OS-2

Programming Assignment - 4 REPORT

CS22BTECH11056

Somalaraju Bhavya Shloka

INTRODUCTION:

Aim of this Assignment is to implement the solutions for Writer preference Readers-Writers problem and Fair Readers-Writers problem using semaphores in C++ and to observe the time taken by threads in each of the above solutions.

DESCRIPTION:

WRITERS Preference

- **lock1** , **lock2** semaphores are used to ensure mutual exclusion while updating writers and readers count respectively.

- **writerPrefer** semaphore ensures writer preference by allowing only one thread (writer) to access the critical section at a time. If a writer is waiting (i.e, when waiters == 1), new readers are blocked until the writer finishes.

- When a reader thread wants to enter the critical section: It waits on **readerhandle** semaphore to ensure no writer is currently accessing the critical section.
- It increments the **readers** count and checks if it's the first reader (i.e, when readers == 1). If so, it waits on **writerPrefer** semaphore to block writers from entering.
- After finishing reading, it decrements the readers count and releases the **writerPrefer** semaphore if no readers are left (i.e, when readers == 0).

- When a writer thread wants to enter the critical section: It waits on **lock1** semaphore to ensure mutual exclusion for updating **writers**.
- It increments the writers count and checks if it's the first writer. If so, it waits on **readerhandle** semaphore to block new readers from entering.
- It then waits on **writerPrefer** semaphore to ensure no other writer is currently accessing the critical section.
- After finishing writing, it decrements the writers count and releases the readerhandle semaphore to allow new readers.
- Average times are calculated by storing the average time of each thread in vectors.
- This solution does avoid starvation of writer threads but not reader threads. When there are many writer threads and less readers then the readers might starve until all the writers use the resource.

Fair Readers-Writers

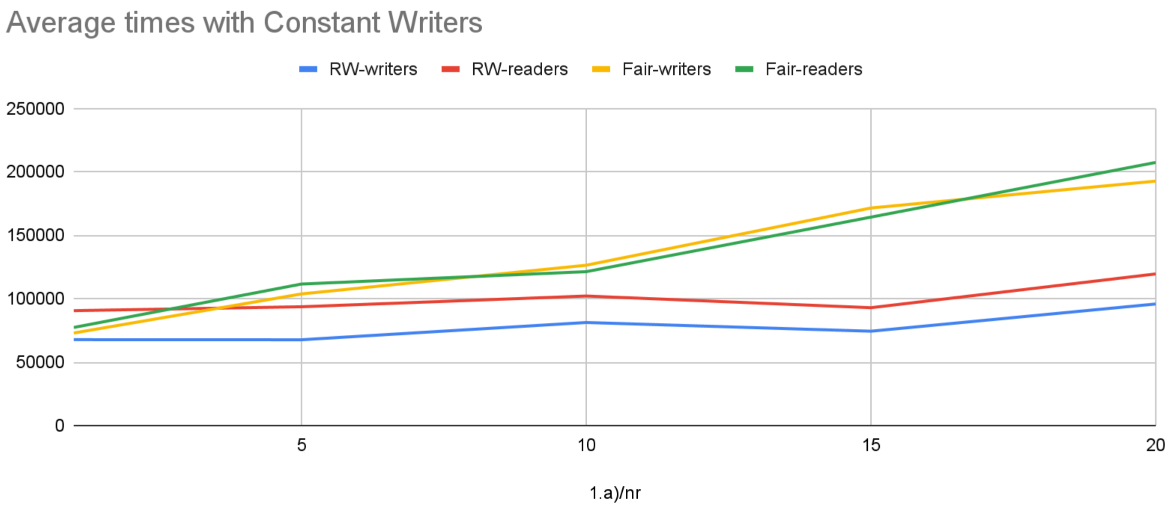
- The implementation ensures that both readers and writers are given fair access to the critical section without starving either. This is achieved through the use of the **waitQueue** semaphore.
- **Writers** are allowed to enter the critical section only when the **waitQueue is available**.
- Readers wait on the waitQueue semaphore before acquiring the lock1, ensuring that they have a fair chance to access the critical section without being blocked indefinitely.
- The **readers count** is updated atomically using **lock1**, ensuring correctness in the number of readers accessing the critical section simultaneously.
- The **resource** semaphore ensures mutual exclusion between writers, preventing concurrent writes to the critical section.
- Neither readers nor writers are starved of access to the critical section. Writers are given access only when no other readers are active, and readers are given access as soon as the **waitQueue** is available, ensuring fairness and preventing indefinite blocking.

