Implementing TAS, CAS, Bounded Waiting CAS

Topic - Implementing 3 techniques of solving Critical section problem using atomic instructions and analyzing their performance.

Submitted By: Kushagra Indurkhya
CS19B1017

Contents

Introduction

All the methods are implemented in c++ and make use of the <atomic> header to make sure that the instruction runs atomically on the hardware.

Comparing Performances

For comparing the performances of the three methods I am comparing:

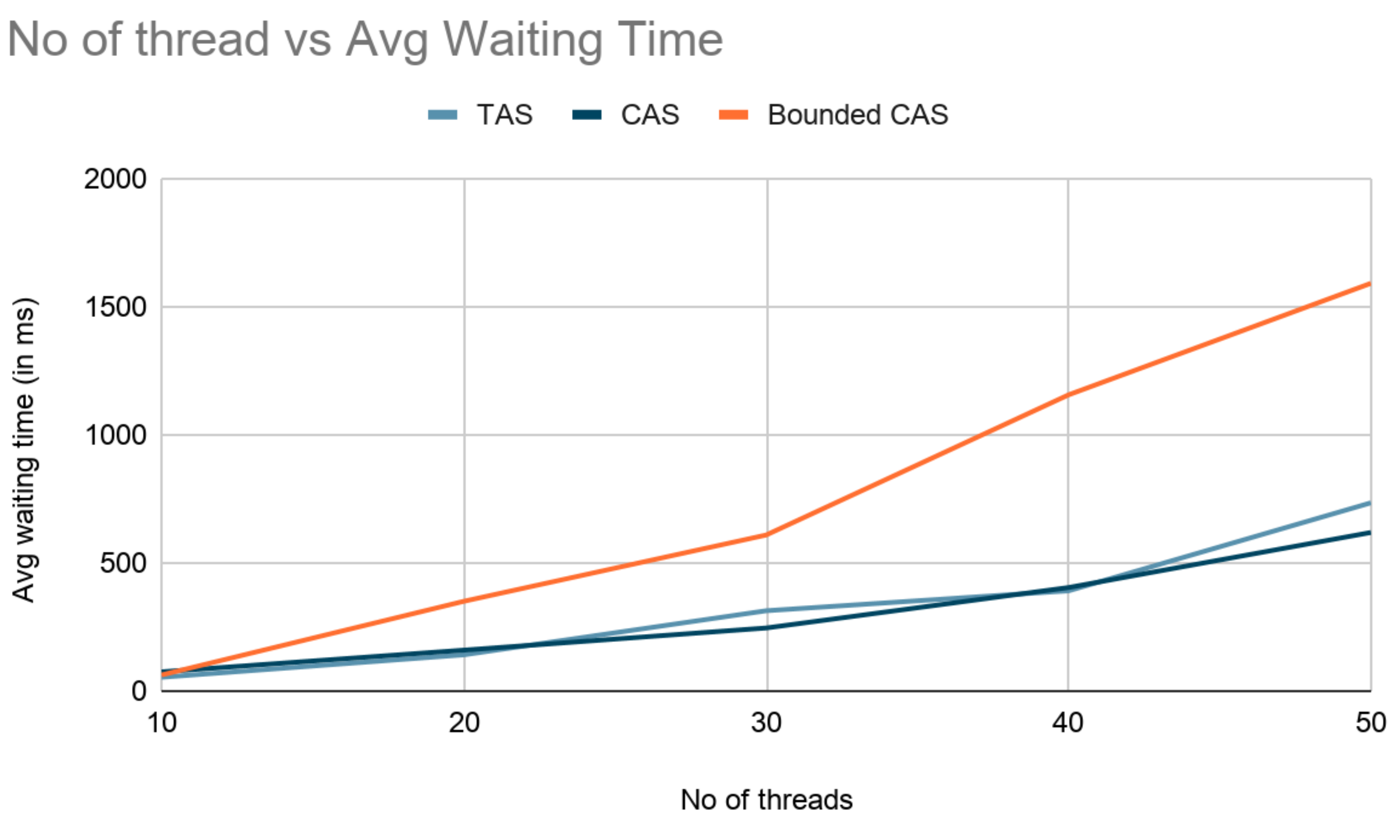
- the average waiting time - Average of Time taken between the request and start of execution of the critical section of all processes.

