Trung “Jason” Nguyen - ttn190009

Tri Ngo - tdn190002

Project Overview

- **Purpose:** Provide a reliable and fault-tolerant distributed data storage system that consistently maintains data across multiple servers while handling network partitions and server failures.

System Architecture

- **Nodes:**
    - **Data Servers (S0-S6):** Store and manage data objects. Participate in replication and synchronization.
    - **Clients (C0-C4):** Initiate read and write requests to the servers.
- **Communication:**
    - **Reliable FIFO Channels:** Utilize Java’s ServerSocket for reliable communication between clients and servers, and among servers.
    - **Hash Function (H):** Determines the three servers responsible for storing an object (Ok) based on H(Ok), H(Ok) + 2 modulo 7, and H(Ok) + 4 modulo 7. Use Object ID as Ok. For example, object 1 will be stored on server 1, 3 and 5.

Key Features & Requirements

1. **Object Replication:**
    - **Write:** Clients write to a primary server (determined by the hash function), which subsequently replicates the data to two other servers.
    - **Read:** Clients can read an object's value from any of the three replicas.
2. **Fault Tolerance:**
    - **Minimum Two Replicas:** Writes should succeed if at least two of the three replicas are available.
3. **Concurrent Write Handling:** If concurrent writes to the same object occur, ensure consistent ordering on all replicas (enforced with Java ServerSocket TCP connection)
4. **Network Partition Handling:**
    - **Partition Tolerance:** The system should continue to function within partitions, allowing permitted updates.
    - **Replica Synchronization:** After partition merges, synchronize outdated replicas with current data.

Design Considerations

- **Data Structures:**
    - Custom object class with ID and content
    - Use hashmap to store object

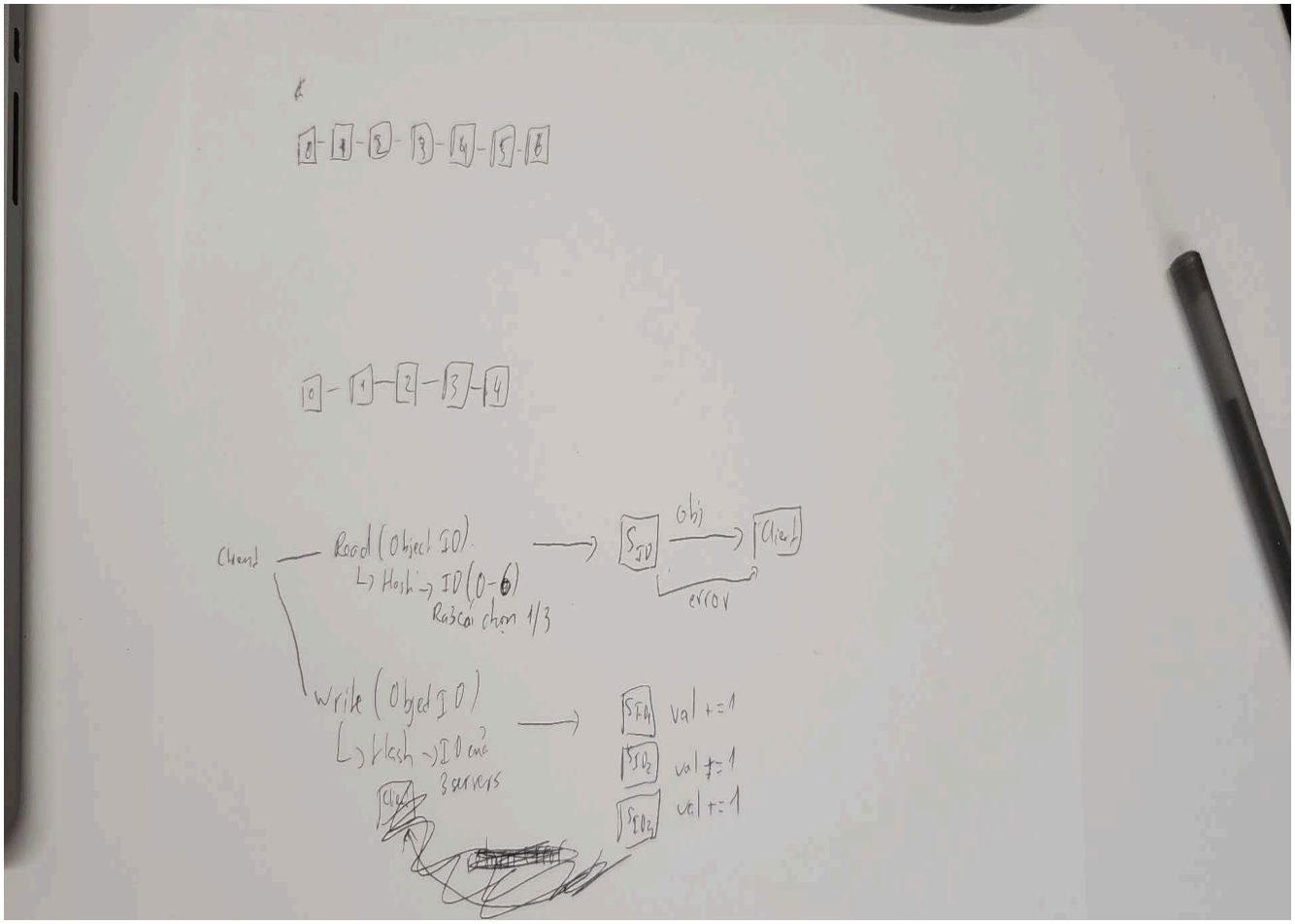
- **Replica Coordination:**
    - **Primary Server Responsibility:** The primary server is responsible for coordinating writes with other replicas.
- **Failure Detection:**
    - **Heartbeats:** Send heartbeat message to test replicas' condition before write
- **Error Handling:**
    - **Return Codes:** Define clear error codes for read/write failures.
    - **Client Retries:** Implement retry mechanisms with timeouts on the client-side.

Testing

- **Unit Tests:** Test individual components (hash function, data structures, communication logic).
- **Failure Scenarios:**
    - Simulate network partitions, server crashes, and concurrent writes.
    - Verify system behavior aligns with requirements.

Technology Choices

- **Programming Language:** Java


0 - 1 - 2 - 3 - 4 - 5 - 6
0 - 1 - 2 - 3 - 4
Client
Read (Object ID)
↳ Hash → ID (0-6)
Raisonnement 1/3
obj
error
write (Object ID)
↳ Hash → ID and 3 servers
val += 1
val += 1
val += 1