MCIS 6263

Assignment 2

OBJECTIVES

- Getting familiar with the core Hadoop components
- Getting familiar with MapReduce Programming
- Getting familiar with Big Data processing using Hadoop MapReduce framework

DESCRIPTION

In this assignment, you'll install the QuickStart version of Cloudera's Express distribution of Hadoop. You'll also write your first MapReduce program which should implement the same functionality that you implemented in Assignment 1.

The assignment involves the following tasks:

Task 1: Install VMWare Workstation Player

The first step in building your Hadoop system is to build a cluster where Hadoop components can be deployed and run.

For the purpose of learning we will build our cluster in a virtual environment and deploy the different Hadoop services on a single host.

We will use VMWare as our virtual machine player. The VMWare player is free for non-commercial. You can download the right version for your platform from the following link:

https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/12_0

Make sure to select the option for Free non-commercial use when asked for the license key.

Task 2: Download and run the Cloudera VMWare Virtual Machine

Cloudera provides a QuickStart distribution of Hadoop for beginners in the form of a VMWare Linux virtual machine with Cloudera Hadoop Distribution pre-installed and pre-configured on it.

This distribution contains the core Hadoop components (HDFS, MapReduce, and Yarn), a number of components from the Hadoop eco-system, and a number of other management tools provided by Cloudera. This will save you the time needed to install and configure all this components.

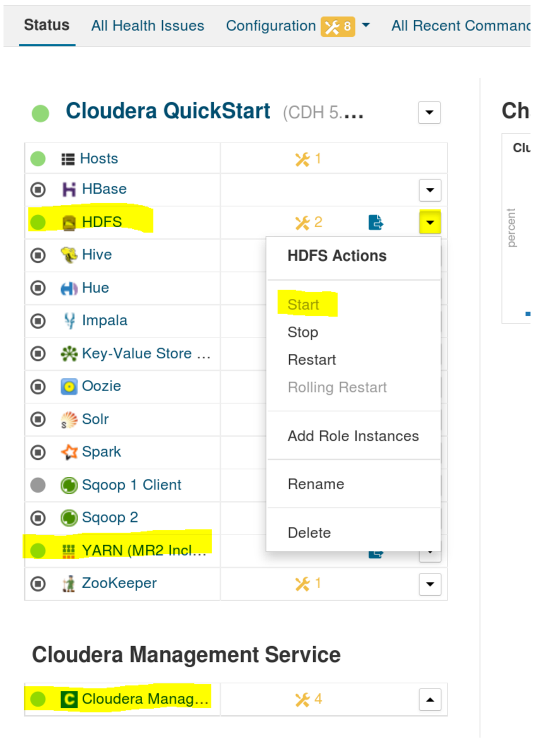
- Go to this URL: http://www.cloudera.com/downloads/quickstart_vms/5-8.html
- From the Platform list select "VMWare"
- Click "DOWNLOAD NOW"
- You'll be asked to register. Register as a student and enter our university information in the "Company" field
- The download process will start and a .zip file will be downloaded on your machine

- Decompress the .zip file to any folder on your machine (make sure you have enough space on the drive where you'll decompress the VM file).
- Run the VMWare player that you installed in Task 1
- Open the Cloudera VM that you just downloaded by selecting "File > Open" and then navigating to the path where you decompressed you VM file and selecting the .vmdk file founded in that path.
- Edit the settings of this virtual machine as follows:
  - Increase the memory size to 8192 MB (8GB)
  - Increase the number of processors to 2
- Start the virtual machine (You need to make sure that Hyper-v is enabled on your OS for VMWare to be able to run your VM)

