Big Data Analytics

By Metin Senturk

Spark Quick Rewind

- Spark enables applications in Hadoop clusters to run up to 100 times faster in memory and 10 times faster even when running on disk.
- Spark lets you quickly write applications in Java, Scala, or Python.
- It comes with a built-in set of over 80 high-level operators.
- In addition to Map and Reduce operations, it supports SQL queries, streaming data, machine learning and graph data processing.

Spark Quick Rewind

- Hadoop MapReduce is a great solution for one-pass computations, but not very efficient for use cases that require multi-pass computations and algorithms.
- Spark allows programmers to develop complex, multi-step data pipelines using directed acyclic graph (DAG) pattern.
  - It also supports in-memory data sharing across DAGs, so that different jobs can work with the same data.
- Spark runs on top of existing Hadoop Distributed File System (HDFS) infrastructure to provide enhanced and additional functionality.
- We should look at Spark as an alternative to Hadoop MapReduce rather than a replacement to Hadoop.

Spark Quick Rewind

- Spark holds intermediate results in memory rather than writing them to disk which is very useful especially when you need to work on the same dataset multiple times.
- It’s designed to be an execution engine that works both in-memory and on-disk.
    - It can store part of a data set in memory and the remaining data on the disk.

Resilient Distributed Dataset (RDD)

- RDDs do not have a **schema**, which means that they do not have a columnar structure. Records are just recorded row-by-row, and are displayed similar to a list.
- Spark stores data in RDD on different partitions
- RDDs are rearranging the computations and optimizing the data processing.
- They are fault tolerance because an RDD know how to recreate and recompute the datasets.
- RDDs are immutable.
  - You can modify an RDD with a transformation but the transformation returns you a new RDD whereas the original RDD remains the same.
- RDD supports two types of operations:
  - Transformation
  - Action

Resilient Distributed Dataset (RDD)

Transformations

- Transformations don't return a single value, they return a new RDD.
- Nothing gets evaluated when you call a Transformation function, it just takes an RDD and return a new RDD.
- Some of the Transformation functions:
    - map, filter, flatMap, groupByKey, reduceByKey, aggregateByKey, pipe, and coalesce.

Actions

- Action operation evaluates and returns a new value.
- When an Action function is called on a RDD object, all the data processing queries are computed at that time and the result value is returned.
- Some of the Action operations:
    - reduce, collect, count, first, take, countByKey, and foreach.

Spark DataFrame

- Have all the feature of RDDs, and **schema**.
- They are not **pandas DataFrames.**
- Conceptually same with a table in a relational database.
- DataFrames can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing RDDs.
- DataFrame in Apache Spark has the ability to handle petabytes of data.
- Operations on Spark DataFrame runs on parallel, but not in Pandas

Spark DataSets

- Similar to DataFrame, but **strongly typed**.
  - the type is specified upon the creation of the DataSet and is not inferred from the type of records stored in it.
- Because of strong-typed, available only in Scala and Java, not in Python. (Python is **dynamically typed**)

