Data Engineering mit Apache Kafka

Projektbericht

von

Johannes Weber

Matrikelnummer: 11010021

und

Julian Ruppel

Matrikelnummer: 11010020

09.02.2018

SRH Heidelberg

Fakultät für Information, Medien und Design

Big Data und Business Analytics

Dozent

Frank Schulz

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Aufgabe und Ziel

Im Rahmen dieses Projektes war es die Zielstellung sich mit Data Ingestion, Data Storage sowie Data Retrieval vertraut zu machen.

Data Ingestion ist die Beschaffung der Daten. Dies kann entweder mit Hilfe eines Data Streams erfolgen oder einer statischen Datenquelle - also eine einfache Datei die lokal auf einem Rechner angelegt wird und Daten beinhät wie z. B. eine Comma-separated values (CSV) oderJavaScript Object Notation (JSON) Datei. Unter einem Data Stream versteht mann einen kontinuierlichen Datenstrom wie z. B. die Erstellung von immer wieder neuen Twitter Nachrichten. Ein wichtiges Merkmal eines Data Streams ist, dass man nicht vorhersehen kann wann der Datenstom zu Ende ist - er könnte theoretisch unendlich sein. Im Falle von eines Datenstroms von Twitter Nachrichten ist es nicht abzusehen wann jemals die letzte Twitter Nachricht geschriebenb wird. Für unsere Aufgabe ist darauf zu achten, dass der Datenstrom über eine API öffentlich zugänglich ist und immer auf dem aktuellsten Stand gehalten wird.

Data Storage ist die Speicherung der Daten. Hierbei wurde uns lediglich die Anforderung gestellt, das wir für die Speicherung die Streaming Plattform Apache Kafka verwenden. Des Weiteren war es uns gestattet die Daten in einer relationen Datenbank, NoSQL Datenbank oder mit Spark Streaming speichern, falls nur die Nutzung von Apache Kafka unsere Anforderungen nicht genügt.

Data Retrieval ist die Beschaffung der Daten aus einer Datenbank mit SQL Abfragen und die abschließende Audgabe der Ergebnisse in Form von Tabellen oder einfachen Visualisierungen. Die Visualisierung der Daten sollte in einem virtuellen Notebook erfolgen. Als virtuelles Notebook durften wir uns entscheiden zwischen Apache Zeppelin oder Jupyter.

Unsere Aufgabe ist es ein geeignetes Szenario für die Bewältigung dieser Aufgabe zu finden und umzusetzen.

Wir entschieden uns für unser Szenario die Socrata Application Programming Interface (API) von NYC Open Data zu nutzen. NYC Open Data ermöglicht es allen "New Yorkernünd somit auch der ganzen Welt sogenannte Open Data also frei zugängliche Daten einfach zu konsumieren.[1].

NYC Open Data ermöglicht es uns sowohl einen kontinuierlichen Data Stream als auch eine statische CSV Datei zu konsumieren. Dank diesem Umstand entschieden wir uns im Rahmen dieses Projektes beide Möglichkeiten umzusetzen und zu vergleichen. Die Data Ingestion mit der CSV Datei setzte Julian Ruppel mit der Programmiersprache Java um und den kontunierlichen Datenstrom über die Socrata API wurde von Johannes Weber mit der Programmiersprache Python ausgelesen.

Auch bei der Data Retrieval entschieden wir uns dafür sowohl Apache Zeppelin als auch Jupyter zu nutzen und zu vergleichen. Die Beschaffung und Auswertung der Daten mit Apache Zeppelin übernahm Julian Ruppel. Johannes Weber bereitete die Daten mit der Programmiersparche Python auf und visualisierte sie in einem Jupyter Notebook.

Kapitel 2 - *Werkzeuge und technische Rahmenbedingungen* beschäftigt sich detailierter mit den verwendeten Tools, Frameworks und Programmiersprachen und Kapitel 3 - *Lösungsansatz* erläutert das gewählte Szenario mit den Unterkapitel Data Ingestion, Data Storage und Data Retrieval sowie der verwendeten Architektur. [2]

[1] **?**

[2] Einleitung von Johannes Weber

Kapitel 2

Werkzeuge und technische Rahmenbedingungen

Im folgenden Abschnitt werden einige wichtige technische Werkzeuge, die im späteren Verlauf eingesetzt werden, kurz und prägnant erläutert.

Apache Kafka ist eine verteilte Data-Streaming Plattform der Apache Software Foundation, die ursprünglich von LinkedIn entworfen wurde. Beliebt ist Kafka im BigData Umfeld wegen seiner Skalierbarkeit und Fehlertoleranz. Zu den Einsatzszenarien zählen vor allem Stream Processing, es kann aber auch als reiner Message Broker oder Speichersystem für Streaming Data verwendet werden. 2014 haben sich die verantwortlichen LinkedIn Mitarbeiter unter der Firma Confluent vom Mutterkonzern getrennt und sind seither an der Weiterentwicklung maßgeblich beteiligt. Geschrieben ist die quelloffene Software in der JVM-basierten Pogrammiersprache Scala, welche objektorientierte und funktionale Aspekte vereint.

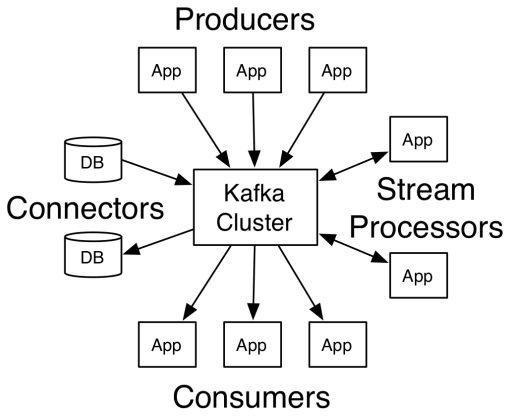

Abbildung 2.1: Apache Kafka Architektur[?]

