Real-Time Software Transactional Memory

Shai Caspin

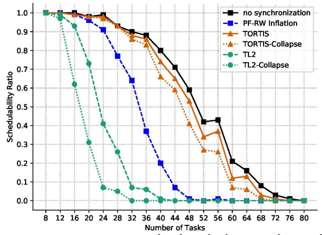
THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Motivation

Software Transactional Memory (STM) is a well-studied computer science abstraction that eases programmer burden of synchronizing concurrent code. Code that needs synchronization is annotated like so:

```
transaction {
        ...synchronize this….
}
```

STM generally works by executing transactions and then retrying them if conflicts occur. Retries are hard to predict, making STM impractical for **real-time systems**, where we need timing guarantees and bounds on execution.

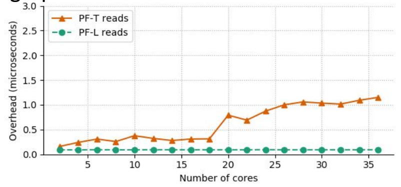


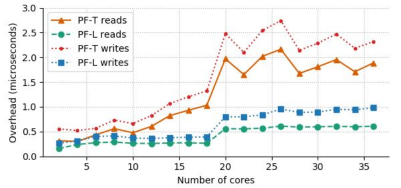
We present an entirely lock-based and **retry-free** approach to STM. This approach (TORTIS) outperforms a lock-based retry-dependent approach (TL2) in terms of schedulability [1]. Higher schedulability means less missed deadlines.

Overheads

Lock-based STM implementation should use locks that scale with core counts. The locks with the lowest blocking bounds are **phase-fair reader/writer locks** designed by Anderson and Brandenburg [2].

Overheads for these locks is high, so we developed a new locking protocol, the PF-L that improves overheads over the standard high-performance PF-T.

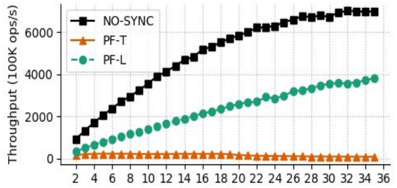


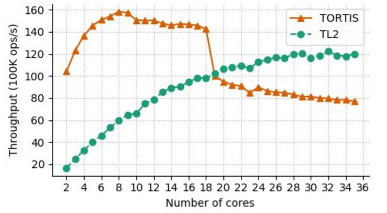


Our new protocol, PF-L was able to reduce overheads for most workloads, shown here for all-reads and 50% writes.

By minimizing cache-line invalidations and allowing core-local read spin locations overheads were reduced.

Throughput





STM is evaluated using throughput and scaling with increased core counts. Time for 100k operations on a red-black tree are measured with varying write (insert) workloads. Here we see throughput trends for an all-read workload and a 5% write workload.

Lock-based STM only experiences linear scaling for reading where parallel accesses are allowed, and does not allow for parallelism for writes as provided by the retry-based approach. Throughput trends show retry-free STM can be practical with low-enough lock overhead.

Takeaways

My work contributed to several aspects advocating for retry-free STM for real-time systems including:

- Empirically showing temporal gains of retry-free lock-based synchronization.
- Designing low-overhead locking protocol that is more practical in the application domain.
- Comparing retry-free and retry-based STM in practical applications and case studies.

This work helps redefine performance for STM in systems where synchronization is needed. Performance is not strictly throughput, but can also mean promising temporal guarantees.

References

[1] D. Dice, O. Shalev, and N. Shavit. "Transactional locking II". In:International Symposium on Distributed Computing. Springer. 2006, pp. 194–208

[2] B. Brandenburg and J. Anderson. "Spin-based reader-writer synchronization for multiprocessor real-time systems". In:Real-Time Systems 46.1(2010)