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

小野, 貴継
九州大学大学院システム情報科学府

井上, 弘士
九州大学大学院システム情報科学研究院

村上, 和彰
九州大学大学院システム情報科学研究院

出版情報：Proceedings of Symposium on Advanced Computing Systems and Infrastructures. 5, pp.19-26, 2007-05-23
バージョン：
権利関係：

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

小野貴継[†]　井上弘士[††]　村上和彰[††]

本稿では，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を提案する．一般に，メモリアーキテクチャの評価には，メモリ参照のアドレス・トレースに基づいたシミュレーションを行う．しかしながら，評価対象の増加により，評価時間が長くなる傾向にある．トレースに基づくシミュレーションにおいて，1 回あたりのシミュレーション時間はアドレス・トレースの削減によって短縮できるが，精度が低下するという問題がある．そこで，本手法はメモリアクセスの特徴を活用して高い精度維持しつつトレース・サイズを削減し，シミュレーション時間の短縮を実現する．キャッシュ性能測定に基づく評価実験の結果，本手法はトレース・サイズを平均 98.8%削減し，そのときのキャッシュ・ミス率の予測誤差は平均 0.067 パーセンテージ・ポイントであった．

Fast, Accurate Memory Architecture Simulation Technique Using Memory Access Characteristics

Takatsugu Ono,[†] Koji Inoue[††] and Kazuaki Murakami[††]

This paper proposes a fast, accurate memory architecture simulation technique. To design memory architecture, the first steps commonly involve using trace-driven simulation. However, expanding the design space makes the evaluation time increase. A Fast simulation is achieved by a trace size reduction, but it reduces the simulation accuracy. Our approach can reduce the simulation time while maintaining the accuracy of the simulation results. In order to evaluate validity of proposed technique, we measured the cache miss ratio. In our evaluation, the proposed technique reduces the trace size 98.8% and cache miss ratio differs from 0.067 percentage point on an average.

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．実際，現在でも CMP を評価対象としてトレース・ドリブン方式が広く利用されている[4),16)]．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・ドリブン方式シミュレーション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．

そこで本稿では，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する．また，アプリケーションとして MPEG2 お

† 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University
†† 九州大学大学院システム情報科学研究院
Faculty of Information Science and Electrical Engineering, Kyushu University

よび SPEC2000 ベンチマークを用いた定量的評価を行い有効性を明らかにする．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・トレースを取得する．そして，それを小規模なアドレス・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．なお，このようにして生成した小規模なトレースによるシミュレーションはキャッシュに限らず，メモリアーキテクチャの評価に幅広く活用できる．

以下，第 2 節でこれまでに行われてきた関連研究について述べる．第 3 節では提案手法について述べ，第 4 節でその有効性を調査する．第 5 節で簡単にまとめ，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[15]．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[18]．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 が挙げられる[5]．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 R.Bell らの手法が挙げられる[8),9]．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[17] および SimPoint[10),11] 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．SimPoint を用いてアドレス・トレースを採取する方法も提案されている[6]．

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，提案手法におけるメモリアーキテクチャ・シミュレーションの方がより高い精度を達成できる．実際第 4.2.3 節での定量的評価において，提案手法の方が SimPoint よりも高速かつ高精度であることを示している．

3. メモリアーキテクチャシミュレーション法

3.1 用語の定義

本稿で使用する用語について定義する．

- フル・トレース：アプリケーション・プログラムの実行開始から終了までのメモリアクセスの時刻とアドレスの集合
- サブ・トレース：フル・トレースの部分集合であり，互いに素である．

3.2 提案手法の概要

本手法の概要を**図 1** に示す．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・トレースを第 3.3 節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．なぜなら，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，

