2パス限定投機システムの提案 － スレッドコード生成処理系 －

福田 明宏 † 十鳥 弘泰 † 佐藤 和史 †† 米田 淳一 †† 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 †† 宇都宮大学大学院工学研究科情報工学専攻

1 はじめに

近年、半導体集積技術が向上したことにより計算機システムにおいて利用可能なハードウェア資源は増大している。しかしその一方でクロック速度の向上や命令レベル並列性の抽出による計算機の速度向上は頭打ちとなりつつあり、スレッドレベル並列性の抽出による速度向上が求められている。

このような背景から本研究室では 2 パス限定投機方式が提案している [1]。2 パス限定投機方式はスレッドレベル並列性を抽出するマルチスレッド実行モデルであり、トレースベースのシミュレーションによってすでにその有効性が示されている。そこで 2 パス限定投機方式を実現する 2 パス限定投機システムの設計を行った。このシステムを実現することにより 2 パス限定投機方式においての詳細な評価が可能となる。

本研究では 2 パス限定投機システムにおいて実行されるスレッドコードを生成する処理系の開発を行う。

2 2 パス限定投機方式

2 パス限定投機方式はプログラムの実行前にプログラム中の各ループ内の実行経路 (パス) のうち実行頻度の高い上位 2 本のパスについて最適化したコード (投機コード) をそれぞれ作成しておき、プログラムの実行時には 2 つのパスのどちらが実行されるかを予測しながら投機的にプログラムを実行する方式である。予測はループの 1 イテレーションごとに行い、予測された投機コードを 1 つのスレッドに割り当て実行を行う。これらの処理を複数のスレッドで並列的かつ投機的に行うことで、高速なマルチスレッド実行を実現する。

この方式では投機的にスレッド生成、実行を行うため予測が誤っていた場合は投機を失敗したスレッドとそのスレッド以降に投機実行しているスレッド全てを破棄する。そして投機を失敗したスレッドにはもう一方の投機コードを割り当て実行し、後続のスレッドでは新たに予測を行い、スレッドを割り当てる。2 回目の投機実行にも失敗した場合は同様にスレッドの破棄をしたのち元のループ構造を保持した逐次コード (非投機コード) を実行する。非投機コード実行中には後続のスレッドは非投機コードの実行終了を待つ。

3 スレッドコードの仕様

2 パス限定投機方式はプログラム実行前にコードに最適化を施すことを前提としている。そのため 2 パス限定投機システムを設計するにあたってコードを生成するシステムは必要不可欠である。この章では 2 パス限定投機システムの実行に必要なコードの仕様について述べる。

2 パス限定投機システムではループごとに実行頻度の最も高いパスに特化したコードと実行頻度が 2 番目に高いパスに特化したコード、投機を 2 回失敗したときに実行する元のループ構造を保持したコードの 3 つが必要となる。

例えば図 1(a) のようなループ構造を持つプログラムがあるとする。このループ構造では A → B → G、A → C → D → F → G、A → C → E → F → G の 3 通りのパスが存在する。このパスのうち A → B → G を最も実行頻度の高いパス (#1 パス)、A → C → E → F → G を 2 番目に実行頻度の高いパス (#2 パス) とする。2 パス限定投機方式ではこの#1 パスと#2 パスそれぞれにに特化したコードに対して投機実行を行う。図 1(b) に#1 パスに特化したコードの構造、図 1(c) に#2 パスに特化したコードの構造を示す。図で示している通りこれらのコードは一本のパスに特化しており、その他のパスを一切含まないコードとなっている。

2 パス限定投機システムではパスを予測して投機的にスレッド割り当て、実行を行うために予測が誤っていることがある。そのため投機実行の成否を判定する必要がある。投機実行の成否は assert 命令によって行う。assert 命令は実行したパスが正しいか否かを判定する命令の総称である。assert 命令が不成立で予測が誤っていることを検出する。assert 命令は投機コードの適切な箇所に挿入する必要がある。

