情報処理学会論文誌 Vol. 52 No. 12 3172–3183 (Dec. 2011)

シミュレーション結果の再利用による キャッシュ・ミス率予測技術

小野貴継[†1,†2,*1] 井上弘士[†3] 村上和彰[†3]

本稿ではシミュレーション結果を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ーンが出現する可能性がある．メモリアクセスパター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ー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．

Reusing Simulation Results for Cache Miss Rate Prediction

Takatsugu Ono,[†1,†2,*1] Koji Inoue[†3] and Kazuaki Murakami[†3]

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.

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．たとえば，代表的なキャッシュアクセスの時間見積りツールとして文献 18) がある．これに対し，キャッシュ・ミス率はキャッシュ構成だけでなく，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 つの要因となりうる．

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ーンが異なるプログラムを実行する際，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．従来手法では，類似したメモリアクセスパター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よるキャッシュ・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ー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ーンと

†1 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
†2 日本学術振興会特別研究員
Research Fellow of the Japan Society for the Promotion of Science
†3 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
*1 現在，株式会社富士通研究所
Presently with FUJITSU LABORATORIES LTD.

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ーンを抽出する．データベースに登録されたメモリアクセスパター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，1 度のメモリアクセスパターン比較で，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，2 章で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．3 章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．4 章で提案手法の有効性を評価し，最後に 5 章で本稿をまとめる．

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お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} はキャッシュ構成あた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

評価時間の短縮を目的として，これまで多くの研究が行われてきた．T_{sim} を削減するために，シミュレータの高速化手法[1),17)] や，統計的な情報を利用したベンチマーク合成技術[3),9),11),13)]，ベンチマーク・プログラムからシミュレーションの対象とする部分を選択する手法[10),12),14),19)] などが提案されている．

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[4)]．つまり，式 (1) の T_{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している．この特徴量に対して PCA（Principal Components Analysis）とクラスタ解析を行い，複数の入力データを用いてプログラムを実行した際の特徴量の類似度を測定している．同一のクラスタに複数の入力データが分類される場合，冗長なシミュレーションを行うことになる．そこで，この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，精度を低下させることなく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[20)]．この手法もまた，T_{input} を削減可能である．これはメモリアクセスの特徴を reuse distance[2)] 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．

N_{conf} を削減する手法も提案されている．Mattson らはすべてのキャッシュサイズに対するミス率を 1 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[8)]．Hill らは Mattson らの手法を拡張し，1 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[6)]．

本稿で提案する手法は式 (1) の T_{sim} および N_{conf} の削減を目的としている．評価に用いる入力データによってプログラムを実行した際のキャッシュ・ミス率を，シミュレーションではなくデータベース参照により予測することで T_{sim} の削減を目指す．さらに，データベース内に複数キャッシュ構成のシミュレーション結果を登録することで，1 度のデータベース参照でそれらのミス率を予測可能である．したがって，N_{conf} が削減される．

プログラムの特徴とシミュレーション結果をデータベース化し，特徴を比較することによって結果を予測する手法が提案されている．Hoste らは，ある計算機上で実行したときの性能とプログラムの振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舞いからその性能を予測している[7)]．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．

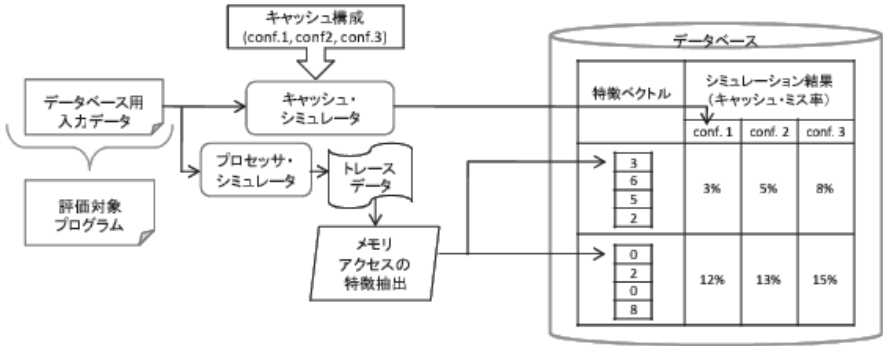

図 1　データベース生成フェーズ
Fig. 1　Database generation phase.

