6J-04

Valgrindにおける並列化された中間表現の簡易評価環境の提案

三浦 崇 † 小渕 裕之 †† 大津 金光 †† 大川 猛 †† 横田 隆史 ††
† 宇都宮大学工学部情報工学科 †† 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

近年普及しているマルチコアプロセッサの性能を有効に活用するには，スレッドレベルでの並列処理が必要である．そこで我々は，ソースコードの参照を必要としないバイナリコードレベルで並列処理コードを自動生成するシステムの研究を行っている [1].

本システムにおいて自動生成する並列処理コードとして，どのようなコードが有効であるかを検討してゆく必要がある．現在は，システムに並列化された中間表現コードを直接埋め込むことで，生成コードの性能について比較検討を行っている．そのため，並列処理コードの変更を行う場合，評価作業の準備に時間的コストがかかる．そこで，並列処理コードの評価の時間的コストを削減するための簡易評価環境を提案する．

2 Valgrind ベース自動並列処理系

命令セットに依存するバイナリコードを並列化するため，並列処理システムの開発には命令セットに依存しないバイナリ変換機能を用いる必要がある．そこで，バイナリ変換フレームワークであり対応しているプラットフォームが多い Valgrind[2] を利用する．Valgrind は，メモリチェッカ等の動的計測ツールを構築するオープンソースのフレームワークである．Valgrind の構成を図 1 に示す．Valgrind は core と tool plug-in によって構成される．guest application と tool plug-in，Valgrind core は同一プロセス上で実行される．

Valgrind core は，解析対象である guest application を基本ブロック単位でバイナリ変換し，バイナリコードを命令セット非依存の中間表現に変換する．変換された中間表現は，tool plug-in によって計測命令が追加される．次に中間表現は Valgrind core によって再びバイナリコードに変換され，translation cache に格納される．格納されたコードは dispatcher を経由して実行される．

我々が開発している並列処理システムでは，プログラムの実行時間の多くを占めるループを並列化して，処理の高速化を図っている．スレッドの生成および並列実行を行う処理は Valgrind core に，並列処理コードの挿入処理は tool plug-in に実装した．本システムは，並列対象の基本ブロックは並列実行，その他は逐次実行を行う．並列対象の基本ブロックがバイナリコードから中間表現に変換されたとき，tool plug-in で並列処理コードを挿入する．次に再びバイナリ変換でバイナリコードに変換し，Valgrind core で並列実行する．本システムでは，親スレッドを含めた 4 スレッドで並列処理を行う．

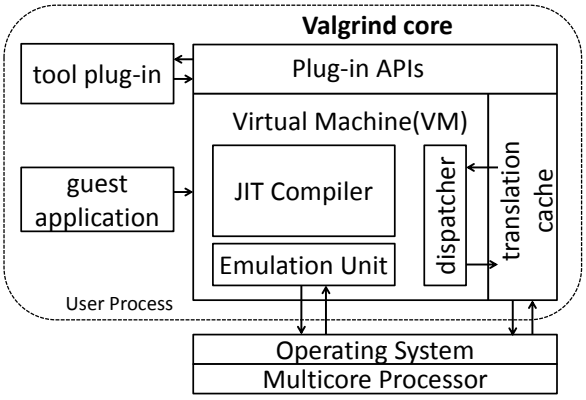

図 1: Valgrind の全体構成

3 簡易評価環境の実現

現在の並列処理コードの検討評価作業では，tool plug-in に直接埋め込む形で並列処理コードを記述している．そのため，並列処理コードの変更に tool plug-in のソースコードの修正および再ビルドが必要である．この時間的コストを削減するために，実行中に外部ファイルから並列化された中間表現コードを入力することで，バイナリコードの変更を可能にする仕組みを導入することが有効であると考えた．そこで，中間表現を外部ファイルにテキスト化して出力し，エディタ等の外部ツールにより中間表現を並列化し，再びファイルとして入力を行うことにより，並列処理コードを簡易に比較検討する手法を提案する．

この手法を実現するために，中間表現を外部ファイルに出力する機能，外部ファイルから一行ずつ入力する機能，入力された文字列の中間表現を数値化された中間表現に変換する構文解析する機能の実装を行った．ファイル入出力機能および構文解析する機能は tool plug-in に実装した．tool plug-in には，中間表現の出力および入力の両方が実装されているため，入出力の制御は実行時に入力モードおよび出力モードを選択することで行う．

4 入出力機能の動作アルゴリズム

並列処理コードの評価は，以下の流れで行う．

1. 中間表現を外部ファイルに出力
2. エディタ等の外部ツールで中間表現を並列化
3. 並列化された中間表現を外部ファイルから入力

Proposal of a Simple Evaluation Environment of Parallelized Intermediate Representation in Valgrind

†Takashi Miura, ††Hiroyuki Obuchi, ††Kanemitsu Ootsu, ††Takeshi Ohkawa and ††Takashi Yokota
Department of Information Science, Faculty of Engineering, Utsunomiya University (†)
Department of Information Systems Science, Graduate School of Engineering, Utsunomiya University (††)

