Installing Spark on Windows

1) Install Java
https://www.youtube.com/watch?v=7u5jJHSnGbw&t=0s&list=PLjJPQ

2) Install Spark (pre-built)
   a) download winutils.exe
   b) set HADOOP_HOME
   c) download Spark latest version from
      spark.apache.org
   d) extract
   e) Set SPARK_HOME and Set PATH
   f) Verify Spark Installed or not

RDD (Resilient Distributed Dataset)

- Immutable.
- Lazy transformations
- RDDs are created by starting with a file in the Hadoop file system (or any other Hadoop-supported file system), or an existing Scala collection in the driver program, and transforming it.
- Can persist RDD in memory but has to be requested for that.
- Automatically recover from node failures.

RDD Workflow

Source: Edureka
The term **Big Data** has created a lot of hype already in the business world. Hadoop and Spark are both Big Data frameworks – they provide some of the most popular tools used to carry out common Big Data-related tasks. In this blog, we will cover what is the difference between Apache Hadoop and Apache Spark MapReduce.

**Introduction**

**Spark** – It is an open source big data framework. It provides faster and more general purpose data processing engine. Spark is basically designed for fast computation. It also covers a wide range of workloads for example batch, interactive, iterative and streaming.

**Hadoop MapReduce** – It is also an open source framework for writing applications. It also processes structured and unstructured data that are stored in **HDFS**. Hadoop MapReduce is designed in a way to process a large volume of data on a cluster of commodity hardware. MapReduce can process data in batch mode.

**Data processing:**

**Hadoop**: Apache Hadoop provides batch processing. Hadoop developers a great deal in creating new algorithms and component stack to improve access to large scale batch processing.

MapReduce is Hadoop’s native batch processing engine. Several components or layers (like YARN, HDFS etc) in modern versions of Hadoop allow easy processing of batch data. Since MapReduce is about permanent storage, **it stores data on disk, which means it can handle large datasets**. MapReduce is scalable and has proved its efficacy to deal with tens of thousands of nodes. However, Hadoop’s data processing is slow as MapReduce operates in various sequential steps.

**Spark**: Apache Spark is a good fit for both batch processing and stream processing, meaning it’s a hybrid processing framework. **Spark speeds up batch processing via in-memory computation and processing optimization**. It’s a nice alternative for streaming workloads, interactive queries, and machine-based learning. Spark can also work with Hadoop and its modules. The real-time data processing capability makes Spark a top choice for big data analytics.
Resilient Distributed Dataset (RDD) allows Spark to transparently store data on the memory, and send to disk only what's important or needed. As a result, a lot of time that is spent on the disc read and write is saved.

**Real-time analysis**

**Spark**—It can process real-time data i.e. data coming from the real-time event streams at the rate of millions of events per second, e.g. Twitter data for instance or Facebook sharing/posting. Spark's strength is the ability to process live streams efficiently.

**Hadoop MapReduce**—MapReduce fails when it comes to real-time data processing as it was designed to perform batch processing on voluminous amounts of data.

**Ease of Use**

**Spark** is easier to use than Hadoop as it comes with user-friendly APIs for Scala (its native language), Java, Python, and Spark SQL. Since Spark provides a way to perform streaming, batch processing and machine learning in the same cluster, users find it easy to simplify their infrastructure for data processing. An interactive **REPL** (Read–eval–print loop) allows Spark users to get instant feedback for the commands.

**Hadoop**, on the other hand, is written in Java, difficult to program and requires abstractions. Although there is no interactive mode available with Hadoop MapReduce, tools like Pig and Hive make it easier for adopters to work with it.

