一般社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

統計的手法を用いた並列化コンパイラ協調 マルチコアアーキテクチャシミュレータ高速化手法

田口学豊† 木村啓二† 笠原博徳†

† 早稲田大学理工学術院 〒169-8555 東京都新宿区大久保 3-4-1
E-mail: †{gakuho,kimura,kasahara}@kasahara.cs.waseda.ac.jp

あらまし 本稿では，並列化コンパイラと協調しマルチコアアーキテクチャシミュレーションを高速化する手法を提案する．本手法では，まず実機での逐次実行のプロファイルを取得し，そのプロファイル結果を x-means 法でクラスタリングすることにより，評価対象アーキテクチャの詳細シミュレーションを行う箇所を特定する．さらに，クラスタリングの情報と評価対象マルチコアで実行するアプリケーションから，並列化コンパイラは精度切り替えコードを含む並列化コードを生成する．評価の結果，16 コアのシミュレーションを SPEC ベンチマークの equake において誤差 0.04%で 437 倍，MediaBench の MPEG2 エンコーダにおいて誤差 0.04%で 28 倍の速度向上をそれぞれ得ることが出来た．

キーワード 並列化アプリケーション，マルチコアアーキテクチャシミュレータ，統計的手法，高速化

A Parallelizing Compiler Cooperative Acceleration Technique of Multicore Architecture Simulation using a Statistical Method

Gakuho TAGUCHI† Keiji KIMURA† and Hironori KASAHARA†

† Faculty of Science and Engineering, Waseda University 3-4-1 Ookubo, Shinjuku-ku, Tokyo, 169-8555 Japan
E-mail: †{gakuho,kimura,kasahara}@kasahara.cs.waseda.ac.jp

Abstract A parallelizing compiler cooperative acceleration technique for multicore architecture simulation is proposed in this paper. Profile data of a sequential execution of a target application on a real machine is decomposed into multiple clusters by x-means clustering. Then, sampling points for a detail simulation mode in each cluster are calculated. In addition, a parallelizing compiler generates a parallelized code by taking both of the clustering information and the source code of the target application. The evaluation results show, in the case of the simulation for 16 cores, 437 times speedup is achieved with 0.04% error for equake, and 28 times speedup is achieved with 0.04% error for mpeg2 encoder.

Keyword Parallelized application, Multicore architecture simulator, Statistical method, Acceleration technique

1. はじめに

コンピュータアーキテクチャの研究及び開発において，アーキテクチャシミュレータは様々なアーキテクチャの初期評価で大きな役割を果たしてきた．しかしながら，シミュレーションには実機での実行の 10000～15000 倍という膨大な実行時間を要し，これが大きな問題となっている．その為，高速で精度の高いシミュレーション手法の研究が注目されている．

単一コア CPU のマイクロアーキテクチャの研究では，これまでプログラムのある一定区間の IPC を測定するといった形式で多くの評価が行われてきた．しかしながら，この方式では並列化アプリケーションをマルチコア上で実行するときのシミュレーションに対して単純には適用できないといった問題，及びプログラムの特徴的な部分を測定できているか不明であるという問題があった．

これらの問題のうち，プログラムの測定部分の絞り込みに関して，プログラムのある一部分のみを詳細に実行するサンプリング実行による高速化が提案されている．例えば，SimPoint[1]はアプリケーションのベーシックブロックごとの実行回数を計測した情報である BBV を基に詳細実行部分を絞り込む．BBV に対しクラスタリングを施すことでプログラムの特徴を表す部分を見つけ出し，クラスタごとの適切なサンプリング

ポイントを設定する. SimFlex[2]は, 命令数を基準としたプロファイルに対して統計を用いたサンプリングを施し，サンプルを算出する．しかし，マルチコアアーキテクチャにおいては，コア間のリソース競合や同期により，それぞれのスレッドにおいてアイドルやスピニングが起こる為，依然としてマルチコアのシミュレーションに対してこれらの方法を単純に適用することは出来ない．

