1K-6

マルチスレッドコードへのループアンローリング適用による高速化

増澤 英樹 † 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が重要となってきている．そこで，我々は既存の逐次バイナリコードを基にしてマルチスレッド化を行うことによりプログラムの性能を向上させることを目的としたバイナリ変換システムを開発している．

マルチスレッド実行にはスレッド制御処理のオーバーヘッドが生じ，このオーバーヘッドが大きいと性能低下してしまう．そこで，本研究ではスレッドコード最適化の一つであるループアンローリングに着目した．ループアンローリング適用の際，アンローリング回数が大きいほど性能向上の効果が得られる反面，回数が大きすぎるとキャッシュミスによる性能低下という問題がある．そのため，これらのトレードオフを明確にし最高性能となるアンローリング回数を決定する手法の開発が必要となる．本稿ではマルチスレッドコードに対してアンローリング回数を変化させてシミュレーションを行うことでマルチスレッド実行時のキャッシュミスによる性能への影響を定量的に評価する．

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としてスレッドパイプライニング実行モデルを前提としている．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 つのステージから構成される．Continuation ステージではループ変数などの後続のスレッドが実行開始時に必要となる値の計算を行い，次のスレッドを起動する．TSAG(Target Store Address Generation) ステージではスレッド間でデータの依存の可能性が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 ステージではスレッドにおける計算処理の本体を実行

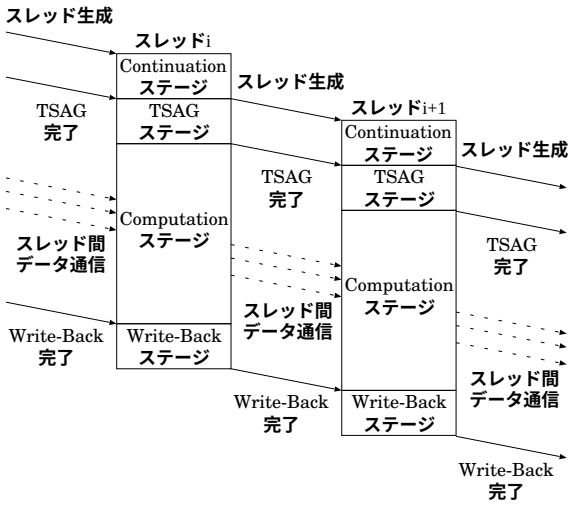
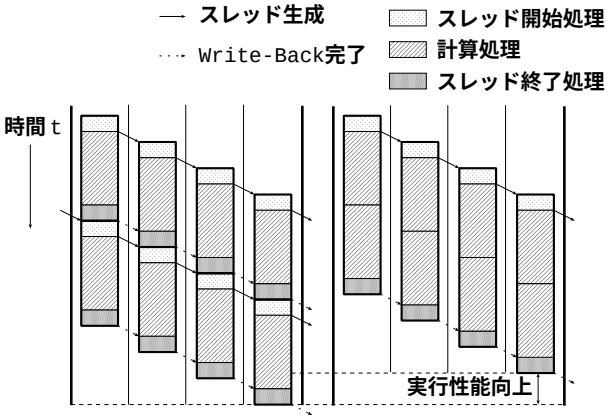


図 1: スレッドパイプライニングモデル



(a)ループアンローリングなし (b)ループアンローリングあり

図 2: ループアンローリングによる性能向上

する．Write-Back ステージではスレッド実行の結果をメモリへ書き戻す．

スレッドパイプライニング実行モデルでは，投機的に実行されたスレッドは後に破棄される可能性がある．そのため，投機状態のスレッドのメモリへの書き込みは一時的にメモリバッファに格納しておき，スレッドの実行が投機状態でなくなった際にメモリバッファからデータキャッシュを介してメモリへと書き戻される．

