Distributed Systems - Programming Assignment 1

Darshan Vijayaraghavan *Umar Ahmed Thameem Ahmed*

Objective

Build a marketplace with client and server interfaces using TCP/IP (socket interfaces) and evaluate their performance (latency + throughput) with a comparison study involving different scenarios.

Overview

We completed our assignment using Python with sockets to communicate. It uses TCP for sending and receiving messages. The client communicates with the server using a CLI interface. The user must create an account and login before performing any action. MySQL was used for the database.

Setup

For the evaluation setup, we wrote a Python script that creates buyers and sellers with unique IDs, logs in, performs a few selected operations for 1000 iterations and records the latency and throughput at each step. The same setup is then run for 10 iterations to calculate the average latency and average throughput. We have taken 2 operations per each buyer and seller client. Then a comparison between these two in different scenarios.

For running concurrent users, the simulation setup creates threads for each client. We ran into problems when running 100 concurrent processes, and had to increase the database connection pool range in the database configuration to tackle this. We implemented a threading barrier to ensure all threads complete account creation and login, then wait until every thread has reached the barrier before starting operations simultaneously.

We have deployed the database, the seller and the buyer servers on GCP to test if the setup works without any code changes other than the .env file. We use two E2 virtual machines to run the servers and CloudSQL to house the product and customer database. We set firewall rules to allow public access to the server components and authorized our VM instances to access the database within the Google private network. We have performed initial testing and have discussed it in the observation section.

Experiment 1 (1 seller + 1 buyer)

Iterations	get_seller_rating		search_items	
	Latency (ms)	Throughput (ops/sec)	Latency (ms)	Throughput (ops/sec)
1	1.27	788.8	0.89	1128.92
2	2.12	471.62	2.23	448.31
3	1.48	674.69	0.99	1014.56
4	1.85	538.91	1.35	739.91
5	1.94	514.02	1.15	866.35
6	1.53	654.66	0.9	1107.54
7	1.26	793.12	1.08	926.78
8	1.43	696.87	0.87	1148.01
9	2.88	346.78	2.57	389.63
10	1.23	811.92	0.86	1166.99
Average	1.699	629.139	1.289	893.7

Iterations	display_items_for_sale		register_item_for_sale	
	Latency (ms)	Throughput (ops/sec)	Latency (ms)	Throughput (ops/sec)
1	1.5	664.4	0.82	1214.11
2	1.41	708.25	0.74	1353.36
3	1.36	736.92	0.73	1372.84
4	1.51	661.78	0.75	1336.3
5	1.93	518.41	0.72	1390.91
6	1.93	519.01	0.73	1370.88
7	2.11	473.59	0.75	1325.57
8	2.39	418.11	0.73	1370.85
9	1.48	674.18	0.73	1372.4
10	1.6	626.3	0.85	1175.32
Average	1.722	600.095	0.755	1328.254

Experiment 2 (10 sellers + 10 buyers)

Iterations	get_seller_rating		search_items	
	Latency (ms)	Throughput (ops/sec)	Latency (ms)	Throughput (ops/sec)
1	5.92	1688.2	6.34	1578.07
2	5.4	1850.4	3.66	2730.38
3	5.37	1862.41	3.74	2673.45
4	5.32	1878.27	3.66	2729.1
5	6.1	1638.23	3.66	2730.39
6	8.22	1216.07	4.26	2346.32
7	9.72	1029.29	4.07	2458.09
8	5.57	1796.55	4.02	2487.98
9	7.79	1283.37	4.18	2390.51
10	5.65	1770.33	4.11	2433.93
Average	6.506	1601.312	4.17	2455.822

