5ZB-5

データ依存関係に着目したスレッド分割方法の検討

三木 大輔　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

本研究室では，バイナリレベルでシングルスレッドコードからマルチスレッドコードへの変換を行うシステムの研究開発を行っている[1]．

これまで，数値計算系アプリケーションではループのイテレーションを単位にスレッド分割することにより，高い速度向上を得ることができた．しかし，非数値計算系アプリケーションでは複雑な依存関係のため，この方法では速度向上を達成することが出来ない．そこで，複雑な依存関係に対する効果的なスレッド分割手法が研究されている[3]．

本研究では，スレッド間データ依存が速度向上に大きく影響することに注目し，データ依存関係を考慮したスレッド分割を行う．そして，その分割手法をSPECint95のアプリケーションに適用し，その効果をシミュレーションにより検証する．

2　スレッド分割

本研究におけるスレッド分割方法は，制御フローに基づくブロック化とデータ依存を考慮したスレッド割り当ての2つから成る．以下にその手順を述べる．

2.1　制御フローに基づくブロック化

マルチスレッド化は実行頻度の高い関数に対して行う．まず始めに関数の制御フローグラフを生成する．次に，制御フローグラフから入口と出口が一つとなる基本ブロックの集合を生成し，それを一つのマクロブロックとする(図1(a))．マクロブロック生成の際は，マクロブロックのネストが起きないようにし，可能な限り小さなマクロブロックができるように分割を行う．

このブロック化により，関数の開始から終了まで実行が一本のマクロブロック列となる．実行の流れを一本化することで，自分のスレッドの先頭で次のスレッドを生成する際に，次のスレッドの実行開始アドレスを一意に決定することができる．このマクロブロックがスレッド割り当ての最小単位となる．最も単純なスレッド割り当てでは，一つのマクロブロックを一つのスレッドに割り当てる．このブロック化では，ループに関しては全体が一つのマクロブロックとなり，イテレーション単位のマルチスレッド実行は行われない．

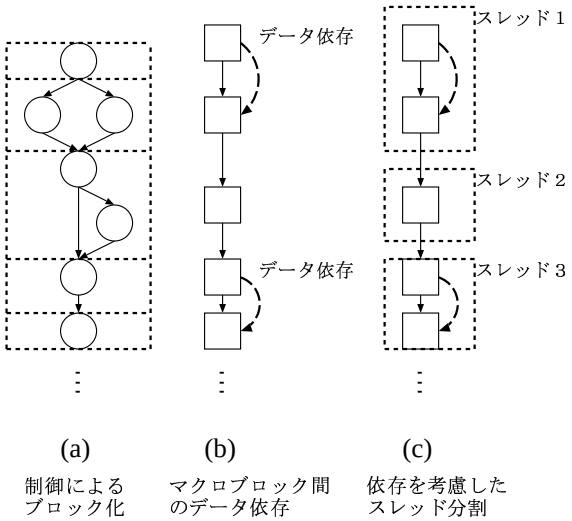

図 1. スレッド分割

2.2　データ依存を考慮したスレッド割り当て

各マクロブロック間にはデータ依存が存在する可能性がある図(1(b))．よって，前節で述べたように一つのマクロブロックを一つのスレッドとした場合，スレッド間でデータ依存となる変数で同期をとる必要が生じる．このためスレッドの実行停止によって速度向上が阻害されしまう．よって，データ依存関係を考慮した分割を行う必要がある．

まず，関数の先頭のマクロブロックから順次依存関係を見ていく．もし後続のマクロブロックと依存関係があった場合はそれを一つのスレッドとして統合し，さらに次のマクロブロックとの依存関係を見る．次のマクロブロックと依存関係がなかった場合は，スレッドサイズ等は考慮せず，そこで無条件にスレッドの分割を行う(図1(c))．この方法により，スレッド分割点の前後でデータ依存が無くなるため，スレッドは実行を停止することなく並列処理を行うことができる．

3　評価

3.1　評価方法

評価にはスレッドパイプライニングモデルシミュレータのSIMCA[2]を用いた．評価手順は，まず対象アプリケーションのソースコードを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する．そして，その生成されたバイナリコードに対して前述のマルチスレッドコードへの

A Consideration of Thread Partitioning Method based on Data Dependencies
† Daisuke Mitsugi, Kanemitsu Ootsu, Takashi Yokota and Takanobu Baba
‡ Department of Information Science, Faculty of Engineering, Utsunomiya University

