Praktikumsbericht II

an der Hochschule des Bundes für öffentliche Verwaltung
im Studiengang Digitale Administration und Cyber-Security
(DACS)

Streaming AIS – Erarbeitung moderner Techniken für die Datenverarbeitung in Big Data-Anwendungen

Name, Vorname	Busenius, Patrick
Einstellungsjahrgang und Kursnummer:	2020/10
Zeitraum des Praktikums:	01.10.2022 – 31.03.2023
Praktikumsbehörde und Organisationseinheit:	/
Ausbildungsleiter/in:	/
Datum der Abgabe:	17.02.2023

Digital Administration and Cyber Security

Abbildungsverzeichnis

1 Einleitung

Im Rahmen meines Vorbereitungsdienstes für den gehobenen nichttechnischen Verwaltungsdienst des Bundes mit Schwerpunkt digitale Verwaltung und Cyber-Sicherheit absolvierte ich vom 01.10.2022 bis zum 28.02.2023 ein Praktikum in einem Sachgebiet, das im Zuge einer Umstrukturierung aufgestellt wurde und sich auf die Entwicklung von Anwendungen im DATAINT-Bereich spezialisiert hat. Dieses Sachgebiet konzentriert sich auf die Konsolidierung von Datenquellen aus verschiedenen Bereichen der Behörde, die Erstellung von APIs und Bibliotheken für diese Datenquellen sowie die Bereitstellung eines Clusters für Big-Data- und Data-Science-Anwendungen.

Wie bereits während meines ersten Praktikums sollten die Studierenden ihr erworbenes Wissen aus den vorangegangenen Semestern durch die Umsetzung eines Projekts unter Beweis stellen und durch praktische Arbeit vertiefen.

2 Vorstellung Projekt

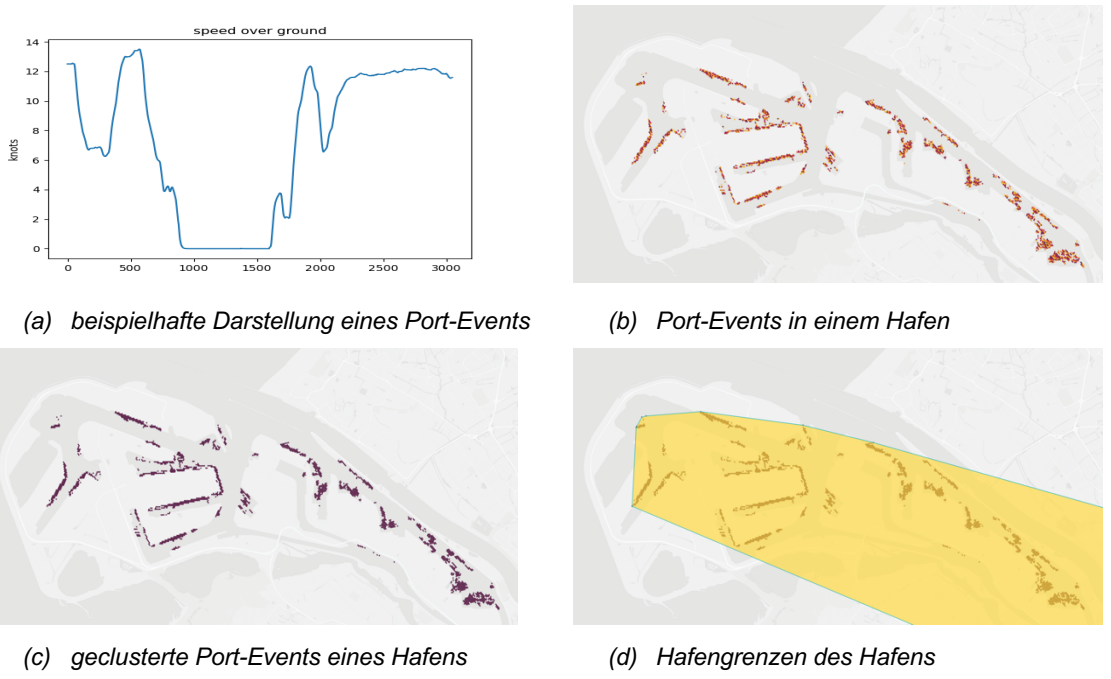
Nach Absprache mit der Leitung des Sachgebiets wurde festgestellt, dass es einige inhaltliche Überschneidungen mit meinem Praktikum im dritten Semester gibt, auf die in diesem Praktikum aufgebaut werden sollte. Insbesondere sollten die gewonnenen Erkenntnisse in eine Form gebracht werden, die für das Sachgebiet relevant ist.
In meinem ersten Praktikum, das ich in einem Bereich absolvierte, der für die Erforschung und Entwicklung von Analyseverfahren für produktionsunterstützende Fachdienste zuständig ist, beschäftigte ich mich mit der Analyse von Daten des Automatic Identification System (AIS) (Automatic Identification System (AIS) Overview | Navigation Center o. J.) .

