Prof. Dr. Bernhard Hollunder
Fakultät Informatik
bernhard.hollunder@hs-furtwangen.de

Hochschule Furtwangen
Robert-Gerwig-Platz 1
D-78120 Furtwangen

Wintersemester 2022/23

Streaming Systems

Praktikum

Bearbeitungszeitraum: 13. Oktober 2022 - 26. Januar 2023

In den Praktikumsaufgaben werden die in der Vorlesung vorgestellten Konzepte und Technologien mit Hilfe von unterschiedlichen Frameworks und Systemen praktisch erprobt. Teilweise werden die Lösungen früherer Aufgaben angepasst oder erweitert. Sie sollten daher beginnend mit Aufgabe 1 "saubere" und erweiterbare Lösungen erstellen.

Die angegebenen Bearbeitungstermine dienen zur zeitlichen Orientierung.

Hinweise zur Abgabe Ihrer Lösungen und den Bewertungskriterien finden Sie am Ende des Dokuments.

Aufgabenübersicht

1. Event Sourcing, nativ, 13. Oktober
2. Event Sourcing, nativ, erweiterte Funktionalität, 20. Oktober
3. Event Sourcing, Event Store mit JMS, 27. Oktober
4. Event Sourcing, Event Store mit Apache Kafka, 3. November
5. Datenanalyse "Auswertung von Taxifahrten" mit Apache Kafka, 10. - 24. November
6. Datenanalyse "Verkehrsüberwachung" mit Apache Beam, 1. und 8. Dezember
7. Complex Event Processing "Verkehrsüberwachung" mit EPL und Esper, 15. und 22. Dezember
8. *Wird zu gegebener Zeit noch festgelegt*

Aufgabe 1 [13. Oktober]

Erstellen Sie eine *Event Sourcing* Anwendung, mit der Positions- und Zustandsänderungen von Objekten verwaltet werden können, die das Interface `MovingItem` implementieren:

```
public interface MovingItem {
   String getName();
   int[] getLocation();
   int getNumberOfMoves();
   int getValue();
}
```

Jedes `MovingItem` Objekt hat einen eindeutigen Namen, eine Position in einem n-dimensionalen Koordinatensystem sowie einen Wert vom Typ `int`. Für Objekte in einem 3D-Raum werden die 3 Werte an den ersten 3 Positionen in dem `int`-Array abgelegt. Die Anzahl der bei einem Objekt durchgeführten Positionsänderungen kann über `getNumberOfMoves()` abgefragt werden.

Prof. Dr. Bernhard Hollunder
Fakultät Informatik
bernhard.hollunder@hs-furtwangen.de

Hochschule Furtwangen
Robert-Gerwig-Platz 1
D-78120 Furtwangen

Über die folgenden Befehle können Objekte erzeugt, vernichtet und deren Zustände verändert werden:

```
public interface Commands {
   void createItem(MovingItem movingItem);
   void deleteItem(String id);
   void moveItem(String id, int[] vector);
   void changeValue(String id, int newValue);
}
```

Ein neues Objekt wird über `createItem(...)` erzeugt, wobei die initiale Position der Nullpunkt ist und `value` den Wert 0 hat. Mit `moveItem(...)` wird der Bewegungsvektor angegeben. Das Argument [2, -4, 7] beispielsweise bedeutet, dass die aktuelle x-Koordinate um 2 Einheiten erhöht wird, die y-Koordinate um den Wert 4 verringert wird und die z-Koordinate sich um 7 Einheiten erhöht.

Über eine Query-Schnittstelle können u.a. die aktuellen Positionen der Objekte abgefragt werden:

```
public interface Query {
   public MovingItemDTO getMovingItemByName(String name);
   public Enumeration<MovingItemDTO> getMovingItems();
   public Enumeration<MovingItemDTO> getMovingItemsAtPosition(int[] position);
}
```

Beachten Sie, dass für die Rückgabe von Daten auf der Query-Seite der eigenständige Typ `MovingItemDTO` (anstatt `MovingItem`) verwendet wird. Dies scheint unnötig zu sein, erzeugt jedoch eine starke Entkopplung der *Write Side* von der *Read Side*. Definieren Sie einen geeigneten Typ `MovingItemDTO`.

