Performance report

Assignment 4

Introduction:

In this programming assignment, we are tasked with enhancing the robustness and reliability of an online marketplace system through strategic replication and protocol implementation. This assignment builds upon the foundational work completed in the first two programming assignments, aiming to overcome the limitations of a single-server setup that has proven to be both a performance and failure bottleneck.

The primary objectives of this assignment are to:

- Design and implement a rotating sequencer atomic broadcast protocol to ensure reliable and ordered message delivery across distributed systems.
- Replicate the customer database across five servers using the newly implemented rotating sequencer protocol, addressing communication unreliability.
- Replicate the product database over five servers utilizing the Raft consensus algorithm, which is robust against server crashes and unreliable communication.
- Replicate the server-side interfaces for sellers and buyers across four servers each, enhancing the system's availability and fault tolerance.
- By achieving these goals, we will significantly improve the system's scalability and resilience, ensuring that the marketplace remains operational and efficient despite server failures or network issues.

For the purpose of evaluation, we deployed the system on the Google Cloud Platform (GCP), with each server component operating on a separate instance. We measured the system's performance by calculating the average response time and throughput for the following critical functionalities:

- Get seller rating on the seller side: This function retrieves the seller's rating, which is a crucial metric for maintaining trust and quality within the marketplace.
- Get seller rating on the buyer side: Accessing the seller's rating from the buyer's perspective is essential for informed purchasing decisions.
- Add items to the cart on the buyer's side: This operation is fundamental to the shopping experience, and its efficiency directly impacts user satisfaction.

Results:

In the results section of our report, we presented a comprehensive analysis of the system's performance under various conditions. We detailed the average response times and throughput rates for each client function across three distinct scenarios, ranging from normal operation to the failure of critical components. This evaluation not only demonstrated the system's current capabilities but also provided insights into its behavior under stress, highlighting the effectiveness of the implemented replication and fault-tolerance mechanisms.

- Average response time for each client function when all replicas run normally (no failures): 50ms
- Average response time for each client function when one server-side seller interface replica and one server-side buyer's interface to which some of the clients are connected fail: 62ms
- Average response time for each client function when one product database replica (not the leader) fails: 50ms
- Average response time for each client function when the product database replica acting as leader fails: 86ms

For the seller:

Instances	Avg Response Time	Avg Throughput
1	50 ms	68.3 API invocations per second
10	89 ms	17.67 API invocations per second
100	120 ms	5.55 API invocations per second

For the buyer side:

Instances	Avg Response Time	Avg Throughput
1	50 ms	51.5 API invocations per second
10	90 ms	15.2 API invocations per second
100	118 ms	2 API invocations per second

Conclusion:

- Normal Operation vs. Failure Scenarios:
    - The average response time remained consistent at 50ms when all replicas were running normally and when a non-leader product database replica failed. This indicates that the system's design effectively mitigates the impact of a single replica failure on performance.

    - However, when a leader replica in the product database failed, the average response time increased significantly to 86ms, suggesting that leader election and the subsequent recovery process in Raft may temporarily affect performance.
    - The failure of one server-side seller and buyer interface replica resulted in a moderate increase in response time to 62ms, which shows some resilience but also room for improvement in handling failures.

- Scalability Analysis:
    - As the number of instances increased, both the average response time and throughput decreased across seller and buyer functions. This is expected due to the increased load on the system.
    - For sellers, the average response time increased from 50ms to 120ms, and throughput decreased from 68.3 to 5.55 API invocations per second as the number of instances grew from 1 to 100.
    - For buyers, a similar trend was observed, with response times increasing from 50ms to 118ms and throughput decreasing from 51.5 to 2 API invocations per second.

- Performance Comparison:
    - The performance degradation under load suggests that while the system scales, there are bottlenecks that could be addressed to improve throughput and response times, such as optimizing database operations or increasing network bandwidth.
    - The relatively stable response time in the face of individual replica failures demonstrates the effectiveness of the replication strategies, particularly the rotating sequencer atomic broadcast protocol and Raft, in maintaining system availability and consistency.