Optimistic Software Transactional Memory Prototype

Name: Sidharth Mishra
Class: CS 252
Assignment: Project Proposal
Due Date: 10/6/2017 - 11:59 PM

Last Modified: October 6, 2017:2:46pm

I Motivation

Concurrency primitives built into languages such as `Java` are powerful but have complex syntax and require careful use. For example, Java provides the `synchronized` blocks and methods for granular locks, but these require careful handling because they might lead to degraded performance due to excessive locking or data-race conditions if not applied at correct spots. Moreover, modern languages such as `Go` supply simpler concurrency directives such as `goroutines` and `channels` but, these too, at times, require locking and can be difficult to reason about. One possible solution is to use *Software Transactional Memory(STM)* to handle concurrency. The STM relieves the developer from thinking about or writing parallel code by letting them write serial code and the STM manager handling the system-specific details to run their code in parallel.

II Objective

The objective of this project is to build a working prototype of a optimistic STM using the working principles mentioned in [1] and [2]. The transactions in the STM will be using timestamp(versioning) protocol and there will be no locks used.

Language(s) of choice:

- Java 1.8.0_144
- Go 1.9

III Progression Timeline

Week 1: October 10 - October 16	Read through [1] and implement the Java version
Week 2: October 23 - October 29	Look up Go, implement working prototype in Go
Week 3: November 1 - November 7	Refine implementation of Java version, implement small examples
Week 4: November 9 - November 15	Refine implementation of Go version, implement small examples
Week 5: November 18 - November 24	Work on deliverables(code and presentation files)

References

[1] N. Shavit and D. Touitou, "Software transactional memory," *Distributed Computing*, vol. 10, no. 2, pp. 99–116, Feb 1997, (Last accessed 10-5-2017). [Online]. Available: https://doi.org/10.1007/s004460050028

[2] M. Weimerskirch, "Software transactional memory," (Last accessed 10-5-2017). [Online]. Available: https://michel.weimerskirch.net/wp-content/uploads/2008/02/software_transactional_memory.pdf

[3] S. P. Jones, "Beautiful concurrency," (Last accessed 10-5-2017). [Online]. Available: https://www.schoolofhaskell.com/school/advanced-haskell/beautiful-concurrency