ISS-SP-339

スレッドレベル並列投機実行のための チェックポイントリスタート方式

濱田　貴弘†　　布目　淳††　　平田　博章††　柴山　潔††

† 京都工芸繊維大学大学院　工芸科学研究科　情報工学専攻

†† 京都工芸繊維大学大学院　工芸科学研究科　情報工学部門

1. はじめに

プログラムの実行時間を短縮する手段として，スレッドを処理単位として並列に実行するスレッドレベル並列処理がある．しかし，逐次実行を前提として記述されたプログラムはまだ多く，このようなプログラムからスレッドレベルの並列性を抽出することは困難な状況にある．そこで，本研究では，複数のスレッド間でメモリアクセスについて依存関係が存在しないものと仮定してスレッドを並列実行する投機的マルチスレッディング方式を開発する．

並列実行を阻害するスレッド間の依存関係には，出力依存，逆依存，フロー依存がある．このうち，出力依存と逆依存に関してはメモリリネーミング[1]で除去可能である．本研究では，フロー依存のみに着目する．

2. 研究目的

チェックポイントを用いる場合と用いない場合のスレッド実行の様子を図1に示す．チェックポイントを用いない場合には，投機的実行の失敗時にはそのスレッドの実行結果をすべて破棄し，スレッドの最初から再実行する．チェックポイントを設けてチェックポイントまでの実行結果を使用することで，このようなペナルティを軽減できるが，チェックポイントをどこに設定すべきかを選定する必要がある．本研究では，スレッド間でのフロー依存によるハザードの発生状況を監視し，その情報をもとに動的にチェックポイントを生成する方式を開発する．

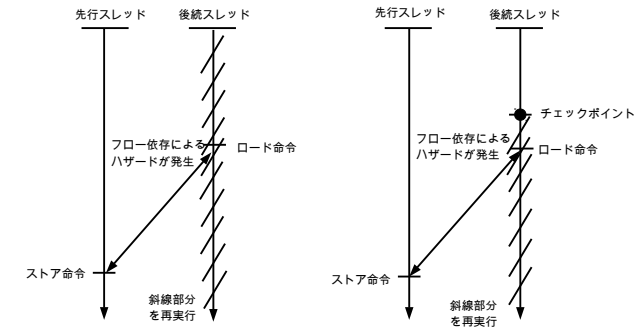


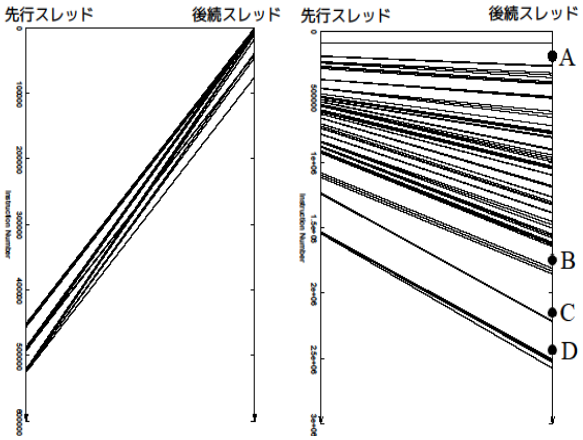
図 1　投機的実行失敗時の処理の違い

3. 予備調査

チェックポイントを設ける場合の効果を調べる．SPEC CPU2006[2]の444.namd と453.povray（データセットは ref）を用いて調査した結果を図2に示す．図2はある先行スレッド（左端）のストア命令と後続スレッド（右端）のロード命令とのフロー依存の関係を線で結んで示したものである．縦軸はスレッド開始時からの実行命令数を表している．

453.povray では，先行スレッドの実行終了付近のストア命令と後続スレッドの実行開始付近のロード命令との間でフロー依存の分布が集中しているので，チェックポイントを設けても並列処理効果は薄いと考えられる．

一方，444.namd では，スレッド全体にフロー依存の関係が分布しているので，例えば，後続スレッドの A～D 点にチェックポイントを設定すれば，スレッド実行の進捗に応じてペナルティを軽減できる．したがって，並列処理の効果も期待できる．



(a) 453.povray　　(b) 444.namd

図 2　フロー依存分布の例

4. まとめ

本研究では，スレッドレベル並列投機実行において，チェックポイントを動的に生成して再実行のペナルティを軽減する方式を開発する．今後，ハードウェアでの実現方式を検討する．

参考文献

[1] 平田博章，藤井崇弘，藤皓平，森田清隆，布目淳，柴山潔，“スレッドレベル並列投機実行のためのメモリリネーミング機構，” 第11回情報科学技術フォーラム講演論文集，vol.1, pp.31-34, Sep. 2012

[2] Standard Performance Evaluation Corporation, “SPEC CPU2006,” http://www.spec.org/cpu2006/.