Iterations	display_items_for_sale		register_item_for_sale	
	Latency (ms)	Throughput (ops/sec)	Latency (ms)	Throughput (ops/sec)
1	5.81	1719.62	4.23	2366.18
2	5.62	1778.23	4.24	2357.92
3	5.64	1772.43	4.23	2362.56
4	5.53	1807.25	4.17	2396.91
5	5.58	1792.02	4.23	2364.64
6	5.68	1761.31	4.46	2242.22
7	5.58	1790.5	4.37	2289.38
8	5.91	1692.01	4.41	2268.51
9	6.22	1608.58	4.43	2258.7
10	8.3	1204.48	4.48	2231.7
Average	5.987	1692.643	4.325	2313.872

Experiment 3 (100 sellers + 100 buyers)

Iterations	get_seller_rating		search_items	
	Latency (ms)	Throughput (ops/sec)	Latency (ms)	Throughput (ops/sec)
1	94.28	1060.78	74.1	1349.68
2	62.23	1607.03	56.98	1755.05
3	65.23	1533.13	50.37	1985.34
4	69.31	1442.8	71.35	1401.51
5	65.84	1518.78	55.62	1797.85
6	67.66	1478.04	48.48	2062.65
7	64.33	1554.59	50.35	1986.27
8	67.27	1486.62	52.63	1900.16
9	68.08	1468.86	48.47	2063.26
10	65.06	1536.97	49.74	2010.57
Average	68.929	1468.76	55.809	1831.234

Iterations	display_items_for_sale		register_item_for_sale	
	Latency (ms)	Throughput (ops/sec)	Latency (ms)	Throughput (ops/sec)
1	68.73	1455	58.32	1714.78
2	106.27	941.09	86.71	1153.27
3	100.59	994.22	82.75	1208.53
4	59.21	1688.88	77.18	1295.7
5	60.74	1646.43	74.53	1341.74
6	68.4	1462.09	70.83	1411.79
7	67.61	1479.15	72.36	1382.06
8	65.24	1532.76	70.3	1422.55
9	70.59	1416.66	81.67	1224.47
10	69.97	1429.15	73.5	1360.51
Average	73.735	1404.543	74.815	1351.54

Key Observations

1. Concurrency Overhead in Low-Load Scenarios:

The single-client scenario shows lower throughput than the 10-client scenario despite having the lowest latency. This result is explained by the server-side threading overhead. With only one client, the server's multi-threaded architecture (spawning a new thread per connection) introduces unnecessary context-switching and synchronization costs that kill the benefits of concurrent execution. The server is optimized for concurrent workloads, making it less efficient for single-client operations.

2. Optimal Performance at Moderate Concurrency:

The 10-client scenario achieves the best performance across all other scenarios. At this concurrency level, the system properly uses the database connection pooling, multi-threaded server architecture and TCP connection perks that the overhead introduced is being amortized across the number of operations performed thus reducing latency and increasing throughput.

3. Resource Content at High Concurrency:

The 100-client scenario shows degraded performance as compared to 10 clients. This can be because of additional wait times due to data connection pool saturation, thread contention, network limits and lock contention.

4. Operation-specific Characteristics:

Read operations consistently show better throughput than write operations. On the other hand, write operations have higher latency due to database commit overhead and potential lock contention.

5. Latency increase proportional to Data sent and received:

We noticed that in display_items_for_sale when we had 7000 items the average latency for the operation was recorded as 244.32ms when compared to 5.987ms in scenario 2. Making it evident that the latency is directly proportional to the data sent over the network.

6. Infrastructure Impact:

Testing on GCP with CloudSQL showed significantly higher latencies compared to local deployment. This might be due to network round-trip time, geographic distance and shared infrastructure. The average latency for scenario 1 and 2 after deploying on GCP are given in the below table. We can see that the average latency has a significantly higher value of 89.261 ms when compared to 1.722 ms in local and the throughput has decreased to 11.209 ops/s from 600 ops/s in Scenario 1 for display_items_for_sale in seller server.

For display_items_for_sale:

	AVG Latency	AVG Throughput
Scenario 1	89.261	11.209
Scenario 2	98.03	102.098

For register_items_for_sale:

	AVG Latency	AVG Throughput
Scenario 1	86.378	11.581
Scenario 2	159.926	65.012