RSTM: A Relaxed Consistency Software Transactional Memory System for Multicores

Jaswanth Sreeram, Romain Cledat, Tushar Kumar, Santosh Pande

STM - Overview

- ❑ Software Transactional Memory model
  - • For Non Blocking Synchronization
  - •"Transaction": Section of code that executes atomically.
- ❑ Implementation
  - • Keep track of versions of shared data.
  - • At commit time, detect read/write conflicts between concurrent transactions.
  - • Abort & retry if conflict detected.

STM Overheads

- ❑ Overheads due to STM implementation
  - • Bookkeeping, conflict resolution, state buffering..
- ❑ Overheads due to STM semantics
  - • Difficult to specify fine grained consistency requirements for shared data
  - • Unnecessary conflicts and aborts due to variables for which "stale" values are acceptable
  - • Large performance penalty esp. for long running transactions.

RSTM – Key Ideas

Sacrifice data consistency for performance

(carefully and where appropriate!)

- ❑ Group shared variables according to the type of synchronization they need
- ❑ Relax synchronization for variable groups that don't require strict STM semantics.
- ❑ **Result:** Fewer aborts due to shared variables for which strict read/write synchronization is unnecessary to produce acceptable results.

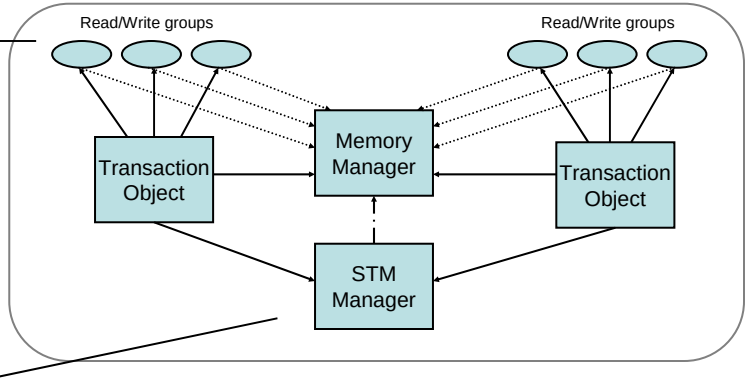
Applications: Games, Multimedia

- ❑ Distinct properties
  - • Large amount of shared state
  - • High-performance requirements (frame rates)
  - • STM overheads are sometimes unacceptable
  - • Large existing codebases
  - • Proving hard to scale using locks (programming complexity)
  - • Are "error-tolerant": Can sometimes tradeoff "accuracy" for performance
  - • Predominantly written in C/C++

Consistency Groups

- ❑ "Collection of named shared variables that a transaction reads/writes, on which a consistency policy is applied"
  - • Not all variables require strict read/write synchronization.
- ❑ Dynamic
  - • The same variable may be in different groups simultaneously or at different instants.

```
atomic T1 {
    openForRead(&a, sizeof(a), GROUP1)
    if(…) {
        openForRead(&a, sizeof(a), GROUP2);
    }
}
```



Consistency Modifiers

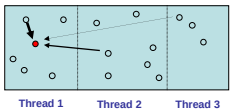
- ❑ Specifies the consistency policy to be applied to a group in a Transaction 'T'
- • **NONE** No read/write conflict check will be performed for this group when T is committing.
- • **SINGLE-SOURCE (T1, T2,..)** Exactly one of T1, T2… allowed to modify shared data in a group without causing a conflict with T.
- • **MULTI-SOURCE(T1, T2…)** Similar to SINGLE-SOURCE except any of T1, T2,… are allowed to modify data.

Bookkeeping

- ❑ Bookkeeping for memory regions instead of program objects.
  - • Vital for C/C++ programs with direct memory accesses.
- ❑ "Zone" : Basic unit of bookkeeping.
  - • A contiguous region of memory with the same metadata (version)
- ❑Smart Zone management
  - • Several strategies to minimize the number of zones to keep track of.

Case Study: Particle Simulation

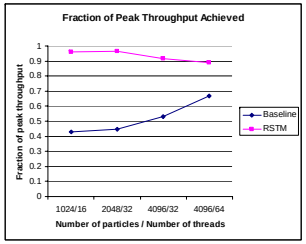
- ❑ Simulates particle interactions & motion (mutual gravity, external forces)
- • Influence of neighbors on a particle depends on the distance between them.

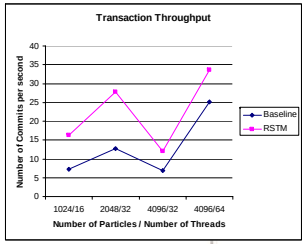


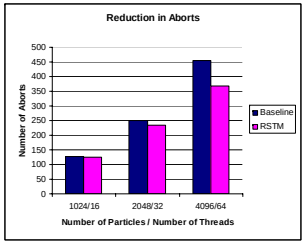
- ❑ Relaxation of STM semantics allows reading stale attributes of far away neighbor particles without aborting at commit.

Performance Results

(Baseline is standard STM with strict consistency)







Georgia Tech College of Computing