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

2009-ARC-181／2009-EMB-11 (18)
2009/1/14

シミュレーション結果の再利用に基づくキャッシュ・ミス率予測法の提案

小野 貴継† 井上 弘士†† 村上 和彰†† 甲斐 康司†††

† 九州大学大学院システム情報科学府 〒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
††† パナソニック（株）プラットフォーム開発センター 〒814–0001 福岡県福岡市早良区百道浜 2-4-16
E-mail: †ono@c.csce.kyushu-u.ac.jp, ††{inoue,murakami}@i.kyushu-u.ac.jp, †††kai.kj@jp.panasonic.com

あらまし 本稿ではシミュレーション結果の再利用することによってキャッシュ・ミス率を予測する手法を提案する．キャッシュ・アーキテクチャの決定には多くのベンチマークおよびその入力データを対象にシミュレーションする必要があるため，評価に長時間を要する．同一プログラムを異なる入力データによって実行する場合でも，類似したメモリアクセスパタンが出現する可能性がある．メモリアクセスパタンの類似度が高い場合はキャッシュ・ミス率もまた近い値になることが予想される．従来手法では，類似したメモリアクセスパタンが出現した場合でもシミュレーションする必要があった．そこで本稿では，ある入力データを用いてプログラムを実行した際のメモリアクセスの特徴とシミュレーション結果を再利用することで，異なる入力データにおけるキャッシュ・ミス率を短時間で予測する．評価の結果，多くのプログラムにおいて高精度かつ短時間で予測可能であることを確認した．
キーワード 性能評価，シミュレーション，メモリ・システム

Predicting Cache Miss Rates via Simulation Results Reuse

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

† Graduate School of Information Science and Electrical Engineering, Kyushu University
Motooka 744, Nishi-ku, Fukuoka, 819–0395, Japan
†† Faculty of Information Science and Electrical Engineering, Kyushu University
Motooka 744, Nishi-ku, Fukuoka, 819–0395, Japan
††† Panasonic Corporation 2-4-16 Momochihama, Sawara-ku, Fukuoka City 814–0001, Japan
E-mail: †ono@c.csce.kyushu-u.ac.jp, ††{inoue,murakami}@i.kyushu-u.ac.jp, †††kai.kj@jp.panasonic.com

Abstract This paper proposes cache miss rate prediction technique reusing simulation results. In order to determine the best cache configuration, designers have to evaluate many cache configurations with many benchmark programs and its input data sets. Similar memory access patterns appear in different input data sets of the same program. We can expect almost the same simulation results if the patterns are similar. However, a conventional approach has to simulate the similar patterns. In this paper, we attempt to predict cache miss rate by means of reusing simulation results and the similarity of memory access patterns. In our evaluation, it is observed that the proposed approach is able to predict the cache miss rate in high accuracy.
Key words Performance evaluation, Simulation, Memory System

1. はじめに

現在，多くの組み込みプロセッサにはキャッシュメモリが搭載されている．オフチップアクセス回数を削減することにより，プロセッサ性能の向上とメモリシステムの低消費電力化を同時に期待できるためである．キャッシュメモリの性能は主にキャッシュ・ミス率とアクセス時間に依存する．したがって，設計者はこれらの値が最小となるようキャッシュの最適な構成を選択しなければならない．

一般に，キャッシュアクセス時間は回路遅延により決定される．したがって，キャッシュ構成と実設計パラメータ値（配線容量やトランジスタ数など）を入手できれば高い精度でキャッ

シュアクセス時間を見積もることが可能である．これに対し，キャッシュ・ミス率はキャッシュ構成のみならず，実行対象となるベンチマーク・プログラムの特性および入力データに強く依存する．そのため，キャッシュ・アーキテクチャの決定には多くのベンチマーク・プログラムおよびその入力データを対象にシミュレーションする必要があり，ひいてはアーキテクチャ決定を長期化させる1つの要因となり得る．

