C-001

動的可換性解析によるスレッドレベル並列投機実行方式
Dynamic Commutativity Analysis for a Thread-Level Speculation on a Multiprocessor

藤 皓平‡　布目 淳†　平田 博章†　柴山 潔†
Kouhei Fuji　Atsushi Nunome　Hiroaki Hirata　Kiyoshi Shibayama

1. はじめに

マルチコアプロセッサが商用のマイクロプロセッサとして市販される一方で，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．そこで我々は，逐次実行を想定して記述されたプログラムに対して，ループのイタレーションをスレッドとして投機的に並列実行する方式を開発している[1]．充分な並列性を引き出すためにはまだ解決すべき課題も多いが，本稿では，その 1 つとして，可換処理に着目した投機実行方式を提案する．

2. 可換処理

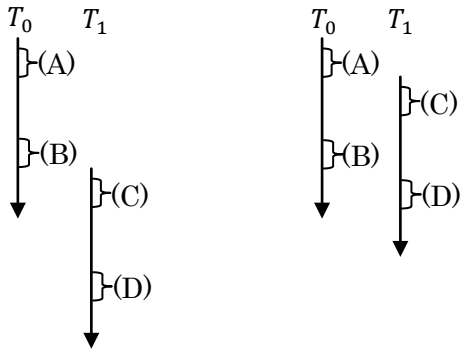
2.1 可換処理の定義

プログラム中において，ある処理を，その実行順序を入れ替えて実行しても，プログラムの整合性が損なわれないという性質を**可換性**[2]と呼び，本稿では，この性質を持つ処理のことを**可換処理**と呼ぶことにする．

文献[3]では，乱数生成処理における可換性に着目して，プログラムの並列性を向上させる方式を提案している．図 1 に，2 個のスレッドT_0，T_1を並列に実行する様子を示す．論理的な順序としてT_0の後にT_1を実行するものとし，図 1 中，(A)～(D)の部分で乱数生成を行うものと仮定する．乱数生成処理では乱数の種となる変数から乱数を生成し，それを新たな種とすることで順に乱数を生成するので，プログラム上では，(A)～(D)の処理は必ずこの順序で実行されなければならない．したがって，T_0の(B)が完了するまで，T_1における(C)の実行を待たせなければならない．しかし，本来の乱数としての使用目的を考えると，乱数を使用する順番，つまり，乱数を生成する順番は入れ替えても差し支えはない．この乱数生成処理の可換性を考慮すると，(A)～(D)のそれぞれをアトミックに実行する必要はあるものの，図 1(b)のように(A)→(C)→(B)→(D)の順に実行することによって，T_0とT_1の並列実行の機会が増す．しかし，このような乱数生成の可換性は，処理の目的や意味と深く関わるため，機械的に判断することは難しい．

2.2 可換処理方式の概要

本稿では，以下のような可換処理を対象とする．一般のプログラムにおいて，そこに記述されたある条件に基づいて，その事象の発生回数などをカウントする場合が比較的多くみられる．その回数を格納する変数を**カウント変数**と呼ぶことにすると，プログラム上では，カウント変数に定数を加えてその値を更新するように記述される．カウント変数に対するアクセスの種類（読み出し/書き込み）とその順序に従えば，先行するスレッドにおいて更新されたカウント変数の値を後続するスレッドで用いることになるので，並列実行の機会を大きく制限することになる．しかし，カウント変数が自身の値の更新以外の目的でアクセスされない場合，カウント変数に対する加算処理はどのような順序で行っても，プログラム全体の計算には支障がない．



(a) 可換性を利用しない場合　(b) 可換性を利用する場合

図 1　スレッドの並列実行の様子

ただし，投機実行中のスレッドがこのような可換処理を実行する場合には，投機失敗時の回復機能を用意しておかなければならない．その実現方式として，(i) 変更前のカウント変数の値を保存しておき，投機失敗時にその保存した値に戻す，(ii) 可換処理の内容を記憶しておき，投機失敗時に逆演算（例えば加算に対して減算）により元の値に戻す，(iii) 可換処理の内容を記憶するだけで実行は行わず，投機スレッドがコミットする時点で実行する，などが考えられる．本方式では(iii)の方式を採用する．

以上をまとめると，本方式では，スレッドの投機実行中に可換処理を検出し，その内容を**遅延処理リスト**と呼ぶ専用のデータ構造に記憶する．検出した可換処理そのものはその時点で実行せず，プログラムに記述されたそれ以降の処理を続行する．投機スレッドがコミットする時点で，遅延処理リスト中の可換処理を実行する．

2.3 可換処理の検出条件

マシン命令レベルでカウント変数に関する可換処理が検出できるための条件を以下にまとめる．プログラム実行中に，アクセスするメモリアドレスごとに，以下の条件を動的に検査することによって，可換性解析を行う．

1. メモリ上のあるデータに対して，以下の順序で命令を実行する．
   (ア) ロード命令
   (イ) (ア)でロードしたデータと定数をソースオペランドとする固定小数点加算/減算命令
   (ウ) (ア)と同一のアドレスに対するストア命令
2. 1.のデータに対して，1.以外でのロード命令およびストア命令は実行しない．

†京都工芸繊維大学大学院工芸科学研究科情報工学部門
‡京都工芸繊維大学大学院工芸科学研究科情報工学専攻
Dept. of Information Science, Kyoto Institute of Technology

