Duale Hochschule Baden-Württemberg Mannheim

Big Data Architektur Dokumentation

Wirtschaftsinformatik Schwerpunkt Data Science

Verfasser:	Julia Pfützer, Mya-Melissa Jahic, Hannah Laier
Matrikelnummern:	7411297, 2774085, 1010595
Kurs:	WWI 20 DSB
Modul:	Big Data
Studiengangsleiter:	Prof. Dr. Bernhard Drabant
Dozent:	Prof. Dr.-Ing. habil. Dennis Pfisterer
Bearbeitungszeitraum:	ca.15.07.2022 – 02.09.2022

Inhalt

1. Einleitung 1

2. Die Architektur und Daten 2

2.1. Web Server 3

2.2. Big Data Processing und Messaging 3

2.3. Containerization 4

2.4. Database Server 4

2.5. Memcached 4

2.6. Kafka Cluster 5

2.7. Web Applikation 5

5. Zusammenfassung 6

6. Literaturverzeichnis i

1. Einleitung

Heutzutage werden jeden Tag eine große Menge an Daten generiert durch alltägliche Handlungen, wie die Benutzung des Smartphones oder das Einkaufen im Supermarkt. Bei sehr großen Datenmengen reichen die herkömmlichen Hard- und Softwaremöglichkeiten nicht mehr aus. Es wird eine besondere Architektur benötigt. Die Daten werden mit Rechner Clustern verarbeitet, die miteinander kommunizieren.

Die wichtigsten Aspekte bei der Verarbeitung von großen Datenmengen sind:

Das Volumen, die Vielfalt, die Geschwindigkeit, die Unsicherheit und laut datasolut auch der Mehrwert.

Das Volumen beschreibt die riesigen Datenmengen die gespeichert und verarbeitet werden.

Die Vielfalt beschreibt die unterschiedlichen Strukturen und Formate, die die Daten aufweisen können.

Die Geschwindigkeit meint die Schnelligkeit in der die Daten verarbeitet werden müssen, um bei der großen Menge an Daten auf dem aktuellen Stand zu bleiben.

Die Unsicherheit zielt auf die Datenqualität ab die die Verlässlichkeit der Daten angibt. Sind die Daten und die Datenquellen nicht verifiziert, können die Daten durchaus Fehlinformationen enthalten.

Der letzte wichtige Aspekt ist der Mehrwert den die Daten bringen. Aufgrund der Muster, die sich in den Daten erkennen lassen kann beispielsweise eine Werbestrategie oder Produktverbesserungen abgeleitet werden. Das generiert dem Unternehmen dann einen Mehrwert. (Wuttke, kein Datum)

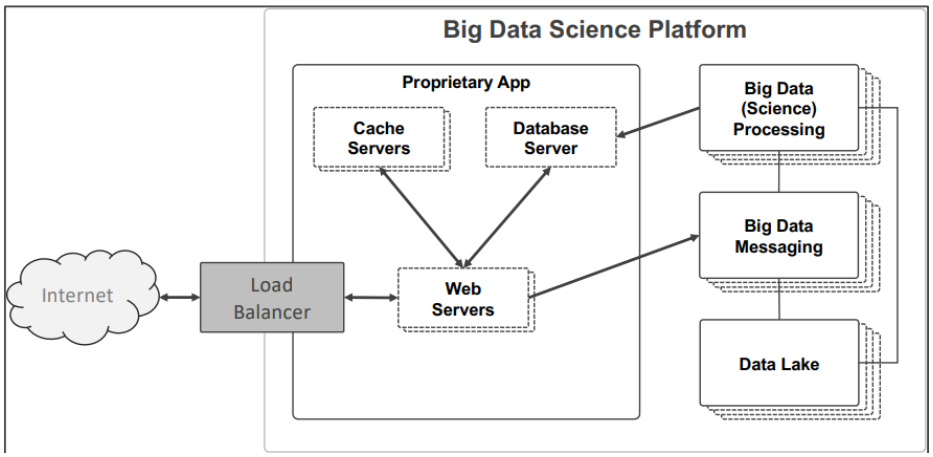
Deswegen ist es wichtig, große Datenmengen verarbeiten zu können. Diese Architektur wird angestrebt:

Das Ziel ist es die vorgegeben Architektur auf ein eigenes Beispiel umzubauen. Hier sind die Fußballspiele der Ligen angezeigt, mit den einzelnen Spielern und dem Austragungsort.

Zunächst wird die Zielarchitektur und die Daten beschreiben und folgend die einzelnen Bestandteile.

