Report

ES15BTECH11002

Design:

Unique ID is passed to each of the threads.

Localtime function is used to get the time.

usleep function is used to simulate the critical section and remainder section time , the random number is generated and moded with 500, to keep the sleep time limited to 500 usec.

The first algorithm which have a inherent problem of starving for writers is taken from the book.

The second algorithm which solves this problem was already discussed in the HW assignment.

Semaphores are used to implement the locking mechanism.

The Readers can access the critical section in any number as long as a writer is not presently accessing it, (Definition of reader writer problem) is taken care of.

RW_fair:

in semaphore is used to prevent starvation, as soon as a writer is available

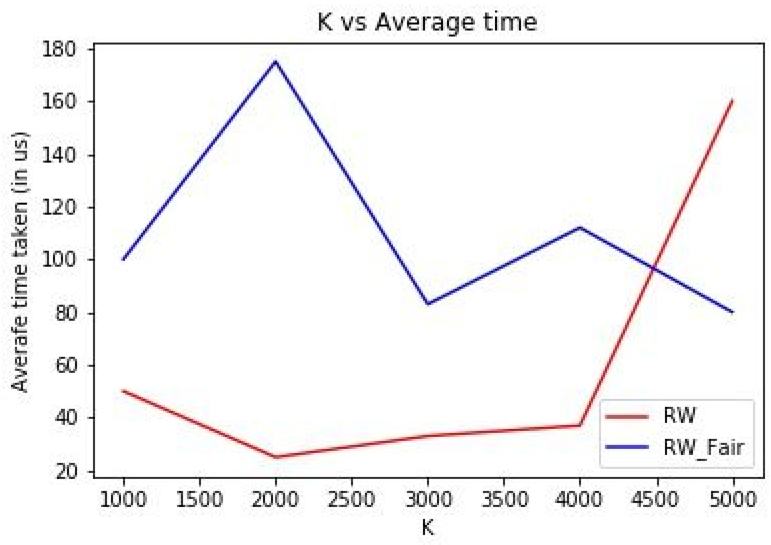
and the remaining processes are if available reader are in critical section, the
writer first calls a wait(in), thereby preventing any more readers to enter and
waits until all the readers(the count is kept using counter variable) complete the critical section. The last reader calls a signal(write), thereby allowing writer to enter the critical section.
After the writer completes the critical section it signals write and in, to let the remaining processes execute.
mutex semaphore is used to prevent race condition by the concurrent operation of two threads on counter variable

Difficulties:

Output comes jumbled sometimes, can be solved using fprintf() command which is atomic.

Graphs:

K vs Average Time



Analysis:

The implementation of reader writer with starvation being allowed for writers seems to perform better for low values of K.

Note the output has been taken by running 10 threads each for readers and writers and varying k by 1000 starting from 1000.

RW also gives a consistent performance for lower values of K.

RW fair on the other hand on average gives a inconsistent performance. RW_Fair may do better when there are a lot of writers compared to readers.