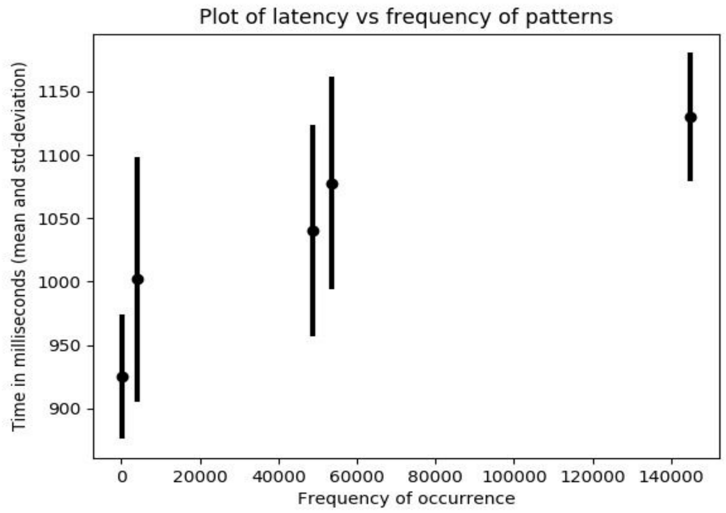
TITLE: Machine Programming 1 – Distributed Log Querier (vandana2, lavisha2)

DESIGN: We have used grpc, an open source RPC (remote procedure call) framework to enable the communication between client and server. The entire MP1 has been coded in Go. We have also implemented concurrency in the code to improve the performance and reduce the time of sequential execution of servers on the VMs. We have also implemented server streaming where we divide the result of the grep into fixed chucks(blocks) and return the blocks to the client. Timeout delay has also been added to make the client wait until server sends back the grep output. We have also supported all flags in grep using system grep and our grep also support regular expressions. If any machine fails, it will detect the failure, give an error message and continue the execution on the rest of the VM servers. We have assumed that machines are fail-stop. We have generalised the code to run grep on any number of VMs by adding the corresponding addresses in the file. We also take the grep flags as an input from the client. The client connects to the other VM servers, sends the grep pattern to be matched. The VM servers run grep locally and send back the grep output to the client. The client integrates the output from all the VMs and displays it with file names and line numbers.

UNIT TEST: We have written code to generate log files that includes known lines (text depends on the VM number) and unknown lines(random string) on the VMs. We test the grep output for the number of occurrences of a few patterns and also verify the corresponding line numbers for the generated log files. We also verify and unit test the frequency of occurrence of a few patterns for the given demo log files.



Count	145000	53600	48800	4000	28
Mean (ms)	1129.6	1077.75	1040.2	1001.8	925.0
Std Dev (ms)	50.7	83.8	83.7	96.7	49.1

1. The graph above is plotted with Number of occurrences of a pattern on x-axis and Time in milliseconds(includes sys and user time using the Linux time command) on the y-axis.
2. We observe that the average mean time reduces as the number of occurrences of a pattern reduces using system grep.
3. We do not observe any set pattern for the standard deviation of time.
4. For a pattern having 145K occurrences, using concurrency grep took 1129.6 milliseconds, and without concurrency we observed 1313.5 milliseconds. Hence the use of concurrent programming improved the performance in Go.