CS3523: Operating Systems - II

Programming Assignment 5 - Report

Soumi Chakraborty

ES19BTECH11017

1 Overview

This assignment implements the Readers-Writers with preference given to readers and the fair Readers-Writers algorithms.

2 Implementation

Each file first declares all the global variables (including the semaphores) required for the algorithms so as to avoid having to pass them to each thread separately. The log file object has also been declared globally so that each thread can write to it without having to reopen and close it for each thread.

2.1 Readers-Writers Algorithm

The `readers()` and `writers()` functions implement the aforementioned mutual exclusion algorithms:

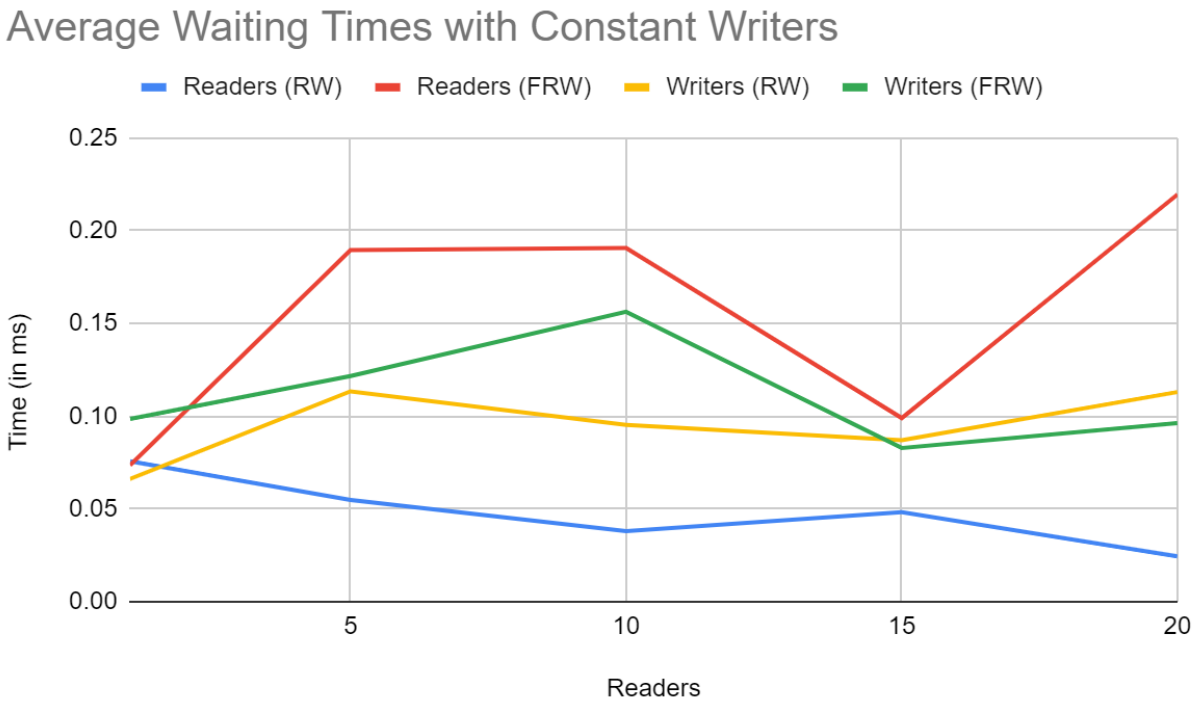
- The generator is seeded at the beginning for the exponential random number generators.
- The functions make use of the `localtime()` function to retrieve the current time in minutes and seconds, and the `gettingtimeofday()` function is used to increase the precision to milliseconds.
- The rest of the structure of the program is the same as the pseudocodes given in the assignment document, the textbook and Wikipedia. Each algorithm has a slightly different entry and exit section while everything remains the same.
- The RW algorithm makes use of two semaphores whereas the FRW algorithm uses 3 semaphores. The functions `sem_wait()` and `sem_post()` are used as the wait and signal functions.
- The `usleep()` function is used to simulate the critical and the remainder sections using the values generated by the exponential distribution function.
- The times have been printed up to micro seconds since the algorithm is very fast and printing the time only up to seconds doesn't gie us any useful information.

The main function only reads the input file into variables creates the required number of reader and writer threads to execute the functions and joins them once they are done executing. It also prints the average time of all the threads combined to a file, and other

stats like worst waiting times and average waiting times for readers and writers separately to the console.