Shared Variables

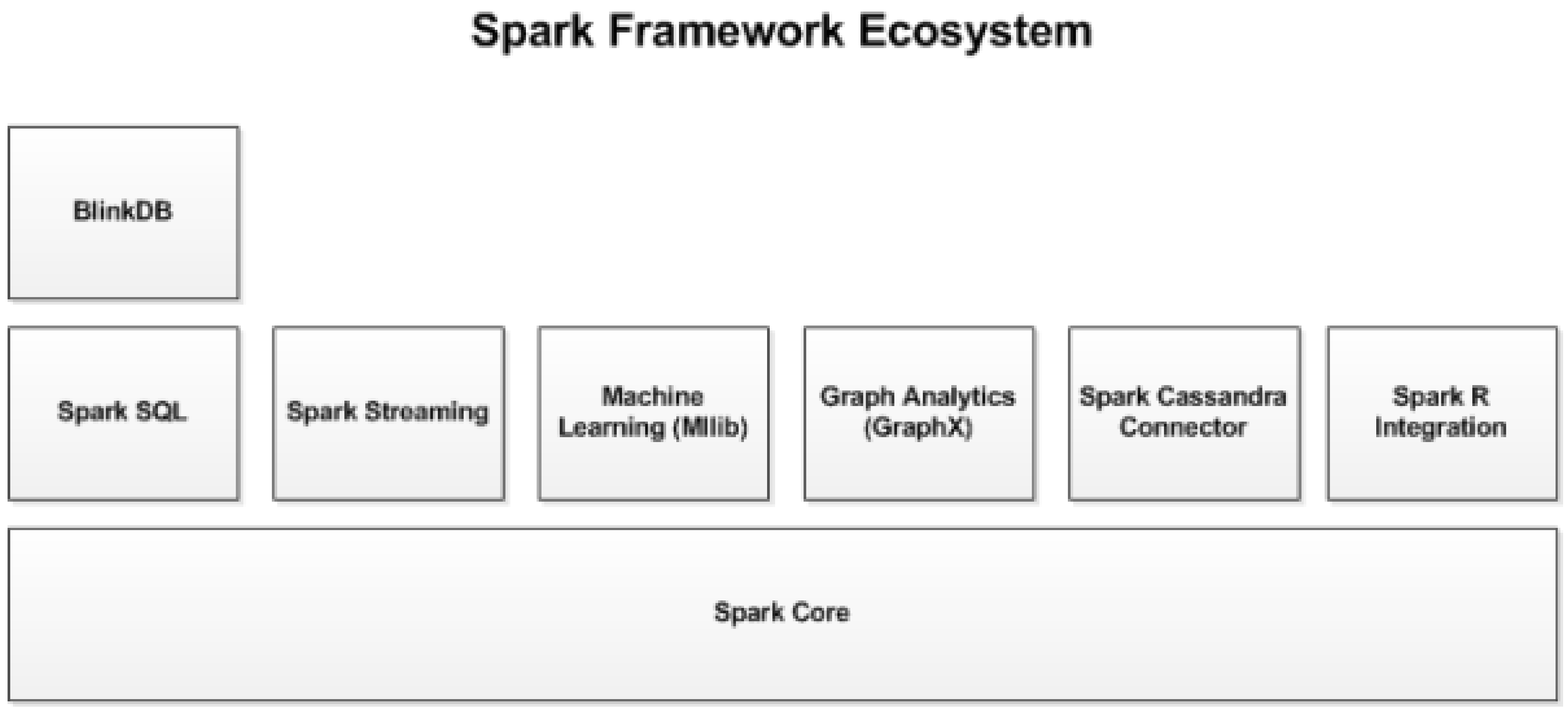
Broadcast Variables

- Broadcast variables allow to keep read-only variable cached on each machine instead of sending a copy of it with tasks.
- They can be used to give the nodes in the cluster copies of **large input datasets more efficiently**.

Accumulators

- Accumulators are only added using an associative operation and can therefore be efficiently supported in parallel.
- They can be used to implement counters (as in MapReduce) or sums. Tasks running on the cluster can add to an accumulator variable using the add method. However, they cannot read its value. **Only the driver program can read the accumulator's value**.