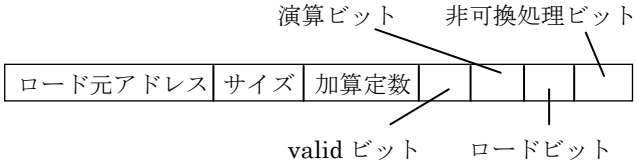


図２　レジスタに付加する情報

3. 可換性解析とその実行制御

3.1 ハードウェア機構

まず，プロセッサ内において，整数レジスタごとに，図 2 に示すように，以下の情報を付加・記憶する．

(i) レジスタにデータをロードした際のメモリアドレス (**ロード元アドレス**)
(ii) メモリ上でのそのデータのサイズ(1/2/4/8 バイト)
(iii) 更新処理の際に加算/減算に使用する定数値（**加算定数**）
(iv) (iii)の加算定数の有効性を示す valid ビット
(v) 加算/減算の別を示す**演算ビット**
(vi) レジスタに値をロードしたことを示す**ロードビット**
(vii) 可換処理対象のデータでないことを示す**非可換処理ビット**

また，1 次キャッシュにおいて，変数ごとにその値をロードしたことを示す**ロードビット**と，可換処理対象の変数でないことを示す**非可換処理ビット**，を設ける．

さらに，プロセッサと 1 次キャッシュとの間のバスに，遅延処理リストを格納するためのメモリを設けて接続する．この**遅延処理リスト用メモリ**には，カウント変数のメモリアドレスとそのデータサイズ，加算/減算の別を示す演算ビット，加算/減算定数値，を記憶する．

3.2 可換処理の検出

2.3 で示した可換処理の検出条件を用いて，以下のマシン命令において可換処理の検出を行う．

(1)　ロード命令

1 次キャッシュにミスヒットする場合，スレッドがそのアドレスへアクセスするのは初めてであり，したがって，そのロードは可換処理に含まれるロード命令であると仮定する．1 次キャッシュ内のそのデータに対するロードビットをセットし，非可換処理ビットをリセットする．また，ロード先のレジスタに関しては，ロード元アドレスとサイズを記憶し，ロードビットおよび非可換処理ビットについては 1 次キャッシュのそれぞれの値をコピーする．

1 次キャッシュにヒットする場合，そのロードは可換処理とは無関係のロードであると判断し，1 次キャッシュ内のロード元アドレスに対応する非可換処理ビットをセットする．同様に，ロード先のレジスタの非可換処理ビットもセットする．

(2)　固定小数点加算/減算命令

まず，ソースオペランドを調べて，一方がレジスタで，もう一方が即値のとき，可換処理のための演算命令である可能性がある．この場合，ソースオペランドのレジスタのロードビットがセットされていて，かつ，非可換処理ビットがリセットされているときには，ソースオペランドに指定されたレジスタのロード元アドレスとロードビット，および非可換処理ビットの値をデスティネーションオペランドに指定されたレジスタのそれぞれにコピーし，また，あわせて加算定数も保存する．ただし，加算定数が既に保存されている場合には可換処理ではないと判断し，非可換処理ビットをセットする．

上記以外の場合は，ソースオペランドとデスティネーションオペランドに指定されたレジスタの非可換処理ビットをそれぞれセットする．

(3)　ストア命令

ストアするデータを格納するレジスタのロードビットがセットされていて，かつ，非可換処理ビットがリセットされている場合，そのレジスタに対応するロード元アドレスとストア先アドレスを比較する．

一致する場合は可換処理であると判断し，1 次キャッシュへの書き込みは行わず，その代わりに，ストア先アドレスと加算定数および演算ビットの値を遅延処理リスト用メモリに書き込む．また，1 次キャッシュ内のストア先アドレスのラインを無効化する．

一致しない場合は，1 次キャッシュへストアを行い，また，ストアするデータを格納するレジスタのロード元アドレスに対する 1 次キャッシュ内の非可換処理ビットをセットする．

ストアするデータを格納するレジスタのロードビットがリセットされている，または，非可換処理ビットがセットされている場合は，可換処理に含まれるストア命令ではないため，通常のストア操作を行うとともに，1 次キャッシュ内のストア先アドレスに対する非可換処理ビットをセットする．

(4)　move 命令

move 命令によってレジスタ間で値をコピーする場合には，コピー元とコピー先の両方のレジスタの非可換処理ビットをセットする．

3.3 可換処理の遅延実行

投機スレッドをコミットする時点で，プロセッサは，遅延処理リスト中の可換処理を実行するための組み込みコードを呼び出す．そのコードが，実行すべき可換処理の内容を遅延処理リスト用メモリから順次読み出し，カウント変数の更新を行う．

4. むすび

本稿では，プログラムのスレッドレベルの並列性抽出の可能性を向上させることを目的として，動的に可換性解析を行う並列投機実行方式を提案した．今後は，本方式のより詳細な実装方式の検討と性能評価を行う．

謝辞

本研究の一部は日本学術振興会科学研究費補助金（基盤研究（C）22500046）の補助による．

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 10 回情報科学技術フォーラム論文集(掲載予定), 2012.
[2] M. C. Rinard and P. C. Diniz, “Commutativity Analysis: A New Analysis Framework for Parallelizing Compilers,” Proceedings of the Conference on Programming Language Design and Implementation, pp.54-67, 1996.
[3] M. J. Bridges, N. Vachharajani, Y. Zhang, T. Jablin, and D. I. August, “Revisiting the Sequential Programming Model for Multi-Core,” Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture, pp.69-84, 2007.