CS425: MP1- shreya28 and ananya13

Design and Implementation:

For this MP, we utilized socket programming in Python and a client-server architecture to run the grep commands. All VMs contained the machine.i.log file and a server.py file. The querying machine contains the log & server file and additionally has the client.py file. In client.py, the querying machine sends a request (the grep command) to all the VMs. The server receives this request (grep command) from the client, and locally runs the command using the subprocess module in Python. The server sends the grep results back to the client, where the client receives the results, and displays the output from each VM.

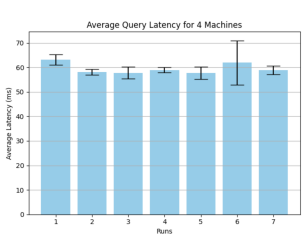
Test Cases

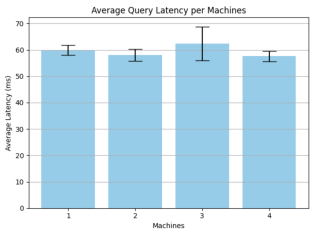
For the test cases, we had a test.py where we randomly generated log files, making sure we had certain special lines in the log file. We stored this as a list and added the grep commands that we wanted to test it with to the end and sent this list to server_test.py. The server_test.py generated the log file on each VM, and ran the grep commands from test.py. We sent the results from the grep commands back to test.py to verify results. The test cases were as follows:

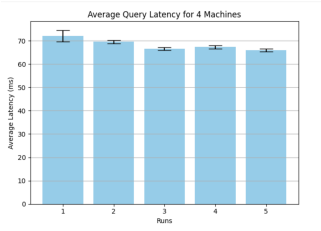
a) Checking count for an infrequent pattern which does not occur on any log file.
b) Checking count for infrequent pattern (occurs < 5 times)
c) Checking count for frequent pattern
d) Crashing one server and checking output of other machines
e) Trying different grep commands such as -E, -i & regex expressions.

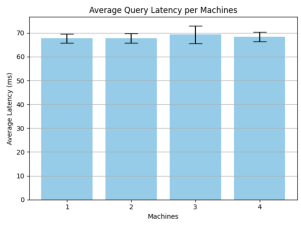
Latency plots:

For frequent patterns, the two plots on the left are what we got. We performed 7 runs/trials for the frequent patterns. The frequent pattern that we chose was to identify all IP addresses. For infrequent patterns, the two plots on the right are what we got. We performed 5 runs/trials. The infrequent pattern we chose was to find the name “garcia”.









From the graphs, we can see that the latency is higher for infrequent patterns than frequent patterns. This is most likely because when the pattern is frequent, it will be cached after the first occurrence so when subsequent searches are made for the same pattern, it is already in memory making the search faster.