3A-6

2パス限定投機システムによる難並列化ループの高速化

十鳥弘泰 † 大津金光 † 横田隆史 † 馬場敬信 †
† 宇都宮大学大学院工学研究科情報システム科学専攻

1 はじめに

近年，マルチコアプロセッサを搭載したコンピュータが普及しているが，複数のコアを十分に活用してプログラム単独の実行性能を高めるためには，プログラムの適切なマルチスレッド化が必要である．特に，非数値処理系のプログラムは，その複雑な制御構造によりスレッドレベル並列性の抽出が困難とされている [1].

そこで，プログラムをとおして頻繁に実行されるループに対して制御依存解析を行い，ループの 1 イテレーションを単位とした細粒度のスレッドレベル並列性を抽出する．ループのうち，頻繁に実行される実行経路 (パス) を抽出し，それらを投機的にマルチスレッド実行することで，複雑な制御構造のループに対しても高速化を達成できると考えた．

本稿では，複雑なループに有効な投機的マルチスレッド実行方式である，2 パス限定投機方式と，同方式を実現したアーキテクチャである 2 パス限定投機システム PALS[2] について述べる．そして実際のプログラムに対する最適化手法とその有効性について述べる．

2 2パス限定投機システム

2.1 2パス限定投機方式

2 パス限定投機方式ではループの 1 イテレーションを 1 つのスレッドとして実行する．ループにおけるパスのうち，プログラムをとおして実行割合の高い上位 2 本のどちらが実行されるかを予測し，投機的にマルチスレッド実行することでループの高速化を達成する．投機対象を限定することにより，2 本のパスに対する投機スレッドコードを用意するだけでよく，イテレーションにおける無用な命令コードの大幅な削除が可能になる．また，パスの予測は二択となるため予測器の構成も簡単になる．

2.2 システムの設計

2 パス限定投機方式では，ループの 1 イテレーションをマルチスレッド実行の対象としているため，スレッド制御にかかるオーバヘッドが実行性能に大きな影響を与える．スレッドの起動やスレッド間通信，投機失敗時の回復処理等にかかるオーバヘッドが大きくなると，マルチスレッド実行の並列性が失われ，性能が大きく低下する．そこで PALS では，(1) パスの予測，(2) 投機実行の制御，(3) 投機失敗時のプログラム整合性の保証，以上をハードウェアで行うことにより，投機実行にかかるオーバヘッドを最小限に抑える．

図 1 に PALS のハードウェア構成を示す．図中の四角はそれぞれハードウェア機構を表しており，矢印は機構間における通信の関係を表している．

マルチスレッド制御機構 (Thread Management Unit: TMU) は内部にパス予測器を持つ．従来の汎用プロセッサに相当しリング状に接続されたスレッド機構 (Thread Unit: TU) に対して，TMU はスレッド生成の指示と実行状態の管理を行う．TU は TMU から受け取ったパス予測結果を基に，該当するパスの投機スレッドコードを実行する．Memory Buffer (MB) および Load Shelter (LS) は，投機的なメモリアクセスを適切に処理するための機構である．PALS では，MB と LS を併せてメモリアクセス機構 (Memory Access Unit: MAU) と呼ぶ．TU と MB は 1 対 1 で接続され，TU は全てのメモリアクセスを MB に対して行う．また，隣接する MB 間は TU と同様の双方向通信を行うリング構造となる．LS は TU からの投機的ストアデータを MB が保証するための補助的役割を担う記憶機構であり，全ての MB と接続される．

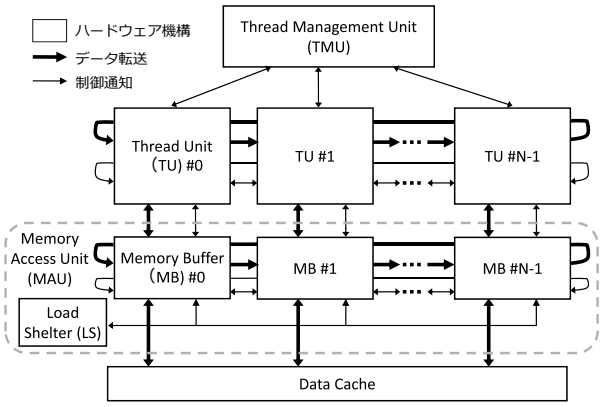

図 1: PALS のハードウェア構成

3 ベンチマークプログラムを用いた PALS の性能評価

本節では，SPEC CINT2000 ベンチマークのループに対して 2 パス限定投機方式を適用する．ベンチマークより，(1) 2 本以上のパスが存在する，(2) イテレーション間に依存関係が存在する，以上 2 点の条件を満たすループを選択する．そして，PALS をクロックレベルで模擬するシミュレータ上でプログラムを実行する．

3.1 対象ループの構造

対象は，`181.mcf` の関数 *dual_feasible()* 中のループである．本ループは，入力データをもとに作られる木構造のデータに対して，イテレーションごとに木の各

Speed-up of Hard Parallelization Loops by Two-Path Limited Speculation System
†Hiroyoshi Jutori, Kanemitsu Ootsu, Takashi Yokota and Takanobu Baba
Department of Information Systems Science, Graduate School of Engineering, Utsunomiya University (†)

ノードへポインタを用いてアクセスし，読み出したデータを使用した比較を行う．本ループでは，比較結果の成否に対応してパスが2本存在している．また，ポインタのアドレス計算に用いるレジスタがイテレーション間の依存関係となっている．

