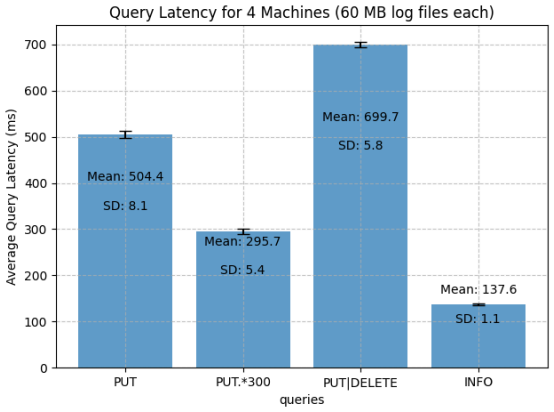
1. **Design flow:**
    a. Command Initiation:
        i. The client receives the pattern and logs the file location as input
        ii. Sends command to all cluster nodes (including itself) via TCP
    b. Server Process (Each Node):
        i. Executes grep command locally upon receiving a request
        ii. Saves output on the local machine
        iii. Replies to client with total line count from grep results
    c. Concurrent Handling:
        i. Client spawns separate goroutine for each node request
        ii. Maintains a list of servers with pending responses
        iii. Removes nodes from the list as responses are received
    d. Response Collection and Error Handling:
        i. The client combines results once all nodes have responded
        ii. Prints combined output to screen
        iii. Implements heartbeat mechanism to detect slow or unresponsive servers
2. **Unit testing:**
    a. Unit test 1-4: Generates data on all the nodes. The script uses the vm.log file to generate the specific part of the log file and faker package to add fake data to add randomness to the test log file on each machine. Then this unit test, runs the test on various patterns like a frequent pattern (PUT), a somewhat frequent pattern (PUT.*301), a rare pattern (PUT.*300) and a specific pattern we inserted in a single log file (GET.*700).
3. **Results and analysis**:



a. As seen in the graph, the bars show the average query latency and the error margins as standard deviation.

b. The difference between the query times is a direct function of the number of lines each grep query returns. This is because, the higher the number of lines, the more time it takes to complete the data processing on each server and write the data in a file.

c. The small standard deviation can be accounted for based on several factors such as network congestion and the time the querying machine takes to synchronize its threads after completing execution.