最適なロールバック・ポイントを選択するネスティッド・トランザクショナル・メモリの評価

伊藤悠二† 塩谷亮太† 五島正裕† 坂井修一†
†東京大学大学院情報理工学系研究科

1 はじめに

近年，広く普及しているマルチコア・プロセッサを最大限利用するために，並列プログラミングの重要性が高まっている．共有メモリ型の並列プログラミングにおいて，ロックを用いない同期通信機構として，トランザクショナル・メモリ (Transactional Memory) と呼ばれる手法が提案されている [1, 2, 3, 4].

トランザクション (transaction) とは，一つのスレッドで実行される一連の処理をまとめたものである．トランザクションは，他スレッドから見れば，不可分に実行されているかのように実行される．この性質をアトミシティ (atomicity) と呼ぶ．

トランザクショナル・メモリの多くの手法では，トランザクションを投機実行する．しかし，他のスレッドでのメモリ・アクセスと競合して，トランザクションを不可分に実行できない可能性がある．この時，トランザクションを停止してそれまでの処理を破棄するアボート (abort) を行い，再実行する．一方，処理をすべて終了したトランザクションは，状態を確定するコミット (commit) を行う．

トランザクショナル・メモリの問題点の一つとして，ネスティッド・トランザクション (nested transaction) の扱い方がある．ネスティッド・トランザクションとは，内側にトランザクションがネストされたトランザクションである．

我々は，トランザクションがロールバックする際に，再実行される処理が最小限になる最適なロールバック・ポイントを選択するネスティッド・トランザクショナル・メモリを提案した [4]．本稿では，この提案手法を説明する．

Evaluation of Nested Transactional Memory Selecting the Optimal Rollback point
Yuji Ito†, Ryota Shioya†, Masahiro Goshima† and Syuichi Sakai†
†Graduate School of Information Science and Technology, The University of Tokyo

2 最適なロールバック・ポイントの選択

2.1 概要

一般に，トランザクション中のメモリ・アクセスが他スレッドと競合した時，トランザクション途中のある処理以降から再実行してもトランザクションのアトミシティを保てる場合がある [4]．このとき，トランザクションの途中にロールバックする部分ロールバック (partial rollback) を行うことができれば，再実行時のペナルティを緩和することができる．

本手法では，これに着目し，ネスティッド・トランザクションの深さと関係なく部分ロールバックを行う．競合する最も古いメモリ・アクセスとその直前の開始点を検索することで，トランザクション開始点の中から最適なロールバック・ポイントを選択する手法である．

トランザクション実行時には，検索のために，競合する最も古いメモリ・アクセスになり得るすべてのメモリ・アクセスの履歴と，開始点の履歴を実行順に取っておく．ここでの履歴とは，いつ各メモリ・アクセスや開始点があったかの記録のことである．これらの履歴は，LogTM[2, 3] と同様に，ログと呼ばれる領域に記録し，投機的なメモリ・アクセスはキャッシュに対して行う．

競合検出時のロールバックは以下のように行う．

1. ログを用いて，競合する最も古いメモリ・アクセスを検索する．
2. 競合する最も古いメモリ・アクセスの直前の開始点を最適なロールバック・ポイントとする．
3. ログ内のライトの履歴を用いて，新しい履歴からロールバック・ポイントまで順に，そのライト以前の値に戻してロールバックする．
4. ロールバック・ポイントから再実行する．

