6J-03

Valgrind ベース自動並列処理系におけるループの実行時オーバヘッド削減

小渕 裕之 † 三浦 崇 †† 大津 金光 † 大川 猛 † 横田 隆史 †
† 宇都宮大学大学院工学研究科情報システム科学専攻 †† 宇都宮大学工学部情報工学科

1 はじめに

マルチコアプロセッサの性能を有効に活用するために, スレッドレベルの並列処理が必要となる. そのため, マルチスレッドコードを自動で生成する自動並列化コンパイラの研究開発が行われているが, これらのコンパイラは通常, ソースコードを必要とする. そこで, 我々は動的バイナリ変換フレームワークである Valgrind[1] をベースにして, バイナリコードを対象とした自動並列処理システムを開発している [2].

現在の本システムでは, バイナリ変換されたコードの実行に際して生じるオーバヘッドが高速化の障害となっており, 特にループの実行における影響が大きい. 本稿では, ループの実行時に生じるこれらのオーバヘッドを削減する手法, およびその効果について述べる.

2 Valgrind ベース自動並列処理系

バイナリコードは CPU の命令セットに強く依存するため, 幅広い命令セットのバイナリコードを対象とした並列化の実現には, 命令セットに依存しないバイナリ変換機能が必要となる. そこで, 我々はバイナリレベルでの自動並列処理システムの開発にあたり, この機能を備えた Valgrind をシステムのベースとした.

Valgrind は, デバッガやプロファイラのような動的解析ツールを構築するためのフレームワークを提供するプロセス仮想マシンである. Valgrind の構成を図 1 に示す. Valgrind は core と tool plug-in から構成される. tool plug-in は core によって中間表現 (IR: Intermediate Representation) に変換された guest application の基本ブロックに対して, 解析用のコードを挿入したり改変を加えたりする. ユーザは tool plug-in を C 言語で記述することにより, 独自の解析ツールやコードの改変ツールを実現することができる. core は VEX と coregrind と呼ばれる二つの部分に分けられる. VEX では基本ブロック単位で動的バイナリ変換を行い, その過程で tool plug-in を呼び出す. coregrind ではスレッドの実行制御や VEX で生成された変換コードの管理と検索, およびその呼出しといった処理をする.

一般にプログラムの実行時間の多くはループで消費されることから, 我々のシステムでは guest application に含まれるループを並列化の対象にする. Valgrind はマルチスレッドで実行できないため, 我々は Valgrind

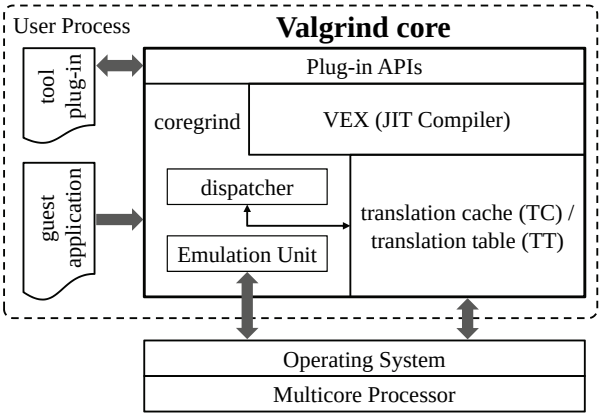

図 1: Valgrind の構成

core に並列処理のためのスレッドを生成し, 制御する機能を追加した. そして, tool plug-in により対象のループを並列化することで, 並列処理を実現している.

3 ループの実行時オーバヘッド削減

ここでは, バイナリ変換されたループの実行に際して生じるオーバヘッド, およびその削減手法について述べる.

3.1 基本ブロックのマージ

VEX で生成された変換コードは必要に応じて再実行されるように, translation cache (TC) に格納され, 各変換コードの先頭アドレスが translation table (TT) に記憶される. 変換コードは dispatcher を経由して実行され, 実行が完了すると dispatcher に制御が戻る. dispatcher では, 次に実行する変換コードを tt_fast と呼ばれるダイレクトマップ方式のキャッシュにより検索する. 検索にヒットした場合は直ちに対象の変換コードが実行され, そうでない場合は VEX により新しい変換コードが生成される.

TC に存在する変換コードが繰り返し実行される場合, それを検索する処理がオーバヘッドになる. そのため, 特にループを実行する際はこのオーバヘッドによる影響が大きくなり, 高速化の障害となる. そこで, ループを構成する基本ブロックを一つの変換コードとしてマージすることにより, ループの実行時に制御が dispatcher に戻る回数を削減する.

