Ein Bild, das Text, Schrift, Visitenkarte, Logo enthält.

Automatisch generierte Beschreibung

Duale Hochschule Baden-Württemberg Center for Advanced Studies

Laborarbeit

Modul Big Data Engineering (W3M20027)

Verfasser: Cenk Yagkan ()

Felix Schneider ()

Niko Sauter (9755750)

Nikola Mihajlovski ()

Professor: Prof. Dr.-Ing. Dennis Pfisterer

Abgabedatum: 19.01.2024

Inhalt

[1 Use Case Beschreibung 1](#_Toc153893065)

[2 Aufbau 1](#_Toc153893066)

[2.1 Generation 1](#_Toc153893067)

[2.2 Ingestion 1](#_Toc153893068)

[2.3 Stream Processing Layer 1](#_Toc153893069)

[2.4 Serving Layer 1](#_Toc153893070)

[3 Reflexion 1](#_Toc153893071)

# Use Case Beschreibung

Unser Usecase bezieht sich auf die Auswertung von Social Media Posts auf der ehemaligen Plattform Twitter. Die Auswertung umfasst die Stream-Verarbeitung simulierter abgesetzter Posts und das aggregieren der enthaltenen Hashtags, um diese später bezogen auf einen Zeitraum auswerten zu können. Somit kann z.B. eine Übersicht über die am meisten Verwendeten Hashtags erstellt werden. Als Beispieldaten wird folgendes, öffentlich zugängliches Datenset verwendet:

[Sentiment140 dataset with 1.6 million tweets (kaggle.com)](https://www.kaggle.com/datasets/kazanova/sentiment140/data)

# Aufbau

Dieses Kapitel beginnt mit einer Beschreibung der Architektur. Die Unterkapitel gehen daraufhin auf die einzelnen Bestandteile ein. Die Architektur orientiert sich am Big Data Engineering Lifecycle:

Transformation:

Abruf der Posts und Transformation durch Spark aus Kafka

Serving:

Bereitstellung der vorverarbeiteten Daten in einer MariaDB

Ingestion:

Aufnahme der Posts Message Queue System Kafka

Generation:

Simulation mittels Python Programm

Die gesamte Applikation wird über ein Helm-Chart und Skaffold gesteuert?

[Big Data Engineering - Application Deployment (farberg.de)](https://farberg.de/talks/big-data-eng/?03c%20-%20Application%20Deployment.md#/20)

## Generation

In unserem Beispiel wird die Echtzeitgenerierung simuliert. Ein Python-Programm schreibt vorgegebene Posts aus einer Textdatei periodisch in den Ingestion Layer. Dieses Programm und die Beispieldaten befinden sich in einem eigenen Docker-Container.

Das simulierte Quellsystem pusht die Daten zu Big Data Analyse Systemen. Dieses Vorgehen ist besonders geeignet für Real-Time-Anforderungen, wie es unser Use Case verlangt.

Publisher und Subscriber: [Big Data Engineering - Big Data: Data Ingestion Layer (farberg.de)](https://farberg.de/talks/big-data-eng/?04d%20-%20Data%20Ingestion%20Layer.md#/24)

Die Daten in Kafka, die Logs zu den Posts, haben folgendes JSON-Format (In den Klammern steht der Json-Key):

|  |  |  |  |
| --- | --- | --- | --- |
| ID (id) | Datum (timestamp) | User (username) | Post-Text (text) |

## Ingestion

Hierfür wird Apache Kafka verwendet, um als Big Data Message Queue System die Daten in fast Real-Time aufzunehmen und die Weiterverarbeitung zu ermöglichen. Dies ist für unseren Use Case von der Verarbeitungsgeschwindigkeit ausreichend und zusätzlich haben wir noch je nach genauer Konfiguration bedingt historische Daten zur Verfügung. Die Entkopplung von Erzeuger und Abnehmer wird so ebenfalls verwirklicht.

Wir haben bei den Beispieldaten einen Zeitstempel und können somit mit dieser Event Time für Auswertungen arbeiten. Die Reihenfolge partitionsübergreifend festzuhalten ist durch den Zeitstempel in den Daten selbst kein Problem.

Kafka: [Big Data Engineering - Big Data: Data Ingestion Layer (farberg.de)](https://farberg.de/talks/big-data-eng/?04d%20-%20Data%20Ingestion%20Layer.md#/21)

## Stream Processing Layer

Als Trigger für unsere Microbatches wählen wir ein festes Intervall. Es werden die Zwischenergebnisse der einzelnen Microbatches aufsummiert. Diese werden einer Datenbank gespeichert. Structured Streaming wird somit angewendet. Apache Spark wird verwendet, da es diese Microbatch-Anforderungen gut umsetzen kann.

[Big Data Engineering - Big Data: Spark: Structured Streaming (farberg.de)](https://farberg.de/talks/big-data-eng/?05d%20-%20Structured%20Streaming.md#/8)

## Serving Layer

Als Output-Mode wählen wir „Update Mode“. Aufgrund der Datenaggregation auf Tagesbasis können wir Append als Möglichkeit ausschließen.

MariaDB? Schema? ConfigMap um Datenbank zu konfigurieren: [Big Data Engineering - Application Orchestration (farberg.de)](https://farberg.de/talks/big-data-eng/?03b%20-%20Application%20Orchestration.md#/27)

|  |  |  |
| --- | --- | --- |
| Wort | Datum | Anzahl |
| #Elon | 18.12.2023 | 2 |
| #Tesla | 18.12.2023 | 5 |
| #Elon | 19.12.2023 | 15 |

Man könnte Wordcloud aus Daten in der Datenbank erstellen.

# Reflexion

Herausforderungen, Lernerfahrungen