CS223 Project Description

DEADLINE: Jun 6, 2022

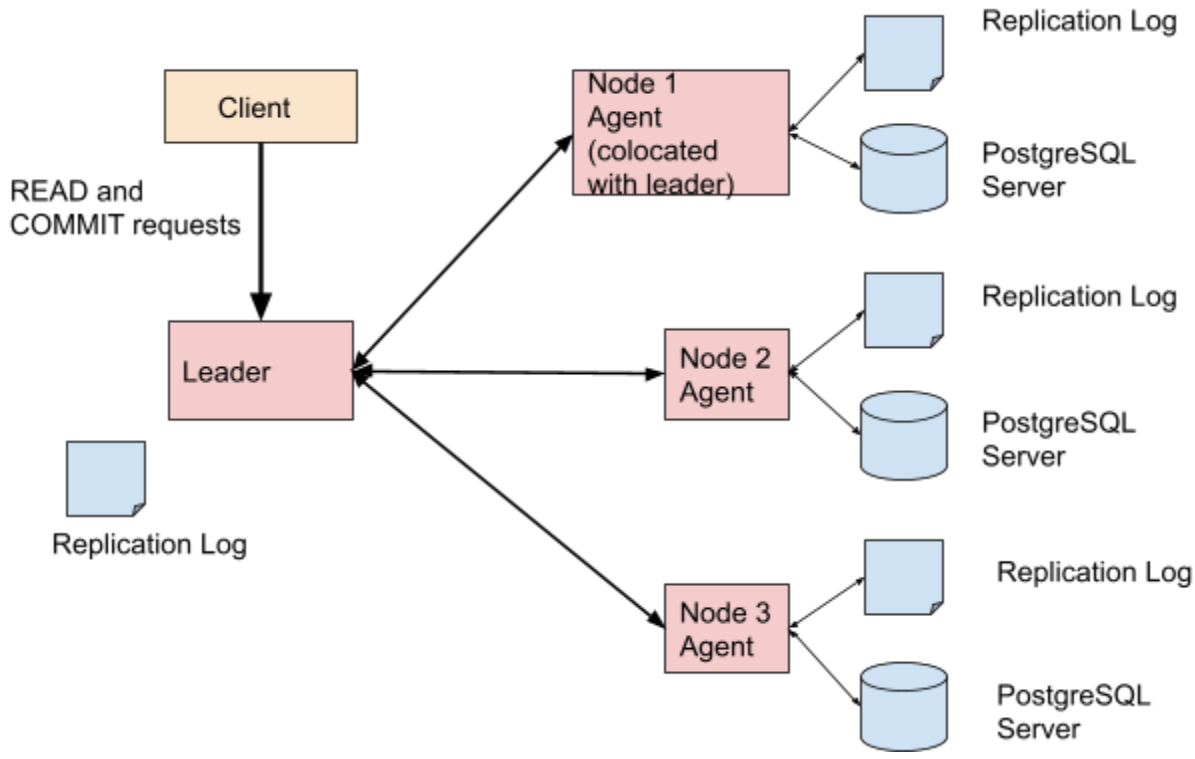
Introduction

In this project you will implement data replication of transactions on top of PostgreSQL. Data is replicated to 3 copies/servers. Each of the nodes will be running a PostgreSQL server along with an agent. You can mimic a replicated system by creating multiple PostgreSQL servers (with different data directory) and agent instances on your machine.

Let us consider an example to illustrate how transactions will run. The replication system has a leader (one of the three copies) and followers (the other two copies). A client makes a connection to the leader to send transactions. The client processes transactions in an optimistic concurrency control way. It sends read operations to the leader and buffers write operations. Once the transaction completes processing, then the client sends a COMMIT request that includes both the read and write sets.

The leader receives two types of requests from clients. A READ request is processed by returning the most recently committed value of the requested data object. A COMMIT request is processed by check whether the transaction can commit or not. If the transaction cannot commit, then an ABORT message is sent back to the client and the transaction is discarded. Otherwise, the transaction COMMITs by applying its writes to the database, responding to the client with a COMMIT decision, and appending the transaction's write-set to the "Replication Log".

After a transaction commits, the leader sends the corresponding entry in the Replication Log to the followers so that they can append the entry to their logs as well (the entry should be at the same log position in all nodes) and apply the write-sets to their databases.



In addition to the basic design above, extend the prototype with two features (1) read-only transactions, and (2) leader replacement.

(1) Read-Only Transactions
Extend the agent at nodes to enable performing a read-only transaction at any node (including a secondary node) while maintaining conflict serializability.

- Describe the algorithm to process read-only transactions.
- Prove that conflict serializability is maintained.

(2) Leader replacement
Extend the agents to be able to tolerate the case of a leader failure. If a leader node fails, then you should be able to replace the leader with a new node. This includes making the new node recover the state by asking the other nodes for information about transactions and the Replication Log.

Important Notes:

- The leader in your case is fixed to be the node which submits the inserts to other nodes.

- You **do not** need to change any of the PostgreSQL's code and the protocol implementation should be done as a layer running on top of the PostgreSQL servers.
- In order to test failure scenarios, you only need to mark a node agent as failed. I.e. it stops sending and receiving messages.

Deliverables

Submit on canvas, a zipped file containing **code, report** and **README** file with the steps to compile and run your code. README should clearly mention if any third party software that is required to be installed. The report should be of 2-3 pages and should mention the design of your code and answers the questions in this document.