2. Die Architektur und Daten

Durch den Browser kann über einen Load-Balancer auf einen Web-Service zugegriffen werden. Der Web Service leitet die Interaktion an den Big Data Messaging Service weiter. Dieser arbeitet zusammen mit Data Lake und Big Data Processing. Hier kann das Produkt verbessert werden. Danach werden die Informationen in das Serving layer gespeist. Die Daten werden in einer Datenbank gespeichert. Der Web Server greift auf den Cache Server zurück oder auf die Datenbank.



Quelle: Aufgabenstellung von Prof. Dr.-Ing. habil. Dennis Pfisterer

Die verwendeten Technologien werden im Folgenden beschrieben. Durch den Browser, über den Load Balancer kubernetes wird auf die Node App zugegriffen. Die Node App liest die Daten aus Memcached und MySQL aus. Die Daten gehen vorher den Weg über Kafka und mit structured streaming zu Spark. Spark macht die Datenvorverarbeitung und speichert seine Checkpoints in HDFS ab. Von dort aus werden die Daten in MySQL gespeichert und können vom Web Server Note App ausgelesen und abgerufen werden.

Die verwendeten Daten sind hier zu finden auf Keggel: https://www.kaggle.com/code/alaasedeeq/european-soccer-database-with-sqlite3/data. Die Daten bestehen aus elf verschiedenen Ländern, elf verschiedenen Leagen, 25979 Spielen, 11060 Spielen und 299 Teams. (Sedeeq, 2020)

2.1. Web Server

Als Web Server wird die Node.js Applikation verwendet. Der Server befindet sich in einem Cluster von node slim. Um sich mit dem Cluster zu verbinden und Daten dort hin schicken zu können wird Kafka JS verwendet. Dafür wird ein Broker übergeben und eine eindeutige Client ID. Beim Aufruf eines Clients wird eine Tracking Massage erzeugt. Diese Nachricht enthält einen Zeitstempel und den Namen des Matchs.

2.2. Big Data Processing und Messaging

Hier wird Spark verwendet. Spark holt sich die Daten aus der Node JS Applikation. Diese Verbindung muss zunächst hergestellt werden. Diese besteht aus der Definition der Tracking Message und der Definition der Kafka Message. Die hier entstehende Nachricht muss im nächsten Schritt auseinander genommen werden von Spark. Dafür wird die unbekannte Nachricht aus dem binären Format in JSON umgewandelt. Aus dem JSON Format wird dann der Zeitstempel der Tracking Message codiert und für die Tracking Message wird die in JSON formatierte Nachricht verwendet.

Als nächstes werden die Daten gruppiert. Die Kriterien dafür ist das Fußballmatch. Um die Änderung anzuzeigen wird nun ein Update in die Konsole geschickt, welches ausschließlich die Änderungen anzeigt.

Die Änderungen die bei der Berechnung eines neuen Batchs entstehen, sollen nicht nur in der Konsole ausgegeben werden, sondern auch dauerhaft festgehalten werden. Dazu wird eine Verbindung zu MySQL hergestellt. MySQL wird hier als Datenbankserver verwendet. Die Funktion wird einmal für jede Partition aufgerufen. Das entspricht einem Spark Host. Für jeden Batch Durchlauf soll es einmal ausgeführt werden. Das wird neben der Partitionsfunktion extra im Code konfiguriert. Das Fußballspiel wird in die Tabelle mit aufgenommen mit dem aktuellen Count. Falls dieses Fußballspiel schon existiert wird der Count auf den neuen Stand gebracht. Der Speichervorgang geschieht mit Hilfe eines Iterators durch die zu speichernden Elemente.

2.3. Containerization

Die Containerization wird mit Docker umgesetzt. Zuerst wird hier die Definition des Containers festgelegt. Das Image bitnami/Spark wird verwendet, da mit Spark gearbeitet wird. Zusätzlich muss das Kafka Plug-In in die Definition mit aufgenommen werden. Es ist nicht automatisch im Spark Image integriert. Das Kafka Plug-In wird extra installiert mithilfe von Ivy. Um Fehler zu umgehen wird Spark als User zur Gruppe 1001 hinzugefügt um die Rechte des Zugriffs zu erhalten.

Im nächsten Schritt werden die Dependencies für Kafka und Spark installiert in einer Python Datei. Ein Textdokument mit den Anforderungen an Spark wird von pip installiert. Diese Dependencies werden gesammelt und verbunden, um es flexibel verwenden zu können, auch auf anderen Rechnern.

Danach wird definiert in welcher Form die Applikation laufen soll. Beispielsweise local in dem aufgebauten Container mit Spark Image.