$$\Sigma(EntryTime\ in\ CS\ -\ TimeRequestedAt)/(n * k)$$

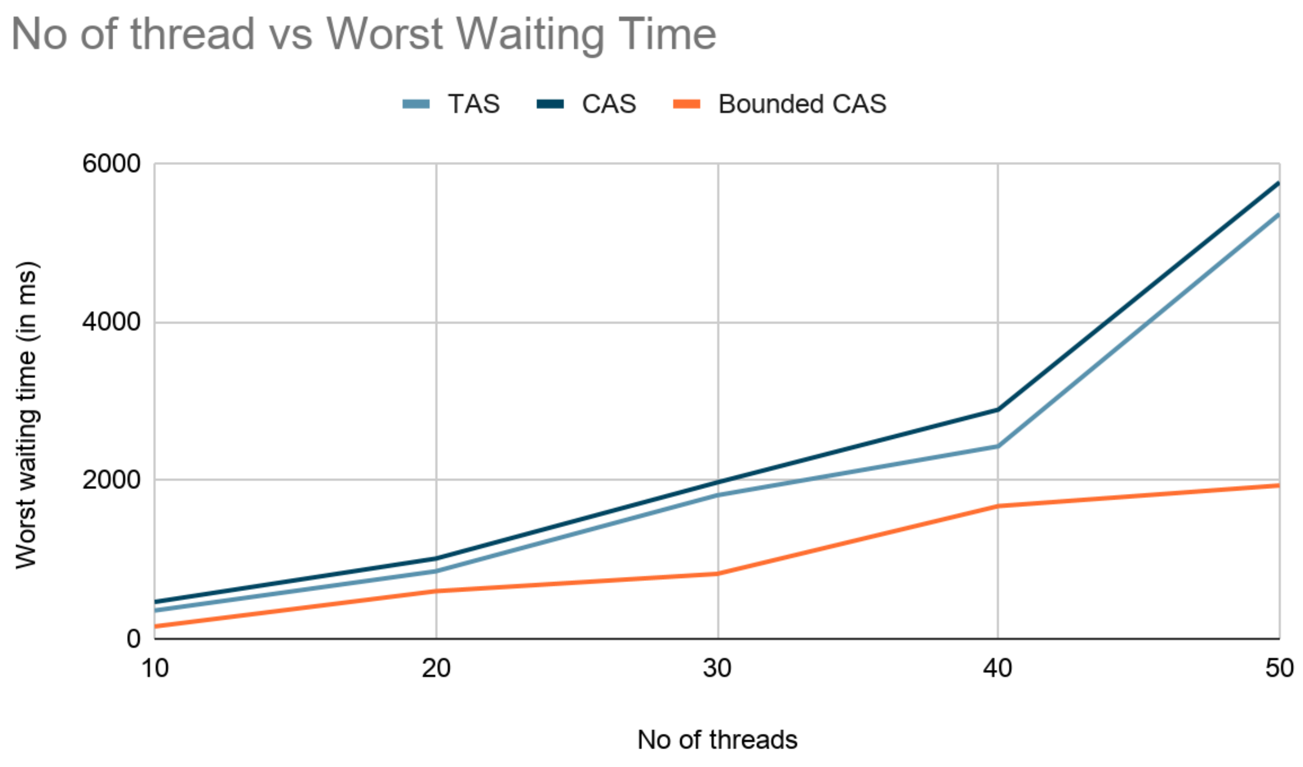
- The maximum (worst) waiting time of every process.

Keeping k=10,λ1= 5,λ2= 10 and varying number of threads from 10 to 50

No of threads vs Avg Waiting Time Graph



No of threads vs Worst Waiting Time Graph



Analysis

From the graphs following observations can be made:

- On increasing the number of threads both Avg waiting time and Worst waiting time are increasing for all three methods which is obvious as more the number of processes more toll on the resources.
- TAS and CAS are comparable in both Avg waiting time and Worst waiting time but TAS gives a slightly better performance in terms of worst waiting time.
- Bounded CAS gives better performance in terms of worst waiting time as it prevents but it lacks in Avg waiting time. It was expected as bounded CAS satisfies bounded waiting(ie prevents a process from being denied entry in critical section again and again as any process takes at most n-1 turns to enter CS) this causes an increase in average waiting time as processes that could have been executed but had to wait for a critical section of older processes to be executed first to satisfy bounded waiting.
- On observing the logs on various inputs, mutual exclusion was satisfied, and bounded waiting was satisfied in bounded CAS.

Conclusion

- Bounded CAS minimizes the worst waiting time, but gives a large avg waiting time.
- CAS has less average waiting time but a large worst waiting time.

Some design details

- Random numbers are generated seeded by epoch time and fed into the distribution average of which was provided in the input to get the simulated time taken by critical and remaining section.
- I have made an additional file logger.cpp containing the common utilities for logging included in all 3 programs.
- When a logger object is created, in the constructor current time is calculated using epoch time and stored.
- Each thread receives a struct to store stats and logs.
- For Correct printing of logs, I have kept logs of each thread in a local buffer which is dumped in a vector of logs after all the threads are joined then sorted by time of creation ties are broken by thread id.
- For accurate exiting time, I have logged the exit time just before exiting.
- Also, each thread stores the waiting time of each process and the worst waiting time observed in that thread.
- I have tweaked the usage of Compare and exchange function of the atomic header to make it work as suggested in the book.