1K-2

制御流が複雑なループに対応したバイナリレベル自動並列化処理系の実装

芝崎 諒　大津 金光　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

近年，マルチコアプロセッサの普及により，マルチスレッド実行による高速化の重要性が高まっている．そこで我々は，バイナリレベルにてシングルスレッドコードをマルチスレッドコードへ変換し，実行速度の向上を図るシステムの研究開発を進めている [1]．

我々は，マルチスレッド化の対象としてループ構造を持つコードに着目しているが，従来のシステムでは単純な構造を持つループを対象としており，複雑な制御流を含むループのマルチスレッド化は困難であった．そのため，そのような構造を持つバイナリコードに対しても，高速なマルチスレッド実行が可能となる変換を行うための研究をこれまで進めてきた [2]．

本稿では，複雑な制御流を含むループに対しても正しい計算結果を得ることのできるマルチスレッド化コードをバイナリレベルで生成する処理系を実装する．

また，そのシステムによって実際のプログラムをマルチスレッド化し，その効果を考察する．

2　マルチスレッド実行モデル

本研究では，マルチスレッド実行モデルについてスレッドパイプライニングモデル [3] を前提としている．図 1 にスレッドパイプライニング実行の流れを示す．各スレッドの実行は，4 種のステージから構成される．

Continuation Stage では，次のスレッドが実行の開始時に必要とするループ変数などの値計算を行い，次のスレッドを起動する．TSAG (Target Store Address Generation) Stage では，スレッド間で依存となる可能性のあるメモリのアドレスをメモリバッファに登録し，メモリアクセスの監視を開始する．Computation Stage では，計算処理本体のコードを実行する．Write-Back Stage では，スレッド実行の結果をメモリに書き戻す．

なお，Computation Stage の実行中に発生するメモリアクセスは全てメモリバッファで監視されている．スレッド間で依存となっている場合にはスレッド間で同期を取り，正しい値の受け渡しを行う．この受け渡しについて実際には，依存変数に対してそのスレッドで最後に値を更新する命令より後に，スレッド間でのデータ送信を行うストア命令 (以降，送信命令) を追加する．これを受けて，スレッド間データ受信を行うロード命令 (以降，受信命令) を次のスレッドに実行させることで，依存変数の正しい値の受け渡しが実現できる．この受信命令は，依存変数に対してそのスレッドで最初に値を参照する命令より前に追加する．

依存変数の値の受け渡しはこのように行われるが，実行するループコードが分岐によって変化する場合，依存変数の値の更新が行われない場合がある．この際には，次のスレッドに依存変数の値の受信の許可を与える命令 (以下，リリース命令) を代わりに実行する．

An Implementation of Automatic Binary-Level Multithreading System that Handles Complicated Loops
† Ryo Shibasaki, Kanemitsu Ootsu, Takashi Yokota, and Takanobu Baba
‡ Department of Information Science, Faculty of Engineering, Utsunomiya University

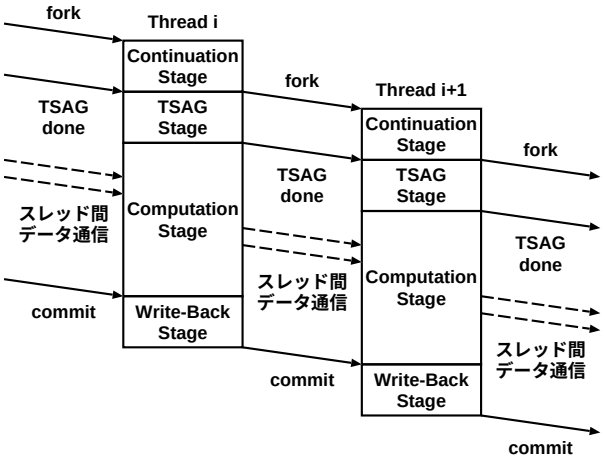

図 1: スレッドパイプライニングモデル

3　並列化コードの実行時における課題

3.1　受信命令の実行時の問題

図 2 に，マルチスレッド化したループコードが分岐によって複数の基本ブロックから構成されている様子を示す．箱形の各命令は全て，ひとつの依存変数に対して実行されるものとする．また，基本ブロックに命令の箱が接している場合，そのブロック中で各命令が上から順に実行されることを示す．

図 2 において，左側のパスを経由する場合は問題なく値の受け渡しが行われるが，右のパスでは，本来のシングルスレッド実行では「値の更新→値の参照」となるところが，ここでは値を参照する前に受信命令が実行されているため，直前に行われた更新の操作が無かったことになり，本来とは異なる不正な値を参照することになる．

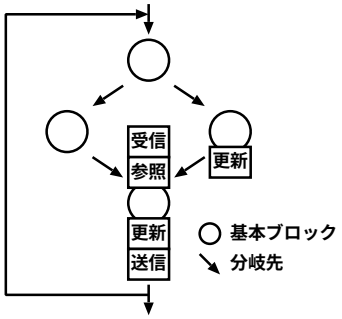

図 2: 受信命令実行時の問題

3.2　リリース命令の実行時の問題

図 3 に，図 2 と同様のループコードがスレッドパイプライニング実行されている様子を示す．

図 3 において，左側のパスでは問題なく値の受け渡しが行われるが，右側のパスでは，前のスレッド i-1 から値を受信していないにも関わらずリリース命令を実行しているため，スレッド i-1 から依存変数の値が送信されるよりも前にリリースが行われる場合，その値が次のスレッド i+1 以降で参照されなくなる．