Spark Framework Ecosystem

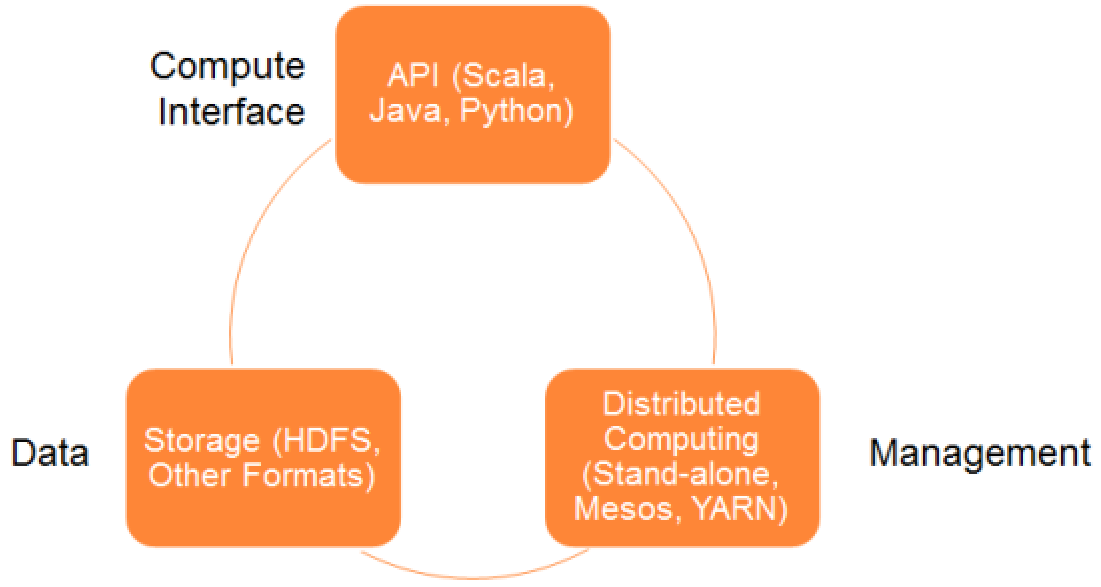


Spark Frameworks

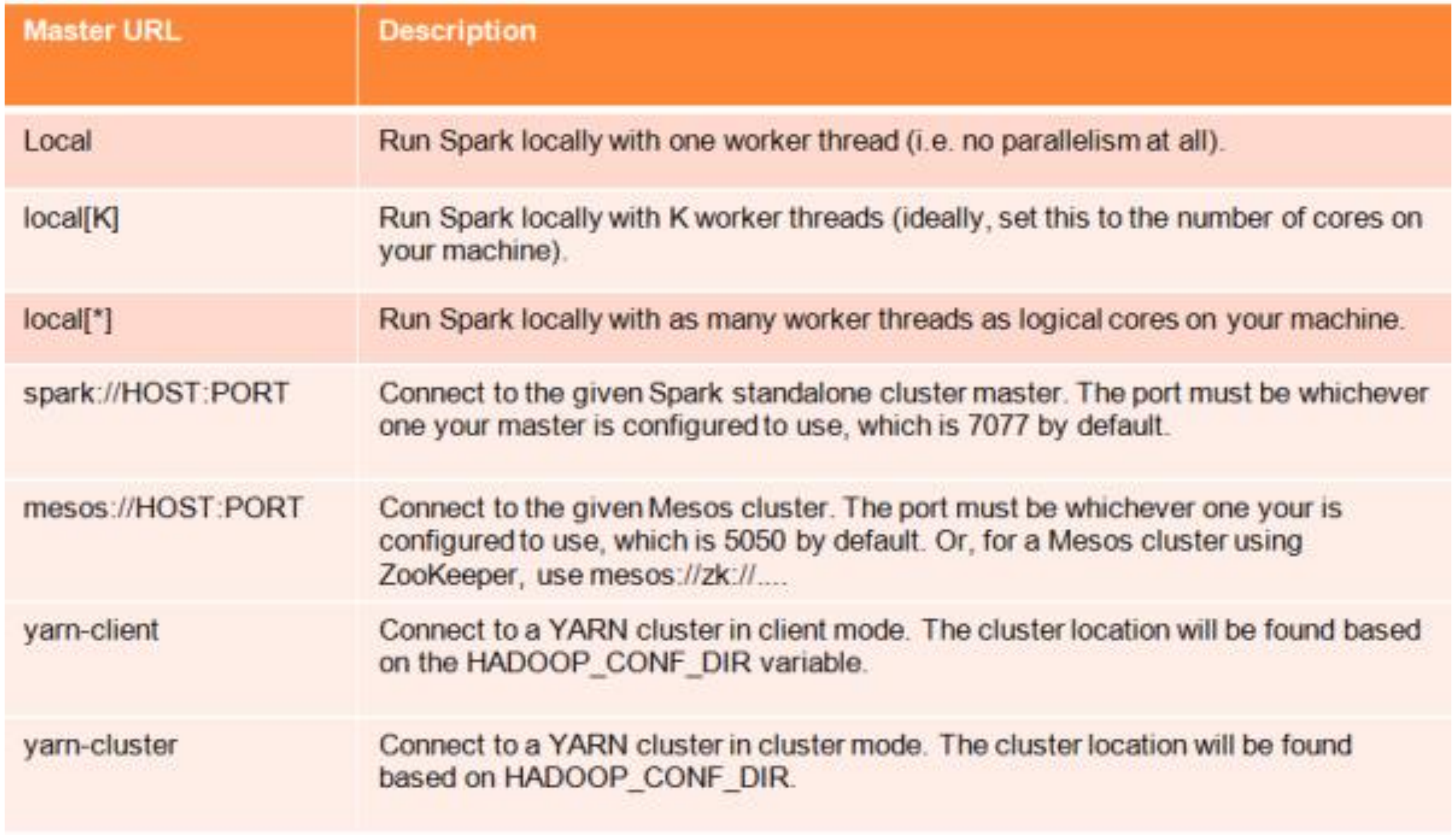
Framworks	Description
Spark Streaming	It can be used for processing the real-time streaming data. This is based on micro batch style of computing and processing.
Spark SQL	Provides the capability to expose the Spark datasets over JDBC API and allow running the SQL like queries on Spark data using traditional BI and visualization tools. Spark SQL allows the users to ETL their data from different formats it’s currently in (like JSON, Parquet, a Database), transform it, and expose it for ad-hoc querying.
Spark Mlib	Spark’s scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives.
Spark GraphX	GraphX extends the Spark RDD by introducing the Resilient Distributed Property Graph: a directed multi-graph with properties attached to each vertex and edge. To support graph computation, GraphX exposes a set of fundamental operators (e.g., subgraph, joinVertices, and aggregateMessages)

