Spark-Dask Project 2021

I/ HADOOP AND SPARK Installation

1/ Prerequires on Ubuntu, Windows 10 WSL

a/ Installation of ssh server

>\# installation of openssh server
>sudo apt install openssh-server
>#
>\# Edit config file to set port to 22 Port 22
>sudo vi /etc/ssh/sshd_config # set Line Port 22

b/ Command to Start, Stop, Restart ssh service

>sudo servive ssh start # start ssh service
>sudo servive ssh stop # stop ssh service
>sudo servive ssh restart # restart ssh service

c/ Java 1.8

> java -version

If command java not found
> sudo apt-get install openjdk-8-jdk

Check java versiion

```
>java -version
openjdk version "1.8.0_191"
OpenJDK Runtime Environment (build 1.8.0_191-8u191-b12-2ubuntu0.18.04.1-b12)
OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode)
```

2/ Hadoop Installation

Step 1 : Hadoop installation

```
> cd /home/myusername
> mkdir local ; cd local
> wget http://mirror.intergrid.com.au/apache/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz
> tar xvfz hadoop-3.3.0.tar.gz
> mv hadoop-3.3.0 hadoop
```

Step 2 : Create hadoop.sh

Create hadoop.sh file */home/myusername/local/hadoop/hadoop.sh*

>vi */home/myusername/local/hadoop/hadoop.sh*

set variable as following (set JAVA_HOME to your Ubuntu Java home value

```
# Set JAVA_HOME (we will also configure JAVA_HOME directly for Hadoop later on)
export JAVA_HOME=/usr/local/Java/1.8.0-xxx

# Set Hadoop-related environment variables
export HADOOP_HOME=/home/hduser/local/hadoop

export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME

export PATH=${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin:$PATH
```

Step 3 : create Hadoop working directories

```
# CREATE HADOOP TMP DIR
> mkdir -p /home/myusername/local/app/hadoop/tmp
> chmod 750 /home/myusername/localapp/hadoop/tmp

# CREATE HDFS WORKINGDIR TO MNG HDFS File System
> mkdir -p /home/myusername/local/var/local/hadoop/hdfs/data
> chmod -R 777 /home/myusername/local/var/local/hadoop/hdfs
```

Step 4 : set Hadoop configuration files

Configuration files in /home/myusername/local/hadoop/etc/hadoop

- hadoop-env.sh

```
JAVA_HOME="true java JOME path"
export JAVA_HOME=${JAVA_HOME}
```

- core-site.xml

```
<property>
      <name>hadoop.tmp.dir</name>
      <value>/home/myusername/local/app/hadoop/tmp</value>
      <description>A base for other temporary directories.</description>
</property>
<property>
      <name>fs.default.name</name>
      <value>hdfs://localhost:9000</value>
      <description>The name of the default file system.</description>
</property>
```

- hdfs-site.xml

```
<property>
    <name>dfs.data.dir</name>
    <value>/home/hduser/var/local/hadoop/hdfs/data</value>
    <final>true</final>
</property>

<property>
      <name>dfs.replication</name>
      <value>1</value>
</property>
```

- mapred-site.xml

```
<property>
      <name>mapred.job.tracker</name>
      <value>localhost:9001</value>
</property>
```

- yarn-site.xml settings

```
<configuration>
      <!-- Site specific YARN configuration properties -->
      <property>
            <name>yarn.nodemanager.aux-services</name>
            <value>mapreduce_shuffle</value>
      </property>
      <property>
            <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
            <value>org.apache.hadoop.mapred.ShuffleHandler</value>
      </property>
</configuration>
```

Step 5 : launch Hadoop services

```
> source /home/mysusername/local/hadoop/hadoop.sh


# Format Data nodes
> hadoop namenode -format

# Starting HDFS
> $HADOOP_HOME/sbin/start-hdfs.sh

# Startin Yarn
> $HADOOP_HOME/sbin/start-yarn.sh

# check that services are launched
> jps

26867 DataNode
28228 Jps
27285 ResourceManager
26695 NameNode
27082 SecondaryNameNode
27420 NodeManager

# check hdfs command
> hadoop fs -ls /
> hadoop fs -mkdir /user
> hadoop fs -ls /
```

3/ Spark Installation

Step 1 : Spark installation

```
> cd /home/myusername
> cd local
> wget https://downloads.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz
> tar xvfz spark-3.2.1-bin-hadoop3.2.tgz
> mv spark-3.2.1-bin-hadoop3.2 spark
```

