Client

The client gets the query from the user and uses a json config file to distribute this query among all the servers listed in the config file. For each server, a separate QueryThread is started, so that the work can be done in a parallel manner. Each thread sets up a socket to connect to the server, sends it's command and then reads the received results into a buffer. Once the client is done receiving data, it adds the result to a thread-safe Queue shared by all the threads and returns. When all the results are received, the main thread waits to join the worker threads..The main thread then pulls each result from the Queue and displays it along with the file name and number of lines, and then also displays total number of lines.

Server

The server listens for an incoming connection on port 45000 in the main thread. Once a connection has been established, it starts a separate thread. This thread is responsible for executing the command and returning the result of that command over the established connection, after which it closes. As a result of this multi-threaded architecture, the server allows simultaneous execution of commands.

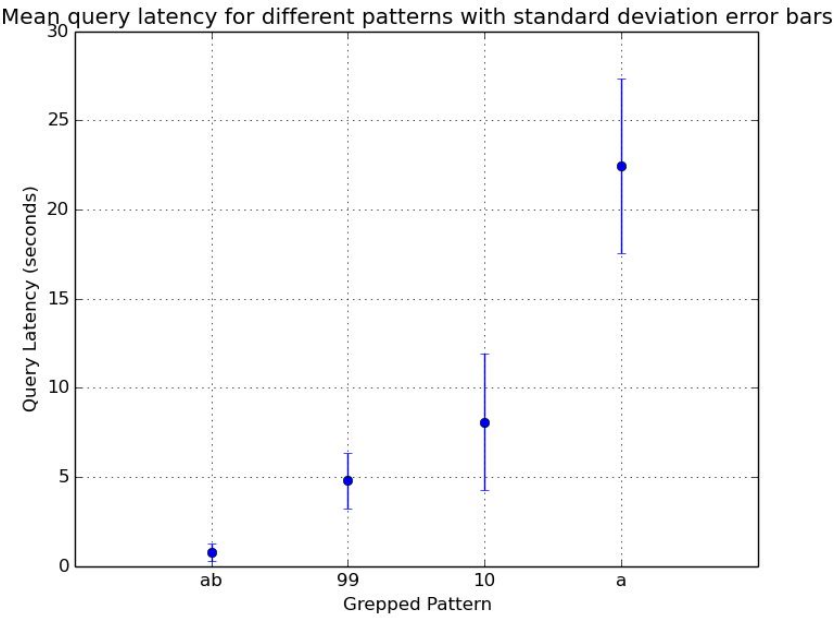
Test

The testing is split into two parts -

1. We generate log files with specific patterns on each server. There are three kinds of patterns classified by frequency - *rare*(once), *somewhat frequent* and *very frequent* patterns, and classified by how many log files the patterns occur in - one, only log files on odd numbered machines, or all logs. There are also random lines added to the log file.
2. Once the files are generated, we run the test suite from any querying machine. The unit tests check the types of patterns generated - patterns not occurring/occuring in one log/some logs/all logs, and patterns classified by frequency. We assert the number of lines matched by our program with the actual number of lines matched, and also assert machine numbers as an extra check.

Average Query Latency + Plots + Analysis

We tested the query latency on 4 log files of ~100 MB each, with patterns of different frequencies. As expected, as the frequency of the pattern increases, the average time taken to grep increases.



Pattern	Total number of times matched
'ab'	~60k
'99'	~600k
'10'	~1million
'a'	~2.5million