

Amit Das
Gautham Ajith

Design/Algorithm Used: On each VM, there is a listener thread that waits on a socket for incoming connections and the main thread which is continuously waiting for user input. Thus, each node has to be both a server and client side. Upon receiving a connection, the listener thread creates a separate thread to handle communication with that client and the other thread goes back to listening. Upon receiving a message from the user, the node multicasts the message to all other nodes it has connected with. The other nodes then receive the message, execute the grep command locally on their files and send the results back to the client that requested it. For a system with n nodes, a node will be running $n + 1$ threads.

Unit Tests: We wrote a distributed unit test that works with 3 nodes, each providing a different query. This test, launched from VM 1, logs into all the other VMs, starts up their servers, and runs the queries with the queries provided. The queries are run on log files that are created up front. The queries test frequent patterns, as well as non-frequent patterns. The script then verifies the line counts from the querier with actual line counts obtained by running remote grep searches. We also wrote a local unit test to test our message module.

Average Query Latency Analysis: Below is a plot showing the query latency which was calculated starting from when a query was initiated to when it received the last response from the other nodes. The plot was created based on 2 queries. One was a GET query which was a frequent pattern returning over 500K lines. The other was searching for the date Aug 2022. This was an infrequent pattern return $< 100K$ lines. 5 queries were run treating each of the 4 machines as clients for each grep pattern. Some general trends is that across all grep patterns, the latency does not deviate significantly regardless of which machine is treated as the client, which aligns with our expectations based on the design. Furthermore, GET messages take significantly longer to retrieve than the infrequent pattern which also makes sense. There is a greater deviation for the more frequent pattern. However, the time required to return line-matches is relatively quick for both queries. (Please note that the query latency times were calculated by disabling printing to the terminal, which takes significant time.) However, we did print out line matches to ensure correctness and did not use the `-c` option, so we were actually transmitting multiple lines (in the order of hundred thousands) across the network.