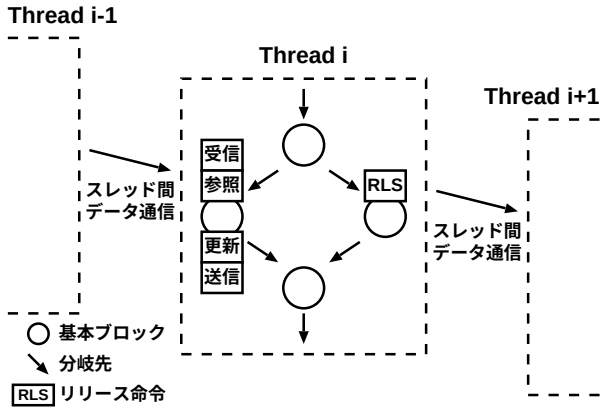

図 3: リリース命令実行時の問題

4 実行時の各問題への対処方法

前述した問題への対処方法 (後述) は，次に挙げる 3 つの条件に沿うものとする．

- 受信命令は，全てのパスで値の更新より先に実行させる
- 全てのリリース命令について，必要があれば先に実行される場所に受信命令を追加する
- 上の方針を満足する範囲で，受信命令を可能な限り遅く実行する

まず，1 つめの条件を満足するために，ループ中の各ブロックに更新命令が存在する場合，そのブロックから到達する可能性のあるブロック全てを要素とする，受信命令の実行を禁止する領域を形成する．この領域は実行順の方向にのみ展開していくので，形成した領域のいずれにも含まれないブロック内であれば，問題無く受信命令を実行できる．3 つめの条件にも従って，この領域に最も近い，つまり隣接しているブロックに受信命令を追加する．

次に，2 つめの条件を満足するため，全てのリリース命令から実行順の逆方向に遡り，受信命令を含む値の更新が無いパスに対しては，リリース命令に可能な限り近い位置へ受信命令を追加する．

この 2 段階の手順を経て追加した受信命令の中には，更新命令の前や他の受信命令の後で実行されるものがある．これらについては，依存変数の値を無意味に重複して更新するので取り除く．

以上のようにして，実行を禁止したブロックに存在する受信命令を取り除き，安全に実行できるブロックに受信命令を追加することで，先述の問題を解消し，正しいマルチスレッド実行を可能にする．

5 評価

制御流が複雑なループに対応したマルチスレッド化システムの評価を行う．

5.1 評価方法

評価には，スレッドパイプライニングモデルシミュレータの SIMCA[4] を使用する．変換対象となるプログラムのバイナリコードは，SIMCA 用 *GCC* クロスコンパイラ (version 2.7.2.3) に最適化オプション "`-O3`" を適用して，シングルスレッドバイナリコードを生成する．そのバイナリコードを，本稿にて説明したアルゴリズムを実装したマルチスレッド化システムに適用して，マルチスレッドバイナリコードを生成する．

評価対象には，SPECint2000 の *175.vpr* のサブルーチン *get_non_updateable_bb()* が含むループのひとつを使用した．このループは，前述したリリース命令実行時の問題が発生するため，従来のシステムによって並列化したプログラムをマルチスレッド実行した場合に，正しい計算結果を得ることができないループである．

評価は，計算結果の正否について，シングルスレッドコードとマルチスレッドコードをそれぞれ実行し，両者の実行結果を比較することで行う．入力データセットに *test* を用いて，マルチスレッド実行のシミュレーションを行なった．

5.2 評価結果

対象ループを含むバイナリコードをシングルスレッド実行およびマルチスレッド実行した結果，それらの実行結果が一致することを確認した．この結果から，本稿で実装したアルゴリズムによって，正しいマルチスレッド実行が可能であることを確認できた．

なお，実行形式の変化による速度向上率については，マルチスレッド実行によって実行サイクル数が約 2 倍に増加していた．原因としては，依存変数が多数存在すること，1 度のループで実行されるイテレーション数が少ないことが考えられる．より多くのプログラムのマルチスレッドコードを生成して実行し，性能を改善できる要素を究明することが今後の課題となる．

6 おわりに

本稿では，制御流が複雑なループに対しても正しい計算結果を得ることのできるマルチスレッドコードを生成する処理系の実装を行なった．また，実装したシステムを実際にプログラムへ適用した結果，従来はマルチスレッド化の対象にできなかったループのマルチスレッド化が可能であることを確認した．

今後の課題として，これまでに研究開発を進めた処理系の高性能化が考えられる．具体的には，実行速度の向上や，正しくマルチスレッド実行できるループの種類を拡大することが挙げられる．

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 (B)18300014，同 (C)16500023，若手研究 (B)17700047) および宇都宮大学重点推進研究プロジェクトの援助による．

参考文献

[1] 大津 金光, 小野 喬史, 横田 隆史, 馬場 敬信, "バイナリレベルマルチスレッド化コード生成手法とその評価," 情報処理学会論文誌ハイパフォーマンスコンピューティングシステム, Vol.44, No.SIG-1 (HPS 6), pp.70–80, 2003.

[2] 芝崎 諒, 大津 金光, 横田 隆史, 馬場 敬信, "複雑な制御流を含むループに対応したバイナリレベル自動並列化処理系の開発," 情報処理学会第 68 回全国大会講演論文集, pp.1-119–1-120, 2006.

[3] J.-Y. Tsai, J. Huang, and et al, "The Superthreaded Processor Achitecture," IEEE Transactions on Computers, Vol.48, No.9, pp.881–902, 1999.

[4] J. Huang, "The SImulator for Multi-threaded ComputerArchitecture (Release 1.2)," 2000.