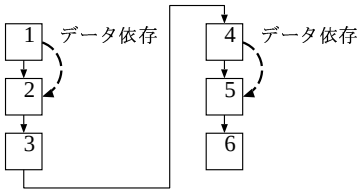

図 2. killtimeのフローグラフ

変換を適用する．

評価対象としてSPECint95のm88ksimの関数killtimeを使用した．評価方法は，killtimeの始まりから終りまでの実行サイクル数を計り，速度向上率を求める．速度向上率はシングルスレッド実行サイクル数をマルチスレッド実行サイクル数で割ることで計算する．入力データセットにはtestとtrainを用い，スレッドユニット台数を4, 8台としてシミュレーションを行った．

3.2 評価結果

killtimeでは，図2のようにマクロブロックが6つ生成される．よって，データ依存を考慮しない場合，一つのマクロブロックを一つのスレッドに割り当てるので，6つのスレッドに分割される．これを単純分割とする．次に，このマクロブロック間のデータ依存を調べると，マクロブロックの前後で依存があるのは1, 2番目のマクロブロック間と4, 5番目のマクロブロック間である．よって，データ依存を考慮してマクロブロックの統合を行った場合，4つのスレッドが生成される．

単純分割とデータ依存関係を考慮した分割の2つのスレッド分割方法による実行結果を図3に示す．データセットtestでは，単純分割を行った場合にはスレッドユニット数4台で速度向上率が0.91となり逆に速度低下となっている．これは，マルチスレッド化や依存変数の値を転送するオーバーヘッドが大きいためと考えられる．それに対し，データ依存を考慮した場合では1.17の速度向上率となり，速度向上を達成している．trainにおいても，単純分割で1.03，データ依存考慮分割で1.36の速度向上率となり本手法によるスレッド分割が有効であることがわかる．なお，testに比べtrainの速度向上率が大きいのは，スレッドの実行サイクル数が影響していると考えられる．testよりもtrainの方が実行サイクル数が大きいが，スレッド制御やスレッド間データ転送のオーバーヘッドは同じため，これらのオーバーヘッドが相対的に小さくなり，より大きな速度向上率が得られた．

4 おわりに

本稿では，マルチスレッド実行のためのスレッド分割において，データ依存関係に着目した分割を行った．シミュレーションを行った結果，データ依存を考慮せず単純に分割した場合に比べて，高い速度向上率を得

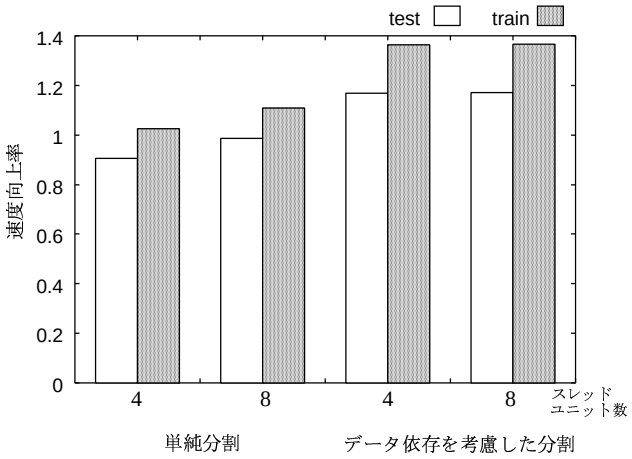

図 3. マルチスレッド実行による速度向上率

ることができた．

今後の課題として，本手法ではマクロブロック間でデータ依存が存在する場合には必ず一つのスレッドに統合したが，データ依存が関数全体に渡って連鎖している場合には全く分割を行うことができない．そこで，スレッド間データ依存の有無だけで分割せず，データ依存によってどの程度速度低下するかを考慮した分割が必要となる．また，スレッドのサイズを分割の基準として取り入れることが考えられる．さらに，今回使用したkilltimeではマクロブロック中にループを含むが，一般的にループの実行サイクル数は大きいため，スレッド間の実行サイクル数に不均等が生じてしまう．本手法ではループを特に考慮していないので，今後はスレッド分割の際にスレッドとなるコードの性質を考慮する必要がある．

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)14380135，同(C)16500023，若手研究14780186）の援助による．

参考文献

[1] 大津 金光，小野 喬史，横田 隆史，馬場 敬信，“バイナリレベルマルチスレッド化コード生成手法とその評価，” 情報処理学会論文誌ハイパフォーマンスコンピューティングシステム，Vol.44, No.SIG-1 (HPS 6), pp.70-80, 2003.

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

[3] 田代 大輔，バルリ ニコ デムス，坂井 修一，田中 英彦，スレッド投機実行におけるエッジに着目したスレッド分割手法，情報処理学会報告書計算機アーキテクチャ研究会2003-ARC-153, Vol.2003, No.40, pp.67-72, May, 2003.