マルチコアに対応したサンプリング高速化手法として，Carlson 等の手法がある[3]．これは，SimPoint と同様に BBV の解析によりアプリケーションの周期性を解析しサンプル対象を設定する手法だが，スレッドごとのアプリケーションの挙動に着目し，機能シミュレーション時にも詳細な同期を行うことでマルチコアアーキテテクチャのシミュレーションに対応している．この手法では，事前アプリケーション解析にシミュレーション対象アーキテクチャと同じコア数のプロファイルが必要となり，また SimPoint 同様 BBV のまとめ方やクラスタリングのパラメータを手動で適宜設定しなければならないといった問題がある．サンプリング実行以外のマルチコアに対する高速化手法としては，Bryan らの手法がある[5]．バリア同期間隔をスレッドごとの進捗の差異のない独立した単位として扱い，シミュレーション自体を並列化するものである．

本稿で提案する高速化手法は，並列化されたベンチマークアプリケーションのマルチコア上での実行を対象とし，これらアプリケーションのメインループをイタレーション単位でサンプリングするものである．本手法では，このような粒度では実行サイクル数等の計測対象の計測値の変動が，アーキテクチャの差異による寄与よりも，アプリケーションそのものの演算ステップ数の変動による寄与が大きいと考え，任意のアーキテクチャによる対象アプリケーションの逐次実行のプロファイル結果によりサンプリング対象をクラスタリングする．クラスタリング後，各クラスタのサンプル数を決定し，これらのサンプルのみを詳細シミュレーションする．対象アプリケーションでは，メインループそのものが並列化される，あるいはメインループ内部が並列化されているため，同期やコア間のリソース競合といった並列化特有のプロセスを考慮にいれた評価が可能となる．

先行研究に対して,本手法は以下の様な特徴を持つ．

1) 同期やコア間のリソース競合を含む粒度でのサンプリングにより，全コアで統一した単位のサンプルを得ることができ，マルチコアアーキテクチャへの対応が可能となる．
2) 本手法で対象とする粒度でのアプリケーションのコスト変動は並列化後も同じ傾向であるという仮定から，事前アプリケーション分析に用いるプロファイルは逐次実行のみとすることができる．
3) プロファイルの解析には x-means クラスタリングと統計手法を用いることで，手動によるパラメータの設定を必要とせず，自動的に最適なサンプルを算出することができる．

さらに本稿では,並列化コンパイラとの連携によるプロファイル部分特定及びサンプリング用ランタイム生成自動化フレームワークについても提案する．

以下，第 2 章で本稿で提案するシミュレーション高速化の手法，第 3 章でコンパイラと協調した高速化のフレームワーク，第 4 章で精度切り替え機能，第 5 章で性能評価，第 6 章でまとめについて述べる．

2. シミュレーション高速化の手法

本章では，本稿で提案するシミュレーション高速化手法について述べる．

2.1. シミュレーション高速化の概要

本稿で提案する高速化手法は，ベンチマークアプリケーションにおける，並列化可能ループまたは並列化可能ブロックを内包するメインループに着目してイタレーション単位でサンプリングを行う．サンプリングとは，ある母集団の部分集合である標本（サンプル）から，その母集団の性質を推定する手法である．本手法では，任意のループのイタレーション毎の実行サイクル数のプロファイルに対してサンプリングを行い，一部のイタレーションの実行サイクル数（サンプル）からループ全体の実行サイクル数（母集団）を推定する．この時，サンプルイタレーションのみ詳細なシミュレーションを行い，その他のイタレーションは簡易で高速なシミュレーションを行う．すなわち，シミュレーション速度の非常に遅い詳細シミュレーションを行う部分をサンプル部分に限定することで，シミュレーション時間の短縮を図る．

2.2. シミュレーションモード

本高速化手法では，次の 2 種類のシミュレーションモード（精度）を適宜切り替えながらシミュレーションを行う．

- 詳細シミュレーション
  キャッシュやパイプライン，相互結合網などのアーキテクチャ構造を詳細に再現しクロックサイクルレベルでシミュレーションを行うモード．サンプルイタレーションに対して行う．
- 機能シミュレーション
  命令実行のみの機能レベルでの簡易で高速なシミュレーションを行うモード．サンプル以外のイタレーションに対して行う．

2.3. サンプル算出の手法

本手法におけるサンプルとは，サンプリング対象ル

ープの全てのイタレーションを母集団とし，その母集団の実行サイクル数を推定するのに必要なイタレーションの数である．この時，精度を高く，より少ないサンプル数でループ全体の実行サイクル数を推定できるサンプルの算出が求められる．

