

Lambda Architecture for Twitter real-time image processing

Alessio Venturi

Università degli Studi di Firenze
Dipartimento di Ingegneria dell'Informazione

Firenze, 22 Aprile 2021



1 Introduction

2 Proposed approach

- Image Processing with Lire
- Serving layer
- Batch layer
- Speed layer

3 Conclusions

- Big Data differs from traditional data processing through its use of parallelism, only by bringing multiple computing resources together we can process terabytes of data
- We need to find ways to analyze a large amount of data
- Lambda Architecture is a particular approach composed by:
 - ◆ *batch layer* : applies batch-oriented technologies (like MapReduce) on a master database. It is effective but it has a high latency
 - ◆ *serving layer* : specialized distributed database that supports batch updates and random reads
 - ◆ *speed layer* : only looks at recent data and uses low-latency techniques to update real-time views. It compensate for the high latency of the batch layer
- Image Processing applied to Big Data can produce some useful applications like to create clusters of a large amount of images and return the most representative ones.

- The main goal was not a perfect image processing analysis but the implementation of the architecture
- Speed layer is started first with the keyword as arguments. It creates the speed layer tables at the start of the execution
- Only tweets that have images in their content and the hashtag formed by the keyword are selected

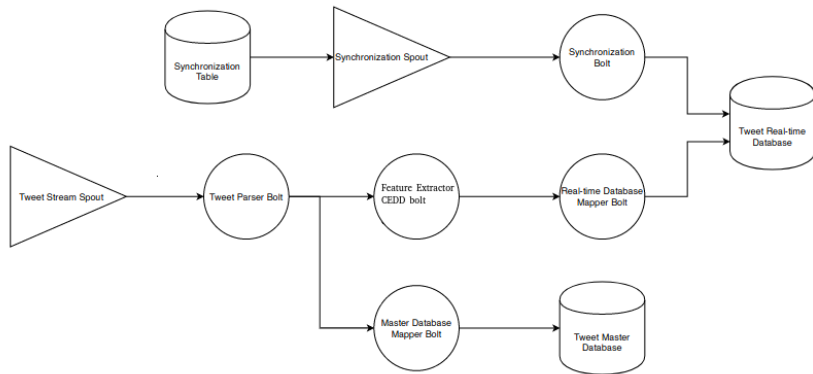
- Developed with *LIRE* (*Lucene Image Retrieval*) library
- We will use CEDD (Color and Edge Directivity Descriptor) to extract the 144-dimensional features vector from each image
- These vectors will be processed inside the K-Means algorithm implemented on MapReduce

Based on Apache HBase and composed by 4 tables:

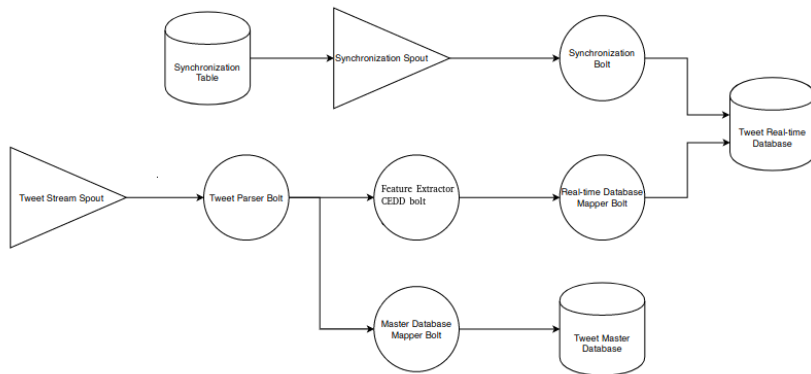
- **tweet master database:** master database of the Lambda Architecture ($\langle TweetID, Keyword, Path_img \rangle$)
- **tweet real-time database:** stores the tweets on which the real-time view is based ($\langle Keyword, Path_img, FeatureVector \rangle$)
- **batch view:** result of the batch processing ($\langle FeatureVector, NearestPath_img \rangle$)
- **synchronization table:** contains the start and the end timestamps of the batch processing ($\langle StartTimeStamp, EndTimeStamp \rangle$)

- Represented by Apache Hadoop
- Implements the K-Means algorithm inside MapReduce job on *tweet master database*
- Writes its results from scratch in *batch view*
- Writes the start and the end timestamps of the computation in *synchronization table*
- Two classes are created to simplify the K-Means algorithm.
 - ◆ Point : each tweet will be a Point (indexC, path_img, listOfFeatures)
 - ◆ Center (extends Point) : it has an index and the numberOfPoint associated with it.

