CS643 - Programming Assignment 2 Instructions

Kendy Colon (krc9)

4/22/2022

GITHUB AND DOCKER HUB

- https://github.com/krc9/Programm2_CS643
- https://hub.docker.com/repository/docker/krc993/krc9_cs643_assignment2

Parallel Training on 4 EC2 instances

This project was built with the Spark data frames API and the MLib libraries, and the application is automatically parallelized and distributed natively.

The wine prediction program is built with Spark DataFrames and MLlib. When you run it on an AWS EMR cluster, it is automatically run in a distributed way for parallelization task execution. For locating dataset files and storing trained models, the Hadoop Distributed Files System (HDFS) was employed.

How to create an AWS EMR Cluster ?

1. From the AWS console
2. Go to EMR Service and click Create Cluster
    a. Enter cluster name (cs643_asg2)
    b. Launch Mode cluster
    c. Vendor Amazon
    d. Release emr-5.3.10(5.35.0)
    e. Select Spark: Spark 2.4.8 on Hadoop 2.10.1 YARN and Zeppelin 0.10.0
    f. Hardware Configurations
        i. Select the instance type (m4.large)
        ii. Number of instances to 4 (1 master 3 slaves)
    g. Select the ec2 key pair or generate one to access the master node
    h. Save the key.pem and set correct permissions
    i. Click on create cluster

Upload files to EMR Cluster Master node

1. Once Cluster goes into waiting state, copy master node dns address and open command prompt on local machine.
2. Open sftp connection to master node
   - sftp -i keypair_name.pem hadoop@ec2-3-128-26-180.us-east-2.compute.amazonaws.com
3. Upload TrainingDataSet.csv, ValidationDataset.csv and wine_training.jar to master node.

SSH Master node :

ssh -i clusterkey.pem hadoop@ec2-54-1e5-3-552.compute-1.amazonaws.com

Copy files to HDFS :

Now that all files are on our master node, we want to migrate them to HDFS so that all slave nodes can access them without having to physically copy them to all ec2 nodes.

1. Use this command to copy files from Master node to HDFS.
   a. hadoop fs -put TrainingDataset.csv /user/hadoop/TrainingDataset.csv
   b. hadoop fs -put ValidationDataset.csv /user/hadoop/ValidationDataset.csv
   c. hadoop fs -put Programm2_CS643.jar /user/hadoop/Programm2_CS643.jar
2. Use this command to verify if files are successfully copied to HDFS
   a. hdfs dfs -ls -t -R

Launch TrainingModel application :

Now that we've completed everything, we'd like to deploy an apache-spark application on the EMR cluster. To run the application, run the following command.

1. spark-submit Programm2_CS643.jar

By heading to the monitor tab and then the spark dashboard, we can validate task execution.

1. This will create a TrainingModel folder and store trained models to it.
2. Verify model is created by executing following :
   a. hdfs dfs -ls -t -R
3. Now copy This folder back to our master node using following
   a. hdfs dfs -copyToLocal TrainingModel /home/hadoop/wine

Because the training is done here, we need to move the training files to our local environment so that we may use them to forecast on a single ec2 with or without docker.

1. Make a tar.gz zip of folder so that we can download it
    a. tar -czf model.tar.gz TrainingModel
2. In our sftp session execute following to download mode.tar.gz on local machine
    a. get model.tar.gz
3. Shut down the ERM cluster

Step 2: Predict wine quality on single ec2 instance

At this stage we are interested in executing prediction code on a single ec2 instance. For that we need TestDataset.csv, wine_predict.jar, model.tar.gz (from task1)

Ec2 instance Create:

- After logging into AWS console,
- Go to EC2 -> launch instance
- Select keypair and launch it.

Ec2 instance pre configuration:

- Do ssh to ec2 instance public dns,
    - ssh -i clusterkey.pem ec2-user@ec2-54-158-81-112.compute-1.amazonaws.com
- **Install SCALA:**
    - wget http://downloads.typesafe.com/scala/2.11.6/scala-2.11.6.tgz
    - tar -xzvf scala-2.11.6.tgz
    - Update PATH environment variable:
        - vim ~/.bashrc
        - copy following lines into file and then save it

