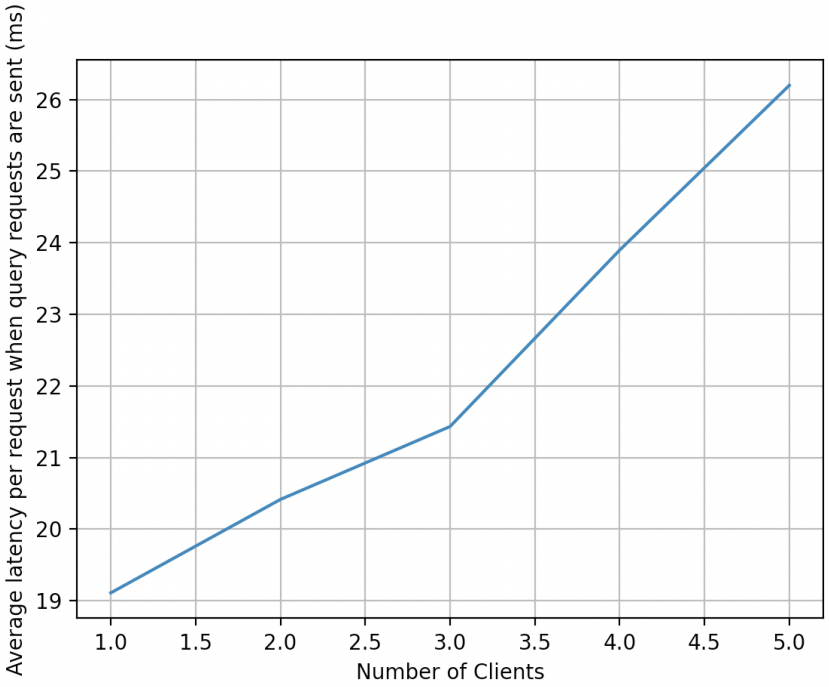
EVALUATION DOCUMENT

Part 1: Socket Programming and Custom Thread Pool Implementation

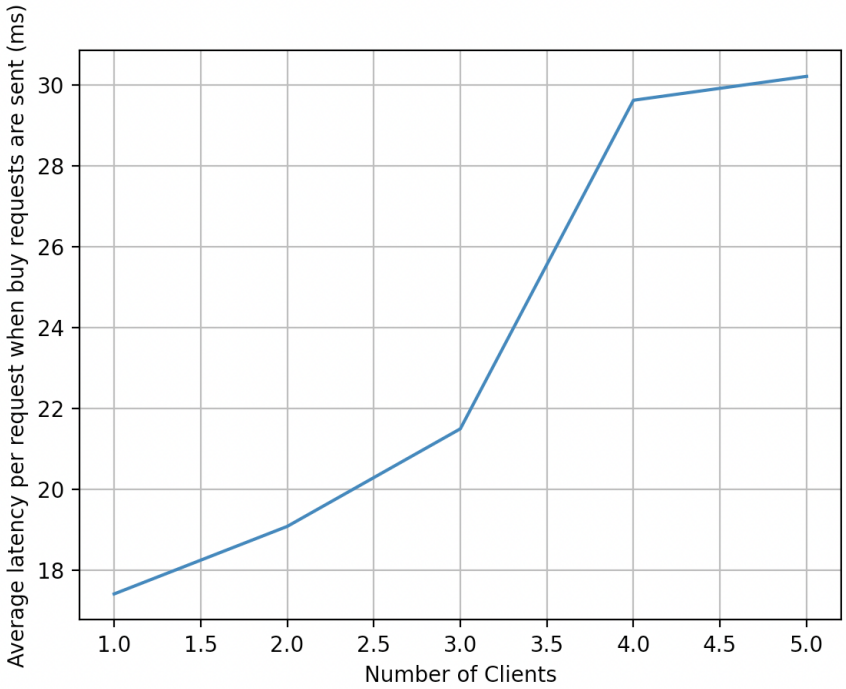
Analysis/Graphs :

1. This is the graph of average latency per request in milliseconds vs the number of clients that are hitting on the server simultaneously. - This test is only for **Query** requests. (100 requests in each client)



