Operating Systems II : CS3523
Spring 2019

Programming Assignment 4 : Implement solutions to Readers-Writers and Fair Readers-Writers problems using Semaphores

Instructor : Dr. Sathya Peri
Author : Puneet Mangla (CS17BTECH11029)

Design of Program :

We implemented and comapred **Readers-Writers and Fair Readers-Writers problems** using Semaphores.

Reader – Writers

Two semaphores rw_mutex and mutex are used.

```
rw_mutex = 1;
mutex = 1;
```

Writer :

```
while (true) {
    wait(rw_mutex);
    . . .
    /* writing is performed */
    . . .
    signal(rw_mutex);
}
```

Reader :

```
while (true) {
    wait(mutex);
    read count++;
    if (read_count == 1)
        wait(rw_mutex);
    signal(mutex);
    . . .
    /* reading is performed */
    . . .
    wait(mutex);
    read count--;
    if (read_count == 0)
        signal(rw_mutex);
    signal(mutex);
}
```

Fair Reader - Writers

Three semaphores rw_mutex , mutex and queue are used.

```
rw_mutex = 1;
mutex = 1;
queue = 1;
```

Writer :

```
while (true) {
        wait(queue);
        wait(rw_mutex);
        signal(queue);
        . . .
        /* writing is performed */
        . . .
        signal(rw_mutex);
}
```

Reader :

```
while (true) {
        wait(queue);
        wait(mutex);
        read count++;
        if (read_count == 1)
            wait(rw_mutex);
        signal(queue);
        signal(mutex);
        . . .
        /* reading is performed */
        . . .
        wait(mutex);
        read count--;
        if (read_count == 0)
             signal(rw_mutex);
        signal(mutex);
}
```

Comparison metrics :

1. **Average waiting time :** the average time taken by a thread to enter the CS.
2. **Worst Case waiting time** : the maximum time taken by a thread to enter the CS.

Algorithm :

1. The main thread reads the "inp-params.txt" and stores all the parameters as global for all theads to be accessible.

2. n_w writer threads and n_r reader threads are created and their id are stored in array pthread_t writer_tid[n_w] and reader_tid[n_r] respectively.

3. default_random_engine is used to generate random numbers from exponential_distribution to stimulate critical and remainder section . Parameters to the distribution were were $1/\mu_{cs}$ and $1/\mu_{rem}$ respectively.

4. void* writer/reader functions stimulate reader and writer . Their implemntation depends on whether we are using simple or fair version.

5. For printing logs fprintf/printf are used instead of cout as cout streams causing mixing of logs.

Graphs :

For comparison of both implementations , we draw four graphs as follows :

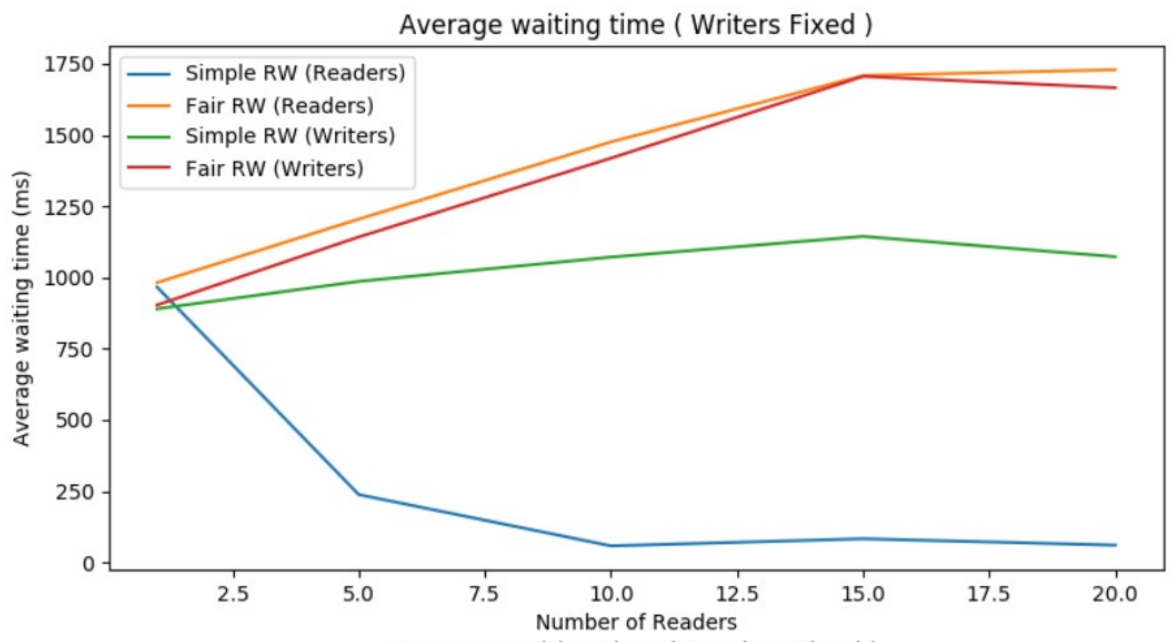
$\mu_{cs} = \mu_{rem} = 0.1$ for all graphs .

