Vol.2011-ARC-196 No.14
2011/7/27

科学技術計算プログラムの構造を利用したメニーコアアーキテクチャシミュレーション高速化手法の評価

石塚 亮[†1] 阿部洋一[†1] 大胡亮太[†1]
木村啓二[†1] 笠原博徳[†1]

本稿ではキャッシュやパイプラインまでシミュレーションする詳細シミュレーションと命令実行のみの高速な機能シミュレーションの両方を用いたシミュレーション精度切り替えによるメニーコアシミュレータの高速化手法を提案する．本手法はメニーコアシミュレータ上で並列化プログラムを実行することを前提としており，このプログラムの一部のみを詳細シミュレーションを行うことにより高速化を図る．このとき，詳細シミュレーションを行うサンプリング部分を実機での逐次実行プロファイル情報とプログラム構造から判断し，その分量を統計的手法により決定する．本手法を比較的規則性の高い科学技術計算である SPEC CPU 95 の TOMCATV，SWIM で及び SPEC CPU 2000 の ART，EQUAKE を用いて統計学的に算出したサンプリングサイズの値を堺に，実行サイクルが収束していくことを示した．これにより，評価したところ，64 コアかつ精度切換えを想定したシミュレーションで，各アプリケーションにおいて，誤差 5%の範囲で約 100 倍の高速化が可能であることを示した．

1. はじめに

コンピュータアーキテクチャの研究において, アーキテクチャシミュレータは非常に大きな役割を担っている．しかしながら, ソフトウェアによるシミュレーションは，評価において実機の約 5000〜10000 倍ほどの多大な時間要する．複数のプロセッサコアを有するマルチコア及びメニーコアのアーキテクチャシミュレーションを行う際にこのシミュレーション時間の増大がさらに顕著になり，現在のコンピュータアーキテクチャ研究の大きな障害となっている．[11)]

シングルプロセッサのシミュレーション評価に関しては，以前からプログラムの一部のみ評価し，その際の IPC を比較するという方法が採られてきた．しかしながら，マルチコアにおける並列アプリケーションの評価においてはプログラムの速度向上率が重要であり，プログラム部分の IPC 比較は適さず，シミュレーションそのものの高速化が求められる．

アーキテクチャシミュレーションの高速化に関しては，高速で精度の高いシミュレーション手法についての研究が注目されている．これらの研究には FPGA によりテストベッドを構築するもの[1)] シミュレータを並列化するもの[2)3)4)] 統計的手法を用い，詳細シミュレーションを行う部分を限定するもの[5)6)7)8)] シミュレーションをトレース採取とタイミング計測を行う 2 段階に分ける手法[12)] 等が行われている．特に，プログラムの一部分を詳細にシミュレーションするサンプリング実行により高速化を図る手法として SimFlex[5)]，SimPoint[6)]，MCCS 及び CBFS[7)] が挙げられる．

SimFlex[5)] は統計的手法により，詳細シミュレーションを行うサンプル量を決定することが特徴として挙げられる．SimPoint[6)] は基本ブロック間の動的な実行経路情報とその頻度に着目し，詳細にシミュレーションを行うプログラムを抽出する．SimFlex[5)] 及び SimPoint[6)] は逐次プログラム用の高速手法であるが，並列化プログラムを対象としたマルチコア・メニーコアシミュレータの高速化手法として，MCSS 及び CBFS[8)] が挙げられる．MCCS は詳細シミュレーションとモンテカルロ法による CPI 推定を交互に繰り返す．CBFS は並列化された科学技術計算プログラムのループ 1 回転分の CPI 推定をカーブフィッティング法により推定する．CBFS は本稿の推定手法と同様にメインループのイタレーションに着目しているが，ループイタレーション間のコスト変動を考慮していない．

今回評価するシミュレータの高速化手法の特徴は，シミュレーションに用いる並列化プログラムの構造に注目することにある．すなわち，並列化されたループ，あるいは並列化対象部分を囲むループのイタレーションの一部を実機上の逐次実行時プロファイルに基づいた統計的手法を用いて期待する誤差に収まる範囲でサンプリング実行する．また，今回の評価アプリケーションは科学技術計算をターゲットとしている．これらは同様の計算を繰り返し行うアプリケーションであるため，実行サイクル数が正規分布となることが期待できる．

本手法では，まず並列化する前の逐次プログラムを任意の実機上で実行する．その際，サ

†1 早稲田大学 基幹理工学部 情報理工学科
Dept. of Computer Science and Engineering, Waseda University

ンプリング対象となるループの 1 イタレーション毎の実行サイクル数を計測する．計測したイタレーション毎のコストから統計的手法により，総実行サイクルの推定値が期待する誤差に収まる最小のイタレーション数を算出する．そのデータを基にサンプリング対象のループを算出したイタレーション回数だけ詳細にシミュレーションを行い，実行結果の確認を行うために残りのイタレーションは簡易なシミュレーションを行う．このようなシミュレーション 2 段階化による高速化手法として TPTS[12] が挙げられるが，TPTS の 1 段階目はトレース採取であり，本手法の実機プロファイル取得とは異なる．

