Assignment - 4 : REPORT

Operating Systems - II : CS3523

CS20BTECH11037, PATNALA VIKAS

Algorithm

- Input will be taken by the main thread and this main thread. And then we will store that input variables globally because every thread should have to access those variables.
- After that we will declare n threads and then we will create those n threads and run the function $testCS$ in these threads and we will pass the id of the thread that thread which is equal to $i+1$
- After that each thread will call the function separately and then here our logic starts
- Here we will restore the value of the thread id and use that value to identify each thread
- $default\_random\_engine$ is used to generate random number from $exponential\_distribution$ to simulate the CRITICAL and REMAINDER section. Parameter for both critical and remainder section were $\frac{1}{\lambda_1}$ and $\frac{1}{\lambda_2}$ respectively.
- After that we will run each thread Critical section k times as given in the question
- Here every time if a process request the CS then we will make use of the function $getSysTime$ which takes the $time\_t$ variable and converts it to our required format
- After that in each ENTRY SECTION we will have some mechanism to stop every process except one to satisfy mutual exclusion property corresponding to the each algorithm TAS, CAS, CAS-Bounded we will has different locking mechanisms for different algorithms
- Here we will output to the file using $fprintf$ because by using this function we will not get the outputs which are mixed for example if two threads request the

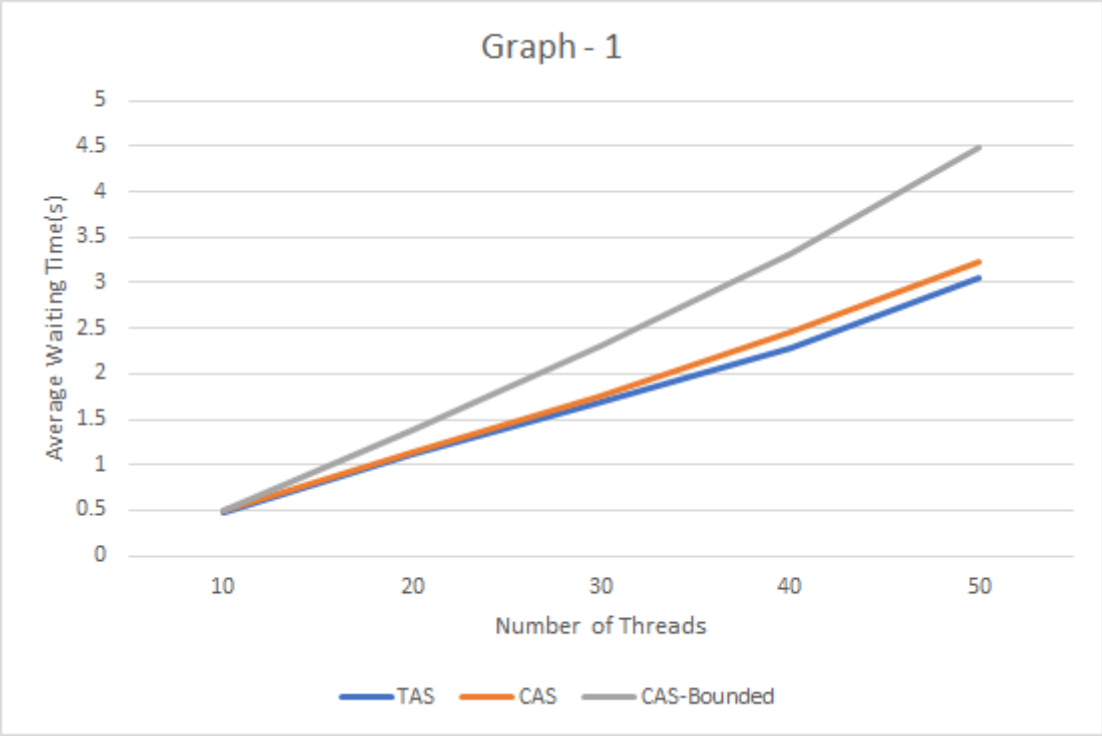
CS at the same then we may have a chance of getting the values mixed using $fprintf$ will not mix the values instead output them individually

- For TAS algorithm we will use the $test\_and\_set()$ predefined atomic function to achieve our mutual exclusion requirement
- For CAS algorithm we will use the atomic variable $lock$ which is initialized to 0 and then the first process coming to the while loop will set the lock to 1 and other process will have to wait there until the lock is again set to 0 and hence here also we will meet mutual exclusion property
- For CAS-Bounded algorithm we will use atomic variable $lock$ and array $waiting[]$ to achieve mutual exclusion and bounded waiting here the process entering will set lock the other process from entering CS using key variable. After that if a process finishes its CS it will check whether any process is waiting for CS in sequential order. A new process will enter the CS if and only if variable key == 0 or waiting is false
- In each of the above algorithms we will keep the thread which is executing the CS to sleep for certain amount of time which we will get using the random generator. After that every process will have to sleep for another certain amount of time when leaving the REMAINDER SECTION
- After that the control comes to the main function and then the main function will wait until the threads complete their execution

