Performance Comparison of Reader-Writer Algorithms

SURBHI (CS22BTECH11057)

March 17, 2024

1 Introduction:

The provided codes implement the Reader-Writer problem using two different approaches: RW (Reader-Writer) and FAIR-RW (Fair Reader-Writer). The problem involves multiple threads accessing shared resources (data) where readers can access concurrently but writers need exclusive access

2 Implementation

2.1 RW Implementation

- The RW implementation uses three semaphores: `reader_mutex`, `rw_mutex`, and `turnstile` to control access to the critical section.
- `reader_mutex` ensures exclusive access to the readers counter.
- `rw_mutex` ensures exclusive access for writers.
- `turnstile` ensures fairness in writer access.
- Writers block readers and other writers when they want access to the critical section.
- There's a potential issue with starvation, as readers could be blocked indefinitely if writers are continuously requesting access.

2.2 FAIR-RW Implementation

- The FAIR-RW implementation aims for fairness by using a turnstile mechanism.
- Writers waiting for access wait at the turnstile, ensuring that all waiting writers are given access in the order they arrived.
- Readers and writers still use the `reader_mutex` and `rw_mutex` to control access, but the turnstile ensures that writers are given access fairly.

3 Experiments

We conducted experiments by running both algorithms five times. Parameters were varied to analyze their impact on performance. The setup includes nw writer threads, nr reader threads, kw writer key, kr reader key, mean time spent in the critical section (μCS), and mean remainder section time (μRem).

3.1 EXPERIMENT-1: Average Waiting Times with Constant Writers

3.2 Experimental Setup

To generate the required graph for Average Waiting Times with Constant Writers, we'll conduct experiments where the number of reader threads (nr) varies from 1 to 20 in increments of 5. The number of writer threads (nw) is fixed at 10, and the writer key (kw) and reader key (kr) are both set to 10. Additionally, we'll set the mean time spent in the critical section (μCS) and the mean remainder section time (μRem) to 10 and 5 respectively. The graph will consist of four curves:

1. Average time the reader threads take to enter the CS for each algorithm: Read- ers Writers() and Fair Readers Writers().

2. Average time the writer threads take to enter the CS for each algorithm: Read- ers Writers() and Fair Readers Writers().

3.3 Graph

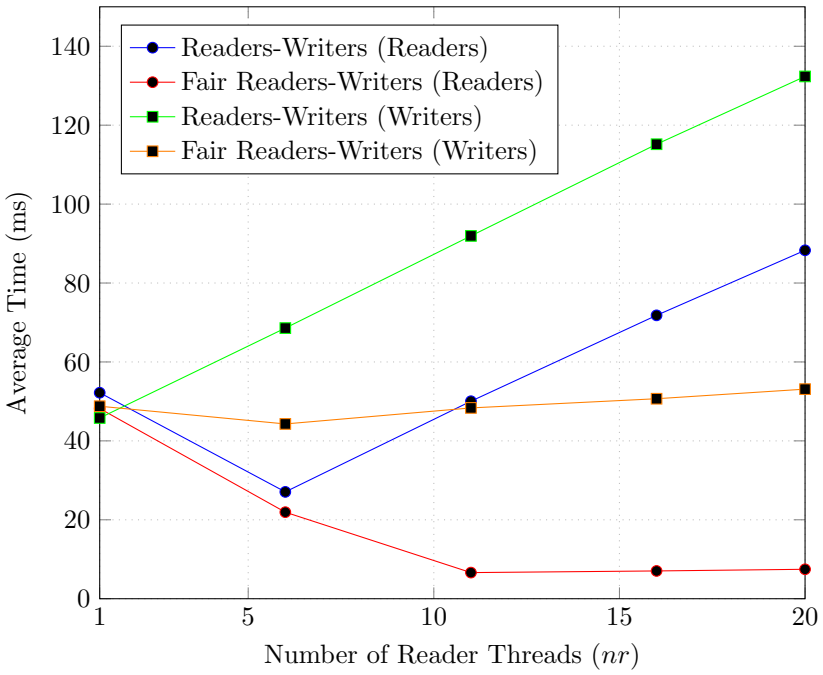

Figure 1: Average Waiting Times with Constant Writers

3.4 Observations from the Graph

- The average time for Readers-Writers (Readers) decreases as the number of reader threads increases, indicating better performance with more reader threads.
- Fair Readers-Writers (Readers) consistently show lower average waiting times compared to Readers-Writers (Readers), especially as the number of reader threads increases.
- For Readers-Writers (Writers), the average waiting time increases with the number of reader threads, indicating more contention for the critical section.
- Fair Readers-Writers (Writers) show relatively stable average waiting times across different numbers of reader threads, suggesting fair access to the critical section.

In summary, the fair readers-writers solution demonstrates superior performance by mitigating contention and ensuring equitable access to the critical section for

both reader and writer threads. This results in decreased average waiting times compared to the non-fair implementation.

