Cascading Stall の抑制によるハードウェアトランザクショナルメモリの高速化

三宅 翔*，間下 恵介，山田 遼平，津邑 公暁（名古屋工業大学）

Contorolling Cascading Stall on Hardware Transactional Memory

Sho Miyake, Keisuke Mashita, Ryohei Yamada, Tomoaki Tsumura (Nagoya Institute of Technology)

1. はじめに

マルチコア環境における共有リソースへのアクセス調停にはロックが広く用いられているが，並列性の低下やデッドロックの発生などの問題がある．そこで，ロックを用いない並行性制御機構としてトランザクショナルメモリ（TM）[(1)]が提案されている．本稿では，TM で発生する問題である **Cascading Stall** を解消し，並列度を向上させるために，各スレッドの依存関係を「深さ」として管理し，その深さが閾値を超えた場合に特定のトランザクションをアボートさせる手法を提案する．

2. Hardware Transactional Memory

TM は，データベースにおけるトランザクション処理をメモリアクセスに適用した手法であり，従来ロックで保護されていた処理範囲をトランザクションとして投機的に実行する．なお，TM におけるトランザクションの投機実行では，共有リソースに対する更新の際に更新前の値を別領域に保持する必要がある（**バージョン管理**）．また，トランザクションを実行するスレッド間において競合が発生していないかを常に検査する必要がある（**競合検出**）．TM のハードウェア実装である Hardware Transactional Memory （HTM）では，バージョン管理および競合検出の処理を高速に行うことができる．この HTM ではトランザクション間で競合が発生した場合に，一方のトランザクションをストールさせることで競合を回避している．

3. Cascading Stall とその解決手法

HTM では，複数のスレッドがそれぞれトランザクションを実行した際に，あるスレッドがすでにストール状態となっているスレッドと競合し連鎖的にストールしてしまう，Cascading Stall と呼ばれる問題が発生する．Cascading Stall が発生すると，ストールの連鎖により並列度が低下してしまう．本稿では，この Cascading Stall を効率的に解消する手法を提案する．この手法では Cascading Stall において，ストールの連鎖の段数を深さと呼ぶ．各スレッドは，自身の深さを管理し，実行トランザクションにおいて Stall が発生および解消した際に自身の深さを更新する．そしてその深さが一定の閾値に達すると，Cascading Stall が発生したことを検出し，関係するトランザクションをアボートすることで Cascading Stall を解消する．

Cascading Stall を検出し，特定のトランザクションをアボートさせるため，各スレッドが自身の深さの値を記憶するための深さカウンタを，各コアに追加する必要がある．また，深さの値が閾値に達した場合に特定のトランザクションをアボートさせるため，競合相手のスレッドが実行されているコアの ID を記憶しておく必要がある．そこで，そのための Core Bitbap も各コアに追加する．なお，これらの追加ハードウェアは，N コア N スレッドで実行する場合，1 コアあたり深さカウンタに 2bit，Core Bitmap に $N-1$bit 必要となる．プロセッサ全体での容量としては，16 スレッドを実行可能な 16 コア構成を仮定すると，深さカウンタの 4bytes と Core Bitmap の 30bytes を合わせて 34bytes となり，プロセッサ全体の容量と比べてごく小容量で実現することが可能である．

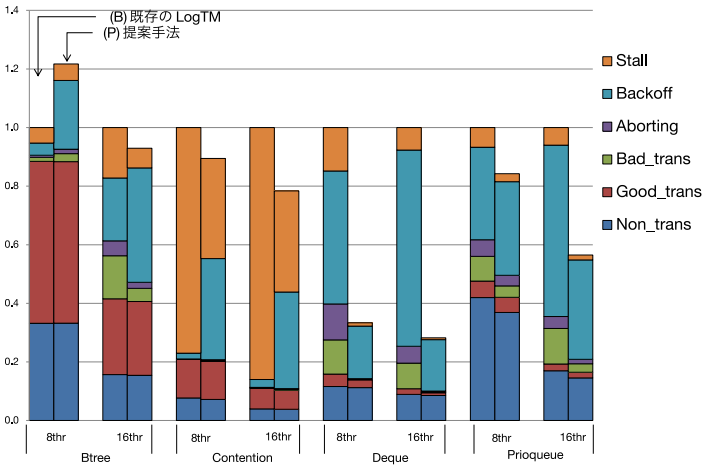

Fig.1　Execution cycles ratio

4. 評価

提案手法をシミュレータ上に実装し，評価を行った．シミュレータには HTM の研究で広く用いられている Simics 3.0.31 と GEMS 2.1.1 の組合せを用いた．プロセッサ構成は 32 コアの SPARC V9 とし，OS は Solaris 10 とした．提案手法の評価には GEMS microbench から 4 個のプログラムを使用し，8 および 16 スレッドで実行した際の既存手法と提案手法の実行サイクル数を比較した．また，Cascading Stall 発生を検出する深さの閾値は 2 とした．その結果を Fig.1 に示す．評価の結果，本提案手法により Cascading Stall が解消され，既存モデルに対して平均で 36.0%，最大で 71.8% の性能向上を得ることができた．

文　献

(1) Maurice Herlihy and J. Eliot B. Moss: Transactional Memory: Architectural Support for Lock-Free Data Structures, Proc. 20th Annual Int'l Symp. on Computer Architecture , pp.289–300 (1993)