以下，2 章では全て詳細にシミュレーションしたときの全実行サイクル数を推定する手法について，3 章では評価アプリケーションについて，4 章ではそれぞれの評価結果，最後の 5 章でまとめをそれぞれ述べる．

2. 実行サイクル数推定手法

本手法ではプログラムにおけるイタレーション回数のサンプリング対象となるループに注目する．このループに対して，全イタレーション数のうち一部を詳細にシミュレーションし，残りを高速かつ簡易なシミュレーションを行うことによって高速化する．その際，詳細にシミュレーションするイタレーション数を明らかにする必要がある．

本章では並列化される前の逐次プログラムを任意の単一プロセッササーバ上で実行したデータを基に詳細にシミュレーションするイタレーション数を算出し，一部のみ詳細にシミュレーションした実行サイクル数から全て詳細にシミュレーションしたときの実行サイクル数を期待する誤差の範囲で推定する手法を述べる．

本稿が対象とする科学技術計算プログラムの多くは，複数の並列化可能ループを 1 つのループが囲む構造をとっている．本稿ではこの並列化ループを内包するループをサンプリング対象とする．ただし，並列化ループを内包するループが収束ループの場合は並列化ループそのものをサンプリング対象とする．

2.1 イタレーション回数算出手法

まず，サンプリング対象のループに 1 イタレーション毎の実行サイクル数を計測するコードを挿入し，任意の実機上で逐次実行させる．このプロファイルの結果，ループ途中でプログラムの挙動が変わるプログラムの場合，挙動が相違する範囲でループを分割し，それぞれのサンプリングサイズを算出し，シミュレーションを行う．

今回はアーキテクチャの差異を確認するため，Intel 社の Xeon と IBM 社の Power5 上でそれぞれ 1 コアを使用し評価を行った．3.1 節で述べる，SPEC CPU 95 ベンチマークの TOMCATV のサンプリング対象ループのイタレーション毎の実行サイクル数を図 1，図 2 示す．

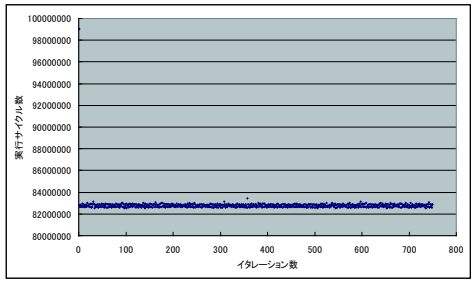

図 1 XEON で計測した TOMCATV のイタレーション毎の実行サイクル数

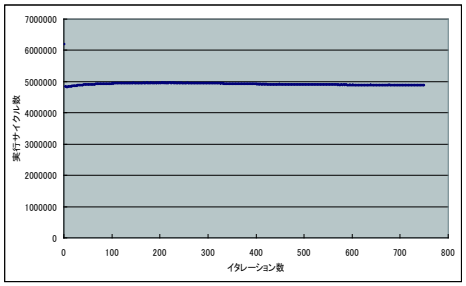

図 2 POWER5 で計測した TOMCATV のイタレーション毎の実行サイクル数

図 1，図 2 から，キャッシュやパイプラインの状態が安定していない最初の数イタレーションは実行サイクル数の変化は大きいが，すぐに，イタレーション毎の実行サイクル数の変化は小さくなり，一定の値を取ることがわかる．また，実機の違いによってイタレーション毎

の実行サイクル数に大きな差が無いことも確認できる．この結果から，サンプリング対象となるループの実行サイクル数はプログラム依存であり，一部の実行サイクル数から全体の実行サイクル数を推定できる可能性が高いことを示している．

このように実機から取得した実行サイクル数より統計的手法を利用し，期待する誤差の範囲で全実行サイクル数が推定可能となる，詳細にシミュレーションを行うイタレーション回数（サンプル回数）を決定する．取得したイタレーション毎の実行サイクル数の標準偏差と平均値を算出し，許容する誤差（信頼度）を決定する．例えば誤差 5%以下と目的とするならば，信頼度は 0.05 となる．これらの値を用いて，以下の式で計算する．

$$\lfloor \text{サンプル回数} \rfloor \geq (\frac{\text{上側 P\%点}}{\text{信頼度}} \times \frac{\text{標準偏差}}{\text{平均値}})^2 \quad (1)$$

この式から得られたサンプル回数の値が，詳細にシミュレーションするイタレーション回数となる．3.4 節で述べる SPEC CPU 2000 の EQUAKE のように，同一ループイタレーションであってもその回転数の範囲でコスト変動の挙動が異なる場合は，挙動が相違する範囲ごとにサンプリングを行うこととする．

2.2 推定実行サイクル数計算式

決定したサンプル回数の数だけ，サンプリング対象のループを詳細にシミュレーションする．このデータから全て詳細にシミュレーションした際の実行サイクル数を以下の式から算出する．

$$\text{推定全実行サイクル数} = \text{詳細シミュレーションサイクル数} \times \frac{\text{全イタレーション回数}}{\text{サンプル回数}} \quad (2)$$

