3A-4

ソースコード解析情報に基づく細粒度マルチスレッド制御法の検討

森山　英明†　　乃村　能成‡　　谷口　秀夫‡

岡山大学工学部†　岡山大学大学院自然科学研究科‡

1. はじめに

近年，1つのチップ上で，複数の命令流を並列実行するプロセッサが実用化されている．細粒度スレッドの並列実行を追及したプロセッサとして，Fuce（Fusion of communication and execution）プロセッサ[1]が開発されている．

Fuce において，プロセスは複数の細粒度スレッド（以降，マイクロスレッド）で構成される．マイクロスレッドは，継続による同期処理によってのみ制御され，一度実行が始まると中断されることなく処理される．また，継続の同期処理はハードウェアによって管理される．

本稿では，Fuce の細粒度マルチスレッド環境で OS によるスケジューリングを行うために，同時に走行するマイクロスレッド数の最大数を，一定以下に制限する制御法について示す．具体的には，ソースコードを静的に解析し，解析結果から優先度を設定し，ソースコードに制御を組み込むことで，ソースコード解析情報に基づく制御を行う．

2. 細粒度マルチスレッド環境での制御

OS による制御として，マイクロスレッドの同期カウンタ値を制御することにより，プロセス全体の動作速度を調整する制御手法を提案[2]した．ここで，制御手法には，次の2通りがある．

静的制御方式：

ソースコードにおいて，全てのマイクロスレッドの同期カウンタ値を増加させ，OS による特別な継続により制御する方法．

動的制御方式：

OS が各マイクロスレッドの起動状況を動的把握し，同期カウンタ値を制御する方法．

どちらの方式も，OS によって定期的に各マイクロスレッドへ継続を発行する（同期カウンタ値を減らす）ことで，同時走行マイクロスレッド数を制御する．なお，静的制御方式の考えに基づき，全てのマイクロスレッドに特別な継続を埋め込んだ状態から，ある確率に基づいてマイクロスレッドを選択し継続を発行することで，同時走行マイクロスレッド数を制限する制御方法も提案[3]している．この制御法は，プログラムの処理内容を考慮せずに継続順を決定するので，スケジューリングオーバヘッドが小さい特徴がある．反面，制御を効果的に行えない場合がある．そこで，本稿では，静的制御方式の考えに基づき，さらにプログラムの処理内容を考慮に入れてマイクロスレッドの継続順を決定することで，同時走行マイクロスレッド数を制限する方法を検討する．

3. ソースコード解析情報に基づく制御法

3.1 基本方針

同時走行マイクロスレッド数を制御するために，各マイクロスレッドの実行順序を考慮する．このために，制御の流れを表す継続関係グラフを導入し，その形状から各マイクロスレッドに優先度を設定する．OS がマイクロスレッドへ特別な継続を発行する順番は，優先度を基に決定する．

図1は，マイクロスレッドの継続関係の例をグラフで表している．例えば，th5は th2，th3からの継続に加え，OS からの特別な継続を発行されることで，動作を開始する．

ここで，各マイクロスレッドの優先度を，マイクロスレッド間の継続関係から設定する．

(1) 継続順

各マイクロスレッド間には，継続の順番から半順序関係が成立する．例えば，th1は th4に先立って実行される必要があるため，th1 ＞ th4の関係が成立する．この関係に基づいて，各マイクロスレッドに優先度を設定する．マイクロスレッド間の継続の順番関係は，継続関係グラフにおける，開始点からその点までの長さによって求める．

(2) 継続命令発行数

継続命令発行数の多いマイクロスレッドの実行が遅れると，同時走行マイクロスレッド数の変化が大きくなる原因につながる．継続関係グラフにおいて，あるマイクロスレッドの継続命令発行数は，グラフ上の点の出次数から求める

A Fine Grain Multithread Control Method based on Source Code Analysis
†Hideaki Moriyama,
†Faculty of Engineering, Okayama University
‡Yoshinari Nomura and Hideo Taniguchi
‡Graduate School of Natural Science and Technology, Okayama University

ことができる.
なお，優先度は，OS が継続を発行する順番を決定する目的でプログラム走行前に静的に設定されるので，プログラムの走行中に優先度は変化しない.