多くのベンチマーク・プログラムやその入力データを対象にシミュレーションする場合，類似したメモリアクセスパタンが異なるプログラムまたは同一プログラム上で異なる入力データを実行する際に出現することがある．特に，後者は同一プログラムであることからその出現頻度は比較的高いと考えられる．メモリアクセスの特徴が類似している場合，シミュレーション結果もまた近い値になると考えられる．従来手法では，類似したメモリアクセスパタンが出現した場合でもシミュレーションする必要があった．

そこで本稿では，キャッシュ構成の最適化を短期間で実現するための手段として，シミュレーション結果の再利用に基づくキャッシュ・ミス率予測技術を提案する．本手法はまず，ある入力データを対象にメモリアクセスパタンを抽出するとともに，複数のキャッシュ構成を対象としてキャッシュ・ミス率を測定する．このとき得られたメモリアクセスパタンとミス率からデータベースを生成する．次に異なる入力データを対象にメモリアクセスパタンを抽出する．データベースに登録されたメモリアクセスパタンとこれとを比較することで，異なる入力データ実行時のキャッシュ・ミス率を予測する．本手法の特徴は，一度のメモリアクセスパタン比較で，データベースに登録された複数のキャッシュアーキテクチャのミス率を予測可能な点にある．これにより，キャッシュ・ミス率を短時間で予測可能となる．評価の結果，多くのベンチマーク・プログラムにおいて高い精度で予測可能であることを確認した．

以下，第2節にて従来方式における問題点および関連研究について述べる．第3節では提案するシミュレーション結果再利用による予測手法について説明する．第4節で提案手法の有効性を評価し，最後に第5節にてまとめと今後の課題について述べる．

2. 従来方式における問題点と関連研究

2.1 従来方式とその問題点

主にキャッシュ・ミス率は，キャッシュサイズ，連想度，ブロックサイズおよびブロック置換アルゴリズムに依存する．そのため，設計者がキャッシュ・アーキテクチャを決定する際，与えられたベンチマーク・プログラムと入力データに対して多くのキャッシュ・シミュレーションを実施しなければならない．たとえば，評価対象となるキャッシュ構成の数を N_{conf}，ベンチマーク・プログラム数を N_{prog}，各プログラムにおいて準備された入力データ数（たとえば，静止画伸長プログラムの場合は入力対象画像数）を N_{input} とする場合，すべてのキャッシュ構成にけるミス率を測定するためには式 (1) で示す評価時間 T_{eva} が必要となる．

$$T_{\mathrm{eva}} \propto T_{\mathrm{sim}} \times N_{\mathrm{conf}} \times N_{\mathrm{prog}} \times N_{\mathrm{input}} \quad (1)$$

ここで，T_{sim} はキャッシュ構成当りの平均シミュレーション時間である．単純にすべての組合わせについてシミュレーションを実施する従来手法ではキャッシュ性能評価に多くの時間を要するという問題が生じる．

2.2 関連研究

Eeckhout らは与えられたプログラムの代表的な入力データを選択する手法を提案している [1]．つまり，式 (1) の N_{input} を削減する手法である．複数の入力データを用いてプログラムを実行し，プログラムの振る舞いを分岐予測の精度やキャッシュ・ミス率など 20 の特徴量によって表わしている．これらの特徴に基づいてクラスタリングし，各クラスタから代表的な入力データを選択する．これにより，精度の低下を抑制しつつ冗長なシミュレーションを回避可能であり，評価時間の短縮を図ることができる．Zhong らはメモリアクセスの特徴に基づいて，異なる入力データサイズのキャッシュ・ミス率を予測する手法を提案している [2]．この手法もまた，N_{input} を削減可能である．メモリアクセスの特徴を Reuse Distance [3] によって抽出し，Zhong らが構築した予測モデルを用いて容量性のミスを予測する手法である．

