PODC 2005

NB: This poster is a reconstruction.
It may not precisely match the version
presented at the conference.

Using LL/SC to Simplify Word-based Software Transactional Memory

by
Virendra J. Marathe and Michael L. Scott
Department of Computer Science
University of Rochester

LL/SC as a restricted KCSS *(K-Compare-Single-Swap)*

Here K = 2. Already-read value of B is 4.

```
old = LL(&A)
if (B == 4)                      // compare
   if (SC(&A,new) == false)      // compare & swap
      return false
   else
      return true
else
   return false
```

NB: requires that A and B always play the same roles

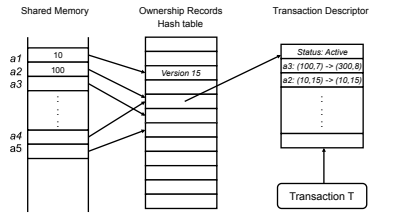
Software Transactional Memory

- A nonblocking synchronization construct
- Makes concurrent programming (almost) as simple as sequential programming
- Gives disjoint access parallel concurrency

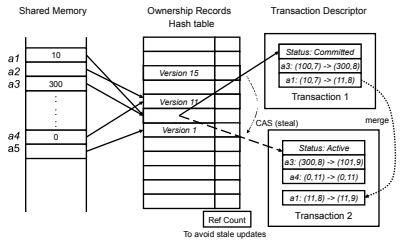
Word-based Software Transactional Memory (WSTM)

- by Harris and Fraser [OOPSLA'03]
- API for accessing shared memory
  - STMStart
  - STMRead(addr), STMWrite(addr,value)
  - STMCommit(), STMAbort()
  - STMValidate()

WSTM Data Structures



WSTM Transactions



WSTM Drawbacks

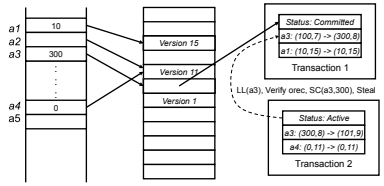
- Bounded memory blow-up problem – due to merging and false conflicts
- Complexity – merging, stealing, redo, reference counts
- False merging – merge even when not necessary

2CSS Application in Atomic Copyback

```
old = orec
   ⋮
LL(addr)
if (orec == old)                 // compare
   if (SC(addr,newval))          // compare and swap
      return true
   else
      return false
```

NB: meets the requirement that orec and addr always play the same roles

Proposed Simplification



Tradeoffs of Proposed Simplification

+ Simplicity – No merging, no stealing, no redo, just selective helping
+ No reference counts
+ No false merging

− Too many LL/SCs (equal to number of locations updated)

▪ Future Work – Empirical evaluation of proposed simplification