3. 評価アプリケーション

本章では今回シミュレーションをするにあたって使用したアプリケーションの特徴と並列化した際のプログラムについて述べる．これらのプログラムの並列化には OSCAR 自動並列化コンパイラ[9] を用いた．

3.1 TOMCATV

tomcatv は SPEC CPU 95 に含まれるプログラムの 1 つであり，vectorized mesh を生成するプログラムである．tomcatv はサブルーチンや関数を持たないプログラムで，実行時間の大部分は 1 つのメインループによって占められている．このメインループにはイタレーション間の並列性はないが，内部は並列実行可能ループで構成される．TOMCATV のプログラム構造を図 3 に示す．本評価ではこのメインループをサンプリング対象とした．

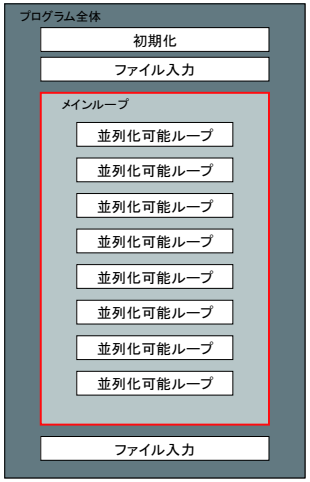

図 3 TOMCATV のプログラム構造

3.2 SWIM

swim は SPEC CPU 95 に含まれるプログラムの 1 つであり，Shallow water 方程式の求解プログラムである．swim は 1 つのメインループがあり，メインループから呼ばれるサブルーチンが CALC1，CALC2，CALC3，および CALC3Z の 4 つ存在する．また，これら 4 つのサブルーチンのうちの CALC3Z は初めの 1 回転目のみ呼ばれる．SWIM のプログラム構造を図 4 に示す．本評価ではこのメインループをサンプリング対象とした．

3.3 ART

ART は SPEC CPU 2000 に含まれるプログラムの 1 つであり，ニューラルネットワークを用い画像認識を行うアプリケーションである．トレーニングデータを用いて学習した後，スキャンデータの中からトレーニングデータと一致するものを見つけ出す．ART のプログラム構造を図 5 に示す．ART には，train_match 関数及び，match 関数が存在し，それぞれ 1 つのメインループが存在するが，回転数が不定である．本評価ではメインループの内部に存在する並列化可能ループをサンプリング対象とした．

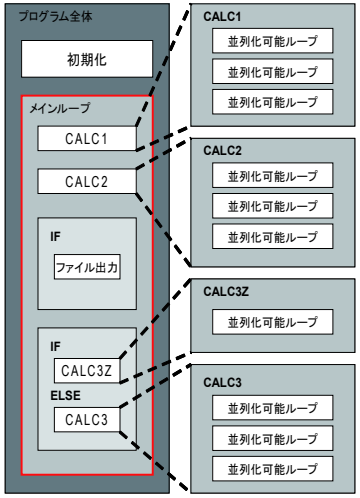

図 4 SWIM のプログラム構造

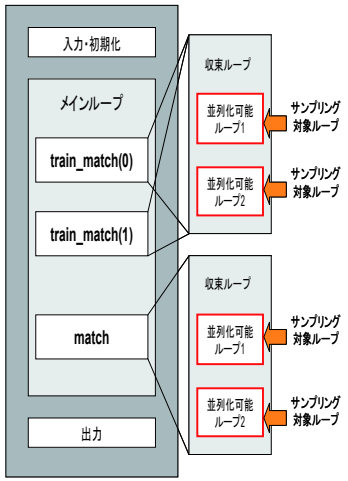

図 5 ART のプログラム構造

3.4 EQUAKE

EQUAKE は SPEC CPU 2000 に含まれるプログラムの 1 つであり，盆地のような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．EQUAKE のプログラム構造を図 6 に示す．EQUAKE は 1 つのメインループを持つ．このメインループのプロファイル結果図 7 を参照すると，250 回転目とそれ以降でプログラムの挙動が大きく異なっていることがわかる．このような場合，実行が大きく変動する部分でサンプリングの区間を切り分け評価する．本評価ではメインループを 250 回転以前と以降に分離し，それぞれをサンプリング対象とした．

3.5 サンプリングするイタレーション回数

Xeon における各アプリケーションのサンプリング対象ループの回転数，標準偏差，平均値，サンプル数を表 1 に示す．2 章で述べた，プログラムの逐次実行の値と，式 (1) よりサンプル数を算出した．

表 1 各アプリケーションの回転数，標準偏差，平均値，サンプル数

アプリケーション	回転数	標準偏差	平均値	サンプル数
TOMCATV	750	605.7×10^3	82.8×10^6	1
SWIM	900	362.2×10^3	82.8×10^6	1
ART(train_match ループ 1)	3209	3.67×10^5	8.14×10^6	2
ART(train_match ループ 2)	554	7.67×10^4	2.76×10^6	2
ART(match ループ 1)	19597	1.30×10^6	6.28×10^6	9
ART(match ループ 2)	4700	2.41×10^4	4.27×10^5	3
EQUAKE(250 回転まで)	250	2.28×10^7	2.68×10^8	12
EQUAKE(250 回転以降)	3605	1.73×10^5	2.30×10^8	1

