Christopher Shi
33061961
cshi@umass.edu

1 Evaluation and Performance Measurement

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

- From figure 2 and figure 4 we can wee that the average latency of 1 client for part 2 is 17.92 ms while the average latency of 1 client for part 1 is 12.38 ms. Trivially we can see that the part 1 implementation of Lookup is more efficient than the implementation in part 2. This could be due to the fact that the part 2 implementation utilizes thread locks before performing the read operation required of Lookup. The part 1 implementation does not need locks because no write operations are utilized, only read operations, which means that synchronization is not necessary.

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

- From all the figures, we can see that as the number of clients increase, the latency increases as well. Given that both parts implement a thread per request model, this is the expected response, as opposed to a thread per connection model, in which we would have expected to see the latency remain constant until the number of clients is greater than the number of threads in the pool.

1.3 How does the latency of lookup compare to trade? Does synchronization play a role in your design and impact performance of each?

- From figure 2 and figure 3 we can see that the average latency of the trade function is similar to the average latency of the lookup function. Synchronization does play a role in my design, but the impact on performance should be the same for both functions, since a lock is utilized for both. This is reflected in the measured results. The implementation of a read-lock for the Lookup function and write-lock for the Trade function could yield differing latency times, as a read-lock is ideally faster than a write-lock.

1.4 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?

- From figure 5, we can see that as the number of clients surpasses the size of the thread pool, there is no significant jump in latency. This is because part 1 implements a thread per request model, and as such this is the expected response, as opposed to a thread per connection model, in which we would have expected to see the latency remain constant until the number of clients is greater than the number of threads in the pool, at which point we would expect a drastic increase in latency.

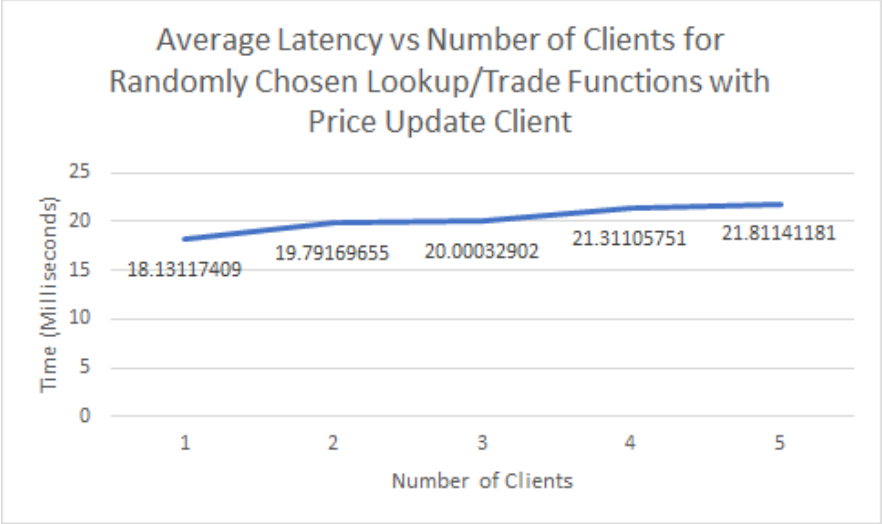

Figure 1: Average Latency of Requests for gRPC Server with Randomly Chosen Lookup and Trade Requests. An additional client was run in tandem that updated the prices of the stocks at random increments.

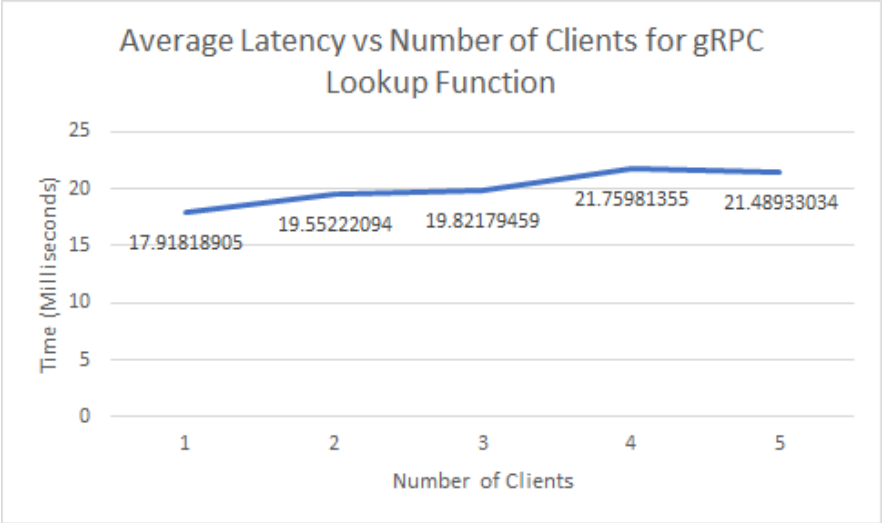

Figure 2: Average Latency of Lookup Requests for gRPC Server.