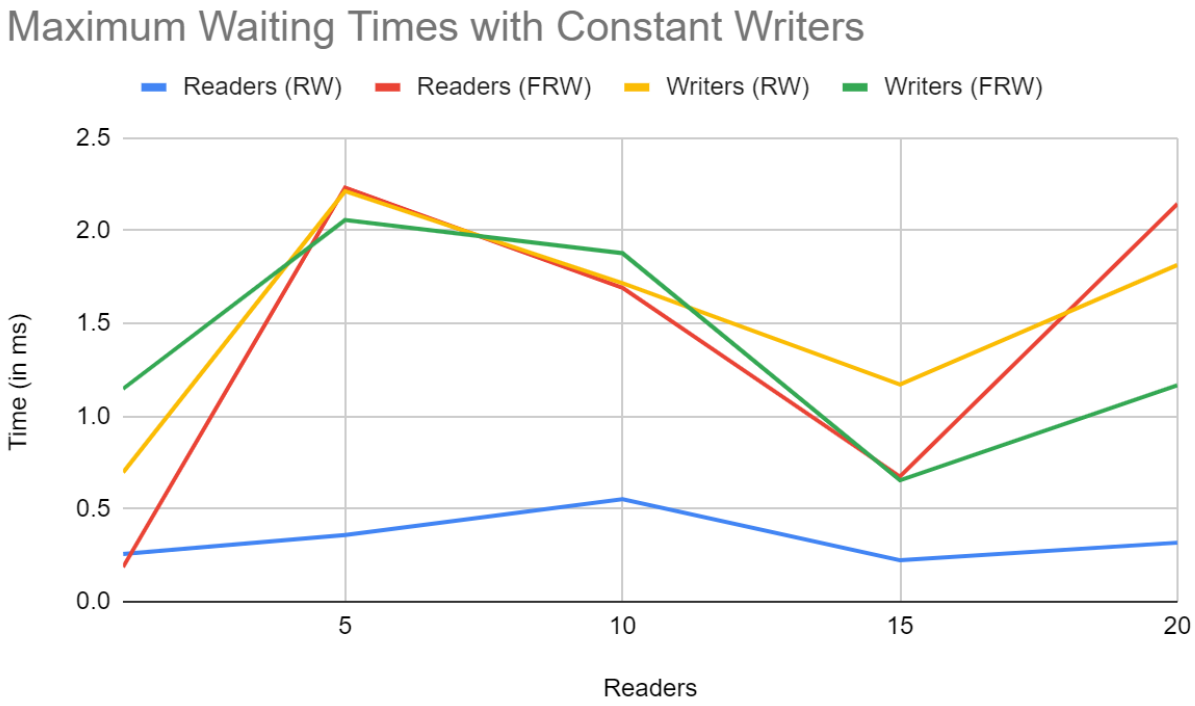
3 Comparison

3.1 Average Waiting Times (constant writers)



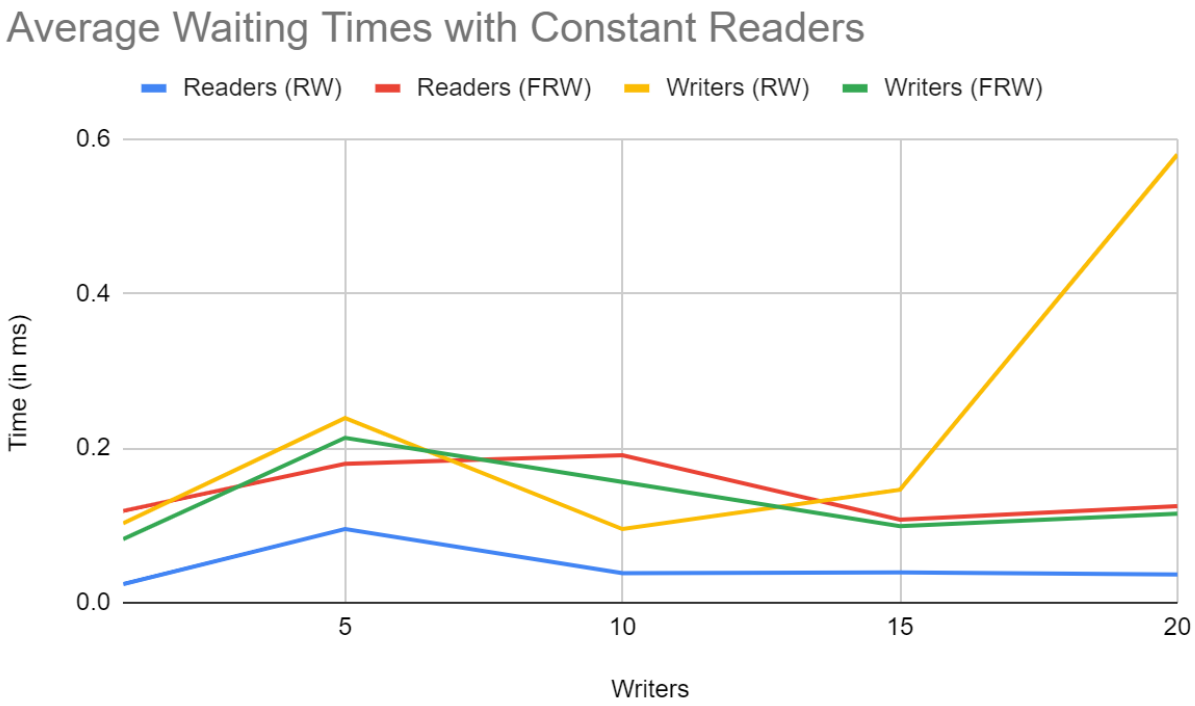
It is clear from the graph that as a general trend, the RW readers have the lowest average waiting time which is in accordance with the expected behaviour since the algorithm prefers readers. Moreover, the readers and writers in the FRW algorithm become similar as the number of readers increases.