Task 3: Configure your Hadoop Cluster

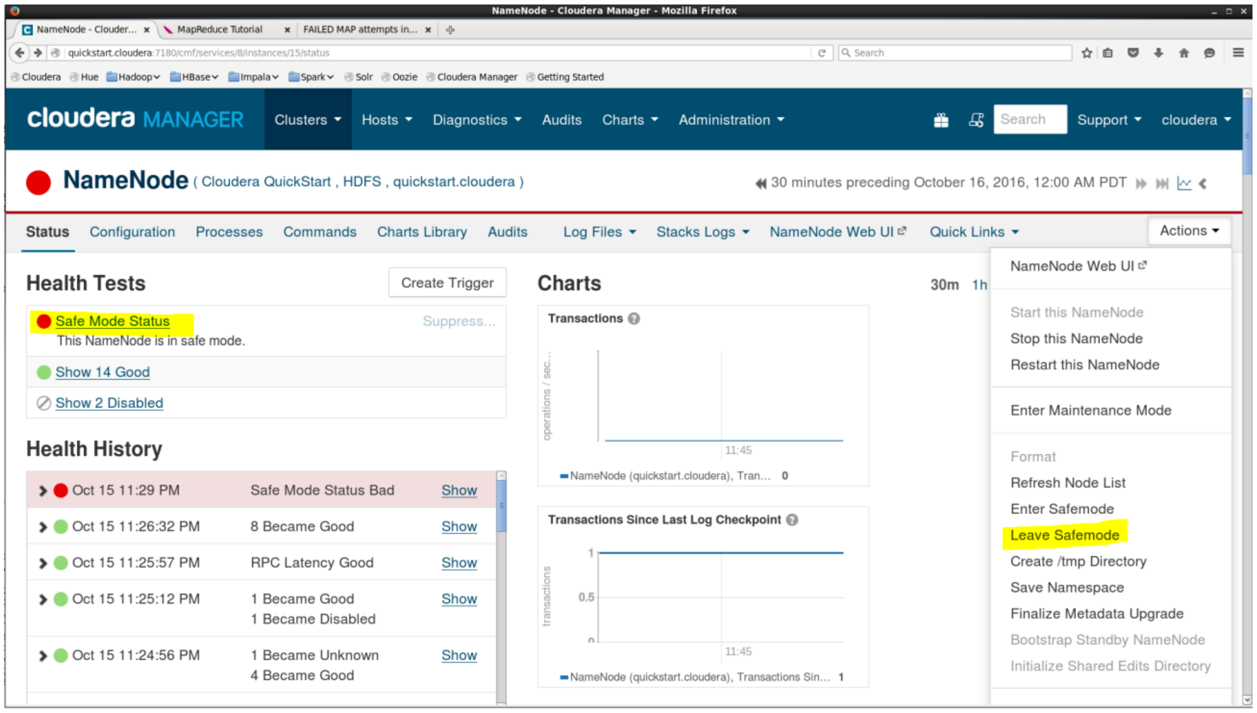
In this assignment we will be using a single-machine cluster. All the necessary services to run the code you'll write in this assignment will run on the VM. These include the HDFS services (DateNode, NameNode, and SecondaryNameNode) and the YARN services (ResourceManager and NodeManager).

This can be done by opening the Cloudera Manager in browser: http://quickstart.cloudera:7180 and starting the HDFS and the YARN services as show in the figure below.



Since, we will need to move the data to the cluster, you need to disable the SafeMode of the NameNode (Name nodes are defaulted to run in SafeMode which puts the data in a read-only mode).

This can be done from the Cloudera UI as well as shown in the figure below.



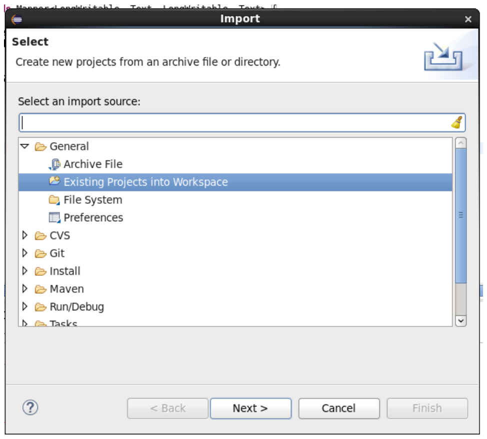
Oct 15 11:29 PM	Safe Mode Status Bad	Show
Oct 15 11:26:32 PM	8 Became Good	Show
Oct 15 11:25:57 PM	RPC Latency Good	Show
Oct 15 11:25:12 PM	1 Became Good 1 Became Disabled	Show
Oct 15 11:24:56 PM	1 Became Unknown 4 Became Good	Show

Task 4: Prepare the Eclipse Environment

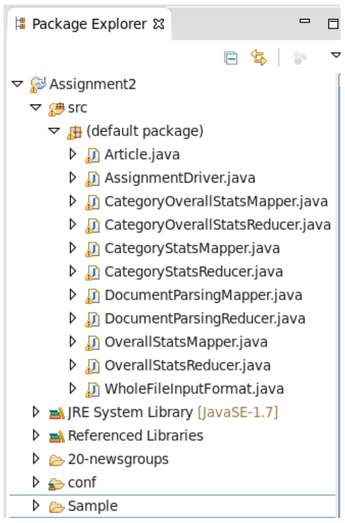
You'll use Eclipse for developing the MapReduce code of this assignment. The Cloudera VM comes with Eclipse pre-installed on it. You can find the shortcut to start it on the desktop of the VM.

