PROGRAMMING WITH HDFS AND SPARK

CLASS-VII

Instructor: Palash Gupta

Re-Cap

- Introduction to Python
- Practical Session with Python
- Installing & utilizing New Python Package
- Introduction to UNIX and Shell
- Practical Session with Shell Scripting
- Running a python script in UNIX Cron Job with Shell Script Support

What we are going to Cover today?

- Accessing HDFS using Python
- Loading HDFS Files in Spark
- Running Basic Transformation and Action in Spark
- Understanding Spark Submit Details

Accessing HDFS using Python

- Standard Method
  - ❑ Install hdfs package
  - ❑ Defining .hdfscli.cfg in User home directory
  - ❑ Importing Config or Insecure Client in your python program
  - ❑ Accessing HDFS

- Another Method
  - ❑ Use os python package to run hadoop fs or hdfs dfs command
  - ❑ Use subprocess to get command output if require

- Important Note:
  - Python Package is going to python 3.6 rather 3.4
  - Python 3.6 is not supported with our Spark 2 Version
  - We did some workaround to continue our exercise

Loading HDFS Files in Spark using python

- Creating RDD from HDFS Files:

```
counts = sc.textFile("hdfs://bdrenfdludcf01:9000/names1")
counts = sc.textFile("hdfs://bdrenfdludcf01:9000/names1, hdfs://bdrenfdludcf01:9000/names2")
```

- Creating DataFrame from multiple HDFS Files or Local Files:

```
df = spark.read.format("csv").option("header", "true").load(" hdfs://bdrenfdludcf01:9000/names.csv")

df_fs = spark.read.format("csv").option("header", "true").load("../Downloads/*.csv")

df_hdfs = spark.read.format("csv").option("header", "true").load(" hdfs://bdrenfdludcf01:9000 /*.csv")
```

Running Basic Transformation and Action

SN	Key Item	Process
1	Dataframe Schema	>>>df.printSchema()
2	Column Renaming	>>>from pyspark.sql.functions import col >>>df_CM.select(col("BTS_ID").alias("BTS_ID_CM"), col("SEG_ID").alias("SEG_ID_CM")).show()
3	Dropping a column	>>>df_CM.drop("BTS_ID").show()
4	Adding new Column with a pre-defined value	from pyspark.sql.functions import lit >>>join_complete = join.withColumn('Brand_Name', lit('Uber'))
5	How to keep only distinct rows in a Dataframe	>>>df_epsbear_raw = df_epsbear_raw.distinct()
6	How to repartition or split a Dataframe like RDD partitioning	>>>df_traffic_tmp.repartition(5) >>>numbers = sc.parallelize([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15],7):
7	Perform Union Action	>>>df_epsbear_raw = df_epsbear_raw.unionAll(df_epsbear_tmp)

Understanding Spark Submit Details

- Interactive Method
  - pyspark
  - Will get both SparkContext and SparkSession entry point
- Local Mode
  - Locally it will initiate executor within driver program rather in cluster mode
  - Used mainly for development and testing

Understanding Spark Submit Details Contd.

- Standalone Cluster Mode
  - ❑ Deploy Client Mode
  - ❑ Deploy in Cluster Mode with or without supervision
  - ❑ Python doesn't support Cluster Mode
  - ❑ It uses internal Spark Cluster Manager

It also supports running using:

- Apache MESOS
- Kubernetes Cluster

- Running on Yarn
  - ❑ Use Yarn as a Cluster Manager
  - ❑ Use spark.yarn.executor.nodeLabelConfiguration to limit running nodes

QUESTION & ANSWER

THANKS FOR ATTENDING THE CLASS & YOUR CO-OPERATION

References

- https://spark.apache.org/docs/2.3.0/sql-programming-guide.html
- https://spark.apache.org/docs/latest/configuration.html
- https://spark.apache.org/
- https://spark.apache.org/docs/2.1.0/api/python/pyspark.html
- https://stackoverflow.com/questions/31610971/spark-repartition-vs-coalesce
- https://spark.apache.org/docs/latest/spark-standalone.html
- https://spark.apache.org/docs/latest/submitting-applications.html