Usage of Apache Spark and Graphx

VLBAII – System Architectures

Magdeburg Research and Competence Cluster
Workgroup Business Informatics I

Faculty of Computer Science
Department of Technical and Business Information Systems

Otto-von-Guericke-University Magdeburg

SUBASH PRAKASH
subash.prakash@st.ovgu.de

Agenda

- Introduction
- Apache Spark vs Map Reduce
- Project Template
- Apache Spark and Underlying eco system
- Spark Cluster Overview
- Operation of Spark
- Use Cases
- Relativity of the topic to lecture
- Current trends in the field
- Conclusion and Future Work

Introduction

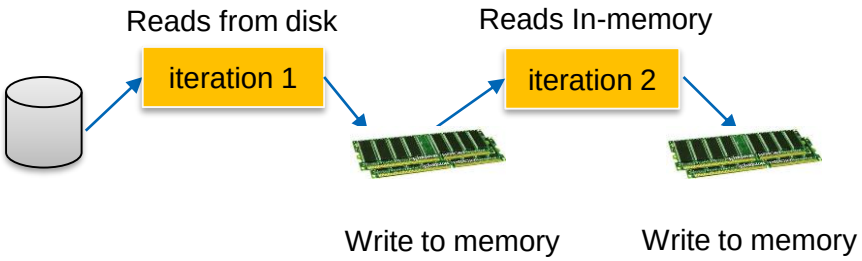
- Growth of data has been increasing rapidly from different sources:

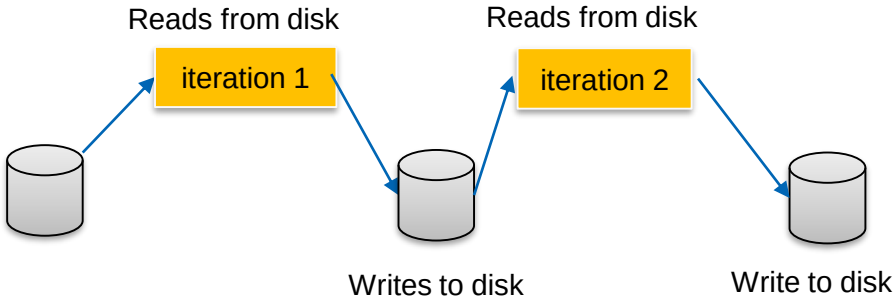
1. Internet/Online Data
2. Mobile Data
3. Financial Data
4. Graph Data from Social Networking sites
5. Sensor Data

It is very important to handle such big data coming very day every minute

Apache Spark vs MapReduce

VS





Cont…

- Below are key differences to know:

Map Reduce	Apache Spark
Disk to Disk Computing: 1. MapReduce inserts barriers, and it takes a long time to write things to disk and read them back. Hence MapReduce can be slow and laborious.	In Memory Computing: 1. Spark can keep things in memory without I/O operations, so one can keep operating on the same data quickly.
Batch Processing	Real Time Processing
Computation time is slow	Computation time is fast
Native to Java	Native to scala, java, python and R

Project Use Case:

Task

Task:

- Pr
  an
- Pe
- Pl
- Ma
- Us

System Requirement

- Setup of Mas
- Setup of Wo
- Setup of Spa
- More informa

Goal

The idea to understand how spark and graphx can be used on large dataset and performing it's operations.

Use Case:

- ***Analysis of Massive Data Set (IMDB) and processing it.***
- ***Spark Streaming an Introduction usage using twitter***