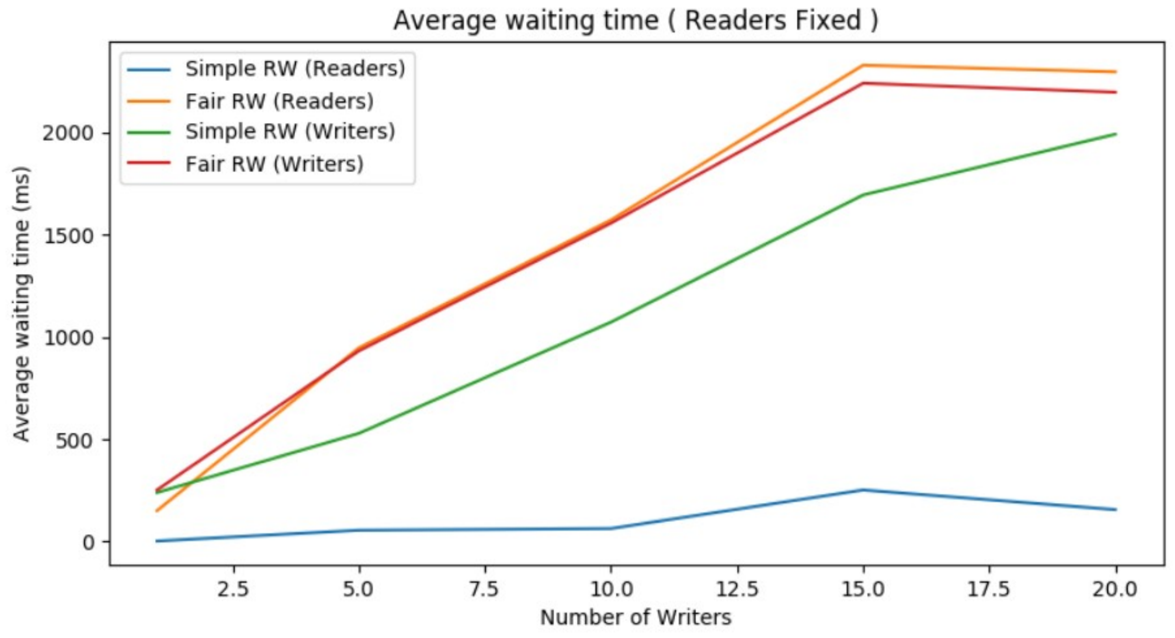
1. **Average Waiting Times with Constant Writers:**
   In this graph, we measure the average time taken to enter the CS by reader and writer threads with a constant number of writers. Here we vary the number of reader threads `n_r` from 1 to 20 in the increments of 5 on the X-axis. We have all the other parameters fixed: Number of writer threads, `n_w = 10`, `k_r = k_w = 10`.

2. **Average Waiting Times with Constant Readers:**
   In this graph, we measure the average time taken to enter the CS by reader and writer threads with a constant number of readers. Here we vary the number of reader threads `n_w` from 1 to 20 in the increments of 5 on the X-axis. We have all the other parameters fixed: Number of writer threads, `n_r = 10`, `k_r = k_w = 10`.

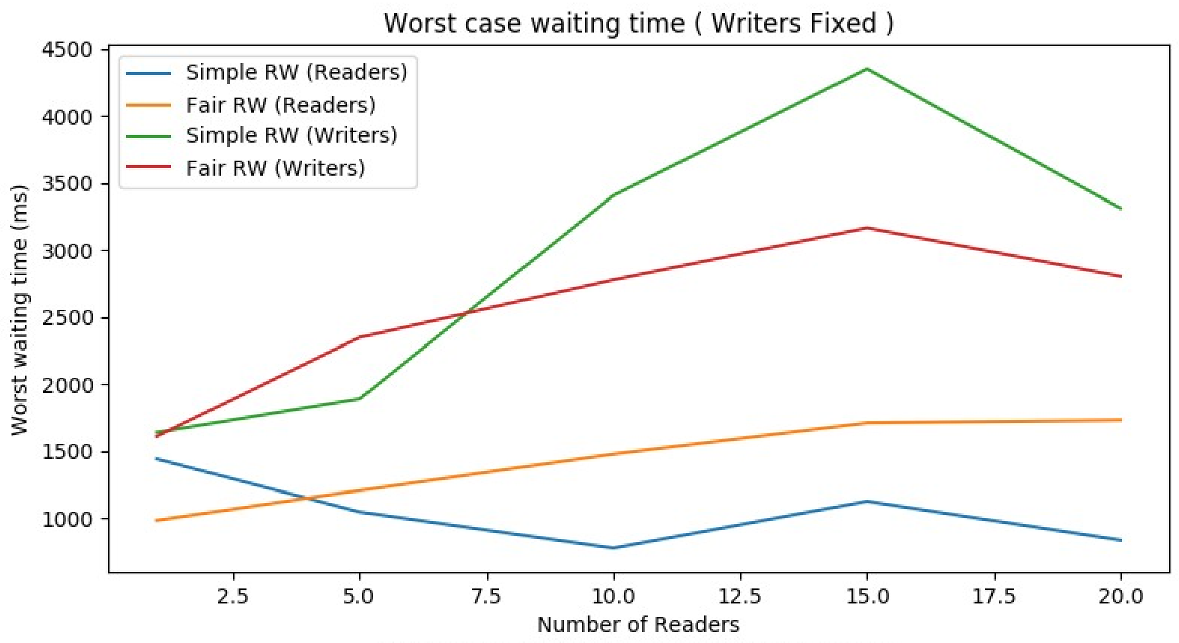
3. **Worst-case Waiting Times with Constant Writers:**
   In this graph, we measure the worst time taken to enter the CS by reader and writer threads with a constant number of writers. Here we vary the number of reader threads `n_r` from 1 to 20 in the increments of 5 on the X-axis. We have all the other parameters fixed: Number of writer threads, `n_w = 10`, `k_r = k_w = 10`

