INTERNATIONALE HOCHSCHULE

Portfolio im Modul

Projekt: Data Engineering

DLMDWWDE02

im Studiengang Master of Science Informatik

über das Thema

Entwicklung einer batch-basierte Datenarchitektur für eine datenintensive Applikation

Abstract

von

Stefan Stimmer

Betreuer:	Prof. Dr. Max Pumperla
Matrikelnummer:	IU14077512
Abgabedatum:	16.02.2026

Abstract

Ziel dieses Projekts war der Entwurf und die prototypische Umsetzung einer batch-basierten Datenarchitektur zur Verarbeitung umfangreicher Wetterdaten. Grundlage bildete ein synthetischer Wetterdatensatz von Kaggle mit Messwerten für mehrere Standorte (Prasad22, 2020).

Die Pipeline sollte große Datenmengen zuverlässig aufnehmen, persistent speichern, vorverarbeiten und aggregieren, sodass diese für analytische Zwecke sowie ein potenzielles Machine-Learning-Training nutzbar sind. Ein besonderer Fokus lag auf klarer Entkopplung der Komponenten, Reproduzierbarkeit der Datenverarbeitung und einer nachvollziehbaren Datenhaltung, wie sie für dateninstensive Systeme empfohlen wird (Kleppmann, 2017).

Methodik und Architektur

Die Lösung folgt einem modularen Microservice-Ansatz. Ein Ingestion-Service (Kafka Producer) liest die Wetterdaten ein und publiziert sie als Events in Apache Kafka. Kafka dient als Messaging-Schicht zur Entkopplung zwischen Datenquelle und nachgelagerten Komponenten. Außerdem wird AKHQ als leichtgewichtige Web-Oberfläche zur Inspektion und Verwaltung von Kafka verwendet. Ein Persistenz-Service (Kafka Consumer) konsumiert die Events und speichert sie unverändert und append-only in einer Raw Storage Zone. Dadurch bleibt eine auditierbare Rohdatenbasis erhalten, die jederzeit erneut verarbeitet werden kann.

Der Batch-Processing-Job wird periodisch ausgeführt und übernimmt die eigentliche Datenaufbereitung. Er liest die Rohdaten und berechnet Tagesdurchschnitte pro Ort als zentrale Aggregation. Die berechneten Kennzahlen werden strukturiert in einer PostgreSQL-Datenbank als Processed Zone gespeichert. Zur besseren Anschaulichkeit wurde zusätzlich ein Streamlit-Dashboard implementiert, das ausgewählte Aggregationen visualisiert und als Vorschau für ein mögliches nachgelagertes Machine-Learning-Modell dient. Die gesamte Umgebung wird containerisiert und mittels Docker Compose reproduzierbar orchestriert.

Ergebnisse und Bewertung

Die entwickelte Pipeline konnte erfolgreich implementiert und lokal ausgeführt werden. Daten werden zuverlässig von der Quelle über Kafka in die Raw Zone ingestiert, periodisch verarbeitet und anschließend in strukturierter Form in der Datenbank bereitgestellt. Das Dashboard ermöglicht eine einfache visuelle Überprüfung der aggregierten Wetterdaten.

Besondere Herausforderungen ergaben sich aus den Abhängigkeiten zwischen den Microservices sowie dem Umgang mit dem Datenzustand während der Entwicklung. Insbesondere musste berücksichtigt werden, ob Daten bereits ingestiert oder verarbeitet wurden und wie Wiederholungen der Pipeline idempotent gestaltet werden können. Diese Aspekte erforderten zusätzliche Aufmerksamkeit bei der Orchestrierung und beim Design der Batch-Logik.

Reflexion

Durch bereits vorhandene praktische Erfahrungen mit Docker und Apache Kafka konnte die grundlegende Infrastruktur vergleichsweise schnell aufgebaut werden. Gleichzeitig zeigte das Projekt, dass insbesondere die Orchestrierung mehrerer Microservices und das konsistente State-Management zusätzliche konzeptionelle Sorgfalt erfordern, auch wenn die Einzeltechnologien vertraut sind.

In einem Folgeprojekt würde mehr Zeit in die konzeptionelle Vorplanung des Datenflusses investiert werden. Insbesondere sollte früher klar definiert werden, wann Daten in welcher Zone überschrieben, neu berechnet oder unverändert beibehalten werden.

Positiv ist die klare Trennung der Verantwortlichkeiten durch den Microservice-Ansatz sowie die gute Nachvollziehbarkeit durch die append-only Raw Zone. Die Kombination aus Kafka, Batch Processing und einer relationalen Processed Zone hat sich als geeignet für periodische Analysen erwiesen.

Strategie für eine zweite Pipeline zur Stream-Prozessierung

Neben der bestehenden batch-basierten Verarbeitung kann die Architektur relativ einfach um eine Stream-Processing-Pipeline erweitert werden. Die bereits eingesetzte Kafka-Schicht bietet hierfür eine geeignete Grundlage, da sie als zentrale Event-Streaming-Plattform sowohl Batch- als auch Echtzeitverarbeitung parallel unterstützen kann (Kleppmann, 2017). Der bestehende Ingestion-Service würde unverändert bleiben und weiterhin Wetterereignisse in ein Topic produzieren.

Für die Stream-Verarbeitung würde ein zusätzlicher Stream-Processing-Service als weiterer Kafka-Consumer eingeführt werden. Technologisch könnten hierfür beispielsweise Kafka Streams, Apache Flink oder Spark Structured Streaming verwendet werden. Dieser Service würde die eingehenden Events kontinuierlich verarbeiten und zeitbasierte Window-Aggregationen berechnen, etwa gleitende Durchschnittstemperaturen oder Niederschlagsmengen in 5- oder 60-Minuten-Fenstern pro Ort. Die Ergebnisse könnten in eine separate Realtime-Tabelle innerhalb der bestehenden PostgreSQL-Datenbank oder in eine dedizierte Realtime-Zone geschrieben werden.

Wesentlich für eine robuste Stream-Pipeline ist die korrekte Behandlung von Zustellsemantiken und Datenkonsistenz. In einer pragmatischen Umsetzung wäre mindestens eine at-least-once-Verarbeitung mit idempotenten Datenbank-Upserts sinnvoll. Für höhere Anforderungen könnte perspektivisch eine exactly-once-Semantik durch Transaktionen oder Checkpointing des Stream-Frameworks implementiert werden. Zusätzlich sollten Batch- und Stream-Pipeline logisch getrennt bleiben, beispielsweise durch unterschiedliche Consumer Groups, Topics und Zieltabelle.

Durch diese Erweiterung würde die bestehende Architektur sowohl robuste periodische Analysen als auch Near-Real-Time-Auswertungen unterstützen. Gleichzeitig bliebe die Raw Zone als verlässliche Reprocessing-Basis erhalten, wodurch die Wartbarkeit und Nachvollziehbarkeit des Gesamtsystems gewährleistet bleibt.

Literaturverzeichnis

Kleppmann, M. (2017): Designing data-intensive applications. The big ideas behind reliable, scalable, and maintainable systems. 1st Edition. Sebastopol, CA: O'Reilly

Prasad22. (2020). Weather data [Data set]. Kaggle. https://www.kaggle.com/datasets/prasad22/weather-data