**System Description**

**Overview:**

This system demonstrates causal consistency across multiple datacenters and clients. The core idea is to ensure that operations are applied in an order that respects causal relationships, using a centralized server for coordination and dependency management.

This system has 2 parts:

**Classes:**

* **Server**
  + The server facilitates communication between the clients, managing data consistency across multiple other connected servers
  + Ensures casual dependencies by delaying further write operations until all dependencies are satisfied
* **Client**
  + Sends requests for writing to their respective servers

**Communication Protocol**

Each request follows a simple protocol where messages are formatted as {port:}{operation}:{message}

Examples:

Client register request - 50001:RegisterRequest:[]

Writing request to a server - 50001:WRITE:x,I lost my wedding ring

Server replicates write to another server - REPLICATE:x-I found it

**Program Structure:**

**Server Class:**

* Attributes:
  + HOST and PORT: Define the address and port the server listens on.
  + Clients: Set of registered clients
  + Dependency: Keeps track of the dependency for future write operations
  + Datacenters: A list of other data centers to replicate writes with
  + Storage: Stores writes from clients in key:value pairs
* Methods:
  + start\_server(): Initiates the server, listens for incoming peer connections, and spawns new threads to handle each connection.
  + handle\_client(): Processes messages from peers, such as file registration, file chunk requests, and file location requests.
  + RegisterClient(): Adds a client to the server’s set
  + writeToStorage(): Stores data and manages dependencies for write requests
  + replicateWrite(): Replicates writes to all other connected data centers, respecting dependencies
  + send\_to\_server(): Facilitates communication to other servers

**Peer Class:**

* Attributes:
  + host and port: Define the address and port for the peer’s server.
  + Datacenter\_port: The port of the server that the client communicates with
* Methods:
  + start\_server(): Initiates the peer’s server
  + writeToDataCenter(): Prepares writes to be sent
  + send\_to\_server(): Sends messages to server

**What works and what doesn’t:**

All functionality of the system works.

**How to run it:**

In the console, simply run:

python3 demo.py

Once the last server has collected all writes, press ctrl+C to end the script.

**Sample Output:**

This is the output from one run of the code. There are long periods where peers are simply downloading chunks, so for the sake of brevity, those areas are not commented on. Each red comment signifies an important step in the system.

**Servers and Clients start up**

Server listening on 127.0.0.1:65432

Server listening on 127.0.0.1:65433

Peer listening on 127.0.0.1 : 50001

**Client registers with Server1**

Client 50001 Sending RegisterRequest:[] to server...

Starting RegisterRequest for Client 50001

Server response: RegisterRequest from 50001 completed

**Client writes a message to Server1 which delays the replication to Server2**

Client 50001 Sending “WRITE:x,I lost my wedding ring” to server

Starting WRITE for Client 50001

Delaying message to 65433

Server response: WRITE from 50001 completed

**Client writes a new message “y” with a dependency of “x” to Server1**

Client 50001 Sending “WRITE:y,I found it” to server

Starting WRITE for Client 50001

Replicating message to 65433

Server response: WRITE from 50001 completed

Starting REPLICATE for Client 65432

REPLICATING: y

Dependency == x

**Server2 hasn’t received message “x”, so it delays the processing of “y”**

Server 65433: DEPENDENCY x NOT FOUND, Delaying replication until dependency satisfied

**Server1 finally sends “x” to Server2**

Replicating message to 65433

Starting REPLICATE for Client 65432

REPLICATING: x

Dependency == [‘ ‘]

Server response: REPLICATE from 65432 completed

**Server2 has now processed message “x” and it can now process “y”**

Dependency has been satisfied, writing message to storage

Server response: REPLICATE from 65432 completed