2C-2

UNIX 環境を利用した細粒度マルチスレッド制御法の評価手法

小川　泰彦[†]　　乃村　能成[†]　　日下部　茂[‡]　　谷口　秀夫[†]　　雨宮　真人[‡]

岡山大学工学部[†]　　九州大学大学院システム情報科学研究院[‡]

1. はじめに

近年，プロセッサの性能向上のため，SMT や CMP といった 1 つのチップ上で複数の命令流を実行するプロセッサが実用化されている．我々は，並列処理と親和性の高いデータフローモデルを基盤として，スレッドの並列実行を追及した Fuce プロセッサ[1]を開発している．

Fuce は，複数のスレッド実行ユニットを搭載し，複数のスレッドを並列に処理することができる．Fuce におけるスレッド（以降マイクロスレッド）は，粒度が非常に細かく，継続概念によるイベント駆動によってのみ制御され，排他的に実行され，中断されることがないという特徴を持っている．マイクロスレッドは，ハードウェア（以降 H/W）で管理されるため，マイクロスレッド間の継続による同期処理は，H/W によって高速に処理される．また，同期が解決したマイクロスレッドは，H/W のキューに投入され，高速に処理される．したがって，Fuce では，大量のマイクロスレッドを高速に実行可能である．

しかしながら，マイクロスレッドは，OS の関知しないところで次々と実行される．また，プロセスは，多数のマイクロスレッドの集合体として構成されるため，このままでは，OS がプロセスのスケジューリングを行うことができない．

そこで，マイクロスレッドの継続を制御する同期カウンタ値を増減することで，マイクロスレッド間の同期を意図的に遅延させ，プロセス全体の動作速度を調整するという手法を提案[2]している．

ここでは，提案した細粒度マルチスレッド制御法の効果を評価する方法として，UNIX 環境を利用した評価手法について述べる．

2. 評価手法の要求条件

提案した細粒度マルチスレッド制御法の効果を評価するため，制御によりマイクロスレッドの動作がどのように変化したかを観測できる必要がある．しかし，マイクロスレッドの動作を観測するには，以下のような課題がある．

(1) 現在，Fuce は開発中である．また，Fuce の仕様として，マイクロスレッドの動作の詳細な履歴を取ることは難しい．

(2) VHDL で作成した Fuce プロセッサのシミュレータでは，評価対象を変えるごとに検証用のコードを書き直す必要があるため，工数や柔軟性の面に問題がある．

そこで，上記の課題に対処し，マイクロスレッドの動作を観測する方法として，UNIX 環境を利用した評価手法を提案する．

3. 評価手法

3.1 マイクロスレッドの動作シミュレーション

マイクロスレッドの動作を UNIX 環境上のプロセスでシミュレートする．具体的には，「マイクロスレッド」を「プロセス」に対応させ，「マイクロスレッド間の継続による同期制御」を「プロセス間のセマフォによる同期制御」によって実現する．「マイクロスレッドを生成する H/W 命令」を「プロセスを生成するシステムコール fork」に対応させる．このようにして，Fuce 上のマイクロスレッドを UNIX 環境上のプロセスとして動作させる．

提案制御法では，OS がマイクロスレッドの動作を制御するために，マイクロスレッドに「OS からの継続」を挿入する．これにより，マイクロスレッドは，OS からの継続がない限り，同期が解決せず実行されない．この機構もセマフォによって実現する．具体的には，制御を行うプロセスを用意し，このプロセスが継続の挿入に相当する制御用のセマフ

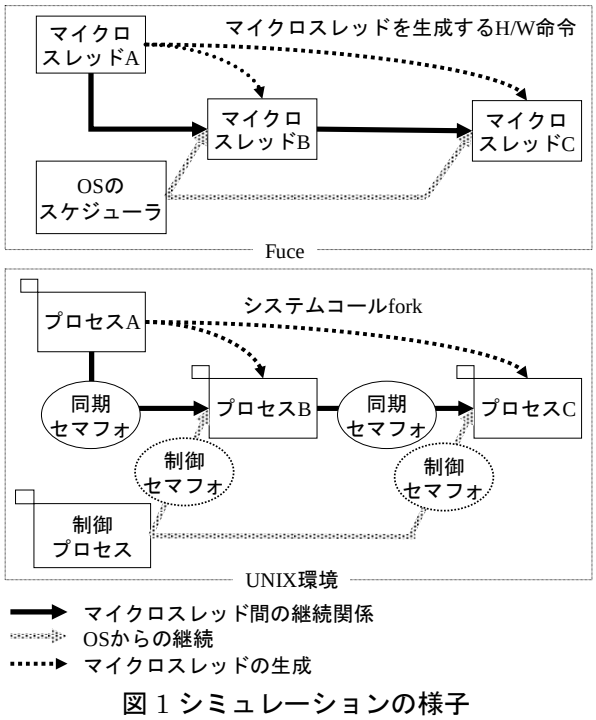

図 1 シミュレーションの様子

A Method for Evaluation of Multithread Control Mechanism on UNIX
†Yasuhiko Ogawa, Yoshinari Nomura and Hideo Taniguchi
†Faculty of Engineering, Okayama University
‡Shigeru Kusakabe and Makoto Amamiya
‡Graduate School of Information Science and Electrical Engineering, Kyushu University

