

バイナリ変換によるマルチスレッド化コード変換システムとその評価*

安濃 隆弘 小野 喬史 横田 昌之 大津 金光 横田 隆史 馬場 敬信†

5ZB-05

宇都宮大学工学部情報工学科‡

1 はじめに

従来のプロセッサシステムの高速化には限界が見えて来た。これからのプロセッサの高速化にはマルチスレッド実行を前提としたプログラミングを行うことが必要となる。しかし、マルチスレッドプログラミングは容易ではないため、シングルスレッドコードからマルチスレッド化コードを自動生成するシステムが有効となる。

シングルスレッドコードからマルチスレッドコードを生成するには、ソースコードレベルにおいて行うことが最良の方法であるが、マルチスレッド化によるアプリケーションの高速化にはソースコードの参照が必要となる。しかし、アプリケーションプログラムのソースコードはその全てが参照可能なわけではない。

以上を背景とし、我々はバイナリ変換によるマルチスレッド化と実行時最適化を行うことで、既存のアプリケーションとの互換性を維持しつつ、実行性能を高めるシステム^[2]を提案している。本稿では、そのパイロットシステムとして試作・評価したマルチスレッド化コード自動変換システムについて報告する。

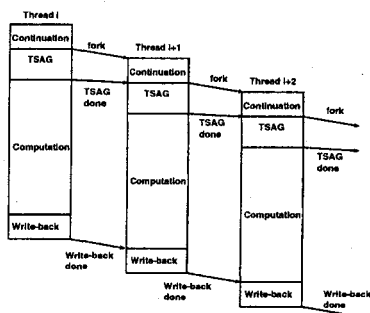


図 1: スレッドパイプライン

2 マルチスレッドモデル

本研究が対象とするマルチスレッドモデルは、各スレッドが 4 つの単純なステージで構成されるスレッドパイプライン

パイプラインモデルである。図 1 にスレッドパイプラインの実行の様子を示す。

対象となるコードをループのイテレーション単位により分割してスレッド化し、その後各スレッドを Continuation, TSAG (Target Store Address Generation), Computation, Write-back の 4 つのステージに分割する。Continuation Stage では、主に次スレッドの生成とループ変数の計算を行う。TSAG Stage では、スレッド間のデータ依存となるメモリのアドレスを Memory Buffer に対して通知する。Computation Stage で、計算本体のコードを実行する。最後に、Write-Back Stage で結果の書き戻しを行い、スレッド実行を終了する。

3 システム構成

図 2 に本研究で試作したバイナリレベルマルチスレッド化ツール **Multithread code GENERator (MGENE)** の構成を示す。

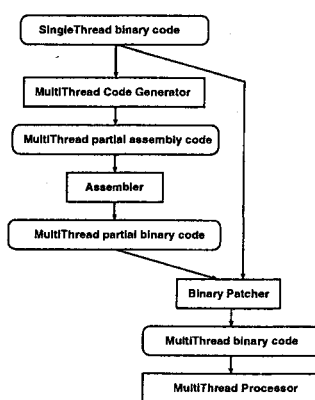


図 2: システムの構成

シングルスレッドコードからマルチスレッド化コードを生成するために、まず対象アプリケーションのシングルスレッドコード中で高速化が期待できるサブルーチンを決定する。

MGENE はシングルスレッドバイナリコードを入力し、上で指定したサブルーチン部分をマルチスレッド化し、その部分的なアセンブリコードを生成する。

MGENE を用いて生成した部分バイナリコードとシン

*Evaluation of a Binary-Level Multithreading System

† Takahiro Anno, Takafumi Ono, Masayuki Yokota, Kanemitsu Ootsu, Takashi Yokota, and Takanobu Baba

‡ Department of Information Science, Faculty of Engineering, Utsunomiya University

