tgm
Die Schule der Technik

JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

Systemtechnik Theorie

xHIT 2024/25, Gruppe A

Protokoll Zlabinger 5DHIT

Theorieprotokoll

Christof Zlabinger 5DHIT

19. Dezember 2024

Bewertung:		Version:	1.0
Betreuer:	Michael Borko	Begonnen:	05.09.2024
		Beendet:	XX.XX.XXXX

Inhaltsverzeichnis

lisitings

1 Verschluesselung

Symetrisch mittels einem Key
Asymetrische als verstaerkung mittels public/private key
Checksums z.B.: Sha256, Sha512, md5, …
Bei veraenderung komplett anderer Hash
Es kann trozedem sein dass 2 verschiedene daten gleichen Hash haben
Bei Sha256 groessere wahrscheinlichkeit als bei Sha512 weil weniger bits

2 BUS

SPI I2C BUS
BUS controller um collisions zu vermeiden
Wenn collisions -> Beide teilnehmer warten eine zufaellige Zeit

3 Konsistenz in Datenbanken

3.1 Konsistenz

3.1.1 Einheitlich

Datentyp Der Datentyp gibt an wie die Daten zu interpretieren sind und wie gross dieser ist. Fuer Konsistenz ist es wichtig dass die Interpretation gleich oder mapbar ist.

SQL subsprachen Data Definition Language Data Manipulation Language Data Query Language

3.1.2 Regeln (Constraints)

z.B: NotNull, Unique Helfen bei Konsistenz koesten aber Zeit (Alles ueber 0.5s lade Zeit ist lange)
CRUD = Create Read Update Delete

3.1.3 Deterministisch

Computer sind deterministisch gebaut/programmiert wesshalb es eigendlich nicht moeglich ist zufaellige Zahlen zu generieren. Desshalb wird PRNG (Pseudo Randon Number Generator) genutzt. Dieser verwendet Sensoren, deren Stand nicht vorhergesehen werden kann, wie z.B.: CPU temperatur, Fan speed, ...

4 Transaktionskonzepte

ACID = Atomacity Consistency Isolation Durability

Atomar Atomar bedeutet, dass eine Transaktion entweder ganz durchgefuehrt wird oder garnicht.

CAP CAP = Consistency Availability Partitiontolerant Es koennen hoechstens zwei dieser Ziele gleichzeitig erfuellt sein. Bei ACID muessen alle fier Ziele erfuellt sein.

Damit Daten konsitent sind muessen sie am Anfang und am Ende konsitent sein aber mussen bzw. koennen in der Mitte nicht konsitent sein.

4.1 Isolation Levels (psql)

Isolation Level	Diry Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read uncommited (Deadlocks moeglich)	Allowed, but not in PG	Possible	Possible	Possible
Read commited	Not possible	Possible	Possible	Possible
Repeatable read	Not possible	Not possible	Allowed, but not in PG	Possible
Serializable	Not possible	Not possible	Not possible	Not Possible

Es gibt einen Unterschied zwischen Lesen zum anzeigen und Lesen zum schreiben. Beim Lesen zum schreiben muessen die gelesenen Daten gesperrt werden.

4.2 Durability

Dauerhaftes Spechern der Daten. Wird sichergestellt dadurch dass die veraenderten Daten welche im RAM sind gespechert werden, dann auf die DB uebernommen werden und dann ueberprueft wird ob diese korrekt gespeichert werden koennen. Falls dies nicht der fall ist wird ein rollback durchgefuehrt.

4.3 Geschwindigkeit

ACID ist am sichersten aber ist auch am langsamsten. BASE = Basically Available Soft-state Eventually consistant

BA: Atomates speichern der Anfang in einer Queue und deren sequenzielle ausfuehrung. S: Nach einer gewissen Zeit in der queue werden die Anfragen getimeouted. E: Wenn alle Anfragen bearbeitet wurden ist die Datenbank konsitent.

Jede Anfrage kann von BASE mit der Hilfe von ACID bearbeitet werden.

5 Build prozess

.h/.hpp files (Header files) beinhalten die Struktur des dazugehoerigen .c/.cpp files. Der inhalt der header files wird vom pre processor in das C/C++ file kopiert. Da C nicht objekt orientiert ist gibt es kein method overloading.

