Ari ‘aith’ I.

	‘Old’ Reader-writer lock	Fair Reader-writer Lock
Reader calls	1,804,664	149,385
Writer calls	25,772	1,775,407
Total calls	1,830,436	1,924,792

In part 2 I implemented a clever way of implementing a Fair Reader-Writer Lock, outlined by Vlad Popov and Oleg Mazonka. It builds on the standard approach of ‘draining out,’ or preventing further Readers from entering a Critical Section after a Writer has attempted to acquire the lock. Once all writers have been drained, then a single Writer may enter. This implementation relies on each Reader incrementing both the counter of Readers who’ve entered and exited the Critical Section in order to efficiently check whether or not Writers can enter the Critical Section. The *downside* is that an integer overflow of that counter may occur. In C++ this equates to 2,147,483,647 Reader calls.

This implementation definitely gives more space towards the Writers than the old version did, as well as increase net throughput. The issue is that now there are too many Writer calls. This is due to the Writers-first nature of the ‘draining out’ mechanism. In a system where there are many more Readers than Writers, this issue won’t be as prevalent. But for these tests, where there are 6 Readers to 2 Writers, the issue imbalance is shown.

References

https://arxiv.org/abs/1309.4507 (TODO turn into citation)