22.1 Lesson Summary - Introduction to Big Data

Big Data refers to a large amount of varied, potentially unreliable data often provided from a number of disparate sources at a fast rate. Performing data analytics with this kind of data requires solutions specifically tailored to big data.

Concept: Apache **Hadoop** is a collection of tools that allow for the distribution and processing of data across a network of (usually low cost) hardware. Hadoop allows the processing and storage of big data to be cheaper and more scalable than using traditional, large, high-cost servers.

* Suppl link: https://hadoop.apache.org/

Concept: **MapReduce** is a software and programming model for processing big data. MapReduce involves two steps, **mapping** and **reducing**. The mapping step breaks up the data and preprocesses it. Reducing aggregates the data after the mapping step. You can perform MapReduce operations in Python using the MRJob library. To use the MRJob library you should create a class that inherits from the MRJob class and specifies how data should be handled in *mapper* and *reducer* methods. To get the max value of a space separated list of integers using MapReduce you could use the following code:

from mrjob.job import MRJob

class Int\_Max(MRJob):

def mapper(self, \_, line):

for int\_str in line.split():

yield "int", int(int\_str)

def reducer(self, key, values):

yield key, max(values)

if \_\_name\_\_ == "\_\_main\_\_":

Int\_Max.run()

* Activity: 01-Ins\_MapReduce, 02-Evr\_Word\_Count, 03-Evr\_MrJob\_CSV

Concept: Apache **Spark** is a distributed processing system for large-scale data processing. Spark supports a number of programing languages and provides you with ways to organize and process your data that are more efficient and familiar than MapReduce. Spark can be run on a desktop but it requires properly configuring a virtual machine on your computer. An easier option is to run Spark in a Jupyter Notebook hosted on Google Colabratory. On Colabratory you can load Spark with the following code:

*import os*

*# Find the latest version of spark 3.0 from http://www-us.apache.org/dist/spark/ and enter as the spark version*

*# For example:*

*# spark\_version = 'spark-3.0.1'*

*spark\_version = 'spark-3.0.1'*

*os.environ['SPARK\_VERSION']=spark\_version*

*# Install Spark and Java*

*!apt-get update*

*!apt-get install openjdk-11-jdk-headless -qq > /dev/null*

*!wget -q http://www-us.apache.org/dist/spark/$SPARK\_VERSION/$SPARK\_VERSION-bin-hadoop2.7.tgz*

*!tar xf $SPARK\_VERSION-bin-hadoop2.7.tgz*

*!pip install -q findspark*

*# Set Environment Variables*

*os.environ["JAVA\_HOME"] = "/usr/lib/jvm/java-11-openjdk-amd64"*

*os.environ["SPARK\_HOME"] = f"/content/{spark\_version}-bin-hadoop2.7"*

*# Start a SparkSession*

*import findspark*

*findspark.init()*

*from pyspark.sql import SparkSession*

*spark = SparkSession.builder.appName("DataFrameBasics").getOrCreate()*

* Activity: 05-Ins\_Pyspark\_DataFrames\_Basics
* Suppl link: <https://spark.apache.org/>, <https://colab.research.google.com/notebooks/welcome.ipynb>

Concept: Spark supports its own versions of DataFrames that behave similar to Pandas DataFrames except they can contain data disturbed across a network of computers. You can create a Spark DataFrame using the following code:

*dataframe = spark.createDataFrame([*

*(0, "I am the first row of some words"),*

*(1, " I am the second row of some words"),*

*(2, " I am the third row of some words")*

*], ["id", "words"])*

*dataframe.show()*

You can import data into a Spark DataFrame using the SparkFiles object. For example:

*from pyspark import SparkFiles*

*url = "https://s3.amazonaws.com/sample.csv"*

*spark.sparkContext.addFile(url)*

*df = spark.read.csv(SparkFiles.get("sample.csv"), sep=",", header=True)*

*# Show DataFrame*

*df.show()*

Spark DataFrame functionality is very similar to Pandas DataFrame functionality. You can convert a Spark DataFrame into a Pandas DataFrame using the following code:

import pandas as pd

pandas\_df = spark\_dataframe.toPandas()

* Activity: 05-Ins\_Pyspark\_DataFrames\_Basics