科学研究費助成事業　　研究成果報告書

平成 ２６ 年　 ５ 月 ２９ 日現在

機関番号：３２６８９
研究種目：若手研究(B)
研究期間：2011～2013
課題番号：２３７０００６４

研究課題名（和文）プログラムの大域的構造を利用したメニーコア・シミュレーションの高速化に関する研究

研究課題名（英文）A Study of Acceleration Technique for Many-core Architecture Simulation Considering Global Program Structure

研究代表者
木村　啓二（Kimura, Keiji）

早稲田大学・理工学術院・教授

研究者番号：５０３１８７７１

交付決定額（研究期間全体）：（直接経費）　　3,300,000円、（間接経費）　　990,000円

研究成果の概要（和文）：本研究では、マルチコア・メニーコアのアーキテクチャシミュレーションにおいて、並列化されたアプリケーションをマルチコア上で実行するという前提の基、シミュレーションの精度を適宜切り替えながら、高速かつ高精度にシミュレーションを行う手法を提案する。
本手法を4つの異なる特性を持つアプリケーションを用い、16コアのマルチコアアーキテクチャを想定して評価した結果、最大443倍の速度向上を誤差0.52%で得ることができ、平均では218倍の速度向上を2.76%の誤差で得られることが確認できた。

研究成果の概要（英文）：A fast and high accuracy architecture simulation technique for multi-core and many-core processors are proposed in this study. By this proposed technique, an architecture simulator changes its precision and simulation speed appropriately under the assumption that a parallelized application is executed on a multi-core or a many-core.
The evaluation results with four applications each of which has different characteristics show the 16-core multicore simulation gives 443 times speedup within 0.52% error in maximum, and 218 times speedup within 2.75% error on average.

研究分野：総合領域

科研費の分科・細目：情報学・計算機システム・ネットワーク

キーワード：コンピュータアーキテクチャ　マルチコア　メニーコア　アーキテクチャシミュレーション　並列化アプリケーション

１．研究開始当初の背景

コンピュータアーキテクチャの研究には、ソフトウェアにより構築されたアーキテクチャシミュレータが非常に大きな役割を果たしてきた。これらのシミュレータの利用によりまだ存在しないコンピュータのアーキテクチャを容易に実験することが可能となった。しかしながら、ソフトウェアによるシミュレーションは実機の 5,000～10,000 倍の実行時間を要求する。これは、例えば実機上で実行時間が１分のシミュレーションに４日～7 日要することを意味する。この膨大なシミュレーション時間が、今後の発展が期待される数十～数千コア規模のメニーコア研究を行う際の大きな障害となっている。

このようなシミュレーション時間の増大を解決するため、これまでに様々な取り組みが国内外で行われてきた。単一 CPU シミュレーションの高速化手法として、パイプライン構造やキャッシュを考慮した高精度シミュレーションと命令実行のみの低精度シミュレーションを組み合わせる手法が提案されている。これらの方式ではシミュレーション精度を実用的な範囲に保ったままシミュレーション時間を大幅に削減できる。しかしながら、これらの研究を拡張してマルチコア上の並列プログラム実行のシミュレーション高速化を果たした研究は研究開始当初存在しなかった。

また、マルチコアやマルチプロセッサシステムのシミュレーションに関しては、マルチプロセッサ上の並列シミュレーションによる高速化が以前から提案されている。しかしながら、このような方法ではシミュレータ上で実行する対象プログラムの並列性やシミュレータを実行するマルチプロセッサの同期オーバーヘッドにシミュレータそのものの並列処理速度向上が律速されてしまい、場合によっては１CPU で通常のシミュレーションを行った場合より遅くなってしまう。

さらに、FPGA でアーキテクチャシミュレータを構築するアプローチがあるが、評価ボードを含めたシステム一式を構築及び維持するためのコストはソフトウェアシミュレータに比べて著しく高い。

上記のように、アーキテクチャシミュレータの高速化に対する高い欲求があり、様々な手法が提案されたが、マルチコア・メニーコアに対する高速かつ利便性の高い手法は存在しなかった。

２．研究の目的

本研究の目的は、マルチコア・メニーコアプロセッサのソフトウェアシミュレーションに要する膨大な時間を大幅に削減することである。この目的のために、並列化前のプログラムの実機での実行情報とループなどのプログラムの大域情報を利用し、プログラムの実行状況が均質な区間を必要最小限の分量だけ高精度のシミュレーションを行うことにより、シミュレーションの精度を保ったままシミュレーションの時間を短縮する。