            § export SCALA_HOME=/home/ec2-user/scala-2.11.6

            § export PATH=$PATH:/home/ec2-user/scala-2.11.6/bin

        - Reload bash_profile

        - source ~/.bashrc
    - Install Java
        - wget --no-check-certificate -c --header "Cookie: oraclelicense=accept-securebackup-cookie" https://download.oracle.com/java/17/archive/jdk-17.0.1_linux-aarch64_bin.rpm
        - sudo rpm -Uvh jdk-17.0.1_linux-aarch64_bin.rpm
- **Install SPARK:**
    - wget https://archive.apache.org/dist/spark/spark-2.4.5/spark-2.4.5-bin-hadoop2.7.tgz
    - sudo tar xvf spark-2.4.5-bin-hadoop2.7.tgz -C /opt
    - sudo chown -R ec2-user:ec2-user /opt/spark-2.4.5-bin-hadoop2.7
    - sudo ln -fs spark-2.4.5-bin-hadoop2.7 /opt/spark
    - Update PATH Environment
        - vim ~/.bash_profile
        - Copy following lines into file and then save it
            - export SPARK_HOME=/opt/spark
            - PATH=$PATH:$SPARK_HOME/bin
            - export PATH
        - Reload profile
            - source ~/.bash_profile

- **Upload trained model and jar files :**
    - Login to ec2 instance using sftp
    - sftp -i clusterkey.pem ec2-user@ec2-54-226-161-118.compute-1.amazonaws.com
    - put Programm2_CS643_Predict.jar
    - put TestDataset.csv
    - put training_output/*gz
- **Extract model.tar.gz :**
    - tar -xzvf model.tar.gz
- **Disable unnecessary log4j :**
- cp $SPARK_HOME/conf/log4j.properties.template $SPARK_HOME/conf/log4j.properties
- vi $SPARK_HOME/conf/log4j.properties
- Change line 19 to the log level from INFO to ERROR
- log4j.rootCategory=ERROR, console
- Save the file and exit the text editor.

Run wine-predict application :

- spark-submit Programm2_CS643_Predict.jar

Step 3 : Predict wine quality using docker:

Using Docker to predict wine quality on TestDataset.csv. We need to have the complete local file path to TestDataset.csv and pass it as an input argument to Docker while it is executing. So that before starting, TestDataset.csv may be transferred to the local docker environment.

The test filename must be TestDataset.csv, and the file must be stored in the container's data/ directory. To accomplish this, use the -v argument to map volumes.

Install Docker

- sudo yum install docker
- Start Docker daemon
  - sudo systemctl enable docker.service
  - sudo systemctl start docker.service
- Check that dockers is up
  - sudo systemctl status docker.service

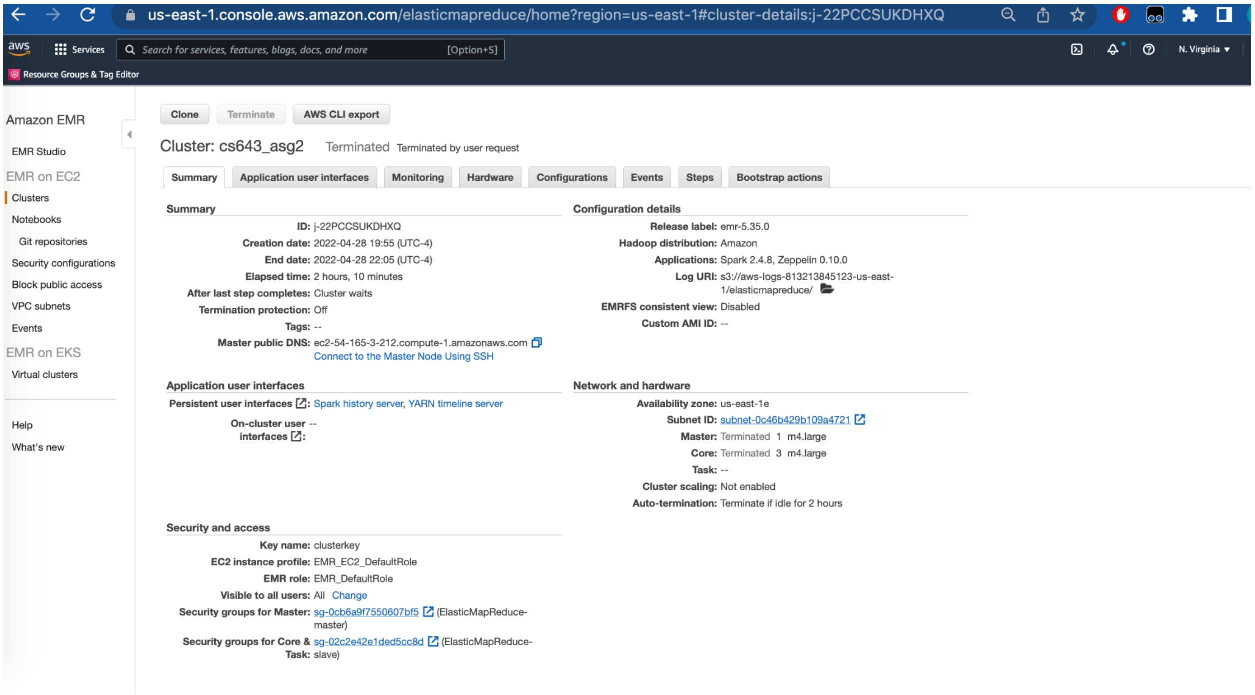
Execute the following command.

