Implementing TAS, CAS, and Bounded Waiting CAS Mutual Exclusion Algorithms

Ananya Mantravadi
CS19B1004

ENTRY SECTION

This is the part of the code which is accessible to all threads. As all the threads request access to the critical section, we allow only one thread to execute. Bounded wait guarantees that no thread waits forever in the entry section and each thread gets a fair chance to enter the critical section.

CRITICAL SECTION

This is the part of the code where only one thread can be executed at a time.

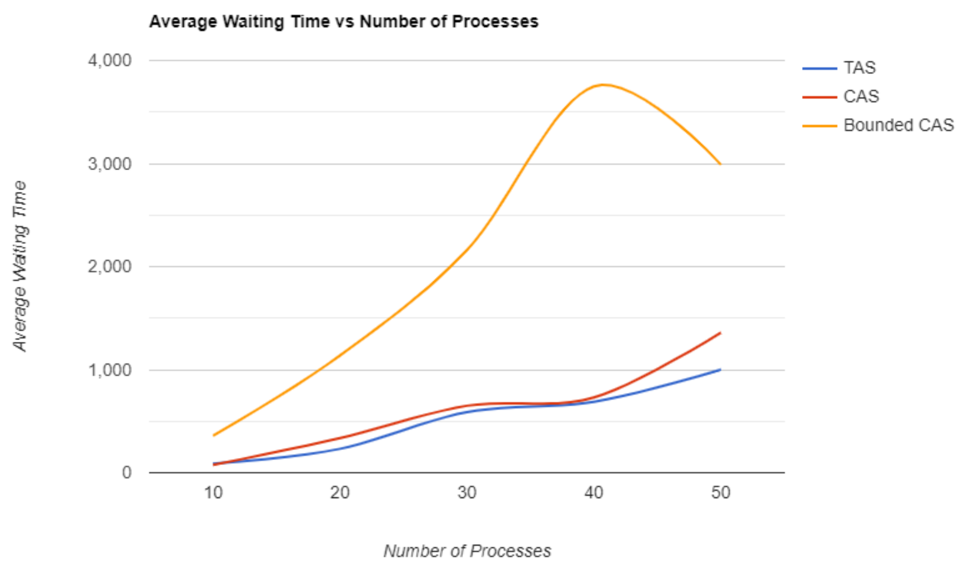
REMAINDER SECTION

This is the part of the code where the thread exits from the critical section to allow another thread into it.

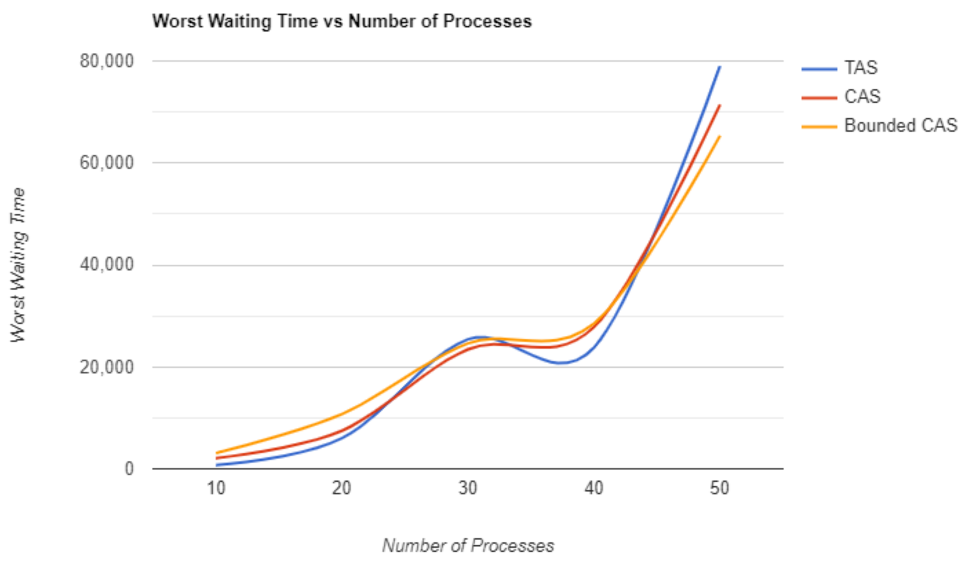
IMPLEMENTATION

The program to implement all three algorithms is similar. We create n threads and make them request to run in testCS function for k times. Each thread enters the entry section, critical section, and exits via the remainder section. We measure the average time spent by the threads waiting in the entry section, and also the maximum amount of time spent waiting as the worst-case time taken. TAS and CAS use `atomic_flag_test_and_set` and `compare_exchange` from <atomic> header file to implement the critical section. For bounded CAS, we create a global array, waiting_queue, and use it to ensure each thread is equally probable of entering the critical section.

GRAPH 1; Average Waiting Time vs Number of Processes



GRAPH 2; Worst Waiting Time vs Number of Processes



CONCLUSION

We observe that the Average Waiting Time is almost similar for both TAS and CAS while it is significantly higher for Bounded-CAS. Since we additionally check for threads that are waiting, this takes up more execution time and in turn, increases the waiting time. But with respect to Worst Waiting Time, Bounded-CAS, TAS, and CAS give almost the same worst waiting time.