ECE 459: Programming for Performance
Assignment 2

Ryan Qin

February 23, 2021

1 Message Passing

The message passing technique uses the “unbounded” channel from the crossbeam crate. Inside the main function, instead of calling “check_all()” which checks all possible combinations sequentially in a single thread, a new method “check_parallel()” is called. This function splits the task into multiple threads, with each thread checking only a sub-tree of the entire possible solution space. More specifically, each thread only checks the possible secrets starting with a certain alphabet, and all threads do this simultaneously. For example, the default alphabets to checks in “a-z0-9”, which would in this case produces 36 threads with first thread checking the secret “a...” and last checking “9...”, (“.” means an arbitrary alphabet). If either thread finds the match, it will put the solution secret onto the unbounded channel. The channel receiver is in the main program thread, placed after declaring and defining the threads. It is blocked until something is received, which is when the main program thread knows it has found the solution and can proceed finishing. Before finishing it shuts down all the threads by calling “drop()”, although it might not need to as the program will finish in a foreseeable amount of time, which shuts down all the child processes anyways.

2 Shared Memory

The shared memory technique is similar to the one above, with slight difference in how the solution is stored. The threads are created in the same manner, where by default 36 threads are created and all start exploring a sub-tree of the solution space simultaneously. However, if one thread finds the solution it puts it onto a shared memory block protected by mutex. Specifically, this memory block is of type “Arc(Mutex(Vec(u8)))”. The thread finding the solution needs to lock the memory block in case other threads access or change it at the same time (which never happens for this problem, because only 1 thread can ever succeed). In the main program thread, there is never ending loop running until the shared memory block has been written to, by a thread that has found the solution. When the loops ends, it then reads content from the mutex and proceed ending the programming by first shutting down all the threads.

Disclaimer: I understand the purpose of the assignment is to practice inter-communication of threads, which might seem more complicated than the provided solution. However, the assignment specification only says to improve the speed by using threads, message passing, and shared memory, all of which are full-filled by the solution. If a problem can be solved in an easier way, why would anyone not?