CSCI-1380: Distributed Computer Systems
Homework #3
Assigned: 04/11/2019
Due: 04/23/2019

1 Overview

This homework assignment is radically different from the other assignments: it requires you to think about different systems and design chocies and pick the ones that satisfies your goals. Unlike prior homeworks, this homework will be an entirely open-ended exploration of a distributed systems problem.

In this homework you will be designing your own Distributed System for the provided application. You have the following tools at your disposal when designing your system:

Distributed Datastore (e.g. Cassandra, Dynamo, MegaStore, GFS – L16-L18): You have access to a Data Management System (DMS). This DMS supports the ability to replicate data across multiple data centers for redundancy.

The throughput for reads and writes linearly increases with the number of machines in the system.

Consistency can be increased at the cost of availability and partition tolerance. Similarly, availability can be improved at the expense of consistency.

Replication (e.g. Raft, Lazy –L9-L13): The above data-stores come with predefined replication schemes and transactional semantics, however, you are free to alter and enhance their guarantees by changing replication paradigms and introducing transactional semantics (when missing). For example, "Dynamo with Raft for read/writes". If you choose to alter semantics, please discuss the implications of this design change: specifically, how does the design change impact performance, availability, and consistency.

2 Application

2.1 Social Banking

Design a distributed system for an online social banking site. This social banking site is a mobile-application. Users connect from around the world. The site supports two class of services:

1. **Financial-Transfers:** If a user has money in their account, they can transfer money to another. Your company will be liable to lawsuits if user balances are inconsistent. The user should also be able to query their current account balance. This service provides the following actions to users.
   (a) Transfer amount: Allows users to transfer amounts between two accounts.
   (b) View current balance: Allows a user to view their current account balance.
2. **Social Posts:** The service maintains a wall for each user. The wall comprises of all completed transactions which the user wishes to make public. Users can comment on the public transactions of other users. Due the online social network aspects of the posts and comments, users are willing to tolerate inconsistencies in posts and comment orderings. This service provides the following actions to users.
   (a) Make transfer public: Adds a successful balance transfer to the user's wall.
   (b) Comment on transfer: Allows a user to comment on a specific post on a user's wall.
   (c) View Wall: allows a user to view his wall or any user's wall.

3 System Design

1. Provide an overview of your distributed system. In the system, the users or client will interact with a FrontEnd service, this FrontEnd service will hide the complexity of your system from the users. The FrontEnd service will interact with the storage layer (simple key-value or advanced datastore). The FrontEnd will also include your application logic for interacting with multiple replicas, and interacting with a coordinator (for transactions). Be sure to include the tools you would use and a justification for each design decision.

   Be sure to discuss at least the following properties of your system and how they were implemented:

   - Fault-Tolerance
   - Availability
   - Consistency

2. Provide a design diagram describing the different work-flows, i.e., how a user interacts with a front-end, which in turn interacts with replicas (DMS). What happens for reads? What happens for writes? What happens during a network partition? It may be helpful to think about multiple users trying to read/write the same keys, e.g., write to the same account balance. For each service type, discuss interactions: reads and writes to the datastore to implement the action logic.

3. Why does your system work well for the expected use-case? How do your choices ensure that the system is fast (but acceptable)?

4. Are there failure cases your system is vulnerable to? How does your system prevent or gracefully handle these cases?

5. Hint: You may want to use different building blocks for different types of services. While you can use different building blocks for each service, the actions within a service must all utilize the same building blocks.

4 Handing In

Once finished, you should hand in a PDF with your answers on Gradescope. Gradescope will allow you to select which pages contain your answers for each part of each question.

Please do not put your name on any page of your handin! This will allow us to do anonymous grading through Gradescope.

Please let us know if you find any mistakes, inconsistencies, or confusing language in this or any other CS138 document by filling out the anonymous feedback form:
http://cs.brown.edu/courses/cs138/s19/feedback.html.