To assist you with writing the code, a skeleton of the project is given to you in the assignment attachments.

- Download the **Assignment.zip** from the assignment page on blackboard and copy it to the VM (using drag and drop).
- In the VM, extract the compressed file. The result will be a folder named Assignment2 that contains the eclipse project and the 20-newsgroups data.
- Copy the folder to this path: /home/cloudera/workspace/ (this is where the eclipse project should go)
- To open the project in Eclipse, follow the following steps:
  - Start Eclipse (by clicking shortcut on desktop)
  - From Eclipse, go to **file > import**
  - From the window the opens up choose **General > Existing Project into Workspace**
  - Click **Next**.
  - Enter: "/home/cloudera/workspace/Assignment2" in the select root directory field
  - Click finish



Task 5: Develop the MapReduce program



```
Package Explorer
Assignment2
  src
    (default package)
      Article.java
      AssignmentDriver.java
      CategoryOverallStatsMapper.java
      CategoryOverallStatsReducer.java
      CategoryStatsMapper.java
      CategoryStatsReducer.java
      DocumentParsingMapper.java
      DocumentParsingReducer.java
      OverallStatsMapper.java
      OverallStatsReducer.java
      WholeFileInputFormat.java
  JRE System Library [JavaSE-1.7]
  Referenced Libraries
  20-newsgroups
  conf
  Sample
```

The MapReduce program that you need to develop in this assignment should do the same exact processing that you did in Assignment 1, but this time using the MapReduce framework.

A skeleton for the project is provided for you. Some of the code is already written for you. For the rest, you are given extensive detailed comments in each file that tells exactly what to do.

The project also includes the data. Both the sample data (Folder name "Sample") and the full data set (folder name "20-newsgroups").

The program should consist of 4 MapReduce Jobs as follows:

- **Job1: Parsing the data set**

  This job is similar to Assignment 1 / task 1. It's goal is to load the data set, parse the documents and convert the data to one large TSV file.

  - **Input**: The 20-newsgroups folder.
  - **InputFormat**: WholeFileInputFormat (this is a custom InputFormat that guarantees that the whole file will be processed by one Mapper. The code for this custom InputFormat is given for you). The **Key** of this InputFormat is of type Text and contains the full absolute path of the file being processed. The **Value** is of type Text and contains the full content of the file.
  - **Output**: A TSV file containing the following columns
    1. File Full Path
    2. Category
    3. Sender
    4. Subject
    5. Body
    6. Affiliation
  - **OutputFormat:** FileOutputFormat
  - **Mapper**: DocumentParsingMapper
    - **Input Key,Value**: Text, Text
    - **Output Key,Value**: Text, Article
      Article is a custom Value type. The definition is given for you. The constructor of the Article class takes the file content (i.e. the Mapper input Value) and the file path (the Mapper Input Key) and will do the parsing for you! Take a look at the Article class to learn how it works.
  - **Reducer**: DocumentParsingReducer
    - **Input Key,Value**: Text, Article
    - **Output Key,Value**: Text, Article
      Notice that the class Article has an override for the "toString()" method which outputs the columns 2-6 from above. Column 1 (the file path) is the Key and the reducer will print it at the beginning for you.

- **Job2: Computing the overall dataset statistics**
  The goal of this job is to consume the output of Job 1 and compute the first two statistics from Assignment 1 / Task 2:
    - The total number of documents (which should be same as the number of lines in the file)
    - The average word count of the document "body" in the data set. To get this, you need to compute the number of words in the "body" column for each document (line), then compute the average. For simplicity, assume that words are separated by single spaces.
  - **Input**: The output of Job1.
  - **InputFormat**: TextInputFormat (each line is processed by one Mapper. Key is LongWritable representing the line number. Value is the line content)
  - **Output**: A TSV file containing the two stats. For Example:

```
Count      43.0
Average    260.3488372093023
```

  - **OutputFormat**: FileOutputFormat
  - **Mapper**: OverallStatsMapper
    - **Input Key,Value**: LongWritable, Text
    - **Output Key,Value**: LongWritable, IntWritable
  - **Reducer**: OverallStatsReducer
    - **Input Key,Value**: LongWritable, IntWritable
    - **Output Key,Value**: Text, DoubleWritable

