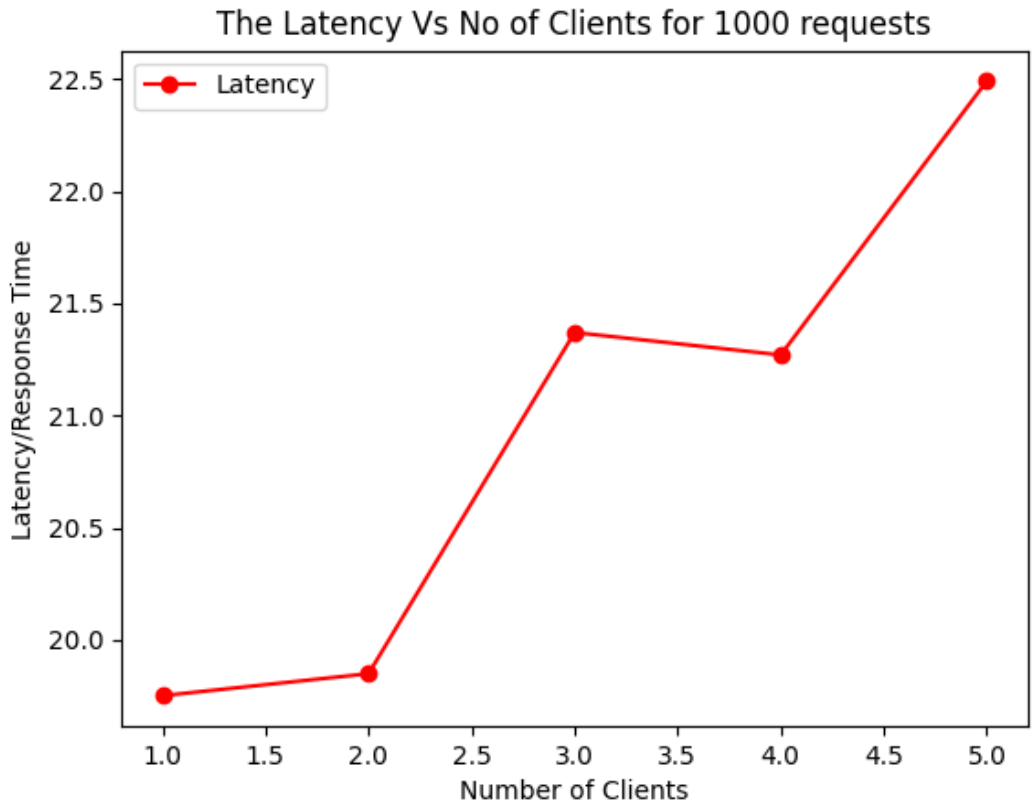
# Evaluation Document

**Part -1: Implementation with Socket Connection and Handwritten Thread Pool**

Fig below shows the latency according to the increase in number of clients for 1000 requests from each client.