4. 評価結果

本章では 3 章で挙げた各アプリケーションのシミュレーションしたイタレーション数とそこから算出した推定実行サイクル数の推移を示す．また，評価時間の制約から全イタレーションの詳細実行は行っていないが，今回シミュレーションできた最も多いイタレーション

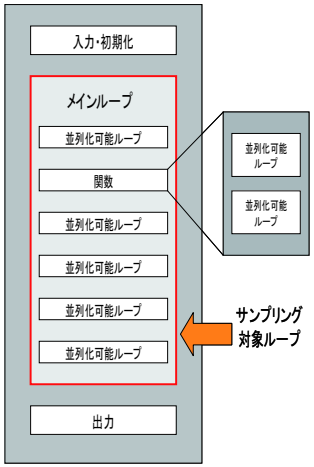

図 6 EQUAKE のプログラム構造

数から算出した推定実行サイクル数を基準としたそれぞれの誤差も示す．以下これを暫定誤差と述べる．なお，暫定誤差の算出式を以下に示す．

$$\text{誤差} = \frac{\text{推定全実行サイクル数-全実行サイクル数}}{\text{全実行サイクル数}} \times 100 \qquad (3)$$

また，今回シミュレーションするアーキテクチャの仕様を表 2 に示す．

表 2 シミュレーションアーキテクチャ仕様

命令セット	SPARC V9
コア数	1，16，32，64
L1cache	32kB(I/D)
L1cache latency	1
L2cache	64kB，512kB
L2cache latency	4
memory latency	60
キャッシュ構成	L2 スヌープ

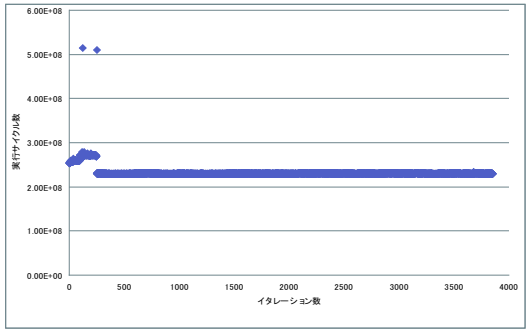

図 7 EQUAKE のプロファイル結果

なお，使用コア数が 32 コアで L2cache サイズが 512kB アーキテクチャで TOMCATV，SWIM のデータが全てキャッシュに乗る．

4.1 TOMCATV

TOMCATV は本来，ループを 750 回転するプログラム構造であるが，全て詳細にシミュレーションしようと試みると 64 コアで推定 4，5ヶ月ほどかかり現実的に不可能である．よってループのイタレーション数を変化させた際における推定実行サイクル数と暫定誤差の推移を評価の対象とする．本評価では，シミュレーションするイタレーション数を 1 回，5 回，10 回，20 回，40 回と変化させた．さらに，TOMCATV に対してキャッシュ最適化[10] を適用した時の推定実行サイクル数も同様の方法で算出した．

TOMCATV のシミュレーションするイタレーション数を変化させた時の推定実行サイクル数の推移と暫定誤差の評価結果を図 8，図 9 にそれぞれ示す．なお，暫定誤差は 40 回転の推定実行サイクル数を実行サイクル数と仮定し，算出した．

図 8 のキャッシュサイズが 64kB の 32pe と 64pe のグラフや，図 9 の 32pe や 64pe のグラフより，コア数が増加するにつれ，誤差の幅が大きくなっていることがわかる．しかしながら，シミュレーションするイタレーション数が増加するにつれ誤差は小さくなっていく．さらに図 10，図 11 に 64 コアでシミュレーションした際の推定実行サイクルと暫定誤差の