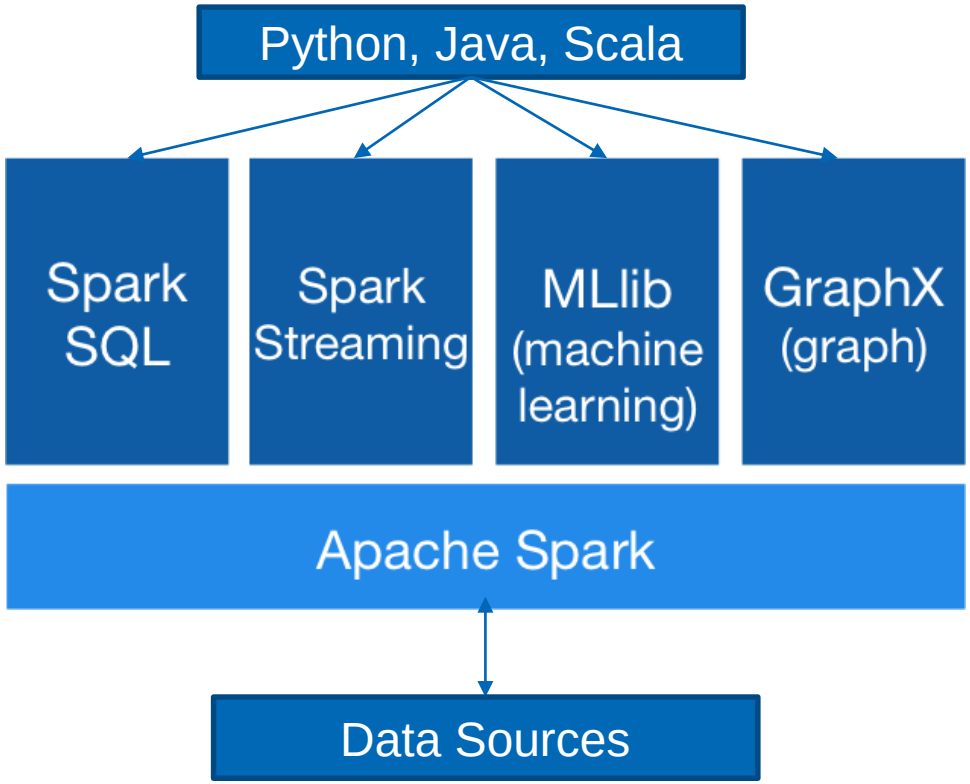
Apache Spark

source: http://www.toptal.com

- Spark is an Apache project advertised as “lightning fast cluster computing”.
- It has a thriving open-source community and is the most active Apache project at the moment.
- Spark provides a faster and more general data processing platform.
- Spark lets you run programs up to 100x faster in memory, or 10x faster on disk, than Hadoop.

Apache Spark
unified analytics engine for large-scale data processing

Apache Spark supports data analysis, machine learning, graphs, streaming data, etc. It can read/write from a range of data types and allows development in multiple languages.



Spark Core

- Foundation for parallel and distributed processing of large datasets.
- Accountable for all the basic I/O functionalities, scheduling and monitoring the jobs on spark clusters, task dispatching, networking with different storage systems, fault recovery and efficient memory management.
- Makes use of a special data structure known as RDD (Resilient Distributed Datasets)

Spark SQL

- Acts as a library on top of Apache Spark
- SQL type query can be ran on rdd's
- DataFrame constitutes the main abstraction for Spark SQL. They are nothing but the schema RDDs.

Spark Streaming

- A light weight API that allows developers to perform batch processing and streaming of data with ease, in the same application.
- It can work with different live streams such as twitter, kafka.
- *DStream*, which represents a continuous stream of data. It can easily be integrated from sources as kafka.

Source: https://www.spark.apache org

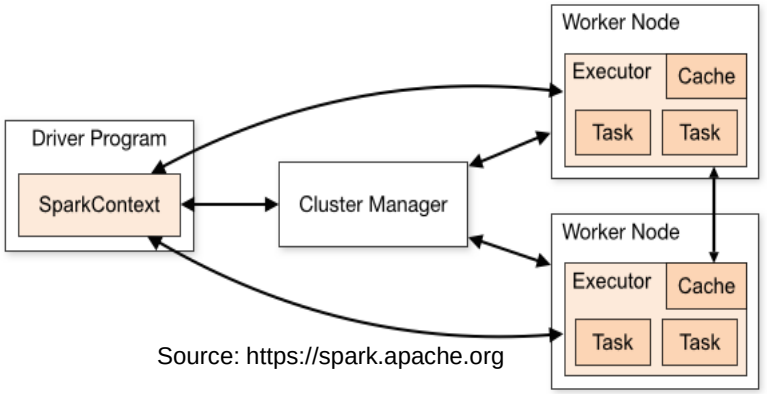
Spark MLib

- A low–level machine learning library that can be called from Scala, Python and Java programming languages
- Simple to implement and use.
- Algorithms for classification and clustering are available.