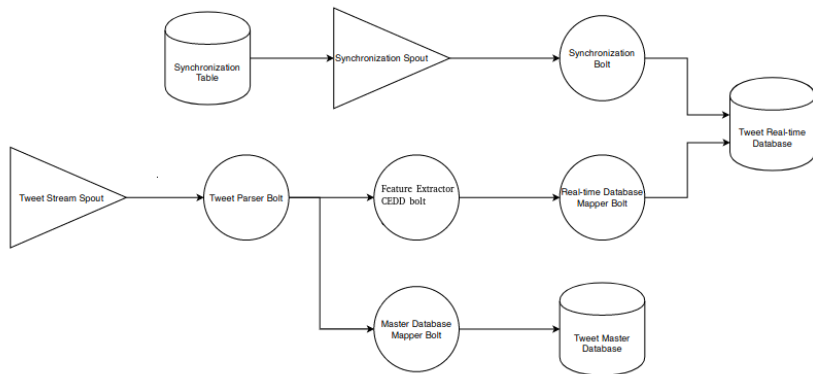
- Driver coordinate the job, set the configuration, create the initial center, start the K-Means loop in MapReduce and update the *batch view*
- Mapper takes a tweet in input and outputs a $\langle Center, Point \rangle$ tuple
- Combiner takes a $\langle Center, Point \rangle$ tuple as input and adds up all the vector features of the points associated with a center and returns a $\langle Center, Point \rangle$ tuple
- Reducer takes a tuple in input, do the average dividing the sum (computed in combiner phase) with the number of points associated with the center. The output will be a $\langle IntWritable, Center \rangle$ tuple for each center required



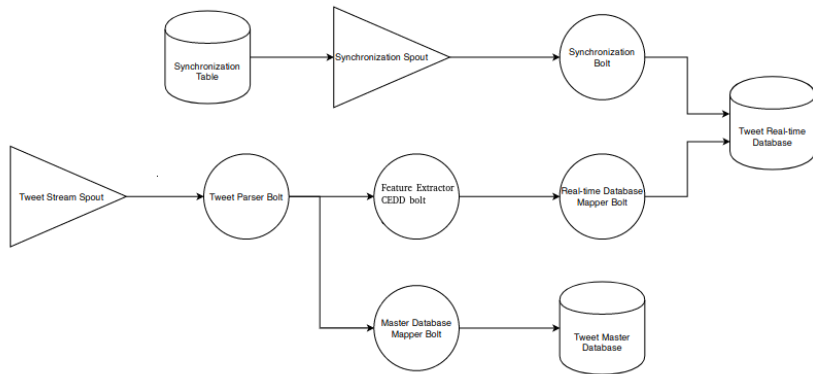
- **tweet stream spout**: gets a real-time stream of tweets with *Twitter4j* library and filters them
- **tweet parser bolt**: parse a tweet object to a tuple



- **master database mapper bolt**: inserts tweets in *tweet master database*



- **feature extractor CEDD bolt**: extract a 144-dimensional vector from each image in input
- **real-time database mapper bolt**: inserts tuples in *tweet real-time database*



- **synchronization spout:** checks when batch processing ends
- **synchronization bolt:** deletes already processed tweets from *tweet real-time database*

- It has been shown an implementation of a Lambda Architecture capable of getting the most representative images of real-time tweets using several techniques together.
- The final result of K-Means can be seen in the "Hbase Shell" by looking the "tweet_batch_view" table
- As a future development could be added :
 - ◆ A GUI to visualize easily the result
 - ◆ A filter to select only the images that truly reflect the keyword
 - ◆ A different approach of K-Means