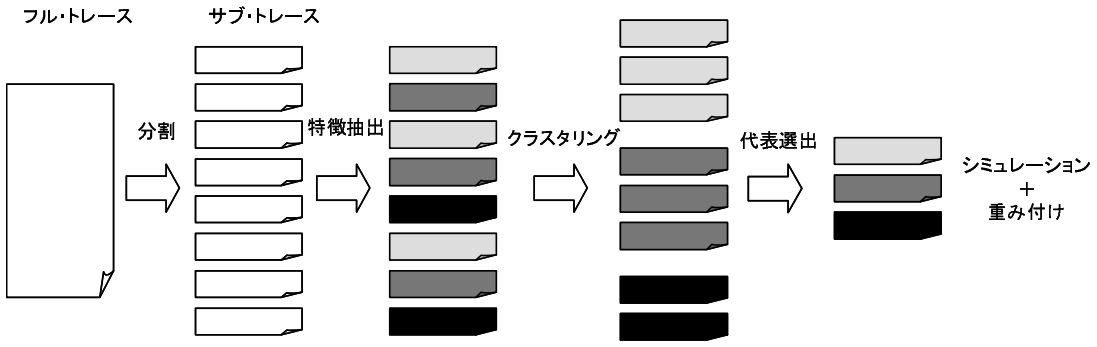


図 1 提案手法の概要

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．

3.3 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と予想される．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．命令単位で分割した場合，各命令間の時間はすべて一定とみなされることから，メモリアクセスがある時間に集中的に発生している場合や，逆に発生していない場合などの時間的特徴は抽出できない．一方，時間単位で分割すると時間的特徴は抽出可能である．メモリシステムのシミュレーションにおいて，この時間的な特徴は重要である．たとえば，CMP や SoC といった他コアや IP とバスを共有しているアーキテクチャにおいて，バスの性能をシミュレーションする状況を考える．ある時間にバスへのアクセスが集中すると競合が発生する可能性が高くなる．命令単位で分割した場合はメモリアクセスの時間的な特徴を考慮しないため，バスのシミュレーション精度は低下すると考えられる．時間単位で分割すると時間的な特徴を抽出できることから，精度の低下は前者ほど生じない．本稿では評価対象をキャッシュとしているが，バスの性能評価などへの適用を考え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 2** 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 2 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 つなので，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．

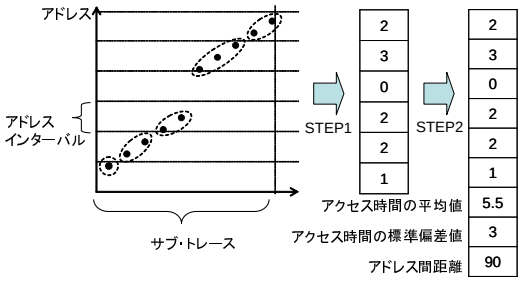


図 2 特徴抽出手順

(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 3** の例を用いて説明する．図 3 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 3(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 3(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 3(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 3(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．

3.4 クラスタリングと代表サブ・トレースの選出

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，第 3.3 節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．クラスタリング方法は既存の手法が適用できる．本手法において，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[1] 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に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．

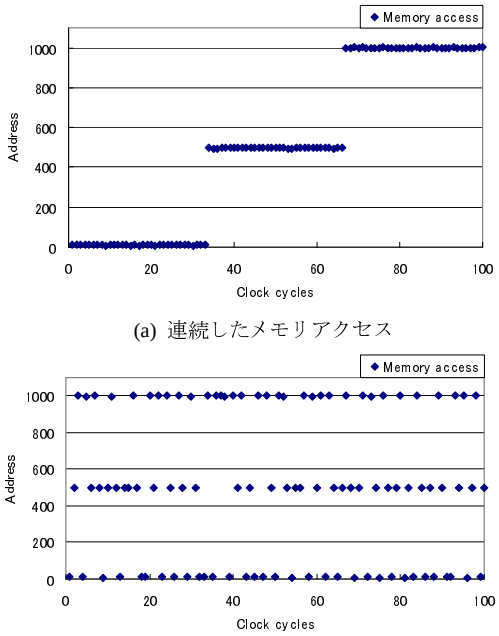
3.5 代表サブ・トレースによるシミュレーション

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．

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

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．



(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 3 アドレス間距離

4. 評　　価

4.1 評 価 環 境

第 3 節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 L1 キャッシュを対象とし，命令およびデータキャッシュのミス率を測定する．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．また，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[12] で MPEG2 のデコードプログラム[7] と SPEC2000 ベンチマーク[14] を実行し，フル・ト

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-means
クラスタ数		500

表 2 キャッシュ構成 (キャッシュサイズ KB/ウエイ数 W)

Instruction	Data		
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

レースを採取した．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 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 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 akiyo，carphone ならびに foreman の 3 つの動画像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 164.gzip，175.vpr，176.gcc ならびに 197.parser の 4 つの整数プログラムと，177.mesa および 183.equake の 2 つの浮動小数点プログラムを使用した．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．

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[1] 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[2] 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．

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

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．第 4.2 節で示す結果はすべて 10 回のシミュレーションした結果の平均値である．また，提案手法が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 2** 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．