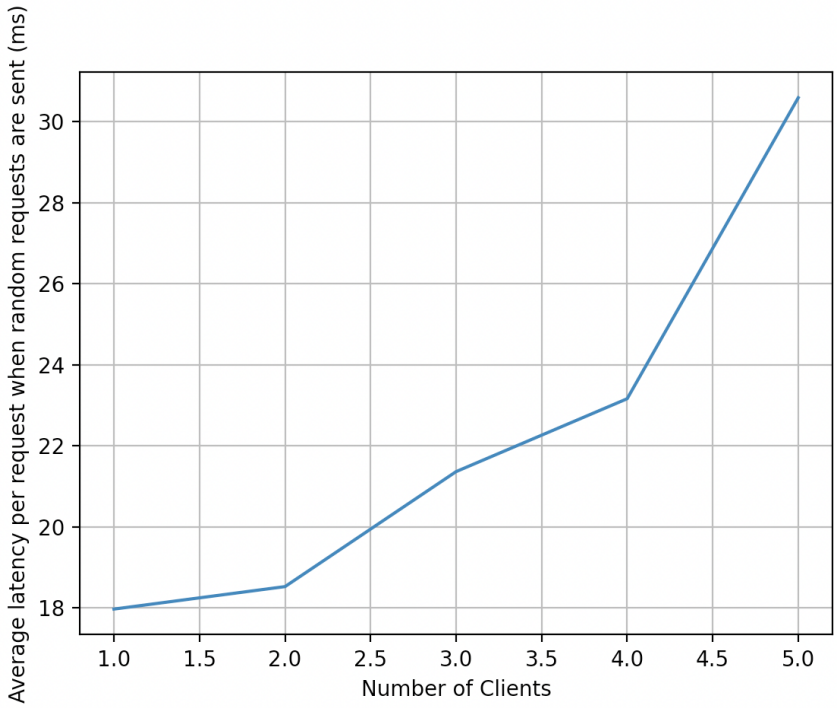
1: 19.11ms
2: 20.415ms
3: 21.433334ms
4: 23.8975ms,
5: 26.198002ms

2. This is the graph of average latency per request in milliseconds vs the number of clients that are hitting on the server simultaneously. - This test is only for **Buy** requests. (100 requests in each client)



1: 17.41ms
2: 19.08ms
3: 21.493332ms
4: 29.6175ms
5: 30.208002ms

3. This is the graph of average latency per request in milliseconds vs the number of clients that are hitting on the server simultaneously. - This test is for **both buy and query requests generated randomly.** (100 requests in each client)



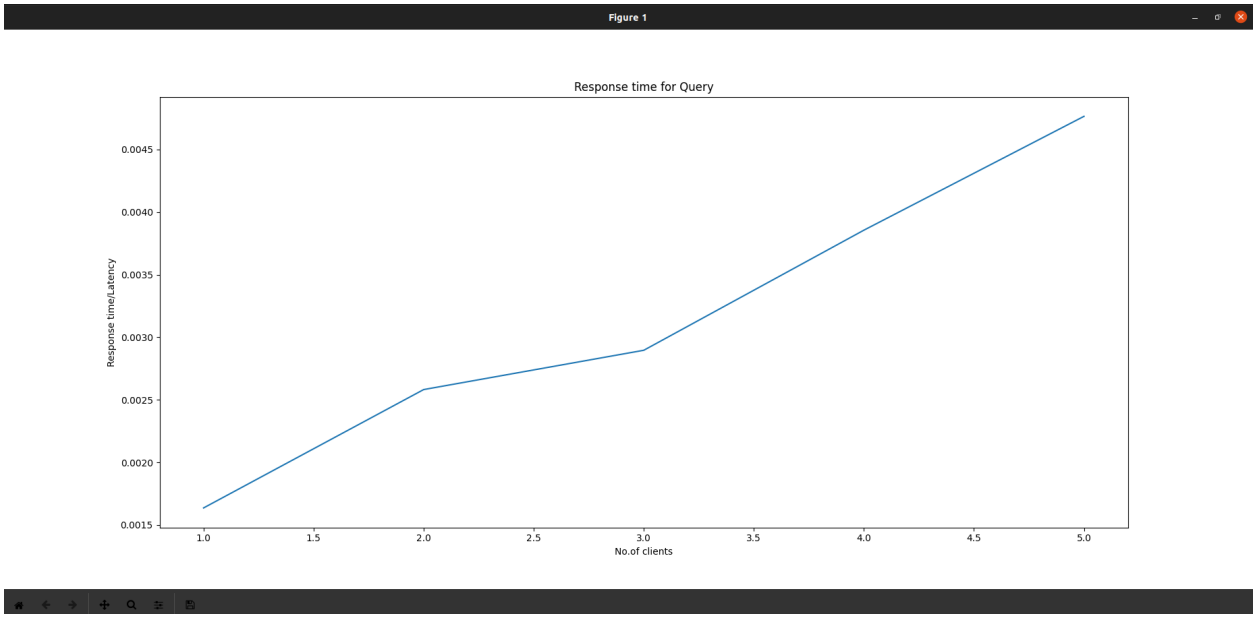
1: 17.97ms
2: 18.525ms
3: 21.363333ms
4: 23.1625ms
5: 30.592001ms

Part 2: Implementation with gRPC and Built in Thread Pool:

Graph displaying the latency as the load goes up for **Query(ItemName)** with no. of clients on X-axis and the latency in seconds on Y-axis for 100 concurrent requests.