2.1 Automatic Identification System

AIS ist ein automatisches Ortungs- und Identifikationssystem, das in der Schifffahrt eingesetzt wird, um Schiffe und andere Wasserfahrzeuge in Echtzeit zu erfassen und zu identifizieren. Es verwendet Funkwellen, um Daten wie Position, Geschwindigkeit, Kurs und Identifikationsnummer von Schiffen zu übertragen und zu empfangen.
Der Einsatz von AIS in der zivilen Schifffahrt wird durch internationale Vorgaben der Internationalen Seeschifffahrtsorganisation (IMO) und der Internationalen Fernmeldeunion (ITU) geregelt. Demnach müssen bestimmte Schiffe wie Handelsschiffe ab einer bestimmten Größe, Tanker und Passagierschiffe AIS-Geräte an Bord haben und betriebsbereit halten. Dadurch wird sichergestellt, dass Schiffe sicher und effizient überwacht und gesteuert werden können und die Wahrscheinlichkeit von Unfällen oder Zusammenstößen verringert wird.

2.2 Zusammenfassung der Inhalte des ersten Praktikums

Im ersten Praktikum beschäftigte ich mich mit der Verarbeitung von AIS-Daten und den möglichen Einsatzfeldern dieser Daten. Konkret sollte eine Hafengrundlagendatenbank aus Bewegungsdaten der internationalen Schifffahrt erstellt werden. Hierzu wurden zunächst AIS-Daten analysiert, um mögliche Muster, die in Häfen oder hafenähnlichen Gebieten auftreten, erkennen und eindeutig zuordnen zu können. Häfen und hafenähnliche Gebiete weisen eine sehr geringe Geschwindigkeit oder eine Geschwindigkeit von 0 Knoten über längere Zeit auf (siehe Abbildung 1a). Wenn man die erkannten Port-Events auf einer Karte betrachtet, lassen sich Hafenstrukturen deutlich erkennen (siehe Abbildung 1b). Durch die Anwendung eines Cluster-Algorithmus wie z.B. DBSCAN (Khan u. a. 2014) können die zuvor gewonnenen Port-Events zu einem Hafen-Cluster zusammengefasst werden (siehe Abbildung 1c). Auf der Grundlage dieser regionalen Muster konnte im Anschluss die Umrandung des Hafengebiets ermittelt werden (siehe Abbildung 1d).



(a) beispielhafte Darstellung eines Port-Events

(b) Port-Events in einem Hafen

(c) geclusterte Port-Events eines Hafens

(d) Hafengrenzen des Hafens

Abbildung 1: Schritte des Data Driven Port Index (DDPI)

2.3 Implementierung einer Verarbeitungskette der AIS-Daten

Die damals gewonnenen Erkenntnisse sollten im Anschluss von einem theoretischen Ansatz oder einer Proof-of-Concept-Entwicklung zu einer für die Produktion geeigneten Anwendung weiterentwickelt werden. Das Ziel des Praktikums bestand darin, einen Prozess (siehe Abbildung 2) zu entwickeln, der neben der Bereitstellung der AIS-Rohdaten auch weitere Analyseergebnisse liefert. Für die Umsetzung sollte auf Techniken und Frameworks aus dem Big-Data-Tech-Stack zurückgegriffen werden.

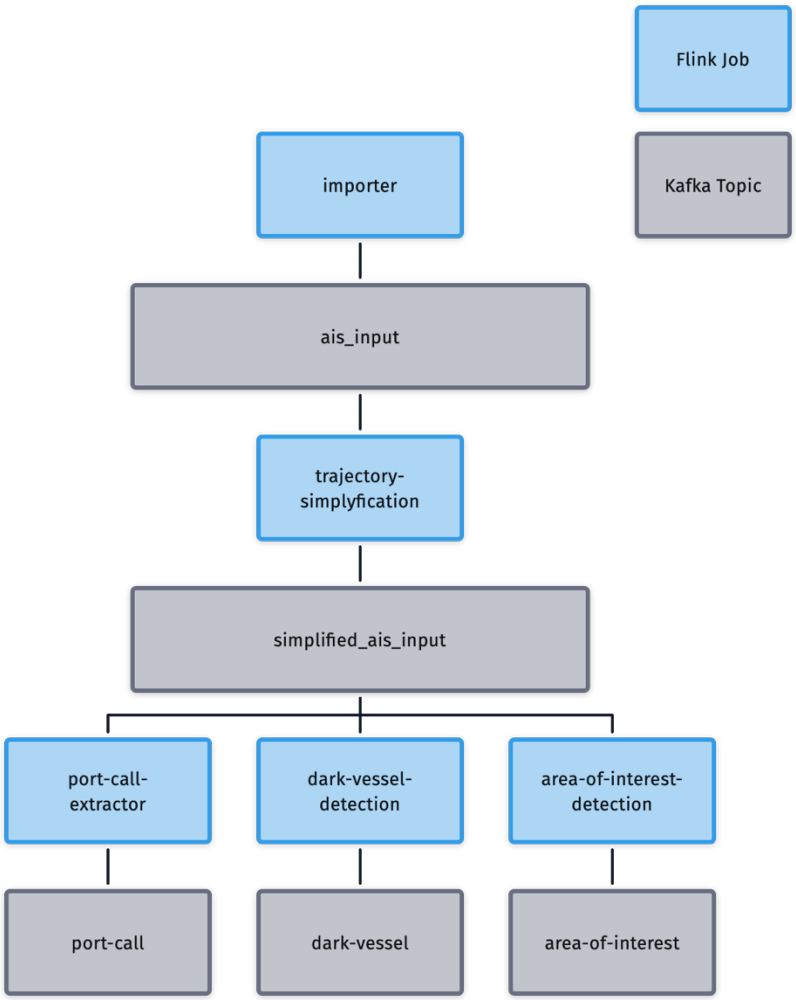


Abbildung 2: Übersichtsdarstellung der einzelnen Verarbeitungsschritte

3 Unterschiedliche Verarbeitungsarten

Die Verarbeitung großer Datenmengen lässt sich grundsätzlich in zwei Gruppen unterteilen: Batch-Processing und Stream-Processing. Im Folgenden werden die beiden Arten der Datenverarbeitung im Detail betrachtet, wobei auf ihre jeweiligen Besonderheiten eingegangen wird.

