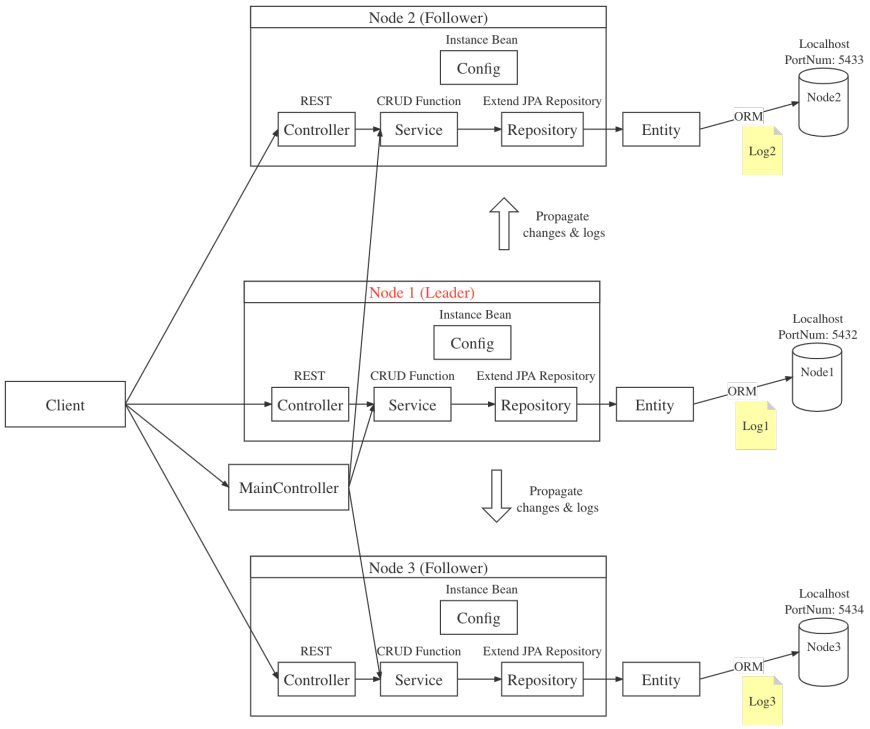
ICS223 Project

Wangyang Xu , Junhao Dong, Xuesong Bai

June 2022

1 System Architecture

In this section, the system architecture and corresponding system components are introduced below.



- **Postgres DB**
  In this project, the databases are generated by using Object Relational Mapping(ORM).There are 3 database servers and each database has its own port number (from 5432 to 5434). For each table in the database server, there are two columns, "id" and "message". For each agent, it connects to a separate data server.

- **Log File**
  For each agent, it has a log file that records all updated records in the database. Before the transaction successfully commits/aborts, the operations will be written into the log file and the log file is stored in a fixed location.

- **SpringBoot Module**
  For each agent, it has a springboot module that includes four layers, "Repository", "Service" and "Controller" and "Configuration". The "Repository" layer extends JPA repository to provide interfaces of accessing and dealing with data in the database.[1] For the "Service" layer, it invokes the

interfaces from "Repository" layer to implement CRUD. After that, each service needs to be instantiated to a JavaBean and the implementation of "Transaction Manager" and "Entity Manager" in the "Configuration" layer. Additionally, based on the previous layers, REST API has been provided in the "Control" layer.

- **Operation Respond**
  For all of operations, if they have been successfully implemented, the client will get a "200" status code. If the client gets other status codes, it means that the operations fail.

2 Read-only transaction

To support read-only transaction processing, a explicit lock is introduced. The lock is similar to the original shared lock but it can only be lifted. Modifying the lock or upgrading the lock will not be accepted. With this method, we can enforce the data to be consistent during the read-only transaction. During the process, we found out that the key factor to maintain the consistency of the schedule is a combination of some important transaction operation. Take the bank transfer transaction for example, in the transaction $W_A(x)$, $r_B(x)$, $w_A(y)$, $r_B(y)$, the order of the operation decides that T_B is reading a partial effect of T_A. If we ensure the order of all key operations to be the same as the order in some serial schedules, the consistency of the database can be enforced.

To achieve the check, we implemented an algorithm. Firstly, every transaction participating in schedule S will create a node T in the serialization graph. For each read-after-write conflict, an edge will be added to the graph T_p to T_q where T_q implements a read operation after T_p finished write operation on the same object. For each write-after-read or write-after-write conflict, an edge will also be added. Whether the graph is acyclic can be tested using a graph algorithm of $O(n^2)$. Using this method, we sacrifice some concurrency performance in exchange for conflict serialization.

3 leader replacement

We will introduce relevant algorithm that can be helpful to deal with leader replacement

1. **Prerequisite**
   As the project mentioned, we can mark a node agent as failed anytime. Therefore, we need implement Write ahead log to ensure the latest logs are stored persistently.

2. **WAL feature**
   It provides backup logs when leader node crashed. The new leader node can fetch logs directly and reconstruct the systems.

3. **New leader election**
   By comparing the length of logs between the remain nodes, select the node with a longer log as the new leader. In case they can the same length of log, just randomly pick up a node as the new leader. Afterwards, the new leader should send a 'PREPARE' message to the other node, Once it receive 'ACK' message, then the election is finished

4. **Recovery algorithm**
   Due to WAL feature, leader's log continued logging the operations before leader crashed. By comparing the original leader's log, the new leader can find out which transactions are committed or still active by itself. Then, it begins to redo all the transactions, and undo the active transactions following the core concept of Aries Recovery

5. **Replication Phase**
   After new leader recover from the old leader's log, it send entry of its logs to the followers so that they can receive the latest logs and apply the corresponding operations to their databases

6. **Old leader Re-connection**
   If old leader reconnect in a while after its crashed, it will send the 'CHECK' message to other nodes, and receive the new log entry from the new leader. Then, it can recover according to the latest logs.

References

[1] Boot, S. Spring boot reference documentation, 2022.