CS 410/510
Large Scale Systems

Assignment 3:
Implementing MiniSpanner

Overview:
In this course, our goal is to create **MiniSpanner**. MiniSpanner takes inspiration from **Google's Spanner** database. Despite being created nearly two decades ago, Spanner still acts as the backbone for all Google's applications.

In this course, over **a span of four assignments**, you will be creating MiniSpanner. Thus, all the assignments are **incremental** as you will get an opportunity to use your existing code in your current assignment.

Goal:
The goal of your third assignment is to implement a sharded replicated architecture on top of Apache ResilientDB (incubating).

CS 410 students will add to their sharded replicated architecture their previously implemented **Two Phase Commit (2PC)** protocol.
CS 510 students will add to their sharded replicated architecture their previously implemented **Three Phase Commit (3PC)** protocol.

Key Challenges:

- ResilientDB, by default, assumes a replicated system. However, the goal is to create a sharded-replicated architecture.
- In a sharded-replicated architecture, the system consists on multiple shards and each shard manages a distinct set of data-items. Within each shard, the data is fully replicated. Among the replicas, one of the replicas acts as the leader. The leaders of all the shards participate in 2PC/3PC protocol to decide on the fate (commit or abort) of a distributed transaction.
- Like in any partitioned/sharded system, the leader of each shard (in this assignment) can receive the client transaction. The leader that receives client transaction acts as the **global-coordinator**, initiates concurrency control and commit protocol.
- ResilientDB, by default, runs PBFT protocol and does not have any notion of a concurrency control and commit protocol. ResilientDB's PBFT implementation

assumes that the proxy collects all the client requests and forwards them to the leader to initiate PBFT consensus. Post PBFT consensus, each replica of ResilientDB executes the transaction and replies to the client.

Key Expectations:

- You are **expected** to modify the architecture of ResilientDB to implement the sharded-replicated architecture.
- Such a change may require adding some additional threads and functions to ResilientDB.
- In practice, the number of shards and number of replicas per shard are dependent on the application. However, you **can assume** that these numbers are fixed; the number of shards = 4 and the number of replicas per shard = 4.
- You are **expected** to modify the proxy to allow it to send client requests to the leader of different shards. Notice that in the current ResilientDB architecture, the proxy sends requests to only one replica (that is the leader). In this assignment, you must **ensure** that the proxy alternates sending the requests.

  For example: first batch of request is sent to the leader of first shard, second batch to the leader of second shard, third batch to the leader of third shard, fourth batch to the leader of fourth shard, fifth batch to the leader of first shard, and so on.

- Like Assignment 2, you **can assume** that the leader (shard) that receives the transaction acts the coordinator (coordinating shard) and all the other shards are participating shards. Thus, every shard participates in each transaction; each transaction requires access to 4 partitions/shards.
- Once the coordinator receives a transaction, it should **initiate** 2PC/3PC protocol. During this 2PC/3PC protocol, you **should forward** the transaction to the leaders of other shards.
- You are **not expected** to run or implement any concurrency control protocol.
- Once the commit protocol is finish, the leader of each shard should forward this transaction to its replicas. Ideally, forwarding the transaction to each replica implies the leader initiating the PBFT consensus protocol among its replicas. Post PBFT consensus, only the replicas of the coordinator shard forward the response to the client.

  In your assignment, if **you find it hard** to implement the above step, then **you have two options**:

  1. You can avoid leader forwarding request to all the replicas. If you do so, then you should make all the leaders execute the request (post commit protocol) and

reply to the client. Notice that you need minimal to no changes in execute and response functions but need to call them explicitly.

2. You can forward the request to the replicas and ensure that only replicas of the coordinating shard execute the request and reply to the client.

Assignment Tasks:
Following are the tasks that you need to perform as part of this assignment.

1. Implement your architecture.
2. Measure the throughput of your commit protocol at the coordinating shard.

Deliverables:
As part of this assignment, you are expected to provide the following deliverables.

1. A latex report explaining how you implemented sharded-replicated architcture. Please provide details of what functions you modified or added to the existing codebase.
2. In your latex report include snapshots of any function that you modified. These snapshots should show your changes.
3. Please state your machine configuration and the observed throughput/latency.