Spark Graphx

- GraphX is a graph computation engine built on top of Spark that enables users to interactively build, transform and reason about graph structured data at scale.
- It comes complete with a library of common algorithms.

Spark Cluster Overview



Source: https://spark.apache.org

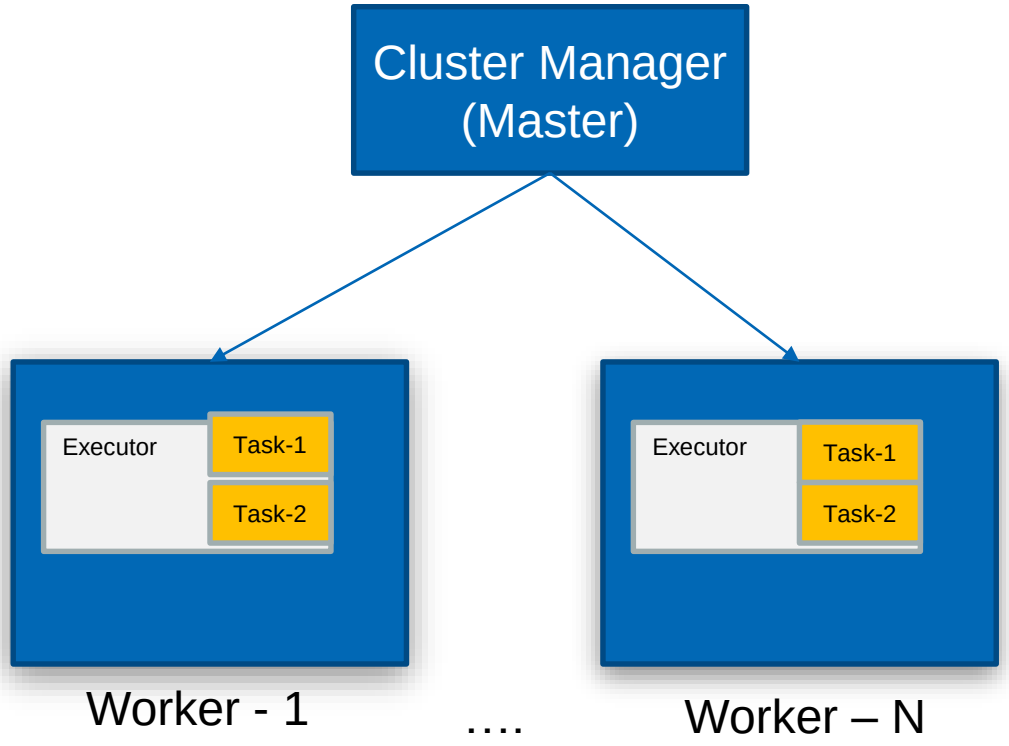
Components:

1. Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in your main program (called the driver program).
2. Specifically, to run on a cluster, the SparkContext can connect to several types of cluster managers (either Spark's own standalone cluster manager, Mesos or YARN), which allocate resources across applications. Once connected, Spark acquires executors on nodes in the cluster, which are processes that run computations and store data for your application. Next, it sends your application code (defined by JAR or Python files passed to SparkContext) to the executors. Finally, SparkContext sends tasks to the executors to run.