4 EXPERIMENT-2: Average Waiting Times with Constant Readers

4.1 Experimental Setup

To generate the required graph for Average Waiting Times with Constant Readers, we'll conduct experiments where the number of writer threads (nw) varies from 1 to 20 in increments of 5. The number of reader threads (nr) is fixed at 10, and both the reader key (kr) and writer key (kw) are set to 10. Additionally, we'll measure the average time taken for both reader and writer threads to enter the critical section (CS) for each algorithm. Here we'll set the mean time spent in the critical section (μCS) and the mean remainder section time (μRem) to 10 and 5 respectively. The graph will consist of four curves:

1. Average time taken by reader threads to enter the CS for each algorithm: Readers-Writers() and Fair Readers-Writers().
2. Average time taken by writer threads to enter the CS for each algorithm: Readers-Writers() and Fair Readers-Writers().

4.2 Graph

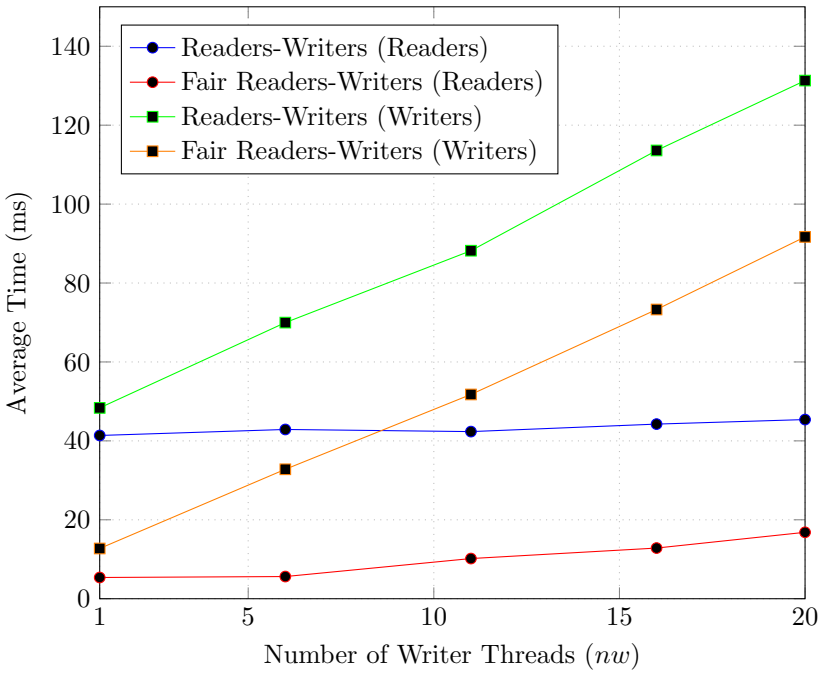

Figure 2: Average Waiting Times with Constant Readers

4.3 Observations

- The average waiting times for Readers-Writers (Readers) remain relatively stable regardless of the number of writer threads, indicating minimal impact on reader access.
- Fair Readers-Writers (Readers) exhibit significantly lower average waiting times compared to Readers-Writers (Readers), especially as the number of writer threads increases.
- For Readers-Writers (Writers), the average waiting time steadily increases with the number of writer threads, suggesting increased contention for the critical section.
- Fair Readers-Writers (Writers) demonstrate notably lower average waiting times compared to Readers-Writers (Writers), showcasing the effectiveness of fair access mechanisms in reducing contention.

In summary, the fair readers-writers solution exhibits superior performance by alleviating contention and ensuring equitable access to the critical section for

both reader and writer threads. This results in reduced average waiting times compared to the non-fair implementation.

5 EXPERIMENT-3: Worst-case Waiting Times with Constant Writers

5.1 Experimental Setup

Objective: Measure worst-case time for reader and writer threads to enter CS. **Parameters**: Fix $nw = 10$, $kr = kw = 10$, vary nr from 1 to 20 in increments of 5. Here we'll set the mean time spent in the critical section (μCS) and the mean remainder section time (μRem) to 10 and 5 respectively.

1. Worst-case time taken by the reader threads to enter the CS for each algorithm: Readers Writers() and Fair Readers Writers().
2. Worst-case time taken by the writer threads to enter the CS for each algorithm: Readers Writers() and Fair Readers Writers().