Mattson らはすべてのキャッシュサイズに対するミス率を一度のシミュレーションによって求める手法を提案している [4]．Hill らは Mattson らの手法を拡張し，一度シミュレーションすることで得られたメモリアクセスのトレースデータから，キャッシュサイズおよび連想度の異なるキャッシュ構成のミス率を見積もる手法を提案している [5]．これらの手法は高速にシミュレーション可能であるが，異なる入力データを対象とする場合は再度トレースデータを取得する必要がある．

Hoste らは，ある計算機上で実行した際の性能とプログラムの振る舞いを用いてデータベースを作成し，異なるマシン上で実行した他のプログラムの振る舞いからその性能を予測している [6]．Hoste らの手法は，予測の準備として特徴量とシミュレーション結果を事前に取得しデータベースを生成するという点で本稿の提案手法と同じである．しかしながら，Hoste らの手法は計算機の相対的な性能予測に限られることに対して，提案手法はキャッシュ・メモリの絶対性能を予測可能な点で異なる．

3. シミュレーション結果再利用による予測手法

3.1 用語の定義

本節では，本稿で使用する用語について定義する．

- 評価対象プログラム：キャッシュ・アーキテクチャの評価に用いるプログラムを意味する．
- 評価対象入力データ：評価対象プログラムの入力データの1つ．評価対象プログラムおよび評価対象入力データを用いてシミュレーションすることで，キャッシュ・アーキテクチャの性能を評価する．
- データベース用入力データ：評価対象プログラムの入力データの1つ．ただし，評価対象入力データとは異なる入力

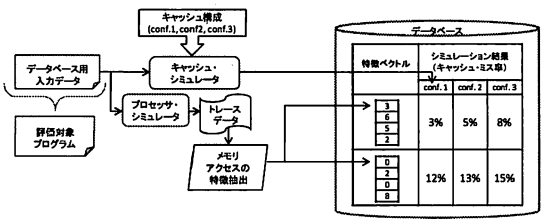

図 1　データベース生成フェーズ

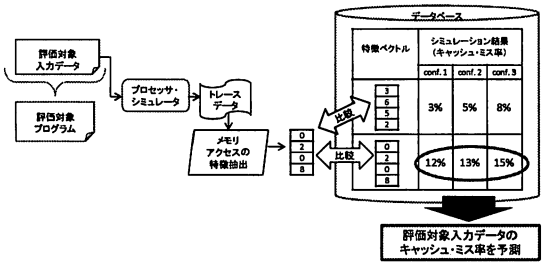

図 2　キャッシュ・ミス率予測フェーズ

データとする．評価対象プログラムおよびデータベース用入力データを用いてシミュレーションし，その結果をデータベースに登録する．

3.2 概　　要

本手法はシミュレーション結果を予測するキャッシュ・ミス率予測フェーズと，その準備のためにメモリアクセスの特徴とシミュレーション結果を取得するデータベース生成フェーズとに大別できる．

- データベース生成フェーズ：データベース生成フェーズの概要を図 1 に示す．評価用プログラムとデータベース用入力データセットを対象に，キャッシュ・シミュレータを用いて複数のキャッシュ構成のミス率を測定する．次に，データベース用入力データによって実行された評価対象プログラムのメモリアクセスに関するトレースデータを取得し，メモリアクセスの特徴を抽出する．ここで，メモリアクセスの特徴はベクトルで表わすこととする．特徴抽出に関する詳細は第 3.3 節で述べる．各キャッシュ構成におけるミス率とメモリアクセスの特徴をデータベースに登録する．
- キャッシュ・ミス率予測フェーズ：キャッシュ・ミス率予測フェーズの概要を図 2 に示す．評価対象入力データを用いて評価対象プログラムを実行し，データベース生成フェーズと同様にメモリアクセスのトレースデータを取得後，特徴を抽出する．得られた特徴とデータベースに登録された特徴とを比較する．最も類似度の高い特徴を持つインターバルのシミュレーション結果を用いて，評価対象入力データによって評価対象プログラムを実行した場合の各キャッシュ構成に対するミス率を予測する．類似度の定義およびキャッシュ・ミス率の予測方法については第 3.5 節で議論する．

