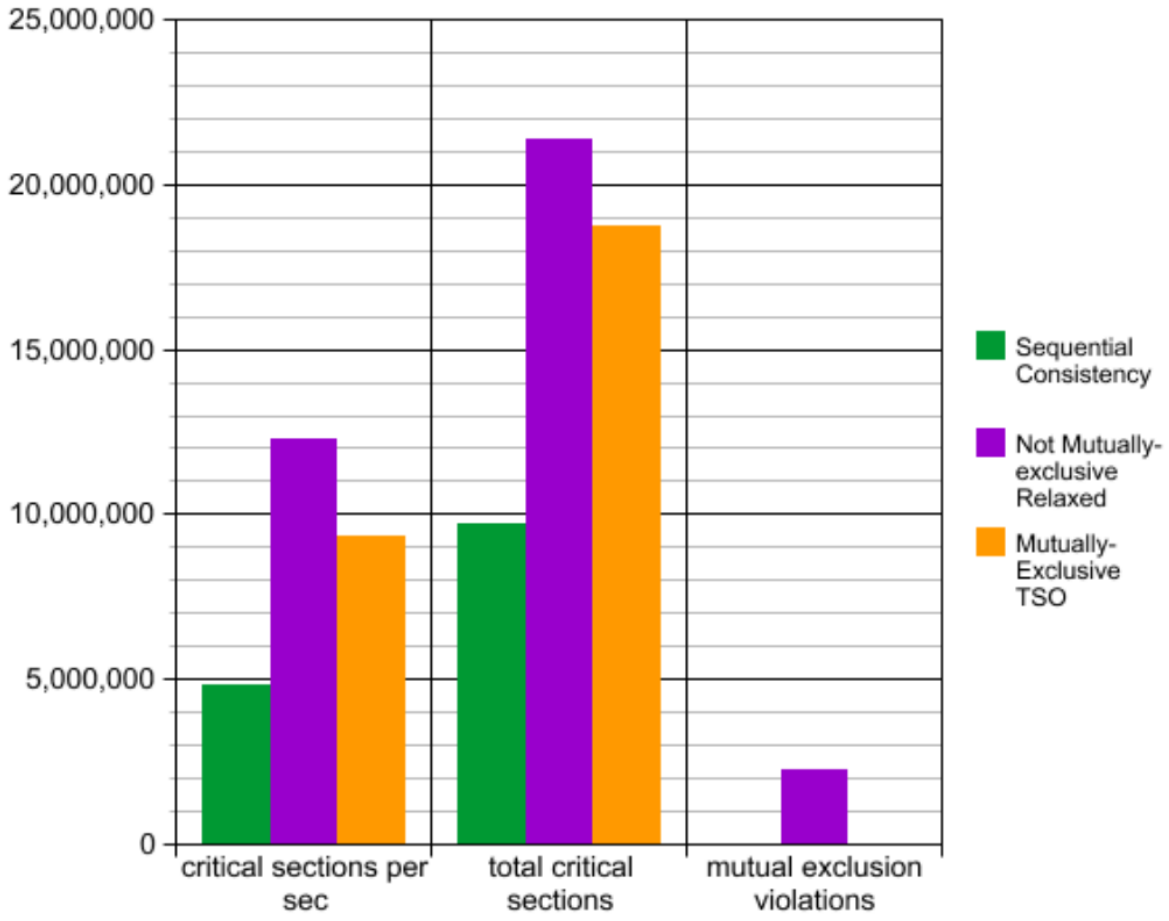
Ari ‘aith’ Iramanesh

In part 3 we compare Dekker’s algorithm under Sequential Consistency to the weaker Total Store Order memory model (TSO).

A typical run of the implementations:

	Sequential Consistency	Not Mutually-exclusive Relaxed	Mutually-exclusive TSO
# critical sections per second	4,847,310	12,272,200	9,356,490
# total critical sections	9,694,619	21,345,220	18,712,973
# mutual exclusion violations	0	2,279,279	0
% of times mutual exclusion violated	0	9.28638%	0



The Mutually-exclusive TSO version, which puts a FENCE between every possible store followed by a load, ensures that mutual exclusion occurs despite the weaker memory model. This consistently results in a 2x total throughput increase from the Sequentially Consistent version. This speedup is demonstrated because under Sequential Consistency, the Store Buffers are flushed into Main Memory whenever a store or load is performed. This is because of how c++ compilers place an mfence instruction before or after every store or load under SC. Writing to Main Memory is obviously expensive, and in TSO, we can avoid having to write to Main Memory! However, fences *must* still be placed whenever a store is followed by a load in order to maintain program functionality. This is because under TSO, a store followed by a load allows that store and that load to be reordered. Fences, in this scenario, ensure that the stores are followed by the loads.