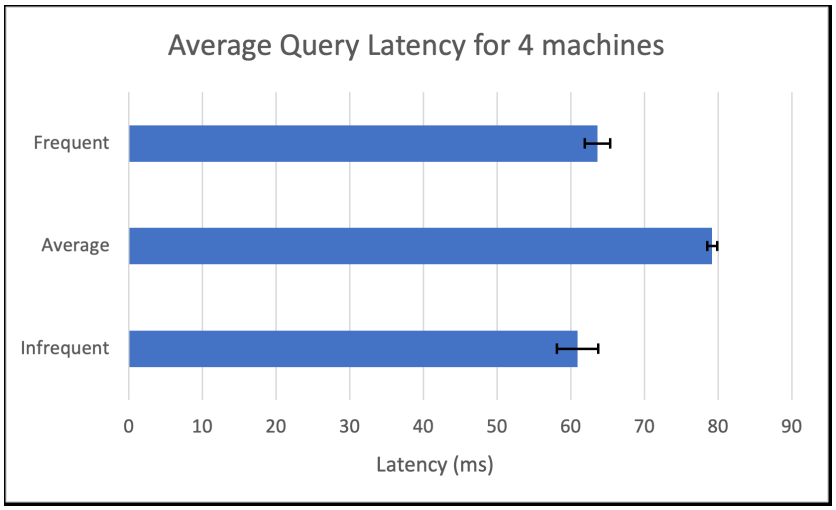
MP 1 — Distributed Log Querying **Diego & Louis**

Design. There are two components in our Golang distributed querier, a TCP server and a TCP client. Each machine runs a server that takes in a grep command and responds with the output of executing the command in a new shell. On any machine, when the client is run with a grep query, it retrieves the list of machines from a file and connects to every machine in parallel. Then, it sends the grep query, and aggregates the corresponding output in sorted machine order for a consistent output.



Performance. We decided to parallelize all outgoing connections from the client to ensure maximum performance. The average query latency over 60 MB Apache logs over 5 trials for different queries (frequent queries implying more matches) is shown in the plot above. For the frequent query, we queried for the number of lines containing valid IP addresses with a regular expression (every line). For the average query, we looked for the total number of POST requests made by clients (around 30k lines). Finally, for the infrequent query, we looked at a specific subset of IPs that only appeared once in a log (once). As evident in the bar graph, the average case is significantly slower relative to the other queries. The trend is unexpected since frequent queries should take the most time to run (and thus have a higher latency). However, this is very dependent on the regex pattern, separate from the performance of our distributed services. Overall, the query latency is a respectable sub-100 milliseconds, thanks to the parallelized connections and simple connection protocol.

Tests. The server test, named *TestServerMultipleClients*, simulates several concurrent connections to the server, including connection establishment, query transmission, and response verification. The client test, named *TestQueryServers,* assesses if the server is handling the grep commands correctly with a series of queries of varying ”pattern frequencies”.

Reflection. While the distributed log querier performs fairly well for our needs, there are some improvements that could be made. Security is lacking, given that we are directly executing user input without authentication. The output is limited to the result of grep commands. If performance was more of an issue, switching to UDP may also improve network latency. There are also faster grep distributions that can be built right into the program, without needing to invoke a new shell, but we chose the most simple approach. Overall, the distribution of our distributed querier, with separate TCP server and client components, enables parallel processing and efficient handling of grep queries across multiple nodes.