3.3 メモリアクセスの特徴抽出

メモリアクセスの時間局所性および空間局所性は，キャッシュ・ミス率と関係が深いことから，ミス率を予測する上で重要な特徴である．この特徴はプログラムの実行とともに変化すると考えられる．そこで，プログラムの実行開始から終了までを一定の間隔で分割し，各区間においてメモリアクセスの特徴を抽出すると効果的であると考えられる．提案手法は一定数のロード・ストア命令が実行される間に，どのアドレス付近にどれだけのメモリアクセスが生じているのか，という情報に基づいてキャッシュ・ミス率の予測を試みる．したがって，プロセッサ・シミュレータ等により取得したロード・ストア命令のトレースデータを一定の命令数によって分割し，各命令インターバルにおけるメモリアクセスの特徴を抽出する．

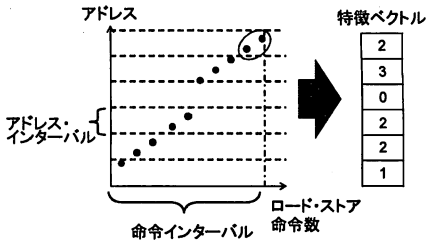

図 3　メモリアクセスの特徴抽出法

図 3 はある命令インターバルにおけるメモリアクセスを表している．縦軸はアドレスであり，横軸はロード・ストア命令数である．空間的局所性を抽出するために命令インターバルのアドレス空間を図 3 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．メモリアクセスの空間局所性を抽出するために各アドレス・インターバルにおけるメモリアクセス数をカウントする．たとえば，図 3 では最上位のアドレス・インターバルにおいてメモリアクセス数は楕円で囲んだ 2 つであることから，表の最上位の要素に 2 を格納する．同様に他のアドレス・インターバル内のメモリアクセス数を調べる．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．すべての命令インターバルを対象に同様の方法で特徴ベクトルを求める．メモリアクセスの特徴はキャッシュ構成に依存しない指標に基づいて抽出する．したがって，各入力データに対して一度だけ特徴抽出作業を行えば良い．

アドレス・インターバルおよび命令インターバルの値が小さいほど，メモリアクセスの特徴抽出精度は向上すると考えられる．アドレス・インターバル値を小さくすると，特徴ベクトルの次元数が増加する．これにより，データベース参照に要する時間が長くなるという問題が生じる．さらに，プログラムによってメモリアクセスの特徴が異なることから，適切なアドレス・インターバルも異なる可能性が高い．したがって，各プログラムにおいて精度との関係を調査した上で，アドレス・インターバルを決定することが望ましい．

3.4 データベースの生成

データベース用入力データによってプログラムを実行した際の，命令インターバルごとのメモリアクセスの特徴およびキャッシュ・ミス率をデータベースに登録する．このとき，より多く

のキャッシュ構成を対象にミス率を求めることによって，キャッシュ・ミス率予測フェーズにおける予測可能なキャッシュ構成が増加する．これにより，一度のデータベース参照でより多くのキャッシュ構成に対する予測が可能になる．

3.5 シミュレーション結果の予測

3.5.1 絶対性能予測

キャッシュ・ミス率（絶対性能）を予測するために，各命令インターバルのメモリアクセスの特徴をデータベースに登録された特徴と比較する．具体的には，特徴ベクトル間のユークリッド距離を算出する．評価対象入力データによってプログラムを実行した際の，命令インターバルの特徴ベクトルを V_{T_t}，データベースに登録された特徴ベクトルを V_{D_t} とするとこれらの特徴ベクトル間の類似度 S_s は式 (2) で表わされる．ここで E は2つの特徴ベクトル間のユークリッド距離を求める関数である．類似度 S_s の値が大きいほど，類似度が高いと判断する．

