Operating Systems 2
Name :- Ajinkya Bokade
Roll No :- CS17BTECH11001

Aim

To implement TAS, CAS and Bounded Waiting with CAS mutual exclusion (ME) algorithms and calculate average waiting time taken by a process to enter the critical section and worst case time taken by a process to enter critical section.

Design of Program

The input parameters to the program are n (Number of threads), k (Number of times each thread enter the critical section), λ1 (Average for exponential distribution of delay value t1), λ2 (Average for exponential distribution of delay value t2). t1 is delay value for simulation of critical section and t2 is delay value for simulation of remainder section.
All these parameters are given through a text file named 'inp-params.txt' to each of the three programs.

Entry Section is the part of code where the thread makes a request to enter critical section.
Critical Section is the part of code which can be executed by at most one thread due to presence of shared variables.
Exit Section is the part of code where thread completes
Waiting time is the time difference between the time when thread just starts to execute the critical section and the time when it made a request to enter critical section.
Bounded waiting time guarantees that no thread have to wait forever to enter critical section.

All the three techniques have a similar design except the implementation of lock. First in the main function, n threads are created (pthreads) and each thread calls a function named testCS. In the testCS function, a for loop runs for k iterations since each thread tries to enter the Critical section k times. In each iteration, time at which each thread enters is printed to a file using printf.

Then time when thread which has just entered critical section is printed to file and thread is allowed to sleep for a time calculated by exponential_distribution to simulate the computational time spent in critical section. Then time at which thread completes the critical section is printed to file and then that thread is allowed to sleep for a time calculated by exponential distribution to simulate the time spent in remainder section. Waiting time is calculated by the time difference between the time when the thread accesses critical section and the time at which threads had made request to enter to critical section.

Implementation of Test and Set technique

For test and set technique, we have used C++ atomic_flag for lock (included in <atomic> header) and initialized it to ATOMIC_FLAG_INIT which is used to to make value of lock false. Then we have used C++ atomic_test_and_set_explicit function which changes the lock value to true and returns old value of lock. As a result disabling the access of other threads to critical section. After the thread has executed critical section, we have used atomic_flag_clear_explicit function which makes value of lock to false thus enabling other threads to access the critical section.

Implementation of Compare and Swap technique

For Comapare and swap technique, we have initalized lock variable which is atomic to 0. We have set the expectedValue of lock to 0. Thus after the entry section, we have used C++ compare_exchange_weak function (lock.compare_exchange_weak(expectedLockValue, 1)) to check if value of lock is equal to expected value. If it is, then lock value is set to 1 and function returns old value of lock. As a result other threads can access the critical section at the same time.
After the thread has executed critical section we have set lock value to 0 so that other threads can access critical section.

Implementation of Bounded Compare and Swap technqiue

This is same as Compare and Swap technique except that here bounded waiting requirement is followed, that is any thread wishing to enter critical section must do so within n−1 turns.
A boolean array waitingQueue is created which denotes the status of each thread. Bounded waiting is ensured by checking after each thread have executed critical section, if any thread among the remaining n−1 threads whose waiting status is true is set to false, as a result that thread gets entry to critical section. If waiting status of no thread is true, then lock is set to 0. Since here bounded waiting requirement is followed, worst case time for any thread to enter critical section would be less as compared to other two techniques.

Comparison between TAS and CAS and Bounded CAS

Graphs

Input parameters :− Number of threads n = 10, 20, 30, 40, 50
Number of times each thread enters CS k = 10 (kept same for all test cases)
Lambda for delay t1 = 2 (same for all test cases)
Lambda for delay t2 = 2 (same for all test cases)

Average values for TAS

n	Average time taken to enter the CS (in seconds)	Worst case time taken by a process to enter the CS (in seconds)
10	9.26	94
20	19.71	162
30	30.21	272
40	41.0275	325
50	52.06	390

