OS-2 Programming Assignment-4 Report

CS20BTECH11009

Input:

This programs takes the input as “inp-params.txt” in which it has n(no of threads), k (no of times) , lamda1 , lambda2

Code Design:

1) Irrescpective of type of the algorithm here most of the code remains the same. Only the entry section and exit section will be different.first we will see that.

2)We define the number of threads n and number of times k as global varibles and lambda1 , lambda2 in main function.we also define a vector of type double which is used for storing the waiting time for each iteration.

3) The program takes the input file “imp-params.txt” and n , k , lambda1, lambda2 are stored.

4) then we will call a function which will create the threads and we give the Function TESTCS as parameter for threads. And in the function itself the threads are joined and the threads were deleted.

5)And then in the main function, the average_waiting_time and maximum_waiting_time functions are called which takes vector as input.

6) Now we come to the function TESTCS. In that index , t1 , t2 are passed as argument. This t1, t2 are passedas arguments for sleep function.In this function a for loop is there which runs k times.In each loop, at the start the request time (time at which process request) is noted and then entry section is called and now after the entry section the entry time (the time at which it enters the critical section) is noted .

7) Then the using wait_times vector , the waiting time for each iteration is pushed into the vector. Now the CPU will sleep for t1 /1000 ms. Which here is considered as critical section. And now exit section is called which follows by noting the exit time of the loop.Now we have the entry and exit sections of Cas , Tas , Cas-bounded.

8)

TAS:

Entry section: we use the atomic_flag varible and call the test_and_set() function.Here the main purpose of this atomic variables it is not affected by others.

Exit section: we use clear() function to implement Exit section.

CAS:

Entry section: we use the atomic_bool variable. It is intialised as false initially.using the atomic_compare_exchange_strong function the entry section is implememted.

Exit section : The atomic_bool variable is made false.Here also we use atomic varibale as it is affected by others.

CAS-Bounded:

Entry section: Its an modified version of CAS. Here a pointer of type bool which is dynamically alocated in main() is declared globally.so using atomic_compare_exchange_strong function and the status of each process and changes in it the Entry section is implemented.

Exit section: Also including the status of each process and checking it with index the lock is made false.Thus Exit section is done.

Output:

1) For each it will give average waiting time , maximum waiting time in terminal

2) For each terminal it will generate a txt file in which it has information about at what time it has entered and and at what request has made , and at what time it has exited the system.

Graphs:

The values for k = 10 , lambda1 = 1 , lambda2 = 5. Here we keep these parameters constant and vary the number of threads from 10 to 50. The values in the graph we got by taking the average of 5 times for particular number of threads.

