Advanced Contention Management for Dynamic Software Transactional Memory
(W. N. Scherer III et al., 2005)

Presented by Yi Huang
Fall 2008

A paper presentation for CSE891, *Formal Methods in Software Development: Reliable Computing with Threads*
Overview

Non-blocking synchronization
• Motivation
• Progress guarantees

Contention management
• Goal
• Requirement
• Protocols
Lock-based synchronization

Problems

- Deadlock
- Priority inversion
- Convoying
Non-blocking synchronization

Goal
• To prevent the problems in lock-based synchronization

Condition
• The suspension of one or more threads will NOT stop the potential progress of the remaining threads

Feature
• Progress guarantee
Non-blocking progress guarantees

- Obstruction-freedom
- Lock-freedom
- Wait-freedom
Wait-freedom

Condition
• Every thread can eventually complete

Guarantee
• Per-thread progress
• No livelock
• No starvation
Lock-freedom

Condition
- Some threads can eventually complete

Guarantee
- System-wide progress
- No livelock
- No starvation
Obstruction-freedom

Condition
- Every isolated thread can eventually complete

Guarantee
- Conditional progress
- No livelock
- No starvation
Contention management

Goal

• “Isolate” threads for obstruction-free algorithms
  (obstruction-freedom + contention management = wait-freedom)

Requirement

• Any partially-completed operation can be aborted and
  changes rolled back
Contention manager

Definition
• Executable encapsulation of a contention management protocol

Obligation
• Assists a transaction to decide whether to
  • Back off and retry
  • Abort itself and restart
  • Abort the enemy and continue
Contestion management protocols

Basic
• Polite, Karma, Eruption, Kindergarten, and Timestamp

Advanced
• PublishedTimestamp and Polka
Polite

Requires

- The number of attempts a transaction opens each object

Backs off

- Variable time exponential to the number of attempts
- At most $M$ times

Aborts the consulting transaction

- Never

Aborts the enemy

- After $M$ backoffs
Karma

Requires

- The number of attempts a transaction opens each object
- The number of objects a transaction has opened (a.k.a. the transaction’s priority)

Backs off

- Constant time
- Theoretically unlimited number of times

Aborts the consulting transaction

- Never

Aborts the enemy

- When number of conflicts exceeds priority difference
Eruption

Similar to Karma, except that a transaction’s priority propagates to the enemy.
Kindergarten

Requires

- A *hit list* of enemies that a transaction has aborted previously

Backs off

- Constant time
- Fixed number of times

Aborts the consulting transaction

- After all backoffs

Aborts the enemy

- If the enemy exists in the hit list
Timestamp & PublishedTimestamp

Requires
- The timestamp recording when a transaction begins
- A mechanism to detect defunct transactions

Backs off
- Depending on the mechanism
Aborts the consulting transaction
- Never

Aborts the enemy
- If the enemy has a newer timestamp or appears defunct
Detecting defunct transactions

Timestamp
  • Defunct flag

PublishedTimestamp
  • Recency value + inactivity threshold
Polka

Requires
- The same with Polite and Karma

Backs off
- Variable time exponential to the number of attempts
- Number of times equal to priority difference

Aborts the consulting transaction
- Never

Aborts the enemy
- Like Karma
- When the transaction writes and the enemy reads
Prioritizing contention management protocols

Goal
- To control each thread’s cumulative throughput

Approach
- Assign each thread a *base priority* (BP)
- Tune protocols using BP, such that each thread’s cumulative throughput is proportional to its BP

Example: prioritized Karma
- Increase a transaction’s priority by BP, instead of one
Summary

Non-blocking synchronization prevents some problems of lock-based synchronization

Progress guarantees of non-blocking algorithms depend on non-blocking conditions

Obstruction-freedom and contention management separate the progress concern from the correctness concern

Polka gives top or near-top performance across a wide variety of benchmarks (not shown in previous slides)