2パス限定投機システムによる非数値処理系プログラムの高速化

十鳥弘泰† 大津金光†
横田隆史† 馬場敬信†

1. はじめに

マルチコアプロセッサを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マルチスレッド化による性能向上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，頻繁に実行される2本の実行経路 (パス) を抽出する．2本のうちどちらが実行されるかを予測し，実際の実行より先行して並列実行 (投機実行) することにより，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 パス限定投機方式を実現したマルチコアプロセッサ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，SPEC ベンチマークを利用したシステムの性能評価を行い，本システムの有効性を示す．

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て投機実行する．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，パスの予測は二者択一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．さらに，投機が失敗した場合はレジスタやメモリを投機スレッド実行以前の状態に戻す処理が必要となり，大きなオーバヘッドとなることが考えられる．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．

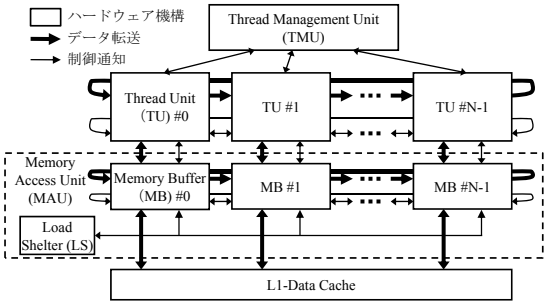


図 1 PALS のハードウェア構成

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 は投機メモリアクセス量が MB 容量を超えたとき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．

† 宇都宮大学大学院工学研究科

ⓒ 2011 Information Processing Society of Japan

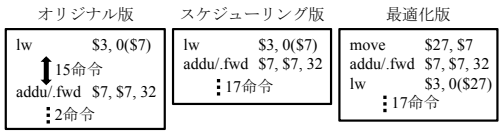


図 2 各スレッドコードにおける依存関係を持つ命令の位置の変化

3. SPEC ベンチマークを用いた性能評価

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，といった条件を満たすループを選択し，PALS をクロックレベルで模擬するシミュレータでプログラムを実行し評価する．

対象は，`181.mcf` の関数 *dual_feasible*() および関数 *primal_bea_mpp*() 中のループとした．

3.1 PALS での実行に適したコードの作成

対象ループでは，ロードアドレス計算用レジスタをイテレーション終端で更新し，直後のイテレーション先頭でのロードに使用する．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．このため，単純なマルチスレッド化を適用したスレッドコード (以下，オリジナル版) では，直前のイテレーションが終わるまでロードを行うことができず，同期待ち時間が大きくなる．

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] を適用する (以下，スケジューリング版)．また，スレッ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．さらに，ループの実行を通して同じ値をロードする命令をループ外に移動し，空き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 (以下，最適化版)．

図 2 に関数 *dual_feasible*() の各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．`$7` は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値を読み出す．また，`addu` 命令は次の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．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．

3.2 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．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．

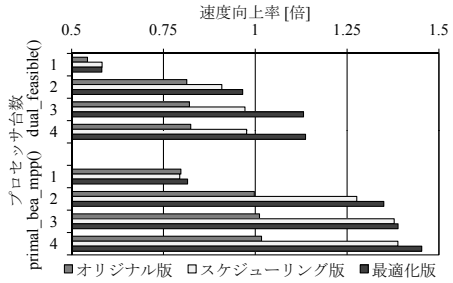


図 3 各スレッドコードの速度向上率

TU2〜4 台の場合，スケジューリング版では速度向上率は最大約 1.38 倍となり，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，最大約 1.45 倍となった．MB を介したメモリアクセスは TU で実行されるその他の命令に比べると非常にコストが高いため，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．

4. 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，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 1.45 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のため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．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の検討が今後の課題である．

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) 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, pp. 19–24 (2009).
3) 福田ほか: 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 回全国大会講演論文集, pp. 1-83–1-84 (2011).

ⓒ 2011 Information Processing Society of Japan