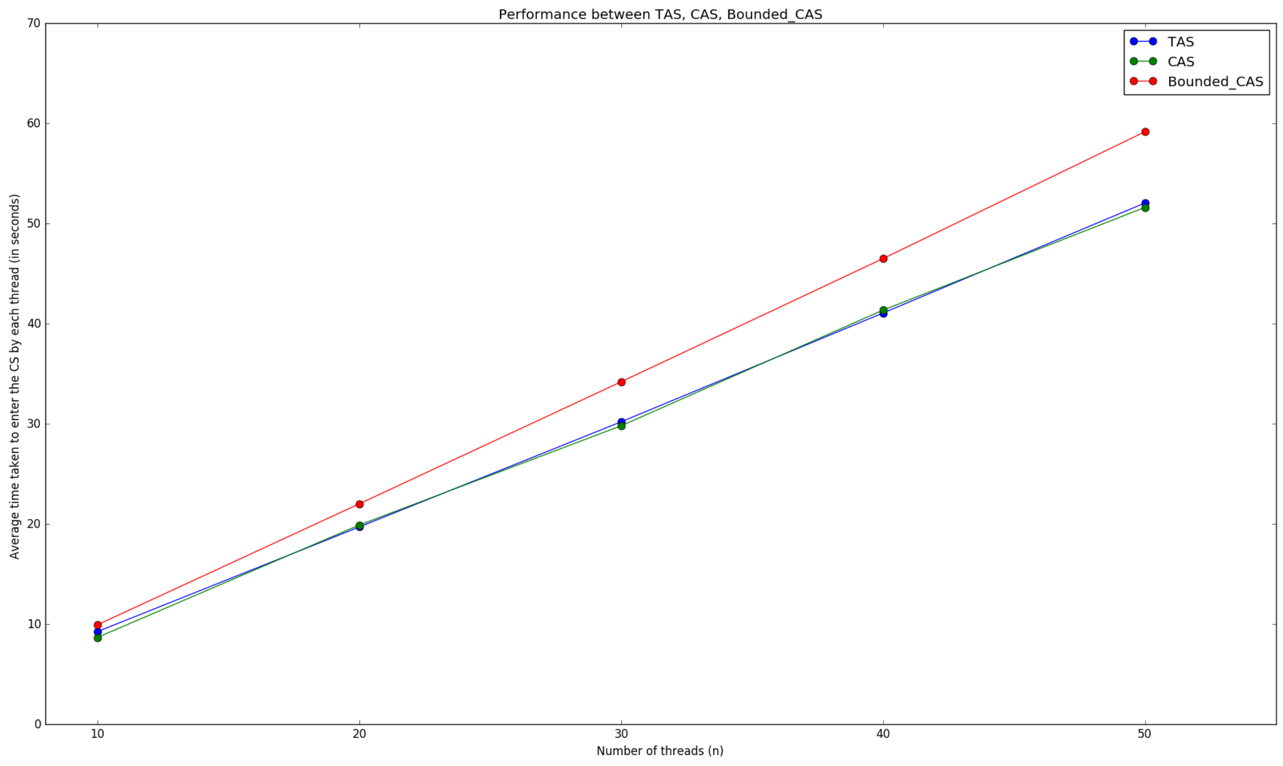
Average values for CAS

n	Average time taken to enter the CS (in seconds)	Worst case time taken by a process to enter the CS (in seconds)
10	8.68	53
20	19.905	137
30	29.8133	239
40	41.3625	304
50	51.62	326

