パスベーススレッド分割手法に基づいた自動並列化システムの実装

伊里 拓也 † 小川 大仁 † 大津 金光 † 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

近年マルチコアプロセッサが広く普及しているが，その性能を活かすためにはプログラムのマルチスレッド化が必要である．

プログラムのマルチスレッド化においては，ループレベル並列性を活用したループのマルチスレッド化が一般に効果的とされている．しかし，浮動小数点演算系プログラムでは高い並列性を得ることができるが，整数演算系プログラムにおいては並列性を得ることが難しいという問題がある．

そこで，我々はループ並列化と異なるマルチスレッド化手法として，プログラムの実行割合の最も高いパス (#1 パス) を基にスレッド分割を行う，パスベーススレッド分割手法を提案した [1][2]．

本研究では，パスベーススレッド分割手法によるプログラムのマルチスレッド化処理を行う自動並列化システムの実装を行うものである．

2 自動並列化システム概要

我々が開発している自動並列化システムの構成図を図 1 に示す．本システムでは，マルチスレッド化対象となるバイナリコードを入力とし，静的自動並列化システム (STO) および動的自動並列化システム (DTO) の 2 つの並列化システムによりマルチスレッドコードを生成する．本研究では，パスプロファイル情報を基に静的な解析によりマルチスレッド化を行うものとして，STO に対してパスベーススレッド分割手法の実装を行う．

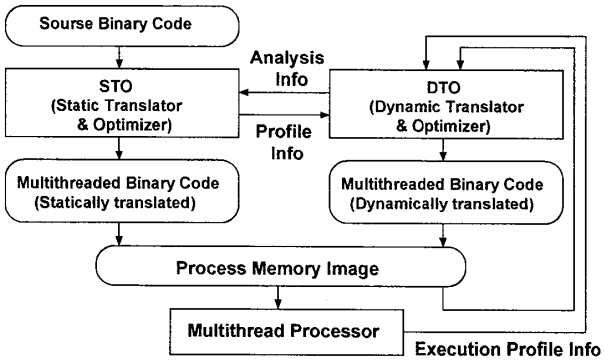

図 1: 自動並列化システム構成図

3 パスベーススレッド分割処理

3.1 処理手順

パスベーススレッド分割処理手順は，大きく以下の 5 つのステップに分けられる．

1. 対象コードの#1 パス情報の取得
2. #1 パス上の基本ブロック間データ依存解析
3. スレッド分割候補点の列挙およびスレッド分割点の決定
4. 各スレッドが持つ基本ブロックを確定
5. スレッド制御命令の挿入

パスベーススレッド分割手法では，まずマルチスレッド化対象コードのパスの実行割合情報をプロファイリングにより取得し，対象コードにおける#1 パスを特定する．この時，実行パス上にループが含まれている場合，ループを 1 つのマクロブロックとして扱う．これはループイテレーション中でスレッド分割を行った場合，反復実行中に#1 パス以外を通り投機ミスを起こす可能性が高いので，イテレーション中でのスレッド分割は行わないためである．

次に，#1 パス上の基本ブロック間でのデータ依存を解析し，スレッド間依存が存在しない基本ブロック境界をスレッド分割候補点とし列挙する．このスレッド分割候補点から，マルチスレッド実行が最速となるようなスレッド分割点を決定する．

スレッド分割点が決定した段階では，各スレッドは#1 パス上の基本ブロックのみを含んでいるが，ここで#1 パス上以外の基本ブロックを各スレッドに含める処理を行う．#1 パス以外のパスもスレッドに含めることにより，投機ミスの発生を抑え高速化効果を高めることができる．

最後に，スレッド制御命令を適切な位置に挿入しマルチスレッドコードの完成となる．

3.2 問題点

パスベーススレッド分割手法では，最も実行割合の高いパスのみを対象とすることで，制御構造の複雑さを軽減し，スレッド間依存の無いマルチスレッド実行が可能という特徴がある．