マルチスレッド実行の対象はループのみであるためそれ以外のコードはシングルスレッドで実行される。そのためマルチスレッド実行の開始と終了を宣言する情報をコードに付加しなければならない。

また、スレッドの処理の終了位置は静的に判明するためスレッドの終了を知らせる情報を付加する。

マルチスレッド実行では同時に複数のスレッドが処理を行うためスレッド間通信を考慮する必要がある。投機コードは 1 本のパスに特化しているので静的に後続のスレッドに送るべきデータが決まる。そのためコードにスレッド間通信を行うための情報を付加せねばならない。

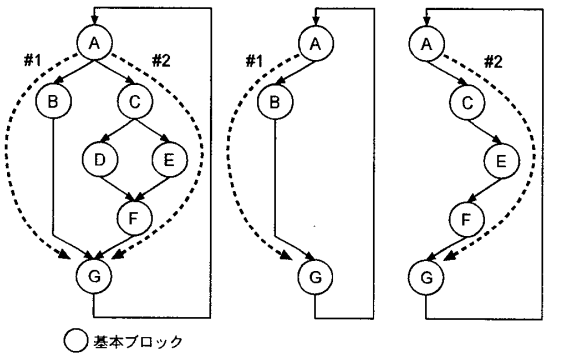

図 1: ループ内パスの例

4 スレッドコード生成処理系

以上のコードの仕様を踏まえた上で 2 パス限定投機システム用の最適化を施すスレッドコード生成処理系

Proposal of Two-Path Limited Speculation System
– Thread Code Generation System –
†Akihiro Fukuda and Hiroyoshi Jutori
††Kazufumi Sato, Junichi Yoneda, Kanemitsu Ootsu, Takashi Yokota and Takanobu Baba
Department of Information Science, Faculty of Engineering, Utsunomiya University (†)
Department of Information Science, Graduate School of Engineering, Utsunomiya University (††)

について検討した。スレッドコード生成処理系が行う処理は以下の 4 点である。

- 投機コード、非投機コードの生成
- assert 命令の挿入
- スレッド制御をサポートする情報をコードに付加
- スレッド間通信をサポートする情報をコードに付加

これらについての詳細な説明は以下の節で行う。

4.1 投機コード、非投機コード

投機コードは条件分岐などパスの実行に必要のない処理を全て削除し、パスに沿った基本ブロックのみを抽出したコードとなる。非投機コードは元のループ構造を保持しているので元のコードをそのまま使用する。

4.2 assert 命令

assert 命令は実行したパスが正しいか否かを判定する命令である。投機実行の成否は assert 命令によって行われるためスレッドコード中に assert 命令を適切な位置に挿入する必要がある。プログラム実行中にパスを決定するのは分岐命令であるから投機コード中の分岐命令を assert 命令に置き換えればよい。スレッドコード生成処理系では元の分岐命令とそのパスが辿る分岐先によって対応する assert 命令に置き換える。

4.3 スレッド制御

2 パス限定投機システムでは投機コードを生成したループに対してのみマルチスレッド実行を行うためコードにマルチスレッド実行開始や終了を告げる情報を付加する必要がある。そのために start2path 命令と stop2path 命令という 2 パス限定投機システム特有の命令を用いる。start2path 命令はマルチスレッド実行を開始する命令であり、stop2path 命令はマルチスレッド実行を終了する命令である。スレッドコード生成処理系では適切な位置にこの命令を記述し、2 パス限定投機システムのマルチスレッド実行を指示する。

また、スレッドの終了を指示するために thread end bit を使用する。2 パス限定投機システムでは拡張が容易という点から PISA[2] をベースとした命令セットを想定している。PISA の命令長である 64bit のうち未使用領域である annote フィールドの 16bit に機能 bit として thread end bit を定義する。スレッドコード生成処理系では命令のオペコードの後ろに/.end と記述することで thread end bit を付加する。図 2 に投機コードの例を示す。図中では sw 命令に thread end bit が付加されている。thread end bit は投機コードの最後の命令に付加する。thread end bit が付加された命令が実行されるとそれを実行したスレッドは終了する。