2.3.1. 任意の実機での実行プロファイル取得

メインループのイタレーション毎の大局的な挙動は，プログラム構造と入力に依存するという前提から，ベンチマークアプリケーションを任意の実機で実行して取得したプロファイル情報に，シミュレーション対象アーキテクチャにおけるベンチマークプログラムのコスト変動の挙動も従うと仮定する．

そこでまず，ベンチマークアプリケーションを任意の実機で逐次実行し，イタレーション毎の実行サイクル数を計測する．このプロファイルに統計手法を用いたサンプリングを施し，得たサンプルをシミュレーションに用いる．

2.3.2. クラスタリング手法

任意の実機での逐次実行から得たプロファイルに対して統計処理を施す前に，クラスタリングによるプロファイルの分割を行う．母集団を偏差の小さい集合に分割することで，統計手法を施した際のサンプル数を減少させることができる．ここでは，クラスタリング法として x-means 法[5]を用いる．x-means 法とは，まず小さなクラスタ数 k_0 による k_0-means クラスタリングを行い，分割後の各集合に対して，分割が適当でないと判断されるまで 2-means クラスタリングを再帰的に行う手法である．ここでは分割停止基準をサンプル数の和とし，この値が改善するか否かにより分割停止の判断を行う．この x-means 法を用いることにより，入力集合に対して最適なクラスタ数の決定が自動化できる．イタレーションごとの実行サイクル数のプロファイルに対してクラスタリングを施した例を図 1 に示す．図では横軸がイタレーションの番号，縦軸が実行サイクル数をそれぞれ表し，イタレーションが実行サイクル数に応じて 4 クラスタに分割されていることを示す．

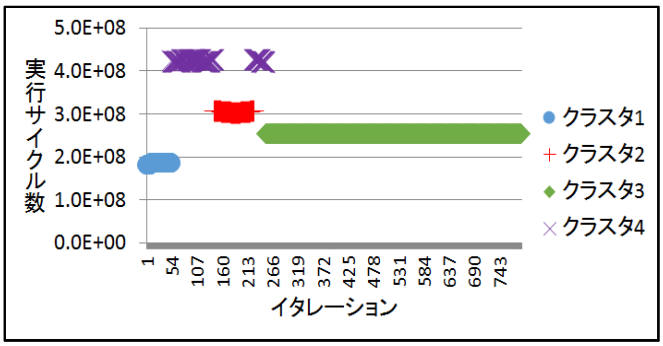

図 1 実行サイクル数によるクラスタリングの例

2.3.3. 統計的手法によるサンプル算出

サンプルはそれぞれのクラスタに対して，推定する全体の実行サイクル数が許容する誤差に収まるように統計手法によって決定する．クラスタリングによって得られた集合を $C_1, C_2, \cdots, C_k$ とする．C_i (i=1,2,…,k)について，次の式に 1 よってサンプル数 n_i を計算する[6].

$$サンプル数 n_i \geq \left\lceil \left(\frac{上側P\%点}{許容誤差率\varepsilon} \times \frac{標準偏差\sigma_i}{相加平均\mu_i} \right)^2 \right\rceil \quad (1)$$

2.4. 精度切り替えシミュレーションと母集団の推定

クラスタ C_i (i=1,2,…,k)それぞれについて，n_i 分のイタレーションの詳細シミュレーションを行う．このモデル図を図 2 に示す．

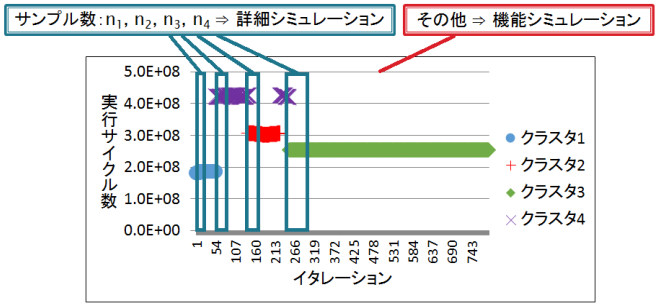

図 2 精度切り替えシミュレーションモデル図

シミュレーション後，クラスタ C_i (i=1,2,…,k)に所属するイタレーションの個数を N_i (i=1,2,…,k)とし，式 2 で全体の推定全実行サイクル数を計算する．

$$推定全実行サイクル数 = \sum_{i=0}^{k} シミュレーション値_i \times \frac{N_i}{n_i} \quad (2)$$

3. コンパイラと協調したシミュレーション高速化のフレームワーク

本稿で提案しているシミュレーション高速化手法は，主にマルチコアアーキテクチャのシミュレーションを対象としており，並列化されたアプリケーションをベンチマークアプリケーションとして実行する．そこで，並列化コンパイラと協調したシミュレーション高速化の一連の手順を自動化するフレームワークを提案する．図 3 に提案フレームワークの処理フローを示す．