グルスレッドコードを **Binary Patcher** を用いて結合し、最終的なマルチスレッドバイナリコードを生成する。

4 マルチスレッド化手法

MGENEにおけるマルチスレッド化のアルゴリズムは以下の通りである。

まず、対象アプリケーションのバイナリコードの解析と中間表現への変換を行う。次に、基本ブロックへの分割および制御フロー解析を行い、ループ構造を検出する。その後、データフロー解析により、ループ変数とイテレーション間依存を検出する。最後に、スレッドパイプライニングに基づきマルチスレッド化する。

マルチスレッド化コードにはシングルスレッドコードには存在しないスレッド制御命令が挿入される。計算部分に対してスレッド制御にかかるコストが大きい場合は、ループアンローリングを行うことで1イテレーション当たりの Computation Stage の処理量を大きくし、スレッド制御命令によるコストの影響を小さくすることができる。

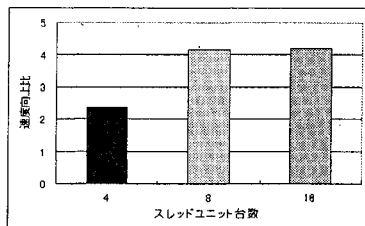


図 3: jpeg における速度向上比

5 評価

バイナリレベルにおいてマルチスレッドコードを生成し、その実行性能を評価した。

スレッドパイプライニング実行モデルを実現するために、本研究では Superthreaded Architecture^[4] を用いている。

Superthreaded Architecture で用いられる各スレッドユニットは従来の汎用プロセッサに、スレッドユニット間での通信・同期を行うための機構を追加したものである。1プロセッサはスレッドユニットを複数並べた構成であり、各スレッドユニットで L1 キャッシュを共有している。

評価には、スレッドパイプライニングモデルであるスーパースレッドアーキテクチャシミュレータ SIMCA^[3] (SIMulator for Multi-threaded Computer Architecture)

を用いた。また、対象となるシングルスレッドコードのバイナリは、SIMCA 用 gcc クロスコンパイラ (最適化オプション-O2) を用いて生成したコードを用いた。

評価用のプログラムとして、SPECint95 の jpeg を用い、シングルスレッド実行とマルチスレッド実行により、それぞれの対象としたループ部分の計算のクロック数を計測し速度向上比を求めることで評価を行った。

図3にループアンローリングを8回行った際の速度向上比を示す。速度向上比はシングルスレッド実行時を1とする。スレッドユニットが4台のとき2.3倍、8台で4.18倍の速度向上を得た。

6 おわりに

今後主流になると予想されるマルチスレッドプロセッサの普及を前提として、バイナリレベルでのシングルスレッドコードからマルチスレッド化コードへの変換を行うツール MGENE を試作し、SPECint95 の jpeg を用いて評価を行った。

アプリケーションのマルチスレッド化は対象となるプログラムのループ部分に着目し、スレッドパイプライニングモデルに基づき変換を行った。その際、ループアンローリングを用いることで、高速化できることを示した。

今後の課題は、様々なアプリケーションへの対応と実行性能の向上を得ることである。

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) 課題番号 12680328) の援助による。

参考文献

- [1] 大津金光, 小野喬史, 馬場敬信: バイナリレベルにおけるマルチスレッド化コード生成手法, 情報処理学会研究報告 (ARC-141), p41-46 (2001)
- [2] K.Ootsu, T.Yokota, T.Ono, and T.Baba. A Binary Translation System for Multithreaded Processors and its Preliminary Evaluation, Proc. 5th Workshop on Multithreaded Execution Architecture and Compilation, pp13-21(2001)
- [3] J. Huang. The SIMulator for Multi-threaded Computer Architecture (SIMCA), Release 1.2. <http://www.mount.cs.umm.edu/Research/Agassiz/simca.html>
- [4] J. Y. Tsai, J. Huang, and et al, "The Superthreaded Processor Architecture," IEEE Transactions on Computers, Vol. 48, No.9 (1999)