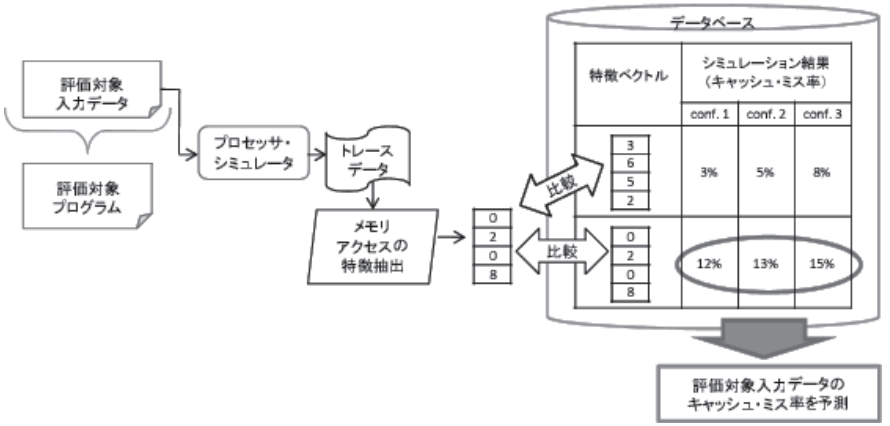

図 2　キャッシュ・ミス率予測フェーズ
Fig. 2　Cache miss rate prediction phase.

- 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

評価対象プログラムを複数の入力データを用いて実行する場合，そのプログラムの構造によっては，ある一定の区間に着目するとメモリアクセスの特徴が類似している可能性がある．この場合，キャッシュ・ミス率もまた近い値になると考えられる．この特徴の類似度を利用して，異なる入力データのシミュレーション結果を予測する．

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すこととする．特徴抽出に関する詳細は 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 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，ミス率を予測するうえ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

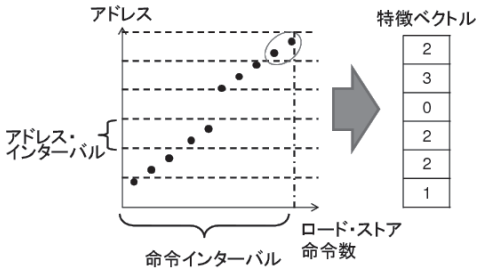

図 3　メモリアクセスの特徴抽出法
Fig. 3　Memory access characterization.

がって，プロセッサ・シミュレータなど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 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 1 度だけ特徴抽出作業を行えばよい．データベース生成フェーズおよびキャッシュ・ミス率予測フェーズともに同一の一定命令数によって分割した特徴ベクトルを用いるものとする．

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うえで，アドレス・インターバルを決定することが望ましい．命令インターバルと精度との関係については 3.5.1 項で議論する．

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，1 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．

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

データベースにおいて最も類似度が高い命令インターバルのミス数を用いてミス率を予測する．類似度が同じ命令インターバルが複数ある場合，どの命令インターバルを用いても問題ない．なぜなら，類似度が等しければキャッシュ・ミス率も近い値だと考えられるからである．したがって，このような場合はデータベース探索において最初に類似度を計算した命令インターバルを用いる．

1 つの命令インターバルのみの類似度を比較するだけでは，高い予測精度を得られない可能性がある．なぜなら，キャッシュ・ミス率は直前のキャッシュ状態に依存するからである．この問題を回避する手段は主に 2 つある．まず，命令インターバルの間隔を対象とするキャッシュサイズに対して十分長くすることである．これにより，直前のキャッシュ状態の差による精度への影響を緩和することができる．予測対象とするキャッシュサイズが小さい場合に有効である．次に，予測精度向上のために予測する命令インターバルの直前に実行されるインターバルの類似度も考慮することである．これは特に大きなキャッシュサイズを対象にミス率を予測する場合は効果的である．この場合，V_{T_t} のインターバルが実行される以

前の時間的に連続した命令インターバルの特徴を $V_{T_{t-i}}$，同様に V_{D_t} 以前のインターバルの特徴を $V_{D_{t-i}}$ とすると，直前の時間的に連続した n インターバルの特徴の類似度を考慮した S_l は式 (3) によって表される．

$$S_l = \frac{1}{\sum_{i=0}^{n} E(V_{T_{t-i}}, V_{D_{t-i}})} \tag{3}$$

たとえば $n = 1$ の場合，予測対象命令インターバルの時間的に 1 つ前のインターバルの類似度を考慮することになる．対象とするキャッシュ構成の大きさに応じて n を変更することで予測精度の向上が見込める．

予測するキャッシュ・ミス率 *cmr* は式 (4) で求められる．ここで，*p_miss* はデータベースに登録された命令インターバルにおいて最も類似度が高いインターバルのミス数である．*access* は命令インターバルの値で決定される．p_miss_i および $access_i$ は，それぞれインターバル i における *p_miss* と *access* を表す．たとえば，命令インターバルの値が 1 M ロード・ストア命令数である場合，*access* の値は 1 M である．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 (4) 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．

$$cmr = \left(\frac{\sum_{i=1}^{\mathrm{interval}} p\_\mathrm{miss}_i}{\sum_{i=1}^{\mathrm{interval}} \mathrm{access}_i} \right) \times 100 \tag{4}$$

本手法は命令インターバルにおけるメモリアクセスの特徴が類似しており，キャッシュ・ミス率も近い値である場合は高い精度で予測が可能になる．これは入力データによってキャッシュ・ミス率が大きく異なる場合でも同じであり，プログラム全体ではなく命令インターバル単位でキャッシュ・ミス率が類似していればよい．しかしながら，類似度が高い一方，キャッシュ・ミス率が大きく異なる場合は予測精度が低下する．

