

MPCS 53014 Big Data Architecture Final Project

Shichen Sun

shichen@uchicago.edu

1. Overview

This project provides a platform for analyzing taxi trip data. It uses a Lambda architecture consisting of a **Batch Layer**, a **Speed Layer** and a **Serving Layer** to process and analyze large-scale taxi trip data in near real-time. The platform supports querying aggregated statistics (e.g., average fare, trip count, total fare) for specific taxi companies and displays results in a web-based interface.

The system leverages **Hive** and **HBase** for data storage, **Kafka** for streaming data ingestion, and **Node.js** for the web application layer. Users can input a taxi company name via the web interface and retrieve detailed statistics.

Data source is Kaggle: <https://www.kaggle.com/datasets/aungdev/chicago-transportation-2020?resource=download>

2. Architecture

2.1 Batch Layer

The batch layer is responsible for precomputing aggregated statistics for taxi companies. These statistics are stored in a dedicated HBase table (`shichen_taxi_trips_by_company`) and include:

- Total number of trips (`trip_count`)
- Total fare (`total_fare`)
- Total trip duration in seconds (`total_trip_seconds`)
- Most frequent pickup and dropoff areas (`top_pickup_area`, `top_dropoff_area`)

Workflow:

1. Raw data is processed in **Hive** and aggregated by company.
2. The results are written to the `shichen_taxi_trips_by_company` table in HBase.

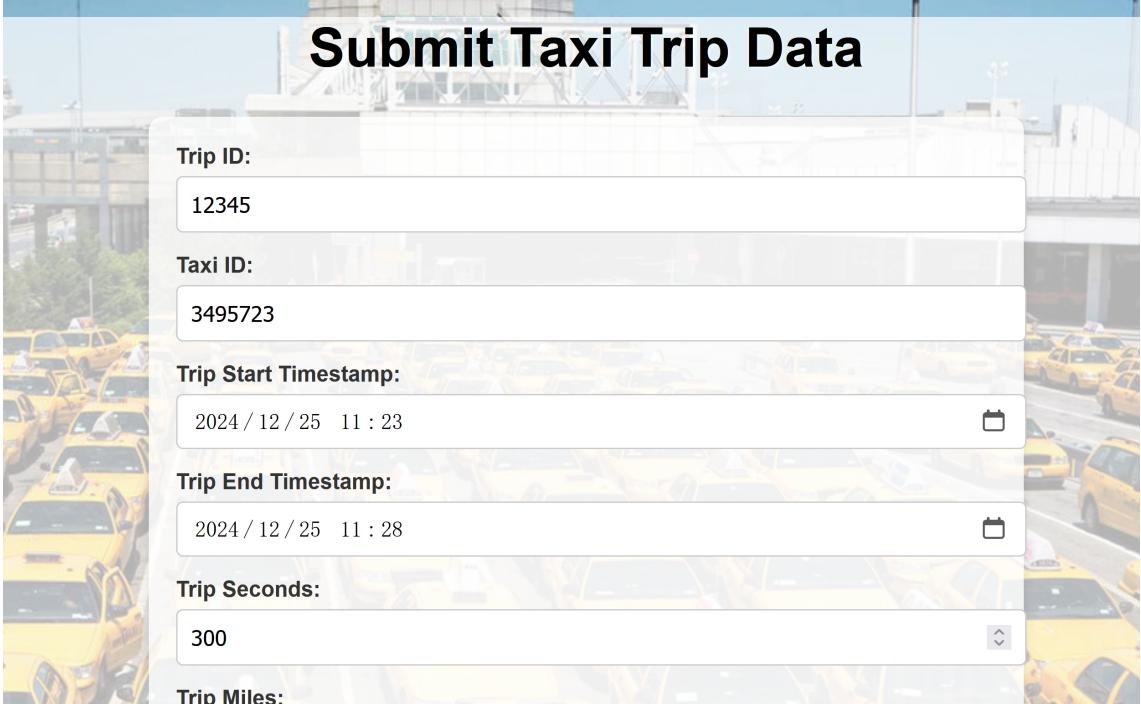
2.2 Speed Layer

The speed layer processes incoming taxi trip events in near real-time. It uses Kafka to stream data, and a Spark job consumes the data, calculates statistics, and writes them into an HBase table (`shichen_taxi_trips_speed`).

Workflow:

1. In the webpage, user can submit new taxi data and the data would be fed into my Kafka topic `shichen-taxi_trip-events`.