2.4. Database Server

Als Datenbank wird MySQL verwendet. Hier ist der erste Schritt die Konfiguration der Datenbank. Die Datenbank wird in einer ConfigMap konfiguriert. Es werden zwei Tabellen erstellt. Die erste Tabelle beinhaltet die Matches mit den Namen, der Beschreibung und der Überschrift. In der zweiten Tabelle werden die Ränge der Matches gespeichert. Um die Top 10 Matches anzeigen zu können müssen zuerst mindesten 10 verschieden Matches aufgerufen werden. Die Tabelle mit den Matches enthält die Matches, eine Beschreibung dazu, eine Überschrift. Der Primärschlüssel sind die Matches. In der anderen Tabelle sind die Matches gespeichert und der Count dazu, der den Rang festlegt und die Top 10 Bewertung ermöglicht.

Ein weiterer Bestandteil der ConfigMap sind die Matches. Diese werden auch in die ConfigMap gecoded.

Die Konfiguration wird im nächsten Schritt in den MySQL Container eingebaut, es wird an den Docker entry point gemountet mithilfe einer weiteren ConfigMap.

2.5. Memcached

Ist ein mit dem Web Server verbundenes Speicherobjekt Caching System. Das System unterstützt dynamische Webanwendungen in dem die Last auf die Datenbank reduziert wird. (Anon., kein Datum)

Hierfür wird eine Konfigurationsdatei aufgesetzt. In dieser werden zwei replicas spezifiziert und memcached wird in einen Container mit dem Image Memcached:alpine platziert.

2.6. Kafka Cluster

Kubernetes dient der Verwaltung von Containern. Kubernetes kann durch Kafka Cluster erweitert werden. Kafka dient der Kommunikation innerhalb des Clusters. (Anon., kein Datum) (Anon., kein Datum)

2.7. Web Applikation

Des weiteren muss noch die Web Applikatio zusammengebaut werden. Das Konfigurationsfile startet mit dem Deployment. Für den Container wird das Image Farberg/popular-slides verwendet und der App Name popular – slides. Mit Skaffold wird die Spark App und die Web App zusammengebaut und lauf fähig gemacht. Damit zusammen hängen die Dateien popular-slides-app unnd popular-slides-spark. In poplular-slides-app ist das Deployment festgelegt, der Service und Ingres. In popular-slides-spark ist die ConfigMap festgelegt und ein Deployment.

5. Zusammenfassung

Dadurch, dass heutzutage große Mengen an Daten verarbeitet werden, müssen spezielle Architekturen her in denen die Daten miteinander kommunizieren können. Unter den wichtigsten Aspekten der Datenverarbeitung gehört das Volumen, die Vielfalt, die Geschwindigkeit und der Mehrwert.

In der Architektur ermöglicht der Load Balancer den Zugriff auf den Web-Service, welcher Interaktionen an den Big Data Messaging Server weiterleitet. Auch der Data Lake und das Big Data Processing spielen hier eine große Rolle, unter anderem als Ablageort der Daten. Außerdem kann über den Web Server auf die Cache Server und den Datenbank Server zugegriffen werden, in dem ebenso Daten gesichert werden können. Die in der Dokumentation verwendeten Daten befassen sich dabei mit Daten aus dem Bereich Fußball.

Die eigentliche Datenverarbeitung findet im Big Data Processing und Messaging statt, indem Nachrichten in JSON Formate verarbeitet werden. Die Änderungen in den Daten werden in der MySQL Datenbank gespeichert, in der auch die ganzen Informationen zu den einzelnen Fußball Bereichen stehen. Mit Docker kann die Containerization umgesetzt werden. Dazu gehören Kafka, das zur Kommunikation innerhalb des Clusters dient, sowie Spark.

Die MySQL Datenbank wird auch in einer ConfigMap konfiguriert. Hier sind alle relevanten Spalten der Fußballspiele enthalten. Top Matches können durch das Aufrufen von mindestens zehn Matches angezeigt werden.

Memcached reduziert dabei die Last auf der Datenbank.

Literaturverzeichnis

Anon., kein Datum [Online]
Available at: https://memcached.org/

Anon., kein Datum [Online]
Available at: https://cloud.google.com/kubernetes-applications?hl=de

Anon., kein Datum [Online]
Available at: https://kafka.apache.org/

Sedeeq, A., 2020. [Online]
Available at: https://www.kaggle.com/code/alaasedeeq/european-soccer-database-with-sqlite3/data

Wuttke, L., kein Datum [Online]
Available at: https://datasolut.com/was-ist-big-data/