マルチコアプロセッサでも複数コアで共有しない L1 キャッシュなどを対象とする場合は本手法の効果が期待できる．しかしながら，共有する場合はプログラム実行のタイミングなどの要因により，メモリアクセスの特徴の種類がシングルコア実行時と比較して増加することが予想される．この課題を解決するためには，データベース生成時により多くの特徴を登録するなどの対策が必要である．

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 (4) 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，予測精度が高いといえる．

したがって，相対性能の予測精度は式 (5) に示すスピアマン順位相関係数 r_s によって表すこととする．

$$r_s = 1 - \frac{6 \sum_{i=1}^{c} d_i^2}{c^3 - c} \tag{5}$$

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 に近いほど，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 (6) で表される．

$$T_{\mathrm{conv}} = T_{\mathrm{sim\_eva}} \times N_{\mathrm{input\_p}} \times N_{\mathrm{conf}} \tag{6}$$

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，N_{input_p} は評価対象入力データの数とする．

提案手法によるキャッシュ・ミス率予測時間 T_r は式 (7) によって求められる．

$$T_r = T_{\mathrm{db}} + T_p \tag{7}$$

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 (8)，式 (9) で表すことができる．

$$T_{\mathrm{db}} = T_{\mathrm{e\_db}} + T_{\mathrm{sim\_db}} \times N_{\mathrm{conf}} \tag{8}$$

$$T_p = (T_{\mathrm{e\_p}} + T_{\mathrm{ref}}) \times N_{\mathrm{input\_p}} \tag{9}$$

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 (6) および式 (7) より次式を満たす必要があることが分かる．

$$T_{\mathrm{sim\_db}} < T_{\mathrm{sim\_eva}} \times N_{\mathrm{input\_p}} - \frac{T_{\mathrm{e\_db}} + (T_{\mathrm{e\_p}} + T_{\mathrm{ref}}) \times N_{\mathrm{input\_p}}}{N_{\mathrm{conf}}}$$

ここで，$N_{input\_p} = 1$ として，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．

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

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．組み込みプロセッサを対象とすることから，評価に用いるプロセッサは ARM 命令セットアーキテクチャとした．ベンチマーク・プログラムは組み込みプロセッサを対象とするため，MiBench [5] から 22 種類を用いる．また，提案手法が組み込み向けベンチマーク・プログラム以外にも適用可能であること，そして比較的高いキャッシュ・ミス率のプログラムにおいても有効であることを確認するため，SPEC2000 [16] から 2 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[5]，*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．本評価ではデータベース用入力データを 1 つだけ用いるが，提案手法はデータベース用入力データを 1 つに限定するものではない．SPEC2000 では，データベース用入力データとして TEST を，評価対象入力データとして TRAIN を用いた．ただし，TEST および TRAIN を実行開始からそれぞれ 200 M 命令，1 G 命令を対象とした．データベース用入力データおよび評価対象入力データにおけるロード・ストア命令数を**表 1** に示す．

各入力データにおけるキャッシュ・ミス率は**図 4** に示すとおりである．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 [15] 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 3.5.2 項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 2** に示す．各パラメータのすべてを組み合わせた 27 のキャッシュ構成を対象に評価を行った．置換アルゴリズムは一般的に用いられる LRU とした．キャッシュサイズが比較的小さいことから，十分なメモリアクセス数を確保することで直前のキャッシュの状態の影響を緩和することが可能だと考えられる．したがって，命令インターバルは 1 M ロード・ストア命令とし，特徴ベクトル間の類似度は式 (2) で求めるものとする．アドレス・インターバルは 16 B，32 B，64 B，128 B，256 B，512 B，1,024 B を対象に絶対性能予測精度および相対性能予測精度に関する評価を行った．その結果，アドレス・インターバルが大きくなるほど精度が低下することを確認した．したがって，予測精度を上げるにはアドレス・インターバルが小さいほうが良い．しかしながら，アドレス・インターバルが小さい場合，特徴ベクトルの次元が増加することによりデータベース参照時のユークリッド距離算出に要する計算量が増加し，予測時間が長くなる．つまり，予測時間の観点から考えるとアドレス・インターバルは大きいほうが良い．ここでは，精度と予測時間を考慮して 32 B を対象に評価結果を示し議論する．

表 1　各プログラムにおけるロード・ストア命令数
Table 1　The number of load and store instructions.