３．研究の方法

本研究ではまず、できるだけ多くの実プログラムに対して調達可能な限りの計算サーバを用いて調査することにより手法の開発を行う。具体的には、各種プログラムの大域構造を同時並行的に調査すると共に、プログラムの各部分をどの程度実行すれば所望の精度を得ることができるか評価を行いつつ手法を確立する。

これと並行して、並列化コンパイラと連係するシミュレータの開発を行う。より具体的には、高精度シミュレーションを適用するプログラム部分とその分量の自動抽出手法のコンパイラに対する実装、及びそれらの情報をシミュレータに伝えるインタフェースの開発を行う。

以上、プログラム構造の解析結果に基づく詳細評価部分の特定手法、及び開発したコンパイラ及びシミュレータにより、手法の評価と拡張を行う。

４．研究成果

(1) サンプル地点の特定

本研究で提案する、高精度にシミュレーションを行う地点を特定する手法を以下に示す。

まず、シミュレーションするターゲットアーキテクチャ上で実行するプログラムを特定する。本手法が対象とするプログラムは自動並列化コンパイラあるいは手動により並列化されたプログラムであり、並列化部分はループにより囲まれているものとする。このようなプログラム構造は多くの並列化プログラムで見られる。

次に、実在のコンピュータ上で上記のプログラムを逐次実行し、並列化部分を囲むループの繰り返し（イタレーション）ごとの実行時間をプロファイルする。この時、プロファイルを取得するコンピュータの種類は特に問わない。これは、プロファイル対象となるような実行時間の大きいループにおける実行時間変化の傾向は、コンピュータの種類よりもむしろ、プログラムの挙動、すなわちイタレーション中の計算量に大きく依存するという前提による。

取得したイタレーションのプロファイルは、同程度のコストのグループにクラスタリングされる。これらのクラスタごとに詳細実行を行うイタレーションの数（以下、サンプル数）は、関連研究で多く用いられている統計における区間推定の式を利用して求める。前述におけるイタレーションのプロファイル結果を用いて以下のように求める：

$$n_i = \left(\frac{z \cdot V_x}{\varepsilon}\right)^2$$

ここで、n_iはクラスタiのサンプル数、zは上

側 P%点、V_xは変動係数、εは許容誤差率をそれぞれ表す。

クラスタリングには x-means 法を用いる。x-means 法とは再帰的に k-means (k=2)を事前に決めた条件が満たされるまで再帰的に適用するクラスタリング手法である。本研究では、x-means の停止条件を、総サンプル数が少なくならないこととした。

以上の様にクラスタごとのサンプル数を決定して、さらに各クラスタでシミュレーション時に詳細実行を行うループ中の繰り返し地点を決定する。

シミュレーション後に、対象ループの総実行サイクル数Sを以下のように算出する。

$$S = \sum_i \left(\frac{N_i}{n_i} \sum_j S_{ij}\right)$$

ここではN_iはクラスタi のイタレーション数、n_iはクラスタi のサンプル数、S_{ij}はクラスタiにおけるj番目のサンプルイタレーションの実行サイクル数をそれぞれ表す。

上記は総実行サイクル数の算出方法について述べているが、キャッシュミス数などの他の計測項目についても同様に求めることができる。

(2) 並列化コンパイラとの連携

(1)で述べたシミュレーションのフローを並列化コンパイラと連携させた様子を図 1 に示す。

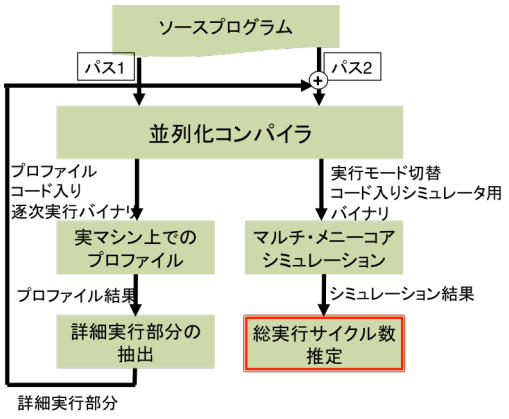

図 1 並列化コンパイラと連携したシミュレーションフロー

パス 1 ではソースプログラム中の並列化部分を含むループのイタレーションの実行サイクル数を計測するプロファイルコードを並列化コンパイラで挿入する。コンパイルされたプログラムは(1)で述べた通り、いずれかの実マシン上で実行され、そのプロファイル結果より詳細実行を行うイタレーションの場所を抽出する。プロファイルを採取するループの特定には、本研究ではコンパイラ指示文を用いた。