4.2 評 価 結 果

4.2.1 トレース・サイズ削減率

命令，データキャッシュを対象としたサブ・トレースの削減率の平均をそれぞれ**図 4** および**図 5** に示す．縦軸がフル・トレースに対する小規模ベンチマーク・トレースの削減率であり，横軸はベンチマークである．命令キャッシュの削減率は平均 99.38%，データキャッシュは平均 98.22%であった．また，クラスタリングに要した時間は，フル・トレースによるシミュレーション時間の約 6 倍程度であった．第 1 節でも述べたように，小規模ベンチマーク・トレースの目的は，アーキ

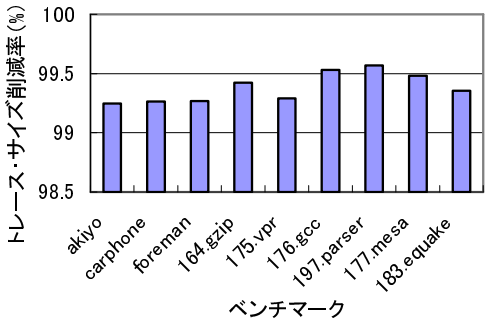


図 4 命令キャッシュにおけるトレース・サイズ削減率

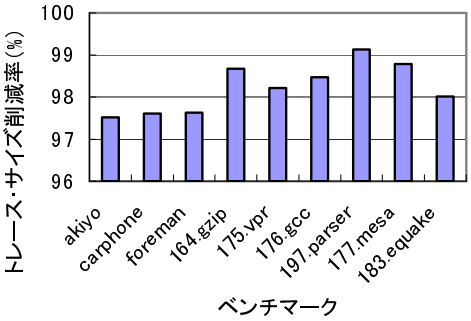


図 5 データキャッシュにおけるトレース・サイズ削減率

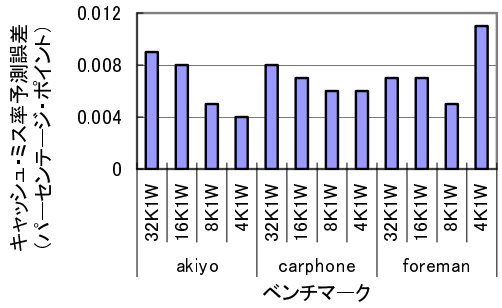


図 6 命令キャッシュにおける MPEG2 の
キャッシュ・ミス率予測誤差

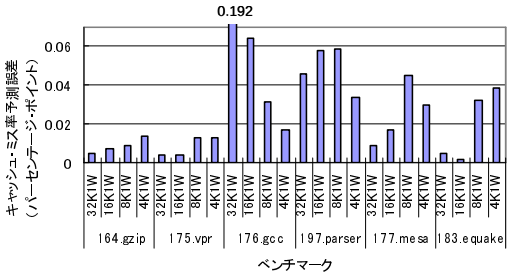


図 7 命令キャッシュにおける SPEC2000 の
キャッシュ・ミス率予測誤差

テクチャに依存しないトレースを一度生成するだけで，異なるアーキテクチャでも再利用することである．したがって，クラスタリングに比較的長い時間が必要であっても，様々なキャッシュ構成でシミュレーションする場合さほど問題にならない．また，ランダム・プロジェクション[11] といった特徴ベクトルの次元を縮小する手法などの適用によりクラスタリング時間を大幅に短縮することは可能である．

4.2.2 キャッシュ・ミス率の予測誤差

MPEG2 における命令キャッシュのミス率の平均予測誤差を**図 6** に，SPEC2000 の場合を**図 7** に示す．図 6 および図 7 共に，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 6 における予測誤差の平均は 0.0069 パーセンテージ・ポイントと極めて小さい．MPEG2 は同じ処理を一定周期で繰り返すことから，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．図 7 の平均予測誤差は約 0.031 パーセンテージ・ポイントであった．多くのベンチマークにおいて 0.05 パーセンテージ・ポイント以下と MPEG2 より劣るものの，小さな値を示している．

次にデータキャッシュにおける，MPEG2 のミス率の予測誤差を**図 8** に，SPEC2000 の場合を**図 9** に示す．図 8 および図 9 共に，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 8 では，予測誤差の平均は約 0.061 パーセンテージ・ポイントと小さい値を示した．このことから，同一アプリケーションにおいて入力データが異なっても予測精度が高いことがわかる．図 9 において，予測誤差の平均は約 0.171 パーセンテージ・ポイントであった．

