Learning Management System using gRPC
(Milestone - 3)
Report

Submitted by -

Sanskar Bharadia 2024H1030068P

Aarnav JP 2024H1030075P

Amir Faizal S 2024H1030073P

Submitted to -

Dr. Pratik Narang

(Course Instructor - Advance operating System)

Objective

The objective of Milestone 3 is to implement the Raft consensus protocol within the distributed Learning Management System (LMS). This protocol ensures that critical data, such as grades and student progress, remains consistent across multiple nodes, even in cases of node failure. The focus was on building a fault-tolerant, reliable system using a simplified Raft protocol, involving leader election and log replication.

Overview of Raft Consensus Algorithm

Raft is a consensus algorithm that is commonly used to manage a replicated log across distributed systems. It is designed to ensure consistency in a system where multiple nodes need to agree on a shared state, such as which actions to take or which data to commit, even in the presence of network partitions or node failures.

Raft Consensus Protocol -

1. **Leader Election**:
    - Each node in the LMS cluster initiates leader election if it does not receive an update from an active leader within a specific timeout.
    - The leader coordinates data replication and handles all critical updates.
    - Election mechanics involve `requestVote` and `requestVoteReply` RPCs. Nodes compare terms, and the one with the highest term and log receives votes.
2. **Log Replication**:
    - When a leader is elected, it begins sending heartbeat to other followers using the `appendEntries` RPC.

- Followers acknowledge these entries via the `appendEntriesReply` RPC. Each entry includes the term, previous log index, and commit index to maintain the correct sequence.
- The log replication mechanism allows for fault tolerance, ensuring that updates to critical data are logged consistently across all nodes.
- Whenever an instructor updates grades of a student by using `GradeSubmission` RPC. The grades are replicated across all the nodes in the system.

Usage of Raft in Our LMS Project

In the Learning Management System (LMS) project, Raft is used to ensure consistency and fault tolerance across multiple nodes, particularly in a distributed setup where several servers need to maintain a synchronized state. By applying Raft, the LMS achieves high availability and resilience, allowing it to continue functioning even if some nodes fail or if there is a network partition. Below are key points on how the Raft algorithm's communication mechanisms are used within the LMS project:

3. **RequestVote Description**: In a distributed LMS setup, when a node (e.g., a server managing assignments or grades) wants to become the leader, it sends out a `RequestVote` to other nodes. This request asks the other nodes to vote for it as the leader for a specified term. The message includes information about the sender's log, specifically the `lastLogIndex` and `lastLogTerm`, which indicate the state of the node's log.
   - **Usage in the LMS**: Suppose the current leader node handling assignment submissions goes down, a follower node might issue a `RequestVote` to become the new leader. This ensures that new assignment submissions and queries can still be processed even during leader node failure.
4. **RequestVoteReply Description**: After a node requests votes from other nodes, it receives responses in the form of `RequestVoteReply`. This reply indicates whether the recipient node grants its vote to the sender. If a majority of the nodes grant their vote, the requesting node becomes the leader for that term.
   - **Usage in the LMS**: When a node (e.g., a server processing student queries) requests to be elected as the leader, it gathers votes from other nodes using this mechanism. If it gets the majority vote, it becomes the leader and can now handle all requests, ensuring that no interruption occurs for LMS users during leader transitions.
5. **AppendEntries Description**: Once a leader is elected, it starts sending batches of log entries to follower nodes to replicate its log. These entries can be either empty heartbeat or can be a new grade entry that needs to be replicated. The leader also sends information about the previous log entry (`prevIndex`, `prevTerm`) and the current commit index to ensure followers stay synchronized.
   - **Usage in the LMS**: For example, when an instructor grades an assignment the leader node adds this action as a new log entry and sends it to the follower nodes using `AppendEntries`. This guarantees that all nodes have an updated and consistent view

of the LMS operations, allowing students and instructors to access the same data regardless of which node they connect to.

6. **AppendEntriesReply(from, to, term, entryAppended, matchIndex)**
   - **Description**: After receiving an `AppendEntries` request, a follower node responds with `AppendEntriesReply`. This reply indicates whether the log entries were successfully appended to the follower's log (`entryAppended`) and provides the `matchIndex` indicating the latest index that the follower node's log matches with the leader's log.
   - **Usage in the LMS**: When follower nodes in the LMS receive new log entries (grade update), they send back an `AppendEntriesReply`. This response helps the leader track which nodes are fully synchronized and ensures that a majority of nodes have replicated the log before the leader considers the action committed. This guarantees data integrity, ensuring that operations like submitting assignments or posting grades are reliably stored across multiple servers.

Benefits of Using Raft in This LMS Project

Using Raft can bring several benefits to an LMS with a distributed architecture:

1. **Data Consistency**: With Raft, every server maintains the same log of LMS actions, ensuring data consistency. This is crucial for educational platforms, where losing assignment submissions or grades could cause significant issues.
2. **Fault Tolerance**: By replicating data and achieving consensus across nodes, Raft enables the system to continue functioning even if some servers fail. This fault tolerance is essential in high-availability LMS systems, where downtime needs to be minimized.
3. **Scalability**: Raft's ability to manage consensus across distributed nodes allows the LMS to scale horizontally. As demand grows, the system can add new nodes without sacrificing consistency, as Raft will ensure the new nodes are synchronized with the rest.