- This ensures there is no starvation of both types of threads by serving them in the order of their arrival.

Experiments:

1.

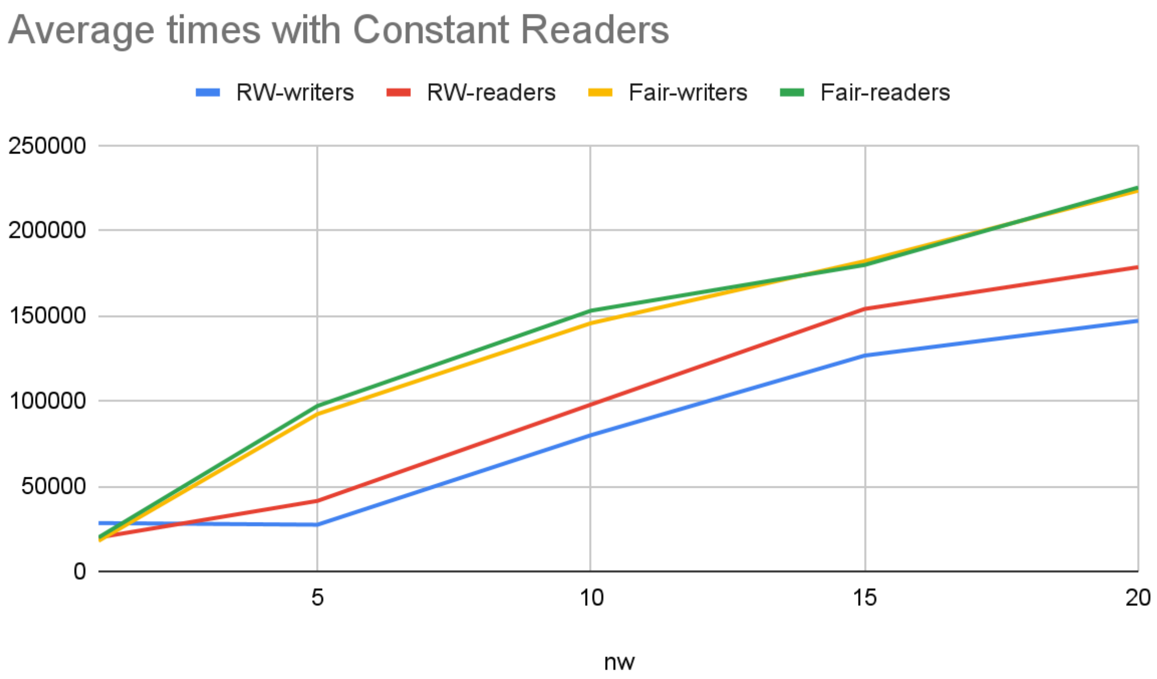
	RW		Fair	
nr	RW-writers	RW-readers	Fair-writers	Fair-readers
1	67739	90582	72951	77326
5	67645	93661	103771	111568
10	81246	102119	126395	121329
15	74383	92908	171517	164315
20	95836	119531	192697	207425



2.

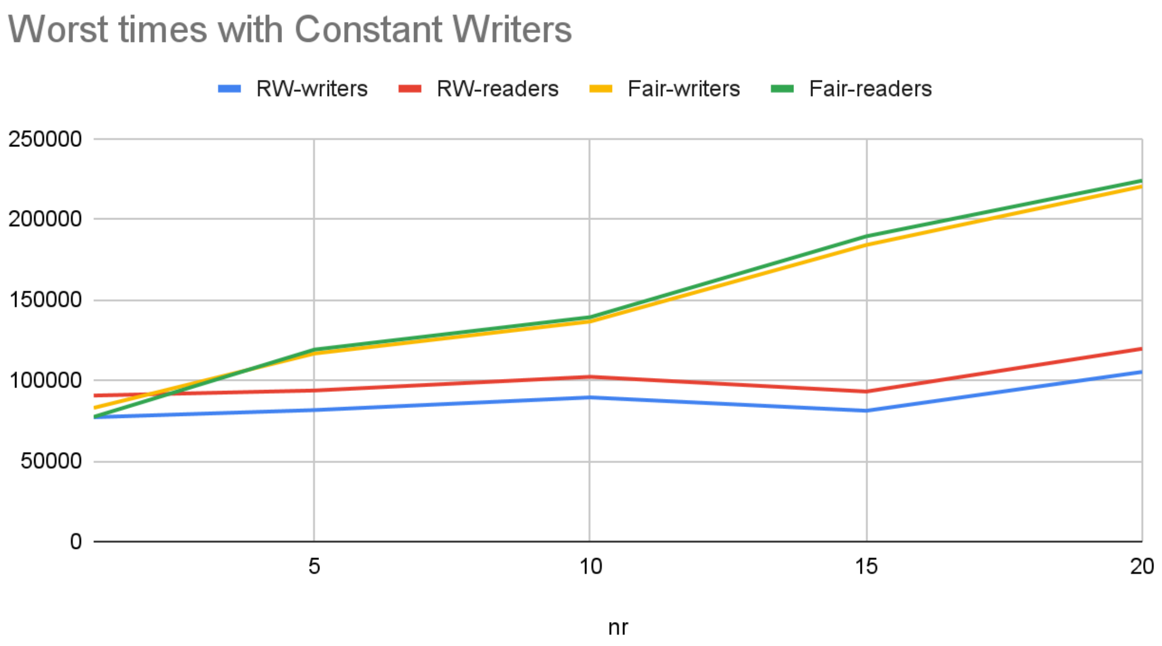
	RW		Fair	
nw	RW-writers	RW-readers	Fair-writers	Fair-readers
1	28324	19868	17906	19886
5	27354	41332	92252	97050
10	79907	97867	145625	152960

