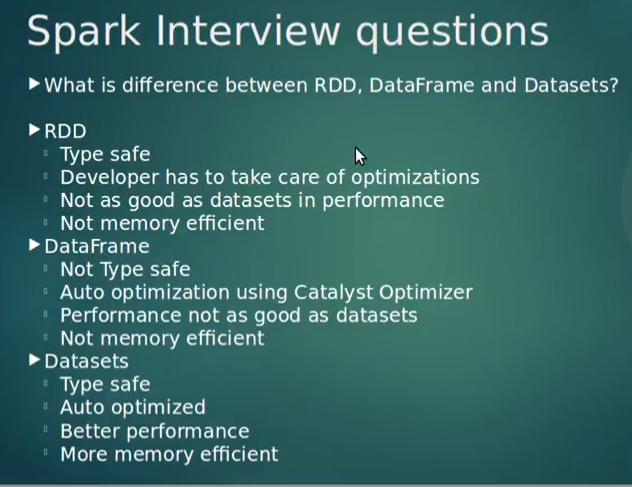


**Spark vs. MapReduce**

1. 1. Spark does not need a file server, whereas MapReduce can store files in a Hadoop distributed file system.
2. 2. Spark outperforms MapReduce by up to 100 times when it comes to running operations.
3. 3. Using MapReduce MapReduce writes the remaining data to disc for each Map (here, the input data is processed and stored in an HDFS after which the mapper method produces small chunks of data) and Reduce (here, the input data from the map stage is processed and produced a new set of output for storage in the HDFS) procedure. whereas The majority of the data is loaded into memory after each Spark shift.
4. If the memory in Spark runs out, it will overflow onto the disc.
5. MapReduce writes the majority of the data to disc after each map and reduces operation.
6. Spark retains the bulk of the data in memory after each transformation.



## SparkContext

### What is PySpark SparkContext?

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

### Parameters

SparkContext has some parameters that are listed below:

* **Master**: The URL of the cluster SparkContext connects to
* **AppName**: The name of your job
* **SparkHome**: A Spark installation directory
* **PyFiles**: The .zip or .py files send to the cluster and then added to PYTHONPATH
* **Environment**: Worker node environment variables
* **BatchSize**: The number of Python objects represented. However, to disable batching, set the value to 1; to automatically choose the batch size based on the object size, set it to 0; and to use an unlimited batch size, set it to −1
* **Serializer**: This parameter tells about an RDD serializer
* **Conf**: An object of L{SparkConf} to set all Spark properties
* **profiler\_cls**: A class of custom profilers used to do profiling; however, pyspark.profiler.BasicProfiler is the default one

The most widely used parameters among these are **Master** and **AppName**. The initial code lines for any PySpark application using the above parameters are as follows:

from pyspark import SparkContext

sc = SparkContext("local", "First App")

After getting done with the configuration settings and initiating a SparkContext object, which Spark does by default, check out the files in the application that you want to run on PySpark and understand how you can use a feature called SparkFiles, provided by Spark, to upload these files.

## SparkFiles and Class Methods

### What is a SparkFile?

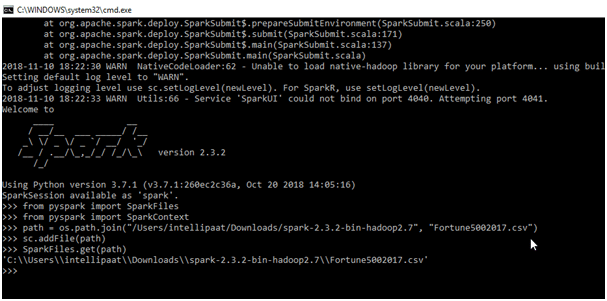
A SparkFile is what you will use when you need to upload your files in Apache Spark using SparkContext.addfile().

**Note:** Here, to perform class methods in SparkFiles, a file named ‘path’ is created, and the dataset is uploaded on it using os.path.join(“path”,”filename”).

### Class Methods: How to Use Them?

SparkFiles contain the following two types of class methods:

* **get(Filename)**: This class method is used when you need to specify the path of the file that you added using SparkContext.addfile() or sc.addFile()
  + **Input**:
* >>> from pyspark import SparkFiles
* >>> from pyspark import SparkContext
* >>> path = os.path.join("/Users/intellipaat/Downloads/spark-2.3.2-bin-hadoop2.7", "Fortune5002017.csv")
* >>> sc.addFile(path)
* >>> SparkFiles.get(path)
  + **Output**:



* **getRootDirectory()**: It is used to specify the path to the root directory where the file that you added using SparkContext.addFile() or sc.addFile(), exists.
  + **Input**:

>>> from pyspark import SparkFiles

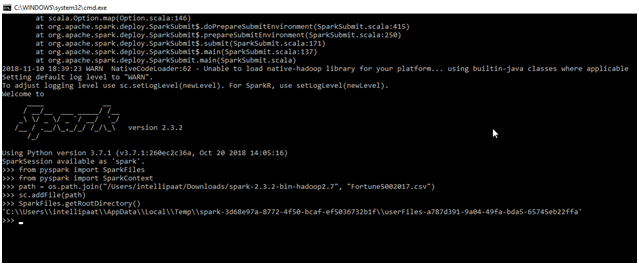
>>> from pyspark import SparkContext

>>> path = os.path.join("/Users/intellipaat/Downloads/spark-2.3.2-bin-hadoop2.7", "Fortune5002017.csv")

>>> sc.addFile(path)

>>>SparkFiles.getRootDirectory()

* + **Output**:



Now, you are acquainted with SparkFiles and have understood the basics of what you can do with them. It is time to understand the datasets in Spark. Let’s go on!