Software Transactional Memory and Conditional Critical Regions

Word-Based Systems
Introduction: Motivation

- Language Support for Lightweight Transactions (Harris & Fraser)
- Implement a declarative style of concurrency control where programmers indicate desired safety properties
- New syntax implements Hoare’s Conditional Critical Region
- Implemented using their STM
**CCR – Conditional Critical Region**

atomic (condition) {
    statements;
}

Means

1. Wait until **Condition** is satisfied
2. Execute **statements** atomically

**Pattern**

```java
public int get() {
    atomic (items != 0) {
        items --;
        return buffer[items];
    }
}
```

**Example Use**
CCR Implementation

- Built on top of Software Transactional Memory
- Uses all traditional STM commands and STMWait

```java
boolean done = false;
while (!done) {
    STMStart ();
    try {
        if (condition) {
            statements;
            done = STMCommit ();
        } else {
            STMWait();
        }
    } catch (Throwable t) {
        done = STMCommit ();
        if (done) {
            throw t;
        }
    }
}
```
STM Heap Structure

Application Heap

- A1: 7
- A2: 100
- A3: 200
- A4: 500
- A5: 600

Ownership Records

- Version 1
- Version 1
- Version 1
- Version 1
boolean done = false;
while (true) {
    STMStart();
    readvalues;
    if(STMValidate()){
        statements;
        done = STMCommit();
        if(done) {
            break;
        }
    }
}


Transaction Management

- STMStart()
- STMAbort()
- STMCommit()
- STMValidate()
- STMWait()
STM - Simple Transaction

Application Heap

A1: 7
A2: 300
A3: 200
A4: 500
A5: 600

Ownership Records

1. Version 15

1. Version 8

Version x

Version y

Transaction Descriptor 1

Committed

A1: (7,15) -> (7,15)
A2: (100,7) -> (300,8)
STM Collisions - Abort

Application Heap
- A1: 7
- A2: 100
- A3: 200
- A4: 500
- A5: 600

Ownership Records
- Version x
- Version y

Transaction Descriptor 1
- Active
- A1: (7,15) -> (7,15)
- A2: (100,7) -> (300,8)

Transaction Descriptor 2
- Active
- A2: (100,7) -> (9,8)

Committing
- Transaction Descriptor 1
- Committing
- A2's Ownership Record is unavailable! Commit Fails - abort

- Transaction Descriptor 2
- Aborted
- A2: (100, 7) -> (9,8)
boolean done = false;
while (!done) {
    STMStart ();
    try {
        if (condition) {
            statements;
            done = STMCommit ();
        } else {
            STMWait();
        }
    } catch (Throwable t) {
        done = STMCommit ();
        if (done) {
            throw t;
        }
    }
}

atomic (condition) {
    statements;
}

Abort and wait

But when do I wake up?
STM Sleep

Application Heap

A1: 7
A2: 100
A3: 200
A4: 500
A5: 600

Ownership Records

A2: (100,7) -> (300,8)
A1: (7,15) -> (7,15)

Version x

Transaction Descriptor 1

Active

A1: (7,15) -> (7,15)

Asleep

Transaction Descriptor 2

Committing

A2: (100,7) -> (300,8)
A1: (7,15) -> (7,15)

I failed my CCR condition, so I aborted and fell asleep.

Wake-up
STM Stealing - Optimization

Application Heap
- A1: 7
- A2: 100
- A3: 200
- A4: 500
- A5: 600

Ownership Records
- Version x
- Version y

Transaction Descriptor 1
- Committed
  - A1: (7,15) -> (7,15)
  - A2: (100,7) -> (300,8)

Transaction Descriptor 2
- Active
  - A3: (200,8) -> (8,9)
  - A2: (300,8) -> (300,9)
STM – HTM Parallels: Data Copies

Old Data
HTM: XCOMMIT
STM: Xaction Descriptor Entry

New Data
HTM: XABORT
STM: Xaction Descriptor Entry

Copy

Commit

Original Memory
STM – HTM Parallels: 3 Tier Implementation

HTM
- Transaction Runtime
- Snoopy Cache
- Shared Memory

STM
- Transaction Descriptors
- Ownership Records
- Application Heap
Questions

?????????