Department of Computer Science

University of Kaiserslautern

Master Thesis

Offline caching in web applications for AntidoteDB

Server Khalilov

red17electro@gmail.com

University of Kaiserslautern
Department of Computer Science
Software Engineering

Leader:
Prof. Dr. Arnd Poetzsch-Heffter

Supervisor:
Dr. rer. nat. Annette Bieniusa

Technische Universität
Kaiserslautern

Zusammenfassung

Zusammenfassung auf deutsch

Abstract

The purpose of this thesis is to explore the possibilities of developing an offline web application, which would serve as a client for the AntidoteDB database. We developed a prototype of the application and designed the architecture of the application in such a way that both offline and online functionalities are possible.

To model the data stored offline in the local database of a web-browser, Conflict-free Replicated Data Types (CRDTs). It lets to ease the task of merging the changes.

Apart from that, the client-server protocol was designed, in order to support the functionality of the application. The paper could be divided into two parts: firstly, the problem of designing mentioned solution is going to be discussed. Secondly, there is an implementation part and a description of how specific problems were tackled.

Acknowledgement

I would like to take this opportunity and express my biggest gratitude to everyone, who supported me through all the ups and downs I had during my time at the Univesity of Kaiserslautern.

Firstly, I am very grateful to Prof. Dr. Arnd Poetzsch-Heffter and my thesis supervisor Dr. rer. nat. Annette Bieniusa for making it possible for me to work on the topic that suits my area of interest. Moreover, I want to thank Dr. rer. nat. Annette Bieniusa for having the door to her office always open for a discussion, whenever I needed it. I could not ask for more.

Next, I would like to thank the stuff at Software Technology Group, who made me feel very welcome and patiently answered any questions I had. Especially, Mathias Weber, Peter Zeller, and Deepthi Akkoorath.

Finally, I want to mention my family and, particularly, my parents. They did their very best to encourage and motivate me throughout the whole period of my study here. Nothing of it would ever be possible without them. Thank you.

Ich versichere hiermit, dass ich die vorliegende Masterarbeit mit dem Thema „Offline caching in web applications for AntidoteDB“ selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.
Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, habe ich durch die Angabe der Quelle kenntlich gemacht.

Kaiserslautern, den 15. December 2018 Server Khalilov

Contents

1 Introduction

In this chapter we are going to discuss the motivation, research questions and the scope of this thesis.

o Problem context (What is the broader context of this work?) o Problem description (What is the specific problem or challenge addressed by this work?) o Research question(s) (What are the actual questions that should be answered by this research? Why are they interesting? To whom? Why is there no obvious answer? If this is getting too much, then details of discussion can be deferred until later, e.g., to the methodology section.) o Summary of results and contributions (What is the outcome of this work in simple terms?) o Structure of the thesis (Introduce briefly the remaining chapters.)

1.1 Motivation

The motivation of this thesis is to explore the possibillities of implementing a web-client for the AntidoteDB with a support of caching and by utilizing the main features of the AntidoteDB.

1.2 Research questions

The following research questions are going to be addressed in this thesis:

- **RQ1.** How efficient is it to use a web-client with cache rather than without it?
- **RQ2.** What are the methods available to implement web-applications that would be able to work off-line and in the conditions of poor network connections?
- **RQ3.** What could be a scalable solution for transmitting CRDT data between a server and clients?

1.3 The structure

The structure of this thesis will be divided into the following subsections:

- Description of the main requirements of the application;
- Design of the architecture
- Description of the implementation phase
- Evaluation
- Conclusion

2 Background

2.1 Related Work

This chapter is needed, if the thesis is somewhat research-oriented. (This should be usually the case.) In some cases though, the thesis may also be more application- or implementation-oriented, in which case a designated related work chapter may not be necessary, but instead citations are just used appropriately throughout the thesis, but specifically in the chapters Introduction and Background. Literature search could be based on DBLP.

Maybe you can mention SwiftCloud here.

https://github.com/SyncFree/SwiftCloud

3 Requirements

In this chapter, we will describe the requirements for the desired web-application.

3.1 Problem Description

The application should showcase the functionality of communicating with an AntidoteDB database while online. As well as that, it should be reliable, which means it should load instantly and work even in uncertain network conditions. The application should be able to respond quickly to user interactions, be highly available (possible to work with it even when there is no internet at all).

4 Design

In this chapter, we are going to describe the design part of the solution.

4.1 Main components

First of all, let us briefly have a look at the main components:

