TECO Research Group

Marcel Köpke
Matthias Budde
Till Riedel

Implementierungsbericht

Version 1.0

Visualizing & Mining of Geospatial Sensorstreams with Apache Kafka

---

Jean Baumgarten
Thomas Frank
Oliver Liu
Patrick Ries
Erik Wessel

26. August 2018

Inhaltsverzeichnis

1 Einleitung

Dieser Implementierungsbericht dient als Übersicht der Implementierungsphase.

Die Implementierungsphase unseres Projekts wurde in zwei Hälften unterteilt, von denen die erste drei Wochen und die zweite vier Wochen dauerte. Bezieht man die zusätzliche Woche zur Klausurvorbereitung mit ein, summiert sich die Phasendauer auf acht Wochen.

In der ersten Hälfte wurde hauptsächlich...

In der zweiten Hälfte wurde schließlich...

Nachfolgend werden Abweichungen der Implementierung vom Entwurf analysiert, eine Übersicht zu Unit-Tests gegeben und auf die implementierten Muss- und Wunschkriterien eingegangen.

2 Änderungen am Entwurf

2.1 Bridge

- Um das Problem zu beheben, dass FROST in versendeten MQTT-Nachrichten nur `@iot.navigationLink`s für zusammenhängende Objekte angibt (statt einer Menge an `@iot.id`s), wurde eine neue Klasse `FrostIotIdConverter` erstellt, dessen Methoden genau diese Konvertierung bewerkstelligen.
- Das Format, in dem konvertierte MQTT-Nachrichten zu Kafka geschickt werden, wurde geändert (von `byte[]` zu einem Avro-Objekt). Dies hat folgende Auswirkungen auf dieses Modul:
  - Die `getSensorIdFromMessage`-Methode der Klasse `MqttMessageConverter` wurde entfernt.
  - Neun neue Klassen wurden hinzugefügt, die die zu versendenden Avro-Objekte repräsentieren.
  - Die Klasse `SchemaRegistryConnector` wurde entfernt, da die enthaltene Funktionalität nicht mehr benötigt wird.

2.2 Database

- Im Laufe der zweiten Implementierungsphase ist klar geworden, dass die Datenbank näher an dem Core arbeiten muss. Dadurch entfallen alle von `HTTPServlet` erbenden Klassen, da Zugriffe auf die Datenbank nun direkt von dem Core aus durchgeführt werden. Dennoch bleibt die Klasse `Facade` erhalten, um eine Implementierung der Servlets, sollte dies in der Zukunft nötig sein, ohne große Umstände zu ermöglichen.
- Alle Klassen, die für die Verwaltung von veralteten Daten zuständig sind, entfallen (also von `Maintainer` erbende Klassen und die Klasse `MaintenanceManager`). Memcached bietet die Möglichkeit, beim Hinzufügen eines Eintrags eine Zeit zu setzen, nach dem dieser Eintrag abläuft (d.h. gelöscht wird). Dies ist eine effizientere Lösung um alte Einträge zu entfernen als durch einen Maintainer.
- Da einzelne Datenwerte nun durch ein `ObservationData`-Objekt dargestellt werden, entfallen die Klassen `ClusterID` und `ZoomLevel`.

- Wegen obigem Grund wurde ebenfalls die Klasse `KafkaToStorageProcessor` in `ObservationDataToStorageProcessor` umbenannt. Diese bietet über die Fassade nun zwei Funktionen `add` und `get` für `ObservationData`-Objekte an.

3 Muss- und Wunschkriterien

4 Unit-Tests