Erstellen Sie – basierend auf diesen Schnittstellen-Vorgaben – eine *Event Sourcing* Anwendung gemäß der Komponenten der Grobarchitektur von Folie "CQRS with Event Sourcing". Hierzu können Sie wie folgt vorgehen:

- Erstellen Sie eine Implementierungsklasse für `MovingItem`.
- Erstellen Sie eine Implementierungsklasse für `Commands` bzw. einen *Command Handler*. Sie benötigen ein Domänenmodell, um Validierungen durchführen zu können. Soll beispielsweise ein weiteres Objekt mit einem bereits vergebenen Namen angelegt werden, so wird dieses Kommando zurückgewiesen.
- Einfachheitshalber soll in dieser Aufgabe das Domänenmodell nicht auf Basis der im *Event Store* abgelegten Ereignisse aufgebaut werden. Stattdessen kann eine Map zur Verwaltung der aktuell vergebenen Objektnamen verwendet werden, weitere Informationen werden im Domänenmodell (zunächst) nicht benötigt. Wird ein Kommando akzeptiert, werden die erforderlichen Ereignisse erzeugt. Ein Beispiel für ein Ereignistyp ist etwa `MovingItemCreatedEvent`. Im einfachsten Fall wird ein Kommando auf ein entsprechendes Ereignis abgebildet. Je nach Kommando muss auch das Domänenmodell aktualisiert werden.
- Die erzeugten Ereignisse werden im *Event Store* abgelegt. Verwenden Sie hierzu eine Datenstruktur wie etwa eine `BlockingQueue`. Hinweis: In späteren Aufgaben kommen an dieser Stelle verteilte Messaging Systeme zum Einsatz.
- Implementieren Sie die `Query` Schnittstelle. Hierzu benötigen Sie einen *Query Handler* als eigenständige Komponente. Überlegen Sie sich eine geeignete Datenstruktur für das zugrundeliegende *Query Model* (z.B. eine Map), um Anfragen auf einfache Weise beantworten zu können.

- Nun benötigen Sie noch eine Projektion, die die im *Event Store* eingegangenen Ereignisse auswertet und das *Query Model* aktualisiert.

Erstellen Sie eine Client-Anwendung, die `MovingItem` Objekte erzeugt und deren Positionen und Werte ändert. Hierzu bietet es sich an, u.a. mit einem Zufallszahlengenerator zu arbeiten, der für die erstellten Objekte Bewegungsvektoren erzeugt und diese über den `moveItem(...)` Befehl anwendet.

Achten Sie auf eine strikte Trennung zwischen dem Domänenmodell und dem *Query Model*, sodass beide Komponenten unabhängig voneinander weiterentwickelt werden können. Überlegen Sie sich, wie Sie auf sinnvolle Weise die Korrektheit und Vollständigkeit Ihrer Lösung prüfen können. Setzen Sie nachfolgend das Testkonzept um.

Aufgabe 2 [20. Oktober]

Erweitern Sie Ihre Lösung von Aufgabe 1 dahingehend, dass folgendes Verhalten umgesetzt wird. Nach einer bestimmten Anzahl von Bewegungen (z.B. 20) eines Objektes soll dieses entfernt werden. Wird also das zwanzigste Mal der `moveItem(...)` Befehl für ein Objekt angewendet, wird diese Bewegung nicht ausgeführt. Stattdessen wird das Objekt gelöscht, der Name des Objektes wird freigegeben und kann nachfolgend wieder vergeben werden. Überlegen Sie sich, wie das Domänenmodell auf einfache Weise erweitert werden kann, um diese Funktionalität umzusetzen.

Betrachten Sie nun folgende Erweiterung. Bei der Durchführung von `moveItem(...)` mit einem Argument ungleich dem Nullvektor bei einem Objekt soll überprüft werden, ob sich auf der neuen Position bereits ein Objekt befindet. In diesem Fall soll letzteres entfernt werden. Wie kann diese Funktionalität umgesetzt werden? Ist es sinnvoller das Domänenmodell geeignet anzupassen oder sollte eher eine erweiterte Query-Schnittstelle genutzt werden? Prüfen Sie die Möglichkeiten und setzen Sie eine Variante um.

Aufgabe 3 [27. Oktober]

Modifizieren Sie die Lösung von Aufgabe 2 dahingehend, sodass ein JMS Messaging System wie etwa Apache ActiveMQ zur Verwaltung der Ereignisse genutzt wird. Der *Command Handler* nimmt hierbei die Rolle des Nachrichtenproduzenten ein; die Projektion konsumiert die erzeugten JMS-Ereignisse und aktualisiert das *Query Model*.

Ermitteln Sie auch die durchschnittliche Zeit für den Transport eines Ereignisses vom Produzenten zum Konsumenten.

Aufgabe 4 [3. November]

Setzen Sie nun Apache Kafka zur (persistenten) Speicherung der Ereignisse im *Event Store* ein.

Des Weiteren soll nun das Domänenmodell auf Basis der im *Event Store* gespeicherten Ereignisse dynamisch bei der Abarbeitung eines Befehls aufgebaut werden. Realisieren Sie entsprechende *Event Store*-Abfragen wie etwa `loadNamesOfMovingItems()` als Apache Kafka Konsumenten. Dies bedeutet auch, dass die in Aufgabe 1 - 3 verwendete Map zur Speicherung der Objektnamen nicht mehr benötigt wird.

