6K-4

コード差分実行による時短シミュレーション法の効果に対する考察

椎名 敦之 † 大津 金光 † 大川 猛 † 横田 隆史 † 馬場 敬信 †
† 宇都宮大学大学院工学研究科情報システム科学専攻

1 はじめに

近年の計算機アーキテクチャの高性能化に伴い，それらの性能を最大限活用するための最適化についても研究が進められている．新規アーキテクチャに対して最適化手法を評価する場合に対象プログラムのバイナリコードを用いてサイクルベースの詳細なシミュレーションを行う方法がある．複数種の最適化手法を適用したコードを用いてシミュレーションを行い，最適化技術を評価していくことになる．

しかし，サイクルベースの詳細なシミュレータの実行速度は実機と比較すると数桁以上遅く，実行に多大な時間を要する．シミュレーション対象のプログラムによっては1回の実行に数日から数週間かかるものも少なくない．このようなシミュレーションを評価する最適化手法の数だけ実行すると莫大な時間がかかるという問題がある．その改善策として，我々はチェックポインティングとコード差分実行を用いた時短シミュレーション法 [1] を提案する．

本稿では，提案手法を複数のベンチマークに適用した場合の時間短縮の効果について述べる．総シミュレーション時間の見積りを元に，提案手法の短縮効果に影響する要素について考察する．

2 インクリメンタルソフトウェアシミュレーション

図1は，ベンチマーク SPEC CPU2000 をマルチスレッドモデルのシミュレータ SIMCA [2] で実行した際の実行時間をまとめたものである．これらのプログラムに対して様々な最適化手法を試行して評価を行う場合，従来は繰り返し回数に比例した時間がかかる．

我々が通常行う評価作業は，プログラム全体ではなくその一部に対して最適化を施した場合の評価となることが多い．そのため最適化手法の評価では，一部にそれぞれ異なる最適化が適用されたプログラムを実行していくことになる．我々は，このシミュレーションを繰り返し行う評価体系において，最適化されていない各シミュレーションで共通な部分の実行情報は必要ないという点に着目し，インクリメンタルソフトウェアシミュレーション (ISS) を提案した．

ISS は以前の実行情報を利用し評価に必要な部分のみを実行し，評価に必要ない部分の実行を省略することで評価にかかる時間の短縮を図るシミュレーション手法である．ISS はシミュレーションプロセスの保存と再利用を可能にするチェックポインティングと，それにより生成されるチェックポイントファイルを使用し最適化部分のみを実行するコード差分実行によって実現される．初回の実行の際，評価範囲が実行される前の状態のシミュレーションプロセスをチェックポイントファイルに保存しておく．生成されたチェックポイントファイルには評価範囲が実行される直前までの実行情報が記録されており，これを用いることで保存された時点から処理が再開される．2回目以降の実行では初回で生成されたチェックポイントファイルを元に，本来実行されるコードを別のコードに切り替える．

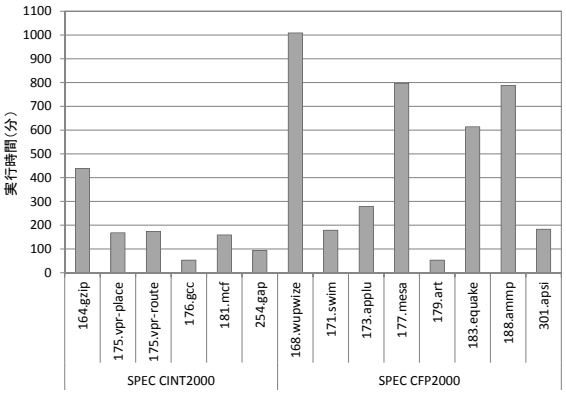

図 1: SPEC CPU2000 のシミュレーション時間

3 ISS の有効性

3.1 短縮効果の見積り

我々は予備評価として，いくつかの評価作業について提案手法が従来よりも短時間で評価が終わることを示した [1]．本節では，一般的なプログラムに対する ISS の有効性について述べる．

1回のシミュレーション時間を t，評価対象の部分の実行割合を r，評価を行う最適化手法数を n とすると，ISS での総シミュレーション時間 T_{iss} は，初回のみシミュレーションを最後まで実行し2回目から n 回目は評価対象の部分のみを実行するため，式 (1) で求められる．従来手法の総シミュレーション時間 T_{usual} は実行時間が t のシミュレーションを単に n 回実行した時間である．

$$T_{iss} = t + tr(n-1) = t(1 + r(n-1)) \quad (1)$$

$$T_{usual} = tn \quad (2)$$

Consideration about Effectiveness of Evaluation Time Reduction Method by using Code Substitution
†Atsushi Shina, Kanemitsu Ootsu, Takeshi Ohkawa, Takashi Yokota and Takanobu Baba
Department of Information Systems Science, Graduate School of Engineering, Utsunomiya University (†)

