MP1 Report
by: Nicolas Yepes (nyepes2), Ivo Kusijanovic (ilk2)

Design: Each machine in the cluster acts as both a client and a server. The server runs continuously, listening for incoming requests. The client runs on demand. As a client, a machine sends a message containing the arguments to run the distributed grep command (e.g., raw strings or regex) to a list of known machines (VMs) in parallel. As a server, each machine receives the arguments, runs grep on its local log file, and sends the results back to the originating client. The client aggregates responses from all servers, outputting either the raw matching lines or, the match count per machine and for the entire cluster. If a server ever fails, the client handles it gracefully by timing out and continuing the process.

Unit Tests: In each machine we use a script to generate a random log containing frequent, medium, and rare ip addresses and random urls, some of which can be matched by regex. Its purpose is to see that the distributed grep can handle frequent, medium, and rare raw string patterns as well as regex. We check this by comparing the actual and expected output. Furthermore, we test the log queries behavior when some machines are shut down.

Query Latency: The figure below shows the query latency when running our distributed grep implementation across a cluster of 4 machines each containing a 60MB log file. We time the implementation without the "-c" flag and pipe the output to /dev/null. Now the results aligned with what we expected as frequent patterns had to send larger byte streams across the network (i.e. each machine had to send the raw matching lines for the local grep command back to the client). However, a surprising thing was that the regex pattern ran surprisingly fast. It is true that not that much data had to be sent, however, we thought that having to compute and resolve the regex was going to take a lot longer. Also we can notice that frequent patterns had a pretty large standard deviation. This is likely due to our implementation sending 1024 byte packets at a time. Therefore, the time from sending to receiving the message was dependent on the current network bandwidth and speed at the execution time. This wasn't the case for the other cases, as they had few if not a single 1024 byte packet to send.