Rebaca

White Paper

Big Data Governance using Kafka-Spark-Cassandra Framework

For more info visit www.rebaca.com

Objective:

Create a distributed computation framework using Spark and Cassandra to process raw streams and batch data published in Kafka topics for deriving actionable insights.

Features Supported:

- **Data ingestion:** The incoming data obtained from REST calls are placed in Kafka topics. The data is distributed into partitions to parallelize the handling of a topic across multiple brokers.
- **Data Processing & Storing**: Streaming data is consumed in Spark using Dstream, which is then filtered and aggregated through Spark RDDs. The RDDs are then computed on different nodes of the cluster, implementing the business logic. The resultant datasets are saved in the Cassandra DB using bulk insert.
- **Data Analysis using ML:** The inbuilt machine learning libraries in Spark ML Lib are used for Advanced Analytics.
- **Data Visualization:** Representation of Analytics in web browsers using D3.js.

Key Advantages of the framework:

Cassandra and Spark work in a distributed fashion, along with the storage nodes. Spark performs operations in parallel, taking advantage of the available cluster's processing power. Spark computes, analyses and transforms this data.

With the advent of increasing data streams, most organizations deal with the pain of sorting the data to extract meaningful analyses. Spark enables real-time operations in memory, instead of performing repeated batch jobs. This facilitates the framework to exponentially process the continuous flow of data, which is crucial to any big data application.

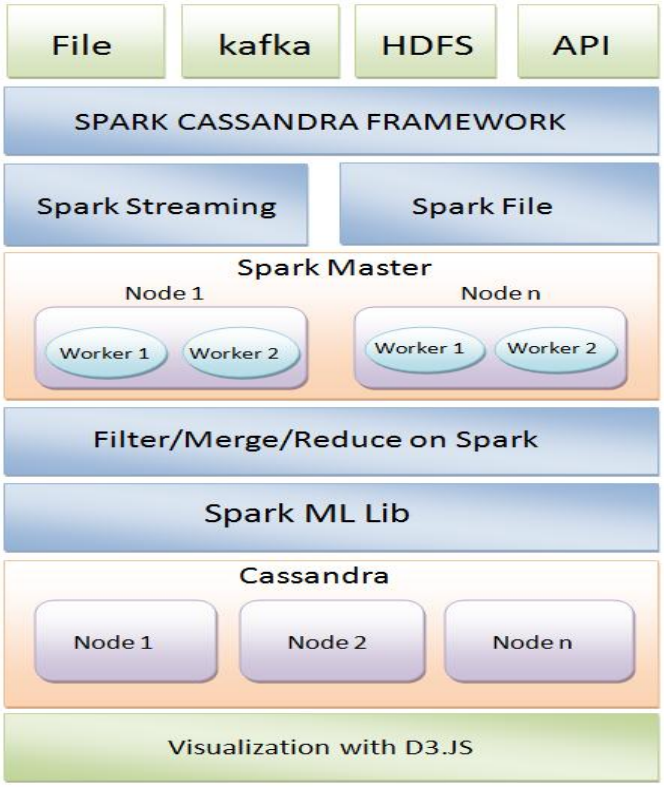
Tools & Technologies used:

***Data Ingestion**: Java, Kafka 0.8*

***Data Storing & Processing:** Cassandra 3.0.6, Spark 2.0.1*

***Data Analysis**: Spark Machine learning library – MLlib*

***Data Visualization**: D3.js*



Case Study: Advanced Analytics on Big Data using a Kafka – Spark – Cassandra framework

Crux of the Project:

The objective is to create Advanced Analytics on Big Data using a ***Kafka – Spark – Cassandra*** framework, prepared for processing stream & batch data derived from disparate sources.

Here we have used CDR (call detail records) data from a telecom operator as a sample.

Input Data Description:

A call detail record (CDR) documents the transactions passing through telecommunication gateways or devices. The record documents various attributes of the call, such as start time, duration, completion status, source number, destination number etc.

Few sample parameters found in a CDR are – *logTimestamp, acctStatusType, accountNumber, username, framedIp, nasIp, nasId, sessionId, terminationCause, and sessionTime*

Framework Implementation:

Input CDR data received through REST APIs are published in Kafka topics. The framework distributes this data into partitions so as to parallelize a topic across multiple brokers. The framework contains default implementations for File Receiver, Kafka Receiver, and other customized receivers that are implemented using simple framework interfaces. These receivers encapsulate the connection to source and retrieve data using Spark APIs. The APIs include Spark Reduce and map, group & aggregate over Spark RDD. Finally, the resultant dataset is saved in Cassandra.