3.1 Batch-Processing

Batch-Processing bezieht sich auf die Verarbeitung großer Datenmengen in einem einzigen Arbeitsdurchlauf, auch Batch-Job genannt. Dabei werden Aufgaben in einer sequenziellen Reihenfolge ausgeführt, ohne dass Benutzereingaben erforderlich sind. Für die Batch-Verarbeitung werden die Daten im ersten Schritt in kleinere Pakete aufgeteilt und im Anschluss durch weitere Schritte prozessiert. Batch-Processing wird häufig verwendet, um Aufgaben wie das Ausführen von Berichten, das Erstellen von Rechnungen, die Verarbeitung von Transaktionen oder die Durchführung von Datensicherungen zu automatisieren.

Der Batch-Job wird typischerweise von einer Job-Scheduler-Software gesteuert, die den Job auf einem einzelnen Computer oder auf einem Cluster aus mehreren Servern ausführt. Dadurch kann die Verarbeitung großer Datenmengen effizient erfolgen, da mehrere Aufgaben gleichzeitig ausgeführt werden können. Batch-Processing bietet viele Vorteile, darunter die Möglichkeit, Aufgaben automatisch und ohne Benutzereingriff auszuführen, die Freisetzung von Ressourcen durch die parallele Verarbeitung mehrerer Aufgaben sowie die Möglichkeit, komplexe Aufgaben zu automatisieren, die möglicherweise mehrere Schritte erfordern.
Allerdings hat Batch-Processing auch einige Nachteile. Da die Verarbeitung zeitversetzt erfolgt, kann es schwierig sein, Fehler zu identifizieren und zu beheben. Auch kann es schwierig sein, auf Echtzeit-Informationen zuzugreifen, da die Verarbeitung im Hintergrund stattfindet. Trotzdem wird Batch-Processing in vielen Unternehmen und Organisationen aufgrund seiner Effizienz und Automatisierungsmöglichkeiten eingesetzt.

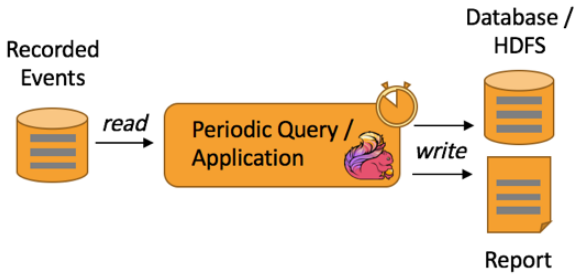


Abbildung 3: Übersicht Batch-Processing (Apache Flink: Use Cases o. J.)

3.2 Stream-Processing

Stream Processing ist eine Technologie, die es ermöglicht, Datenströme in Echtzeit zu sammeln, zu verarbeiten und zu analysieren. Der grundlegende Aufbau eines Stream Processing-Systems besteht aus einer Reihe von Komponenten, die zusammenarbeiten, um die Datenströme zu verarbeiten und zu analysieren:

- Datenquelle: Die Datenquelle kann eine externe Quelle sein, die Datenströme in Echtzeit generiert, wie z.B. soziale Medien, IoT-Geräte oder Finanzdaten.
- Eingabe: Die Eingabe ist die erste Komponente, die Datenströme aus der Datenquelle empfängt und sie für weitere Verarbeitung bereitstellt.
- Transformationen: Die Transformationen sind eine Reihe von Operationen, die auf den Datenströmen durchgeführt werden, um sie für weitere Analysen vorzubereiten. Dies kann das Filtern unerwünschter Daten, das Konvertieren von Daten in ein anderes Format oder das Zusammenführen von Daten aus verschiedenen Quellen beinhalten.

- Aggregationen: Die Aggregationen sind eine Reihe von Operationen, die verwendet werden, um Datenströme zu gruppieren und zusammenzufassen, um Trends und Muster zu erkennen.
- Ausgabe: Die Ausgabe ist die Komponente, die die endgültigen Datenströme bereitstellt, die bereits durch die Verarbeitung und Analyse optimiert wurden.

Insgesamt ermöglicht Stream Processing eine schnelle und effiziente Verarbeitung von Datenströmen in Echtzeit. Dadurch können Unternehmen und Organisationen schnelle Entscheidungen treffen und Prozesse anpassen, während die Daten generiert werden.

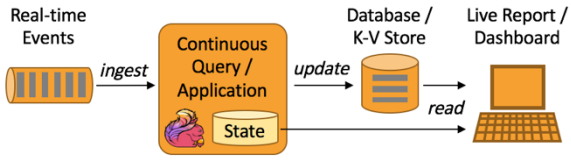


Abbildung 4: Übersicht Stream-Processing (Apache Flink: Use Cases o. J.)

4 Frameworks

Im ersten Schritt musste ich mich mit einer Vielzahl von für mich neuen Techniken zur Verarbeitung von Big Data auseinandersetzen und ihre Eignung sowohl für den konkreten Anwendungsfall als auch für den Einsatz in der vorhandenen Architektur untersuchen.

4.1 RabbitMQ

RabbitMQ ist eine Open-Source-Message-Broker-Software, die für die Implementierung von Message-Queues verwendet wird. Es handelt sich um eine plattformunabhängige Software, die auf verschiedenen Betriebssystemen ausgeführt werden kann (Messaging that just works — RabbitMQ o. J.).