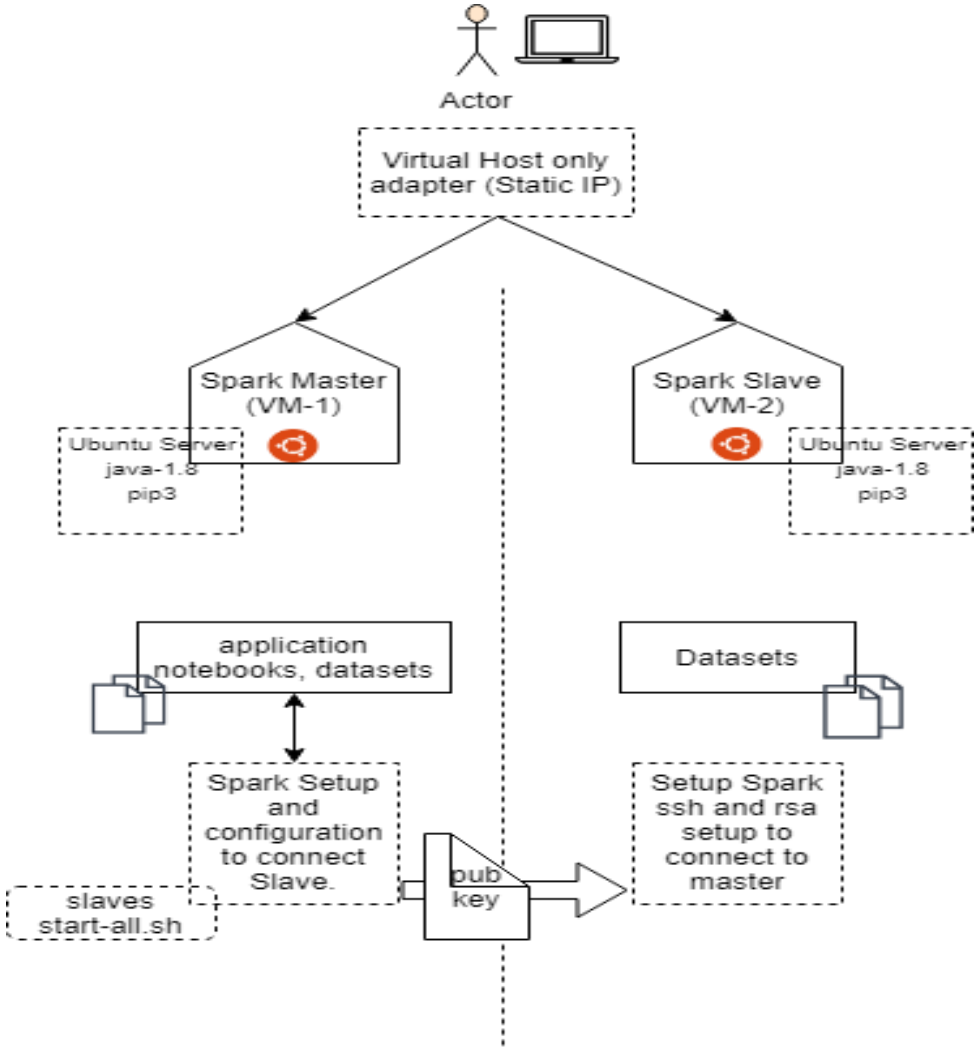
Cont…

- Monitoring of the master can be done on the web UI at http://<<master>>:8080
- Monitoring for the worker and jobs can be done with http://<<worker>>:4040

