Fault-Tolerant Distributed Declerative Programs

INF-3981 Master thesis for Moritz Jörg

Advisor: Weihai Yu

1 Introduction

We generate and consume data all over our computing devices, which can be online or offline at times. Ideally, the data are stored on the most appropriate devices and we are able to access the data even when the devices are offline.

Data is typically managed by a database management system and accessed through programs in a data manipulation and query language. SQL and Datalog are examples of declarative data manipulation and query languages.

The systems that process the data should have high availability and low latency in the face of in the face of server and network failures. With the ability to recover without the running system to ensure minimal loss of performance and correctness. and accuracy.

2 Problem definition

We are currently designing PRAD [1], an extension to Datalog, where we can turn a normal Datalog program into a distributed one, such that the data can be partitioned and replicated at specified devices. The data are always accessible at the specified devices, even when they are offline. The programs are eventually consistent. That is, when the devices are connected, the results that the distributed program generates, will eventually be equivalent to the ones that a normal non-distributed program does.

One of the key design goals of PRAD is fault tolerance through replication. When a replica is down, a PRAD program can continue to run, but the replication degree decreases and so does the number of site-crashes the program can tolerate.

3 Methodology

In the thesis we will be applying the Software engineering principles and practices, which form the backbone of any robust development process. Focusing on efficiency, maintainability, and reliability throughout the software lifecycle. The semester will be diveded into sprints, each containing certain milestones as shown in Section 5.

4 Objective

In this Master thesis, Jörg will work on repairing PRAD programs at runtime to maintain the fault-tolerance degrees.

5 Schedule

The following schedule roughly outlines the work items for the thesis. It will start on January 8th, 2024 and last for 6 months, until June 3th 2024.

- **January 2024**: Initial design and implementation.
- **February 2024**: Implementation and testing.
- **March 2024**: Evaluation and refinement.
- **April 2024**: Evaluation and reporting.
- **May 2024**: Finalzing and reporting.

References

[1] O. Qayyum and W. Yu, "Toward Replicated and Asynchronous Data Streams for Edge-Cloud Applications", in *37th ACM/SIGAPP Symposium on Applied Computing (SAC)*, ACM, 2022, pp. 339–346. doi: 10.1145/3477314.3507687.