Submit Taxi Trip Data



A form for submitting taxi trip data. It includes fields for Trip ID, Taxi ID, Trip Start Timestamp, Trip End Timestamp, Trip Seconds, and Trip Miles. Each field has a placeholder value and a calendar icon for timestamp selection.

Trip ID:	12345
Taxi ID:	3495723
Trip Start Timestamp:	2024 / 12 / 25 11 : 23
Trip End Timestamp:	2024 / 12 / 25 11 : 28
Trip Seconds:	300
Trip Miles:	

2. A Spark Streaming application consumes Kafka events, computes statistics for the latest data, and writes the results to HBase.

```
{"tripId": "125", "taxiId": "TAXI002", "tripStartTimestamp": "2024-12-05T10:00:00Z", "tripEndTimestamp": "2024-12-05T10:00:00Z", "tripSeconds": "1", "tripMiles": "20.0", "fare": 25.5, "tips": "5.0", "tolls": "5.0", "extras": "5.0", "tripTotal": "5.0", "paymentType": "ABC", "company": "ABC", "pickupCentroidLatitude": "5.0", "pickupCentroidLongitude": "5.0", "dropoffCentroidLatitude": "5.0", "dropoffCentroidLongitude": "5.0"}, {"tripId": "UNKNOWN", "taxiId": "UNKNOWN", "tripStartTimestamp": "2024-12-07T23:20:55.750Z", "tripEndTimestamp": "2024-12-07T23:20:55.750Z", "tripSeconds": "0", "tripMiles": "0", "pickupCensusTract": "UNKNOWN", "dropoffCensusTract": "UNKNOWN", "pickupCommunityArea": 0, "dropoffCommunityArea": 0, "fare": 0, "tips": 0, "tolls": 0, "extras": 0, "tripTotal": 0, "paymentType": "CASH", "company": "UNKNOWN", "pickupCentroidLatitude": 0, "pickupCentroidLongitude": 0, "pickupCentroidLocation": "UNKNOWN", "dropoffCentroidLatitude": 0, "dropoffCentroidLongitude": 0, "dropoffCentroidLocation": "UNKNOWN"}, {"tripId": "UNKNOWN", "taxiId": "UNKNOWN", "tripStartTimestamp": "2024-12-07T23:23:50.780Z", "tripEndTimestamp": "2024-12-07T23:23:50.780Z", "tripSeconds": 0, "tripMiles": 0, "pickupCensusTract": "UNKNOWN", "dropoffCensusTract": "UNKNOWN", "pickupCommunityArea": 0, "dropoffCommunityArea": 0, "fare": 0, "tips": 0, "tolls": 0, "extras": 0, "tripTotal": 0, "paymentType": "CASH", "company": "UNKNOWN", "pickupCentroidLatitude": 0, "pickupCentroidLongitude": 0, "pickupCentroidLocation": "UNKNOWN", "dropoffCentroidLatitude": 0, "dropoffCentroidLongitude": 0, "dropoffCentroidLocation": "UNKNOWN"}, {"tripId": "14111", "taxiId": "123", "tripStartTimestamp": "2024-11-11T11:11", "tripEndTimestamp": "2024-11-12T11:11", "tripSeconds": "0", "tripMiles": "0", "pickupCensusTract": "UNKNOWN", "dropoffCensusTract": "UNKNOWN", "pickupCommunityArea": 0, "dropoffCommunityArea": 0, "fare": 0, "tips": 0, "tolls": 0, "extras": 0, "tripTotal": 0, "paymentType": "0", "company": "0", "pickupCentroidLatitude": 0, "pickupCentroidLongitude": 0, "pickupCentroidLocation": 0, "dropoffCentroidLatitude": 0, "dropoffCentroidLongitude": 0, "dropoffCentroidLocation": "0"}, {"tripId": "11111", "taxiId": "11111", "tripStartTimestamp": "2024-12-25T11:23", "tripEndTimestamp": "2024-12-25T11:53", "tripSeconds": "30", "tripMiles": "5", "pickupCensusTract": "UNKNOWN", "dropoffCensusTract": "UNKNOWN", "pickupCommunityArea": 0, "dropoffCommunityArea": 0, "fare": 20, "tips": 5, "tolls": 0, "extras": 0, "tripTotal": 0, "paymentType": "0", "company": "Taxi Affiliation Services", "pickupCentroidLatitude": 0, "pickupCentroidLongitude": 0, "pickupCentroidLocation": 0, "dropoffCentroidLatitude": 0, "dropoffCentroidLongitude": 0, "dropoffCentroidLocation": 0}, {"tripId": "12345", "taxiId": "3495723", "tripStartTimestamp": "2024-12-25T11:23", "tripEndTimestamp": "2024-12-25T11:28", "tripSeconds": "300", "tripMiles": "5", "pickupCensusTract": "UNKNOWN", "dropoffCensusTract": "UNKNOWN", "pickupCommunityArea": 0, "dropoffCommunityArea": 0, "fare": 346, "tips": 200, "tolls": 0, "extras": 0, "tripTotal": 0, "paymentType": "0", "company": "Taxi Affiliation Services", "pickupCentroidLatitude": 0, "pickupCentroidLongitude": 0, "pickupCentroidLocation": 0, "dropoffCentroidLatitude": 0, "dropoffCentroidLongitude": 0, "dropoffCentroidLocation": 0}
```

