Operating Systems–2

CS 3510 Spring 2019

Programming Assignment 4:

Implement solutions to Readers-Writers and Fair

Readers-Writers problems using Semaphores

INSTRUCTOR: DR. SATHYA PERI

Report By:
Vijay Tadikamalla
CS17BTECH11040

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Task

To solve the Readers-Writers problem and Fair Readers-Writers problem using Semaphores as discussed in the class in C++. You have to implement these two algorithms and compare the average and worst-case time taken for each thread to access the critical section (shared resources).

Approach and Implementation

1. To achieve our above-mentioned goal make two functions reader and writer which takes thread_index as a parameter. We will pass this function along with the thread index to each thread and calculate the average time and worst time taken for both reader and writers
2. Functions and data-types of the chrono library (and other libraries) like
    a. **std::chrono::system_clock, std::chrono::system_clock::now()**
    b. **std::chrono::time_point**
    c. **struct tm - Time structure**
    d. **struct tm *localtime(const time_t *timer)**

    were used to calculate the average waiting time and max waiting time.
3. **int usleep(useconds_t *usec*)** function was used to suspend the execution of the thread for microsecond intervals
4. **template <class RealType = double> class exponential_distribution**: This is a random number distribution that produces floating-point values according to an exponential distribution, which is described by the following probability density function:

    $$p(x|\lambda) = \lambda e^{-\lambda x} \quad , \; x > 0$$
5. We make two exponential distributions and pass the value of 1/μp, 1/μc in the constructor. Later this can be used to obtain random numbers t1 and t2 with values that are exponentially distributed with an average of μp, μc seconds.

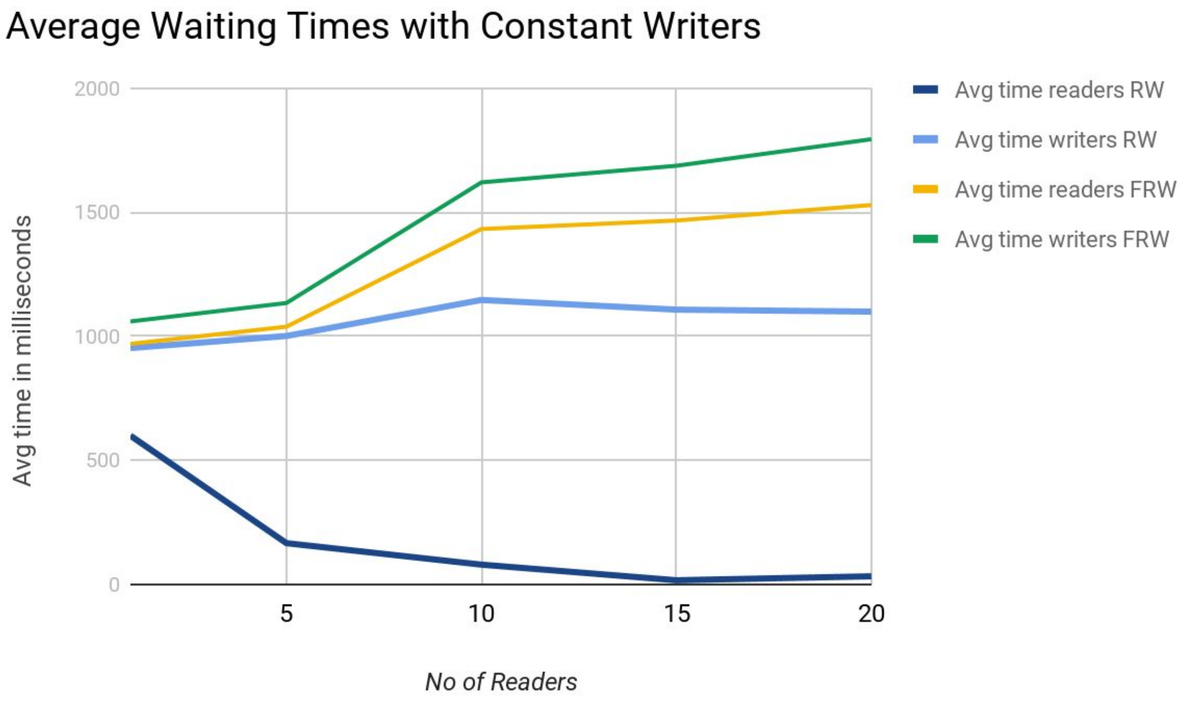
```
distribution1 = new exponential_distribution<double>(1/mu_p);
distribution2 = new exponential_distribution<double>(1/mu_c);
```

6. In Reader writer problem we use semaphores to ensure that no reader is reading while a writer is writing. At the same time, we allow multiple readers to read at the same time.
7. Two semaphores “mutex” and “rw_mutex” are used in this implementation. The semaphore rw_mutex is used to ensure the mutual exclusion between readers and writers. The semaphore mutex is used to prevent multiple updates to a shared variable reader_count. This variable tells us the no of readers inside the critical section.

8. The above-mentioned implementation can result in starvation of writers. So in the fair reader-writer solution, we use another semaphore “in” we to ensure that no one starves.