15	126588	153967	181984	179777
20	146989	178497	223384	225209



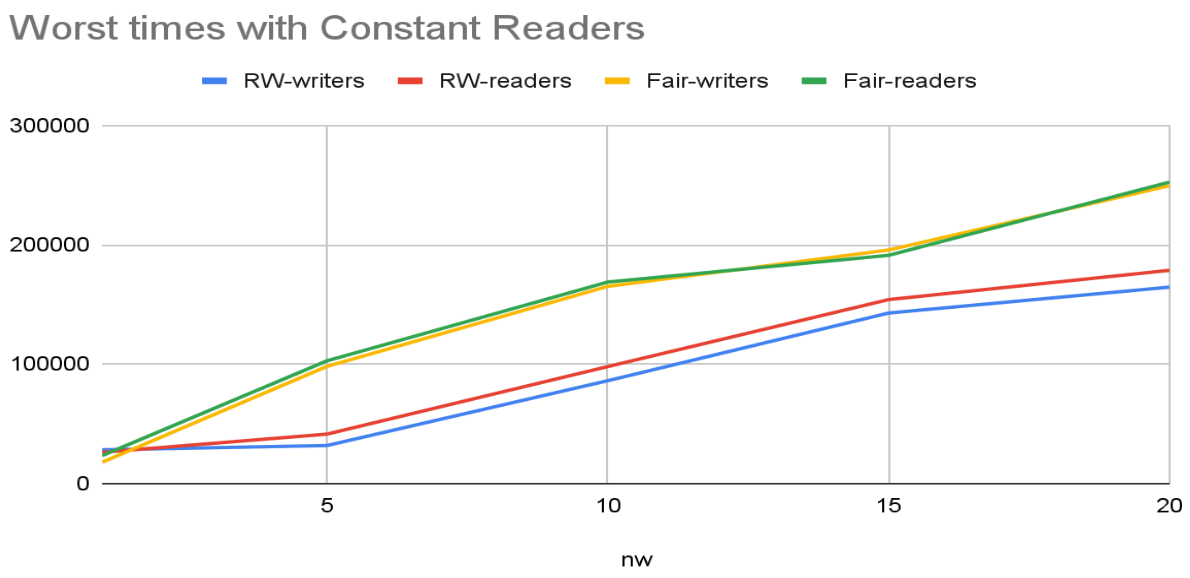
3.

	RW		Fair	
nr	**RW-writers**	**RW-readers**	**Fair-writers**	**Fair-readers**
1	77129	90582	82928	77326
5	81571	93707	116659	119098
10	89420	102207	136530	139225
15	81152	93060	184024	189396
20	105254	119638	220337	223952



4.

	RW-worst		Fair-worst	
nw	RW-writers	RW-readers	Fair-writers	Fair-readers
1	28324	26213	17906	23451
5	31878	41433	98123	102804
10	86164	97989	165179	168785
15	142833	154082	195583	191144
20	164519	178586	249464	252453



OBSERVATIONS:

We can see from all the above graphs that both the times of **readers and writers in the Fair READERS-WRITERS** solution are **almost same** in any condition whether reader threads or writer threads are varied.

And in all the graphs we can see that the writer's preference READERS-WRITERS solution is taking less time than the FAIR algorithm. And reader threads in the Writers preference solution are slightly taking more time than the reader threads.