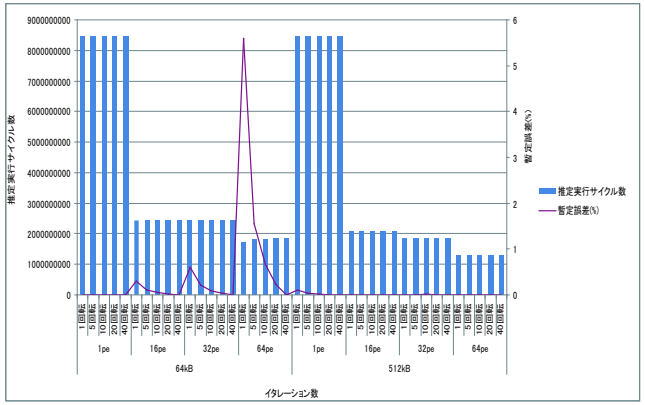

図 8 TOMCATV の推定実行サイクル数と暫定誤差の推移

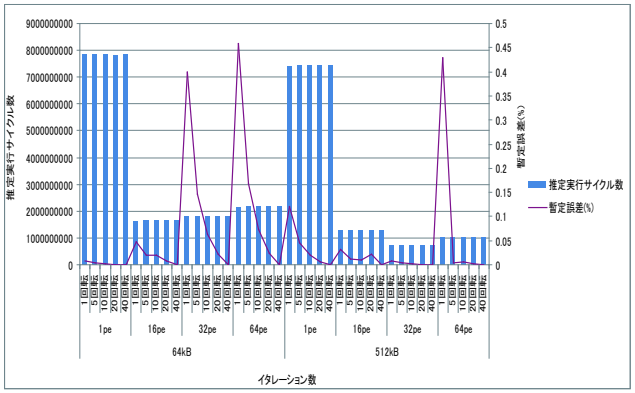

図 9 TOMCATV(キャッシュ最適化) の推定実行サイクル数と暫定誤差の推移

推移をそれぞれ示す．

図 10，11 の暫定誤差の推移から，サンプリングサイズが 1 ということもあり，誤差は殆どないという結果を得た．

4.2 SWIM

SWIM は元のプログラムでは，ループを 900 回転するプログラム構造であるが，TOMCATV 同様，全て詳細にシミュレーションを行うことは現実的ではない．よってループのシミュレーションするイタレーション数を変化させた時の推定実行サイクル数と暫定誤差の推移を評価の対象とする．4.1 節の TOMCATV と同様に，イタレーション数は 1 回，5 回，10 回，20 回，40 回と変化させた．なお，第 3 章で述べたように SWIM は 1 回目だけ呼び出される関数が異なる．そこで，推定実行サイクル数を以下のように変形させた．

$$\begin{aligned}\text{推定実行サイクル数} = (&\text{詳細シミュレーションサイクル数} \\ &-1\text{ 回転目の詳細シミュレーションサイクル数}) \\ &\times \frac{899}{\text{サンプル数-1}} + 1\text{ 回転目の詳細シミュレーションサイクル数}\end{aligned} \quad (4)$$

さらに，の SWIM のプログラムをキャッシュ最適化[10] を適用した時の推定実行サイクル数も同様の方法で算出した．

SWIM のシミュレーションするイタレーション数を変化させた時の推定実行サイクル数と暫定誤差の推移を図 12，13 にそれぞれ示す．なお，暫定誤差は 40 回転の推定実行サイクル数を実行サイクル数と仮定し算出した．

図 12，13 の 32pe，64pe のグラフより，SWIM についても，シミュレーションするイタレーション回数が少なく，コア数が増加すると誤差の幅が大きくなることがわかる．また，64 コアでシミュレーションした際の，推定実行サイクル数と暫定誤差の推移を図 14，15 にそれぞれ示す．

14，15 の暫定誤差の推移から，SWIM も TOMCATV 同様，イタレーション数を変化させてもサンプリングサイズ以上シミュレーションをしても，推定実行サイクル数は収束しており，誤差は非常に小さい．コア数の変化やキャッシュサイズの変化，及びキャッシュ最適化の有無に対しても誤差は微少である．仮にシミュレーションするイタレーション数を増加させても結果は同様であると考えられる．

4.3 ART

ART の実行時間の 9 割以上を占める，train_match 関数，match 関数はループ回数が不定であるため，表 1 で示したように train_match 関数，match 関数のメインループ内部に存在する，それぞれの並列化可能ループに対してシミュレーションを行った．ART のシミュ

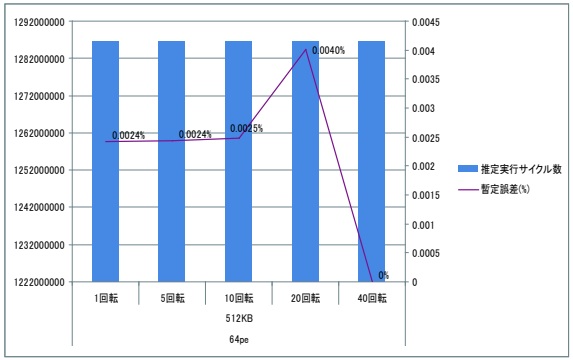

図 10 64 コアでシミュレーションした際の TOMCATV の推定実行サイクル数と暫定誤差の推移

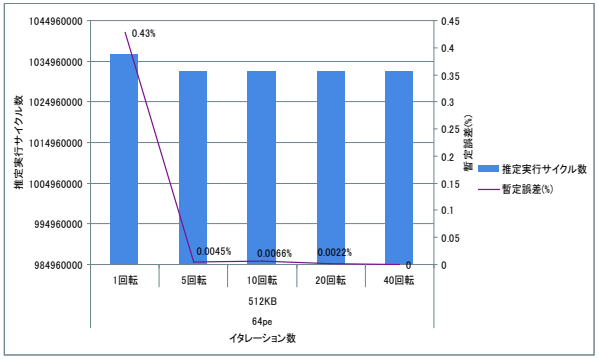

図 11 64 コアでシミュレーションした際の TOMCATV(キャッシュ最適化) の推定実行サイクル数と暫定誤差の推移

レーションするイタレーション数と推定実行サイクル数の暫定誤差の推移の評価結果を図 16 に示す．なお，暫定誤差は 25 回転シミュレーションしたデータを基準に算出した．