図1に履歴の様子を示す．同図では，各開始点とメモリ・アクセスの履歴を上から下に新しいものとなるように記録している．例えば，履歴の上から2行目は，

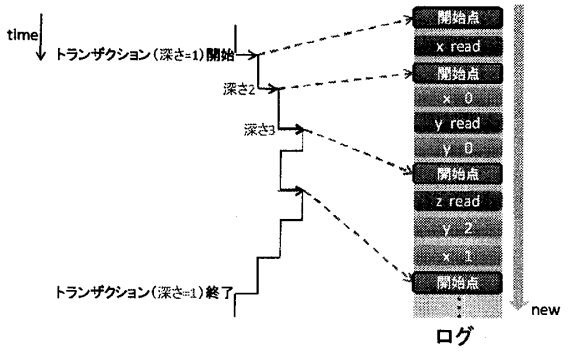

図 1: 履歴の取り方

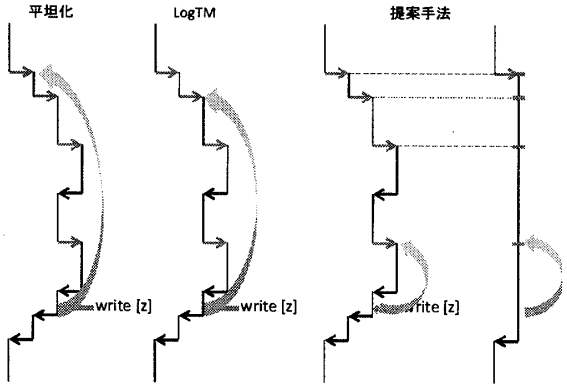

図 2: 最適なロールバック

x がリードされたことを示し，上から 4 行目は，x がライトされ，そのライト以前の値が 0 であったことを示している．同図で，他スレッドによる x のリードがあった場合，古い順に，ここでは上から履歴を調べれば，履歴の上から 4 行目の x へのライトが競合する最も古いメモリ・アクセスであるとわかる．このことから，競合する最も古いメモリ・アクセスの直前の開始点である深さ 2 の開始点が最適なロールバック・ポイントとして選択される．

2.2 既存の手法との比較

既存の手法と本手法のロールバックの様子を図 2 に示す．同図は，深さ 2 のトランザクションの途中で他スレッドと競合し，ロールバックする際の様子である．

平坦化 平坦化では，内側のトランザクションを外側のトランザクションの一部として扱う．内側のトランザクションに競合があると，図 2 の左図のように，外側のトランザクションの処理までが破棄されてしまい，実行効率が低下する．

LogTM LogTM[3] では，必ずしも最適な部分ロールバックができない．深さに着目してメモリ・アクセスの履歴を管理するため，同じ深さのトランザクションを区別できない．このため，内側のトランザクションが終了すると，以降は外側の一部として扱われる．従って，図 2 の中央図のように，終了していないトランザクションの開始点にしか部分ロールバックできない．

3 まとめ

本稿では，最適なロールバック・ポイントを選択する手法について説明した．本手法では，トランザクション実行時に必要な履歴を取っておくことで，競合検出時にそれらを検索し，最適なロールバック・ポイントを選択する．あらゆるネスティッド・トランザクションにおいて，競合とは関係ない処理を繰り返すことを最小限にして，再実行時のペナルティを緩和することができる．

参考文献

[1] M. Herlihy and J. E. B. Moss. Transactional Memory: Architectural Support for Lock-Free Data Structures. In *Proceedings of the 20th Annual International Symposium on Computer Architecture* (1993).

[2] Kevin E. Moore, Jayaram Bobba, Michelle J. Moravan, Mark D. Hill and David A. Wood. LogTM: Log-based Transactional Memory. In *Proceedings of the Twelfth IEEE Symposium on High-Performance Computer Architecture* (2006).

[3] Michelle J. Moravan, Jayaram Bobba, Kevin E. Moore, Luke Yen, Mark D. Hill, Ben Liblit, Michael M. Swift and David A. Wood. Supporting Nested Transactional Memory in LogTM. In *Proceedings of the 12th international conference on Architectural Support for Programming Languages and Operating Systems* (2006).

[4] 伊藤 悠二，塩谷 亮太，五島 正裕，坂井 修一. 最適なロールバック・ポイントを選択するネスティッド・トランザクショナル・メモリ. 情報処理学会研究報告 *2009-ARC-184* (2009)