我々のシステムはこのスレッドパイプライニング実行モデルを基にし，ループのイテレーションを単位としたマルチスレッド実行により高速化を達成する．しかし，マルチスレッド実行にはシングルスレッド実行にはないスレッド制御処理があり，そのオーバーヘッドが大きいと実行性能低下の可能性がある．そこで，本研究ではループアンローリングに着目した．ループアンローリングの適用により本来の計算処理量を増えるので，スレッド制御処理のオーバーヘッドが相対的に小さくなり性能が向上する．

3 ループアンローリング適用による利点と欠点

3.1 ループアンローリング適用による利点

ループアンローリングの適用によってスレッド制御処理のオーバーヘッドが相対的に小さくなる様子を図 2 に示す．(a) では 1 イテレーションを 1 スレッドとして (b) ではループアンローリングの適用により 2 イテレーションを 1 スレッドとしているマルチスレッド実行を示したものである．(b) ではループアンローリングの適用により，1 回のスレッド制御処理のオーバーヘッドで 2 イテレーション分の計算処理が割り当てられることとなるので (a) と比べて半分のオーバーヘッドで同じイテレーション数の計算処理ができる．また，ループアンローリングの適用により命令スケジューリング等の最適化機会が増加し，命令レベル並列性の向上やループの終了判定の削減等の従来のループアンローリングの利点も得られるので大きな性能向上が望める．

3.2 ループアンローリング適用による欠点

アンローリング回数を大きくした方が前項で述べた利点が増える一方で，アンローリング回数を大きくしすぎるとキャッシュミスによる性能低下が起こる．キャッシュミスには命令キャッシュとデータキャッシュミスが

Speed-up by Loop Unrolling for Multithreading
†Hideki Masuzawa, Kanemitsu Ootsu, Takashi Yokota and Takanobu Baba
Department of Information Science, Faculty of Engineering, Utsunomiya University (†)

ある．

命令キャッシュミスはループアンローリングの適用によりコードサイズが増加することでミスが起き性能低下する．ループアンローリングの適用による命令キャッシュミスはシングルスレッド実行でも起こりうる現象である．

データキャッシュミスについてはスレッドパイプライニング実行モデルを前提としたマルチスレッド実行特有のもので，ループアンローリングの適用により 1 スレッドあたりのストア命令が増えることから Write-Back ステージでメモリバッファからメモリへと書き戻す際にデータキャッシュミスを起こすことで性能低下の原因となる．

3.3 最適なアンローリング回数

ループアンローリングはアンローリング回数が大きい方が性能向上が得られる反面，大きすぎるとキャッシュミスによる性能低下が起こるという問題がある．そこで，アンローリング回数を増やすことでの性能向上とキャッシュミスによる性能低下のトレードオフを明確にする必要がある．

そこで，本稿では最高性能を達成するアンローリング回数を実際のプログラムを使って評価し，評価結果から最高性能となるアンローリング回数を決定する．

4 評価

4.1 評価方法

評価には SimpleScalar の *sim-outorder* をベースとしたスレッドパイプライニングモデルシミュレータである SIMCA[1] を用いる．各スレッドユニットが *sim-outoreder* をベースとしており，命令パイプラインやキャッシュの振る舞いを正確にシミュレートすることができる．評価対象アプリケーションとして，SPECfp2000 の *171.swim* を用いる．ループアンローリングの対象としたのはサブルーチン *calc2_()* に含まれるループの 1 つである．データセットは test を使用する．シミュレーションに使用した命令セットは 1 命令 8 バイトである．

評価対象であるプログラムは 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 を適用してコンパイルし，バイナリコードを生成し，その生成されたコードに対してバイナリレベルでマルチスレッド化を行い，ループアンローリングおよび他の最適化を行う．

評価は，スレッドユニット台数を 4 台とし，対象ループの開始から終了までの実行サイクル数を計測し，シングルスレッド実行時のサイクル数との比を求めることにより速度向上率を求めることで行う．

