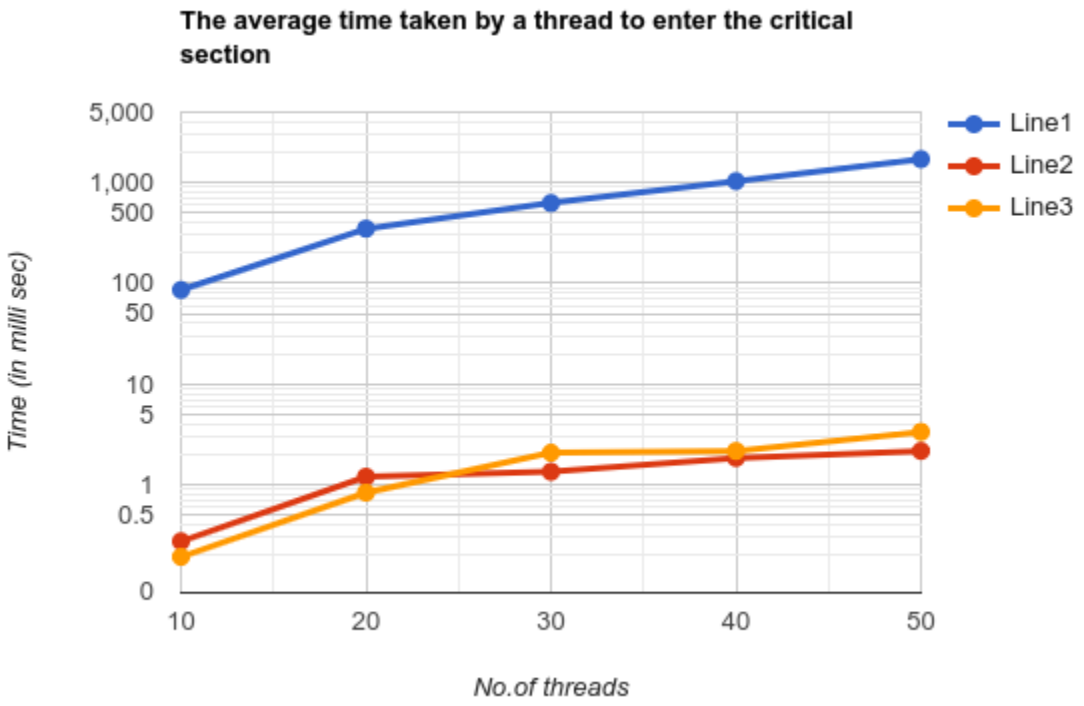
REPORT

COMPARISON OF THE PERFORMANCE OF TAS, CAS, AND BOUNDED CAS ME ALGORITHMS

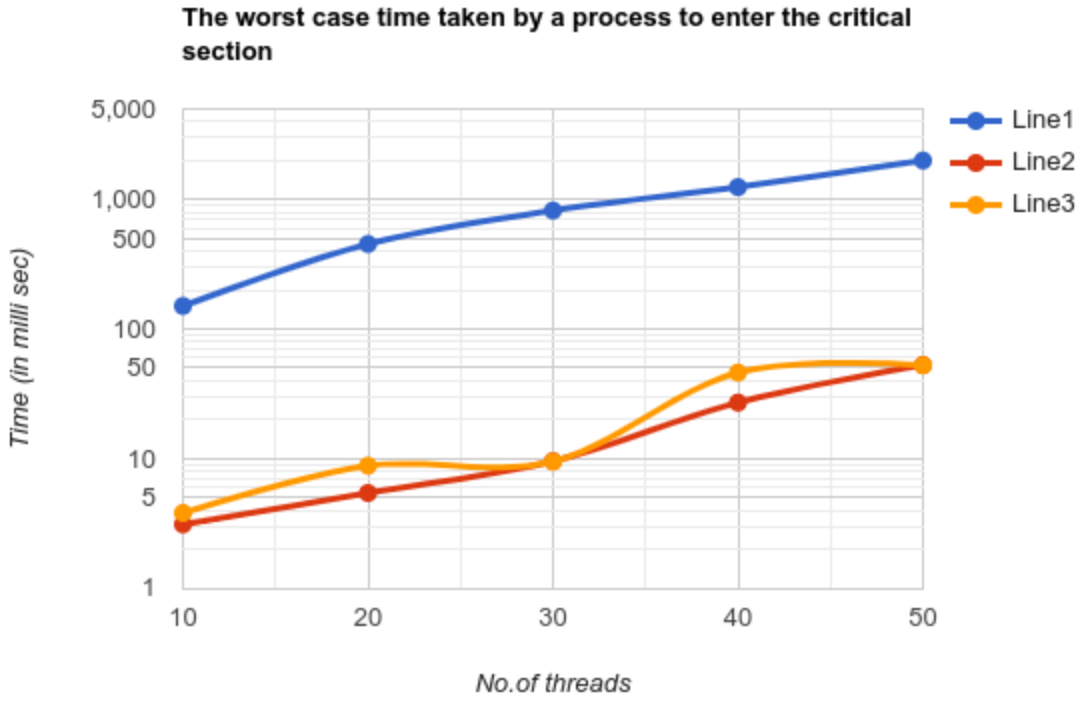
The three algorithms :- Test and Set , Compare and Swap and Bounded Waiting with Compare and swap along with mutual exclusion , are the hardware instructions which help us solve the critical-section problem in a relatively simple manner. Here's a look at the comparison between the three algorithms.

Plot 1



Line1 - Test and Set algorithm (tas)
Line2 - Compare and Swap algorithm (cas)
Line 3 - Bounded Waiting with Compare and swap along with mutual exclusion (cas_me)
(Note: Same convention in the following graph.)

Plot 2



Conclusion : The time taken by a process to enter in the critical section is highest in tas algorithm followed by cas_me and cas algorithms (quite similar). Let's understand as to why there is a considerable difference in the time taken.

Tas algorithm works in this way - The first process will enter the critical section at once as TestAndSet(lock) will return false and it'll break out of the while loop. The other processes cannot enter now as lock is set to true and so the while loop continues to be true. Once the first process gets out of the critical section, lock is changed to false. So, now the other processes can enter one by one. However, after the first process, any process can go in. There is no queue maintained, so any new process that finds the lock to be false again can enter.
And thus mutual exclusion and progress is ensured while bounded waiting is not ensured.
So when a processor has obtained a lock, all other processors which also wish to obtain the same lock keep trying to obtain the lock by initiating bus transactions repeatedly until they get hold of the lock. This increases the bus-traffic requirement significantly. It slows down the overall section of code.

While on the other hand compare and swap algorithm sets the key to true and then swaps with lock. But due to hardware implementation and the way the code is written, it helps us achieve a much lesser waiting time than test and set algorithm.

But Bounded Waiting with Compare and swap along with mutual exclusion algorithm adds waiting[i] to each process and checks whether or not a process has been waiting. It maintains a

ready queue with respect to the process in the critical section. Once the ith process gets out of the critical section, it does not turn lock to false so that any process can avail the critical section now, which was the problem with the previous algorithms. Instead, it checks if there is any process waiting in the queue. If there is no process waiting then the lock value is changed to false and any process which comes next can enter the critical section. If there is, then that process' waiting value is turned to false, so that the first while loop becomes false and it can enter the critical section. That's how it ensures bounded waiting and hence the average waiting time is quite lesser as compared to the other algorithms.

Source :
https://www.geeksforgeeks.org/hardware-synchronization-algorithms-unlock-and-lock-test-and-set-swap/
https://en.wikipedia.org/wiki/Test-and-set
My code (for graphs).