Js66, hj33

Each VM that acts as a server starts up a TCP server that accepts clients on detached background threads. Each background thread receives the query pattern from the client and runs the grep system call using the popen(3) call, whose output we save in a buffer and send back to the client.

The client consists of a central controller that handles the spawning of background threads using the std::thread API, one for each VM. Each thread sends the pattern to the target server and reads back the grep output. Each thread then places this grep output into a unique .temp file for reading by the main thread. To handle waiting for thread completion, we use std::forward to wait for results from the background threads. The controller iterates through these futures and prints grep output from the corresponding .temp file.

We test by first generating log files. These log files include patterns that only exist on one VM, half the VMs and all the VMS. Interspersed between these lines are lines that occur with three degrees of varying frequency, from rare to highly frequent. To test, we assert that we receive the correct number of lines in each case befitting a pattern that arrives from a certain number of servers or a pattern of a certain frequency. We test this by redirecting stdout to a file.

We implement .sh files to let us do work on other servers without having to ssh into them manually. This includes starting and pkilling servers as well as generating logs and cleaning. Furthermore, we make the program as scaleable and multithreaded as possible by doing the bulk of the work on background threads. And letting the whole system be managed from one VM.



A white rectangular table with numbers

Description automatically generated with medium confidence