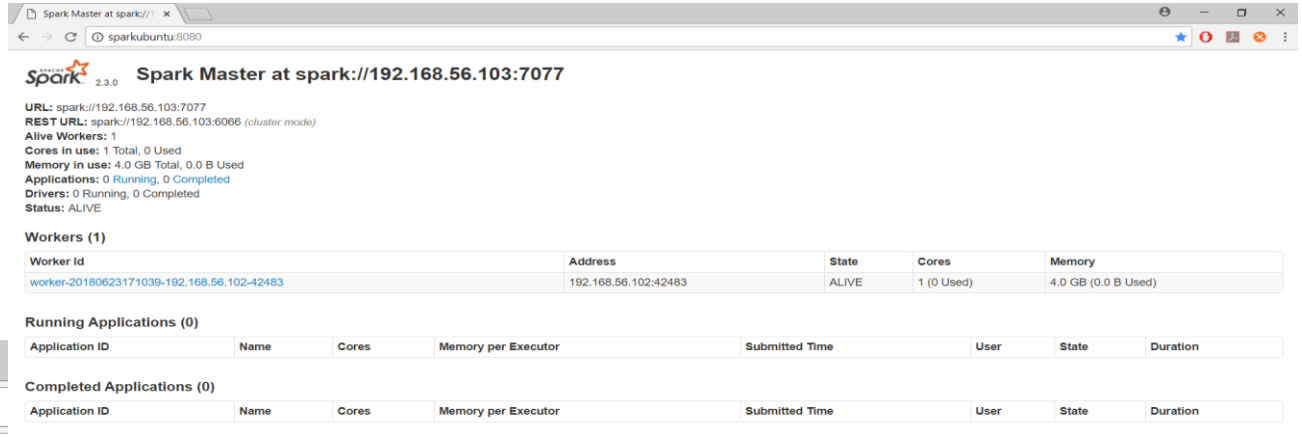
Architecture

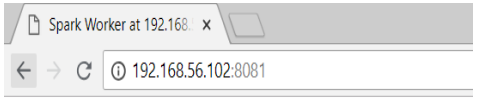


Server Level Diagram



Screenshot of the Spark Web UI





Spark 2.3.0

Spark Worker at 192.168.56.102:42483

ID: worker-20180623171039-192.168.56.102-42483
Master URL: spark://192.168.56.103:7077
Cores: 1 (0 Used)
Memory: 4.0 GB (0.0 B Used)

Back to Master

Running Executors (0)

ExecutorID	Cores	State	Memory	Job Details	Logs

Resilient Distributed Data Set (RDD)

- RDDs (Resilient Distributed Datasets) is *Data Containers*
- RDD is created by either parallelize() or reading a data file
- All the different processing components in Spark share the same abstraction called RDD
- Immutable in nature.
- As applications share the RDD abstraction, one can mix different kind of transformations to create new RDDs
- It is Fault tolerant

DataFrames

- A DataFrame is a Dataset organized into named columns.
- It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations under the hood.
- DataFrames can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing RDDs.
- The DataFrame API is available in Scala, Java, Python, and R.

RDDs vs. DataFrames

- RDDs provide a low level interface into Spark
- DataFrames have a schema
- DataFrames are cached and optimized by Spark
- DataFrames are built on top of the RDDs and the core Spark API

Spark Operations

- There are two spark operations:

- Transformation

TRANSFORMATION (Creates a new RDD)	map filter sample groupByKey reduceByKey sortByKey intersection	flatMap union join cogroup cross mapValues reduceByKey

