Real–time Analysis and Visualization Dashboard using Big-Data Approach

Project Group 10 : Kartik Nautiyal and Nitaant Vyas

Introduction

- Volume of data is increasing

REAL TIME DATA

- Real-time data
  - Sensors
  - Transactions

- Fast paced environment
  - Real-time monitoring

Problem Statement and Scope

- ETL Pipeline
- Connected to a real-time dashboard
- Big Data Approach

Dataset

Dataset - NYC Yellow Taxi Trip Data

- Contains transaction data about NYC Yellow cab taxi rides
- Time Period: Jan 2015 and Jan-March 2016.
- Real-World application: Uber/Lyft transactions
- Around 8 GB

Kaggle:
https://www.kaggle.com/datasets/elemento/nyc-yellow-taxi-trip-data

Dataset

Field Name	Description
tpep_pickup_datetime	The date and time when the meter was engaged.
tpep_dropoff_datetime	The date and time when the meter was disengaged.
Passenger_count	**The number of passengers in the vehicle. This is a driver-entered value.**
Trip_distance	**The elapsed trip distance in miles reported by the taximeter.**
Pickup_longitude	Longitude where the meter was engaged.
Pickup_latitude	Latitude where the meter was engaged.
RateCodeID	The final rate code in effect at the end of the trip.

Dataset

Field Name	Description
Store_and_fwd_flag	This flag indicates whether the trip record was held in vehicle memory before sending to the vendor,
Dropoff_longitude	Longitude where the meter was disengaged.
Dropoff_ latitude	Latitude where the meter was disengaged.
Payment_type	**A numeric code signifying how the passenger paid for the trip.**
Fare_amount	The time-and-distance fare calculated by the meter.
Extra	Latitude where the meter was engaged.

Dataset

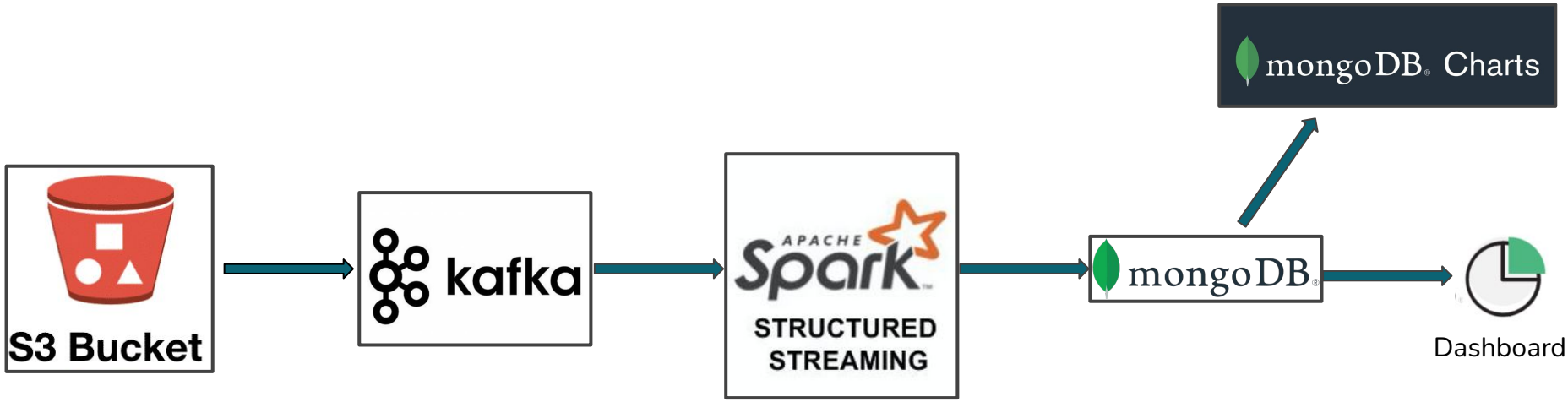
Field Name	Description
MTA_tax	0.50 MTA tax that is automatically triggered based on the metered rate in use.
Improvement_surcharge	0.30 improvement surcharge assessed trips at the flag drop. the improvement surcharge began being levied in 2015.
Tip_amount	Tip amount – This field is automatically populated for credit card tips.Cash tips are not included.
Tolls_amount	The elapsed trip distance in miles reported by the taximeter.
Total_amount	**The total amount charged to passengers. Does not include cash tips.**

Architecture

Problem Statement and Scope

- Implement ETL pipeline from S3 to MongoDB.
- Use S3 bucket as the source of “live data”.
- Implement it as the as Kafka Producer.
- Apache Spark as Kafka Consumer.
- Aggregate and save it in MongoDB.
- Visualize live Data in Mongo Charts.