5.1 Prozess

Das .h/.hpp file wird vom pre processor in das .c/.cpp file kopiert. Der compiler verwandelt dieses dann in .o/.obj files. Der Assembler verwaltet das startup.a/.s file woraus der Linker dann ausfuehrbaren code macht welche mittels des Flash tools geflashed werden.

6 CAP-Theorie

Server = Diener

RPC, RMI, gRPC sind remote prozedure call implementationen welche fuer mittelware verwendet werden koennen. MVC = Model View Controller MVVM wird fuer web based implementations verwendet. DAO = Data Acess Object welches ueberprueft ob bestimmte Daten korrekt sind.

6.1 CAP

Consistency, Availability wurden bereits gemacht Partition tolerance bedeutet das sub-systeme eigenstaendig funktionieren und die Daten nach einem Ausfall wieder miteinander syncen. Es sind immer nur 2 der 3 CAP Ziele gleichzeitig moeglich.

7 Timer, Delay, FPU

7.1 Cpu Cycles

Ein delay kann mittels zaehlen von CPU Cycles hergestellt werden. CPU clock speed / f/2

7.2 Timer

Timers haben ein Limit wie klein der delay sein kann wodurch ein offset entstehen kann der dann haendisch ausgebessert werden muss.

8 DiffSync

8.1 Introduction

Collaborate in real time Syncronization algorithm in an application is hard to change DS can be used in existing applications

Suited for:

- unreliable/slow networks

- identical code on server/client
- No history required
- Scalable

Use cases:

- Pair programming on different systems
- Remote debugging

No problem when collaborating with someone or yourself with autosave feature because all versions are kept in sync

8.2 Alternatives

Pessemistic Only edit by one user at a time Whole file is locked for other users Refined version would be to only lock subsections but created problems for small files

- Has to be supported by the application

Not good for unreliable networks

Edit-Based Capture *all* edits (Typing, cut, paste, formatting, ...) and mirror them across all devices If an edit is lost it will create a fork -> two different files

Three way merge Client sends changes to server -> server performs 3 way merge -> merge gets send to the clients If someone changes something while someone else is changing something it will result in a conflict Not scalable

8.3 Dual shadow method

Client text and server shadow must be identical every half syncronization or server text and client shadow

Checksums after patch must be identical otherwise entire body must be sent again

8.4 Topology

Multiple clients connected to a single server. The number of clients can become an issue -> load balanced servers with shared DB Or server to server communication (two or more servers are connected with eachother). Servers can be added without a problem but can only be disconnected if no client is connected. Latency may become an issue -> Can lead to complicated collisions. Can be reduced by avoiding long server chains and by increasing the sync time between servers

8.5 Diff/Patch

Diff/Patch algorithms are crucial. Efficiency improvements in these algorithms boost system performance. Can be used for plaintext and many more. Diff updates server shadows with client changes. It needs to differentiate minimal edits from semantic intent ensuring meaningful updates. Compares identical texts efficiently. Identifies shared starting/ending strings to speed up comparisons. Detects if a change is an addition or removal bypassing complex calculations.

Patches apply accurately, even if texts aren't exact matches, using the Bitap algorithm for efficient near-match location. Merging non-text content may use a "last user winsäpproach to prevent incorrect combinations. Handling Collisions: Systems can address patch errors either by frequent, silent synchronization or user intervention on failed patches, balancing usability with data accuracy.

8.6 Adaptive timing

Client update frequency is critical for system responsiveness. Infrequent updates lead to costly operations and edit conflicts, while overly frequent updates strain the network. The Guaranteed Delivery Method reduces resource use by batching diffs for transmission. An adaptive system adjusts update frequency based on user activity, keeping updates efficient and responsive by shortening intervals during high activity and lengthening them when activity is low.

8.7 Future work

The fuzzy patch operation uses a three-way merge comparing Server Text and two versions of the Server Shadow. It only supports single synchronization packets limiting performance in high-latency connections. Additionally edits from all users are blended making attribution and individual rollbacks difficult.

9 Pico CAN

CAN Controller CAN = Controler Area Network

9.1 Pico Dev env for NixOS

Repo: https://github.com/czlabinger/NixPicoEnv/

In VSCode oeffnen und devpods extention installieren und starten. SSH auf den erstellten container. Das Template befindet sich in src und alles ausserhalb von src kann ignoriert werden. Die CMakeLists.txt anpassen um die gewuenschten projects zu importieren.

Building:
'sh src/build.sh'

9.2 Operation-Modes MCP2515

- **Normal Mode:** Standardmodus, in dem der Controller Nachrichten senden und empfangen kann.