RabbitMQ unterstützt verschiedene Messaging-Protokolle wie AMQP (Advanced Message Queuing Protocol), STOMP (Streaming Text Oriented Messaging Protocol) und MQTT (Message Queuing Telemetry Transport). Dadurch kann RabbitMQ in einer Vielzahl von Anwendungsfällen eingesetzt werden (Which protocols does RabbitMQ support? — RabbitMQ o. J.).

Die Kernfunktionalität von RabbitMQ besteht darin, Nachrichten zwischen verschiedenen Anwendungen zu vermitteln. Es ermöglicht die Entkopplung von Anwendungen, die asynchrone Kommunikation, die Verteilung von Aufgaben und die Skalierung von Anwendungen. RabbitMQ arbeitet auf der Grundlage von *Produzenten* und *Konsumenten*. Ein Produzent sendet eine Nachricht an eine Warteschlange, und ein Konsument empfängt diese Nachricht aus der Warteschlange.

RabbitMQ bietet auch fortgeschrittene Funktionen wie die Möglichkeit, Nachrichtenprioritäten festzulegen, die Verarbeitung von Nachrichten mit hoher

Geschwindigkeit, die Zuordnung von Nachrichten zu bestimmten Konsumenten und die Möglichkeit, Nachrichten auf der Grundlage von Regeln weiterzuleiten.

RabbitMQ bietet eine Vielzahl von Client-Bibliotheken für verschiedene Programmiersprachen wie Java, Python, Ruby und .NET. Diese Bibliotheken erleichtern die Integration von RabbitMQ in bestehende Anwendungen.

RabbitMQ ist eine leistungsstarke und zuverlässige Lösung für die Implementierung von Message-Queues. Es wird von vielen Unternehmen und Organisationen weltweit eingesetzt, um komplexe Anwendungen zu unterstützen und Geschäftsprozesse zu optimieren.

4.2 Apache Kafka

Apache Kafka ist eine verteilte Streaming-Plattform, die für die Verarbeitung von Datenströmen in Echtzeit entwickelt wurden (Narkhede, Shapira, und Palino 2017). Es handelt sich um eine Open-Source-Software, die von der Apache Software Foundation entwickelt und unterstützt wird. Kafka verwendet ein verteiltes Architekturmuster, um Skalierbarkeit und Zuverlässigkeit zu gewährleisten. Es besteht aus mehreren *Brokern*, die in einem Cluster zusammengeschlossen sind (Apache Kafka o. J.). Jeder Broker ist für die Speicherung und Verteilung von Daten zuständig, die in Form von *Topics* organisiert sind. Ein Topic ist ein Kanal für Datenströme, der von Produzenten, Anwendungen, die Daten generieren und Konsumenten, Anwendungen, die Daten verarbeiten genutzt werden kann.

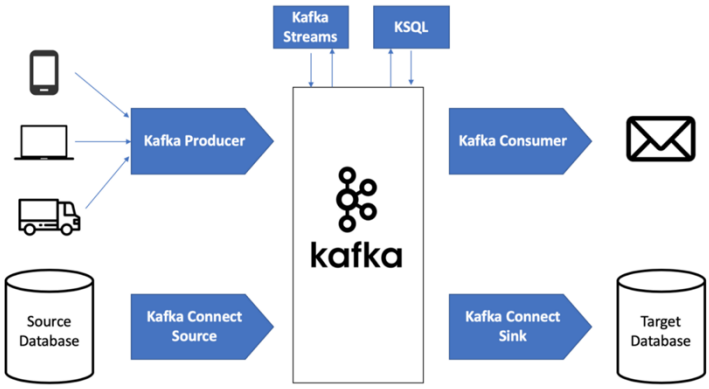


Abbildung 5: Übersichtsdiagramm Apache Kafka (Maarek 2022)

Kafka ist in der Lage, eine hohe Anzahl von Datenströmen gleichzeitig zu verarbeiten, und ist daher ideal für Big-Data-Anwendungen, Streaming-Analytik und Echtzeit-Verarbeitung. Es ermöglicht eine zuverlässige Übertragung von Datenströmen in Echtzeit, unabhängig von der Größe oder Komplexität der Daten.

Kafka ist auch hochgradig skalierbar und kann auf Tausende von Brokern skaliert werden. Es bietet eine Reihe von Tools und APIs, die die Integration in bestehende Anwendungen erleichtern. Apache Kafka ist eine der am weitesten verbreiteten Streaming-Plattformen und wird von vielen großen Unternehmen und Organisationen weltweit eingesetzt. Es bietet

eine leistungsstarke und flexible Möglichkeit, Datenströme in Echtzeit zu verarbeiten und ist daher ein wichtiger Bestandteil moderner Datenverarbeitungssysteme.

4.3 Apache Spark

Apache Spark ist ein Open-Source-Framework für Big-Data-Analysen und -Verarbeitung, das entwickelt wurde, um schnelle und effiziente Datenverarbeitung in großen Datenmengen zu ermöglichen. Es wurde von der Apache Software Foundation entwickelt und unterstützt und ist in Java, Scala, Python und R verfügbar (Tandon u. a. 2022).

Spark ermöglicht es, große Datenmengen auf verteilten Systemen zu verarbeiten, indem es den Arbeitsspeicher des Clusters effizient und intelligent nutzt. Es bietet eine Vielzahl von Bibliotheken und Tools für Datenverarbeitung, -analyse und -visualisierung, einschließlich maschinellen Lernens, Stream-Verarbeitung und Graph-Verarbeitung.

