Research Interest Statement

Haosen Wen
(Advised by Prof. Michael L. Scott)

Computer Science Department, University of Rochester, Rochester, NY 14627

Abstract. My research focuses on persistent software transactional memory, an application of concurrent data structures on non-volatile byte-addressable memories (NVBM). Given past explorations on both persistent semantics for concurrent data structures (by Joe Izraelevitz et al.) and object-based transactional memories (by Maurice Herlihy et al. and Virendra Marathe et al.), I believe I can build a software transactional system that supports composable atomicity and persistence, and has better performance and scalability.

Overview

As traditional volatile RAM approaches the limit of its capacity growth, non-volatile byte-addressable memories (NVBM) are likely to be the future of main memory, with a tempting side-feature of persistency, which suggests research into corresponding data structures and applications. In the last few years, semantics of concurrent data structures in the context of persistency [1], together with concurrent data structures based on such semantics, enabled applications to be developed with NVBMs, and software transactional memory (STM) is one promising candidate.

State-of-the-art approaches to STM for persistent memory, like Mnemosyne [2] and NV-heaps [3], tend to execute pfences after every persist of object or log records, which can be overkill for correctness that compromises too much on performance. As a counter-example, persistent data structures like Dalí [4], a periodically-persistent concurrent hash map that provides buffered durable linerizability on persistent main memory and volatile caches, showed us a more relaxed approach to persistency with lower synchronization overhead. Hence, a possibly promising plan would be a new object-based STM based on Dalí.

Summary of past work

My work so far has focused mainly on the implementation of concurrent data structures.

In my first semester, my research on potential bottlenecks of inter-thread/inter-process communication brought some fundamental understandings of how factors like context switching, cache locality and TLB misses affect the performance of concurrent programs. This work provided guidance for later research on concurrency. This is still an open research topic for me.

During my second semester, I help implemented Polytree, a general framework for easily creating nonblocking concurrent tree structures, which provides thread-safe lookups, atomic node updates, and atomic sub-tree substitutions.
When implementing comparing tree structures for Polytree, like Bonsai Trees [5], we wished for a memory management scheme that would be robust to thread stalling, easy to use, and fast. Inspired by epoch-based memory reclamation algorithms [6, 7] and some state-of-art approaches like Hazard Eras [8], we developed Interval-Based Memory Reclamation [9], which was just accepted for presentation at PPoPP’18. Prior to our work, epoch-based memory management schemes were the fastest and most user-friendly ones. To ensure correctness, an operation must protect (or reserve) all blocks that are (or will be) detached after its start time (epoch) from being reclaimed, so that it can safely access every block that it can possibly reach. Unfortunately, an unbounded number of blocks can be reserved by some stalled thread, since all blocks detached after its start epoch will be reserved. Our work solved this problem by marking a finite range of epochs “reserved” per thread, and all objects whose life intervals intersect with any “reserved” ranges will be protected from reclamation, so the total number of unreclaimable blocks is bounded.

**Ongoing works and future plans**

I am currently taking a first step into persistent data structures by re-implementing Dalí, since the original code belongs to HP and is not available to us. I’m also tuning Dalí with different implementations of its persisting fence, a mechanism that flushes new records in the hashmap to persistence periodically. Candidates for the fence include but are not limited to whole cache flush, Intel’s Cache Allocation Technology (CAT), and Page Coloring techniques. My next step will be implementing an object-based software transactional memory based on Dalí. I will try to extend some object-based transactional memory designs like DSTM [10], ASTM [11] and/or RSTM [12] in the context of persistent main memory, or try to build a new STM system that takes inspiration from them.

**References**