Aufgabe 5 [10. – 24. November]

Die Aufgabenbeschreibung finden Sie unter `http://www.debs2015.org/call-grand-challenge.html`. Erstellen Sie eine Lösung für *Query 1: Frequent Routes*. In aller Kürze: es sollen Datensätze von Taxifahrten im Großraum New York analysiert werden. Es geht darum, die am häufigsten gefahrenen Strecken innerhalb eines festgelegten Zeitraumes (hier 30 Minuten) zu

bestimmen. Eine Strecke wird durch ein Start- und Zielsegment festgelegt. Details können Sie der DEBS-Aufgabenbeschreibung entnehmen.

Setzen Sie Apache Kafka als Messaging System ein. Sie können sich an folgenden Schritten orientieren:

1. Erstellen Sie einen Datentyp zur Repräsentation einer Taxifahrt. Schreiben Sie einen Importer, der die Einträge aus einer Datei einliest und entsprechende Taxifahrt Instanzen erzeugt.
2. Erstellen Sie einen Apache Kafka Produzenten, der für die eingelesenen Fahrten *Records* erzeugt und diese in ein Topic einstellt.
3. Ein Apache Kafka Konsument liest die *Records* aus dem Topic und verarbeitet diese gemäß *Query 1: Frequent Routes*. Verwenden Sie das in der Aufgabenbeschreibung definierte Ausgabeformat. Geben Sie auch die Zeitdifferenz zwischen Einlesen und Verarbeitung eines Datensatzes aus.

Hinweis zu den Datensätzen. Die Datei `sample_data.csv` enthält 10.000 Fahrten und folgt dem Datenformat in der Aufgabenbeschreibung.

Für umfangreiche Performanztests kann mit der Datei `trip_data_1.csv` gearbeitet werden. Diese Datei enthält 14.776.616 Fahrten. Achtung: die Reihenfolge der Attribute einer Fahrt unterscheidet sich von dem DEBS-Format. Information zu den Metadaten finden Sie in der Datei `trip_data_1-MetaData.txt`. Passen Sie Ihren Importer entsprechend an. Wem dies immer noch zu wenige Daten sein sollten: es stehen noch weitere große Dateien zur Verfügung.

Für ganz Neugierige / Mutige und außer Konkurrenz: Wo es eine *Query 1* gibt, kann eine *Query 2* auch nicht weit weg sein.

Aufgabe 6 [1. und 8. Dezember]

Erstellen Sie in dieser Aufgabe eine Apache Beam Pipeline zur Aufbereitung und Aggregation von simulierten Sensordaten. Hierzu entwickeln Sie zunächst einen Testdatengenerator, der Datensätze mit folgender Struktur erzeugt: `id: val-0, val-1, val-2, ..., val-n`. Beispiel:

```
2: 20.4,33.5,40.0
1: 35.6,17.4,28.6
2: 42.1,34.7
1:
3: 28.4,21.1
```

Ein Datensatz soll folgendermaßen interpretiert werden: Die führende Zahl legt den Namen einen Sensors fest. Die nachfolgenden Zahlen sind die zu einem Zeitpunkt gemessenen Geschwindigkeitswerte in der Einheit Meter pro Sekunde. Die Anzahl der Sensoren auf einem Streckenabschnitt sowie die Anzahl der zu einem Zeitpunkt erzeugten Geschwindigkeitswerte sowie die minimale und maximale Geschwindigkeit soll konfigurierbar sein. Ebenso kann die Taktung, also der zeitliche Abstand zwischen zwei erzeugten Datensätzen, durch zwei `int` Zahlen `m1` und `m2` eingestellt werden. Dies bedeutet, dass der nächste Datensatz mindestens `m1` und höchstens `m2` Millisekunden nach dem vorhergehenden erzeugt wird. Es sollen gelegentlich auch negative Geschwindigkeitswerte produziert werden.

Die Datensätze sollen in einen Apache Kafka Topic eingestellt werden. Eine Apache Beam Pipeline soll diese Datensätze über den `KafkaIO` Adapter einlesen und nachfolgend aufbereiten, bereinigen sowie aggregieren.

Die Aufgabe besteht darin, die Durchschnittsgeschwindigkeit in der Einheit km/h für die Sensoren in Zeitfenstern vorgegebener Länge (z.B. im 30 Sekundentakt) zu ermitteln. Diese Werte können nun genutzt werden, um

1. den zeitlichen Verlauf der Durchschnittsgeschwindigkeit an einer Messstelle (d.h. eines Sensors) sowie
2. die Durchschnittsgeschwindigkeiten auf einem Streckenabschnitt (definiert durch eine Folge von Sensoren) zu einem bestimmten Zeitpunkt

zu ermitteln.