4.4 スレッド間通信

2 パス限定投機方式では複数のスレッドで並列に処理を行うためスレッド間の通信が頻発する。そのためスレッド間通信がボトルネックになる可能性が考えられ、性能の低下を招く恐れがある。それを極力防ぐためには高速なスレッド間通信が必要不可欠である。そこで 2 パス限定投機システムでは高速なスレッド間通信を実現するため隣り合うスレッド間でレジスタの値の送受信を行うレジスタデータ通信を採用する。

投機コードではパスが限定されているため送受信すべきレジスタを静的に決定できる。そのためスレッドコード生成処理系ではこれらの情報をコードに付加し、レジスタデータ通信をサポートする。

レジスタの送信は forward bit を使用して行う。forward bit は thread end bit と同様に命令長 64bit のうちの未使用領域に定義する。プログラム実行時に forward bit を付加した命令が実行されたらその命令で更新されるレジスタの値を後続のスレッドに送信する。スレッドコード生成処理系では命令のオペコードの後ろに/.fwd と記述することで forward bit を付加する。forward bit を付加するのは投機コード内でレジスタの値が確定する命令である。図 2 では演算命令の addiu 命令と addu 命令に forward bit が付加されている。これらの命令では 1 番目のオペランドのレジスタの値を更新するため、1 番目のオペランドで指定されているレジスタの値を後続のスレッドに送信することになる。

また、プログラムの整合性を保つためにスレッドが値を同期待ちするべきレジスタはそのスレッドが使用するレジスタである。パスごとに使用するレジスタは異なるため先行するスレッドから値を受け取るべきレジスタは投機実行するスレッドのコードが#1 パス、#2 パスのどちらであるかに左右される。そのためにパスごとに同期待ちするべきレジスタ群 (以後、bit mask) を定義しておき、実行時には予測されたパスに応じて bit mask を選択し、スレッドに同期待ちを促す。スレッドコード生成処理系ではパスごとにレジスタの依存解析を行い、#1 パスが選択されたときの bit mask、#2 パスが選択されたときの bit mask をそれぞれ生成する。

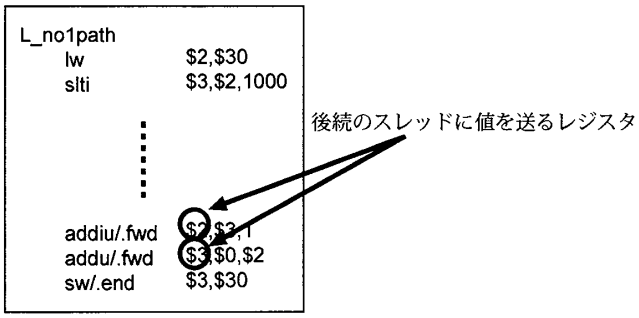

図 2: 投機コードの例

5 おわりに

本稿では 2 パス限定投機方式、スレッドコード生成処理系について述べた。現在、スレッドコード生成処理系ではパスに沿って基本ブロックを忠実に抜き出して投機コードを生成している。パスに特化した投機コードでは条件分岐を含まないため基本ブロックに跨った最適化を施すことが可能であると考えられる。そのため今後の課題としては生成したスレッドコードに最適化を施し、より最適化されたコードを生成することが挙げられる。

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)19500037，同 (C)20500047）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] 横田 隆史, 斎藤 盛幸, 大津 金光, 古川 文人, 馬場 敬信, “2 パス限定投機方式の提案”, 情報処理学会論文誌：コンピューティングシステム, Vol.46, No.SIG 16(AGS-12), pp.1-13, 2005.

[2] Doug Burger, Todd M. Austin, “The SimpleScalar Tool Set, Version 2.0”. University of Wisconsin-Madison Computer Sciences Department Technical Report # 1324, 1997.6