パス 2 ではパス 1 で取得した詳細実行を行うイタレーション部分の情報をもとに、プログラム中でシミュレーション精度切り替えコードを挿入したシミュレータ用バイナリプログラムを生成する。

このようにプロファイル用コードの挿入、及びシミュレーション精度切り替えコードの挿入を並列化コンパイラが行うことにより、ユーザの負担軽減が可能となる。

(3) 本手法を適用したときの、実行サイクルス及びキャッシュミス回数の推定値の誤差を評価した。

評価には SPEC CPU 2000 の 183.equake, Mediabench の MPEG2 encoder, SPEC CPU 2000 の 179.art、及び SPEC CPU 2006 の 470.lbm を用いた。

また、シミュレーション対象のアーキテクチャは Oracle SPARC プロセッサコアを持つマルチコアであり、評価では 1, 4, 8, 16 コアのシミュレーションを行い評価した。

4 つのプログラムを用いたときの総実行サイクル数、L1 キャッシュミス回数、及び L2 キャッシュミス回数を図 2 から図 にそれぞれ示す。図中、1-warmup、2-warmup は詳細実行前にイタレーションを 1 回あるいは 2 回余分に詳細実行し、キャッシュ等の状態を整えておく実行形態を表す。また、0-warmup はそのようなウォーミングアップのイタレーション実行が存在しないことを表す。

図 2 より、183.equake ではどのコア数においてもいずれの計測項目も誤差がウォーミングアップ実行無しで 2%未満と、非常に低い誤差でシミュレーション結果が得られたことがわかる。

図 より、MPEG2 encoder では、L2 キャッシュミス以外では実行サイクル数及び L1 キャッシュミス回数のいずれも、どのコア数のシミュレーションでも最大 2%強程度の低い誤差で推定できていることがわかる。L2 キャッシュのミスに関しては、コア数が 8 及び 16 でウォーミングアップ実行無しのときに誤

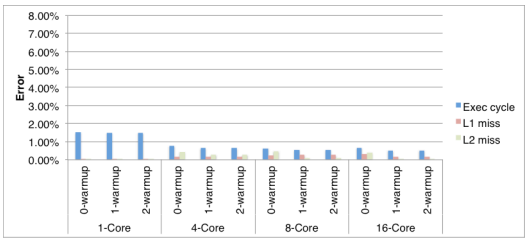

図 2 183.equake のシミュレーション誤差

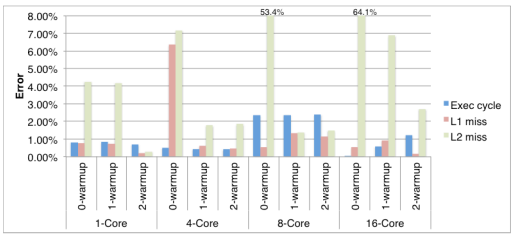

図 3 MPEG2 encoder のシミュレーション誤差

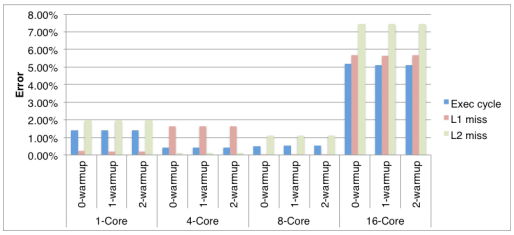

図 4　179.art のシミュレーション誤差

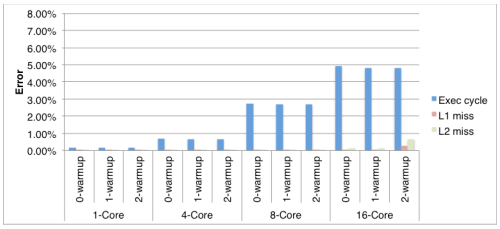

図 5　470.lbm のシミュレーション誤差

差が 50%を超えてしまう。これは、L2 キャッシュミスの回数が相対的に少なく、詳細実行部分での計測のわずかな回数の差が、大きく影響を与えるためである。しかしながら、ウォーミングアップ実行を加えることによりこの誤差は小さくなっており、1 回のウォーミングアップ実行により 6.9%、2 回のウォーミングアップにより 2.7%まで誤差が低下している。