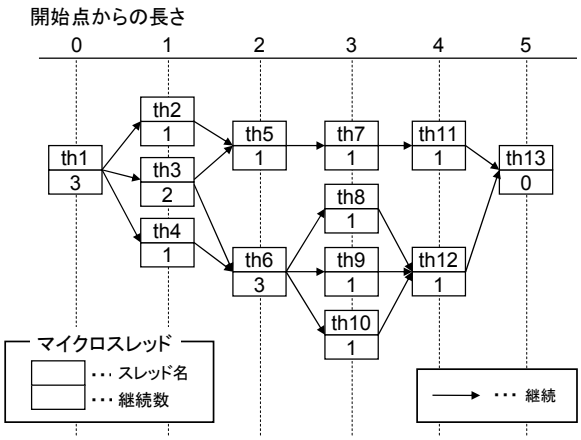

図 1 マイクロスレッド継続関係グラフ

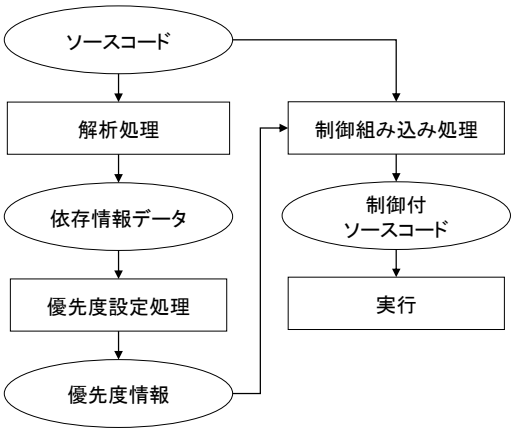

図 2 実行までの流れ

3.2 処理の流れ

図2に，ソースコード解析から実行までの処理の流れを示し，以下に具体的な制御の方法を説明する.

＜ソースコード解析処理＞

ソースコード解析処理では，ソースコードを入力として静的に解析を行い，マイクロスレッド間の継続関係を出力する．Fuce のアセンブラ言語のソースコードは，MIPS の命令セットに準じている．マイクロスレッド間の継続関係は，ソースコード中の継続命令を抽出することで明らかになる．出力である依存情報データは，図1のマイクロスレッド継続関係グラフ相当の情報を持つ.

＜優先度設定処理＞

優先度設定処理では，依存情報データを入力として，各マイクロスレッドへ優先度を設定し，マイクロスレッドへ継続を発行する順番を優先度情報として出力する．優先度は，依存情報データで表されるマイクロスレッド間の継続関係グラフを基に設定する．優先度は，継続関係グラフ上の点 v（マイクロスレッド v に相当する）の優先度を p(v)で表し，以下で定義する.

$$p(v) = a \cdot l(v) + b \cdot o(v)$$

l(v)は開始点から点 v までの最長路の長さを，o(v)は点 v の出次数を表す．式中の a，b は，l(v)，o(v)にそれぞれかかる重み係数である.

＜制御組み込み処理＞

制御組み込み処理では，優先度情報を基にソースコードに制御を組み込むことで，制御付きソースコードを出力する．ソースコードに，以下の処理を追加する.

(1) 各マイクロスレッドの同期カウンタ値を1増加する
(2) 優先度情報にしたがって継続を発行する制御マイクロスレッドを作成する

これにより，制御付きソースコードでは，各マイクロスレッドに継続が挿入され，制御マイクロスレッドによって，周期 Tω ごとにマイクロスレッド M 個へ継続を発行する．制御組み込み時に，Tω と M の値の設定を行うことで，同時走行マイクロスレッド数を調整する.

4. おわりに

ソースコードを静的に解析し，マイクロスレッド間の継続順と継続命令発行数に基づき，各マイクロスレッドの優先度を設定し，同時走行マイクロスレッド数を制限する制御法について記述した．今後は，制御の有効性を評価する予定である.

参考文献

1) 雨宮聡史，松崎隆哲，雨宮真人，“排他実行マルチスレッド実行モデルに基づくオンチップ・マルチプロセッサの設計，” 情報処理学会研究報告，Vol. 2003，No. 119，pp. 51-56，2003.
2) 乃村能成，雨宮聡史，日下部茂，谷口秀夫，雨宮真人，“細粒度マルチスレッド環境でのスケジューリングオーバヘッド低減機構，” 情報処理学会研究報告，Vol. 2004，No. 63，pp. 129-134，2004.
3) 小川泰彦，乃村能成，日下部茂，谷口秀夫，雨宮真人，“細粒度マルチスレッド環境でのスケジューリングオーバヘッド低減機構の評価，“ 情報処理学会研究報告，Vol. 2006，No. 15，pp. 47-53，2006.