図 16 の暫定誤差の推移より，シミュレーションするイタレーション数が少ないほど推定実行サイクル数に大きな誤差が出ることがわかる．特に PE 数が多い場合，それが顕著に現れる．しかしながら，サンプリングサイズ以上シミュレーションを行うことにより推定実行サイクル数が収束していくことがわかる．より明確に示すため，サンプリング回数が大きい match 関数並列化ループ 1 を 64 コアでシミュレーションした際の推定実行サイクル数と暫定誤差の推移を図 17 に示す．

図 17 の 9 回転前後の推定実行サイクル数と暫定誤差より，シミュレーションするイタレーション数がサンプリングサイズ未満の場合，推定実行サイクル数が大きく異なるが，サンプリングサイズ以上シミュレーションを行うと，推定実行サイクル数に大きな変化はなくなり，値が収束していくことがわかる．さらにシミュレーション回数を増加させた場合においても同様の結果であると予想される．

4.4 EQUAKE

EQUAKE はメインループを 250 回転までと 250 回転以降に分け，それぞれに対し高速化手法の評価を行った．表 1 で示したように 250 回転までは 12 イタレーション，250 回転以降は 1 イタレーションを基準としてシミュレーションを実行している．EQUAKE のシミュレーションするイタレーション数と推定実行サイクル数の暫定誤差の推移の評価結果を図 18 に示す．なお，誤差は 15 回転シミュレーションしたデータを基準に算出した．また，評価時間の関係上 32pe までのデータとなっている．

図 18 の 16pe，32pe の推定実行サイクル数と暫定誤差の推移より，EQUAKE の評価においても，シミュレーションするイタレーション数が少なく，PE 数が多いほど推定実行サイクルに大きな誤差が生じる．シミュレーションするイタレーション数を増やすにつれ，誤差が収束，すなわち推定実行サイクル数が収束していくことがわかる．

EQUAKE も ART 同様，より明確に示すためサンプリングサイズが大きい 250 回転前までの評価に関して，32 コアでシミュレーションした際のイタレーション数と推定実行サイクル数の暫定誤差の推移の評価結果を図 19 で示す．

図 19 の推定実行サイクル数の推移より，シミュレーションするイタレーション数を増やしていくにつれ，誤差が小さくなり，推定実行サイクル数が収束していくことがわかる．また，この評価によって回転数によりプログラムの挙動が大きく変化するプログラムにおいても，サンプリングする対象を分離することにより精度の高い高速化手法が適応可能であるこ

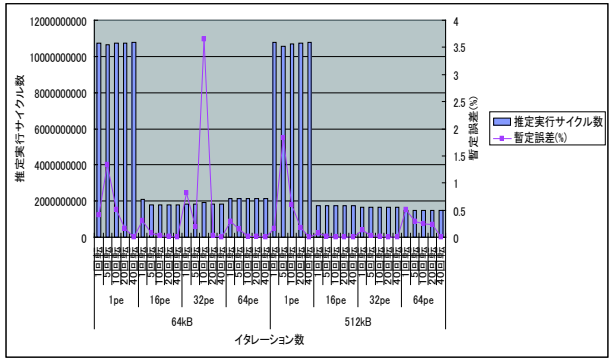

図 12 SWIM の推定実行サイクル数と暫定誤差の推移

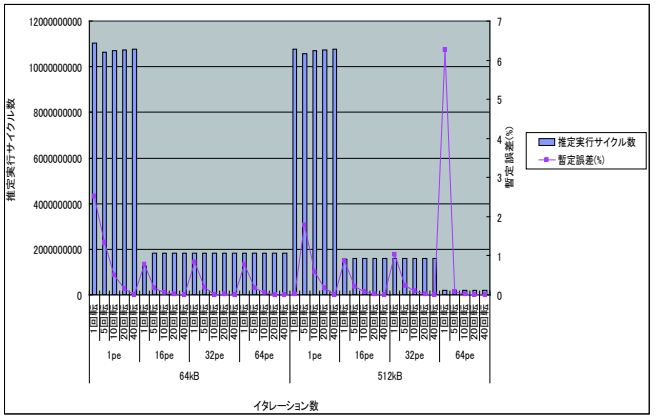

図 13 SWIM(キャッシュ最適化) の推定実行サイクル数と暫定誤差の推移

とを示した.

4.5 シミュレータ高速化

以上の結果より，コア数，キャッシュサイズ，キャッシュ最適化の変化，回転数によるプログラムの変化によらず，本稿によるサンプル対象ループのイタレーション数を縮減することにより，並列化プログラムの実行サイクル数を期待する誤差の範囲で見積もり可能なことが解った.

64 コアを想定したシミュレーションにおいて，TOMCATV のサンプリング対象ループ 40 回転の詳細シミュレーション時間は 216 時間であったが，機能シミュレーション時間は 1 時間 50 分と約 120 倍高速である．SWIM のサンプリング対象ループ 40 回転の詳細シミュレーション時間は 350 時間であり，機能シミュレーションでは 3 時間 10 分と約 107 倍ほど高速である．ART，EQUAKE においても機能シミュレーションは 100〜130 倍程度詳細シミュレーションより高速である.