This static data is also used as input training data in Spark ML Lib for statistical analysis using its inbuilt machine learning libraries.

Graphical representation of the session timings and output octets

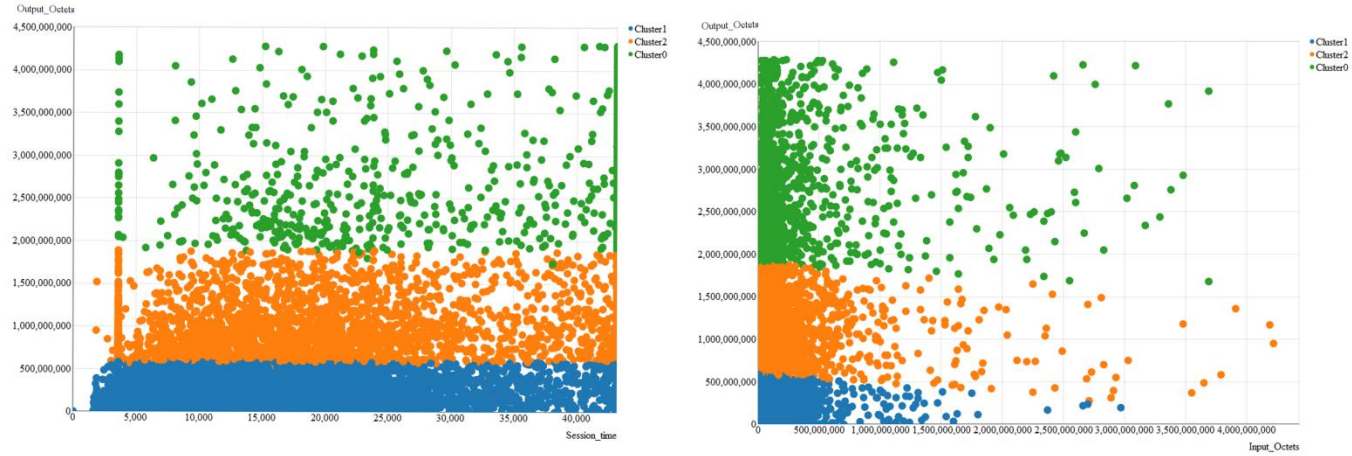
Sample Business Insights using D3.js:

The derived Analytics is generated in the web browser using D3.js. Some sample analytics are –

- *Total data Usage /Individual User*
- *Total data usage/day/destination*
- *Highest Session Termination Counts in a destination*
- *Top 10 data usage destination*
- *Active/Inactive users in the selected destination*
- *Call Volume Line Chart*
- *Call volume/Revenue/Elapsed Time Based on Rate Plan*

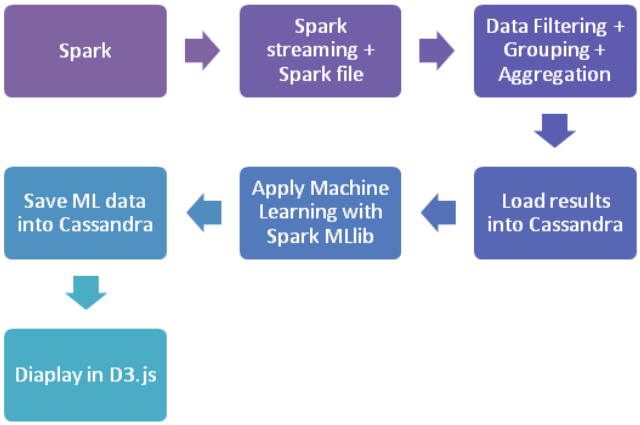
Machine learning using spark MLlib for cluster analysis by K-means:

The InputOctets (downloading data), OutputOctets (Uploading data) and session time are used from the input CDR data as cluster parameters. K-means clustering partitions n observations into k clusters where, each observation belongs to the cluster with the nearest mean. In the K-means technique, the k centroid (k being the number of clusters) is randomly chosen from 3 dimensional spaces or parameters and is calculated using the Euclidean distance measure. Next the model chooses new centroids for each cluster and groups the closest points near these centroids to form new clusters replacing the old ones. This process is repeated until the new centroids remain unchanged creating 3 definitive clusters based on user data consumption pattern and their session timings.



Kmeans clustering over CDR data
(based on User's Uploading/Downloading usage with session time)

Technology Flow:



Other Use Cases:

1. Telecom
2. Security
3. IT Infrastructure Services
4. E-Commerce
5. BFSI
6. Game Industry
7. Weather Forecasting
8. Healthcare Industry
9. Sentiment analysis of twitter feeds