- **Sleep Mode:** Energiesparmodus, in dem der Controller minimalen Strom verbraucht und keine Nachrichten verarbeitet. Ein externer Wake-Up kann den Controller wieder aktivieren.
- **Loopback Mode:** Interner Testmodus, bei dem gesendete Nachrichten direkt empfangen werden, ohne dass sie den CAN-Bus verlassen. Ideal für Tests ohne externes CAN-Netzwerk.
- **Listen-Only Mode:** Empfängermodus, in dem der Controller nur empfangene Nachrichten liest, ohne ACK-Bits zu senden. Wird oft für die Busdiagnose verwendet.
- **Configuration Mode:** Konfigurationsmodus, in dem Einstellungen wie Bitrate und Filter des Controllers festgelegt werden können.

9.3 Pins

MCP2515	Pico
VCC1(5V)	VBUS
3,3V(OUT)	VCC(3,3V)
PIN_MISO	GP4
PIN_CS_A	GP5
PIN_SCK	GP6
PIN_MOSI	GP7

10 Firebase

10.1 Clonen des Repos

```
git clone https://github.com/firebase/friendlychat
```

10.2 Erstellen der flake.nix für NixOS specific behavior

```
# flake.nix
{
  inputs = {
    nixpkgs.url = "github:NixOS/nixpkgs/nixpkgs-unstable";
    systems.url = "github:nix-systems/default";
  };

  outputs =
    { systems, nixpkgs, ... }@inputs:
    let
    eachSystem = f: nixpkgs.lib.genAttrs (import systems) (system: f nixpkgs.legacyPackages.${sy
    in
```

```
    {
      devShells = eachSystem (pkgs: {
        default = pkgs.mkShell {
          buildInputs = [
            pkgs.nodejs
            pkgs.nodePackages.firebase-tools
          ];
        };
      });
    };
}
```

Wird benötigt, um Dynamically Linked Libraries auf NixOS Systems verwenden zu können. Im Directory der flake ausführen:

```
nix develop
```

11 Start

11.1 Login in Account

```
firebase login
```

11.2 Authorize Firebase CLI

```
firebase use --add
```

11.3 Deploy App

```
firebase deploy --except functions
```

12 Install Dependencies

```
npm i
```

ausführen, um alle Dependencies zu installieren.

13 Setup im Webinterface

Im Firebase Webinterface musste der Storage erstellt werden und das Project angelegt werden.

14 Implement Functions

14.1 Enable Functions in Cloud Console

```
// Import the Firebase SDK for Google Cloud Functions.
const functions = require('firebase-functions');
// Import and initialize the Firebase Admin SDK.
const admin = require('firebase-admin');
admin.initializeApp();

// TODO(DEVELOPER): Write the blurImages Function here.

// TODO(DEVELOPER): Write the sendNotification Function here.
```

14.2 Welcome Message

```
// Adds a message that welcomes new users into the chat.
exports.addWelcomeMessages = functions.auth.user().onCreate(async (user) => {
  functions.logger.log('A new user signed in for the first time.');
  const fullName = user.displayName || 'Anonymous';

  // Saves the new welcome message into the database
  // which then displays it in the FriendlyChat clients.
  await admin.firestore().collection('messages').add({
    name: 'Firebase Bot',
    profilePicUrl: '/images/firebase-logo.png', // Firebase logo
    text: `${fullName} signed in for the first time! Welcome!`,
    timestamp: admin.firestore.FieldValue.serverTimestamp(),
  });
  functions.logger.log('Welcome message written to database.');
});
```

Deploy der neu erstellten functions:

```
firebase deploy --only functions
```

14.3 Exception beim Deployen

Dieser Befehl wirft eine Exception: `No targets in firebase.json match '–only functions'`
Versuchte Lösungswege:

- `firebase deploy functions`: Keine Error Message, nur ein Hinweis auf `firebase [command] –help`
- `firebase deploy`: Wirft zwar keinen Error, aber die Funktion erscheint nicht im Webinterface und wird auch nicht ausgeführt.

- `firebase deploy functions –add: unknown option –add`

15 Resources

15.1 Pico

Pico sdk: https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-c-sdk.pdf

CAN Controller: https://elearning.tgm.ac.at/pluginfile.php/116746/mod_resource/content/0/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf

Firebase: https://firebase.google.com/codelabs/firebase-cloud-functions#0