$$S_s = \frac{1}{E(V_{T_t}, V_{D_t})} \tag{2}$$

予測するキャッシュ・ミス率 *cmr* は式 (3) で求められる．ここで，*p_access* は本手法により予測したアクセス数を，*p_miss* はミス数を表す．また *interval* はすべての命令インターバル数である．絶対性能の予測精度は式 (3) によって求められる予測ミス率と，実際のミス率との差で表すことができる．

$$cmr = \left(\frac{\sum_{i=1}^{\text{interval}} p\_\text{miss}_i}{\sum_{i=1}^{\text{interval}} p\_\text{access}_i} \right) \times 100 \tag{3}$$

3.5.2 相対性能予測

キャッシュ構成間の相対的な性能はアーキテクチャの探索において重要である．式 (3) によって求められる予測ミス率から，各キャッシュ構成間の相対的な性能を予測する．ある入力データを用いた場合，キャッシュ構成 A が最も性能が高く，次に性能が高いキャッシュ構成は B といったように順位を示すことによって相対性能を表すことができる．予測した順位が実際の順位と近いほど，予測精度が高いと言える．

したがって，相対性能の予測精度は式 (4) に示すスピアマン順位相関係数 r_s によって表わすこととする．

$$r_s = 1 - \frac{6 \sum_{i=1}^{c} d_i^2}{c^3 - c} \tag{4}$$

ここで，c は評価対象とするキャッシュ構成数，d_i は予測した順位と実際の順位との差である．r_s の値が 1 に近い程，相対性能の予測精度が高いことになる．逆に-1 は順位がすべて逆であることを意味する．

3.6 予測時間

評価対象プログラム数と評価対象入力データ数がそれぞれ 1 であるとき，従来方式による評価時間 T_{conv} は式 (1) から導くことができ，式 (5) で表わされる．

$$T_{\text{conv}} = T_{\text{sim\_eva}} \times N_{\text{conf}} \tag{5}$$

ただし，T_{sim_eva} は評価対象入力データを用いて評価対象プログラムを実行した際のシミュレーション時間とする．

表 1 キャッシュ構成

ライン サイズ (B)	キャッシュ サイズ (KB)	ウェイ数
16	2	1
32	4	2
64	8	4

提案手法によるキャッシュ・ミス率予測時間 T_r は式 (6) によって求められる．

$$T_r = T_{\text{db}} + T_p \tag{6}$$

T_{db} はデータベース生成フェーズの時間であり，T_p はキャッシュ・ミス率予測フェーズの時間である．T_{db} および T_p は，それぞれ式 (7)，式 (8) で表すことができる．

$$T_{\text{db}} = T_{\text{e\_db}} + T_{\text{sim\_db}} \times N_{\text{conf}} \tag{7}$$

$$T_p = T_{\text{e\_p}} + T_{\text{ref}} \tag{8}$$

ここで，T_{e_db} および T_{e_p} は各フェーズにおけるメモリアクセスの特徴抽出時間，T_{sim_db} はデータベース用入力データを評価対象プログラム上で実行した場合のシミュレーション時間であり，T_{ref} はデータベース参照および予測ミス率算出時間を表す．

従来手法と比較して提案手法の予測時間を短縮する，つまり $T_r < T_{conv}$ が成り立つためには式 (5) および式 (6) より次式を満たす必要があることが分かる．

$$T_{\text{sim\_db}} < T_{\text{sim\_eva}} - \frac{(T_{\text{e\_db}} + T_{\text{e\_p}} + T_{\text{ref}})}{N_{\text{conf}}} \tag{9}$$