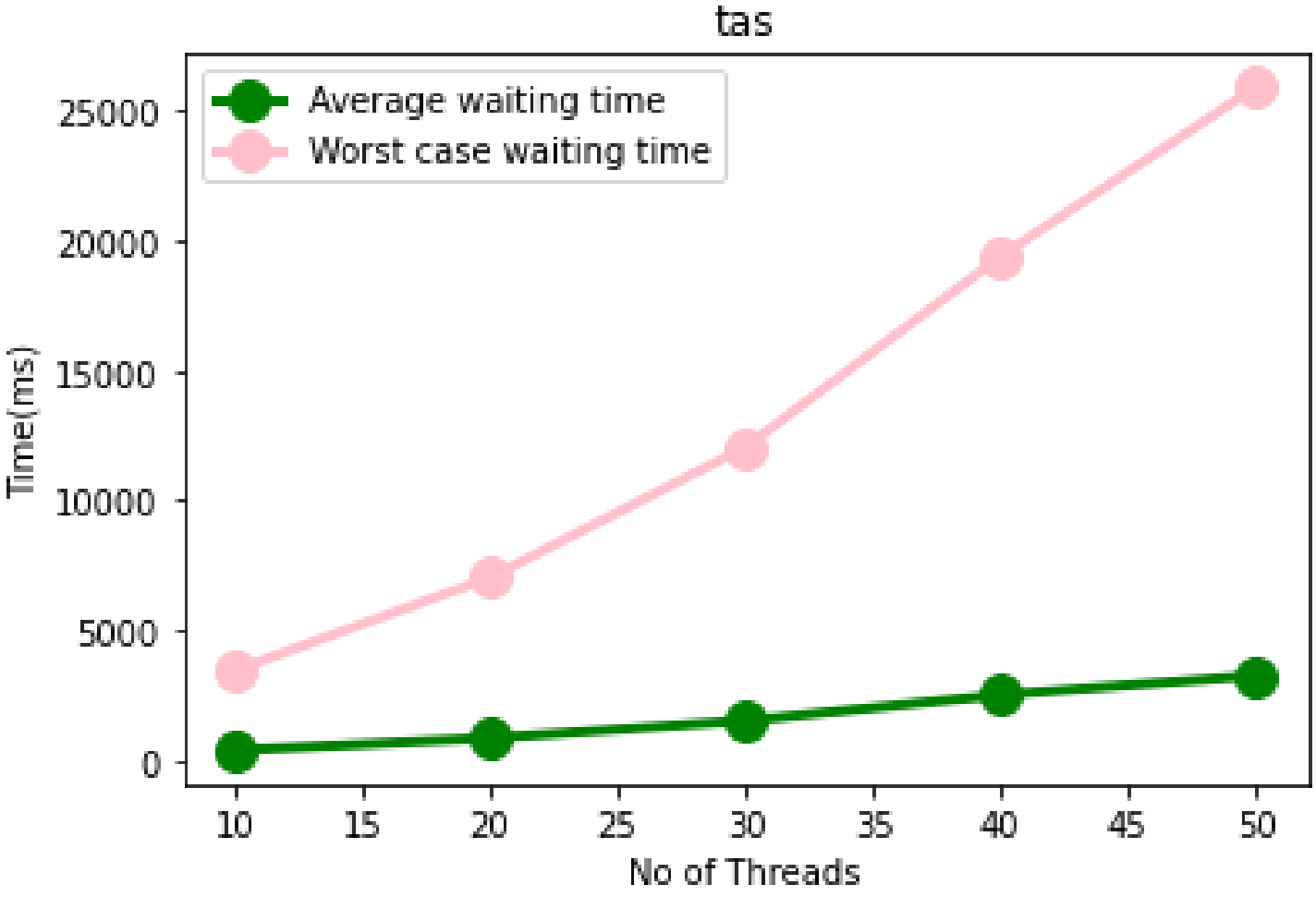

Fig1 : TAS

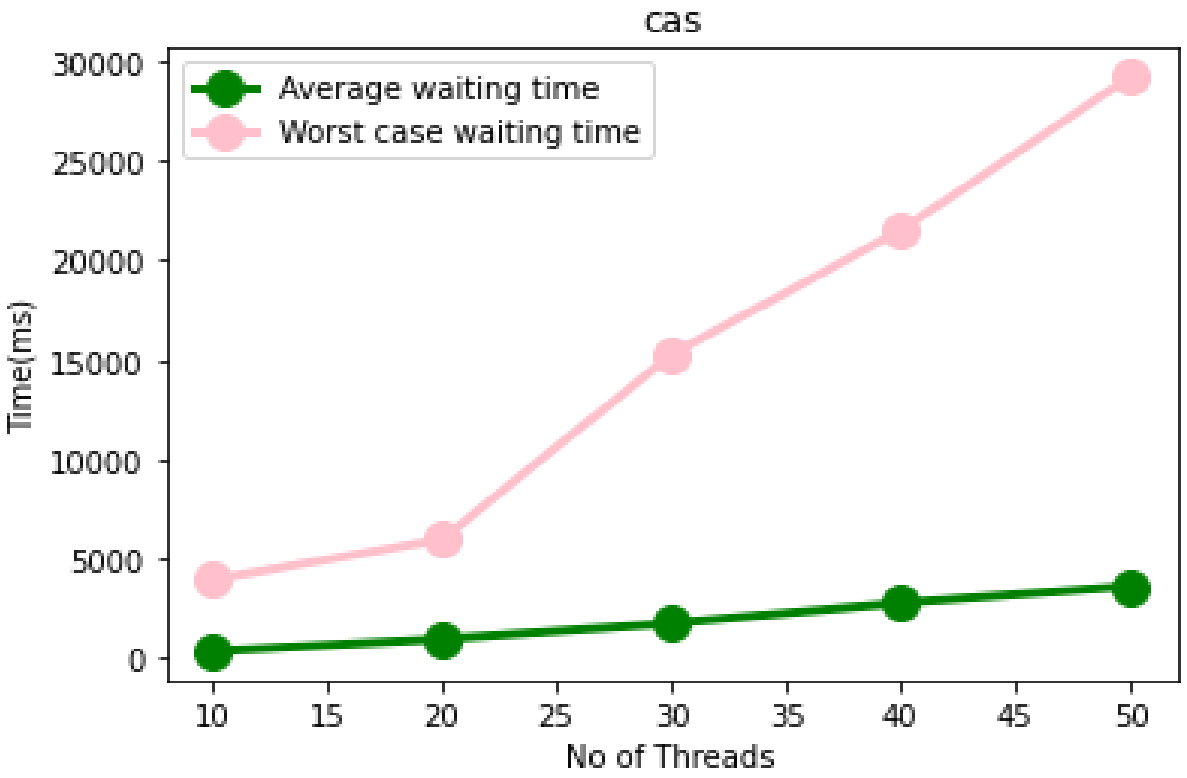

Fig2: CAS

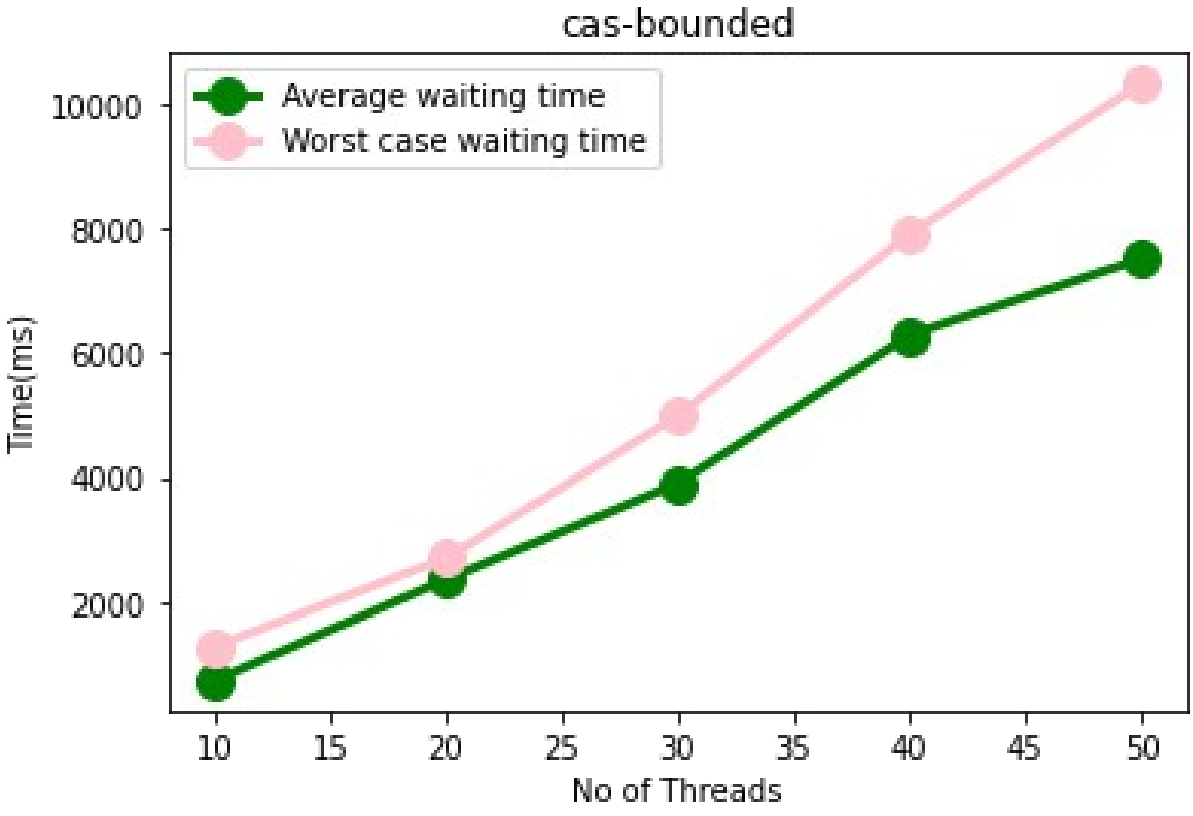

Fig3: CAS-Bounded

From these 3 graphs we can see that

1) As the value of n increases, the number of threads increases, resulting in an increase in a loop that runs k times, resulting in greater waiting time and also the worst case waiting time. As the value of n rises, the graph rises with it.

2)Cas-bounded has a longer average waiting time than Cas and Tas because it has more while loops and waiting in it, and it has more variables to manage. It additionally checks for boundedness.The average waiting time for Cas and Tas are almost same..because both only checks for mutual exclusion. The Tas and Cas are more or less the same.

3) Cas-bounded has lower worst case waiting time because the main purpose of cas-bounded is to avoid the continous waiting of many process eventhough they are ready to execute. So the waiting times of all the processes is decresed resulting in a decrease in the worst case waiting time when compared to Tas and Cas.

4) Cas and Tas have more or less the same average waiting time and worst case waiting time.

5) we can also observe that in Cas and Tas there is not much increase in average waiting but a high increase in worst case waiting time

6)But whereas in Cas-bounded the speed of rise in both average waiting time and the worst case waiting time is same. This tells us that almost worst case waiting time and the average case waiting time is not too much. Which also tells that Cas-bounded has ensured that almost all the processses waiting time difference is not too much as compared to Cas and Tas.