Group 14 – MP1 Report
Manan Patel (manandp2), Mihir Shah (mihirs2)

Design

For our design, we run grep separately on each server, using the system's grep binary. The servers return the raw grep output as a byte array, along with the number of lines matched to the query, to the client. The number of matched lines is processed on the server to reduce load on the client.

The client sends the grep query to each server. Once all the queries have been sent, the client waits for each server's response in a first-come, first-served manner. As grep outputs are received from the servers, the client prints the grep outputs and writes them to a corresponding file (per-server). Once all servers have responded, the client prints the total sum of matching lines across all servers and then closes all of the connections.

Testing

We wrote several unit tests using Go's testing framework. We generate log files with both known and unknown lines for each test. For the server, we added tests to catch all successful cases by matching the output of a server's grep function with an expected output. We also tested for edge cases where arguments were undefined, grep returned a status code, etc.

For the client we started by testing reading the server addresses from a file. We also wrote a distributed test from the client side. Similar to the server tests, this test had an expected output to compare which servers found matches. It also generates logs on each server with known and unknown lines. The known lines are based on the VM number (even or odd), to ensure matches are only found from odd numbered VM's.

Analysis

For this analysis, we ran the query `dist-grep "mac os" -i` five times with all servers running, but only counting the time it took for the first 4 servers to respond. The average response time for all four servers was 3.59 seconds, with a standard deviation of 0.144 seconds. This was a relatively frequent pattern with ~90,000 matches per server. Since the query had a large number of matches, it justifies the average response time being higher. The standard deviation is low which is expected because the query and machine states/log files are constant between trials. The first trial was slightly slower which was unexpected. We believe this is due to caching the file read on the servers.