出力モードでは，guest application を Valgrind 上で逐次実行しながら，すべての基本ブロックの中間表現を外部ファイルに出力する．

次に入力モードでの動作の流れを図 2 に示す．入力モードでは，実行時に並列対象の基本ブロックの先頭アドレスを指定する．tool plug-in に渡された基本ブロックが並列対象であった場合，用意した外部ファイルから並列化した中間表現を読み込む．文字列として読み込まれた中間表現を，構文解析の機能により数値化された中間表現に変換し，ここで得られた中間表現の基本ブロックを Valgrind core に渡す．これにより，並列化された中間表現をバイナリコードに変換し並列実行する．tool plug-in に渡された基本ブロックが並列対象でない場合，何もせず基本ブロックを Valgrind core に渡す．

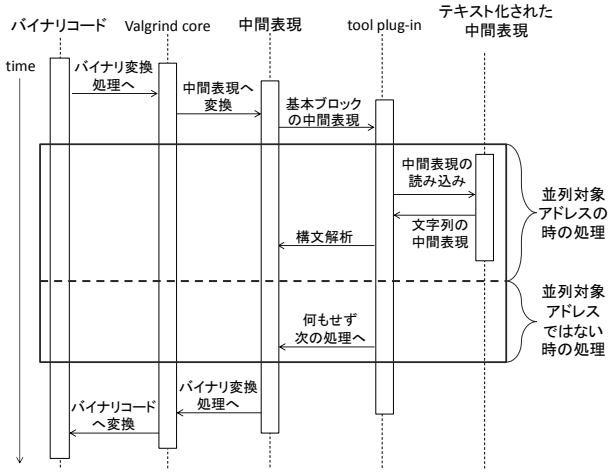

図 2: 入力モードでの動作の流れ

5 簡易評価環境の機能検証

簡易評価環境の機能検証として，外部から並列化した中間表現を入力することでプログラムの並列化を行う．guest application には 1 から 10000 の総和計算プログラムを使用する．並列処理は，ループの各繰り返しを循環的に複数のスレッドに割り当てるサイクリック分割で行う．そのため，あらかじめ Valgrind core に，ループ変数の初期値としてスレッド ID（親スレッドは 1，各子スレッドは 2，3，4）を設定した．

並列対象ループの中間表現をエディタ等の外部ツールで並列化するために，テキスト化した中間表現を取得する必要がある．そこで，出力モードで実行し，guest application の中間表現を取得した．また出力された中間表現を参照し，並列化対象ループの基本ブロックの先頭アドレスが 0x4004dc であることを確認した．取得した中間表現の一部を図 3 に示す．サイクリック分割による並列処理を行うため，ループ変数の増分値を 1 から 4 に書き換えた．

並列化した中間表現を入力モードで読み込んだときの実行結果を図 4 に示す．図 4 の mode と address は独自に追加した引数である．mode には，出力を表す

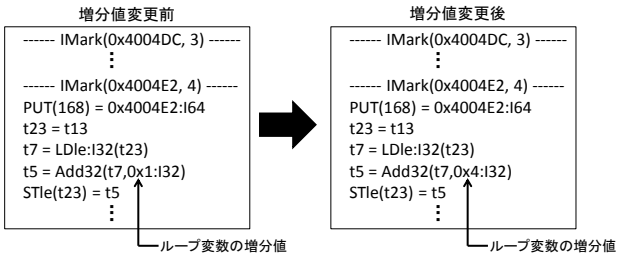

図 3: テキスト化された中間表現の編集

“OUT” もしくは入力を表す “IN” の文字列を引数として与えて入出力の制御を行う．address には，並列対象ループの基本ブロックの先頭アドレスを指定する．--mode = IN と--address = 0x4004dc を引数として追加して実行することで，実行時に並列化した中間表現を読み込み，期待した結果が得られた．

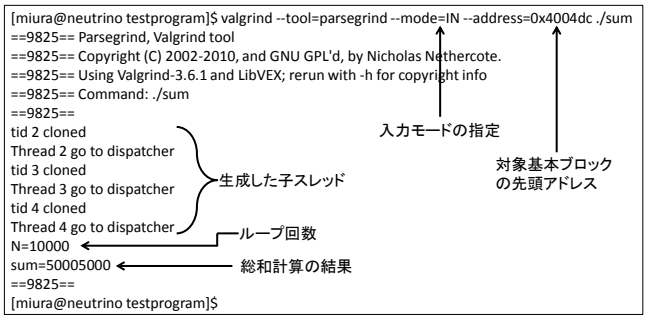

図 4: 並列処理コードの検証結果

6 おわりに

本稿では，外部ファイルに出力された中間表現を編集して再び入力とすることで，バイナリコードを変更する手法について述べた．これにより，tool plug-in のソースコードの修正および再ビルドにかかる時間を削減した．今後，様々なループで並列処理コードを評価するために，並列化対象ループを検出する必要がある．そのため，中間表現に含まれるループをインデントして出力する機能を実装予定である．

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

参考文献

[1] 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.

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