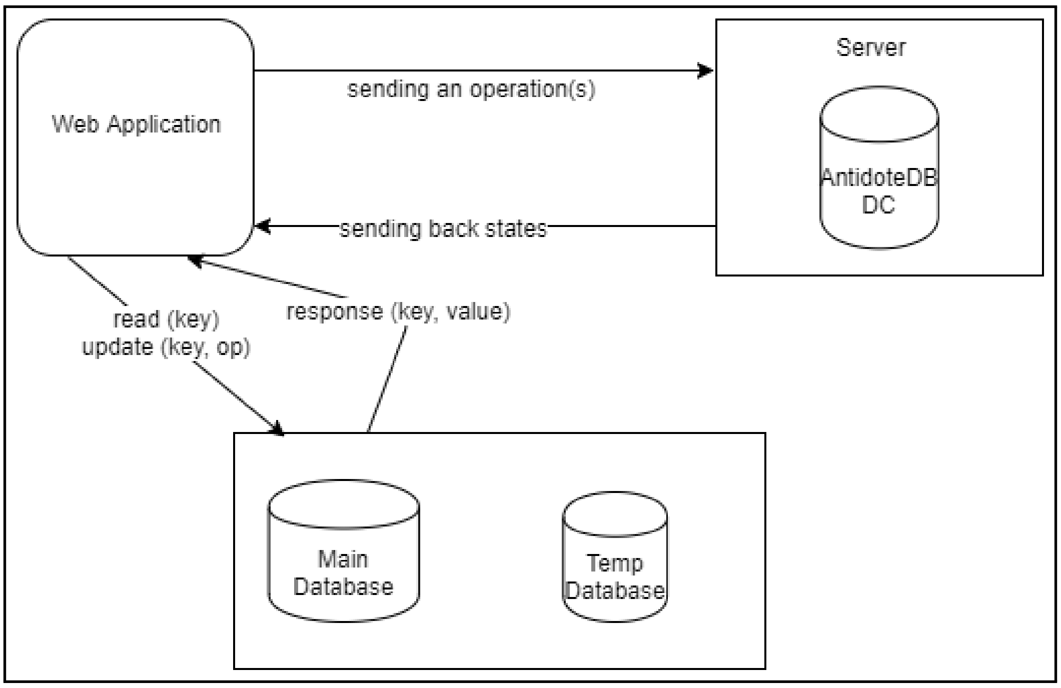

Figure 4.1: An overview of the system's design.

Web Application

It is a client application, which runs in the web-browser and supports interactive commands performed by user. It sits on top of the database layer.

These are the supported commands:

- *read(key)* – an asynchronous function that pulls database changes concerning the key that the user passed.
- *update(key, op)* – an asynchronous function that processes user-made update.
  - *key*: a key, which is going to be updated;
  - *op*: an operation, which is going to be performed on the key;

Server

It is a configured AntidoteDB server that supports the following scenari-os:

- receiving an array of operations performed on a CRDT-object according to the key and applying them on the server;
- sending back to the client the state of requested CRDT-object / objects according to their state on the server;

A database layer

This layer should consist of two databases - *main database* and a *temporary* one.

- Main database: this database stores the actual states of CRDT-objects.
- Temporary database: this database stores user-added operations performed on the states, which are stored in the main database.

When a user performs *read* by *key* operation from cache, the following actions are taking place:

- Firstly, the state of the object ***O*** is going to be found by ***key*** in the *Main database*
- Next, from the *temporary database*, operations ***o*** performed on the object ***O*** are selected
- Afterwards, selected operations ***o*** applied on the object ***O***.

Listing 4.1: The example behaviour for the function, which should run at the start of the application.

```
// starting point of the application, which is going to be
    called on document onload event
function start(){
    read(); // read the latest changes.

    // add listener for a custom event 'add-data' (when the
       user tries to add new changes)
    addEventListener('add-data', callback);

  callback = function(){
        update();
  }
}
```

- Finally, the object from the previous step is returned back as a response to the application.

Above you can look at the function, which should run at the starting point of the application.

Listing 4.2: The example behaviour for the function, which should run at the read time.

```
// Read function that pulls database changes
// @param key: the key of the object, for which the update
    was requested;

function read(key) {
    responseArray = []; // define an empty array, which is
        going to be sent back

    state = get state of object o by key from the main
        database
    operations = get operations performed on the object o
        from the temp database

    latest state = apply operations over the state to get
        latest state of object o
    responseArray.push(latest state);

    return reponseArray;
}
```

Listing 4.3: The example behaviour for the function, which should run at the update time.

```
// update function that processes user-made update
// @param key: a key for the object that should be updated;
// @param op: operation performed on the object for the
    specified key;
// TODO: add the support for multiple changes also!

function update(key, op){
  // as we need to have operations sorted, it should be added
      to the temp database in the following way:
  // {key, [op1, op2, op3, ... ]}
  add op to the temp database for the found key;
  return responseStatus;
}
```

5 Technologies

This chapter consists of detailed description of used technologies to accomplish thesis goal.

5.1 Service Workers

5.2 IndexDB database

6 Architecture

Architecture

6.1 General overview of the architecture

6.2 Communication protocol description

7 Implementation

Implementation

This chapter is needed specifically, if the work is meant to deliver an actual implementation of some software system. No low level details or extensive code fragments should be included, but non-trivial implementation issues are to be explained, if they could not be reasonable covered at the design level; see Chapter Design. Many thesis do not need an Implementation Chapter because implementation details can be deferred to software documentation or the appendix. Also, some illustrative details of implementation may be placed in other chapters.

8 Evaluation

Evaluation

This chapter is needed, when results from the previous chapters need to be systematically discussed. This is the case, when an implementation needs to be assessed, or the results of a case study or an experiment need to be interpreted.

9 Conclusion

Conclusion
This is always the first chapter of the thesis. The chapter should be short (up to 5 pages). The chapter should feature sections as follows (where applicable): o Summary (Summarize this work in an insightful manner, assuming that the reader has seen the rest.) o Limitations or threats to validity (Point out the limitations of this work. In the case of empirical research, discuss threats to validity in a systematic manner.) o Future work (Provide insightful advice on where this research should be taken next.)

9.1 Summary

9.2 Future Work

List of Figures

List of Tables

List of Code

Bibliography