3.2 PALSでの実行に適したコードの作成

本ループでは，アドレス計算用レジスタをイテレーションの終端で更新し，直後のイテレーションの先頭でのロードに使用する．このため，単純なマルチスレッド化を適用した投機スレッドコード (以下，オリジナル版と呼ぶ) では，直前のイテレーションでの計算が終わるまでロードを行うことができず，スレッドの並列性をほとんど得ることができない．そこで，同期待ち時間を低減するため，リストスケジューリングをもとにしたコードスケジューリング [3] をオリジナル版に適用する (以下，スケジューリング版と呼ぶ)．

PALSではMBを介したメモリアクセスを行うため，ロード命令のコストは他の命令より高くなる．そこで，スレッドコードの先頭でアドレス計算用レジスタのコピーを行い，本来のアドレス計算より早くレジスタの更新と次スレッドへの送信を行う．また，データの比較に用いる変数は，ループの実行を通して同じ値が使われるが，アセンブリコードではイテレーションごとにメモリから値をロードしている．このため，このロード命令を本ループの直前に行い，空いているレジスタに値を格納しておくことでロード命令を1つ削除することができる (以下，最適化版と呼ぶ)．

図2に各投機スレッドコードにおけるアドレス計算用レジスタの位置の変化を示す．図において，`$7`がアドレス計算用レジスタであり，`lw`命令によりメモリからデータをロードする．また，`addu`命令は次のイテレーションでアクセスするアドレスを計算する命令であり，`/.fwd`は`$7`の値を次のスレッドへ送信することを表す．オリジナル版では，`lw`命令と`addu`命令の間には命令が15存在するため，計17命令分の実行時間が同期待ちとなる．スケジューリング版では，`lw`命令の直後に`addu`命令を移動したため，これら2命令分の実行時間が同期待ちとなる．最適化版では，レジスタをコピーする`move`命令を追加し，`lw`命令より前に`addu`命令を移動することで，同期待ちは`move`と`addu`命令の実行時間となる．

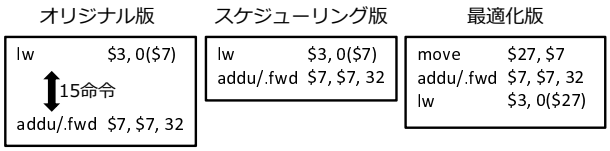

図 2: 各投機スレッドコードにおけるアドレス計算用レジスタの位置の変化

4 実行結果および考察

図3に各スレッドコードの速度向上率を示す．速度向上率は，元のプログラムを逐次実行した際のサイクル数と，マルチスレッド実行でのサイクル数より算出した．スレッドの起動は2クロックで行えるものとし，PALSにおける各ハードウェア間の通信には1クロックかかるものとした．パス予測成功率は，約96%となった．

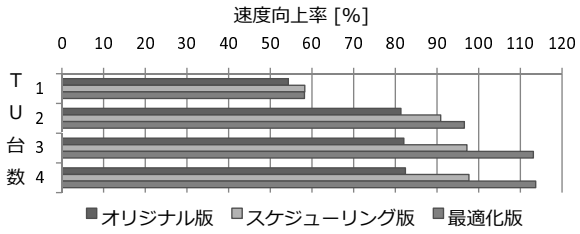

図 3: 各スレッドコードの速度向上率

TU1台の場合は，マルチスレッド実行にかかる処理がすべてオーバヘッドとなるため速度向上率は低くなる．TUを2〜4台にした場合は，オリジナル版ではどれも約80%となっており，台数による効果は確認できない．スケジューリング版では，速度向上率は約90%から約97%となり，台数による効果が現れている．オリジナル版に比べると大幅な速度向上となっており，レジスタ同期待ち時間低減の効果が高いことが分かる．最適化版では，TU3台および4台の場合に顕著な速度向上率を示し，約113%となった．ロード命令に関する最適化の効果が非常に高いことが分かる．

5 おわりに

本稿では，複雑な制御構造を持つループに対する有効なマルチスレッド実行のアプローチとして，2パス限定投機システムPALSによる実行について述べた．そして，SPEC CINT2000ベンチマークにおけるデータ依存を含んだループを，PALSシミュレータ上でマルチスレッド実行し，逐次実行に対してプロセッサ4台で最大約1.1倍の速度向上を達成することができた．

PALSでは，投機的なデータの整合性の保証により他の命令に比べてメモリアクセス命令のコストが高くなる．今後は，整合性を保証した上で低コストなメモリアクセスを実現するメカニズムについて検討する予定である．

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)20500047，同 (C)21500049，同 (C)21500050) および宇都宮大学若手萌芽的研究プロジェクトの援助による．

参考文献

[1] 中島浩: “非数値並列計算の動向と展望”，日本ソフトウェア科学会: コンピュータソフトウェア，Vol.25，No.3，pp.3-43–3-48，2008.

[2] 十鳥弘泰ほか: “2パス限定投機方式を実現するマルチコアプロセッサPALSの提案”，信学技報，Vol.109，No.319 (CPSY2009–46)，pp.19–24，2009.

[3] 福田明宏ほか: “2パス限定投機システムにおけるコードスケジューリング手法とその評価”，情報処理学会 第73回全国大会，講演番号 2H-9，2011年3月 (発表予定)．