4.2.3 関連研究との比較

本節では，第 2 節で述べた SimPoint[10),11)] との定量的比較を行い，本手法の有効性を明らかにする．SimPoint3.1[3),13)] を用いて採取したアドレス・トレースと，本手法によるシミュレーション結果およびトレース・サイズ削減率の比較を行った．本手法の各種パラメータは第 4.1 節で述べた表 1 と同じである．キャッシュ構成は，命令キャッシュが，キャッシュ・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 とした．対象とするベンチマークとして，MPEG2 の foreman,

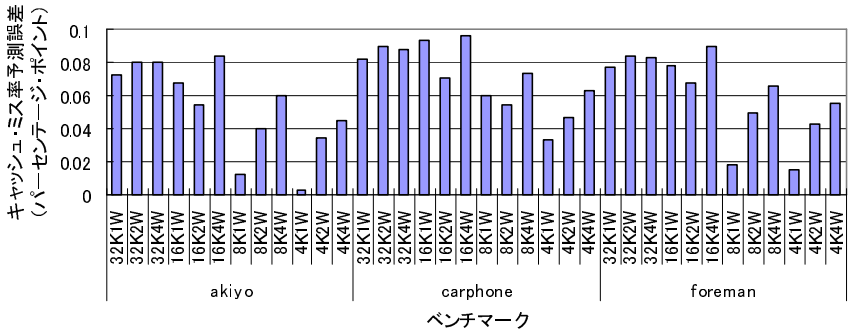


図 8　データキャッシュにおける MPEG2 のキャッシュ・ミス率予測誤差

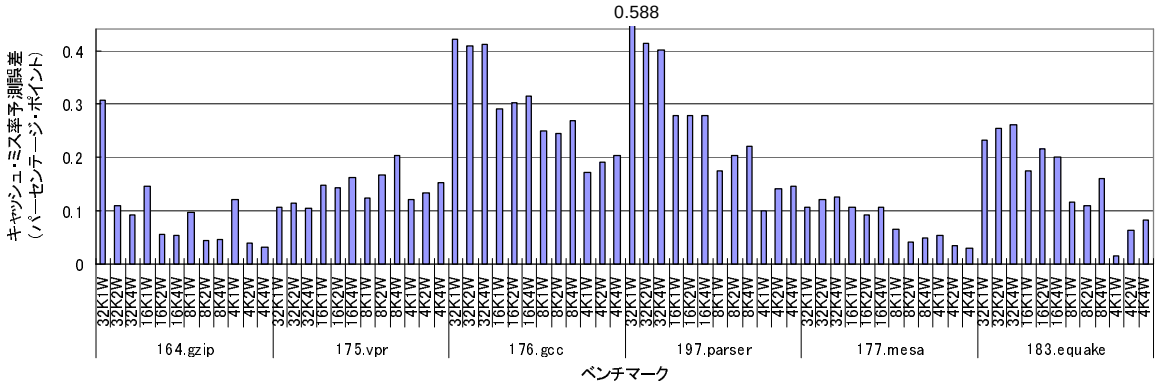


図 9　データキャッシュにおける SPEC2000 のキャッシュ・ミス率予測誤差

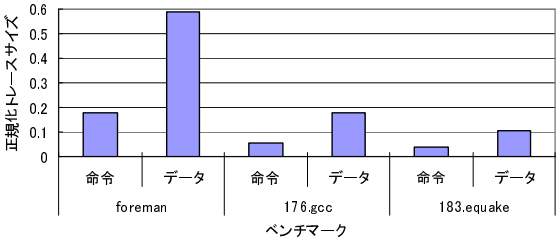


図 10　SimPoint とのトレース・サイズの比較

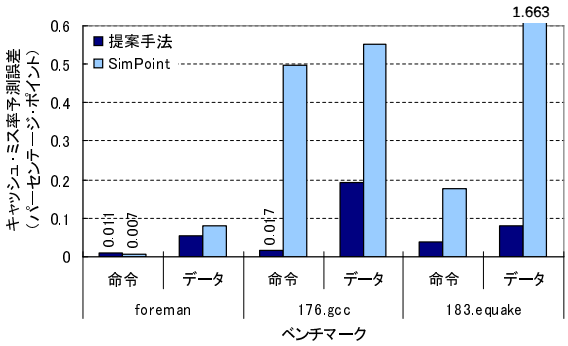


図 11　SimPoint とのキャッシュ・ミス率予測誤差の比較

SPEC2000 の 176.gcc ならびに 183.equake を選択した．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[3]．**図 10** に，SimPoint によって取得してアドレス・トレースを 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．

