Project 4 Executive Summary:

Assignment Overview:

The idea behind Project 4 is that it builds on the previously developed distributed Key-Value Store (KVS) from Project #3 by introducing fault tolerance mechanisms and achieving consensus among replicated servers using the Paxos protocol. The main idea behind Paxos is to ensure a distributed system reaches consensus on a single value among multiple unreliable processes, even in the presence of failures. This enhancement aims to ensure the KVS remains operational and consistent even in the presence of server failures. The core challenge addressed in this project is the integration of the Paxos roles—Proposers, Acceptors, and Learners—into the existing system to manage consensus on state changes across the networked replicas, focusing particularly on PUT and DELETE operations. This is because PUT and DELETE modify data, whereas GET does not. A significant aspect of this project involves handling the random failures of acceptors to demonstrate the robustness and fault tolerance capabilities of Paxos in maintaining consistency and availability in a distributed system. This was achieved via random probability.

Technical Impression:

The transition to a fault-tolerant system using Paxos introduced significant complexity, particularly in managing the state and coordination among the different roles defined by the protocol—Proposers, Acceptors, and Learners. This is particularly challenging for me as I had never built such an implementation before. Implementing Paxos required a more in depth understanding of the consensus algorithms and its application within real-world distributed environments. This protocol was very interesting to learn about as it ensures that despite the presence of failures, the system can agree on a single order of operations to maintain consistency across the distributed state. The process of integrating Paxos was challenging but rewarding as it provided valuable insights into the design and operation of resilient distributed systems that can withstand and recover from node failures.

On the other hand, handling the random failures of acceptor threads posed some interesting challenges, requiring dynamic recovery strategies to ensure that system state could be correctly reconstructed and consensus operations could resume without losing critical data in the process of doing so. Implementing this was a different story, but I ended up using probability to simulate these random failures. Developing mechanisms for failover and recovery was an interesting way to learn the critical importance of designing systems that are not only fault-tolerant but also capable of maintaining their operational integrity under adverse conditions. The practical application of these concepts through the development and testing of the system was highly beneficial, reinforcing the critical role that such fault tolerance mechanisms play in the scalability and reliability of modern distributed applications. This was a new area of exploration for me, and a highly rewarding one too.

Use Case / Application:

In the biotech industry, implementing Paxos within distributed systems can significantly enhance the reliability and accuracy of data management during clinical trials, as accurate (validated) systems are standard practice in this industry. For instance, a Paxos-based system could be used to manage and synchronize patient data updates across multiple data centers that are part of a global clinical trial network. By ensuring consistent and fault-tolerant data management, Paxos helps prevent data inconsistencies that could arise from network failures or discrepancies in data handling protocols across different locations. This capability is crucial in clinical trials where real-time data integrity is essential for accurate monitoring and evaluation of patient responses to treatments.

Saleh Alkhalifa
Apr-12-2024

Project 4 Executive Summary: