Homework 3, Part 1

Cooper Faber

June 2021

1 Results

1.1 Table

Mutex	Throughput	Critcal Sections	ME Violations	% of ME violated
scdekker	$1.05242*10^7$	21048364	0	0%
rdekkers	$2.37052*10^7$	47410326	3292229	6.944%
tsodekkers	$1.67526*10^7$	33505172	0	0%

1.2 Graph

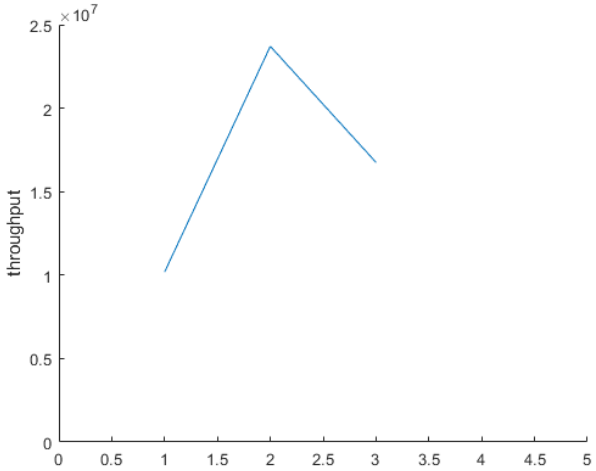

Figure 1: Results

The mutexes are represented by the x-axis in the above graph, where x=1 is sc, x=2 is r, and x=3 is tso.

2 Implementation & Analysis

My implementation 3.1 and 3.2 were simple, and don't particularly need to be touched upon. 3.1 is translated pseudocode, and 3.2 is 3.1 with relaxed memory orders for every single load/store. 3.3 is also simple - going from class notes, TSO only allows store-load reordering, which can be stopped by placing a fence before every load or after every store. I chose the latter, implementing a fence after every store for a total of 6 fences. One is in the init statement, which is very likely not needed (and, at worst, increases overhead). The other 5 are spread across lock&unlock, verifying that sequntial consistency is enforced. The results are lovely: although we don't get the insane speedup that the broken model has, every single test run (out of 5, data above is averaged) beat out the normal model, while retaining mutual exclusion. These results were expected from the implication of the assignment - it would be a little odd if we learned these techniques, only to find out that sequential consistency was faster this whole time.