Step 2 : Create spark.sh

Create spark.sh file */home/myusername/local/spark/spark.sh*

>vi */home/myusername/local/spark/spark.sh*

```
> export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
> export SPARK_HOME=/home/myusername/local/spark
> export PATH=$SPARK_HOME/bin:$PATH
> export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native:$LD_LIBRARY_PATH
```

Step 2 : Lauch spark services

Load Hadoop and spark environment

> source `/home/mysusername/local/hadoop/hadoop.sh`

> source `/home/mysusername/local/spark/spark.sh`

```
# lauch spark services
> sbin/start-master.sh
> sbin/start-slave.sh spark://localhost:7077
```

3/ Anaconda Installation

Step 1 : Anaconda installation

```
cd /home/myusername
> cd local
> mkdir anaconda3
> wget https://repo.continuum.io/archive/Anaconda3-2021.11-Linux-x86_64.sh
```

> sh Anaconda3-2021.11-Linux-x86_64.sh

> … answer question, acept licence et choose /home/myusername/local/anaconda3 directory for installation

Step 2 : Create anaconda.sh

Create anaconda.sh file */home/myusername/local/anaconda3/anaconda.sh*

>vi */home/myusername/local/anaconda3/anaconda.sh*

```
> export PATH=/home/myusername/local/anaconda3/bin:$PATH
> export LD_LIBRARY_PATH=/home/myusername/local/anaconda3/lib:$LD_LIBRARY_PATH
> export PYSPARK_PYTHON=/home/myusername/local/anaconda3/bin/python
```

Step 3 : install pyspark

> source */home/myusername/local/hadoop/hadoop.sh*

> source */home/myusername/local/spark/spark.sh*

> source */home/myusername/local/anaconda3/anaconda.sh*

> conda install pysark

II/ Project Description :

In the BigDataHadoopSpakDaskCourse git project, find the following files :

-Tps--- |-data---------- |iris.csv

| |lena_noisy.jpg

|-TP5----------|iris_ml.py

|-TP6----------|median_filter.py

|

|tools----------|launch_cmd_local.sh

A/ Spark ML : Iris classification

Data : iris.csv

In the python iris_ml.py script, a DecisionTreeClassifier is used to predict the flower species.

1/ Write a Spark Pipeline to transform and process the data before applying the machine learning process.

2/ Write two other versions of this test using :

- The Random Forest Classifier ;
- The Gradient Boosted Tree Classifier (transform the binary Classifier in a multi-class classifier using a tree classifier : for (A,B,C,D,…) classes, use (A, not A), (B, not B), (C, not C), (D, not D),... binalry classifiers.

Compare the performance of the three machine learning models.

B/ Spark : parallelisation of the image processing algorithm « MedianFilter »

The median filter consists in replace the value of a pixel p[i,j] by the median value of the list :

[p[i-1,j-1],p[i-1,j],p[i-1,j+1],p[i,j-1],p[i,j],p[i,j+1],p[i+1,j-1],p[i+1,j],p[i+1,j+1]]

Complete the file « median_filter.py » to write a spark parallel median_filter.py

Execute the script on the « lena_noizy.jpg » image and generate the new file « lena_filter.jpg »

Send me the script completed and te new generated file « lena_filter.jpg »

D/ Dask ML : Iris classification

Data : iris.csv

In the python iris_ml.py script, a DecisionTreeClassifier is used to predict the flower species.

1/ Write a Dask Pipeline to transform and process the data before applying the machine learning process.

2/ Write two other versions of this test using :

- The Random Forest Classifier ;
- The Gradient Boosted Tree Classifier.

Compare the performance of the three machine learning models.

E/ Dask : parallelisation of the image processing algorithm « MedianFilter »

The median filter consists in replace the value of a pixel p[i,j] by the median value of the list :

[p[i-1,j-1],p[i-1,j],p[i-1,j+1],p[i,j-1],p[i,j],p[i,j+1],p[i+1,j-1],p[i+1,j],p[i+1,j+1]]

Complete the file « median_filter.py » to write a dask parallel median_filter.py

Execute the script on the « lena_noizy.jpg » image and generate the new file « lena_filter.jpg »

Send me the script completed and te new generated file « lena_filter.jpg »