本手法で示した，サンプリングサイズのみ詳細シミュレーションを行い，その他の部分を機能シミュレーションに切り替えた際，その高速化率は数十倍〜百倍程度となることが期待できる.

5. ま と め

本稿では，キャッシュやパイプラインまでシミュレーションする詳細シミュレーションと命令実行のみで高速な機能シミュレーションのシミュレーション精度切り替えによる，メニーコアシミュレータの高速化手法について述べた．サンプリング対象ループに着目し，期待する誤差の範囲内で全実行サイクル数を推定可能なイタレーション数を特定し，そのイタレーション数だけ詳細にシミュレーションを行うことにより，少ないシミュレーション時間で高精度の実行サイクル数推定が可能となる．また，回転数の範囲によりイタレーションコスト変動の挙動が異なるプログラムに対しても，それぞれの回転数の範囲に分けて評価することによって適応可能であることを示した．今回評価したアプリケーションである TOMCATV，SWIM，ART，EQUAKE で統計学的に算出したサンプリングサイズで，期待する誤差の範囲内に評価が可能となり，さらにシミュレーションするループのイタレーション数を増やすことで，誤差が収束し，推定実行サイクル数が収束していくことを示した．また，ここで得られたサンプル数はコア数，キャッシュサイズ，及びキャッシュ最適化の有無に変化があっても適用可能なことが解った．さらに，機能シミュレーションは詳細シミュレーションの 106 倍〜166 倍ほど高速であり，サンプリング対象のループを全体の一部のみ

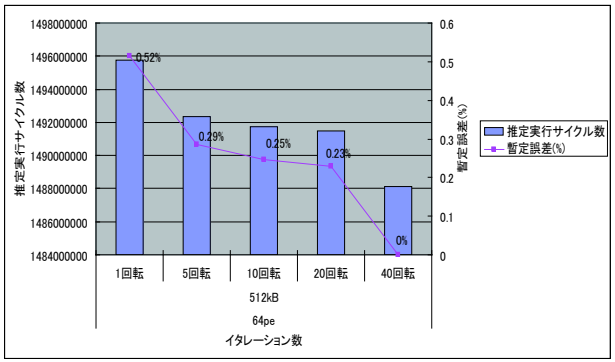

図 14 64 コアで評価した SWIM の推定実行サイクル数と暫定誤差の推移

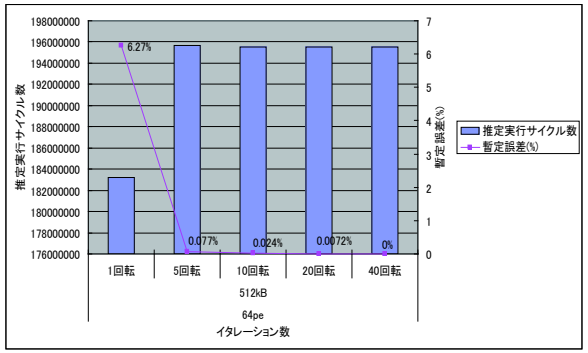

図 15 64 コアで評価した SWIM(キャッシュ最適化) の推定実行サイクル数と暫定誤差の推移

詳細シミュレーションすることにより，数十倍〜百倍程高速化が見込めることが解った．

今後は，動画像エンコーディング，デコーディングなどの入力によって繰り返し毎の挙動が大きく変化するアプリケーションを用いた評価，及び，OSCAR 並列化コンパイラ[9] との連携によるシミュレーションフレームワークの構築を行う予定である．

謝　　辞

本研究の一部は科研費 若手研究（B）23700064 の助成により行われた．

参 考 文 献

1) John Wawrzynek, Mark Oskin, Christoforos Kozyrakis, Derek Chiou, David A. Patterson, Shih-lien Lu, James C. Hoe, Krste Asanovic, “RAMP: Research Accelerator for Multiple Processors”, IEEE Micro, Vol. 27, Issue 2, March/April 2007
2) James Donald, Margaret Martonosi, “An Efficient, Practical Parallelization Methodology for Multicore Architecture Simulation”, IEEE Computer Architecture Letters, Vol. 5, 2006
3) 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 Vol46 No.SIG 12 (ACS 11) pp.84-97 (2005)
4) 松尾治幸，今福茂，大野和彦，中島浩 “共有メモリマルチプロセッサの分散シミュレータ Shaman の設計と実装” 情報処理学会論文誌 Vol44 No.SIG 1 (HPS 6) pp.114-127 (2003)
5) Thomas F. Wenishch, Roland E. Wunderlich, Michael Ferdman, Anastassia Ailamaki, Bavak Falsafi, and James C. Hoe, “Sim-Flex:Statistical Sampling of Computer System Simulation” Micro IEEE, Volume 26, Issue 4, pp.32-42, July-Aug, 2006
6) Erez PerelmanGreg HamerlyMichael Van Biesbrouck Timothy SherwoodBrad Calder “Using SimPoint for Accurate and Efficient Simulation” SIGMETRICS ’03, San Diego, California, USA. ACM 1-58113-664-1/03/0006, June 10—14, 2003
7) Manu Shantharm, Padma Raghavan, Mahmut Kandemir, “Hybrid Techniques for Fast Multicore Simulation”, Lecture Notes In Computer Science, Vol. 5704 Proc. of 15th Intl Euro-Par 2009 Parallel Processing, Aug. 22, 2009
8) Davy Genbrugge, Lieven Eechkout, “Chip Multiprocessor Design Space Exploration through Statistical Simulation”, IEEE Trans. on Computers, Vol 58, No. 12, Dec. 2009
9) H.Kasahara，H.Obata，K.ishizaka “Automatic Cooanse Grain Task Parallel Processing on SMP using Open MP” Proc,of 13th Intl.workshop on language and Compilers for Parallel Processing(LCPC’00) Aug.2000