Graphs

In these graphs each point of the graph is obtained by averaging over five runs. keeping the value of k to be 10 and varying the number of thread from 10 to 50 and taking the values of $\lambda_1 = 0.1$ and $\lambda_2 = 0.2$

Average waiting time (vs) Number of Threads

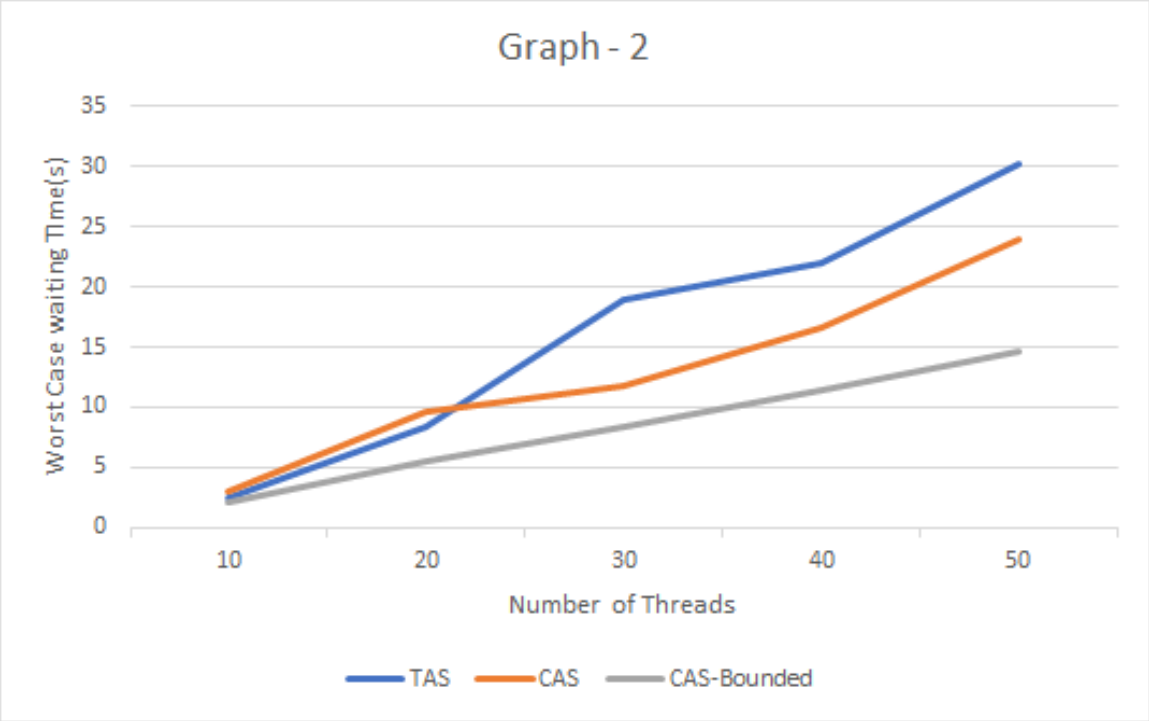


Analysis of this graph between Average Waiting Time and Number of Threads

- Here we can see that the average waiting time TAS and CAS are almost same and values of CAS are slightly larger than that of the value of TAS because as in the case of CAS it has to execute instruction than the TAS instructions and hence we can observe that average waiting time of CAS is slightly larger than average waiting time of TAS
- And here we can see that CAS-Bounded is having the largest average waiting time than others because here every thread have to waiting until the previous threads surroundings will complete for example if threads 2, 3, 4, 5, 6, 7, 8, 9 request CS and thread 2 comes first and executed in CS in between if thread 1 request a CS then after the execution of thread 2 it will check if any thread 3, 4,

5, 6, 7, 8, 9 is waiting and if waiting then we can see that thread 1 will wait until the other threads complete so hence here we can see that average waiting time is high

Worst case time (vs) Number of Threads



Analysis of this graph between Worst Case waiting time and Number of Threads

- Here we can see that worst case waiting time for both TAS and CAS are much higher than that of the CAS-Bounded because we can see that TAS and CAS will not give preference to bounded waiting
- And we can see that CAS-Bounded perform better in case of worst case waiting time as it will give thread almost equal chance to enter the CS and thus avoids starvation on one process which was not taken care in TAS and CAS

Conclusion

- In case of average waiting time TAS performs best over all other mutual exclusion algorithms and CAS-Bounded performs worst over all other mutual

exclusion algorithms

- Where as in the case of worst case waiting time of a process CAS-Bounded performs best among all the Mutual exclusion algorithms and TAS perform worst in the worst case waiting time of a process.