IST769 Lab D

# Object Storage and Spark

### In this lab, we will explore the power of Apache Spark for ad-hoc data analytics over semi-structured data. We will analyze website log files where one row in the log represents a resource loaded from the web server. We will combine this data with geographic data based on IP Address so that we can determine the origins of the users visiting our website.

### Learning Outcomes

At the end of this lab you should be able to:

* Manage data in object storage like AWS S3, or Minio
* Use Apache Spark to perform extract, transform and load of semi-structured data
* Use Apache Spark SQL to perform a basic analysis of the data.t the end of this lab, you should be able to:

### Pre-Requisites

Before you begin:

* Open a terminal window in the lab environment
* Set the current working directory to **advanced-databases**
* Start the following services required by Spark and Minio: **jupyter minio**

### Tools Used In this Lab

The following tools will be used in this lab:

1. To access Jupyter Lab from your Windows host:  
   [http://localhost:8888](http://localhost:8888/)   
   The password is **SU2orange!**
2. To access the Minio client:
   1. Open a terminal from Jupyter
   2. Run this command at the terminal to install the Minio client  
      wget https://dl.min.io/client/mc/release/linux-amd64/mc && chmod +x mc && sudo mv -f mc /usr/local/bin
   3. Setup a Minio alias to [http://minio:9000](http://minio:9000/)   
      access key: minio  
      secret key: **SU2orange!**

# Lab Problem Set

In this section we will use spark to transform raw website log data into a useable format for data analysis. We will first load and clean the logs, then load the ip lookup geographic data, then combine the two into a single output. Finally, we will use spark transformation to answer some basic questions about the data.

1. Connect to the minio client. Create an alias to your minio server, named **ms**. Create a bucket **labd**. Inside the bucket, create folders **iplookup** and **logs**. Copy the 3 log files from **/datasets/clickstream** to the minio **logs** folder. Copy **iplookup.json** to the **iplookup** folder.

When you are finished you should have the following structure. Provide a list of commands necessary to complete this task. And include screenshots to show the files are there.

ms

|

|\_\_ lab

|

|\_\_ iplookup

| |

| |\_\_ iplookup.json

|

|\_\_logs

|

|\_\_ u\_ex160211.log, u\_ex160212.log, u\_ex160213.log

wget https://dl.min.io/client/mc/release/linux-amd64/mc && chmod +x mc && sudo mv -f mc /usr/local/bin

mc alias set minio http://minio:9000 minio SU2orange!

mc mb minio/labd

mc mb minio/labd/logs

mc mb minio/labd/iplookup

mc cp /home/jovyan/datasets/clickstream/iplookup.json minio/labd/iplookup

mc cp /home/jovyan/datasets/clickstream/u\_ex160211.log minio/labd/logs

mc cp /home/jovyan/datasets/clickstream/u\_ex160212.log minio/labd/logs

mc cp /home/jovyan/datasets/clickstream/u\_ex160213.log minio/labd/logs

1. Create a new Spark notebook called **labd.ipynb** write (or copy and edit ) Spark code to setup the Spark session. Make sure your spark session supports Minio access and include the **hadoop-aws** Spark Jar package. Provide a screenshot of your code and the output.   
   NOTE: You do not need Hive support.

import pyspark

from pyspark.sql import SparkSession

# MINIO CONFIGURATION

s3\_host = "minio"

s3\_url = f"http://{s3\_host}:9000"

s3\_key = "minio"

s3\_secret = "SU2orange!"

s3\_bucket = "labd"

# Spark init

spark = SparkSession.builder \

.master("local") \

.appName('jupyter-pyspark') \

.config("spark.jars.packages","org.apache.hadoop:hadoop-aws:3.1.2")\

.config("spark.hadoop.fs.s3a.endpoint", s3\_url) \

.config("spark.hadoop.fs.s3a.access.key", s3\_key) \

.config("spark.hadoop.fs.s3a.secret.key", s3\_secret) \

.config("spark.hadoop.fs.s3a.fast.upload", True) \

.config("spark.hadoop.fs.s3a.path.style.access", True) \

.config("spark.hadoop.fs.s3a.impl", "org.apache.hadoop.fs.s3a.S3AFileSystem") \

.getOrCreate()

sc = spark.sparkContext

sc.setLogLevel("ERROR")

1. Write Spark code to load logs from Minio **labd/logs** into a dataframe **logs1** using **spark.read.text**. print the schema and show 10 rows from the DataFrame. Screenshot the code and output.

logs\_in = f"s3a://{s3\_bucket}/logs/\*.log"

logs1 = spark.read.text(logs\_in)

logs1.show(10)

logs1.printSchema()

1. We need to remove the rows with **#** in front of them, as these are comments in the web server log files. Use the **filter()** function to do this, and save the results into dataframe **logs2**. Show the code and output in your screenshot.

logs2 = logs1.filter("value not like '#%'")

logs2.show()

1. Write back your **logs2** to Minio. Use the **text** format and call the file **logs-no-header.** Include a screenshot of the code and output.

logs\_out = f"s3a://{s3\_bucket}/logs\_no\_header"

logs2.write.text(logs\_out)

1. Read in the **logs-no-header** this time using **csv** to delimit on a space into **logs3**. Add headers (date,time, serverip, method, uri, querystirng, port, username, clientip, useragent, referrer, statuscode), and provide an output of the schema and the first couple of rows from the Dataframe itself in the screenshot.   
     
   Here is what the DataFrame Should look like:  
   Graphical user interface

   Description automatically generated

llogs3 = spark.read \

.option("header", False) \

.option("inferSchema", True) \

.option("sep", " ") \

.csv(logs\_out) \

.toDF("date","time", "serverip", "method", "uri", "querystirng", "port", "username", "clientip", "useragent", "referrer", "statuscode", "\_c12", "\_c13", "\_c14") \

.select("date","time", "serverip", "method", "uri", "querystirng", "port", "username", "clientip", "useragent", "referrer", "statuscode")

logs3.show(10)

logs3.printSchema()

1. Let's handle the IP Address lookup data. Write spark code to load the **iplookup.json** file from Minio into the data frame **ips1**. Show the first 10 rows and print the schema, for a screenshot to include code and output.

iplookup\_in = f"s3a://{s3\_bucket}/iplookup/iplookup.json"

ips1 = spark.read \

.option("multiline", True) \

.json(iplookup\_in)

ips1.show(10)

ips1.printSchema()

1. We need to flatten the nested JSON data, use the **select()** function with dot notation to do this, saving the dataframe as **ips2**. Provide a screenshot of the schema and output of the first few rows.

ips2 = ips1.select("ip", "geography.city", "geography.state", "geography.country", "location.lat", "location.lng")

ips2.show(10)

ips2.printSchema()

1. Now join the two dataframe together on their business key, making the new dataframe **comb1.** Provide a schema and sample of the first few rows in your screenshot.

comb1 = ips2.join(logs3, on = ips2.ip == logs3.clientip, how = "inner")

comb1.show(5)

comb1.printSchema()

1. Write the **comb1** Dataframe in **parquet** format back to Minio in the folder **cleaned-logs**. Again, show evidence the code ran, and the file was created.

cleanedlogs\_out = f"s3a://{s3\_bucket}/cleaned-logs.parquet"

comb1.write.mode("Overwrite").parquet(cleanedlogs\_out)

**IMPORTANT:** When you are finished with the lab, execute:

PS:> docker-compose stop

To turn off all running services, then shut down your Azure Lab instance.