プログラム	評価対象入力データ	データベース用入力データ
adpcm_enc	100,225,935	5,163,299
adpcm_dec	100,225,934	5,163,298
bitcount	183,398,225	12,244,947
blowfish_enc	388,428,071	37,357,266
blowfish_dec	388,428,173	37,357,368
crc32	984,771,274	50,664,607
dijkstra	117,292,392	26,984,841
ghostscript	377,061,290	368,562,633
ispell	443,500,551	4,528,209
jpeg_enc	39,215,368	10,474,944
jpeg_dec	11,170,434	3,011,954
lame	631,310,799	51,913,283
mad	111,406,837	9,367,397
patricia	257,227,891	41,837,353
rijndael_enc	183,336,669	17,613,572
rijndael_dec	172,151,637	16,539,079
rsynth	479,539,354	33,077,063
sha	36,587,102	3,522,046
tiff2bw	58,298,542	14,176,275
tiff2rgba	101,512,239	24,635,729
tiffdither	258,000,394	85,607,511
tiffmedian	207,312,939	56,940,158
175.vpr	308,508,083	61,243,703
179.art	278,327,002	55,987,182

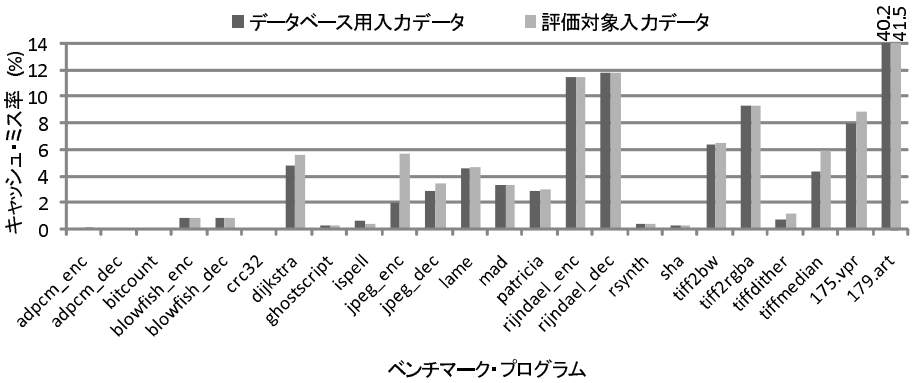

図 4　*small* および *large* のキャッシュ・ミス率
Fig. 4　Cache miss rate of small and large input data.

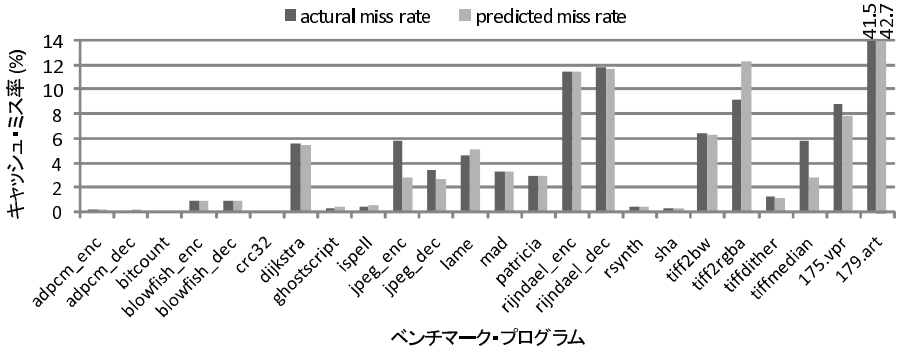

図 5　各プログラムにおける絶対性能予測精度
Fig. 5　Cache miss rate prediction accuracy of all programs.

表 2　キャッシュ構成
Table 2　Cache configuration.

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

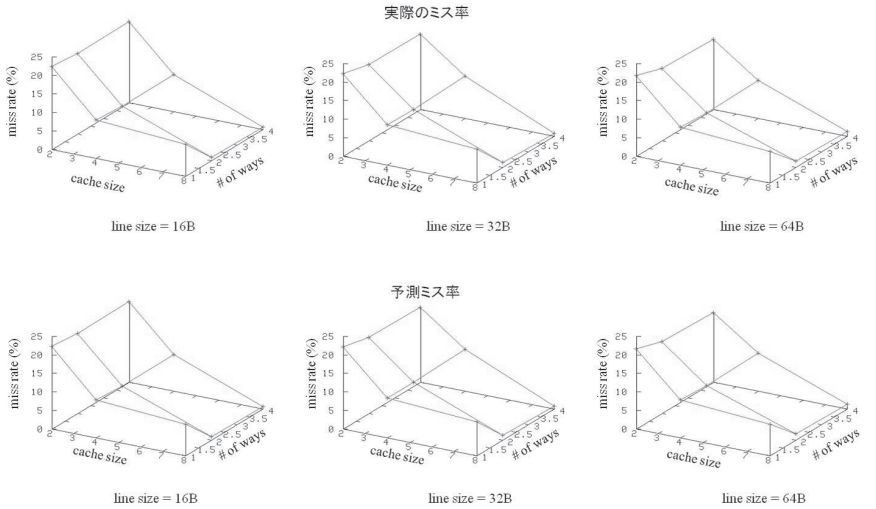

図 6　*rijndael_enc* における複数キャッシュ構成を対象とした絶対性能予測精度
Fig. 6　Cache miss rate prediction accuracy of all cache configurations (*rijndael_enc*).