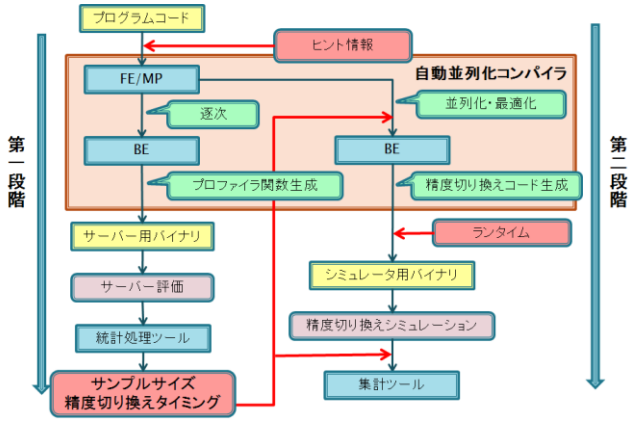

図 3 高速化手法フレームワークの処理フロー図

なおここでは，並列化コンパイラとして，当研究室で開発している OSCAR マルチグレイン自動並列化コンパイラ（以下 OSCAR） [7]を想定している．

このシミュレーション高速化のフレームワークは，二つの段階から構成される．

第一段階ではまずプログラムコードにサンプリングを行う箇所のヒント情報を与える．現段階ではサンプリング対象ループは手動の解析により決定している為，ヒント情報は直接サンプリング対象ループの箇所指定を想定している．将来的には OSCAR のプログラム解析機能を用いて自動的にサンプリング箇所を決定可能とする．その後プログラムの逐次コンパイルを行う．この時，ヒント情報を基に実機での実行プロファイルを取得する為のプロファイラ関数を生成する．その後実機上でアプリケーションを実行し，得た実行プロファイルを統計処理ツールに通すことでサンプル数を得る．同時にこの統計処理ツールは，クラスタリング結果とサンプル数を基に，どのイタレーションで詳細シミュレーションと機能シミュレーションをそれぞれ行うかを表す精度切り替えタイミングを出力する．

次に第二段階では，プログラムコードを自動並列化コンパイラによって並列化する．この時，ヒント情報と第一段階で得たサンプルサイズ及び精度切り替えタイミングを受けて,精度切り替えコードが生成される．その後予め用意した精度切り替えコードのランタイムと共にコンパイルすることでシミュレータ用バイナリを生成し，精度切り替えシミュレーションを行う．最後に，シミュレーション結果と精度切り替えタイミングのクラスタ情報を基に，サンプリング対象ループの全イタレーションのシミュレーション結果を推定する．

4. 精度切り替え機能

本章では，イタレーション単位でシミュレーションモードを切り替える機能について述べる．

4.1. 精度切り替え機能の概要

精度切り替え機能とは，2.2 節で述べた 2 種類のシミュレーションモードを相互に切り替える機能である．シミュレーションモードの切り替えは，アーキテクチャシミュレータの想定 OS に精度切り替えシステムコールを新設し，プログラム中の任意の場所でシステムコールを呼び出すことで行う．この時，2 つのシミュレーションモードは再現要素に差異がある為，全てのプロセッサにおいてモードを統一する必要がある．

本手法ではまず，ベンチマークアプリケーションの精度切り替えポイントでシステムコールを呼び出したプロセッサは，一旦 CPU コア（図中の P0-Pn）を停止し，命令が読めない状態にする．これを最後のプロセッサがシステムコールを呼び出すまで行い，最後のプロセッサがシステムコールを呼び出した時に全てのプロセッサのシミュレーションモードを一斉に切り替え，CPU を動き出させる．

4.2. 精度切り替えコード

精度切り替えタイミングは，詳細・機能の回転数を交互に持つ次のような配列の形で指示する．

```
int sim_count[] = {2,128,1,…};
```