3.2 Maximum Waiting Times (constant writers)



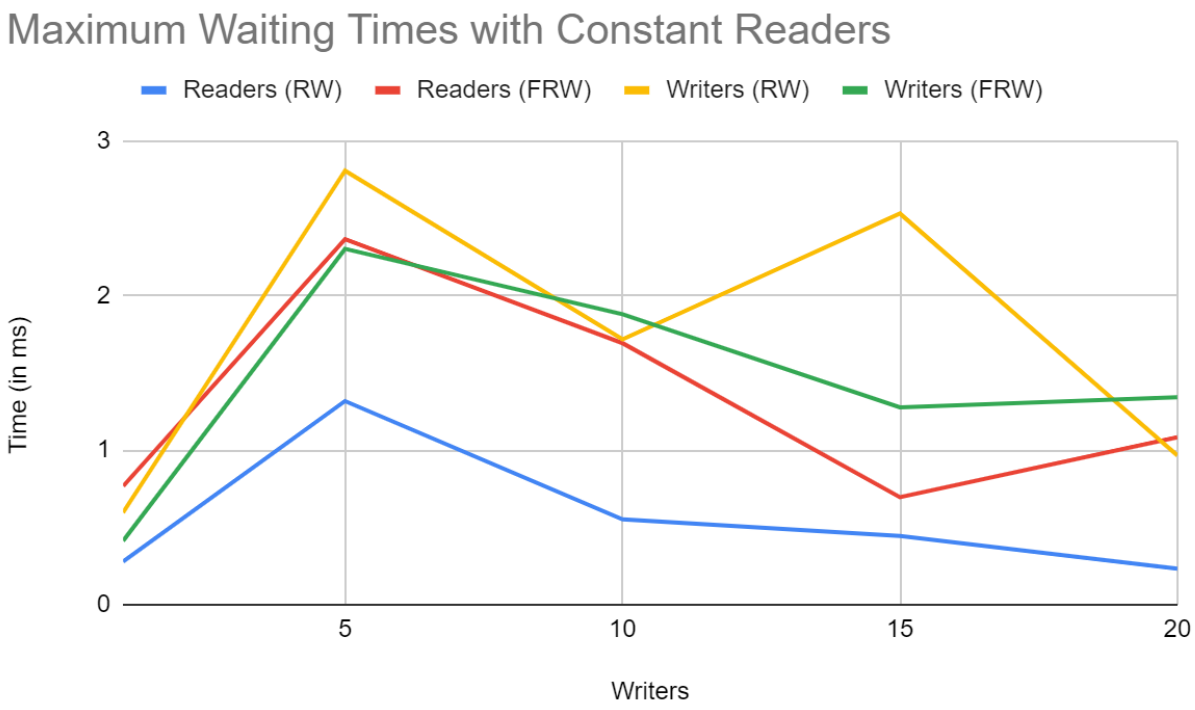
It is clear from the graph that as a general trend, the RW readers have the lowest maximum waiting time which is in accordance with the expected behaviour since the algorithm prefers readers. Moreover, the readers and writers in the FRW algorithm become similar as the number of readers increases.

3.3 Average Waiting Times (constant readers)



It is clear from the graph that as a general trend, the RW readers have the lowest waiting time which is in accordance with the expected behaviour since the algorithm prefers readers. Moreover, the other threads also have similar times since the number of readers is constant, which is the pivotal parameter in terms of timing changes since one algorithm prefers readers and the other is completely fair.

3.4 Maximum Waiting Times (constant readers)



It is clear from the graph that as a general trend, the RW readers have the lowest waiting time which is in accordance with the expected behaviour since the algorithm prefers readers. Moreover, the readers and writers in the FRW algorithm become similar as the number of writers increases.