Java ist eine objektorientierte Open-Source Programmiersprache die ursprünglich von Sun Microsystems entwickelt wurde und heute zum Oracle Konzern gehört. In den letzten beiden Hauptversionen wurde der Sprachumfang um funktionale und reaktive

Aspekte erweitert. Java ist dank der Java Virtual Maschine (JVM) als Laufzeitumgebung plattformunabhängig und hat sich vorwiegend in Enterprise Systemen und Web-Backends etabliert. Im Zuge der Verbreitung von BigData Projekten unter dem Dach der Apache Software Foundation, allem voran Hadoop und Spark, werden JVM sprachen wie Java und Scala nun auch im Bereich BigData eingesetzt.

Python ist eine Open-Source Skripsprache, die sich hauptsächlich durch eine gut lesbare und knappe Syntax auszeichnet und unter anderem das objektorientierte und funktionale Programmierparadigma unterstützt. Im Gegensatz zu Java ist Python dynamisch typisiert und wird interpretiert anstatt kompiliert. Dank eines sehr umfangreichen und ausgereiften Ökosystems aus Frameworks und Bibliotheken zur Datenanalyse und maschinelles Lernen, wie z.B. Googles TensorFlow, ist Python im Bereich BigData und dank der minimalinvasiven Eigenschaften zum Rapid Prototyping beliebt.

PostgreSQL ist ein objektrelationales Datenbankmanagementsystem. Das vollständig ACID-konforme und in C geschriebene quelloffene System zeichnet sich durch einen breiten Funktionsumfang, Stabilität, Standardkonformität, hohe Erweiterbarkeit, und als Resultat dessen, eine weite Verbreitung aus. Neben dem traditionellen zeilenorientierten Eigenschaften bietet PostgreSQL zudem Erweiterungen hinsichtlich verteilter, hoch-parallelisierter und spaltenorientierter Datenverarbeitung, ein Geoinformationssystem sowie Volltextsuche. Auch im Bereich NoSQL bietet PostgreSQL eine dokumentenorientierter Speicherung und durch Erweiterungen sogar Graphen und Schlüssel-Werte-Datenstrukturen. Diese Flexibilität eröffnet PostgreSQL vielseitige Einsatzszenarien, darunter sowohl OLTP als auch OLAP.

Apache Zeppelin ist eine Web-basierte Open-Source Software mit der man sog. Notebooks zur datengetriebenen, interaktiven und kollaborativen Analyse erstellen kann. Es werden eine Vielzahl an Speicher- und Analysetechnologien unterstützt, darunter SQL, Scala, Spark, Python und R. Im wesentlichen werden in einem Notebook Abfrageskripte ad-hoc gegen die diversen Datenquellen ausgeführt und deren Ergebnisse in einem konfigurierbaren Web-Dashboard visuell und interaktiv dargestellt. Dadurch eignet es sich sowohl zur explorativen Datenanalyse als auch zum veröffentlichen und teilen von Analyseergebnissen.

Jupyter ähnelt in den meisten Aspekten Apache Zeppelin, sodass sich alle oben zu Zeppelin genannten Punkte auch zu Jupyter nennen lassen. Die Unterschiede liegen eher im Detail der einzelnen Funktionen sowie der historisch und organisatorisch bedingten Nähe zu bestimmten Schlüsseltechnologien. Für das hier behandelte Forschungsprojekt spielen die individuellen Stärken und Schwächen der beiden Werkzeuge jedoch keine Rolle, weshalb beide Werkzeuge ebenbürtig eingesetzt werden.

SODA (Socrata Open Data API) ist eine quelloffene Open Data Web-Programierschnittstelle des U.S. Amerikansichen dienstleisters Socrata. Anhand URLs werden Datasets adressiert und mittels der an SQL angelehnten *Socrata Query Language* (SoQL) per HTTP GET in unterschiedlichen Datenformaten abgefragt. Zudem stehen SDKs für diverse Programmiersprachen zur Verfügung.

Kapitel 3

Lösungsansatz

In diesem Kapitel wird ein Konzept und Herangehensweise zur Lösung der in 1TODO genannten Problemstellung erörtert.

3.1 Architekturentscheidungen

Lambda vs. Kappa

Warum Lambda? stream ist unendlich. Ergebnisse/Aggregationen zwischenspeichern Aggregationen mit den Live Daten abmischen! Nachteile googlen Quelle verweisen

Warum Kappa? Daten sind endlich aber ausreichend für unsere Bedürfnisse. Haben eh nur Pseudostream. Vorteile googeln

https://www.confluent.io/blog/simplest-useful-kafka-connect-data-pipeline-world-thereabouts-part-1/ Grafik einbetten

3.2 Data Ingestion

3.2.1 Data Ingestion via CSV Datei

Da muss was stehen

3.2.2 Data Ingestion via OData Schnittstelle

Da muss was stehen

3.3 Data Storage

Data Storage

3.4 Data Retrieval

3.4.1 Data Retrieval mit Apache Zeppelin

Da muss was stehen

3.4.2 Data Retrieval mit Jupyter

Da muss was stehen

Kapitel 4

Fazit

Abbildungsverzeichnis

Abkürzungsverzeichnis

JSON JavaScript Object Notation

CSV Comma-separated values

API Application Programming Interface