How Spark Works

1. Data Storage => HDFS, Hbase, etc
2. API => Python, Java, Scala, etc
3. Resource Management => Standalone, Mesos, Yarn, etc


Compute Interface
API (Scala, Java, Python)
Data
Storage (HDFS, Other Formats)
Distributed Computing (Stand-alone, Mesos, YARN)
Management

Initializing Spark



Master URL	Description
Local	Run Spark locally with one worker thread (i.e. no parallelism at all).
local[K]	Run Spark locally with K worker threads (ideally, set this to the number of cores on your machine).
local[*]	Run Spark locally with as many worker threads as logical cores on your machine.
spark://HOST:PORT	Connect to the given Spark standalone cluster master. The port must be whichever one your master is configured to use, which is 7077 by default.
mesos://HOST:PORT	Connect to the given Mesos cluster. The port must be whichever one your is configured to use, which is 5050 by default. Or, for a Mesos cluster using ZooKeeper, use mesos://zk://....
yarn-client	Connect to a YARN cluster in client mode. The cluster location will be found based on the HADOOP_CONF_DIR variable.
yarn-cluster	Connect to a YARN cluster in cluster mode. The cluster location will be found based on HADOOP_CONF_DIR.

SparkContext

- SparkContext is the entry gate for any spark derived application or functionality.
- It is the first and foremost thing that gets initiated when we run any Spark application.
- In PySpark, SparkContext is available as sc by default, so creating a new SparkContext will give an error.

SparkConf

- It provides configurations to run a Spark application. Instance of this class can be used to pass SparkContext.
- Following are some attributes of SparkConf:

• set(key, value)	− To set a configuration property.
• setMaster(value)	− To set the master URL.
• setAppName(value)	− To set an application name.
• get(key, defaultValue=None)	− To get a configuration value of a key.
• setSparkHome(value)	− To set Spark installation path on worker nodes.

StorageLevel

- StorageLevel decides whether RDD should be stored in the memory or should it be stored over the disk, or both.
- It decides whether to serialize RDD and whether to replicate RDD partitions.

PySpark API is Wrapper for Py4J

- Py4J is a package for any python program to communicate with Java Virtual Machine (JVM)
- Pyspark is functional based, for two reasons:
  - Spark main languages is on Scala, it is functional
  - Functional programming is easy to parallelize
- Pyspark does not need threading or multiprocessing, it handles by Spark/

sc.parallelize()

- Any given list, tuple, or data structure is parallelized by the Spark
- No need to do multiprocessing on top of it, as spark distributes it on its own.
- After that, transformations can apply (filter, groupByKey, etc)
- After that, Actions can apply.

Installing Spark with Docker

```
docker run -p 8888:8888 jupyter/pyspark-notebook
```

https://github.com/jupyter/docker-stacks/tree/master/pyspark-notebook

Where to Go?

- https://medium.com/@chris_bour/6-differences-between-pandas-and-spark-dataframes-1380cec394d2
- https://www.analyticsvidhya.com/blog/2016/10/spark-dataframe-and-operations/