1. docker pull krc993/krc9_cs643_assignment2
2. docker run -v /home/ec2-user/data/TestDataset.csv

General use following format :

1. sudo docker run krc993/krc9_cs643_assignment2 /home/ec2-user/data/

https://hub.docker.com/repository/docker/krc993/krc9_cs643_assignment2



Cluster: cs643_asg2 Terminated Terminated by user request

Summary

ID: j-22PCCSUKDHXQ
Creation date: 2022-04-28 19:55 (UTC-4)
End date: 2022-04-28 22:05 (UTC-4)
Elapsed time: 2 hours, 10 minutes
After last step completes: Cluster waits
Termination protection: Off
Tags: --
Master public DNS: ec2-54-165-3-212.compute-1.amazonaws.com
Connect to the Master Node Using SSH

Configuration details

Release label: emr-5.35.0
Hadoop distribution: Amazon
Applications: Spark 2.4.8, Zeppelin 0.10.0
Log URI: s3://aws-logs-813213845123-us-east-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --

Application user interfaces

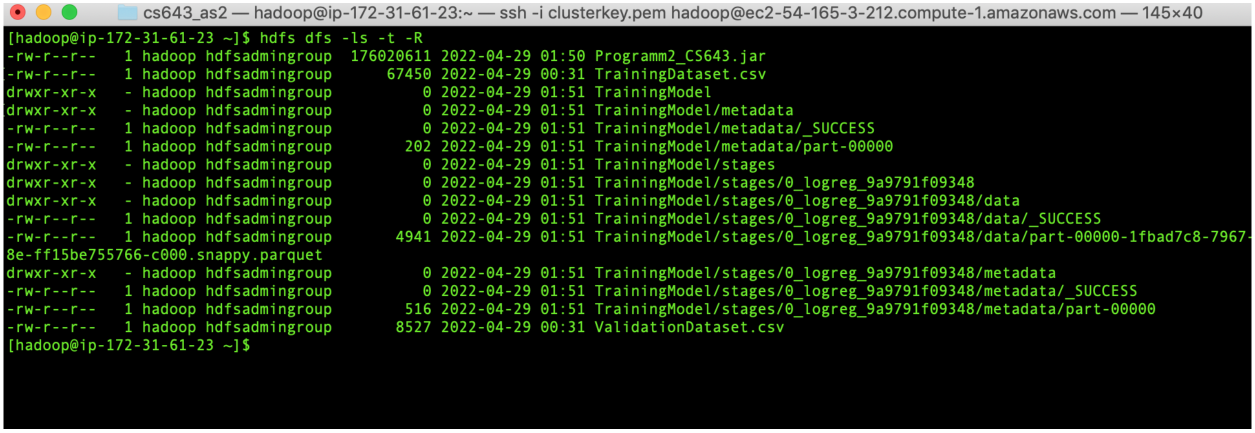
Persistent user interfaces: Spark history server, YARN timeline server
On-cluster user interfaces: --

Network and hardware

Availability zone: us-east-1e
Subnet ID: subnet-0c46b429b109a4721
Master: Terminated 1 m4.large
Core: Terminated 3 m4.large
Task: --
Cluster scaling: Not enabled
Auto-termination: Terminate if idle for 2 hours

