部分ロールバック後の競合抑制によるトランザクショナルメモリの高速化

間下 恵介*，廣田 杏珠，津邑 公暁（名古屋工業大学）

An Efficient Partial Rollback for Transactional Memory

Keisuke Mashita, Anju Hirota, Tomoaki Tsumura (Nagoya Institute of Technology)

1. トランザクショナルメモリ

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．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定義し，トランザクションを投機的に実行することで高い並列性を実現する．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 の一実装である Hardware Transactional Memory（HTM）では，バージョン管理および競合検出の処理をハードウェアによって実現するため，高速に行うことができる．

2. 提案

HTM では，投機実行に失敗しトランザクションを最初から再実行する場合，ログの書き戻しなどの再実行ペナルティが発生する．この再実行ペナルティを削減する手法として，競合を引き起こすアクセスの直前にリスタートポイントを設定し，投機実行が失敗した際に，このリスタートポイントからトランザクションを再実行する，部分ロールバックが提案されている．しかし部分ロールバックでは，競合を引き起こすアクセスの直前から再実行することにより，再競合が発生しやすくなるという問題がある．そこで本研究では，トランザクションがアボートされた際，競合相手のトランザクションのコミットまで待機した後に再実行することで，部分ロールバック後の再競合を抑制する手法を提案する．

3. 実装

部分ロールバック，および提案手法を実現するために，競合アドレス，および自身が待機させているスレッドの ID を記憶するテーブルを各コアに追加する．スレッドはトランザクションの実行中にアボートが発生すると，リスタートポイントの設定のために競合アドレスを追加したテーブルに記憶する．その後，スレッドが再度同じトランザクションを実行し，記憶した競合アドレスに再度アクセスすると，そのアクセスの直前にリスタートポイントを設定する．さらに実行が進み，トランザクションがアボートされる場合，競合相手スレッドのコミットまで再実行せずに待機する．これに対し，競合相手スレッドは待機させているスレッドの ID を記憶する．その後，競合相手スレッドの実行が進みトランザクションをコミットする際，待機させているスレッドに対して実行を許可する Wakeup メッセージを送信する．これを受信したスレッドはリスタートポイントからトランザクションを再実行することで部分ロールバックを実現しつつ，再競合を回避することが出来る．

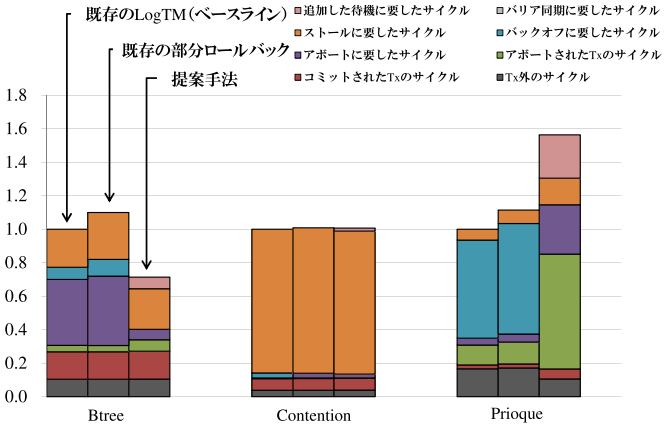

Fig.1　Evaluation Result

4. 評価

提案手法の評価には GEMS microbench を使用し，16 スレッドで実行した際の既存手法，既存の部分ロールバック，および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，Btree において競合の発生を抑制したことで 28.6 %のサイクル数を削減し，性能向上を確認した．

5. おわりに

競合相手のトランザクションがコミットされるまで再実行を待機することで，部分ロールバック後の再競合を抑制する手法を提案した．今後の方針として，並列度を損なわない待機時間の設定などが挙げられる．

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)