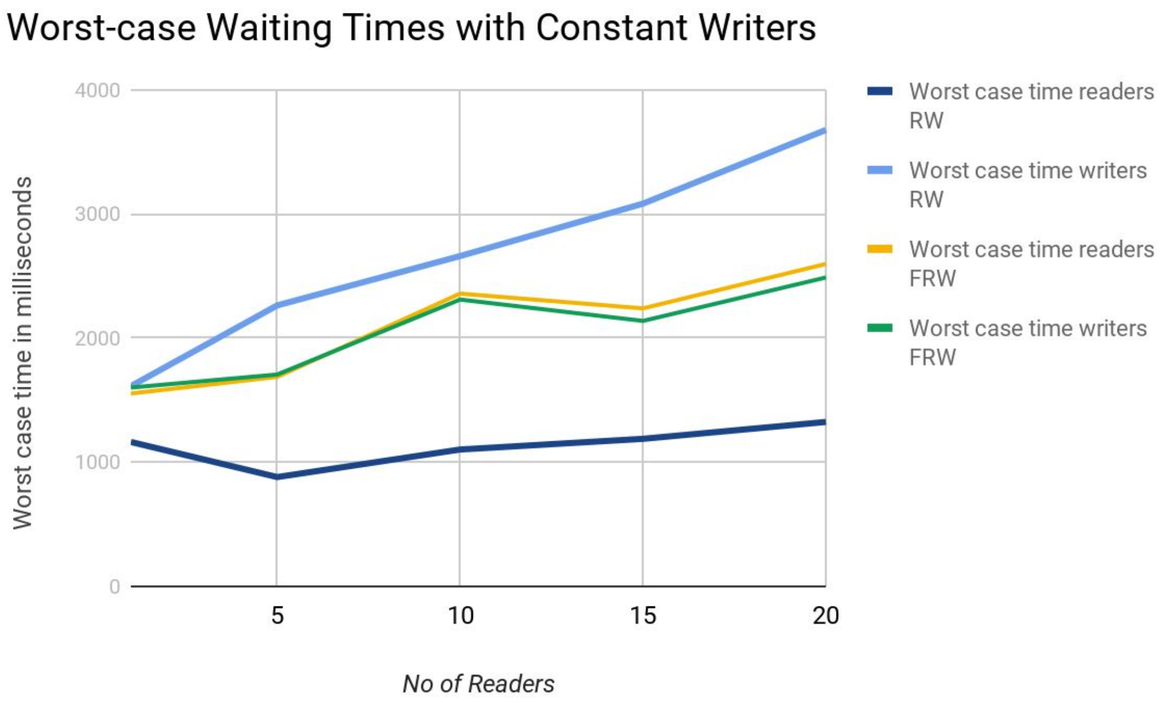
Input Parameters

kr=10
kw=10
μ_cs=0.1
μ_rem = 0.1



Graph1: Observations and inference

1. The average time taken by readers in FRW solution ⩾ The average time taken by readers in RW solution. This significant difference is observed because in the normal RW solution the writers used to starve but in the FRW solution, starvation is prevented which in turn increases the average waiting time of readers.
2. The average waiting time of readers(in RW) is initially high because when no of readers are very low, the CS is mostly occupied by the writers.