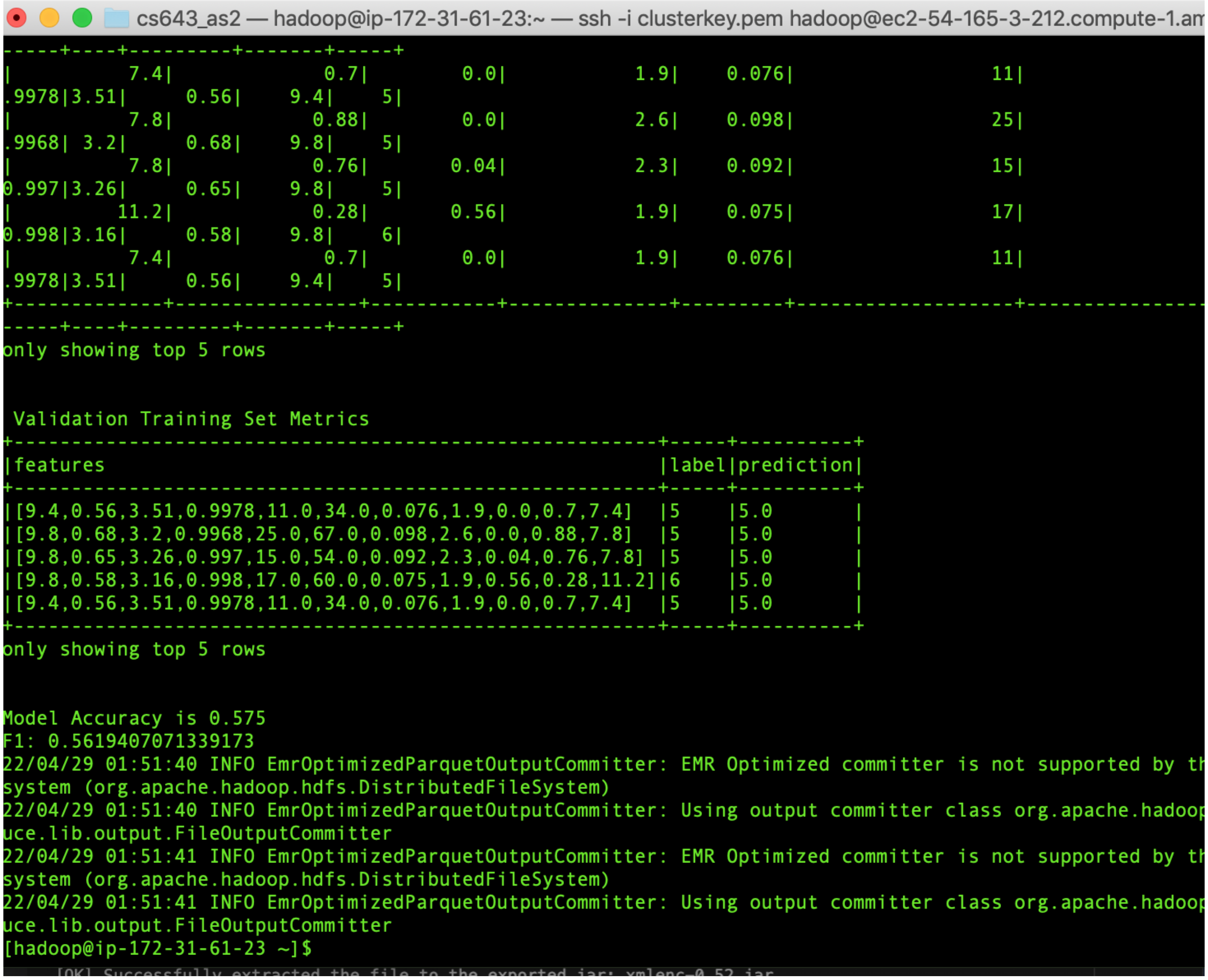
Security and access

Key name: clusterkey
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: All Change
Security groups for Master: sg-0cb6a9f7550607bf5 (ElasticMapReduce-master)
Security groups for Core & Task: sg-02c2e42e1ded5cc8d (ElasticMapReduce-slave)