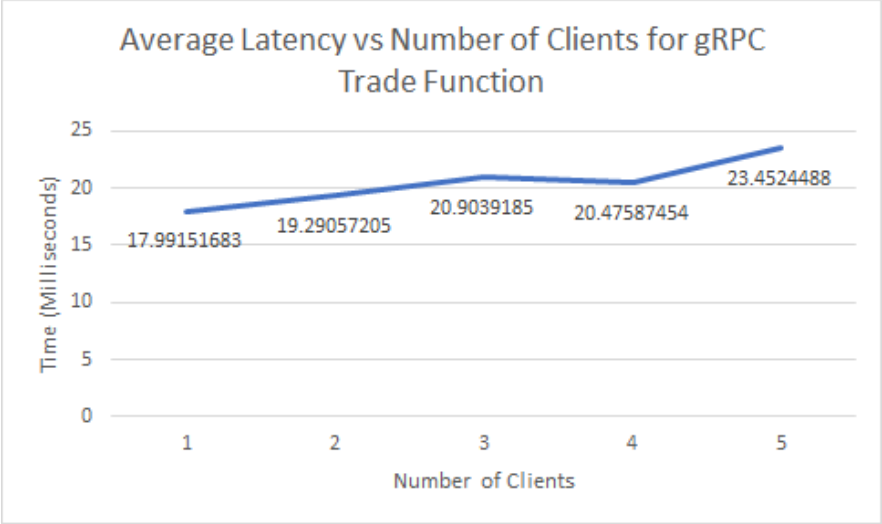

Figure 3: Average Latency of Trade Requests for gRPC Server.

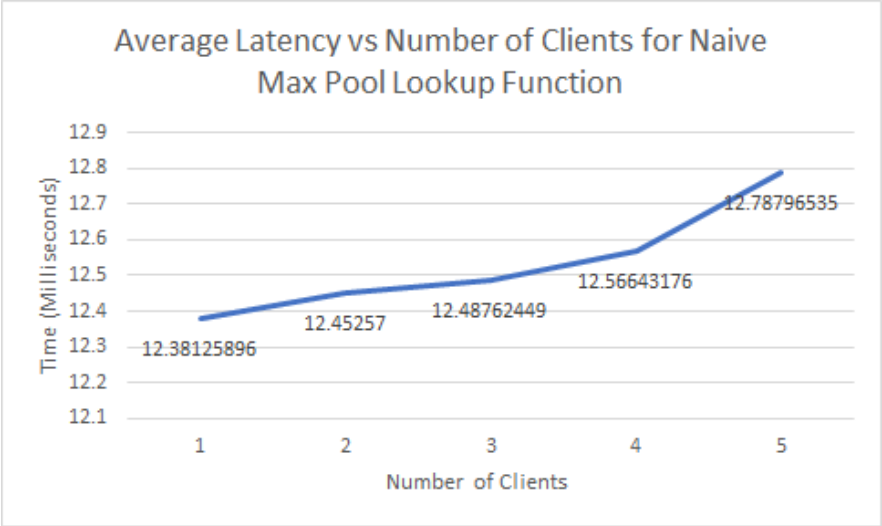

Figure 4: Average Latency of Lookup Requests for Naive Max Pool Implementation Server.

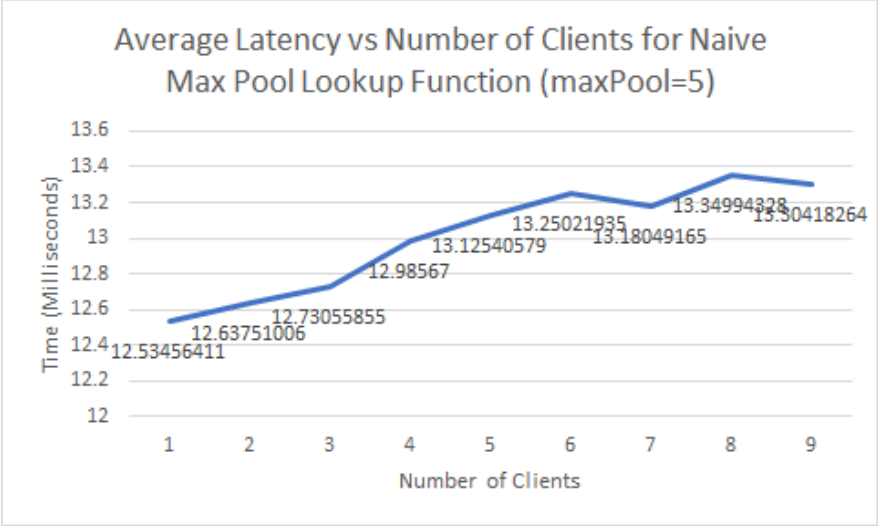

Figure 5: Average Latency of Lookup Requests for Naive Max Pool Implementation Server. A max pool size of 5 was utilized.