Anforderungen

VERTEILTE SYSTEME PRAKTIKUM

FABIAN EMIL ENGLERT, MATTHIAS CHRISTOPHER PETRI

Funktionale Anforderungen

Sensoren

- Sensoren liefern dem Zentralserver Informationen über den aktuellen Zustand des Fahrzeuges (Füllstand, Kilometerstand, Verkehrssituation, Durchschnittsgeschwindigkeit)
- Die Sensoren laufen als eigenständige Docker-Container
- Der aktuelle Zustand, der Sendezeitpunkt und die ID des Sensors werden mit CSV codiert und mittels UDP an die Zentrale übermittelt
- Später werden die Daten per MQTT an die Zentrale übertragen und dort verarbeitet

Zentrale

- Der HTTP Server in der Zentrale verarbeitet HTTP-GET, um dem Nutzer die gewünschten Sensordaten an den Browser zu senden
- Die Historie der Sensordaten wird von dem Zentralserver in einzelne CSV-Dateien abgespeichert
- Asynchrones Lesen und Speichern der einzelnen Sensordaten
- Die Zentrale soll die aktuellen Werte der Sensoren über RPC an die Cloud eines Drittanbieters (Serviceprovider) übermitteln

Serviceprovider

- Der Serviceprovider speichert die Daten persistent in einer Datenbank
- Die Server des Serviceproviders sind redundant ausgelegt, dies wird mit einem Hot-Standby für die einzelnen Server gelöst
- Der Serviceprovider betreibt mindestens 2 Server, um die Redundanz sicherzustellen

Nicht-funktionale Anforderungen

- Performance
- Responsivität
- Ausfallsicherheit

- Docker Version 19.03
- Docker-Compose Version 3.7
- CMake Version 3.13
- Programmiersprache C++ Version C++17
- Java Version 11
- Gradle Version 6.4
- Apache Mosquitto als MQTT Broker
- Apache Thrift
- Eclipse Paho Bibliothek für MQTT

Systemdesign

Sensoren

- Füllstand: unsigned int
- Kilometerstand: unsigned int
- Verkehrssituation: unsigned int
- Durchschnittsgeschwindigkeit: unsigned int
- Jeder Sensor hat eine eindeutige ID

Zentralserver

- Speichern der Werte in Historien für die einzelnen Sensoren (CSV)
- Weiterleiten der Daten an den Serviceprovider alle 10 Minuten

HTTP-Server

- URI für Zugriff auf die einzelnen Sensoren und auf alle Sensoren
- Kommunikation mit Serviceprovider mit RPC (Thrift)

Serviceprovider

- Speichern der Werte der Zentrale in Postgresql Datenbank
- Kommunikation mit HTTP-Server mit RPC(Thrift)
- Die Server werden mit einer eindeutigen Nummer identifiziert

Tests:

HTTP-Server

- Richtige HTTP-GET Anfrage verarbeiten
- Falsche HTTP-GET Anfrage verarbeiten und Fehler zurückgeben

- Die HTTP-Anfragen für den nicht funktionalen und Performance Test werden mit Apache Bench durchgeführt
- Performancetest: 100 Sensoren und 100 HTTP-Anfragen verarbeiten
- Performancetest: 10 Sensoren und 10 HTTP-Anfragen verarbeiten
- Nicht-funktionaler Test: Stabilität des HTTP-Servers mit Stresstest - 1000 HTTP-Anfragen mit jeweils 10 gleichzeitigen Clientverbindungen

Serviceprovider

- Speichern der empfangenen Daten in der Datenbank

Sensoren

- Erfolgreiches Verbinden mit dem MQTT Broker
- Stresstest des MQTT Subscribers mit 200 Fuelsensoren

Zentrale

- Performancetest: 100 Sensoren verarbeiten, deren Daten speichern und an den Serviceprovider übertragen