Spark bietet auch eine leistungsstarke API, die es Entwicklern ermöglicht, Anwendungen in verschiedenen Sprachen zu schreiben, die auf verschiedenen Plattformen ausgeführt werden können. Es unterstützt auch verschiedene Datenquellen und Datenformate, darunter HDFS, HBase, JSON, CSV und viele andere.

Spark nutzt eine Technologie namens "Resilient Distributed Datasets" (RDDs) (RDD Programming Guide - Spark 3.3.1 Documentation o. J.), die es ermöglicht, Daten auf effiziente Weise auf einem Cluster zu speichern und zu verarbeiten. RDDs ermöglichen eine schnelle Verarbeitung von Daten durch parallele Ausführung von Operationen auf einem Cluster von Computern.

Spark kann auch nahtlos mit anderen Big-Data-Tools und -Frameworks wie Hadoop, Cassandra und Amazon S3 integriert werden. Es bietet eine Vielzahl von Möglichkeiten zur Skalierung von Daten und zur Steigerung der Verarbeitungsgeschwindigkeit, was es zu einem wichtigen Werkzeug für die Analyse und Verarbeitung von Big Data macht.

Apache Spark ist ein leistungsstarkes und flexibles Framework für Big-Data-Analysen und -Verarbeitung. Es wird von vielen großen Unternehmen und Organisationen weltweit eingesetzt, um Datenverarbeitungsprozesse zu optimieren, Geschäftsergebnisse zu verbessern und wertvolle Einblicke in Daten zu gewinnen.

4.4 Apache Flink

Apache Flink ist ein Open-Source-Verteilungs-Framework für Stream- und Batch-Verarbeitung großer Datenmengen. Es wurde von der Apache Software Foundation entwickelt und unterstützt und ist in Java und Scala verfügbar. Zusätzlich wird an einer einer weiteren API in Python mit dem Namen PyFlink gearbeitet. Flink wurde entwickelt, um die Verarbeitung von Big Data auf eine schnelle, effiziente und fehlertolerante Weise zu ermöglichen.

Flink ermöglicht die Verarbeitung von Datenströmen in Echtzeit, indem es die Daten direkt von der Quelle liest, verarbeitet und analysiert. Es bietet auch eine Batch-Verarbeitung, mit der Daten in festen Intervallen verarbeitet werden können. Flink ist in der Lage, große

Datenmengen in Echtzeit zu verarbeiten, wodurch Unternehmen schnell auf Änderungen in ihren Daten reagieren und sofort auf Echtzeitereignisse reagieren können.
Flink ist in der Lage, große Datenmengen parallel auf einem verteilten Cluster von Computern zu verarbeiten, um die Verarbeitungsleistung zu maximieren. Es bietet auch eine Vielzahl von Bibliotheken und Tools für maschinelles Lernen, Graph-Analyse und Stream-Verarbeitung. Flink ist auch in der Lage, Daten aus verschiedenen Quellen und Formaten zu lesen, einschließlich Apache Kafka, Hadoop HDFS und Amazon S3.
Ein weiteres wichtiges Merkmal von Flink ist seine Fehlertoleranz. Es ist in der Lage, die Verarbeitung von Daten zu überwachen und bei Fehlern automatisch auf einen anderen Knoten im Cluster umzuschalten, um sicherzustellen, dass die Verarbeitung fortgesetzt wird. Diese Funktion ist besonders wichtig in Umgebungen, in denen die Verarbeitung von Daten kritisch ist.
Apache Flink ist eine leistungsstarke und flexible Plattform für die Verarbeitung von Big Data, insbesondere für Echtzeit-Datenströme. Es wird von vielen großen Unternehmen und Organisationen weltweit eingesetzt, um Echtzeitereignisse zu verarbeiten, Datenanalyse durchzuführen und wertvolle Einblicke in Daten zu gewinnen.

4.5 Entscheidung

Nach der Betrachtung allen Vor- und Nachteilen und der Rücksprache mit meiner Sachgebietsleitung entschied ich mich, die Umsetzung mit dem Streaming Framework Apache Flink in Kombination mit der Message Queue Apache Kafka umzusetzen. Beide Frameworks ließen sich gut in die bestehende Infrastruktur integrieren oder sind bereits in einer ausreichenden Form vorhanden.

5 Komponenten

Nachfolgend sollen die einzelnen Komponenten genauer beschrieben und ihre besonderen Merkmale erläutert werden. Sowohl die TableAPI als auch die DataStreamAPI des Apache Flink Frameworks wurden für die Implementierung der einzelnen Verarbeitungsschritte verwendet. Durch den Einsatz beider APIs konnte einerseits eine umfassende Einarbeitung in das Framework erreicht werden, andererseits konnten die jeweiligen Vorteile in der Entwicklung genutzt und der benötigte Zeitaufwand erheblich reduziert werden. Für die verbindende Schicht wird Apache Kafka als Message Queue verwendet.