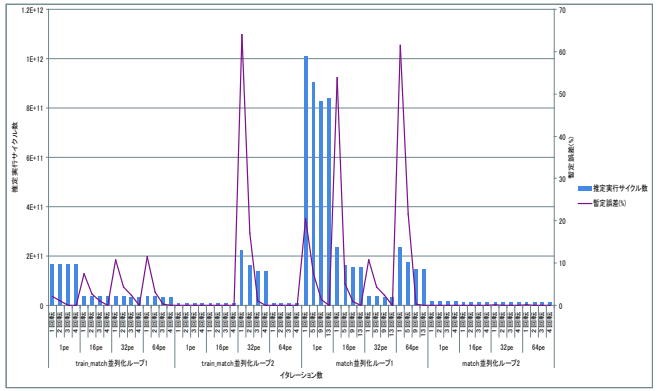

図 16 ART における推定実行サイクル数と暫定誤差の推移

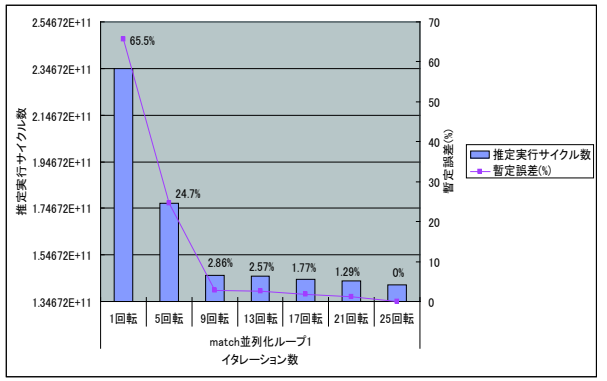

図 17 64 コアで評価を行った ART の推定実行サイクル数と暫定誤差の推移

10) K.ishizaka, M.Obata, H.Kasahara, “Coarse Grain Task Parallel Processing with Cache Optimization on Shared Memory Multi Processor” Proc. of Intl.Workshop on Language an Compilers for Parallel Processing (LCTC2001) Aug.2001
11) D. Burger, et al., “The Future of Architectual Simulation” IEEE Micro, Vol. 30, Issue 3, May/June, 2010
12) Sangyeun Cho, S. Demetriades, S. Evans, Lei Jin, Hyunjin Lee, Kiyeon Lee, M. Moeng, “TPTS: A Novel Framework for Very Fast Manycore Processor Architecture Simulation”, In Proc. of 37th International Parallel Processing (ICPP ’08), pp. 446-453, Sept., 2008

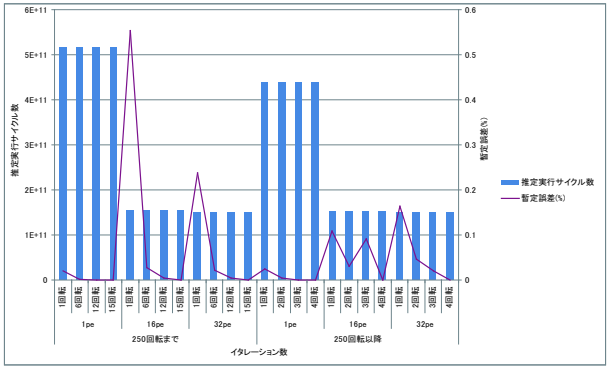

図 18 32 コアで評価した EQUAKE における推定実行サイクル数と暫定誤差の推移

Vol.2011-ARC-196 No.14
2011/7/27

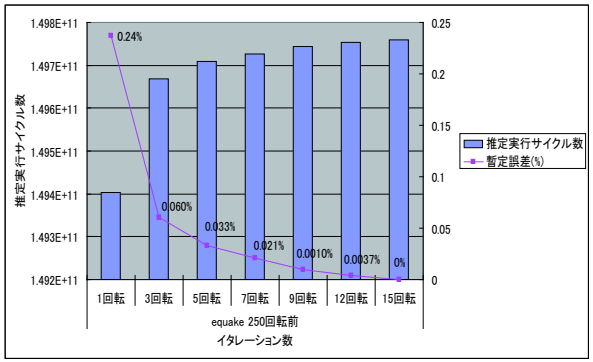

図 19 32 コアで評価した EQUAKE における推定実行サイクル数と暫定誤差の推移