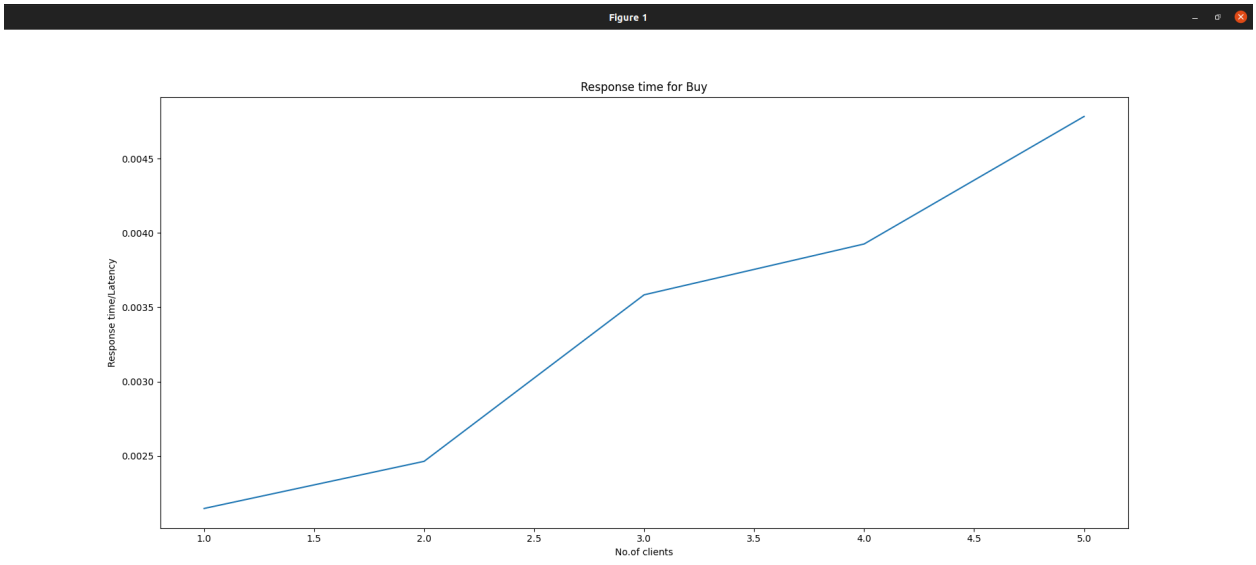
The average latencies of Query ranges from 15 milliseconds to 47 milliseconds.

EVALUATION DOCUMENT



Graph displaying the latency as the load goes up for **Buy(ItemName)** with no. of clients on X-axis and the latency in seconds on Y-axis for 100 concurrent requests.

The average latencies of Buy ranges from 22 milliseconds to 48 milliseconds.



1. How does the latency of Query compare across part 1 and part 2? Is one more efficient than the other?

According to our observations, for socket programming, we observed the average latency ranging from 19.11milliseconds to 26.19milliseconds. The average latency has hit

26.19milliseconds for 5 concurrent clients. In case of gRPC, the average latencies of Query ranges from 15 milliseconds to 47 milliseconds and the average latency has hit 47 milliseconds for 5 concurrent clients. We have observed that socket communication is relatively faster compared to grpc. In socket communication we just establish a to and fro socket connection and use it to communicate the requests back and forth. In contrast a gRpc client calls a stub method which notifies the server that the RPC has been invoked providing the clients metadata for the call, the method name and the specified deadline. Then the server sends back its own initial metadata. Once the server has a client's request message, it creates and populates the response to the client with additional tailing metadata. This additional layer of packing and unpacking leads to a delay in gRpc.

2.How does the latency change as the number of clients (load) is varied? Does a load increase impact response time?

According to our results, in case of both socket and gRPC protocols, we observe that average latency increases as the number of clients are increased i.e., as the load is increased. As the same number of threads are serving a higher number of clients this is as anticipated. As the load goes up, the latency is increased as multiple threads are actively competing for the same system resources such as processor, memory, I/O etc. Refer the graphs above for validating the same.

3.How does the latency of the query compare to buy? You can measure latency of each by having clients only issue query requests and only issue buy requests and measure the latency of each separately. Does synchronization play a role in your design and impact performance of each? While you are not expected to do so, use of read-write locks should ideally cause query requests (with read locks) to be faster than buy requests (which need write locks). Your design may differ, and you should observe if your measurements show any differences for query and buy based on your design.

There is no significant difference between the latencies of Query and Buy. The average latencies of Query ranges from 15 milliseconds to 47 milliseconds and average latencies of Buy ranges from 22 milliseconds to 48 milliseconds. We have implemented a single lock here in each of the Query and Buy methods.Though we have expected buy requests would have more latency as compared to query requests, since a single lock is used for each of the above operations, we have observed similar average latencies in query requests (with read locks) and buy requests (read and write locks). We have sent Query requests from clients to calculate Query performance and sent Buy requests from clients to calculate Buy performance and plotted the graphs separately.

4. In part 1, what happens when the number of clients is larger than the size of the static thread pool? Does the response time increase due to request waiting?

EVALUATION DOCUMENT

The main thread of the server accepts the client connection whenever the client is trying to make a connection with the server. The main thread inserts these requests continuously into the request queue. We can have each worker thread working on each client request(socket connection). For example, if we have created a threadpool of 20 requests and we have 20 clients requesting concurrently, each worker thread will work on each request. But, if we receive one more request and all the threads are still processing the previous requests, then the new client has to wait till one of the threads is done processing the requests. So, when the number of clients is larger than the size of the static thread pool created, the response time increases due to request waiting.