cs643_as2 — hadoop@ip-172-31-61-23:~ — ssh -i clusterkey.pem hadoop@ec2-54-165-3-212.compute-1.amazonaws.com — 145×40

```
[hadoop@ip-172-31-61-23 ~]$ hdfs dfs -ls -t -R
-rw-r--r--   1 hadoop hdfsadmingroup  176020611 2022-04-29 01:50 Programm2_CS643.jar
-rw-r--r--   1 hadoop hdfsadmingroup      67450 2022-04-29 00:31 TrainingDataset.csv
drwxr-xr-x   - hadoop hdfsadmingroup          0 2022-04-29 01:51 TrainingModel
drwxr-xr-x   - hadoop hdfsadmingroup          0 2022-04-29 01:51 TrainingModel/metadata
-rw-r--r--   1 hadoop hdfsadmingroup          0 2022-04-29 01:51 TrainingModel/metadata/_SUCCESS
-rw-r--r--   1 hadoop hdfsadmingroup        202 2022-04-29 01:51 TrainingModel/metadata/part-00000
drwxr-xr-x   - hadoop hdfsadmingroup          0 2022-04-29 01:51 TrainingModel/stages
drwxr-xr-x   - hadoop hdfsadmingroup          0 2022-04-29 01:51 TrainingModel/stages/0_logreg_9a9791f09348
drwxr-xr-x   - hadoop hdfsadmingroup          0 2022-04-29 01:51 TrainingModel/stages/0_logreg_9a9791f09348/data
-rw-r--r--   1 hadoop hdfsadmingroup          0 2022-04-29 01:51 TrainingModel/stages/0_logreg_9a9791f09348/data/_SUCCESS
-rw-r--r--   1 hadoop hdfsadmingroup       4941 2022-04-29 01:51 TrainingModel/stages/0_logreg_9a9791f09348/data/part-00000-1fbad7c8-7967-
8e-ff15be755766-c000.snappy.parquet
drwxr-xr-x   - hadoop hdfsadmingroup          0 2022-04-29 01:51 TrainingModel/stages/0_logreg_9a9791f09348/metadata
-rw-r--r--   1 hadoop hdfsadmingroup          0 2022-04-29 01:51 TrainingModel/stages/0_logreg_9a9791f09348/metadata/_SUCCESS
-rw-r--r--   1 hadoop hdfsadmingroup        516 2022-04-29 01:51 TrainingModel/stages/0_logreg_9a9791f09348/metadata/part-00000
-rw-r--r--   1 hadoop hdfsadmingroup       8527 2022-04-29 00:31 ValidationDataset.csv
[hadoop@ip-172-31-61-23 ~]$
```



cs643_as2 — hadoop@ip-172-31-61-23:~ — ssh -i clusterkey.pem hadoop@ec2-54-165-3-212.compute-1.am

```
-----+----+---------+-------+-----+
|          7.4|             0.7|        0.0|           1.9|    0.076|                  11|
.9978|3.51|     0.56|    9.4|    5|
|          7.8|            0.88|        0.0|           2.6|    0.098|                  25|
.9968| 3.2|     0.68|    9.8|    5|
|          7.8|            0.76|       0.04|           2.3|    0.092|                  15|
0.997|3.26|     0.65|    9.8|    5|
|         11.2|            0.28|       0.56|           1.9|    0.075|                  17|
0.998|3.16|     0.58|    9.8|    6|
|          7.4|             0.7|        0.0|           1.9|    0.076|                  11|
.9978|3.51|     0.56|    9.4|    5|
+-------------+----------------+-----------+--------------+---------+--------------------+----------------
-----+----+---------+-------+-----+
only showing top 5 rows


 Validation Training Set Metrics
+---------------------------------------------------------+-----+----------+
|features                                                 |label|prediction|
+---------------------------------------------------------+-----+----------+
|[9.4,0.56,3.51,0.9978,11.0,34.0,0.076,1.9,0.0,0.7,7.4]   |5    |5.0       |
|[9.8,0.68,3.2,0.9968,25.0,67.0,0.098,2.6,0.0,0.88,7.8]   |5    |5.0       |
|[9.8,0.65,3.26,0.997,15.0,54.0,0.092,2.3,0.04,0.76,7.8]  |5    |5.0       |
|[9.8,0.58,3.16,0.998,17.0,60.0,0.075,1.9,0.56,0.28,11.2] |6    |5.0       |
|[9.4,0.56,3.51,0.9978,11.0,34.0,0.076,1.9,0.0,0.7,7.4]   |5    |5.0       |
+---------------------------------------------------------+-----+----------+
only showing top 5 rows


Model Accuracy is 0.575
F1: 0.5619407071339173
22/04/29 01:51:40 INFO EmrOptimizedParquetOutputCommitter: EMR Optimized committer is not supported by th
system (org.apache.hadoop.hdfs.DistributedFileSystem)
22/04/29 01:51:40 INFO EmrOptimizedParquetOutputCommitter: Using output committer class org.apache.hadoop
uce.lib.output.FileOutputCommitter
22/04/29 01:51:41 INFO EmrOptimizedParquetOutputCommitter: EMR Optimized committer is not supported by th
system (org.apache.hadoop.hdfs.DistributedFileSystem)
22/04/29 01:51:41 INFO EmrOptimizedParquetOutputCommitter: Using output committer class org.apache.hadoop
uce.lib.output.FileOutputCommitter
[hadoop@ip-172-31-61-23 ~]$
    [OK] Successfully extracted the file to the exported jar: xmlenc-0.52.jar
```