ここで，多くのキャッシュ構成を対象に評価する場合を考えると，高速な予測を実現するためには式 (10) を満たすデータベース用入力データを選択しなければならない．

$$T_{\text{sim\_db}} < T_{\text{sim\_eva}} \tag{10}$$

4. 評　　価

4.1 評価環境

提案手法による L1 データキャッシュの絶対性能，相対性能の予測精度および予測時間について評価する．ベンチマーク・プログラムは MiBench [7] から 22 種類を用いる．MiBench の入力データには *small* と *large* の 2 種類が用意されている．*small* と *large* によるシミュレーションは式 (10) を満たすことから [7]，*small* をデータベース用入力データとして，*large* を評価対象入力データとして用いる．つまり，*small* のメモリアクセスの特徴とシミュレーション結果を再利用することによって，*large* のキャッシュ・ミス率を予測する．メモリアクセスの特徴抽出およびキャッシュ・ミス率の測定には，プロセッサ・シミュレータである SimpleScalar [8] を用いた．

予測精度の指標として，絶対性能においてはキャッシュ・ミス率の差を，相対性能においては第 3.5.2 節で述べたスピアマン順位相関係数を用いる．ただし，同順位になるキャッシュ構成がある場合，つまり，キャッシュ・ミス率が同一である場合は

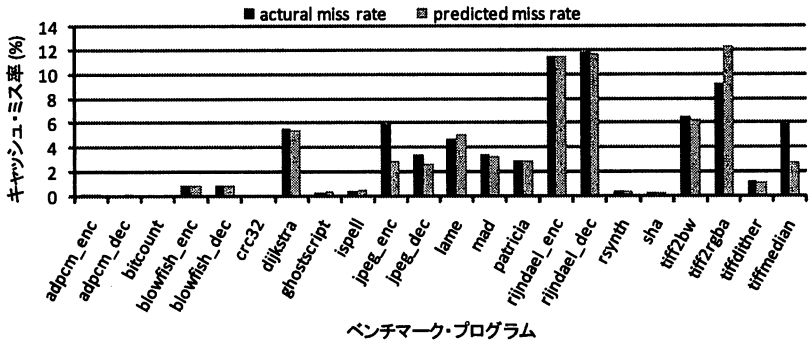

図 4　各プログラムにおける絶対性能予測精度

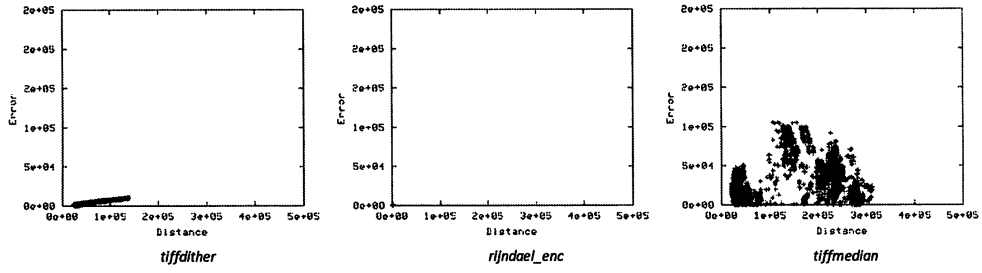

図 5　類似度と誤差の関係

平均順位を用いる．平均順位とは，同一順位になるキャッシュ構成数で，その順位の和を除した値である．

評価の対象としたキャッシュ構成を表 1 に示す．各パラメータのすべての組み合わせた 27 のキャッシュ構成を対象に評価を行った．これらのキャッシュ構成を対象とする場合，各命令インターバルにおいて十分なメモリアクセス数を確保するため，命令インターバルは 1M ロード・ストア命令とした．したがって，特徴ベクトル間の類似度は式 (2) で求めることとする．第 3.3 節で述べたように，アドレス・インターバルは精度に影響を与える重要なパラメータであるが，本稿においては 32B として評価した．

4.2　絶対性能予測精度

図 4 に，本手法によりキャッシュ・ミス率を予測した結果を示す．縦軸はキャッシュ・ミス率，横軸はベンチマーク・プログラムを示している．このときのラインサイズは 32B，キャッシュサイズは 4KB であり連想度は 4 である．*jpeg_enc*，*jpeg_dec*，*tiff2rgba* および *tiffmedian* を除くプログラムでは予測誤差は 1 ポイント以下であった．

