4ZJ-1

チェックポインティングによる 評価条件が可変な高速シミュレーション手法の提案

椎名 敦之 † 大津 金光 † 大川 猛 † 横田 隆史 † 馬場 敬信 †
† 宇都宮大学工学部情報工学科

1 はじめに

新規アーキテクチャの開発では，評価条件を変えながら繰り返しシミュレーションを行う必要がある．しかし，シミュレーションは1回当りにかかる時間が長いことが多く，そのシミュレーションを繰り返し行うと全体の評価にかかる時間が長くなる問題がある．

本稿では，シミュレーションの途中で処理を中断し，評価条件を変えながらシミュレーションを再開することで評価時間に要する時間を短縮する手法について提案する．対象は，プログラム全体ではなくその中での一部分のみでの評価を行い評価対象であるコード部分を変えて実行するタイプの，実際のプログラムバイナリコードを使用したプロセッサのシミュレーションである．シミュレーションの中断・再開のために，実行中のプロセス状態の保存・復元を可能にするチェックポインティング手法を用いる．

2 チェックポインティングによる提案手法

提案手法のイメージ図を図1に示す．図中の上から下への太い矢印はプログラムの処理の流れを表している．図1(a)のようなプログラムに対して，提案手法を用いた評価を行う際の手順を説明する．まず，評価対象コードの部分が実行される直前でチェックポイントデータを取り，シミュレーションプロセスの全状態を保存する．この時，プロセス状態を記録したチェックポイントファイルが生成される．チェックポイントファイルの情報を用いてプロセスを再開すると，図1(b)のように，元のプロセスと同じ処理を復元できる．

チェックポイントファイルに記録されている情報を変更すると，図1(c)のように再開後の処理を切り替えることができる．チェックポイントファイルが存在すれば何度でもプロセスを復元し処理を再開することができるため，更に別のコードに切り替えて実行することもできる．これにより評価対象まで実行し直す手間を省くことができ，評価対象コードの部分が異なるプログラムをそれぞれ実行した場合と比べて，評価にかかる時間を短くできる．

例として，ある1つのループに対して逐次処理コードと並列処理コードとで，どの程度実行サイクル数が変化するのかを比較する場合を考える．通常の方法では，対象ループを逐次処理で行うもの (逐次実行版) と並列処理で行うもの (並列実行版) の2種類のコードを実行しなければならない．この場合は，単純に2回分のシミュレーション時間がかかる．

本手法では，逐次実行版の実行ファイルに並列実行版のコードを埋め込む．シミュレーションの途中，評価対象のループが実行される直前でチェックポイントデータを取る．チェックポイントからそのまま再開すると逐次処理が実行されるが，チェックポイントファイルを書き替えることで再開後の処理を並列実行版のものに切り替えることができる．この場合であれば，逐次実行版のシミュレーションと再開後の処理を並列実行版に切り替えたもののシミュレーションにかかる時間だけで多く評価を行えるようになる．

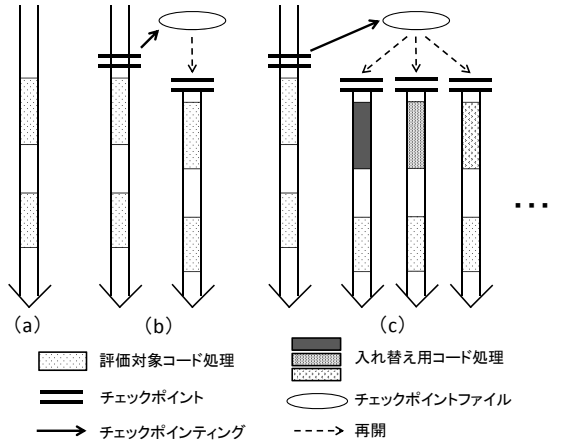

図 1: 提案手法

本手法は2つの段階に分けて実現される．チェックポイントデータの取得とコード切り替えである．

3 チェックポイントデータの取得

今回は本研究室で研究に使用しているSIMCA[1]によるシミュレーションに提案手法を適用する．SIMCAは，マイクロプロセッサの命令セットアーキテクチャのシミュレータであり，実行の結果や実行サイクル数などを調べることができる．

SIMCAではクロスコンパイラを用いて生成した実行ファイルを，シミュレータで実行する．図2(a)にSIMCAに対応しているECOFF形式[2]の実行ファイルの構造を簡単に図示する．実行ファイルにはいくつかのセクションが存在し，例えば.textセクションには各命令が，.rdataセクションには定数がバイナリ形式で記録されている．

Proposal of Fast Evaluation Method using Checkpointing for Repeated Simulation with Variable Conditions
†Atsushi Shina, Kanemitsu Ootsu, Takeshi Ohkawa, Takashi Yokota and Takanobu Baba
Department of Information Science, Faculty of Engineering, Utsunomiya University (†)