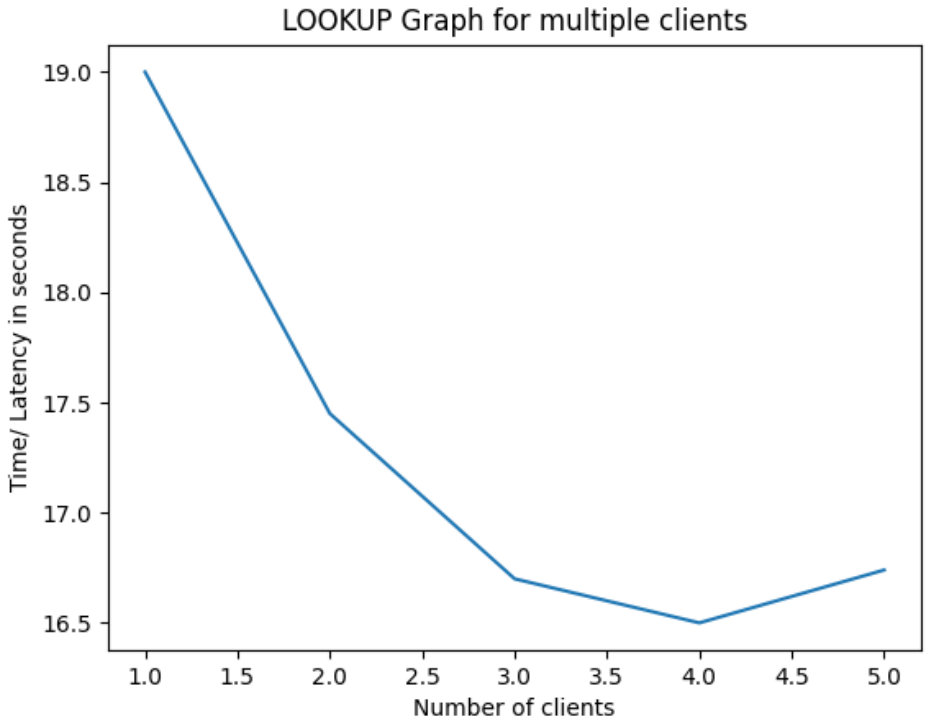



As the No. of clients increases, the response time increases with load. When the clients increased from 1 to 5, we can see in the graph that the response time increased from 19.752 to 22.49sec.

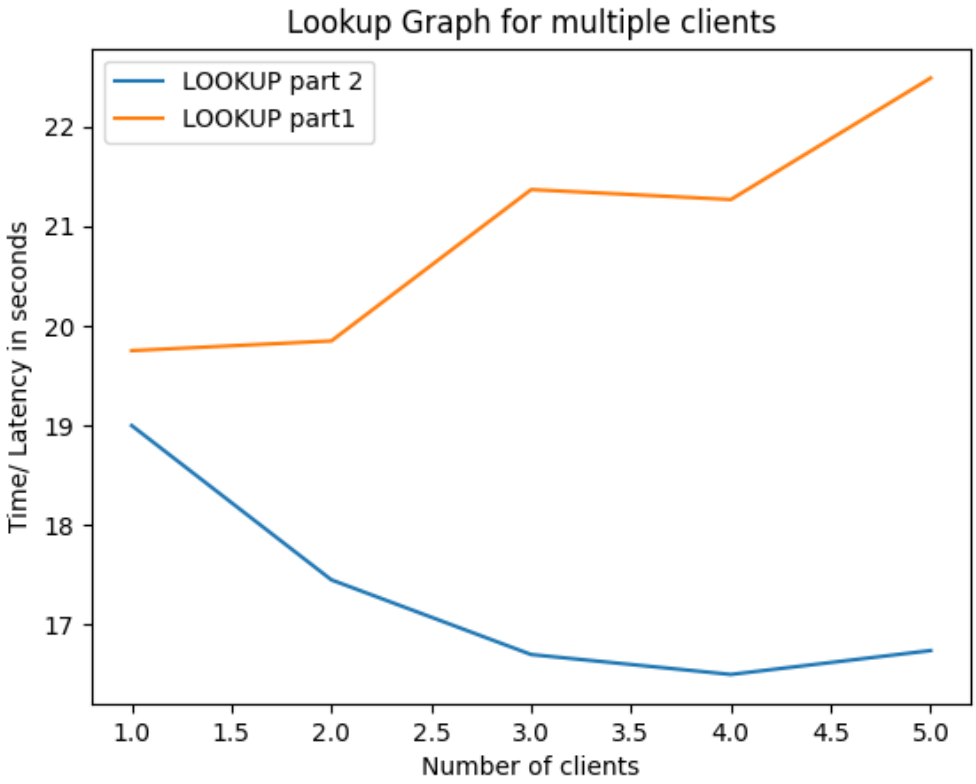
Latency Response time

Time taken for 1 client : 1000 requests - 19.752410174999909 s
Time taken for 2 client:1000 requests - 19.750 , 20.12 -Average - 19.850 s
Time taken for 3 client : 1000 requests - 20.850 , 21.12 , 22.34 - Average - 21.37 s
Time taken for 4 client : 1000 requests - 21.06,19.08,20.98,21.90 - Average - 21.27 s
Time taken for 5 client : 1000 requests - 21.67, 21.86 , 22.23, 22.65, 22.82 - Average - 22.49 s