Hinweise zum Vorgehen:

- Definieren Sie geeignete `PCollection`s und `Transform`s zur Ermittlung der geforderten Ausgabe.
- Verwenden Sie die Zeitstempel, die durch Apache Kafka vergeben werden. Der Testdatengenerator muss somit keine Zeitstempel vergeben.
- Ein Datensatz ohne Geschwindigkeitswert soll nicht berücksichtigt werden. Eine negative Geschwindigkeit (Messfehler) soll ebenso von der weiteren Verarbeitung ausgeschlossen werden.
- Es bietet sich an, mit `Transform`s wie etwa `GroupByKey`, `Combine` und `Window` zu arbeiten.
- Hinsichtlich der Darstellung der berechneten Ergebnisse können Sie eine textbasierte oder graphische Ausgabe verwenden. Die berechneten Werte sollten "intuitiv" dargestellt werden.

Überlegen Sie sich, wie Sie die Korrektheit Ihrer Lösung prüfen können.

Aufgabe 7 [15. und 22. Dezember]

Die Kernfunktionalität von Aufgabe 6 soll nun mittels CEP umgesetzt werden. Ergänzend sollen Sie "komplexe" Ereignisse identifizieren, definieren und umsetzen. Verwenden Sie hierzu das System Esper und EPL.

Erstellen Sie entsprechende EPL-Anfragen zur Datenbereinigung und Berechnung der Durchschnittsgeschwindigkeit.

Nachdem die Durchschnittsgeschwindigkeit für die Sensoren pro Zeitfenster berechnet wurden, sollen diese jeweils als (aggregiertes) Ereignis ist das System eingestellt werden. Definieren Sie hierzu einen entsprechenden Ereignistyp und verwenden Sie die `insert into` Klausel, um Instanzen von diesem Typ zu erzeugen.

Diese erzeugten Ereignisse sollen nun wie folgt weiter verarbeitet werden: Kommt es innerhalb einer bestimmten Zeit (z.B. 30 Sekunden) zu einem starken Abfall der Durchschnittsgeschwindigkeit, soll ein weiteres Ereignis erzeugt werden, welches auf eine mögliche Stauentwicklung hinweist. Definieren Sie hierzu einen weiteren Ereignistyp.

Überlegen Sie sich geeignete Testfälle, wie Sie Ihre die Korrektheit Ihrer EPL-Anfragen testen können.

Aufgabe 7 [12. - 26. Januar]

Diese Aufgabe wird zu gegebener Zeit bekannt gegeben.

Hinweise zur Abgabe und Bewertung

Neben der praktischen Umsetzung der gestellten Aufgaben sollen Sie Ihre Lösungen auch in schriftlicher Form dokumentieren. Erstellen Sie hierzu ein Dokument, in dem Sie die gewählten Lösungsansätze und erzielten Ergebnisse aufgabenweise darstellen und bewerten. Sie können Sie an folgenden Fragen orientieren:

- Welche Lösungs- und Umsetzungsstrategie wurde gewählt?
- Wie schätzen Sie die Leistungsfähigkeit der Lösung ein? Wurden neben der Basisfunktionalität zusätzliche Funktionen realisiert?
- Welche nicht-funktionale Anforderungen (z.B. Skalierbarkeit und Durchsatz) wurden untersucht? Mit welchem Ergebnis?
- Wie haben Sie sich von der Korrektheit und Vollständigkeit der Lösung überzeugt? Gibt es eine systematische Teststrategie?
- Wie lassen sich die berechneten Ergebnisse auf geeignete Weise darstellen? Gibt es naheliegende Visualisierungsmöglichkeiten?
- Welche zusätzlichen Frameworks und Bibliotheken wurden eingesetzt? Warum?

Aus diesen Fragen leiten sich die Bewertungskriterien ab. Beachten Sie, dass einige Aufgaben bewußt nicht vollständig spezifiziert wurden. Seien Sie kreativ und schließen Sie mögliche Lücken auf sinnvolle Weise.

Die Abschlusspräsentationen aller Lösungen finden gruppenweise nach individueller Absprache ab dem 30. Januar statt. Präsentationen von Zwischenergebnissen erfolgen in den Praktikumsstunden.

Die Lösungen sollten in Zweierteams erarbeitet werden.

Die Gesamtnote für das Praktikum setzt sich folgendermaßen zusammen: 2/3 lauffähige Lösungen der Aufgaben und 1/3 Dokumentation. Achten Sie auf “Lesbarkeit” der Dokumentation.