5.1 importer

Der Flink-Job namens "importer" stellt den ersten Schritt der in Punkt 2.3 beschriebenen Pipeline dar. Mit dieser Anwendung soll zunächst eine CSV-Datei ausgelesen und anschließend mit der entsprechenden Formatierung in ein Kafka-Topic geschrieben werden. Für die Implementierung dieses Jobs wird die TableAPI von Apache Flink verwendet. Die TableAPI eignet sich besonders für die Definition von Eingangs- und Ausgangsdatentypen. Bei der Implementierung der Datenquelle (Source) und Datensenke

(Sink) wird analog vorgegangen. Es wird jeweils eine Tabelle mit den entsprechenden Optionen erstellt, die einen Namen, einen Anschluss (Connector), eine Beschreibung der Datentypen und eine variable Anzahl an Optionen enthält. Im konkreten Fall der Source (siehe Abbildung 6) wurde die Konfiguration für eine CSV-Datei vorgenommen. Dazu musste das Dateisystem als Anschluss und das Format "CSV" ausgewählt werden. Zusätzlich können Optionen wie das spezielle Trennzeichen "csv.field-delimiter" konfiguriert werden.

```
table_env.create_table(
    "source",
    TableDescriptor.for_connector('filesystem')
    .schema(get_csv_schema())
    .option("csv.ignore-parse-errors", "true")
    .option("csv.field-delimiter", "|")
    .option("path", csv_file)
    .format("csv")
    .build()
)
```

Abbildung 6: Beispielhafte Erstellung einer Tabelle in Apache Flink

Nach erfolgreicher Erstellung der beiden source- und sink-Tabellen konnte die Programmlogik durch einen einzigen Aufruf realisiert werden (siehe Abbildung 7). Für den Transfer in das Kafka-Topic wurden einfach alle Elemente der source-Tabelle ausgewählt und anschließend in die sink-Tabelle eingefügt.

```
table_env.sql_query("SELECT * FROM source")\
    .execute_insert("sink")\
    .wait()
```

Abbildung 7: Flink Table API schreiben von sink nach source

Im Anschluss an diesen Prozess können alle weiteren Flink-Jobs auf Kafka-Topics als Eingang zurückgreifen und müssen nicht mehr direkt mit dem Filesystem interagieren.

5.2 port-call-extractor

Korrigierter Text: Der Flink-Job zur Erkennung von Aktivitäten in bestimmten Seegebieten wurde mithilfe der TableAPI von Apache Flink erstellt. Die Grundlage für eine performante Berechnung bildet in dieser Anwendung der h3-Algorithmus von Uber (Overview of the H3 Geospatial Indexing System | H3 o. J.) . Mithilfe der von h3 bereitgestellten räumlichen Indizierung von Koordinaten konnte die geforderte Funktionalität durch einen einfachen

Join der TableAPI erreicht werden. Dazu wurden zwei Tabellen mit den Namen "simplified_ais_input" und "area_of_interest_cells" erstellt (siehe Abbildung 8).

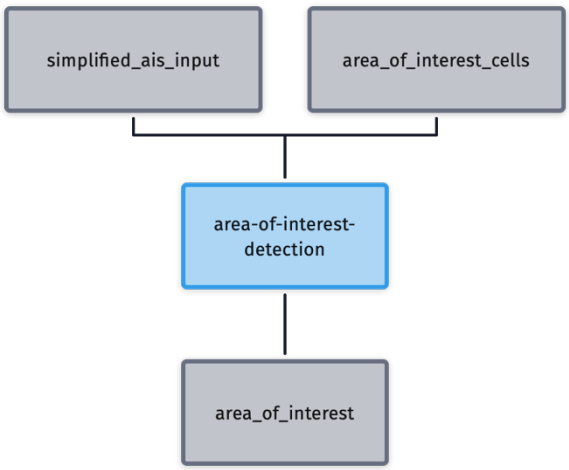


Abbildung 8: Übersichtsdiagramm area-of-interest-detection

Um nun die AIS-Nachrichten von Schiffen, die sich in einem vorher definierten Seegebiet aufhalten, mit einem entsprechenden Label zu versehen, müssen die im ersten Schritt erstellten Tabellen kombiniert werden. Für diese Aufgabe bietet SQL mehrere Optionen (siehe Abbildung 9), welche vor der Implementierung auf ihre Tauglichkeit untersucht werden mussten.

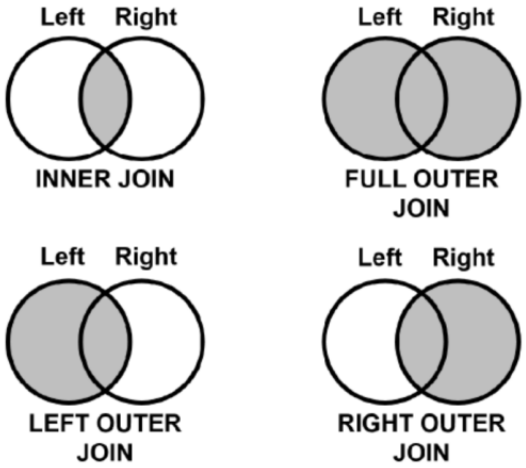


Abbildung 9: Übersicht der SQL Joins (Kumar 2023)

In dem vorliegenden Fall eignet sich der INNER JOIN. Diese Variante liefert ein Ergebnis, wenn der Join-Schlüssel in beiden Tabellen vorhanden ist. Als Join-Schlüssel wurde in diesem Fall die H3-Zelle gewählt. Ist diese in beiden Tabellen vorhanden, so ist auch die Anwesenheit des Schiffes in dem jeweiligen Seegebiet gegeben. Im Anschluss werden alle AIS-Datensätze in das Kafka-Topic mit dem Namen "area_of_interest" geschrieben und stehen für weitere Analysen zur Verfügung.