しかしながら，実際にはスレッド間依存が存在しない点が少なく，またこれらのスレッド分割候補点だけでは，スレッドサイズに偏りができ，高速化効果を得られないなどの問題がある．

そこで，この問題点に対処するためスレッド間依存の解消を用いたスレッド分割を行う必要がある．

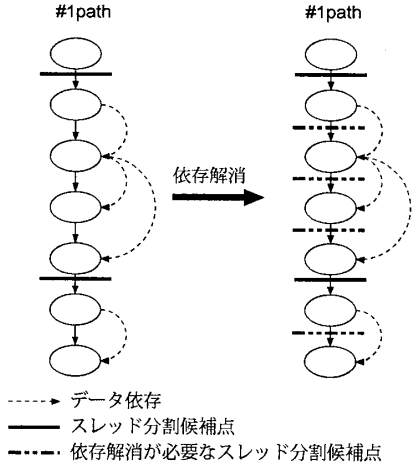

図 2: スレッド分割候補点の列挙

4 依存解消を用いたスレッド分割

4.1 依存解消法

スレッド間データ依存解消法として以下に 4 つの処理を挙げる．

Implementation of Automatic Parallelization System based on Path Based Thread Partitioning Method
†Takuya Iri, Hirohito Ogawa, Kanemitsu Ootsu, Takashi Yokota and Takanobu Baba
Department of Information Systems Science, Graduate School of Engineering, Utsunomiya University (†)

- 変数定義命令の fork 命令前移動による依存解消
- 変数定義命令の後方移動による依存解消
- 変数定義命令のコピーによる依存解消
- 変数使用命令の前方移動による依存解消

これらの処理では，データ依存における変数定義命令と変数使用命令を基本ブロック境界を越えて移動させることで，スレッド間依存の解消を行う．命令移動やコピーによりプログラムの実行結果を損なうような処理は適用できないが，#1 パス上の各基本ブロック境界において，いずれかの処理により依存解消が可能ならばスレッド分割候補点とすることができる．

図 2 は制御フローグラフから#1 パスを抜き出したものである．図 2 の左図のように依存解消を用いなければスレッド分割候補点が 2 つしかできないものに対し，依存解消を用いることでスレッド分割候補点を増加させることが可能となり，より高速なマルチスレッドコードを作成することが可能となる．

4.2 スレッド分割点の決定

スレッド分割候補点の列挙を行った後，そこからマルチスレッド実行時間が最短となり得るスレッド分割点を選択することになる．

依存解消を必要とするスレッド分割候補点をスレッド分割点としてスレッド分割を行う場合，依存解消のための命令移動や命令コピーを行うため，スレッド分割点前後のスレッドのサイズが増減し，マルチスレッド実行時間も増減する．(以後これらをコストと呼ぶこととする)．

更には，ある基本ブロック境界をスレッド分割点として決定した場合，各依存解消処理において命令移動やコピーを行う位置が変化するため，各依存解消処理が適用可能であるか否かが変動することになる．

これらを踏まえた上で，マルチスレッド実行時間が最短となるためのスレッド分割点の決定を行うために，以下の手順を踏む．

1. #1 パスの依存解析情報からスレッド分割候補点の列挙および，各候補点における各依存解消処理によるコストを計算する
2. 各スレッド分割候補点において，依存解消に必要なコストによる影響も含んだマルチスレッド実行終了時間を計算により求め，最短となる点および，その点での依存解消法を決定し，2 つのスレッドに分割する．
3. 分割後のスレッドでスレッド制御による遅延も含めて，最も終了時間が遅いスレッドに対して，手順 1 から繰り返しスレッド分割点決定処理を行っていき，分割によりマルチスレッド実行時間が短縮できなくなるまでこの処理を行う