Architecture Intuition



Data Sourcing: AWS S3

WHAT

- Amazon Web Services
  - on-demand cloud computing platforms
- Pay-as-you-go
- S3: Simple Storage Service (Distributed Storage)

HOW

- Container for files
  - One of their server locations
    - US-East-1 (Virginia)

- Access token and Secret keys

Why

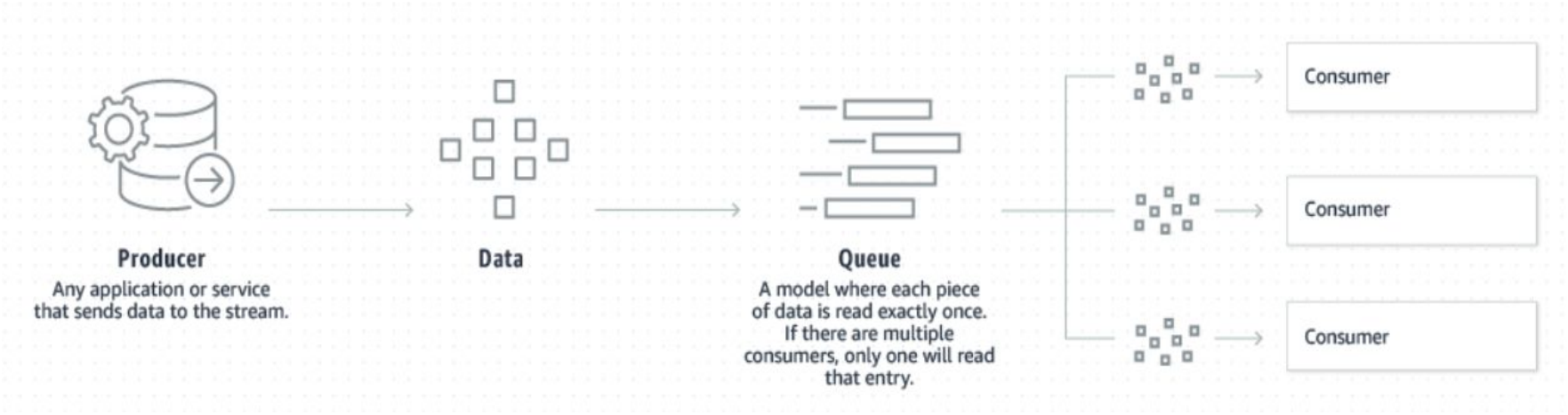
- Cutting Edge Technology and industry standard
- Cloud is the future
- Cheap

Technology 1: Apache Kafka

WHAT

- Decoupling of producer and consumer
- Ordered storage
- Process streams of records in real time

HOW


Producer
Any application or service that sends data to the stream.
Data
Queue
A model where each piece of data is read exactly once. If there are multiple consumers, only one will read that entry.
Consumer
Consumer
Consumer

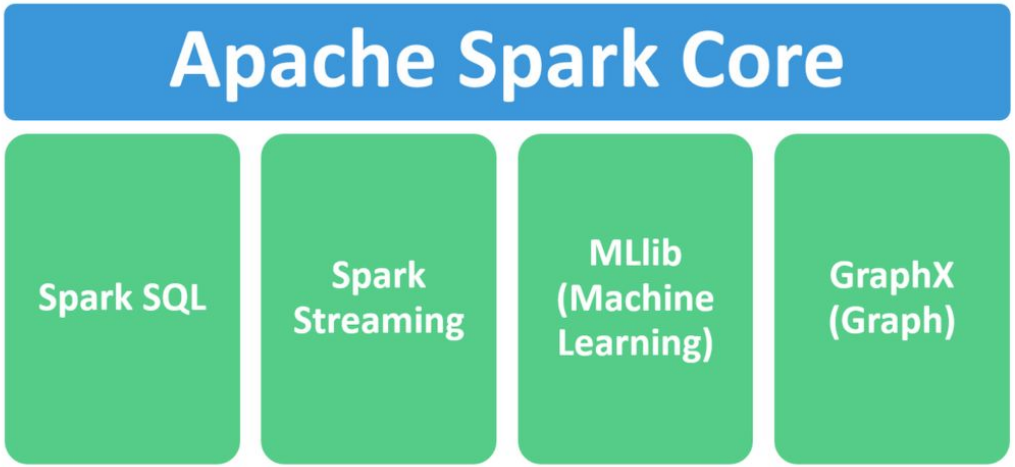
Why

- Queuing service requirement
- Easy Implementation

Technology 2: Apache Spark

WHAT

- Data processing framework



HOW

- Master-worker architecture
- In memory data processing (faster than hadoop map-reduce)
- RDDs, SparkSQL DataFrames, Streams

Why

- Stream processing requirement (Spark Structured Streaming)
- Python Integration (PySpark)
- Writing batch jobs and output mode (mongo)

Technology 3: MongoDB & MongoCharts

WHAT

- MongoDB is an open source NoSQL database management program
- Distributed data management
- Mongocharts: visual representation of MongoDB data

HOW

- Collections have documents
- Json like structure
- Mongocharts connects to MongoDB URI

Why

- Industry moving to No SQL
- Mongo Charts
- Spark - mongo connector

Technology: Docker

WHAT

- Platform that allows you to build, test, and deploy applications quickly.