4.2 評価結果

評価の結果，1 スレッドあたりのイテレーション数が 8 回の時，速度向上率が最高値を示し 4.17 倍となった．

図 3 より 1 スレッドあたりのイテレーション数回数が 24 回の時，速度向上率が急激に低下しているのは，1 スレッドあたりのイテレーション数が 21 回を超えると命令数が命令キャッシュサイズを超えるため，命令キャッシュミス率が大幅に増えたため性能低下したと考えられる．

図 4 より 1 スレッドあたりのイテレーション数が 12 回から最高性能より少しずつ性能が低下している．これは，Write-Back ステージでのデータキャッシュミス率が高くなってきたためと考えられ，今回のプログラムではミス率が 10~ 15%であった．また，Write-Back ステージのデータキャッシュミス率が，1 スレッドあたりのイテレーション数が 1 回から 4 回の時にかけて低くなっている．これは，イテレーション間でメモリへ書き込むアドレスが連続していることと，データキャッシュには 4 つの倍精度データが 1 つのライン上に乗ることから，1 スレッドあたりのイテレーション数が 4 回の時までは 1 つのライン上に 1 スレッド分のデータが乗るためキャッシュミス率が低下したのではないかと考えられる．

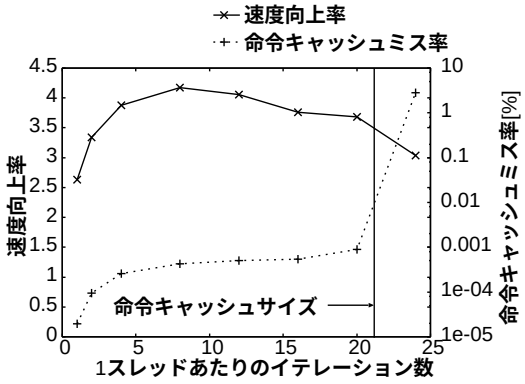


図 3: 速度向上率と命令キャッシュミス率

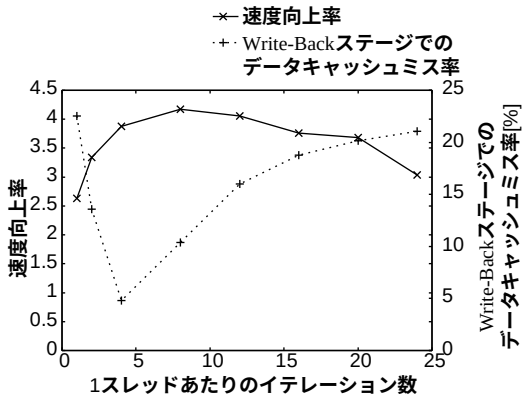


図 4: 速度向上率とデータキャッシュミス率

5 おわりに

ループアンローリングの適用による性能向上と性能低下のトレードオフを明確にするために，性能が最大となるアンローリング回数を実際のプログラムで評価した．評価の結果，今回のプログラムではスレッドコードサイズが命令キャッシュサイズを超えると命令キャッシュミスによる性能低下を起こし，Write-Back ステージのデータキャッシュミス率が 10~ 15%を超えると性能低下の原因となることがわかった．

今後の課題として，評価対象アプリケーションを増やしスレッドコードサイズやメモリアクセス数からアンローリング回数の決定手法を確立し，最高性能を達成するアンローリング回数の決定戦略をシステムに組み込み，性能評価を行うことが挙げられる．

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] J. Huang, “The SImulator for Multi-threaded Computer Architecture(Release 1.2)”, http://www.cs.umn.edu/Research/Agassiz/Tools/SIMCA/simca.html.

[2] 増澤英樹，大津金光，横田隆史, 馬場敬信, “バイナリレベルマルチスレッド化におけるループアンローリングの効果”, 電子情報通信学会コンピュータシステム研究会，Vol.106, No.436, pp.69-74, 2006.