4.2　絶対性能予測精度

4.2.1　全プログラムにおける結果

図 5 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 32 B，キャッシュサイズは 4 KB であり連想度は 4 である．*jpeg_enc*，*jpeg_dec*，*tiff2rgba*，*tiffmedian* および *179.art* 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．高い予測精度を達成した原因および，これらのプログラムで精度が低下した原因に関する考察は 4.2.3 項で述べる．

4.2.2　全キャッシュ構成における結果

表 2 の全キャッシュ構成を対象にキャッシュ・ミス率を予測した結果を示す．図 5 において，高い予測精度を示した *rijndael_enc* における，すべてのキャッシュ構成を対象とした予測結果を**図 6** に示す．各グラフの x 軸および y 軸はキャッシュサイズおよび連想度を示しており，z 軸はキャッシュ・ミス率である．上段に示す結果が実際のキャッシュ・ミス率，下段が本手法により予測した結果である．また，左から順にラインサイズを 16 B，32 B およ

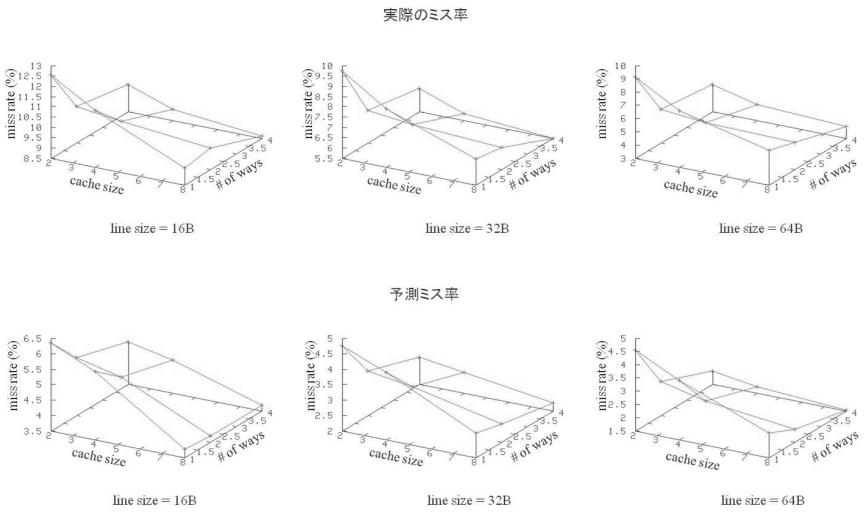

図 7　*tiffmedian* における複数キャッシュ構成を対象とした絶対性能予測精度
Fig. 7　Cache miss rate prediction accuracy of all cache configurations (*tiffmedian*).

び 64 B に固定した結果を示している．これらのグラフを比較すると，どのキャッシュ構成においても高い予測精度を達成していることが分かる．

次に，図 5 において低い精度を示した *tiffmedian* の結果を**図 7** に示す．各キャッシュ構成において，予測キャッシュ・ミス率が実際の結果と異なることが分かる．しかしながら，各キャッシュ構成におけるミス率の相対的な傾向は上段のグラフと類似していることから，相対的なキャッシュ性能の予測は可能であると考えられる．相対性能予測精度に関しては 4.3 節で議論する．

4.2.3　絶対性能予測精度に関する考察

図 5 より，プログラムによって提案手法による予測精度が異なることが分かった．本項ではこの原因につて考察する．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前提で議論を進めた．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いと予測した．逆に相関が低い場合は，キャッシュ・ミス率の予測誤差が大きくなると考えられる．そこで，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・ミス率との関係について調査する．

図 8 にその結果を示す．ここで対象としたプログラムは，図 5 で比較的高い精度を示した *tiffdither*，*lame*，*patricia* および *rijndael_enc* と，精度の低い *jpeg_enc* および *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 1 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あげられる．図 8 の *tiffdither* および *lame* のように距離が長くなることにともない，予測ミス数の差が大きくなる場合である．第 2 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 8 の *patricia* や *rijndael_enc* がこれに該当し，ユークリッド距離が比較的短く，予測ミス数の差がきわめて小さいことが確認できる．低い予測精度を示す *jpeg_enc* および *tiffmedian* では，上述の相関が確認できない．また，広範囲に分布していることから，各命令インターバルにおけるメモリアクセスの特徴およびキャッシュ・ミス回数が大きく異なることが分かる．3.5.1 項において予測精度を向上させるために直前のインターバルを考慮する方式について述べた．*jpeg_enc* と *tiffmedian* を対象に式 (3) を用い，n=1 として評価を行った．その結果，*jpeg_enc* では精度の改善が確認できず，*tiffmedian* においても 0.1 ポイントの改善にとどまった．このことから，直前のキャッシュ状態が精度の低下の主な要因でないことが分かる．つまり，これらのプログラムでは，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 9 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．*tiffmedian* は図 7 において，絶対性能の予測精度は低い結果であったが，上段のグラフと下段のグラフの形状が類似していることから相対性能予測精度は高いと考えられた．実際に，図 9 の結果より相対性能の予測精度は高いことが確認できる．