図 より、179.art では 8 コアまではウォーミングアップ実行の有無にかかわらず、どの項目も 2%以下の誤差となっていることがわかる。しかしながら 16 コアの際に実行サイクル数が 5%強を示すなど、比較的高い誤差となってしまっている。これは、シミュレーション用のバイナリプログラムに埋め込まれた精度切り替え用のコードが、キャッシュのヒット・ミスの挙動を変化させてしまっているためである。精度切り替えコードの挿入方法の改良を行う必要が有る。

図 より、470.lbm では全コア数で実行サイクル数が 5%未満の低い誤差を示しているが、コア数が多くなるほど誤差が大きくなっていることがわかる。これは同一クラスタ内部であっても詳細実行にふさわしくない部分が生じているためであり、クラスタ内のどのイタレーションを詳細実行すべきか、今後改良が必要なポイントとなる。

以上をまとめると、本手法により 16 コアまでのシミュレーションにおいてウォーミングアップ実行を適切に行うことにより、総実行サイクル数、L1 キャッシュミス回数、及び L2 キャッシュミス回数を概ね 5%以下の誤差で推定することが可能であることがわかった。

(4) 本手法適用により得られる速度向上率を、16 コアかつウォーミングアップ実行 1 イタレーション分の条件で表 1 に示す。本手法により、183.equake で最大 443 倍、イタレーション数の少ない(450 回転)の MPEG2 encoder でも 19 回、平均で 218 倍の速度向上を得られることが確認できた。

表 1　16 コア・ウォーミングアップ実行 1 回分の時の速度向上率

183.equake	MPEG2 Encoder	179.art	470.lbm
443	19	81	327

(5) 以上より、16 コアのマルチコアアーキテクチャを想定した評価により、最大 443 倍の速度向上を誤差 0.52%で得ることができ、平均では 218 倍の速度向上を 2.76%の誤差で得られることが確認できた。本研究により、マルチコア・メニーコアの高速かつ高精度なシミュレーションが簡易に行うことが可能となり、今後のコンピュータアーキテクチャの研究発展への寄与が期待できる。

５．主な発表論文等

〔学会発表〕(計　5　件)

(1) 石塚亮, 阿部洋一, 大胡亮太, 木村啓二, 笠原博徳, “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”, 情報処理学会研究報告 Vol.2011-ARC-196-14, July. 2011.

(2) 阿部洋一, 石塚亮, 大胡亮太, 田口学豊, 木村啓二, 笠原博徳, “並列化メディアアプリケーションを対象としたメニーコアアーキテクチャシミュレーションの 高速化の検討”, 情報処理学会第 191 回計算機アーキテクチャ研究会, Vol. 2012-ARC-199, No. 3, Mar. 2012.

(3) 阿部洋一, 田口学豊, 木村啓二, 笠原博徳, “並列化アプリケーションを対象とした統計的手法によるメニーコア アーキテクチャシミュレーションの高速化”, 情報処理学会 第 195 回計算機アーキテクチャ研究発表会, Vol.2012-ARC-203 NO.13, Jan. 2013.

(4) 田口学豊, 阿部洋一, 木村啓二, 笠原博徳, “コンパイラと協調したシミュレー ション精度切り換え可能な マルチコアアーキテクチャシミュレータ”, 情報処理学会 第 195 回計算機アーキテクチャ研究発表会, Vol.2012-ARC-203 NO.14, Jan. 2013.

(5) 田口学豊, 木村啓二, 笠原博徳, “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”, 情報処理学会 第 165 回 SLDM・第 32 回 EMB 合同研究発表会 (ETNET2014) 組込み技術とネットワークに関するワークショップ, Mar. 2014.

〔その他〕

ホームページ等

http://www.apal.cs.waseda.ac.jp/

６．研究組織

(1)研究代表者

木村 啓二 (KIMURA, Keiji)

早稲田大学・理工学術院・教授

研究者番号：50318771