Democratic and Popular Republic of Algeria
Ministry of Higher Education and Scientific Research
Higher School of Computer Science, May 8, 1945, Sidi Bel Abbes

Academic year: 2024-2025 **Semester 2**
2 SC: IASD **Module: Big Data Technologies**

Lab 1: Hadoop

Note: Save your answers in a document for review (consultation).

Exercise 1:

Installation:

Throughout this lab, we will use three containers representing a master node (Namenode) and two slave nodes (Datanodes).

After starting Docker, open the command line and perform the following tasks:

1. Download the Docker image **liliasfaxi/spark-hadoop:hv-2.7.2** uploaded on dockerhub.
2. Create the three containers from the downloaded image. To do this:
   - 2.1. Create a bridge network named **hadoop** that will connect the three containers.
   - 2.2. Create and launch the three containers:
   - The **hadoop-master** container exposes ports **50070**, **8088**, and **8080**.
   - The **hadoop-slave1** and **hadoop-slave2** containers expose port **8042**.
3. Access the master container to start using it.

After executing this step, the result will be:

```
root@hadoop-master:~#
```

You will be in the Namenode shell and can manipulate the cluster as desired. The first thing to do once inside the container is to start Hadoop and YARN. A script is provided for this purpose, called **start-hadoop.sh**. Run this script.

Getting Started with Hadoop:

- Create a directory in HDFS called **input**.
- We will use the file **purchases.txt**[1] as input for the MapReduce processing. This file is already located in the home directory of your master machine.
- Load the **purchases** file into the **input** directory you created.
- Display the contents of the **input** directory.
- Display the last lines of the **purchases** file.

MapReduce:

Presentation:

[1] https://github.com/CodeMangler/udacity-hadoop-course/raw/master/Datasets/purchases.txt.gz

A MapReduce job primarily consists of two types of programs:

- **Mappers:** Extract the necessary data in the form of key/value pairs, allowing them to be sorted based on the key.
- **Reducers:** Take a set of data sorted by their keys and perform the required processing on this data (e.g., sum, average, total, etc.).

WordCount:

We are going to test a MapReduce program using an example: WordCount. WordCount calculates the number of words in a given file by breaking the computation into two steps:

- **Mapping Step:** This step splits the text into words and outputs a text stream where each line contains the word found, followed by the value 1 (indicating that the word was found once).
- **Reducing Step:** This step sums up the 1s for each word to determine the total number of occurrences of that word in the text.

Let's start by creating a Maven project in IntelliJ. Define the following values for your project:

- **GroupId:** hadoop.mapreduce
- **ArtifactId:** wordcount

Open the **pom.xml** file and add the following dependencies for Hadoop, HDFS, and MapReduce:

```
<dependencies>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>2.7.2</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-mapreduce-client-core -->
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-mapreduce-client-core</artifactId>
        <version>2.7.2</version>
    </dependency>
```

```
        <!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-hdfs -->
        <dependency>
          <groupId>org.apache.hadoop</groupId>
          <artifactId>hadoop-hdfs</artifactId>
          <version>2.7.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-mapreduce-client-common</artifactId>
            <version>2.7.2</version>
         </dependency>
    </dependencies>
```

- Create a package **tp1** under the directory **src/main/java**.
- Create the **TokenizerMapper** class, which represents the MAP class.
- Create the **IntSumReducer** class, which represents the REDUCE class.
- Finally, create the **WordCount** class, which represents the Driver class.

Run MapReduce on the Cluster:

In your IntelliJ project:

- Generate the application's JAR file.
- Copy the created JAR file into the master container.
- Return to the shell of the master container and run the MapReduce job on the **purchases.txt** file that you previously loaded into the **input** directory of HDFS.
- Display the content of the generated file.

Exercise 2:

Using the file **purchases.txt** as input, write a Python MapReduce program to solve the WordCount problem and run it on the Hadoop cluster.

Exercise 3:

Using the file **purchases.txt** as input, which contains the following fields: **date time store product cost payment**, write a Python MapReduce program to calculate the total sales per store.

Exercise 4:

Using the dataset **Nigeria.csv** as input, which contains **event_date** in the format **m/d/yy**, write a Python MapReduce program to output a sorted list of dates in the format **yyyy-mm-dd** for each location where the government in Nigeria regained territory (i.e. the **event_type** contains the word "regains").

Exercise 5:

Write a MapReduce program in Python to group anagrams from a text file **(File: input_Anagram.txt)** where each line contains a list of words. The mapper should read each word, sort its characters alphabetically to create a key, and output key-value pairs of the sorted key and the original word. The reducer should group words by their keys and output each key along with a list of anagrams.

For example, given the input **melon barre lemon**, the output should be **elmno ['melon', 'lemon']** and **aberr ['barre']**.