5.3 trajectory-simplification

Der Flink-Job zur Reduzierung oder Vereinfachung der gesendeten AIS-Daten war die Komponente mit der größten Komplexität und höchster Priorität. Die variable Häufigkeit der Aussendungen des AIS-Protokolls (bis zu 30 Nachrichten pro Minute) führt zu erheblicher Redundanz der Daten (Lee und Cho 2022). Mit der Trajektorie-Vereinfachung sollen diese Wiederholungen reduziert werden, sodass alle relevanten Informationen für weitere Verarbeitungsschritte erhalten bleiben, aber dennoch eine nennenswerte Reduktion erfolgt. Dieser Flink-Job sollte mit der DataStream-API umgesetzt werden, da sie sich für diese Art der Verarbeitung durch Keyed State eignet (Stateful Stream Processing o. J.). Unter dem State versteht man den Zustand eines durch einen Schlüssel definierten Objekts. Als Schlüssel wurde die Maritime Mobile Service Identities (MMSI) gewählt, die eine eindeutige Zuordnung eines Schiffes ermöglicht. Neben Kurs, Geschwindigkeit werden auch weitere dynamische Informationen wie z.B. der Tiefgang im Zustand gespeichert.

Für die Erkennung relevanter Nachrichten wird zunächst der erste Datenpunkt jedes Schiffs im jeweiligen State gespeichert. Anschließend werden alle neuen Nachrichten mit dem vorherigen Zustand im State verglichen. Wird eine wesentliche Änderung festgestellt, z.B. ein Kurs, der um mehr als fünf Grad vom vorherigen Zustand abweicht, wird diese Nachricht als neuer State gespeichert und zugleich in das Kafka-Topic mit dem Namen simplified_ais_input übertragen. Bei markanten Änderungen der Geschwindigkeit wird auf die gleiche Art verfahren. Wenn man sich bei der Vereinfachung der Trajektorie auf Änderungen von Geschwindigkeit oder Kurs beschränkt, kann es passieren, dass für Schiffe, die über einen längeren Zeitraum mit derselben Geschwindigkeit und dem gleichen Kurs fahren, keine weiteren Datenpunkte exportiert werden. Um dies zu vermeiden, wurde die Uhrzeit als weiteres Merkmal aufgenommen. Wenn der Zeitstempeldifferenz zwischen dem State und der aktuellen Nachricht größer als fünf Minuten ist, wird die Nachricht automatisch als relevant erkannt und in den State übernommen sowie in das Kafka-Topic exportiert. Auf diese Weise wird sichergestellt, dass für jedes Schiff mindestens alle fünf Minuten eine AIS-Nachricht zur Verfügung steht (siehe Abbildung 9).

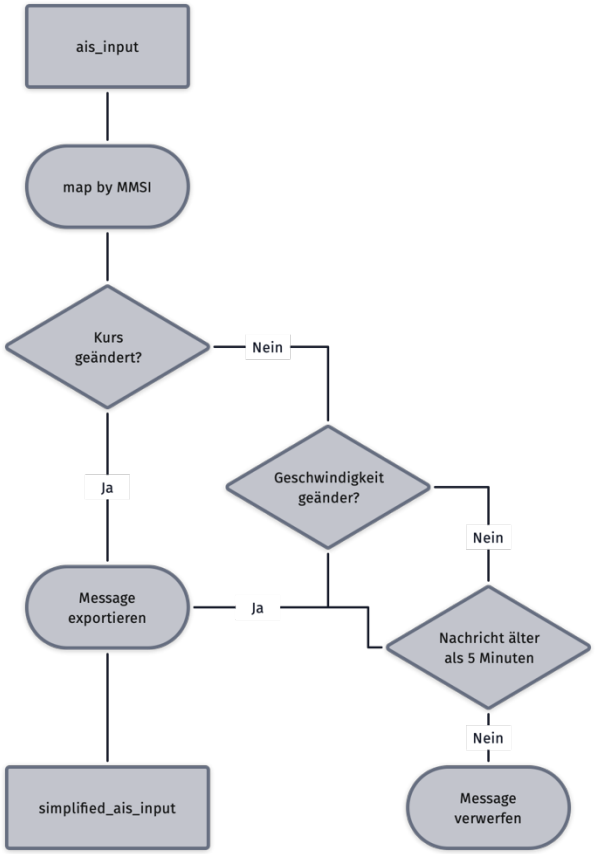


Abbildung 10: trajectory-simplification Ablauf

5.4 port-call-extractor

Der Begriff "Port-Call" bezieht sich auf jeden Aufenthalt eines Schiffes, der der Be- oder Entladung von Ladung oder dem Ein- oder Ausschiffen von Passagieren dient (What Does the Term “Port Call” Mean? o. J.). Wie bereits beim Flink-Job "area-of-interest-detection" (siehe 5.2) wurde auch diese Anforderung mithilfe der TableAPI von Flink und dem Einsatz von Joins umgesetzt. Die Abfrage musste um ein weiteres Element ergänzt werden, nämlich die Überprüfung, ob das Schiff eine Geschwindigkeit von null Knoten sendet. Durch die Abfrage und die anschließende Join-Operation konnten alle Schiffe, die sich im Hafengebiet aufhalten und gleichzeitig keine Geschwindig melden, in ein Kafka-Topic mit dem Namen "port_calls" geschrieben werden.