本稿では，メモリアクセスの特徴の類似度とキャッシュ・ミス率との間には相関があるという予測のもと議論を進めた．つまり，2 つの命令インターバルにおけるメモリアクセスの特徴の類似度が高ければ，その命令インターバル間のミス率の差は小さくなることから，高精度に予測可能である．一方，相関が低い場合は予測精度が低下すると考えられる．図 5 は提案手法によって定義されるメモリアクセスの類似度と，キャッシュ・ミス率との関係を示している．ここで対象としたプログラムは，図 4 で比較的高い精度を示した *tiffdither* および *rijndael_enc* と，精度の低い *tiffmedian* である．各プログラムにおけるデータベース参照時のユークリッド距離と，各命令インターバル間のキャッシュ・ミス数の差の絶対値（以後，予測ミス数の差と呼ぶ）をプロットしたグラフである．横軸はユークリッド距離，縦軸は予測ミス数の差を示している．

予測精度が高い原因は主に 2 つある．第一に，ユークリッド距離と予測ミス数の差に正比例の相関があることが挙げられる．図 5 の *tiffdither* のように距離が長くなることに伴い，予測ミス数の差が大きくなる場合である．第二に，異なる入力データを用いた場合でも，メモリアクセスの特徴およびキャッシュ・ミス回数が各命令インターバルで変化しない場合である．図 5 の *rijndael_enc* がこれに該当する．低い予測精度を示す *tiffmedian* では，上述の相関が確認できない．つまり，予測精度が低いプログラムでは，本稿で提案したメモリアクセスの特徴抽出手法およびユークリッド距離による特徴の類似度の定義によって，距離と予測ミス数の差に相関を見出すことは困難であった．したがって，低い予測精度を示したと考えられる．

4.3　相対性能予測精度

図 6 に相対性能予測精度を示す．横軸はベンチマークを示しており，縦軸はスピアマン順位相関係数である．スピアマン順位相関係数の平均値は 0.971 であり，予測精度が高いことが分かる．一方，*bitcount* においては他のプログラムと比較して低い値を示している．このプログラムにおいては，キャッシュ・ミ

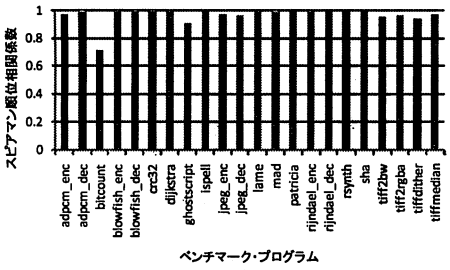

図 6 各プログラムにおけるスピアマン順位相関係数

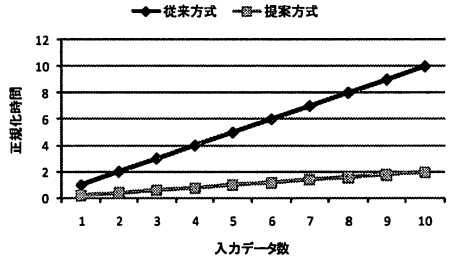

図 7 入力データ数と予測時間の関係

スがほとんど発生しないためミス率は極めて低く，0%に近い．事実上どのキャッシュ構成においても絶対性能はほぼ同じと言える．しかしながら，予測したキャッシュ・ミス率で順位付けすると 0.001 から 0.006 ポイントの予測誤差により予測した順位が実際の順位と異なる．したがって，*bitcount* におけるスピアマン順位相関係数が低下したと考えられる．