本手法では，入れ替え用のコードを.textセクションとその次の.rdataセクションの間に挿入する．入れ替え用コードを埋め込むための充分な領域を確保するために，図2(b)のように.textセクションの後方に入れ替えコード挿入用の領域をあらかじめ確保しておく．具体的には，.text以降のデータを一定の大きさ分だけ後方にずらす．それに合わせて各セクションのファイルオフセット情報，.textセクションのサイズ情報などのヘッダ部分を更新する．その後，コードを入れ替えて評価を行う際に図2(c)のように，空き領域に入れ替え用コードを書き込む．

コード埋め込み用の領域を確保した実行ファイルの実行時，指定したアドレスの命令が実行される際にシミュレータのチェックポイントデータを取る．チェックポインティング処理にはマルチスレッドに対応したオープンソースソフトウェアであるDMTCP[3]を使用する．

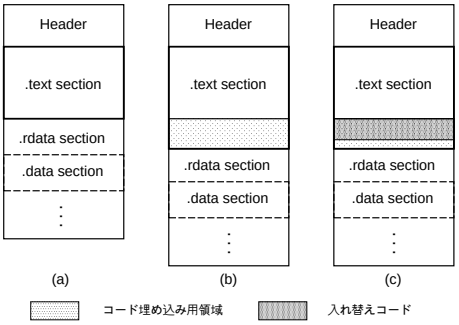

図 2: 実行可能バイナリファイルの変更

4 チェックポイントファイルの書き替え

生成されたチェックポイントファイル内には実行時のメモリの内容などが記録されている．この中にはプログラムも含まれている．これらの情報を書き替え，再開後の処理を変更する．再開後に実行される命令を入れ替え用コードのエントリポイントのアドレスにジャンプする処理に書き替えると，コードの切り替えが実現できる．

5 システムの全体像

提案手法による評価支援システムの全体像を図3に示す．ユーザは通常と同様にシミュレータで実行を行う．

まずは実行可能バイナリファイルにコード埋め込み用の領域を確保する修正を施す．そして，シミュレータで実行し，評価対象コード部分を実行する直前でDMTCPの機能を使ってチェックポイントデータを取得する．実際の評価作業を行う際には，生成されたチェックポイントファイル内のコード領域に入れ替え用のコードを書き込み，元のコードから実行を移すためのジャンプ命令などを書き込むことで，評価コードを切り替えたチェックポイントファイルを生成する．その後，DMTCPにより書き替え後のチェックポイントファイルを使って処理を再開することで元とは異なるコードを実行させる．

本システム実現のために，実行ファイルとチェックポイントファイルのバイナリ書き替え処理を自動で行うツールをそれぞれ作成した．

１つ目のものは，対象となる実行ファイルを入力として受け取る．.textセクションに入れ替え用コードの挿入用の領域を確保した後，新たな実行ファイルとして出力する．

２つ目のものは，チェックポイントファイル，入れ替え用コードのバイナリデータ，ジャンプ設定ファイル（埋め込んだ入れ替え用のコードへのジャンプ命令を上書きするアドレスと，入れ替え用コードが終了後に戻るアドレスが記述されている）を入力とする．チェックポイントファイルの内容を書き替え，新しいチェックポイントファイルとして出力する．

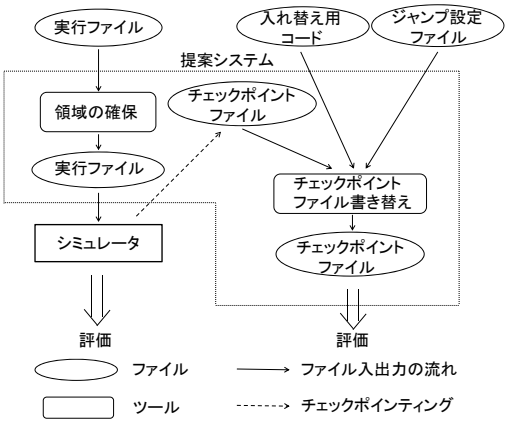

図 3: システムの全体像

6 おわりに

本稿では，チェックポインティングを用いることでシミュレーションに要する時間を短縮する評価手法について述べた．今後は提案システムの完成とその有効性の評価を目指す．

謝辞

本研究は，一部日本学術振興会科学研究費補助金 (基盤研究 (C)21500050，同 (C)21500049) の援助による．

参考文献

[1] Jian Huang: “The SImulator for Multi-thread Computer Architecture Release 1.2”, Technical Report No: ARCTiC-00-05, 2000.

[2] COMPAQ: “Object File / Symbol Table Format Specification”, Tru64 UNIX Version 5.0, 2009.

[3] Jason Ansel, Kapil Arya and Gene Cooperman: “DMTCP:Transparent Checkpointing for cluster computations and the desktop”, IEEE International Parallell and Distributed Processing Symposium, IPDPS 2009, pp.1-12, 2009.