

## 競合相手数に応じたハードウェアトランザクショナルメモリの実行制御

間下 恵介\*, 三宅 翔, 山田 遼平, 津邑 公暁 (名古屋工業大学)

A Transaction Scheduling based on Opponent Count for Hardware Transactional Memory

Keisuke Mashita, Sho Miyake, 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) <sup>(1)</sup>が提案されている。本稿では、実行されている競合相手トランザクションの数に応じて、トランザクションの実行を制御することで競合の発生を抑制し、性能向上を図る。

## 2. ハードウェアトランザクショナルメモリ

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。

## 3. 提案

HTM では、トランザクションは実行されるたびにほぼ同一の命令列を処理することで、同じアドレスにアクセスする機会が多い。そのため、一度競合したトランザクション同士が再度並列に実行される場合、再び競合する可能性が高い。また、同時に実行されるトランザクションが多いほど競合の発生率も高くなる。そこで本稿では、スレッドがトランザクションの実行を開始する際に、他のスレッドで実行中の競合相手トランザクションの数が多く、競合が発生する可能性が高い場合に、トランザクションの実行を待機することで、競合を回避する手法を提案する。

## 4. 実装

提案手法を実現するために、一度競合した相手トランザクションの ID を記憶するテーブルを各コアに追加する。スレッドは競合時に自身の実行するトランザクションの ID と競合相手の ID をテーブルに登録する。そして、スレッドがトランザクションの実行を開始する際、他の全てのスレッドで実行中のトランザクション ID を取得し、その中に自身が実行しようとするトランザクションの競合相手の ID があるかを確認する。もし競合相

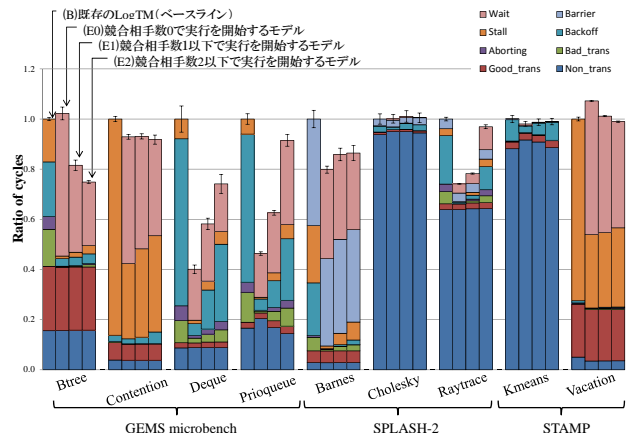


Fig.1 Evaluation Result

手の ID が記憶されていた場合は、競合相手スレッドの数を追加したカウンタに登録し、その値が一定の閾値以上だった場合実行を待機する。また競合相手スレッドは、追加したテーブルに、待機させているスレッドの ID を記憶する。そして自身がコミットすると、待機させているスレッドに対してメッセージを送信する。待機中のスレッドはこれを受信すると、カウンタの値をデクリメントし、値が閾値を下回ると実行を開始する。なお本稿では、閾値を 0 から 2 と変化させて評価を行った。

## 5. 評価

提案手法の評価には GEMS microbench, SPLASH-2 および STAMP ベンチマークを使用し、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。結果を Fig.1 に示す。評価の結果、本提案手法により、多くのプログラムにおいて競合の発生を抑制したことで平均 17.2%、最大 60.2% のサイクル数を削減し、性能が向上した。また、提案手法に必要なハードウェアコストも約 304Bytes と小容量であった。

## 6. おわりに

実行中の競合相手トランザクションの数に応じて実行を待機することで競合を抑制する手法を提案した。今後の方針として、無駄な待機処理の削減と、より適切な競合の検出方法の考案などが挙げられる。

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)