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・ミスがほとんど発生しないためミス率はきわめて低く，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いえる．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

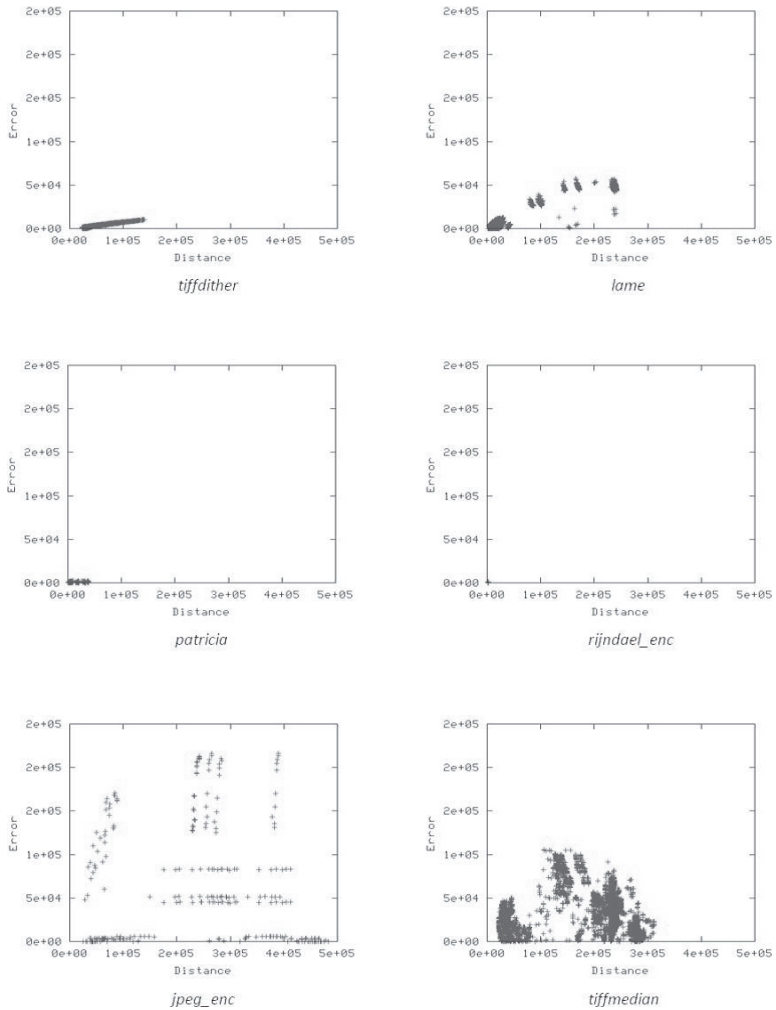

図 8　類似度と誤差の関係
Fig. 8　Relation between similarity and error.

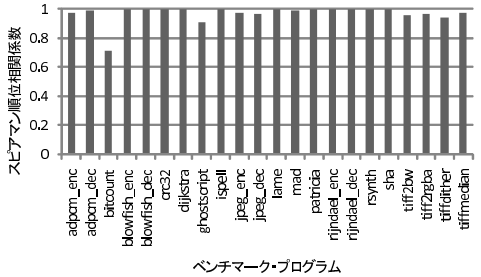

図 9　各プログラムにおけるスピアマン順位相関係数
Fig. 9　Spearman rank-correlation coefficients in each programs.

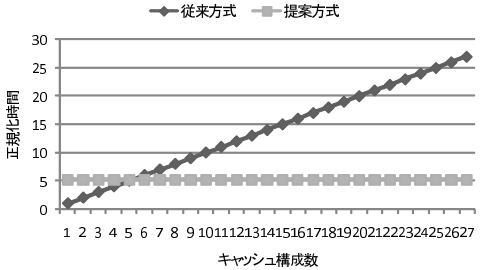

図 10　キャッシュ構成数と予測時間の関係
Fig. 10　Relation between the number of cache configurations and prediction time.

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 10 に，従来方式による評価時間 T_{conv} および本手法のキャッシュ・ミス率予測フェーズ時間 T_p の結果を示す．*tiff2bw* を入力データ *large* および *small* を用いて実行して T_{sim_eva} および T_{sim_db} を求めた．*tiff2bw* は *small* による実行命令数が *large* と比較して 20 分の 1 程度であり[5]，式 (10) を十分満たす．このとき用いたキャッシュ・シミュレータは，SimpleScalar の sim-cache である．sim-cache は実行時間などの評価はできないが，キャッシュ・ミス率を高速に測定することが可能である．縦軸は 1 つのキャッシュ構成に対する従来方式の時間で正規化した値を示しており，横軸はキャッシュ構成数である．

T_p は式 (9) より，N_{conf} に依存しないことが分かる．したがって，キャッシュ構成数の増

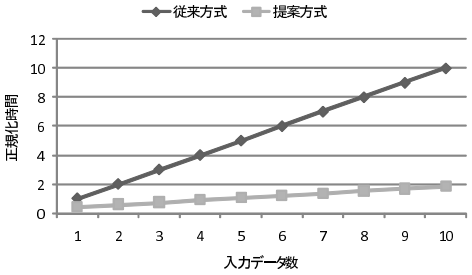

図 11　入力データ数と予測時間の関係
Fig. 11　Relation between the number of inputs and prediction time.

