1. System Overview

This system functions as a central hub for weather data distribution, leveraging a client/server architecture. The core components include:

- **Aggregation Server**: Handles client requests for weather data, processes updates from content servers, stores data long-term, and purges outdated entries as needed.
- **GET Client**: Retrieves weather data from the server via GET requests and displays it in a user-friendly format.
- **Content Server**: Extracts weather data from the local filesystem, formats it into JSON, and uploads it to the Aggregation Server using PUT requests.

2. System Component Diagram

The diagram illustrates the interaction between the GET Client, Aggregation Server, and Content Server. Key components are:

- **Aggregation Server**: Listens for GET and PUT requests, manages a file containing the latest weather data, and ensures data recovery post-server crash.
- **GET Client**: Sends GET requests to the Aggregation Server for weather data, ensuring data consistency through Lamport clocks.
- **Content Server**: Reads local data, converts it to JSON, and sends it to the Aggregation Server via PUT requests.

3. Implementation of Multi-Threading Safety and Consistency Management

To ensure safety and consistency in a multi-threaded environment, the following measures have been implemented:

- **Thread Safety**: The Aggregation Server employs synchronization mechanisms, allowing multiple clients to perform GET operations or multiple content servers to perform PUT operations simultaneously without causing data races or unsafe data modifications. During a PUT operation, conflicts or dangerous modifications to data are prevented.
- **Lamport Clocks**: A local Lamport clock is maintained across servers and clients. This ensures that during message transmission or reception, all servers and clients adjust their clocks accordingly, guaranteeing that PUT and GET operations are executed sequentially without race conditions.

4. Server Replicas and Justification

To enhance fault tolerance and uptime, the system employs multiple replicas of the Aggregation Server. The number of replicas is determined by the expected client and content server workload, as well as the system's requirements for availability and fault recovery. In critical scenarios, at least three replicas are necessary to maintain service availability under high-load conditions.

5. Testing Strategy

A comprehensive testing strategy will be implemented to ensure system reliability and robustness, including:

- **Unit Testing**: Verifies the core functionality of each component, ensuring it works correctly in isolation.
- **Integration Testing**: Confirms that the interactions between components function properly by testing the system with multiple clients and content servers running simultaneously.
- **Failure Testing**: Assesses the system's fault tolerance and recovery by simulating failures such as server crashes or network outages.