また，T_{usual} と T_{iss} から，ISS の適用により短縮できるシミュレーション時間の従来手法の場合に対する割合 R が求められる．

$$R = 1 - \frac{T_{iss}}{T_{usual}} = 1 - \frac{1 + r(n-1)}{n} \quad (3)$$

図 2 に，SPEC CPU2000 における各プログラム中の実行割合の高い上位 5 位までのループの実行割合を示す．図 3 は n=5，10 の場合の R の値をグラフにしたものである．

3.2 性能に影響する要因

ISS の短縮効果に影響する最大の要因として，評価およびコード差分実行の対象となるコード部分の実行割合が挙げられる．ISS は評価に必要の無い部分の実行を省略することでシミュレーション時間を短縮するため，評価対象の部分の割合が大きい場合は実行を省略できる部分が少なくなる．

例えば，上位 5 つのループのそれぞれを評価対象として最適化手法の評価を行う場合，176.gcc や 301.apsi 等の評価範囲の実行割合が低いものでは総シミュレーション時間を 50 %以上短縮可能であり，従来の半分以下のシミュレーション時間で同様の評価が可能である．一方で，対象範囲の実行割合が大部分を占めている 175.vpr や 171.swim 等では，全体を実行した場合との差が小さくなる．評価のために実行しなければならない部分が多いと，実行を省略できる部分も少なくなり効果を期待できなくなる．

図 2 で示したような，上位 5 つのループを対象とし最適化手法の数が 10 である場合は，SPEC CPU2000 の大部分のプログラムにおいては実行を省略することが可能である．

２つ目の要因として，評価を行う最適化手法数が挙げられる ISS は，繰り返し実行すればするほど短縮効果が大きくなるという特徴がある．プログラム全体を実行するのは初回のシミュレーションだけであり，２回目以降のシミュレーションではコード差分実行により必要な部分のみの実行で評価が行えるためである．したがって，1 回あたりの実行で省略できる部分が少ない場合でも最適化手法が多いほど，繰り返しシミュレーション実行を行うことになり，短縮効果が期待できる可能性がある．

その他の要因として，プログラムの実行時間が挙げられる．評価対象の実行割合が高い場合でも，図 1 の実行時間と図 3 の短縮可能な割合からおおよその短縮できる時間が求められる．168.wupwise は評価対象が全体の 80%と大部分を占めているため，ISS を適用しても全シミュレーション時間の約 15%しか短縮できない．しかし，1 回あたりの全体のシミュレーション時間が 1000 分と長いため，約 150 分と多くの時間を短縮することができる．

また，評価範囲の実行割合だけでなく実行回数も短縮性能に影響する．1 回のチェックポインティング

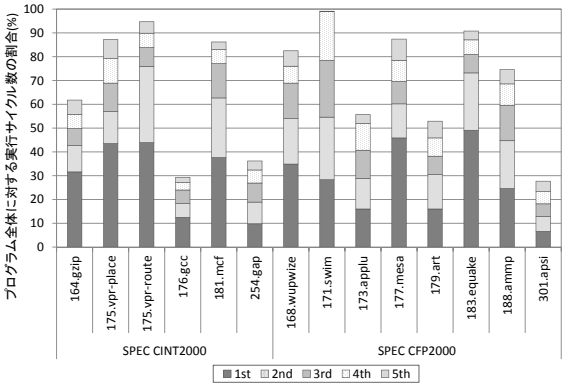

図 2: 実行頻度上位 5 位のループの実行割合

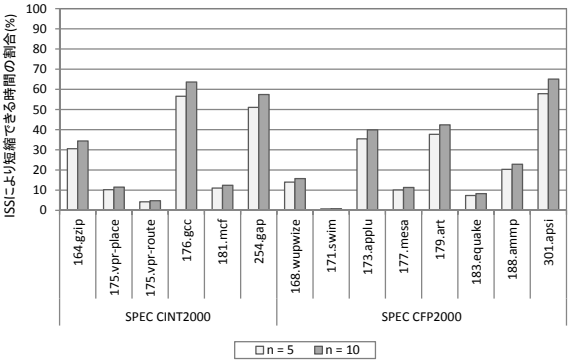

図 3: ISS によるシミュレーション時間の割合

処理のオーバーヘッドはシミュレーション時間と比べると小さく，考慮する必要がほとんどなかった．しかし，188.ammp 等は評価範囲の実行回数が数百万回と多く，実行される直前でチェックポインティングを行うとオーバーヘッドが非常に大きくなってしまう．そのため，チェックポインティングの回数を削減することでさらなる性能向上が達成可能であると考えられる．

4 おわりに

本稿では，提案手法である ISS の有効性について述べた．今後は実際の評価において有効性を確認し，さらに効果を向上させる方法について検討していく予定である．

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)24500055，同 (C)24500054) の援助による．

参考文献

[1] Atsushi Shina et al.: “Proposal of Incremental Software Simulation for Reduction of Evaluation Time,” Proc. 3rd International Conference on Networking and Computing (ICNC), pp.311-315, 2012.

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