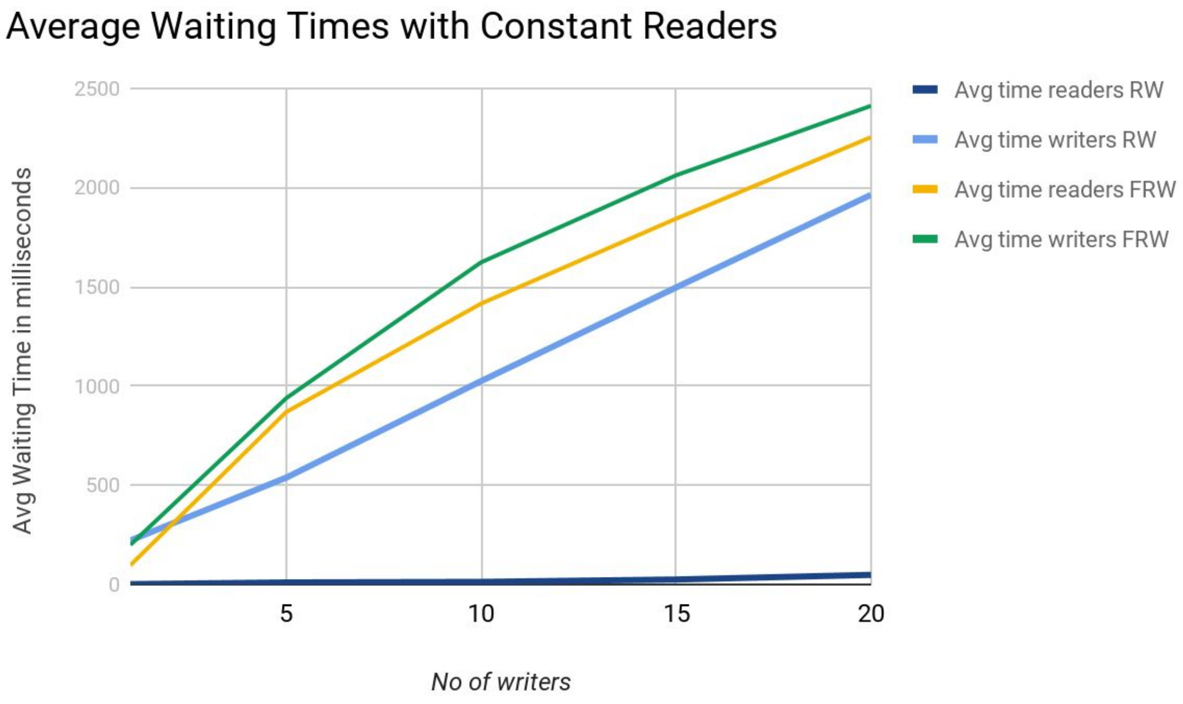


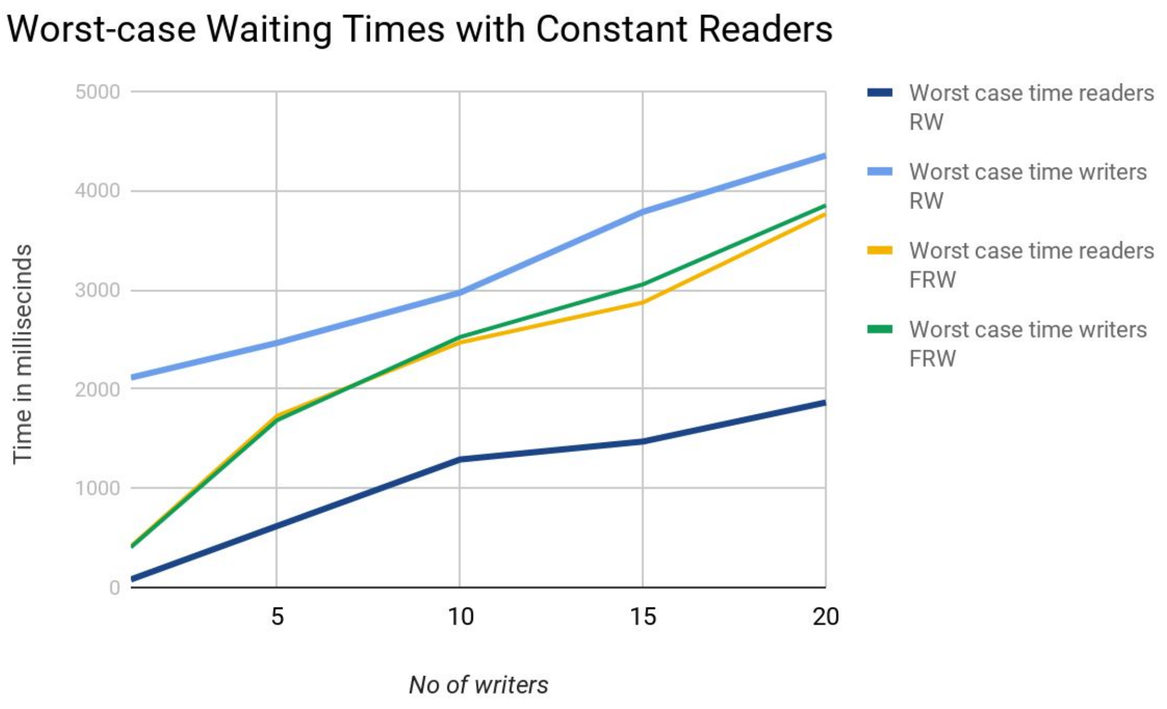
Graph2: Observations and inference

1. We can clearly see from the graph that the starvation of writers is prevented as the worst case time is significantly less in case of FRW as compared to RW
2. Worst case time is almost the same for reader and writer in FRW. This shows that no one is starving.
3. The worst case time for readers in FRW solution $\geqslant$ The worst case time for readers in RW solution. This significant difference is observed because in the normal RW solution the writers used to starve but in the FRW solution, starvation is prevented which in turn increases the average waiting time of readers.

Graph3: Observations and inference

1. Average waiting time increases in all the scenarios as no of writers increase.
2. The average time taken by the reader in FRW solution ⩾ The average time taken by the reader in RW solution. This significant difference is observed because in the normal RW solution the writers used to starve but in the FRW solution, starvation is prevented which in turn increases the average waiting time of readers.





Graph 4: Observations and inference

1. We can clearly see from the graph that the starvation of writers is prevented as the worst case time is significantly less in case of FRW as compared to RW
2. Worst case time is almost the same for reader and writer in FRW. This shows that no one is starving.
3. The worst case time for readers in FRW solution ⩾ The worst case time for readers in RW solution. This significant difference is observed because in the normal RW solution the writers used to starve but in the FRW solution, starvation is prevented which in turn increases the average waiting time of readers.
4. Worst case time increases in all the scenarios as no of writers increase.