CSP554—Big Data Technologies

Assignment #8

Worth: 20 points

Exercise 1) 5 points

Read the article "The Lambda and the Kappa" found on our blackboard site in the "Articles" section and answer the following questions using between 1-3 sentences each. Note this, article provides a real-world and critical view of the lambda pattern and some related big data processing patterns:

1. (1 point) Extract-transform-load (ETL) is the process of taking transactional business data (think of data collected about the purchases you make at a grocery store) and converting that data into a format more appropriate for reporting or analytic exploration. What problems was encountering with the ETL process at Twitter (and more generally) that impacted data analytics?

2. (1 point) What example is mentioned about Twitter of a case where the lambda architecture would be appropriate?

3. (2 points) What did Twitter find were the two of the limitations of using the lambda architecture?

4. (1 point) What is the Kappa architecture?

5. (1 point) Apache Beam is one framework that implements a kappa architecture. What is one of the distinguishing features of Apache Beam?

Exercise 2) 5 points

Read the article "Real-time stream processing for Big Data" available on the blackboard in the 'Articles' section and then answer the following questions:

a) (1.25 points) What is the Kappa architecture and how does it differ from the lambda architecture?

b) (1.25 points) What are the advantages and drawbacks of pure streaming versus micro-batch real-time processing systems?

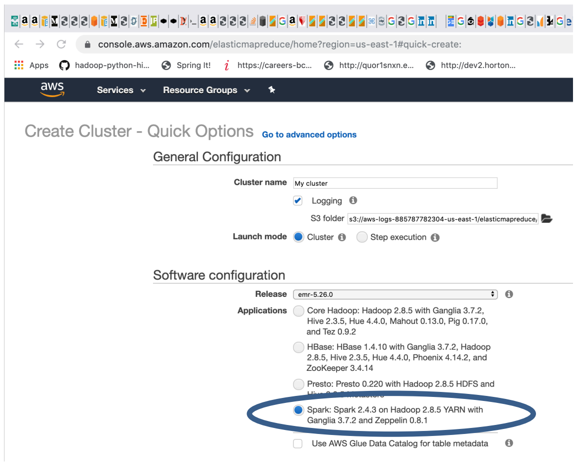
c) (1.25 points) In few sentences describe the data processing pipeline in Storm.

d) (1.25 points) How does Spark streaming shift the Spark batch processing approach to work on real-time data streams?

Exercise 3) 5 points

<u>Step A – Start an EMR cluster</u>

Start up an EMR/Hadoop cluster as previously, but instead of choosing the "Core Hadoop" configuration chose the "Spark" configuration (see below), otherwise proceed as before.



console.aws.amazon.com/elasticmapreduce/home?region=us-east-1#quick-create:

Create Cluster - Quick Options

General Configuration

Cluster name: My cluster

[✓] Logging

S3 folder: s3://aws-logs-885787782304-us-east-1/elasticmapreduce/

Launch mode: [✓ Cluster] Step execution

Software configuration

Release: emr-5.26.0

Applications:
- Core Hadoop: Hadoop 2.8.5 with Ganglia 3.7.2, Hive 2.3.5, Hue 4.4.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.2
- HBase: HBase 1.4.10 with Ganglia 3.7.2, Hadoop 2.8.5, Hive 2.3.5, Hue 4.4.0, Phoenix 4.14.2, and ZooKeeper 3.4.14
- Presto: Presto 0.220 with Hadoop 2.8.5 HDFS and Hive Metastore
- [✓ Spark: Spark 2.4.3 on Hadoop 2.8.5 YARN with Ganglia 3.7.2 and Zeppelin 0.8.1]

Use AWS Glue Data Catalog for table metadata

Step B – Copy the Kafka software to the EMR master node

Download the latest binary Kafka release from https://kafka.apache.org/downloads - please make sure to download the binary release for Kafka 2.13. Use the secure copy (scp) program to move this file to the /home/hadoop directory of the master node. Here is an example of how your command line might look (yours will be somewhat different because your master node DNS name, key-pair and kafka file locations will vary):

```
scp -i ~/emr-key-pair-2.cer /Users/nachdaph/csp554-fall-2021/assignments/kafka_2.13-3.0.0.tgz hadoop@ec2-3-218-249-33.compute-1.amazonaws.com:/home/hadoop
```

Step C – Install the Kafka software and start it

Open up a terminal connection to your EMR master node. Over the course of this exercise, you will need to open up three separate terminal connections to your EMR master node. This is the first, which we will call Kafka-Term:

Enter the following command:

```
tar -xzf kafka_2.13-3.3.1.tgz
```

Note, this will create a new directory (kafka_2.13-3.3.1) holding the kakfa software release.