ォを獲得することで，プロセス間の同期が解決しないようにする．この様子を図 **1** に示す．図 1 では，マイクロスレッド A がマイクロスレッド B と，マイクロスレッド C を生成する．そして，A が B へ継続し，B が C へ継続する．

3.2 観測方法

UNIX 環境で，マイクロスレッドの動作をプロセスにシミュレートさせ，プロセスを並列に走行させようとしても，シングルプロセッサでは同時には 1 つのプロセスしか走行しない．しかし，複数の実行ユニットを持つ環境では，READY キューにつながれているプロセスは，実行ユニットを割り当てられ，並列に処理される．したがって，UNIX OS のスケジューラの READY キューの長さを観測することで，並行に動作するプロセスの動作を観測する．

図 1 に示したようにプロセスを制御し，その動作を観測するマイクロスレッドシミュレータを Linux（Kernel 2.4.7）上に実装した．

3.3 観測結果

図 **2** に示す継続関係にあるマイクロスレッドを取り上げ，動作を観測した．VHDL で作成した Fuce シミュレータのマイクロスレッドの走行時間は，約 500 クロックサイクル分とし，マイクロスレッドシミュレータのプロセスの走行時間は，約 500 ミリ秒とした．制御を行わない場合と，先行して実行されるマイクロスレッドから継続され同期が解決したマイクロスレッドの実行開始を一定時間遅延させるという制御を行った場合について観測した．Fuce シミュレータの観測結果を図 **3** に示し，マイクロスレッドシミュレータの観測結果を図 **4** に示す．

マイクロスレッドの制御を行わない場合と，行う場合それぞれについて，Fuce シミュレータと，マイクロスレッドシミュレータで同一の結果が得られた．

3.4 利点と欠点

本評価手法は，以下の利点を持つ．

(1) UNIX 環境を利用して実現するため，Fuce プロセッサがなくても評価可能である．

(2) マイクロスレッドの詳細な走行履歴をとることが可能である．

(3) マイクロスレッドシミュレータは，マイクロスレッドの継続関係を記述した設定ファイルと，パラメータの組合せによって，柔軟に評価を行うことができるため，工数の削減が期待できる．

(4) マイクロスレッドシミュレータは，C 言語で記述されており，評価の要求に合わせて，拡張や改良を行うことが比較的容易である．

しかし，本評価手法は，以下の欠点も持つ．本評価手法は，プロセスを外見上マイクロスレッドのように動作させるが，マイクロスレッドの内部的な処理はシミュレートしない．そのため，処理内容まで考慮したシミュレーションはできない．

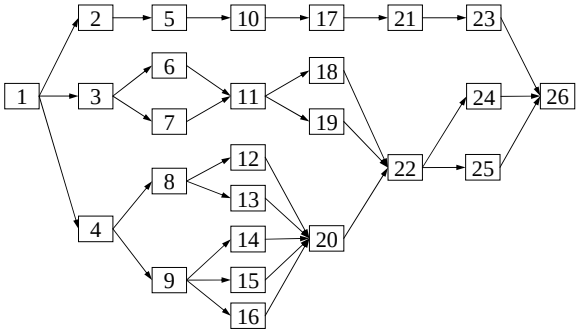

図 2 マイクロスレッド間の継続関係

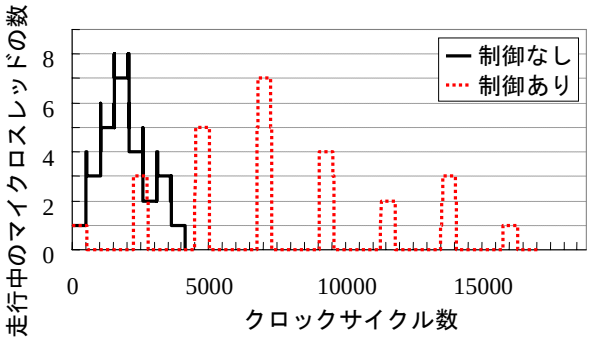

図 3 Fuce シミュレータの観測結果

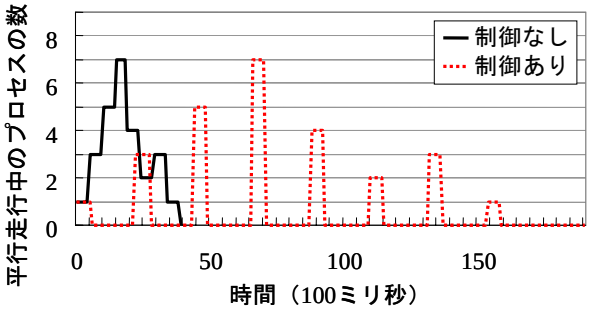

図 4 マイクロスレッドシミュレータの観測結果

4. おわりに

UNIX 環境を利用して，プロセスをマイクロスレッドのように動作させ，その動作を観測することで細粒度マルチスレッド制御法の評価を行う手法を述べた．また，VHDL のシミュレータを用いた手法で得られる結果と比較し，本手法で得られる結果は，信頼できるものであることを示した．今後は細粒度マルチスレッド制御法の評価を行う予定である．

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.