4.4 予測時間の結果

図 7 に，入力データ数と従来方式によるシミュレーション時間 T_{conv} および本手法による予測時間 T_r の関係を示す．T_{sim_eva} および T_{sim_db} は *tiff2bw* を対象として求めた．*tiff2bw* において，入力 *small* による実行命令数は入力 *large* と比較して 20 分の 1 程度であることから [7]，式 (10) を満たす．評価対象入力データの増加に伴い，この関係を満たすデータベース用入力データを用いるという前提のもとで，提案手法の結果を求めた．縦軸は従来方式で 1 つの入力データを対象に評価した時間で正規化した値であり，横軸は入力データ数を示している．なお，N_{conf} は表 1 に示す構成のすべてを対象とするため 27 とした．

図 7 から，提案手法による予測は従来手法と比較して短いことが分かる．従来方式では，*large* を対象にシミュレーションするため，T_{sim_eva} の値が大きくなる．一方，提案方式では *small* を対象にシミュレーションすることから，T_{sim_db} の値は T_{sim_eva} と比較して小さい．また，多くのキャッシュ構成を対象とする場合，式 (10) を満たしていることから T_r は T_{conv} よりも短くなる．

5. おわりに

本稿ではプログラムの異なる入力データにおけるキャッシュ性能を高速に予測するために，シミュレーション結果を再利用する手法を提案した．評価対象プログラムでデータベース生成用入力データを用いて実行し，メモリアクセスの特徴およびキャッシュ・ミス率からデータベースを生成した．評価対象プログラム上で評価対象入力データを実行した際のメモリアクセスの特徴とデータベースに登録された特徴とを比較することで，キャッシュ・ミス率を予測した．評価の結果，従来手法と比較して本手法は多くのプログラムにおいて有効であることを確認した．

本稿における評価ではデータベース生成フェーズとキャッシュ・ミス率予測フェーズにおいて同一のプログラムを対象とし，異なる入力データを用いて性能を予測した．本手法はプログラム間におけるメモリ性能予測にも適用可能である．つまり，データベース生成フェーズとキャッシュ・ミス率予測フェーズにおいて評価対象プログラムが異なる場合でも，類似度の高いメモリアクセスパタンが発生していればミス率を予測可能であると考えられる．また，第 3.3 節にて議論したように，アドレス・インターバルは予測精度に影響を与えると予想され，適切な値は各ベンチマーク・プログラムによって異なると考えられる．今後，アドレス・インターバルと精度の関係について調査する予定である．

謝辞 日頃からご討論頂く九州大学の安浦・村上・松永・井上研究室の皆様に感謝致します．本研究は一部，日本学術振興会の支援ならびにパナソニック株式会社との共同研究による．また，本研究は主に九州大学情報基盤研究開発センターの研究用計算機システムを利用した．

文 献

[1] L. Eeckhout, H. Vandierendonck and K. D. Bosschere: "Workload design: Selecting representative program-input pairs", PACT '02: Proceedings of the 2002 International Conference on Parallel Architectures and Compilation Techniques, pp. 83–94 (2002).

[2] Y. Zhong, S. G. Dropsho and C. Ding: "Miss rate prediction across all program inputs", PACT '03: Proceedings of the 12th International Conference on Parallel Architectures and Compilation Techniques, p. 79 (2003).

[3] C. Ding and Y. Zhong: "Predicting whole-program locality through reuse distance analysis", PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation, pp. 245–257 (2003).

[4] R. L. Mattson, J. Gecsei, D. R. Slutz and I. L. Traiger: "Evaluation techniques for storage hierarchies", IBM System Journal, **9(2)**, pp. 78–117 (1970).

[5] M. D. Hill and A. J. Smith: "Evaluating associativity in cpu caches", IEEE Trans. Comput., **38**, 12, pp. 1612–1630 (1989).

[6] K. Hoste, A. Phansalkar, L. Eeckhout, A. Georges, L. K. John and K. D. Bosschere: "Performance prediction based on inherent program similarity", PACT '06: Proceedings of the 15th international conference on Parallel architectures and compilation techniques, pp. 114–122 (2006).

[7] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge and R. B. Brown: "Mibench: A free, commercially representative embedded benchmark suite", IEEE 4th Annual Workshop on Workload Characterization (2001).

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