- **Job3: Computing per-category stats**
  The goal of this job is to consume the output of Job 1 and compute necessary per-category stats that are needed to compute the last 3 statistics requested in Assignment 1 / Task2. The actual computation of these stats will be done in Job 4 below. The computation we will do here is needed as an intermediate step.
  - **Input**: The output of Job1.
  - **InputFormat**: TextInputFormat (each line is processed by one Mapper. Key is LongWritable representing the line number. Value is the line content)
  - **Output**: A TSV file containing the list of categories and the average body size and document count (these two stats should be written comma separated). Example:

```
alt.atheism              442.5,20
comp.graphics            115.63636363636364,11
comp.os.ms-windows.misc    73.25,4
comp.sys.ibm.pc.hardware   97.5,8
.....
```

  - **OutputFormat**: FileOutputFormat
  - **Mapper**: CategoryStatsMapper
    - **Input Key,Value**: LongWritable, Text
    - **Output Key,Value**: Text, IntWritable
  - **Reducer**: CategoryStatReducer

        - **Input Key,Value**: Text, IntWritable
        - **Output Key,Value**: Text, Text

- **Job4: Computing category overall stats**
  The goal of this job is to consume the output of Job 3 and compute the last 3 statistics requested in Assignment 1 / Task2:
    1. Compute the average number of documents per category.
    2. Find the category with the maximum average "body" word count.
    3. Find the category with the minimum average "body" word count.
  - **Input**: The output of Job3.
  - **InputFormat**: KeyValueTextInputFormat (each line is processed by one Mapper. Key is Text and contains the first column in the input – the category in our case -. Value is the second column – the string containing the average body size and document count comma-separated)
  - **Output**: A TSV file containing the three stats: Example:

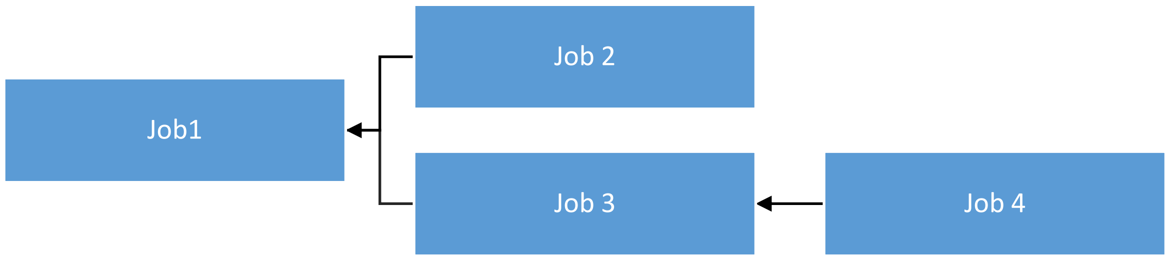
```
Average Document Per Category          10.75
Category With Max Avg Body Word Count          alt.atheism
Category With Min Avg Body Word Count          comp.os.ms-windows.misc
```

  - **OutputFormat**: FileOutputFormat
  - **Mapper**: CategoryOverallStatsMapper
    - **Input Key,Value**: Text, Text
    - **Output Key,Value**: Text, Text
  - **Reducer**: CategoryStatReducer
    - **Input Key,Value**: Text, Text
    - **Output Key,Value**: Text, Text

- **Job Controller**

  You need to create the 4 jobs above and then create a controlled Job for each of them. Then, you need to define the dependencies of the jobs. Next, you need to create a JobController (as we learned in class) to chain the jobs and then run them.

  The following figure shows the job dependency graph.



- **Run the code on the cluster**

  Following what we learned in class:

  - Export your code to a JAR file
  - Use the hdfs dfs command to copy the data to the Hadoop cluster.
  - Execute the code on the cluster

- ## What to Submit
  - The source code of the project in a file named Assignment2.zip
  - The JAR file of the project. Name it Assignment2.jar
    The jar file should take 3 arguments:
    - The input folder
    - The output folder for job 2
    - The output folder for job 3
  - A document in PDF or MS Word format that contains the following:
    - The names of your team member (maximum 3 members per team)
    - The source code of the following files
      - Assignment2Driver.java
      - DocumentParsingMapper.java
      - DocumentParsingReducer.java
      - OverallStatsMapper.java
      - OverallStatsReducer.java
      - CategoryStatsMapper.java
      - CategoryStatsReducer.java
      - CategoryOverallStatsMapper.java
      - CategoryOverallStatsReducer.java

      Make sure to copy the source code in colored format. Make sure that the code is colored to make it easier to read it.

    - The output of Job 2
    - The output of Job 3
    - Detailed instructions of how you loaded the data to the cluster and how you exported and ran the code. Include all the commands and screenshots that show your work.