本手法によるトレース・サイズは SimPoint の約 10%から 60%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 11** 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．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

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．

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．

5. おわりに

本稿では，高速かつ高精度なメモリアーキテクチャのシミュレーション手法を提案し，MPEG2 および SPEC2000 ベンチマークを用いた評価実験によって有効性を明らかにした．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．その結果，トレース・サイズの削減率は平均 98.8%，キャッシュ・ミス率の予測誤差は平均 0.067 パーセンテージ・ポイントであった．したがって，本手法は短時間で高精度のメモリアーキテクチャ・シミュレーションが可能であると言える．さらに，小規模ベンチマーク・トレースは生成時と異なるメモリアーキテクチャにおいてもシミュレーション精度が高く，有効であることを示した．また，関連研究との定量的比較を行い，本手法が高精度かつ高速であることを確認した．

今後は小規模ベンチマーク・トレースをバスの性能評価などへ適用し，その有効性を評価する予定である．

謝辞 本研究は一部，科学研究費補助金（学術創成研究費：課題番号 14GS0218，若手研究 A：課題番号 17680005），21 世紀 COE プログラム「システム情報科学での情報基盤システム形成」ならびに松下電器産業株式会社との共同研究による．

参考文献

1) 麻生英樹ほか: パターン認識と学習の統計学, 岩波書店 (2006).
2) Cluster3.0: http://bonsai.ims.u-tokyo.ac.jp/ mdehoon /software/cluster/software.htm#source.
3) Hamerly, G. et al.: SimPoint 3.0: Faster and More Flexible Program Analysis, *Workshop on Modeling, Benchmarking and Simulation*, Wisconsin, Madison (2005).
4) Hsu, L. et al.: Exploring the Cache Design Space for Large Scale CMPs, *SIGARCH Comput. Archit. News*, Vol. 33, No. 4, pp. 24–33 (2005).
5) KleinOsowski, A. J. and Lilja, D. J.: MinneSPEC: A New SPEC Benchmark Workload for Simulation-Based Computer Architecture Research., *Computer Architecture Letters*, Vol. 1 (2002).
6) Laurenzano, M. et al.: Low Cost Trace-driven Memory Simulation Using SimPoint, *SIGARCH Comput. Archit. News*, Vol. 33, No. 5, pp. 81–86 (2005).
7) MPEG Software Simulation Group: http://www.mpeg.org/MPEG/MSSG/.
8) Robert H. Bell, J. and John, L. K.: The Case for Automatic Synthesis of Miniature Benchmarks, *Workshop on Modeling, Benchmarking and Simulation*, Wisconsin, Madison, pp.88–97 (2005).
9) Robert H. Bell, J. and John, L. K.: Improved Automatic Testcase Synthesis for Performance Model Validation, *Proc. 19th annual international conference on Supercomputing*, ACM Press, pp. 111–120 (2005).
10) Sherwood, T. et al.: Basic Block Distribution Analysis to Find Periodic Behavior and Simulation Points in Applications, *Proc. 2001 International Conference on Parallel Architectures and Compilation Techniques*, IEEE Computer Society, pp. 3–14 (2001).
11) Sherwood, T. et al.: Automatically Characterizing Large Scale Program Behavior, *Proc. 10th international conference on Architectural support for programming languages and operating systems*, ACM Press, pp. 45–57 (2002).
12) SimpleScalar Simulation Tools for Microprocessor and System Evaluation: http://www.simplescalar.org/.
13) SimPoint3.1: http://www.cse.ucsd.edu/ calder/simpoint/.
14) SPEC: http://www.specbench.org/.
15) 高崎透ほか: 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, Vol. 46, No. 12, pp. 84–97 (2005).
16) Vera, J. et al.: A Novel Evaluation Methodology to Obtain Fair Measurements in Multithreaded Architectures, *Workshop on Modeling, Benchmarking and Simulation*, Boston, Massachusetts (2006).
17) Wunderlich, R. E. et al.: SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling., *Proc. 30th International Symposium on Computer Architecture*, IEEE Computer Society, pp. 84–95 (2003).
18) Yi, J. J. and Lilja, D. J.: Simulation of Computer Architectures: Simulators, Benchmarks, Methodologies, and Recommendations., *IEEE Trans. Computers*, Vol. 55, No. 3, pp. 268–280 (2006).