2.3 Serving Layer

Extracted several hbase tables for quick queries:

```
create 'shichen_taxi_trips_by_id', 'trip_info'

create 'shichen_taxi_trips_by_location', 'trip_info'

create 'shichen_taxi_trips_by_pickup', 'trip_info'

create 'shichen_taxi_trips_by_dropoff', 'trip_info'

create 'shichen_taxi_trips_by_date', 'trip_info'

create 'shichen_taxi_trips_by_company', 'trip_info'
```

```
create 'shichen_taxi_trips_by_fare', 'trip_info'
```

```
create 'shichen_taxi_trips_by_geo', 'trip_info'
```

And every time a query reaches, it would query both batch layer and speed layer, and combine the data together to show the final results:

The screenshot shows a web interface for 'Company Statistics'. At the top, there's a search bar labeled 'Company Name:' containing 'Blue Ribbon Taxi Association Inc.' Below it is a green button labeled 'Get Stats'. The background of the main area is a photograph of many yellow taxis on a city street. A modal window titled 'Statistics for Blue Ribbon Taxi Association Inc.' displays a table of metrics:

Metric	Value
Total Trips	324586
Total Fare	\$3743013.26
Total Trip Seconds	239585400.00
Average Fare	\$11.53

3. Components

3.1 Hive Tables

```
shichen_taxi_trips
```

```
shichen_taxi_trips_by_company_hive
```

```
shichen_taxi_trips_by_pickup_hive
```

```
shichen_taxi_trips_by_dropoff_hive
```

```
shichen_taxi_trips_company_aggregated
```

```
shichen_taxi_trips_by_fare_hive
```

Used to preprocess and aggregate historical taxi trip data for the Batch Layer.

3.2 HBase Tables

- `shichen_taxi_trips_by_company`:

Stores pre-aggregated statistics for each taxi company (Batch Layer).

- `shichen_taxi_trips_speed`:

Stores trip-level data for real-time updates (Speed Layer).

3.3 Kafka

- **Topic:** `shichen-taxi-trip-events`
- Streams raw taxi trip events to the Speed Layer.

3.4 Node.js Application

Queries the Batch Layer (`shichen_taxi_trips_by_company`) for precomputed statistics.

Queries the Speed Layer (`shichen_taxi_trips_speed`) for real-time statistics.

Merges results from both layers.

4. Setup

1. Start Spark Streaming Job:

```
spark-submit --master local[2] --driver-java-options "-Dlog4j.configuration=file:///home/hadoop/ss.log4j.properties" --class org.example.StreamTaxiTrips uber-speed_layer-1.0-SNAPSHOT.jar wn0-kafka.m0ucnnwuiqae3jdorci214t2mf.bx.internal.cloudapp.net:9092,wn1-kafka.m0ucnnwuiqae3jdorci214t2mf.bx.internal.cloudapp.net:9092
```

2. Start the Node.js Application:

```
node app.js 3174 http://10.0.0.26:8090 wn0-kafka:9092
```

3. Access the Web Interface:

After configuring the proxy following the instructions on the slides,

Submit the data at this page: <http://10.0.0.38:3174/submit-taxi.html>

Query the statistic at this page: <http://10.0.0.38:3174/>

5. Videos

Check this out to learn how my project works: https://drive.google.com/file/d/1CDOqh3xqAWUVDoZerBMe4ZfjN_0k1TE/view?usp=sharing