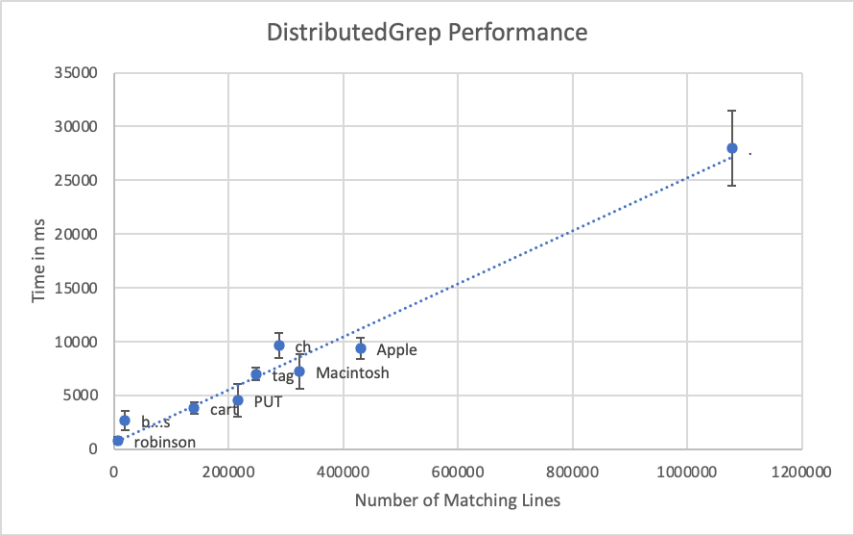
Our DistributedGrep implementation takes a multithreaded approach:
The Main Thread: Spawns server and client threads, interacts with the user and performs input validation.
The Client Thread: Client threads are spawned every time the user inputs a valid grep command (1 for each machine in the network). They create a socket, place the request to their designated server, and print the response in a thread-safe manner.
The Server Thread: One server thread is spawned when the program is run on a machine. It receives connections, reads the query, executes the grep command in a bash shell, and pipes back the output to the client through the socket connection.

We went with this approach because the actual grep tasks are divided across the network i.e. each machine performs its own grep and sends over the results. This also reduces network traffic as only essential data is being transferred. Multithreading in clients allows parallelization of the grep requests across all machines. The overall design is clean as client and server threads are stateless

Server Tests: This verifies if the Server Thread is not running after exit. This also verifies if the server returned the correct grep output.
Client Tests: This tests that the Client can successfully send commands to the assigned servers.
Distributed Log Querier Tests: Tests for console output after the user writes associated input to the console. For instance, this checks for console output when the user inputs an invalid command or an exit command. This also checks for the grep output from all the VMs (Requires other VMs to be on and run the servers) through various grep commands. This also verifies various command input patterns from users (i.e. Invalid command then grep, or grep then invalid command).
For the unit tests, instead of reading from machine.i.log, the server reads from test.i.log, which is changed by the unit tests to reflect the various testing scenarios.



We tested performance over a 30-minute period to minimize the impact of network traffic on accuracy. We found that any queries >10^3 lines took linearly increasing time to complete. This makes sense because both grep and network wait should be O(N) where N is the number of lines, and hence bytes transferred.