ISS-SP-338

トランザクショナルメモリを実現するスヌープキャッシュプロトコル

一井　世界†　　布目　淳††　　平田　博章††　　柴山　潔††
†京都工芸繊維大学大学院　工芸科学研究科　情報工学専攻
††京都工芸繊維大学大学院　工芸科学研究科　情報工学部門

1. はじめに

共有メモリ型のマルチプロセッサが広く普及し，その恩恵を得るためには並列プログラミングが必要である．これを支援するための同期排他制御機構として，トランザクショナルメモリ[1]（Transactional Memory；以下，TM とする）が注目されている．これまでにいくつかの方法が提案されているが，本研究では、メモリアクセス量の多いトランザクション（不可分に操作すべき一連のメモリアクセスの系列）を実行可能で，かつ，それを低オーバヘッドで実現するハードウェア TM（Hardware TM；以下，HTM とする）を新たに提案する．

2. トランザクショナルメモリ（TM）

TM では，トランザクションを投機的に実行し，その終了時点で，実行結果をメモリに反映させる（コミット）．しかし，実行中に他のトランザクションとのアクセス競合を検出すれば，その時点で，競合を解決するためにトランザクションの実行を中止し，それまでの実行結果を破棄する（アボート）．競合の検出と解決のポリシーには，以下の2種類がある．

・**楽観的競合検出**：コミット時に競合検出を行う．
・**悲観的競合検出**：メモリアクセス時に競合検出を行う．

楽観的競合検出を用いると，コミット時に，更新したすべてのデータについて競合検出を行うための時間を要する．一方，悲観的競合検出を用いると，アプリケーションによってはアボートが頻発して性能低下を招く．

また，トランザクションの実行中は，それをアボートする場合に備えてデータの更新前の値と更新後の値の2種のバージョンに分けて管理する．バージョン管理の方法は，以下の2種類に分けられる．

・**Eager Versioning**：メモリの値を書き換え，更新前の値を別領域に保存する．
・**Lazy Versioning**：メモリの値を書き換えず，更新後の値を別領域に保存する．

Lazy Versioning を用いるとコミット時に，また，Eager Versioning を用いるとアボート時に，それぞれ別領域に保存した値を用いてメモリを更新する必要があり，オーバヘッドが生じる．

TM では，上記のポリシーと手法を組み合わせてトランザクションを不可分に実行するが，どの組み合わせにおいても，コミット時とアボート時のいずれかに時間的なオーバヘッドが生じる．

また，多くの HTM では，キャッシュを利用してデータのバージョン管理を行っている．トランザクション中に更新したデータがキャッシュからあふれる場合，例えば，LTM[2]では，1次キャッシュのコントローラがメモリ上の拡張領域にバックアップを行う．この領域に追い出したデータも競合検出の対象としなければならないので，ハードウェア機構の複雑化を伴うとともに，OS やコンパイラにも大きな変更を要する．

3. TM 用スヌープキャッシュの提案

本研究では，プロセッサごとにプライベートなデータキャッシュ（1次キャッシュ）を設け，それらが共有バスを介して2次キャッシュを共有するマルチプロセッサシステムを想定する．トランザクション中に更新したデータを各1次キャッシュに格納し，バージョン管理については Lazy Versioning を用いる．従来の方式とは異なり，競合の検出をメモリアクセス時に，また，解決をコミット時に分割して行う．さらに，コミットしたデータに対しては，実際にアクセスされるまで，実行結果の反映を遅らせる．これにより，コミットとアボートの両方の処理を，オーバヘッドなく高速に行う．

また、1次キャッシュからあふれたトランザクション中の更新データを管理するための専用ユニットを，共有バス上に設ける．このユニットが，必要に応じてあたかも2次キャッシュまたは（他のプロセッサの）1次キャッシュとして動作することにより，1次キャッシュの容量を仮想的に拡張する．追い出されたデータに対する競合検出もこのユニットが（1次キャッシュとして）行うことで，OS やコンパイラに対する変更を最小限に留めながら，メモリアクセス量の多いトランザクションに対応する．

5. むすび

本稿では，低オーバヘッドの HTM システムの概要について述べた．今後は，スヌープキャッシュプロトコルや，データ管理ユニットの詳細設計と性能評価を行う．

参考文献

[1] M. Herlihy, J. Eliot, B. Moss, ”Transactional Memory: Architectural Support for Lock-Free Data Structures,” Proceedings of the 20th Annual International Symposium on Computer Architecture, pp.289-300 (1993).
[2] C. S. Ananian, K. Asancovic, B. C. Kuszmaul, C. E. Leiserson, S. Lie, “Unbounded Transactional Memory,” Proceedings of the 11th International Symposium on High-Performance Computer Architecture, pp. 316-327 (2005).