加により変化することはない．一方，従来方式ではキャッシュ構成数の増加にともないシミュレーション時間が増加する．T_{e_p} はシミュレーションによりメモリアクセストレースを取得し，特徴を抽出する時間が含まれる．つまり，キャッシュシミュレーションのみを高速に実行する sim-cache の実行時間 T_{sim_eva} よりも T_{e_p} が長い．したがって，キャッシュ構成数が 5 以下では提案手法のほうが遅い．プログラムによって提案手法が優位になるキャッシュ構成数は異なる．提案手法がキャッシュ構成数 6 以上で優位になるとは限らないが，*tiff2bw* 以外のプログラムにおいても同様に多くのキャッシュ構成を対象とする際により効果的になる傾向にある．

図 11 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．評価対象入力データの増加にともない，式 (10) を満たすデータベース用入力データも追加す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 2 に示す構成のすべてを対象とするため 27 とした．

図 11 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．T_{sim_db} はプログラムに依存するため他のプログラムではグラフの傾きが異なるが，*tiff2bw* 以外のプログラムにおいても同様の傾向になる．

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，キャッシュ・ミス率を予測可能であると考えられる．類似度の高いメモリアクセスパターンが発生していれば高い精度で予測可能であると考えられる．

謝辞　本稿をまとめるにあたり，ともにご討論いただいた九州大学の安浦・村上・松永・井上研究室の皆様に感謝いたします．なお，本研究は，一部，独立行政法人新エネルギー・産業技術総合開発機構（NEDO）若手グラント，ならびに，科学研究費補助金（課題番号：21680005）による．

参考文献

1) Chiou, D., Sunwoo, D., Kim, J., Patil, N.A., Reinhart, W., Johnson, D.E., Keefe, J. and Angepat, H.: FPGA-Accelerated Simulation Technologies (FAST): Fast, Full-System, Cycle-Accurate Simulators, *MICRO '07: Proc. 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pp.249–261, IEEE Computer Society, Washington, DC, USA (online), DOI:http://dx.doi.org/10.1109/MICRO.2007.16 (2007).
2) Ding, C. and Zhong, Y.: Predicting whole-program locality through reuse distance analysis, *PLDI '03: Proc. ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*, pp.245–257, ACM, New York, NY, USA (online), DOI:http://doi.acm.org/10.1145/781131.781159 (2003).
3) Eeckhout, L., Bell, R.H., Jr., Stougie, B., Bosschere, K.D. and John, L.K.: Control Flow Modeling in Statistical Simulation for Accurate and Efficient Processor De-