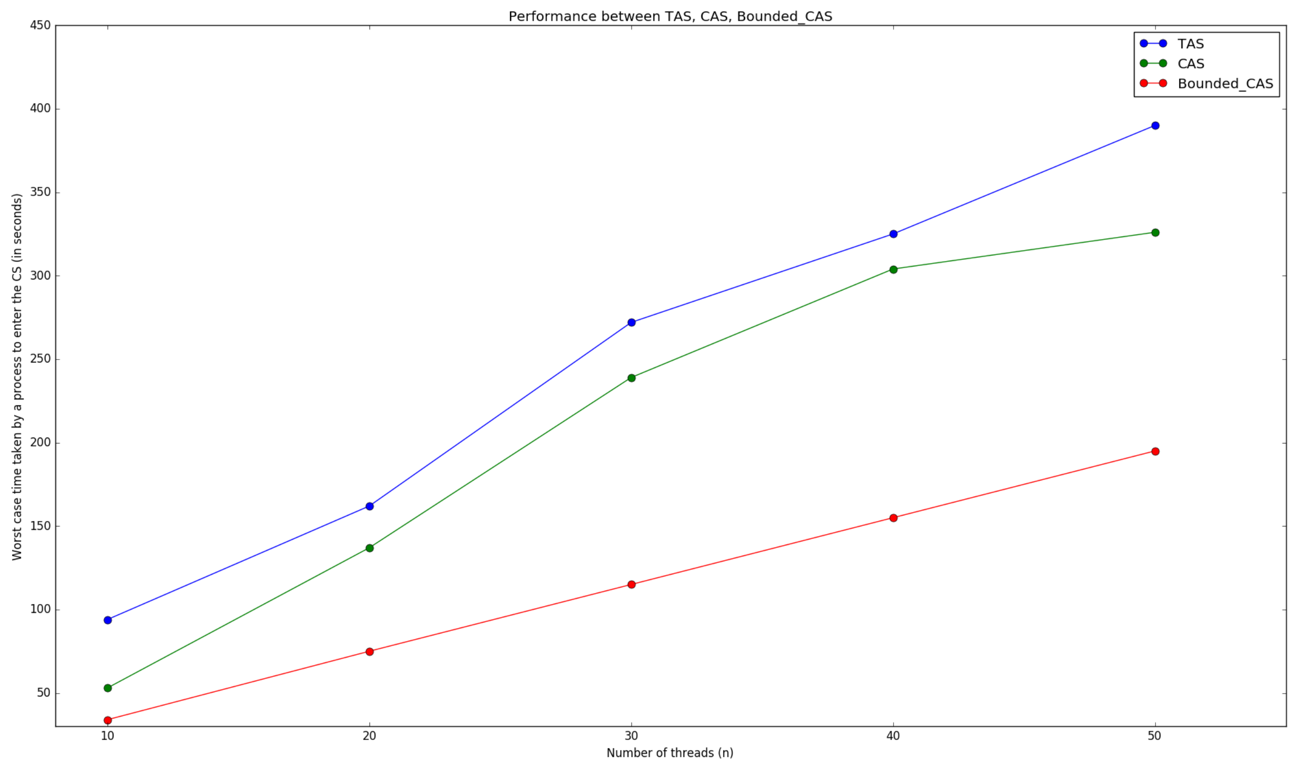
Average values for Bounded CAS

n	Average time taken to enter the CS (in seconds)	Worst case time taken by a process to enter the CS (in seconds)
10	9.94	34
20	22.02	75
30	34.206	115
40	46.507	155
50	59.17	195

Average time taken to enter the CS by each thread (in seconds) vs Number of threads (n)



Worst case time taken by a process to enter the CS (in seconds) vs Number of threads (n)



Conclusion

We observe that average waiting time for each thread to enter Critical Section is almost similar for Test and Set and Compare and Swap while for Bounded Compare and Swap, it is higher than both. Average waiting time is more in case of Bounded Compare and Swap since it runs a loop for checking a thread which is waiting, thus increasing waiting time.

We observe that worst time taken by process to enter critical section is least for Bounded Compare and Swap since it runs a loop to check which thread is waiting to enter critical section thus solving starvation problem, that is no thread has to wait long to get access to critical section. While in Test and Set and Compare and Swap, it is more in case of Test and Set.