Virtualizing Transactional Memory

Ravi Rajwar
Microarchitecture Research Lab
Intel Corporation
ravi.rajwar@intel.com

Maurice Herlihy
Computer Science Department
Brown University
mph@cs.brown.edu

Konrad Lai
Microarchitecture Research Lab
Intel Corporation
konrad.lai@intel.com

What Problems are they trying to solve?

Where does transactional state reside?

• What, then, has to happen on a transactional memory operation? (assume cache miss)

• How do they make this fast? (two mechanisms)
What’s in the?

- XSW
- XADT
- XF
What happens on a context switch?

What happens on a commit?

- How is it made atomic?

![Image](Image 1)

Figure 4 Commit sequence for virtualized transactions in YTM. Here, locations G and F map to the same XF entry. 

What happens on an abort?

What are nested transactions?

- What is flattening?
Architectural Support for Software Transactional Memory

Bratin Saha, Ali-Reza Adl-Tabatabai, Quinn Jacobson  
Microprocessor Technology Lab, Intel Corporation  
bratin.saha, ali-reza.adl-tabatabai, quinn.a.jacobson@intel.com

Why STM rather than HTM?

Basic Implementation Idea?
In STM, what needs to happen on a

- Read
- Store
- Commit
- Abort

What is “eager version management”?

Where is the most performance lost on a STM?

Figure 5: HASTM object-based read barrier

validate() {
  markCount = readMarkCounter();
  resetMarkDir();
  if (markCount == 0) /* no merge or eviction*/
    return;
  /* perform full read set validation */
  for <nonrec, var> in transaction's read set
    if (nonrec != var)
      abort();
}

Figure 6: HASTM validation
What is?

- Aggressive-mode HASTM

Results

The problem

Sequential execution overhead

Figure 12: STM execution time breakdown

Figure 16: Relative execution time for TM schemes
Figure 17: Performance breakdown for HASTM

Application Results

Figure 18: Multi-core scaling for BST

Figure 19: Multi-core scaling for Btree

Figure 20: Multi-core scaling for hash table