5 マルチスレッドコード生成処理

5.1 スレッドの確定

スレッド分割点決定後，#1 パスに含まれない基本ブロックを各スレッドに含めるか判定を行い，各スレッドが持つ基本ブロックの集合を決定する．

そのため，制御フローにより投機ミスが起こらない基本ブロックの集合を作る．図 3 の (a)(b)(c) は，それぞれ制御フローグラフの一部を抜き出したものである．

まず，スレッド先頭の基本ブロックから制御フローに沿って到達可能な基本ブロックの集合 A を作る (図 3(a))．次に，次スレッドの先頭の基本ブロック (次スレッドが存在しない場合は，スレッドの終端の基本ブロック) から，制御フローの逆に辿って到達可能な基本ブロックの集合から次スレッドの先頭の基本ブロックを引いた集合 B を作る (図 3(b))．この 2 つの集合 A と B を掛け合わせた積集合 C(図 3(c)) を作る．

ここから，集合 C に含まれる基本ブロックが後続スレッドの基本ブロックと基本ブロック間データ依存を持たないか調べ，依存を持つ基本ブロックは集合 C から除外していき，集合 D を作る．この集合 D がスレッド間依存が存在せずに，最も多くのパスを含むことができるスレッドとなる．

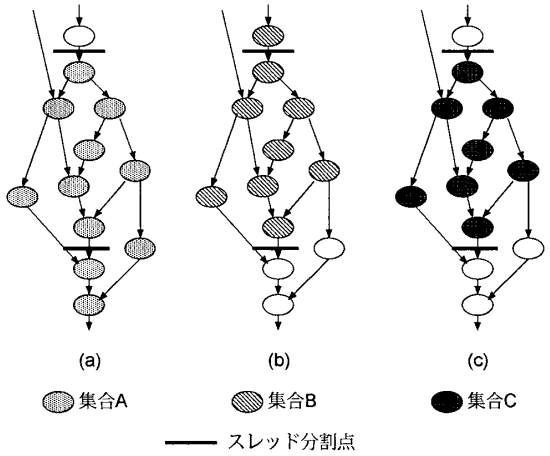

図 3: スレッドが持つ基本ブロック集合の決定

5.2 スレッド制御命令の挿入

各スレッドが持つ基本ブロック集合情報および依存解消処理情報を基に，スレッド制御命令の挿入を行う．

まず，最終スレッド以外のスレッド先頭の基本ブロック前に，fork 命令を挿入する．

次に，各スレッドの後続スレッドの fork 命令を実行する基本ブロックの前にスレッド終了命令を実行する基本ブロックを挿入する．また，スレッドの終端が関数の終端の基本ブロックである場合は，その基本ブロックの return 命令前にスレッド終了命令を挿入する．

また，各スレッド領域から脱出する制御フローの内，スレッド終了命令を行う基本ブロックへ遷移しないものに対し，投機スレッド破棄命令を実行する基本ブロックを挿入する．

最後に，各スレッド分割点において依存解消が必要なものに対し，依存解消のための命令移動及びコピーを行うことで，マルチスレッドコードの完成となる．

6 おわりに

本稿では，パスベーススレッド分割手法によるプログラムの自動並列化システムの実装を目指し，パスベーススレッド分割手法における，依存解消を用いたスレッド分割についての特徴および分割方法について述べた．

今後の課題として，スレッド分割時の最適な依存解消法の選択法及び，それを取り入れた詳細なスレッド分割点決定法を検討する必要がある．

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 (B)18300014，同 (C)19500037，同 (C)20500047）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] 小林崇彦，大津金光，横田隆史，馬場隆信，“パスの実行頻度を考慮したマルチスレッドコード生成手法の検討”，電子情報通信学会コンピュータシステム研究会 (CPSY)，信学技報，Vol.106，No.199，pp.7–12，2006

[2] 小川大仁，小林崇彦，大津金光，横田隆史，馬場隆信，“パスの実行頻度を考慮したスレッド分割手法と初期評価”，情報処理学会 第 69 回全国大会講演論文集，1K-9，pp.1-63〜1-64，2007