**Coffee Shop**

**Kafka**

**Connect**



My data pipe constructed as follows:

1. There is a table in MySql data base named as **coffee**

Table’s script is here:

CREATE TABLE `coffee` (

`id` int(11) NOT NULL,

`product` varchar(45) DEFAULT NULL,

`price` double DEFAULT NULL,

`ptime` datetime DEFAULT NULL,

PRIMARY KEY (`id`)

//sample data

'1', 'COFFEE', '5', '2020-08-25 12:09:49'

'2', 'COFFEE', '5', '2020-08-25 12:10:40'

'3', 'TEA', '3', '2020-08-25 12:40:21'

'4', 'PIE', '10', '2020-08-25 12:47:57'

'5', 'TEA', '3', '2020-08-25 12:49:19'

'6', 'TEA', '3', '2020-08-25 14:56:55'

All the orders of the CoffeeShop are being inserted to this “coffee” table during the day. To dont disturb Coffee Shop Sale application routine, I establisghed a separate pipeline.

I used Kafka Connect to get all new inserted records to the “coffee” table. When a new record inserted, Kafka Connect detects this line, take it and do necessary action which is putting this line data to the Apache Kafka.

Apache Kafka is a message queue to where you produce messages or from where you consume messages. In my BI system, I dont have all the records in the “coffee” table. A summary of the “coffee” table will be enough for the basic decision support queries.

At this point, I used Apache Spark Streaming, to consume messages from Kafka, aggregate data, and persist it to the plain text files.

Then, I used Apache Spark to read text files and produce data which is used by the BI application.

**Necessary Installations for Coffee Shop Data Pipeline**

1. Java 1.8

2. Mysql 8.0.19

3. Apache Kafka2.13-2.6.0

4. Confluent Kafka Connect jdbc 5.5.1

5. Apache Spark 2.4.5

**Commands in Apache Kafka**

**starting zookeeper**

bin/zookeeper-server-start.sh config/zookeeper.properties

**starting kafka server**

bin/kafka-server-start.sh config/server.properties

**starting kafka connect**

**mysql.properties(**this file is required for “Kafka Connect” to connect and retrieve data from mysql**)**

plugin.path=/home/ozhan/Downloads/kafka/confluentinc-kafka-connect-jdbc-5.5.1

name=jdbc-source-connector

connector.class=io.confluent.connect.jdbc.JdbcSourceConnector

tasks.max=1

connection.url=jdbc:mysql://127.0.0.1:3306/STD?user=root&password=xxxyyyzzz&serverTimezone=UTC

table.whitelist=coffee

mode=incrementing

incrementing.column.name=id

topic.prefix=test-mysql-jdbc-

This is the command for starting up Kafka Connect

bin/connect-standalone.sh config/connect-standalone.properties config/**mysql.properties**

//Sample data at this point

{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"},{"type":"string","optional":true,"field":"product"},{"type":"double","optional":true,"field":"price"},{"type":"int64","optional":true,"name":"org.apache.kafka.connect.data.Timestamp","version":1,"field":"ptime"}],"optional":false,"name":"coffee"},"payload":{"id":5,"product":"TEA","price":3.0,"ptime":1598359759000}}

{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"},{"type":"string","optional":true,"field":"product"},{"type":"double","optional":true,"field":"price"},{"type":"int64","optional":true,"name":"org.apache.kafka.connect.data.Timestamp","version":1,"field":"ptime"}],"optional":false,"name":"coffee"},"payload":{"id":6,"product":"TEA","price":3.0,"ptime":1598367415000}}

**Commands in Apache Spark**

**Reading data from apache kafka from apache spark**

val df = spark.readStream

.format("kafka")

.option("kafka.bootstrap.servers", "localhost:9092")

.option("subscribe", "test-mysql-jdbc-coffee")

.option("startingOffsets", "latest")

.load()

val schema = new StructType()

.add("id",IntegerType)

.add("product",StringType)

.add("price",DoubleType)

.add("ptime",LongType)

//this line is only used for reading schema

val coffes = spark.sqlContext.read.json("/home/ozhan/Downloads/kafka/data/data.json")

import org.apache.spark.sql.types.{StringType, LongType, StructField, StructType, DoubleType, IntegerType}

val kdf = df.selectExpr("CAST(value AS STRING) as json").

select(from\_json($"json",schema=coffes.schema).as("data")). select("data.payload"). select("payload.id","payload.price","payload.product","payload.ptime")

val coffeeWin = kdf.groupBy(window('Date, "1 hour"),product).

.agg(sum("price").as("hourly\_sum"),count(\*).as(hourly\_cnt))

val coffeeNewSUM = coffeeWin.selectExpr("window.start" as ptime,"product","hourly\_sum","hourly\_cnt")

coffeeNewSUM.writeStream.format("csv").outputMode("append").start("/home/ozhan/coffeesumdata/").awaitTermination()

//now, sample data at this state

'1598346000', 'COFFEE', 50.00, 10

'1598346000', 'TEA', 18.00, 6

'1598353200', 'PIE', 20.00, 2

**Reading data from local file system for creating BI queries**

val coffeeReadedDF = spark.read.format("csv").option("header", "true")

.option("inferSchema","true")

.load("/home/ozhan/coffeesumdata/")

coffeeReadedDF.createOrReplaceTempView("coffeeReadedDFTemp")

val resDF = spark.sql("select product,from\_unixtime(ptime,'yyyyMMdd') as mtime,sum(hourly\_sum),sum(hourly\_cnt) from coffeeReadedDFTemp group by product,from\_unixtime(ptime,'yyyyMMdd') order by mtime")

resDF.write.format("csv").option("sep",",").option("header","true").save("/home/ozhan/coffeeresdata/")

//Sample Data for this state

'COFFEE', ‘20200824’,40.00, 8

'TEA', ‘20200824’,21.00, 7

'PIE', ‘‘20200824’’,30.00, 3

'COFFEE', ‘20200825’,50.00, 10

'TEA', ‘20200825’,18.00, 6

'PIE', ‘‘20200825’’,20.00, 2

By using this data, you can easily see, daily count and total prices per product and can decide to do discount or raise in the prices. Also by using the data, you can easily create, charts and diagrams that support your decision making mechanisms.