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

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



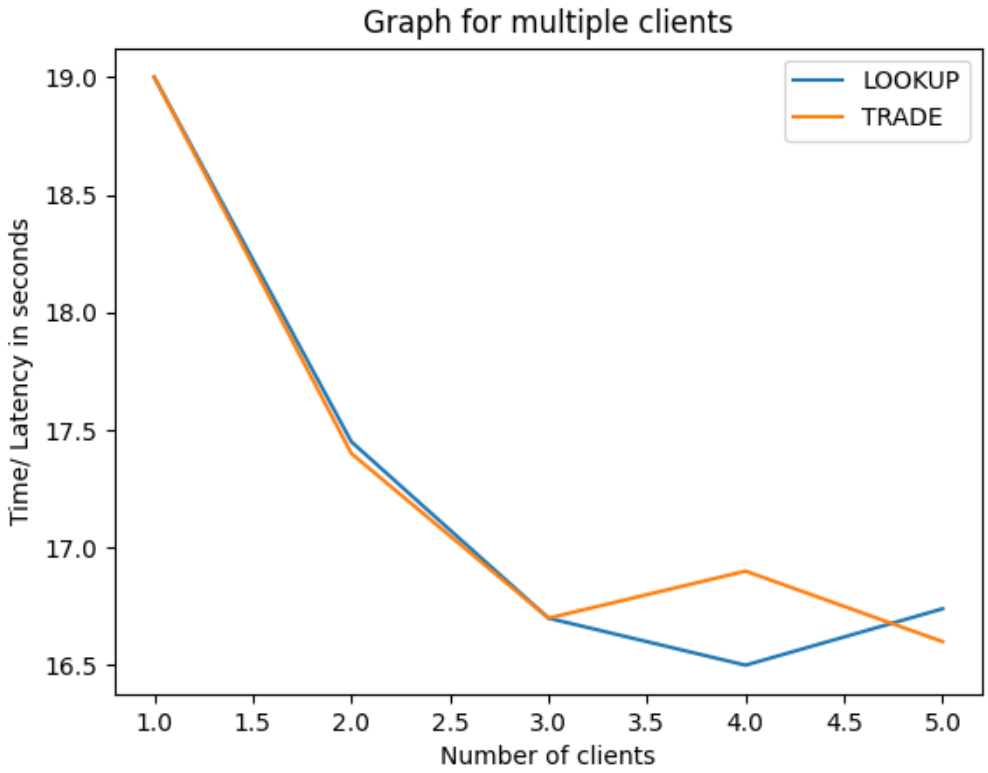

Latency of lookup request in part 2 using gRPC is much more efficient than part 1 lookup which is modelled using handmade thread pool.

The above figure shows that gRPC can handle multiple clients with minimal latency than thread pool in part 1. For 1 client part 2 lookup is taking ~1sec less than the part1. Whereas for 5 clients the efficiency of part 2 is more as it is taking 5sec less.

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

We are observing a gradual increase in the response time even though very minimal while executing the scripts with increase in load.
As you can see in the graph 1, as the client increases from 1 to 5, the latency also increases from 19 to 22.5 seconds.
The load increase in server and response time has a direct relation between each other. When the number of clients is increased to 9 while the number of threads in the thread pool were 6, a noticeable increase in time was observed.

3. **How does the latency of lookup compare to trade? You can measure latency of each by having clients only issue lookup requests and only issue trade requests and measure the latency of each separately. Does synchronization pay 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 lookup requests (with read locks) to be be faster than trade requests (which need write locks). Your design may differ, and you should observe if your measurements show any differences for lookup and trade based on your design.**




The above graph was taken by varying the server load from 1 to 5 where each client sending 1000 requests. The latency of trade and lookup for part 2 looks almost similar, but at one point when number of clients is 4 trade request has taken some time more than the lookup request.

We are using different locks to handle lookup and trade requests. Ideally synchronization plays a role in latency as the write locks take more time than the read locks. But with our design, varying clients from 1 to 5 and 1000 requests we are not able to see much difference between the lookup and trading requests.

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

   Yes. If the number of clients is more than the number of threads in the pool, then the requests will be waiting in the queue until the worker threads finishes its current processing. Hence, there will be noticeable increase in the latency. We tested this by increasing the number of clients to 9 where the thread pool had only 6 worker threads, the latency increased by 7sec.