CMPE 275 Project 1

Gossamer

(A distributed Storage System)

**Team Members:**

Ashutosh Singh

Viraj Nilakh

Shruti Padmanabhan

1. **Technologies used:**

|  |  |
| --- | --- |
| Languages | Java |
| Core Packages | Netty, Google Protobuffer |
| Databases | Redis |
| Builder | Apache Ant |

1. **Main Components:**

Client : Responsible for sending read and write requests

Server: Responsible for responding to messages from the client, participating in leader election

Leader: Responsible for replying to client requests

Follower: Store and send data

1. **Topology:**

Mesh Topology is being used as it was easier to build. Even in case of multiple node failures, alternate path could be found. Addition and deletion of nodes can be done without affecting other nodes.

1. **Workflow:**

**Two types of requests:**

1. **Write Requests from client connected to cluster**

Client sends a write request to cluster requesting leader to distribute the file across the network.

The first step involves client chunking the file and storing it in an in-memory Array List. This arrayList is then converted into Command Request messages which are then passed to an executor service which spawns as many threads as the OS allows and writes these messages to the channel to be sent to the leader.

1. **Read requests from client connected to cluster**

Client sends a read request to leader requesting a particular file by name. Leader replies back with the file chunks and client sorts these chunks and assembles them to a file.

1. **Read request from client requesting files saved to cluster**

Client send a requestAllFilesNames request to cluster to get a list of all files stored on the cluster.

1. **Leader Election:**

Leader election is done by means of Raft algorithm. Raft uses randomized timers running on all nodes. The timer of the node which gets finished first broadcasts the vote request to all other nodes in the cluster and they respond depending on their current state. The leader gets elected if and only if it receives majority of the votes.

1. **Replication**

We are using W=1 replication as it would be easier implementing work stealing as all nodes would have the data and any node can steal message request from a queue of any other node. We are using Redis nodes at each cluster node to keep track of all chunks.

1. **Node discovery**

**For a period of 30 seconds, after the cluster formation is initiated, all nodes broadcast to all other nodes. This allows initial ode discovery and output is written to a route.conf which is used to form edges between nodes.**

**After the initial timer of 30 seconds has expired, the connection thread shuts down and leader election takes place. From now on, only the leader can add nodes to network. Any new node wanting to join the cluster sends a connect request to leader and leader informs other nodes of the new node and connections are formed.**

1. **Client API**

Client allows four options to the user.

**Option 0 :** Exit

**Option 1:** Write file to server

**Option 2:** Read all file names from server

**Option 3:** Read a specific file from server

1. **Knowledge gained**
2. **Netty**
3. **Redis**
4. **ProtoBuff**
5. **Architecure advantages and disadvantages**
6. **Leader election algorithms**
7. **Java multithreading handling**
8. **Contributions:**
9. Ashutosh Singh: Node discovery, Client API, Write by multithreading, Write to leader by messages, Refactoring, Report creation, Testing, Research
10. Viraj Nilakh: Leader election, Replication to followers, Testing, Research
11. Shruti Padmanabhan: MySQL adapter, Research, Testing