

CSCI 5673: Performance Report - Programming Assignment 1

Done by - Sahil Shroff Tanish Praveen Nagrani

Evaluation Configuration

We evaluated the system using the `tools/bench.py` script, which simulates concurrent buyer and seller sessions. The components were deployed as separate processes interacting over TCP sockets.

- Hardware:

Processor - Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz (1.80 GHz)

Installed RAM - 24.0 GB (23.8 GB usable)

System type - 64-bit operating system, x64-based processor

- OS: Windows

- Database: PostgreSQL (Localhost)

API used for simulation -

Buyer - AddItemToCart()

Seller - DisplayItemsForSale()

Scenario 1: 1 Seller, 1 Buyer

- Average Buyer Response Time: 5.99 ms

- Average Seller Response Time: 3.75 ms

- Throughput: 462.5 ops/sec

Explanation

Scenario 1 establishes the baseline performance. With minimal contention (only 2 active threads), the system achieves its highest per-client efficiency. The response time is dominated by basic TCP round-trip and database query execution time.

Scenario 2: 10 Sellers, 10 Buyers

- Average Buyer Response Time: 5.4 ms
- Average Seller Response Time: 5.0 ms
- Throughput: 207.5 ops/sec

Explanation

We observed a dip in aggregate throughput here. This is likely due to the overhead of context switching between threads (Python GIL) starting to outweigh the benefits of concurrency, or potential contention on database locks for the shared items. However, response times remained stable (~5ms), indicating the system is not yet overloaded.

Scenario 3: 100 Sellers, 100 Buyers

- Average Buyer Response Time: 6.1 ms
- Average Seller Response Time: 4.1 ms
- Throughput: 427.7 ops/sec

Explanation

The system demonstrated strong stability under high load (200 concurrent threads). Throughput recovered to near-baseline levels (~427 ops/sec), and crucially, average response times did not degrade significantly (staying around ~4-6ms). This suggests that the threading model and database connection pool (tuned to satisfy this concurrency) are handling the load effectively without request queuing.

Overall Analysis

The system demonstrates successful utilization of TCP sockets for concurrent client communication. The use of a threaded server model allows for stable response times even as the number of clients scales to 100 sellers and 100 buyers. While there is some throughput overhead at medium concurrency (Scenario 2), the system handles high concurrency (Scenario 3) without failing or suffering from extreme latency spikes, validating the architectural choice of a stateless backend with a persistent database.