5.5 dark-vessel-detection

Wie schon der importer (siehe 5.1) und die area-of-interest-detection wurde auch die Erkennung der dark vessel mit der TableAPI von Apache Flink umgesetzt. Unter dark vessel versteht man Schiffe, die trotz der Verpflichtung ohne aktivierte AIS-Sendeanlagen fahren (Graziano und Renga 2021). Das Fehler der Aussendungen stellt immer ein besonderes Merkmal da und sollte daher automatisch erkannt und markiert werden.

Hierfür wird zunächst der jeweils letzte Zeitstempel der jeweiligen Schiffe (als Erkennungsmerkmal dient hier wieder die MMSI) gespeichert, in zyklischen Abständen wird

im Anschluss die Aktualität des letzten Zeitpunktes der Erfassung mit der aktuellen Zeit verglichen. Wird hierbei eine erhebliche Diskrepanz festgestellt, in der aktuellen Umsetzung wurde eine Stunde als Schwellwert definiert, so wird dieses Schiff mit dem Kenner *dark_vessel* versehen. Erscheint ein Schiff mit dieser *dark_vessel* Kennung zu einem späteren Zeitpunkt wieder, kann der Kenner für die jeweilige MMSI wieder aufgehoben werden. Jede dieser Aktionen (setzen und aufheben des Kenners) wird in einem Kafka Topic mit dem Namen dark_vessels hinterlegt und kann zu einem späteren Zeitpunkt nachvollzogen und analysiert werden.
Als mögliche Anwendungsfall kann hier eine Benachrichtigung oder die visuelle Markierung auf einer Karte genannt werden.

6 Fazit

Rückblickend lässt sich sagen, dass alle zu Beginn des Praktikums festgelegten Ziele erreicht wurden. Während der Umsetzung konnte ich mein bereits vorhandenes Wissen erweitern und viele Themen erlernen, die nicht Teil des DACS-Studiengangs sind. Insbesondere die beiden Frameworks Apache Kafka und Apache Flink und ihre besondere Art der Verarbeitung werden sich positiv auf mein weiteres Berufsleben auswirken.

Basierend auf der in diesem Praktikum entwickelten Anwendung können weitere Schritte unternommen werden. Ein mögliches Anwendungsgebiet wäre die Entwicklung eines Dashboards zur Visualisierung der einzelnen Kafka Topics.

Literaturverzeichnis

„Apache Flink: Use Cases“. https://flink.apache.org/usecases.html (2. Februar 2023).

„Apache Kafka“. *Apache Kafka*. https://kafka.apache.org/documentation/#gettingStarted (16. Februar 2023).

„Automatic Identification System (AIS) Overview | Navigation Center“. https://navcen.uscg.gov/automatic-identification-system-overview (12. Januar 2023).

Graziano, Maria Daniela, und Alfredo Renga. 2021. „Towards Automatic Recognition of Wakes Generated by Dark Vessels in Sentinel-1 Images“. *Remote Sensing* 13(10): 1955.

„Join“. //nightlies.apache.org/flink/flink-docs-release-1.16/docs/dev/table/hive-compatibility/hive-dialect/queries/join/ (16. Februar 2023).

Khan, Kamran u. a. 2014. „DBSCAN: Past, present and future“. In *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, , 232–38.

Kumar, Ajitesh. 2023. „Types of SQL Joins Explained with Examples“. *Data Analytics*. https://vitalflux.com/types-of-sql-joins-explained-with-examples/ (16. Februar 2023).

Lee, Wonhee, und Sung-Won Cho. 2022. „AIS Trajectories Simplification Algorithm Considering Topographic Information“. *Sensors* 22(18): 7036.

Maarek, Stéphane. 2022. „The Kafka API Battle: Producer vs Consumer vs Kafka Connect vs Kafka Streams vs KSQL !“ *Medium*. https://medium.com/@stephane.maarek/the-kafka-api-battle-producer-vs-consumer-vs-kafka-connect-vs-kafka-streams-vs-ksql-ef584274c1e (15. Februar 2023).

„Messaging that just works — RabbitMQ“. https://www.rabbitmq.com/ (16. Februar 2023).

Narkhede, Neha, Gwen Shapira, und Todd Palino. 2017. *Kafka: The Definitive Guide*. O’Reilly Media, Inc.

„Overview of the H3 Geospatial Indexing System | H3“. https://h3geo.org/docs/core-library/overview (16. Februar 2023).

„RDD Programming Guide - Spark 3.3.1 Documentation“. https://spark.apache.org/docs/latest/rdd-programming-guide.html (15. Februar 2023).

„Stateful Stream Processing“. //nightlies.apache.org/flink/flink-docs-master/docs/concepts/stateful-stream-processing/ (15. Februar 2023).

Tandon, Akash u. a. 2022. *Advanced Analytics with PySpark: Patterns for Learning from Data at Scale Using Python and Spark*. First edition. Beijing Boston Farnham Sebastopol Tokyo: O’Reilly.

„What Does the Term “Port Call” Mean?“ https://help.fleetmon.com/en/articles/4149410-what-does-the-term-port-call-mean (14. Februar 2023).

„Which protocols does RabbitMQ support? — RabbitMQ“. https://www.rabbitmq.com/protocols.html (16. Februar 2023).

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt habe. Die aus fremden Werken wörtlich oder sinngemäß übernommenen Texte oder Textteile sind unter Angabe der Quellen gekennzeichnet.

Ich versichere, dass ich bisher keine Prüfungsarbeit mit gleichem Thema bei einer Behörde oder Verwaltungseinrichtung vorgelegt habe.

Augsburg, 17.02.2023 _____

Ort, Datum Unterschrift