Cont…

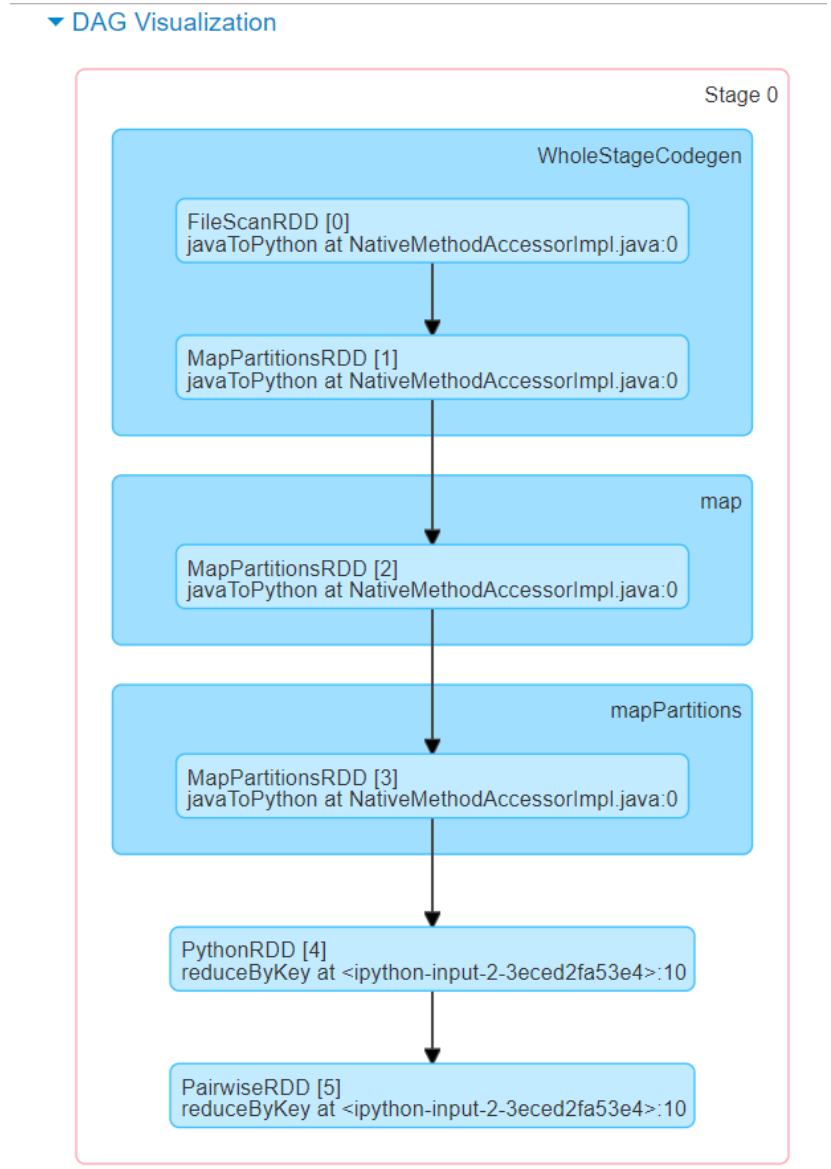
- Actions

Action (returns results to the program)	collect first Count takeSample take lookupKey Reduce takeOrdered countByKey save

Internal of spark

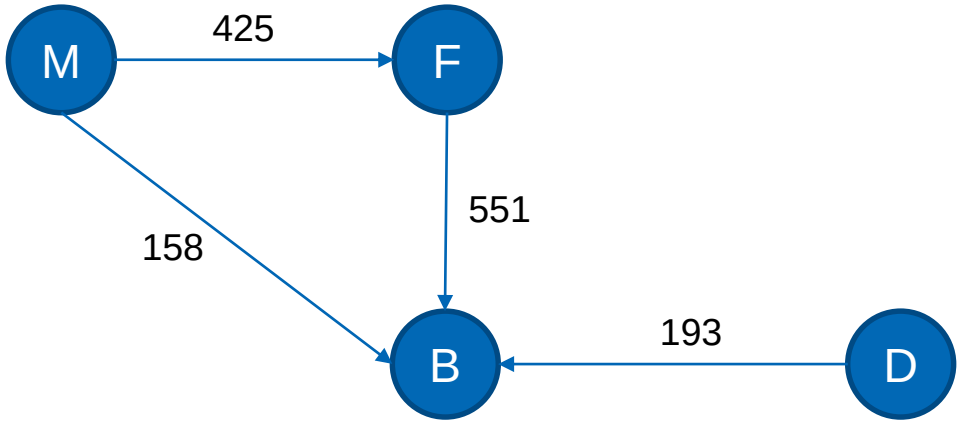
- A Visualization addition to the latest Spark release displays the execution DAG for each job.
- In Spark, a job is associated with a chain of RDD dependencies organized in a direct acyclic graph (DAG) that looks like the following:

Cont..


DAG Visualization
Stage 0
WholeStageCodegen
FileScanRDD [0]
javaToPython at NativeMethodAccessorImpl.java:0
MapPartitionsRDD [1]
javaToPython at NativeMethodAccessorImpl.java:0
map
MapPartitionsRDD [2]
javaToPython at NativeMethodAccessorImpl.java:0
mapPartitions
MapPartitionsRDD [3]
javaToPython at NativeMethodAccessorImpl.java:0
PythonRDD [4]
reduceByKey at <ipython-input-2-3eced2fa53e4>:10
PairwiseRDD [5]
reduceByKey at <ipython-input-2-3eced2fa53e4>:10

Apache Graphx

- Built upon this basic paradigm of graph theory.
- It extends the Spark RDD by introducing a new Graph abstraction.
- The property graph is a directed multigraph with user defined objects attached to each vertex and edge
- Example:



M – Magdeburg
F – Frankfurt
B – Berlin
D - Dresden

Graphx Operators

- Just like RDD, graphx also has same operators such as map , reduce,join,filter etc…
- There are certain graph related operators like in-degree,out-degree, noOfVertices, edges etc …
- It offers page rank functionality, which tells on the importance of each vertex.