**Graph Processing**

**Hadoop**: Most processing algorithms, like PageRank, perform multiple iterations over the same data. MapReduce reads data from the disk and after a particular iteration, it sends results to the HDFS and then again reads the data from the HDFS for next iteration. Such a process increases latency and makes graph processing slow.
**Graph Processing**

**Hadoop:** Most processing algorithms, like PageRank, perform multiple iterations over the same data. MapReduce reads data from the disk and after a particular iteration, it sends results to the HDFS and then again reads the data from the HDFS for next iteration. Such a process increases latency and makes graph processing slow.

In order to evaluate the score of a particular node, message passing needs to contain scores of neighboring nodes. And, these computations require messages from its neighbors, but MapReduce doesn’t have any mechanism for that. Although there are fast and scalable tools, like Pregel and GraphLab, for efficient graph processing algorithms, they are not suitable for complex multi-stage algorithms.

**Spark:** Spark comes with a graph computation library called GraphX to make things simple. In-memory computation coupled with in-built graph support allows the algorithm to perform much better than traditional MapReduce programs. Netty and Akka make it possible for Spark to distribute messages throughout the executors.

**Fault Tolerance**

**Hadoop:** Hadoop achieves fault tolerance through replication. MapReduce uses TaskTracker and JobTracker for fault tolerance. However, TaskTracker and JobTracker have been replaced in the second version of MapReduce by Node Manager and ResourceManager / ApplicationMaster, respectively.

**Spark:** Spark uses RDD and various data storage models for fault tolerance by minimizing network I/O. In the event of partition loss of an RDD, the RDD rebuilds that partition through the information it already has. So, Spark does not use the replication concept for fault tolerance.
Security

Hadoop MapReduce has better security features than Spark. Hadoop supports Kerberos authentication, which is a good security feature but difficult to manage. Hadoop MapReduce can also integrate with Hadoop security projects, like Knox Gateway and Sentry. Third-party vendors also allow organizations to use Active Directory Kerberos and LDAP for authentication. Hadoop’s Distributed File System is compatible with access control lists (ACLs) and a traditional file permissions model.

Spark’s security is currently in its infancy, offering only authentication support through shared secret (password authentication). However, organizations can run Spark on HDFS to take advantage of HDFS ACLs and file-level permissions.

Costs

Both Hadoop and Spark are open-source projects, therefore come for free. However, Spark uses large amounts of RAM to run everything in memory, and RAM is more expensive than hard disks. Hadoop is disk-bound, so saves the costs of buying expensive RAM, but requires more systems to distribute the disk I/O over multiple systems.

As far as costs are concerned, organizations need to look at their requirements. If it’s about processing large amounts of big data, Hadoop will be cheaper since hard disk space comes at a much lower rate than memory space.
Costs

Both Hadoop and Spark are open-source projects, therefore come for free. However, **Spark uses large amounts of RAM** to run everything in memory, and RAM is more expensive than hard disks. **Hadoop is disk-bound**, so saves the costs of buying expensive RAM, but requires more systems to distribute the disk I/O over multiple systems.

As far as costs are concerned, organizations need to look at their requirements. If it’s about processing large amounts of big data, Hadoop will be cheaper since hard disk space comes at a much lower rate than memory space.

Compatibility

Both Hadoop and Spark are compatible with each other. Spark can integrate with all the data sources and file formats that are supported by Hadoop. So, it's not wrong to say that Spark's compatibility with data types and data sources is similar to that of Hadoop MapReduce.

Both Hadoop and Spark are scalable. One may think of Spark as a better choice than Hadoop. However, MapReduce turns out to be a good choice for businesses that need huge datasets brought under control by commodity systems. Both frameworks are good in their own sense. Hadoop has its own file system that Spark lacks. And, Spark provides a way for real-time analytics that Hadoop does not possess.

Hence, the differences between Apache Spark vs. Hadoop MapReduce shows that **Apache Spark is much-advance cluster computing engine than MapReduce**. Spark can handle any type of requirements (batch, interactive, iterative, streaming, graph) while MapReduce limits to Batch processing.

So, this is the difference between Apache Hadoop and Apache Spark MapReduce.