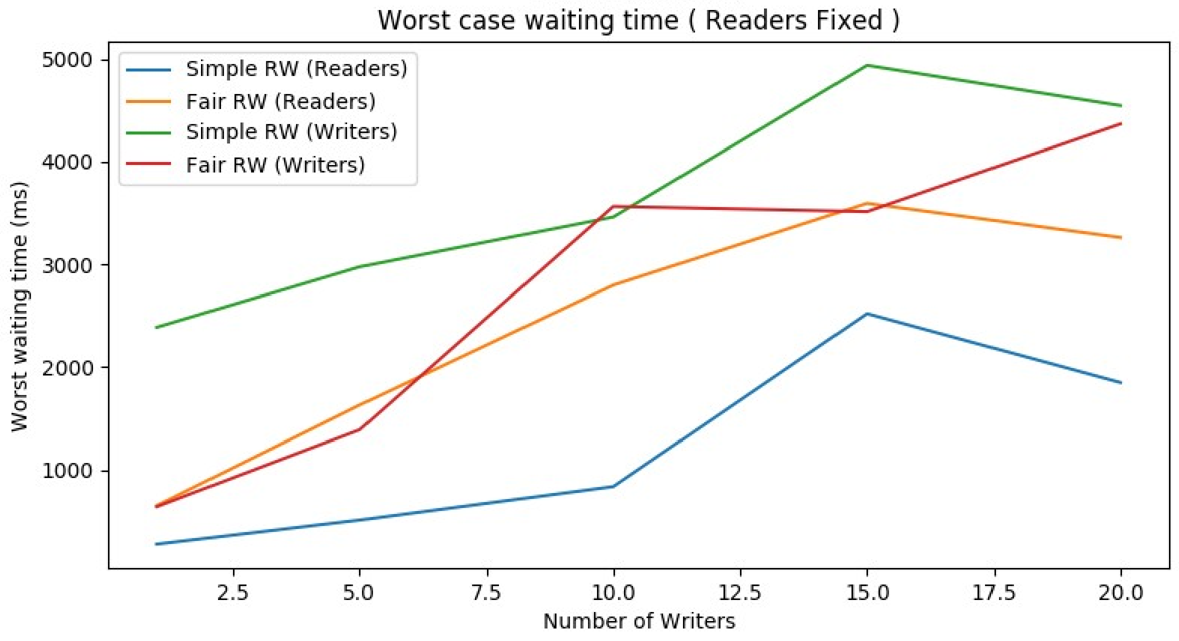
4. **Worst-case Waiting Times with Constant Readers:**
   In this graph, we measure the worst time taken to enter the CS by reader and writer threads with a constant number of readers. Here we vary the number of reader threads `n_w` from 1 to 20 in the increments of 5 on the X-axis. We have all the other parameters fixed: Number of writer threads, `n_r = 10`, `k_r = k_w = 10`.





The above graphs represents case 1 and 2. We can see that the average waiting time in fair RW is more than standard RW in all the scenarios . The reason is that fair RW ensures fairness at cost of increased waiting time of threads who were not waiting in standard RW. The average time of reader threads in standard RW almost converges at end.





The above graphs represents case 3 and 4. We can see that worst case waiting time is less in Fair RW for writers which were starving in satandard RW explaining fairness. The worsst case time of readers is more in case of Fair RW because in ensuring fairness to wriers some readers may have to wait more time than it should in standard RW.

Conclusion :

Fair RW implementation ensures that no thread starves but at the cost of increased average waiting time.

The decrease in worst case time is only evident in case of writers which were starving in standard RW . The decrease is at cost of increase worst case waiting in case of readers.