上記の配列を受けて回転数計算を行い，適切なタイミングで精度切り替えシステムコールを呼び出す関数をランタイムで用意し，すべての並列化されたサンプリング対象ループの冒頭に挿入する．

5. 性能評価

5.1. 評価アプリケーション

5.1.1. equake

equake とは，SPEC(The Standard Performance Evaluation Corporation)によるベンチマーク群である SPEC CPU 2000 に含まれるベンチマークアプリケーションの一つであり，メッシュ状にモデリングした地形を伝わる地震波の影響をシミュレーションするプログラムである．今回の評価を行うにあたり，行列に対するベクトル演算を行う部分で，プログラム中で大きな割合を占める smvp という関数から並列性を抽出する為，ループ構造を再構築したコードを用いた．図 5 に equake のプログラム構造を表した図を示す．

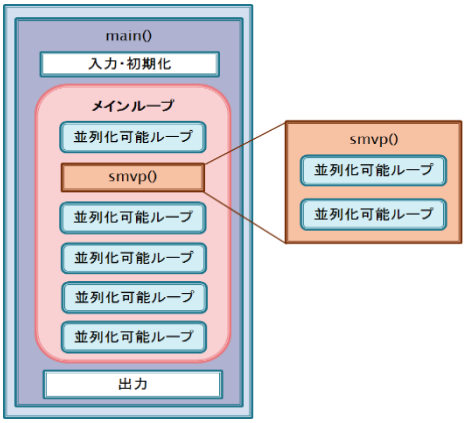

図 5 equake のプログラム構造

equake では，処理の大半を占め，複数の並列化可能ループを内包する外側の大きなメインループをサンプリング対象ループとする．

5.1.2. MPEG2 エンコーダ

MPEG2 エンコーダはメディア処理の性能評価に使われる MediaBench に含まれるベンチマークアプリケーションで, MPEG2 の規格に沿った動画圧縮処理を行うプログラムである．メディアアプリケーションは入力データの違いが計算量に大きく影響を与えることが多い．図 6 に MPEG2 エンコーダのプログラム構造を示す．

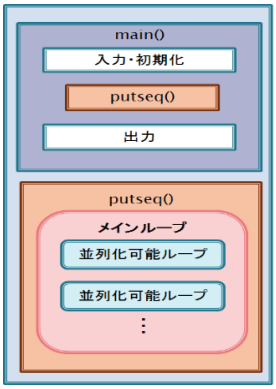

図 6 MPEG2 エンコーダのプログラム構造

MPEG2 エンコーダでは，サンプリング対象ループを putseq()関数のメインループとする．

5.2. 評価環境

実機によるプロファイルの取得を行ったサーバーの仕様を表 1 に，シミュレーションを行ったアーキテクチャの仕様を表 2 にそれぞれ示す．

表 1 実機プロファイル取得サーバーの仕様

CPU	Xeon X5670
CPU Clock	2.93GHz
L2 Cache	12MB
Native Compiler	gcc version 4.3.3
OS	Debian GNU/Linux 6.0.6

表 2 シミュレーションアーキテクチャの仕様

Instruction Set	SPARC V9
CPU コア数（PE）	1, 4, 8, 16
L1 Cache	32KB
L1 Cache Latency	1 clock cycle
L2 Cache	512KB
L2 Cache Latency	4 clock cycle
Main Memory	1GB
Main Memory Latency	60 clock cycle
Cache Protocol	MOESI