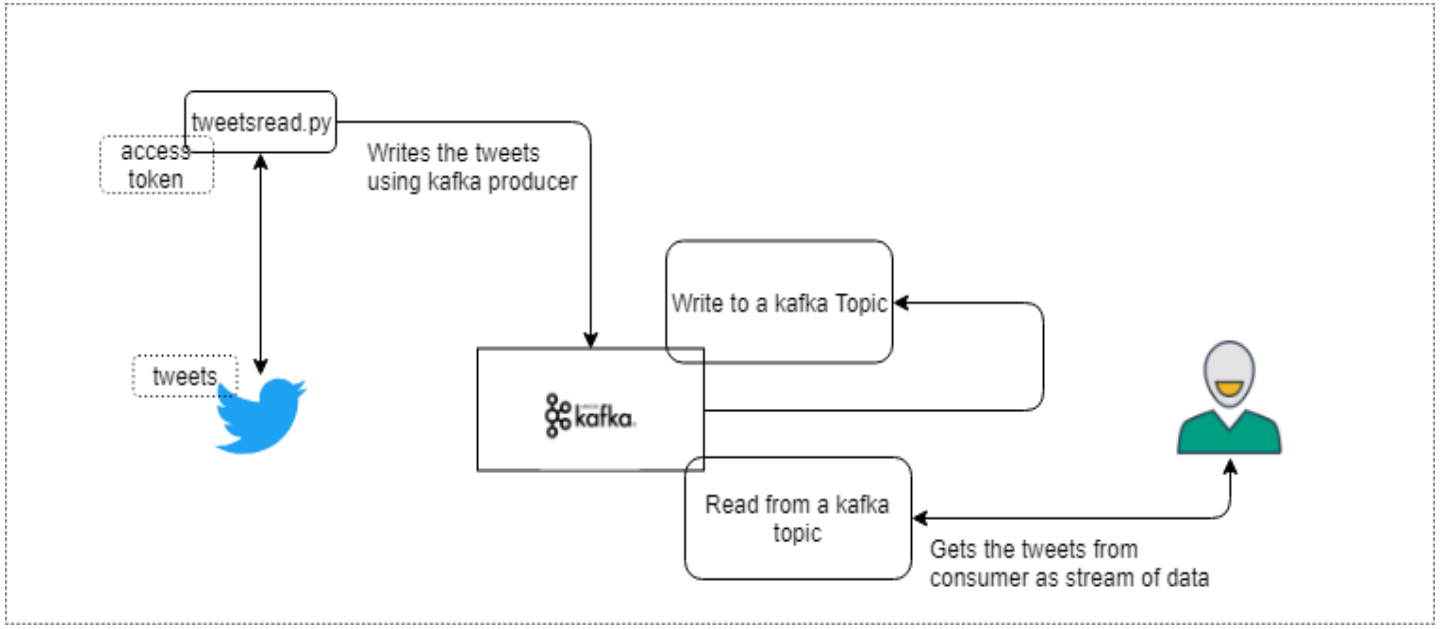
Example: If a twitter user is followed by another twitter user, then his rank will be high.

Use Cases: Analysis of Massive Dataset (Imdb) with spark and Graphx.

- Key points:
- For this purpose, the coding is performed in jupyter–notebook connected with pyspark.
- The dataset is taken from imdb : https://www.imdb.com/interfaces/
- The code is submitted from master node and worker node receives it and performs execution as part of “executor”
- As part of pre–processing everything is performed in the code.

Spark Streaming

- The idea of the streaming implementation is done as below diagram:



Relations with lecture

Some points which I could note are:

- As a system architect, a person should know the ways to creating the spark clusters and make connection between the master and worker.
- Server Architecture design is inspired from the components design from the lecture which gives a broad picture on the installable to be deployed/setup to start working on the clusters
- Understanding the automation perspective. Automation is an aspect which is related.

Current Trends in the field

- Project Tungsten:
  - Optimize memory and cpu usage via binary storage format and runtime codegen
- Ongoing work on making spark stable with working on single node.

Future of Spark

- The growing popularity of Python and Spark
- The advent of BigDL (A open source framework developed by intel to handle big data and deep learning)

- To conclude, everyday very large amounts of data is getting collected from sensors, mobile devices and internet. There is a need now to understand and process such big data. Hence, in the coming days apache spark will be very useful tool to handle Big Data.

THANK YOU

References

- www.wikipedia.org
- www.spark.apache.org
- https://docs.databricks.com/spark
- https://docs.oracle.com/en/cloud/paas/database-dbaas-cloud/csdbi/generate-ssh-key-pair.html#GUID-4285B8CF-A228-4B89-9552-FE6446B5A673
- http://jupyter.readthedocs.io
- http://docs.tweepy.org/en/v3.5.0/getting_started.html
- https://kafka.apache.org/quickstart