- Docker packages software into standardized units called containers that have everything the software needs to run

HOW

- Docker containers
- Docker volumes
- Docker Networks

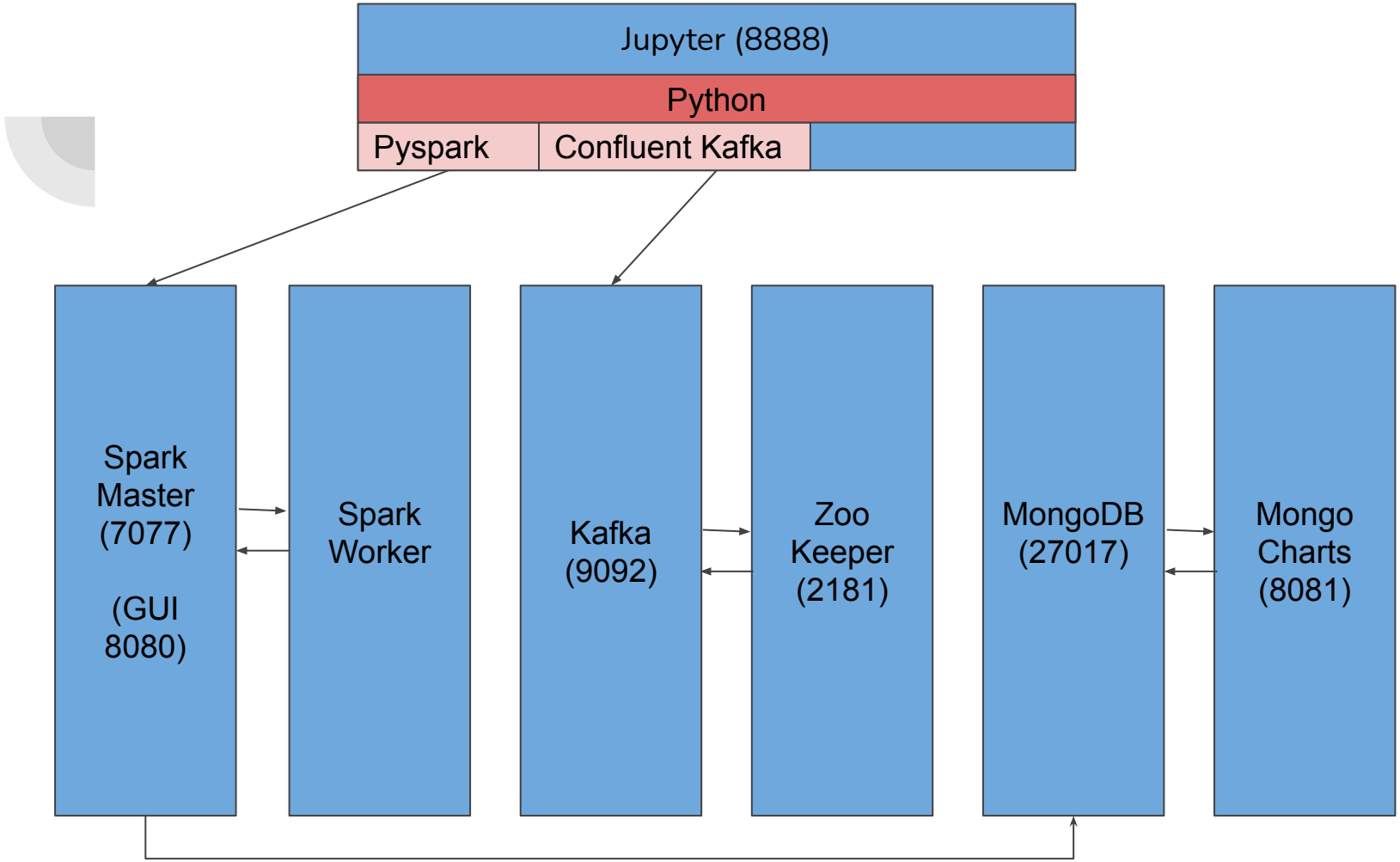
Why

- Industry adoption
- Container isolation
- Environment and Dependency issues

Implementation Structure

System

- Windows running with docker
- 8 GB ram
- Intel Core i7 8750H @ 2.2GHz


Jupyter (8888)
Python
Pyspark
Confluent Kafka
Spark Master (7077)
(GUI 8080)
Spark Worker
Kafka (9092)
Zoo Keeper (2181)
MongoDB (27017)
Mongo Charts (8081)

Current Analysis Approach

Spark


Incoming batch
(10 seconds)
Spark
Outgoing batch
(10 at max)
10 rows, 19
Columns
10 rows, 3
Columns

Demo

Future Analysis Approach

Spark


Incoming batch
(1 minute)
Spark
Outgoing batch
(10 at max)
100 rows, 19
Columns
Aggregated
row, 3
Columns

Results

- Live dashboard on Mongodb charts (refreshes every minute)
- Lazy evaluation
- Everything containerized

Challenges and Issues

- Kafka python libraries
- Spark structured streaming (Writestream)
- CPU and memory issue
- Checking mongo data (maybe mongo atlas?)

Future Work

- Scale of the data can be increased.
- Retention time of kafka topic can be configured
- Dashboard can be developed further
- Analysis approach can be implemented

THANK YOU!

Questions / Suggestions/ Comments?