ループを構成する基本ブロックの集合は図 2 に示すように tool plug-in で一つにマージされる. はじめに, ループの一回目の繰返しにおいて, tool plug-in により各基本ブロックの中間表現をメモリにコピーする. コピーした中間表現はマージするまでメモリに保持しておく必要があるため, コピーする際はシステムの実行が終了するまで内容が失われないように Valgrind core

Overhead Reduction of Loop Execution in Automated Parallel Processing System by using Valgrind
†Hiroyuki Obuchi, ††Takashi Miura, †Kanemitsu Ootsu, †Takeshi Ohkawa and †Takashi Yokota
Department of Information Systems Science, Graduate School of Engineering, Utsunomiya University (†)
Department of Information Science, Faculty of Engineering, Utsunomiya University (††)

内部で管理されている permanent なメモリ領域を使用する．

一つの変換コードとしてマージされたループは, 開始点のアドレスに対応付けられた形で TC 内に格納される．そこで, ループを構成する基本ブロックが揃った時点で VEX にループ全体を変換させるために, 一旦ループの先頭の基本ブロックに対応する TC 内の変換コードを無効化する．これにより, ループの二回目の繰返しにおいて, 再び先頭の基本ブロックに対するバイナリ変換処理が発生する．

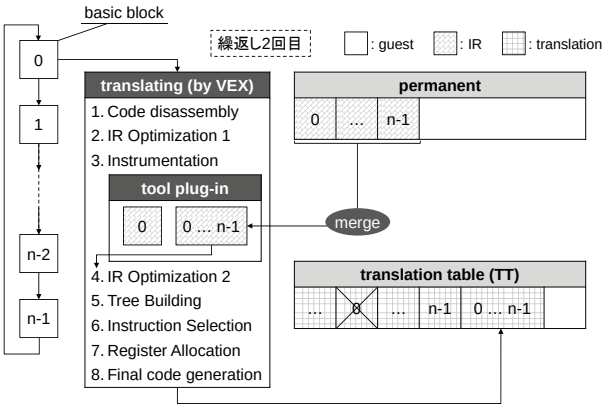

図 2: ループ内の基本ブロックのマージ処理

マージした基本ブロックの間は dispatcher を経由せずに直接ジャンプできるようにする必要があるため, tool plug-in によりマージした基本ブロックの中間表現をバイナリコードに変換する際, ループを抜けるジャンプは除き, 基本ブロック間のジャンプ命令を dispatcher へのジャンプ命令から同じループ内への相対ジャンプ命令に置き換える．そこで, ジャンプ先の相対アドレスを求めるため, バイナリコードを生成する際にジャンプ先/元に関して先頭からのオフセットを用意したテーブルに記憶する．そして, 変換コードが TC に格納された後に先程のテーブルに記憶したオフセットに従い, ジャンプ先/元の TC 上でのアドレスを算出し, それぞれのアドレスの差により相対アドレスを求める．その後, 各相対ジャンプ命令のオフセットオペランドを求めた値に更新する．

3.2 冗長な命令の除外

guest のレジスタ値等の実行コンテクストは guest state と呼ばれるメモリ領域に格納される．変換コードの実行では, guest のレジスタの値を使用したり更新したりするため, guest state へのメモリアクセスが発生する．特に guest IP (Instruction Pointer) の値は guest application の一命令ごとに更新されるため, メモリアクセスによるオーバヘッドが大きくなる．そこで, 中間表現をバイナリコードに変換する際, 最終値を除いて guest IP を更新するための命令を削除することにより, 変換コードの実行におけるメモリアクセスの回数を削減する．

4 評価

提案手法の効果を明らかにするため, 我々は行列積を計算する 3 重ループを対象に, dispatcher と変換コードの実行に要するサイクル数を計測した．評価環境は次の通りで, CPU は Core i7 4770 3.40GHz, メモリの容量は 32GB, OS は Linux (CentOS 6.6), Valgrind のバージョンは 3.6.1 である．図 3 に計測結果を示す．提案手法の適用により, Valgrind で tool plug-in を使用せずに通常実行したときと比較して, 38%のサイクル数を削減した．さらに, 並列処理することで通常実行時に比べ, 84%のサイクル数を削減した．

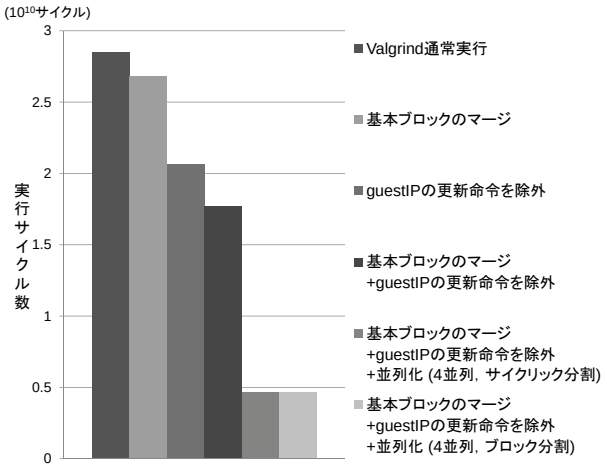

図 3: 各最適化手法と実行性能

5 おわりに

本稿では, 我々が開発している Valgrind ベース自動並列処理系において, バイナリ変換されたループの実行に際して生じるオーバヘッドを削減する手法, およびその効果について述べた．今後の課題は条件分岐を含むループについても一つの変換コードとしてマージできるように, 今回の提案手法を拡張することである．

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，同 (C)25330055，若手研究 (B)25730026) の援助による．

参考文献

[1] Nicholas Nethercote and Julian Seward: “Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation”, Proc. ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation (PLDI 2007), pp.89-100, 2007.

[2] Takayuki Hoshi, Kanemitsu Ootsu, Takeshi Ohkawa, Takashi Yokota, “Runtime Overhead Reduction in Automated Parallel Processing System using Valgrind”, Proc. 1st International Symposium on Computing and Networking – Across Practical Development and Theoretical Research – (CANDAR’13), pp.572-576, 2013.