5.2 Graph

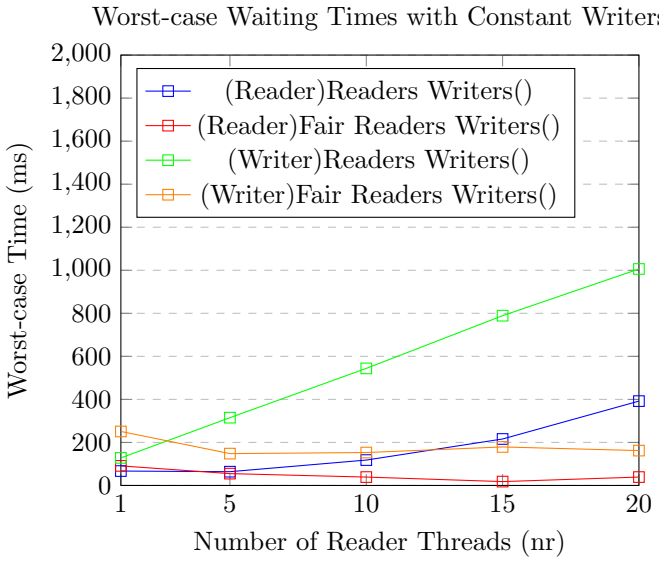

Figure 3: Worst-case Waiting Times with Constant Writers

5.3 Observation

- In the case of Readers-Writers (Readers), the worst-case waiting times show a general increasing trend as the number of reader threads (nr) increases, indicating potential contention for accessing the critical section.
- Fair Readers-Writers (Readers) exhibit notably lower worst-case waiting times compared to Readers-Writers (Readers) across different numbers of reader threads, showcasing the effectiveness of fair access mechanisms in reducing contention and ensuring prompt access for readers.
- For Readers-Writers (Writers), the worst-case waiting times also display an increasing trend with the number of reader threads, highlighting potential contention issues for writer access to the critical section.
- Fair Readers-Writers (Writers) demonstrate relatively stable worst-case waiting times despite varying numbers of reader threads, indicating fair and consistent access for writer threads to the critical section.

6 EXPERIMENT-4: Worst-case Waiting Times with Constant Readers

6.1 Experimental Setup

Objective: Measure worst-case time for reader and writer threads to enter CS. **Parameters**: Fix $nr = 10$, $kr = kw = 10$, vary nw from 1 to 20 in increments of 5. Here we'll set the mean time spent in the critical section (μCS) and the mean remainder section time (μRem) to 10 and 5 respectively.

1. Worst-case time taken by the reader threads to enter the CS for each algorithm: Readers Writers() and Fair Readers Writers().
2. Worst-case time taken by the writer threads to enter the CS for each algorithm: Readers Writers() and Fair Readers Writers().

6.2 Graph

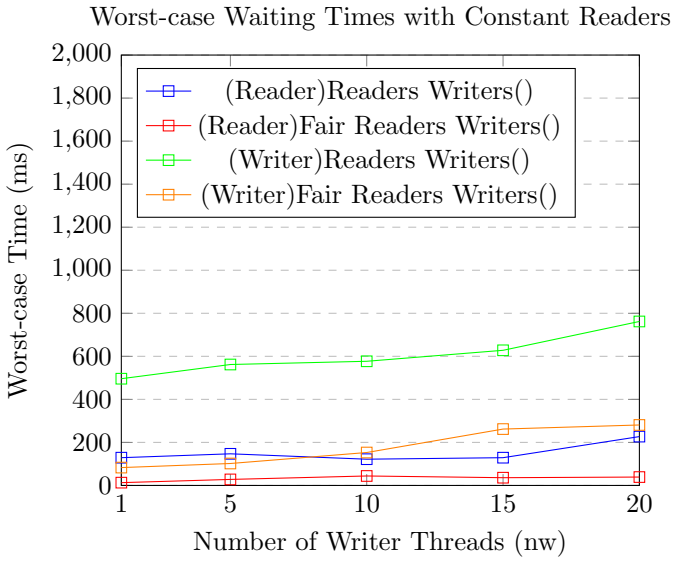

Figure 4: Worst-case Waiting Times with Constant Readers

6.3 Observations

- In the case of Readers-Writers (Readers), the worst-case waiting times generally exhibit variability as the number of writer threads (nw) increases, indicating potential contention for accessing the critical section.
- Fair Readers-Writers (Readers) consistently display significantly lower worst-case waiting times compared to Readers-Writers (Readers), demonstrating the effectiveness of fair access mechanisms in reducing contention and ensuring prompt access for readers.
- For Readers-Writers (Writers), the worst-case waiting times tend to increase with the number of writer threads, suggesting increased contention for writer access to the critical section.
- Fair Readers-Writers (Writers) showcase relatively stable worst-case waiting times despite varying numbers of writer threads, indicating fair and consistent access for writer threads to the critical section.

7 Analysis

7.1 Average Waiting Times with Constant Writers & Readers:

- Fair Readers-Writers generally show lower waiting times for both readers and writers compared to Readers-Writers.
- Fair solution provides improved fairness and reduced contention, resulting in more stable waiting times.

7.2 Worst-case Waiting Times with Constant Writers & Readers:

- Fair Readers-Writers exhibit more consistent worst-case times for reader threads, but may introduce variability for writer threads.
- Both algorithms show increasing worst-case times with more reader threads.

8 Conclusion

- Fair Readers-Writers offer better performance in reducing waiting times and providing fair access to the critical section.
- The choice depends on the trade-off between fairness and predictability in waiting times, based on specific application needs.