sign Studies, *ISCA '04: Proc. 31st Annual International Symposium on Computer Architecture*, p.350, IEEE Computer Society, Washington, DC, USA (2004).
4) Eeckhout, L., Vandierendonck, H. and Bosschere, K.D.: Workload Design: Selecting Representative Program-Input Pairs, *PACT '02: Proc. 2002 International Conference on Parallel Architectures and Compilation Techniques*, pp.83–94, IEEE Computer Society, Washington, DC, USA (2002).
5) Guthaus, M.R., Ringenberg, J.S., Ernst, D., Austin, T.M., Mudge, T. and Brown, R.B.: MiBench: A free, commercially representative embedded benchmark suite, *IEEE 4th Annual Workshop on Workload Characterization* (Dec. 2001).
6) Hill, M.D. and Smith, A.J.: Evaluating Associativity in CPU Caches, *IEEE Trans. Comput.*, Vol.38, No.12, pp.1612–1630 (online), DOI:http://dx.doi.org/10.1109/12.40842 (1989).
7) Hoste, K., Phansalkar, A., Eeckhout, L., Georges, A., John, L.K. and Bosschere, K.D.: Performance prediction based on inherent program similarity, *PACT '06: Proc. 15th International Conference on Parallel Architectures and Compilation Techniques*, pp.114–122, ACM, New York, NY, USA (online), DOI:http://doi.acm.org/10.1145/1152154.1152174 (2006).
8) Mattson, R.L., Gecsei, J., Slutz, D.R. and Traiger, I.L.: Evaluation Techniques for Storage Hierarchies, *IBM System Journal*, Vol.9, No.2, pp.78–117 (1970).
9) Nussbaum, S. and Smith, J.E.: Modeling Superscalar Processors via Statistical Simulation, *PACT '01: Proc. 2001 International Conference on Parallel Architectures and Compilation Techniques*, pp.15–24, IEEE Computer Society, Washington, DC, USA (2001).
10) 小野貴継ほか：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，情報処理学会論文誌コンピューティングシステム，Vol.48, No.19, pp.203–213 (2007).
11) Oskin, M., Chong, F.T. and Farrens, M.: HLS: Combining statistical and symbolic simulation to guide microprocessor designs, *ISCA '00: Proc. 27th Annual International Symposium on Computer Architecture*, pp.71–82, ACM, New York, NY, USA (online), DOI:http://doi.acm.org/10.1145/339647.339656 (2000).
12) Patil, H., Cohn, R., Charney, M., Kapoor, R., Sun, A. and Karunanidhi, A.: Pinpointing Representative Portions of Large Intel® Itanium® Programs with Dynamic Instrumentation, *MICRO 37: Proc. 37th Annual IEEE/ACM International Symposium on Microarchitecture*, pp.81–92, IEEE Computer Society, Washington, DC, USA (online), DOI:http://dx.doi.org/10.1109/MICRO.2004.28 (2004).
13) Robert, H., Bell, J. and John, L.K.: Improved Automatic Testcase Synthesis for Performance Model Validation, *ICS '05: Proc. 19th Annual International Conference on Supercomputing*, pp.111–120, ACM Press, New York, NY, USA (online), DOI:http://doi.acm.org/10.1145/1088149.1088164 (2005).
14) Sherwood, T., Perelman, E., Hamerly, G. and Calder, B.: Automatically Characterizing Large Scale Program Behavior, *ASPLOS-X: Proc. 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp.45–57, ACM Press, New York, NY, USA (online), DOI:http://doi.acm.org/10.1145/605397.605403 (2002).
15) SimpleScalar Simulation Tools for Microprocessor and System Evaluation, available from ⟨http://www.simplescalar.org/⟩.
16) SPEC, available from ⟨http://www.specbench.org/⟩.
17) 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).
18) Wilton, S.J.E. and Jouppi, N.P.: Cacti: An Enhanced Cache Access and Cycle Time Model, *IEEE Journal of Solid-State Circuits*, Vol.31, pp.677–688 (1996).
19) Wunderlich, R.E., Wenisch, T.F., Falsafi, B. and Hoe, J.C.: SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling., *Proc. 30th International Symposium on Computer Architecture*, pp.84–95 (2003).
20) Zhong, Y., Dropsho, S.G. and Ding, C.: Miss Rate Prediction across All Program Inputs, *PACT '03: Proc. 12th International Conference on Parallel Architectures and Compilation Techniques*, p.79, IEEE Computer Society, Washington, DC, USA (2003).

(平成 23 年 3 月 1 日受付)
(平成 23 年 9 月 12 日採録)

小野　貴継（正会員）

昭和 57 年生．平成 21 年九州大学大学院システム情報科学府情報理学専攻博士課程修了．博士（工学）．同年より日本学術振興会特別研究員（PD）．メモリアーキテクチャの評価に関する研究に従事．平成 22 年株式会社富士通研究所入社，現在に至る．データセンタ向けサーバの研究開発に従事．IEEE 会員．

井上　弘士（正会員）

昭和 46 年生．平成 8 年九州工業大学大学院情報工学研究科修士課程修了．同年横河電機（株）入社．平成 9 年より（財）九州システム情報技術研究所研究助手．平成 11 年の 1 年間 Halo LSI Design & Device Technology, Inc. にて訪問研究員としてフラッシュ・メモリの開発に従事．平成 13 年九州大学にて工学博士を取得．同年福岡大学工学部電子情報工学科助手．平成 16 年より九州大学大学院システム情報科学研究院助教授．平成 19 年 4 月より同大学准教授，現在に至る．高性能/低消費電力プロセッサ/メモリ・アーキテクチャ，ディペンダブル・アーキテクチャ，3 次元積層アーキテクチャ，性能評価，等に関する研究に従事．電子情報通信学会，ACM，IEEE 各会員．

村上　和彰（正会員）

昭和 35 年生．昭和 59 年京都大学大学院工学研究科情報工学専攻修士課程修了．同年富士通（株）入社．汎用大型計算機の研究開発に従事．昭和 62 年九州大学助手．平成 6 年九州大学助教授．現在九州大学大学院システム情報科学研究院情報理学部門教授，情報基盤研究開発センター長，情報統括本部長．計算機アーキテクチャ，並列処理，システム LSI 設計技術，等に関する研究に従事．工学博士．平成 3 年情報処理学会研究賞，平成 4 年情報処理学会論文賞，平成 9 年坂井記念特別賞，平成 12 年日経 BP 社 IP アワード，平成 12 年情報処理学会創立 40 周年記念論文賞，平成 14 年電子情報通信学会業績賞をそれぞれ受賞．