Tribune: An Externally Consistent Database Common Access API for the Global Data Plane

Matt Weber, Shiyun Huang and Lawrence Supian
Department of Electrical Engineering and Computer Sciences (EECS)
University of California, Berkeley
Email: matt.weber, jane.huang, lsupian@berkeley.edu

***Abstract*—The Global Data Plane (GDP) is a distributed data storage system with the objective of providing persistent data storage to devices in the internet of things. The GDP has a variety of useful properties including data security, a flat namespace, and location independent routing, that improve the availability and efficiency of universal data storage. As a key component in the Terraswarm project and the SwarmOS, the GDP provides log based storage and routing between sensors and actuators embedded in the physical world. Logs are the fundamental primitive of the GDP, but some CAAPIs (a Common Access Application Programming Interface) require stronger semantics. In this project we will construct a transaction manager that implements Google Spanner-like external consistency semantics for multi-writer logs in the GDPs distributed environment. We compare the trusted Paxos based replication scheme used by Google Spanner to an optimistic concurrency based technique that is compatible with attack tolerant Byzantine Agreement replication. Our long term goal is to provide Ptides style deterministic execution and sensor-to-actuator timing guarantees for swarmlets in the SwarmOS that choose to access time-aware multi-writer logs.**

I. Introduction

Section II gives an overview of the system architecture and design. Section III goes into the implementation details of several crucial algorithms. Section IV elaborates on our test environment setup and the benchmark results. Section V gives the conclusion and addresses the direction of future work.

II. System Architecture

Our first design of system emulates a simplified version of Google Spanner without any **"distributed transaction"**. Because we opt not to handle **"distributed transactions"**, we do not implement the transaction manager which coordinates two phase commits between different Paxos groups. Our first design of system assumes an one-paxos-group environment so we only implement the lock manager at the leader replica.

The responder receives transactions from client applications. It parses each operation line by line and acquires read locks for all data required for the transaction. The responder buffers writes locally. When all the transaction operation finishes, the responder tries to acquire write locks for the locally modified data. If the attempt turns out successful, the responder will try to commit the transaction at the leader.

We only implement the necessary components for read-write transactions. So all transactions would go through the leader replica to validate their respective lock leases before committing via paxos protocol. If the validation is successful, the leader would go through all paxos phases and return the final transaction status (abort or commit) back to responder. The responder will return the transaction status back to client apps.

Below is a diagram for the overall architecture:

//then talk about modifications on the first version to incorporate optimistic concurrency control and byzantine agreement

III. Algorithms and Implementation

In this section, we're going to elaborate on the algorithmic part of the project, which is our choices of concurrency control protocol and commit consensus protocol. We did strict two-phase locking and optimistic concurrency for concurrency control. We also did paxos and byzantine agreement for commit consensus protocol. We would explain in detail how we implement four algorithms in our project and compare the tradeoffs in this section. The benchmark results would be in the next section.

A. Strict Two-Phase Locking

```java
// Hello.java
import javax.swing.JApplet;
import java.awt.Graphics;

public class Hello extends JApplet {
   public void paintComponent(Graphics g) {
      g.drawString("Hello, world!", 65, 95);
   }
}
```

B. Optimistic Concurrency Control

Subsubsection text here.

C. Pseudo Paxos vs Pseudo Byzantine Agreement

Subsubsection text here.

IV. Experiments and Results

Subsubsection text here.

V. Conclusion and Future Work

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.