5.3. 評価結果

5.3.1. サンプリング結果

まず，equake，MPEG2 エンコーダそれぞれの実機プロファイルにクラスタリングを施した図を図 7 と図 8 に示す．

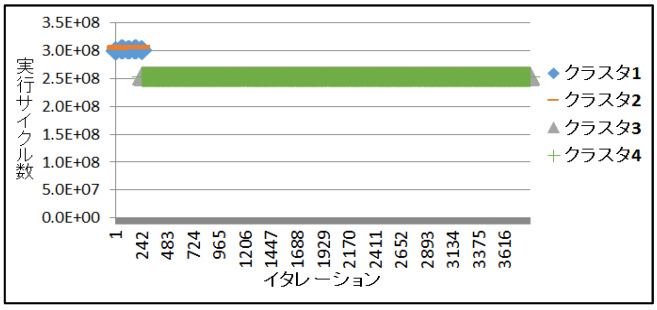

図 7 equake のクラスタリング結果

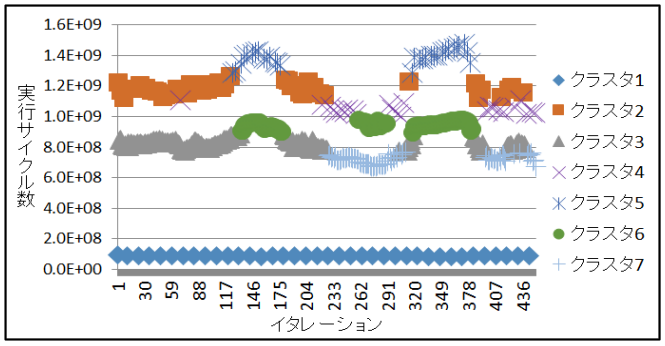

図 8 MPEG2 エンコーダのクラスタリング結果

次に，それぞれのクラスタ統計手法を施し，サンプル数とサンプルイタレーションを算出した結果を表 3, 表 4 に示す．この時，統計手法における実測実行サイクル数と推定実行サイクル数の許容する誤差 ε は 5% とした．

表 3 equake のサンプル情報

クラスタ番号	1	2	3	4
サンプル数	1	1	1	1
サンプル イタレーション	1	10	251	252

表 4 MPEG2 エンコーダのサンプル情報

クラスタ番号	1	2	3	4
サンプル数	2	2	2	2
サンプル イタレーション	1, 11	2, 5	3, 4	68, 221
クラスタ番号	5	6	7	
サンプル数	2	1	2	
サンプル イタレーション	125, 128	135	228, 229	

5.3.2. 精度切り替えシミュレーション結果

得たサンプル情報を用いて，精度切り替えシミュレーションを行った結果について示す．並列実行時のコア（PE）数は 1, 4, 8, 及び 16 とした．全イタレーションの実測全実行サイクル数と，精度切り替えシミュレーションによる推定実行サイクル数の誤差率を図 9 に示す．

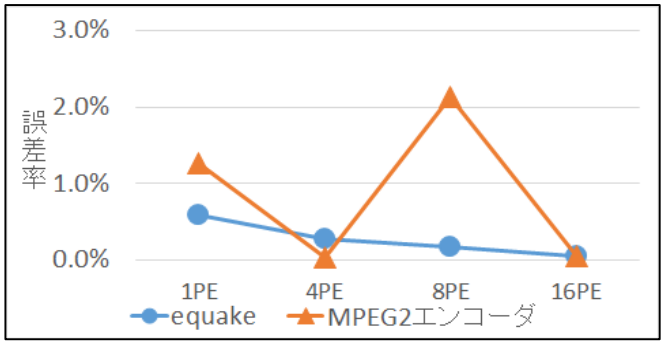

図 9 推定実行サイクル数の誤差率

equake の誤差率は 1PE で 0.59%，4PE で 0.27%，8PE で 0.17%，16PE で 0.04%となり，高い精度で実行サイクル数を推定できていることがわかる．MPEG2 エンコーダについても誤差率は 1PE で 1.26%，4PE で 0.03%，8PE で 2.13%，16PE で 0.04%となり，許容する誤差 5%に収まる十分な誤差率で実行サイクル数が推定できている．

これにより，equake と MPEG2 エンコーダについて，今回対象とした粒度の実行サイクル数の挙動は，アーキテクチャやコア数にはほとんど依存せず一様であることが分かり，逐次実行のプロファイルから得たサンプル情報を用いて，複数 PE のシミュレーションにおいて，一部の詳細シミュレーション結果からループ全体の結果を推定することが可能であることを示した．

5.3.3. 速度向上率

本高速化手法によって得た速度向上率について述べる．詳細シミュレーションと機能シミュレーションの実行速度の比率は，ベンチマークアプリケーションの詳細シミュレーション時のキャッシュミス率によって変動する．キャッシュミス率が高いアプリケーションほど，詳細シミュレーション時の 1 命令にかかるクロック数が増加し，機能シミュレーションとの速度比率が大きくなる．equake，MPEG2 エンコーダそれぞれの 1PE における詳細実行時の平均キャッシュミス率を表 5 に示す．

表 5 1PE 詳細実行時の平均キャッシュミス率

equake	MPEG2 エンコーダ
33.79%	2.58%

equake では機能シミュレーションは詳細シミュレーションの約 800 倍，MPEG2 エンコーダでは約 150 倍高速となっており，得た速度向上率を図 10 に示す．

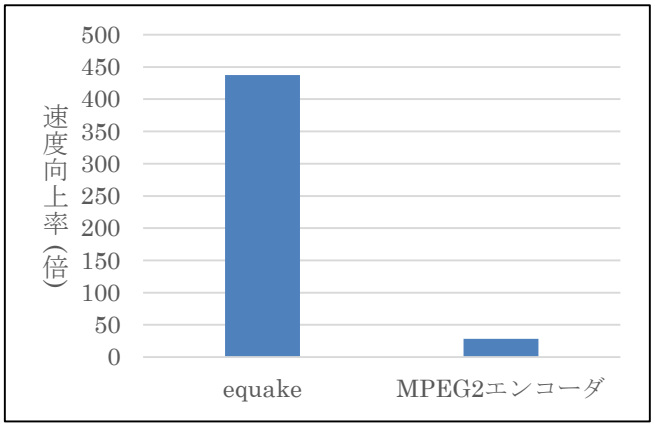

図 10　速度向上率

equake では 437 倍の速度向上率，MPEG2 エンコーダでは 28 倍の速度向上率を得た．MPEG2 エンコーダの速度向上率が equake に比べて低い理由は，キャッシュミス率の違いによる機能シミュレーションの速度向上比だけでなく，プロファイルの偏差が大きいためにサンプル数が大きくなっていること，ループの全イタレーション数が equake に比べて少ない為に全体の詳細シミュレーションの比率が大きくなっていることが挙げられる．

6. まとめ

本稿では，コンパイラと協調して精度を切り替えるマルチコアアーキテクチャシミュレータの統計的手法を用いた高速化手法について述べた．サンプリング対象ループの実機での逐次実行プロファイルから，許容する誤差の範囲内で全実行サイクル数を推定可能なサンプルイタレーションを抽出し，サンプルイタレーションのみの詳細シミュレーションの結果から全体の結果を推定することで，少ないシミュレーション時間で高精度なシミュレーションが可能となった．この時用いた x-means クラスタリングと統計手法の組み合わせにより，人間の手による解析を必要とせず，自動で最適なサンプルを算出することを可能とした．また，コンパイラと協調した高速化の手順を自動化するフレームワークと，精度切り替えのインターフェースを提案した．今回，プロファイルの挙動が一様な科学技術計算と，挙動の偏差が大きいメディアアプリケーションの，それぞれのベンチマークに対して本高速化手法を用いた評価を行った．結果，精度を十分高く保ちながら，equake では 1 コアで 1 年以上かかるシミュレーションを約 1 日に，MPEG2 エンコーダでは 1 コアで 1 か月程度かかるシミュレーションを 1 日に，それぞれ短縮することができた．

謝辞 本研究の一部は科研費若手研究（B）23700064 の助成及び，経産省グリーンコンピューティングシステム研究開発により行われた．

文　献

[1] Erez Perelman el al, “Using SimPoint for Accurate and Efficient Simulation”, Proc. of the 2003 ACM SIGMETRICS, pp. 381-319, 2003

[2] Thomas F. Wenishch et al, “Sim-Flex : Statistical Sampling of Computer System Simulation”, IEEE Micro, vol. 26, no. 4, pp. 18-31, July-Aug., 2006

[3] Trevor E. Carlson el al, “Sampled Simulation of Multi-Threaded Applications”, Proc. of ISPASS, pp. 2-12, April, 2013

[4] Paul D.Bryan el al, “Accelerating Multi-threaded Applicaction Simulation through Barrier-Interval Time-Parallelism”, Proc. of MACOTS,　pp. 117-126, Aug., 2012

[5] 石岡恒憲，“x-means 法改良の一提案 –k-means 法の逐次繰り返しとクラスターの再結合–”，計算機統計学 第 18 巻 1 号，no1，2006.

[6] 加藤洋一，新版 QC 入門講座 サンプリングと抜取検査，日本規格協会，2000.

[7] 笠原博徳，小幡元樹，石坂一久，”共有マルチプロセッサシステム上での粗粒度タスク並列処理，” 情報処理学会論文誌 Vol.42，no.4，Apr 2001.