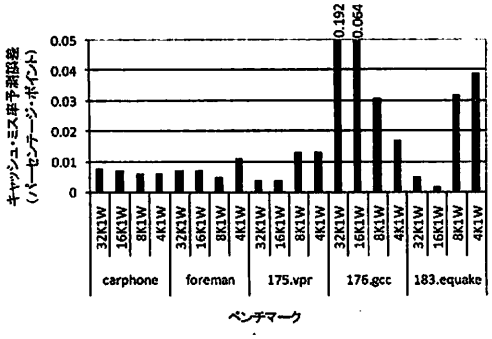


図 5 命令キャッシュにおけるキャッシュ・ミス率予測誤差

これらすべてのベンチマークにおいて，提案手法は SimPoint よりもトレース・サイズが小さく，多くの場合において精度が高いことがわかる．SimPoint より精度が低い場合でも，その差は極めて小さいことから，本手法は SimPoint よりも高速かつ高精度と言える．

4.3 評価結果（異なるキャッシュ構成への適用）

提案手法がフル・トレース採取時と異なるキャッシュ構成において有効であることを調査するため，表 3 に示すキャッシュ構成について評価した．なお，これらのブロック・サイズはすべて 64B である．

命令キャッシュのミス率の平均予測誤差を図 5 に示す．図 5 の縦軸がフル・トレースによって得られたミス率と本手法によって得られたそれとのポイント差で，横軸はベンチマークである．図 5 における予測誤差の平均は 0.024 パーセンテージ・ポイントであった．次にデータキャッシュにおける，予測誤差を図 6 にに示す．図 6 において，予測誤差は平均 0.143 パーセンテージ・ポイントであった．

命令およびデータキャッシュ共に，carphone と foreman は他のベンチマーク・プログラムと比較して，予測誤差が小さいことが分かる．MPEG2 は同じ処理を一定周期で繰り返すこと

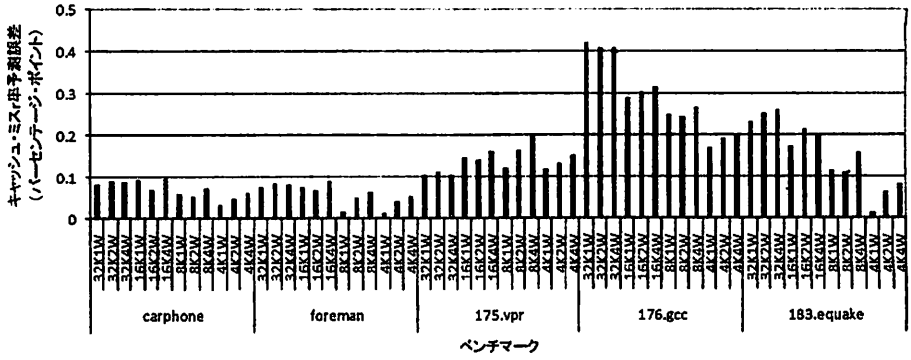


図 6 データキャッシュにおけるキャッシュ・ミス率予測誤差

から，クラスタにおけるサブ・トレースの特徴の類似度が高いことが原因と考えられる．つまり，クラスタ内に特徴が異なるサブ・トレースが少ないため，予測精度が高い．一方，176.gcc は比較的誤差が大きいことが分かる．図 4 の結果から判断して，クラスタ数やウォームアップ・トレースサイズの増加によって大幅な精度の改善が期待できるとは考え難い．したがって，シミュレーション精度を向上させるには，サブ・トレースサイズを小さくするなど特徴抽出精度の向上による方策を検討する必要がある．

5. おわりに

本稿では，高速かつ正確なキャッシュシミュレーション法について述べ，先行研究と定量的な比較を行い本手法の有効性を評価した．プログラムの実行時に得られるフル・トレースから，メモリアクセスの特徴に基づいてより小規模なベンチマーク・トレースを生成した．先行研究と比較して，トレース・サイズは平均 81.7%削減され，キャッシュ・ミス率の予測精度は平均 34.6%高いことが明らかになった．フル・トレースの採取時と異なる様々なキャッシュ構成におけるシミュレーションの結果，トレース・サイズを平均 98.6%削減したにも関わらず，キャッシュ・ミス率の予測誤差は平均 0.083 パーセンテージ・ポイントであった．これらの結果から，本手法は高速かつ正確なキャッシュシミュレーション法であり，キャッシュの設計空間を効率的に探索可能な手法であると言える．

今後はキャッシュ以外のメモリ・アーキテクチャ評価に適用し，有効性を評価する予定である．

謝辞 本論文をまとめるにあたり，共にご討論頂いた九州大学の安浦・村上・松永・井上研究室の皆様に感謝致します．なお，本研究は一部、松下電器産業株式会社との共同研究ならびに科学研究費補助金 (若手研究 A:課題番号 17680005) による．

文 献

[1] 小野貴継，井上弘士，村上和彰："メモリ・アクセスの特徴を活用した高速かつ正確なメモリアーキテクチャ・シミュレーション法"，第 5 回先進的計算基盤システムシンポジウム SACSIS (2007).

[2] 高崎透，中田尚，津邑公暁，中島浩："時間軸分割並列化による高速マイクロプロセッサシミュレーション"，情報処理学会論文誌コンピューティングシステム，46，12，pp. 84-97 (2005).

[3] J. J. Yi and D. J. Lilja: "Simulation of computer architectures: Simulators, benchmarks, methodologies, and recommendations.", IEEE Trans. Computers, 55, 3, pp. 268-280 (2006).

[4] A. J. KleinOsowski and D. J. Lilja: "Minnespec: A new spec benchmark workload for simulation-based computer architecture research.", Computer Architecture Letters, 1, (2002).

[5] J. Robert H. Bell and L. K. John: "The case for automatic synthesis of miniature benchmarks", Workshop on Modeling, Benchmarking and Simulation, Wisconsin, Madison, pp. 88-97 (2005).

[6] J. Robert H. Bell and L. K. John: "Improved automatic testcase synthesis for performance model validation", International Conference on Supercomputing, pp. 111-120 (2005).

[7] R. E. Wunderlich, T. F. Wenisch, B. Falsafi and J. C. Hoe: "Smarts: Accelerating microarchitecture simulation via rigorous statistical sampling.", International Symposium on Computer Architecture, pp. 84-95 (2003).

[8] T. Sherwood, E. Perelman, G. Hamerly and B. Calder: "Automatically characterizing large scale program behavior", International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 45-57 (2002).

[9] T. Sherwood, E. Perelman and B. Calder: "Basic block distribution analysis to find periodic behavior and simulation points in applications", International Conference on Parallel Architectures and Compilation Techniques, pp. 3-14 (2001).

[10] 麻生英樹，津田宏治，村田昇："パターン認識と学習の統計学"，岩波書店 (2006).

[11] SimpleScalar Simulation Tools for Microprocessor and System Evaluation: "http://www.simplescalar.org/".

[12] MPEG Software Simulation Group: "http://www.mpeg.org/mpeg/mssg/".

[13] SPEC: "http://www.specbench.org/".

[14] SimPoint3.1: "http://www.cse.ucsd.edu/ calder /simpoint/".

[15] G. Hamerly, E. Perelman, J. Lau and B. Calder: "Simpoint 3.0: Faster and more flexible program analysis", Workshop on Modeling, Benchmarking and Simulation (2005).

[16] Cluster3.0: "http://bonsai.ims.u-tokyo.ac.jp/ mdehoon /software/cluster/software.htm#source".