Big Data Frameworks

Fereshteh Fakhar Firouzeh

Ph.D. Candidate at Carleton University
Data Scentist, Optimization and ML Researcher
fakhar.behnaz@gmail.com

Data

- "**Data**" can be defined as quantities, characters, or symbols on which operations are performed by computer. Data can be stored or transmitted in the form of electrical signals and recorded on magnetic, optical, or mechanical recording media. In summary, all the facts and figures that can be stored in digital format can be termed as data.
- The amount of data that is being generated in the digital age is growing at an exponential rate. According to the International Data Corporation (IDC), worldwide data generation will grow to 175 zettabytes by 2025 [2]. "**Big data**" is a term that describes a large volume of data that cannot be handled using traditional computing systems or saved on RAM.

Handling Big data

Some options for handeling Big data are:

- Using SQL database to move storage onto a hard drive instead of RAM.
- Using a distributed system to distribute the data to multiple machines/computers.

Local vs. Distributed System

"**Local System**" is a single computer, which shares same RAM and Hard Drive. "**Distributed System**" is a system whose data and calculations are located on different networked computers [5].

- In a local system, a user is limited to the computation resources of a single machine. But, in a distributed system, a user can leverage the power of different networked computers. Therefore, a distributed system provides more cores and capability than even a strong single machine.
- Distributed systems have the advantage of easily scaling. It can be easily scaled by adding more machines. But, there is a limit on how much RAM or how much storage can add to a single computer .
- Distributed systems include *fault tolerance* which is very important for large datasets (i.e. fault tolerance is a fundamental idea of replicating data across multiple machines so even if one goes down, calculations and data still persists and goes on). But, if a local machine crashes due to some error, all calculations and data can be lost.

Hadoop

- Distributed architectures use different ways to distribute a large datasets across multiple machines. **Hadoop** is a way that is used to distribute very large files across multiple machines [6].
- Hadoop uses *Hadoop Distributed File System(HDFS)* and *MapReduce*.
- **HDFS** allows a user to work with big data across a distributed system; hence, it is used to distribute large datasets. A significant point about HDFS is that it duplicate blocks of data for fault tolerance.
- **MapReduce** allows computations on the distributed data.

HDFS

- In HDFS, for the distributed storage, all nodes have their CPU and RAM. The main node (master node) controls the process of either distributing the storage or the calculations to the other nodes (slave nodes).
  - master node/ resource manager/ server monitors and manages workloads, maintains a multi-tenant environment, manages the high availability features, and implements security controls, namely Spark-Standalone, YARN, Apache, and Mesos.
- By default, HDFS uses blocks of data with a size of 128 megabytes.
- Each block is replicated and distributed in a way to support fault tolerance.
- Smaller blocks provide more paralyzation during processing.
- Multiple copies of a block prevent loss of data due to the failure of a node.

MapReduce

- MapReduce is a way of splitting computational tasks to a distributed set of files such as Hadoop distributed file system [3].
- MapReduce consists of a *Job Tracker* and multiple *Task Trackers*.
- The job tracker sends code to run on the task trackers. Then, the task trackers allocate the CPU in memory for the tasks and monitor the tasks on these worker nodes.

Spark

- Spark was created at the AMPLab at UC Berkeley and released in February 2013.
- Spark is an open source project on Apache. (Check Spark documnetion here `http://spark.apache.org/`).
- It is one of the latest technologies for handling Big Data, which has exploded in popularity due to its ease of use and speed [4].

Spark vs. MapReduce

- Spark may be up to 100 times faster. The reason is that Spark does the processing in the main memory of the worker nodes and prevents unnecessary I/O operation with the disks. However, MapReduce has to read from and write data to disk after each map and reduce operation. It can be considered as the key difference between Spark and Mapreduce.
- Hadoop MapReduce is able to work with far larger data sets than Spark.
- Spark is as a flexible alternative to MapReduce since it can create distributed datasets from different storage sources including Cassandra, AWS S3, HDFS, etc. But, MapReduce requires files to be stored in HDFS.
- For linear processing of huge data sets and economical solution, if no immediate results are expected, Hadoop MapReduce can be selected as a framework to use. Spark delivers fast performance, iterative processing, real-time analytics, graph processing, machine learning, etc.

Refer to [1] for more details.

Spark RDDs

- The idea of *Resilient Distributed Datasets* (RDD) is the core of Spark (Resilient means ability to be recomputed from history).
- (RDD) has the following main features:
  - **Immutable distributed collection of data**. RDD is immutable since it can not be modified.
  - **Parallel operation-partitioned**. Internally, Spark distributes the data in RDD. The data in RDD is partitioned. Then, each partition is fed to different nodes across clusters. Consequently, operations on the RDD are parallelly done.

Spark Context vs. Session

- Prior to Spark2.0, the entry point (an entry point is where control is transferred from the operating system to the provided program) of Apache Spark functionality was *Spark context*.
  - Spark context allows a Spark application to access Spark cluster with the help of resource manager.
  - Spark Context can be used to create RDDs (RDD is the main API). For every other APIs, different contexts such as SQL Context, Streaming Context, Hive Context should be defined.
- For Spark2.0 onwards, Spark Session provides a single point entry to interact with underlying Spark functionality.
  - Spark Session allows programming Spark with DataFrame and Dataset APIs.
  - All the functionality provided by Spark context are available in Spark session.
  - In order to use APIs of SQL, HIVE, and Streaming, no need to create separate contexts as Spark Session includes all the APIs.

References

[1] "spark vs. hadoop mapreduce: Which big data framework to choose". [Online]. Available: `https://www.scnsoft.com/blog/spark-vs-hadoop-mapreduce`.

[2] A. Patrizio. "idc: Expect 175 zettabytes of data worldwide by 2025". *Network World*, 2018.

[3] T. Ronald C. "an overview of the hadoop/mapreduce/hbase framework and its current applications in bioinformatics". *BMC bioinformatics*, 11(12):1–6, 2010.

[4] A. Shoro and T. Soomro. "big data analysis: Apache spark perspective". *Global Journal of Computer Science and Technology*, 2015.

[5] A. Tanenbaum and M. Van Steen. *"Distributed systems: principles and paradigms"*. Prentice-hall, 2007.

[6] T. White. *"Hadoop: The definitive guide"*. O'Reilly Media, Inc., 2012.