Then enter this command:

```
pip install kafka-python
```

This installs the kafka-python package.

Now enter the following commands into the terminal:

```
cd kafka_2.13-3.3.1

bin/zookeeper-server-start.sh config/zookeeper.properties &

bin/kafka-server-start.sh config/server.properties &
```

Remember to end each line with the ampersand (&). This starts up a zookeeper instance and the kafka server. Lots of messages should appear. You might need to tap the return/enter key after messages appear to see the Linux prompt again.

Just leave this terminal window alone after you enter these commands. As you interact with kafka this terminal will display low level diagnostic messages which you can ignore.

Step D – Prepare to run Kafka producers and consumers

Open a second terminal connection to the EMR master node. Going forward we will call this terminal connection: Producer-Term.

Open a third terminal connection to the EMR master node. Going forward we will call this terminal connection: Consumer-Term.

Step E – Create a Kafka topic

In the Producer-Term, enter the following command:

```
cd kafka_2.13-3.3.1
```

```
bin/kafka-topics.sh --create --replication-factor 1 --partitions 1 --bootstrap-server
localhost:9092 --topic sample
```

Here we create a new kafka topic called ‘sample’. You can use this command to create a topic with any name you like. Try creating a few more topics.

To list the topics that you created you can enter the following into the Producer-Term (note some default topics already exist):

```
bin/kafka-topics.sh --list --bootstrap-server localhost:9092
```

a)

In the Producer-Term (or some other way) write a small program, call it ‘put.py’, using the vi text or some other way of putting a python program onto the EMR master node. If you like you could use a text editor on your PC/MAC to write the program and then scp it over to your EMR master name.

This program should implement a kafka producer that writes three messages to the topic ‘sample’. Recall that you need to convert values and keys to type bytes. The three messages should have keys and values as follows:

Key	Value
‘MYID’	Your student id
‘MYNAME’	Your name
‘MYEYECOLOR’	Your eye color (make it up if you can’t remember)

Execute this program in the Producer-Term, use the command line (you might need to provide a full pathname depending on where your python program is such as /home/hadoop/someplace/put.py):

```
python put.py
```

Submit the program as your answer to ‘part a’ of this exercise.

b)

In the Consumer-Term, write another small program, call it ‘get.py’, using the vi text or some other way of putting a python program onto the EMR master node.

This program should implement a kafka consumer that reads the messages you wrote previously from the topic ‘sample’ and writes them to the terminal.

The output should look something like this:

Key=MYID, Value=’your id number’

Key=MYNAME, Value=’your name’

Key=MYEYECOLOR, Value=’your eye color’

Execute this program in the Consumer-Term. Use the command line:

```
python get.py
```

Note, if needed you can terminate the program by entering ‘ctrl-c’.

Submit the program and a screenshot of its output as your answer to ‘part b’ of this exercise.

c)

Remember to terminate your EMR cluster!

Exercise 4) 5 points

These steps illustrate how to execute a pyspark (Python) spark streaming job. The job accepts a sequence of lines that the user types in onto one terminal window over a 10 second interval and then counts the number of distinct words in those lines and outputs the word count results to a second terminal window. This continues every 10 seconds. To do this we will set up a Spark EMR cluster and connect two terminal windows to it. In the first we will run the Linux ‘nc’ (Netcat) command. It will open a TCP socket on port 3333. After it does so, any line you then type will be sent out on that port. In another terminal window we will execute a pyspark word count program that will set up the spark streaming pipeline using DStreams. Our initial DStream will be connected to and read the lines from port 3333 and then go on to perform the word count process.

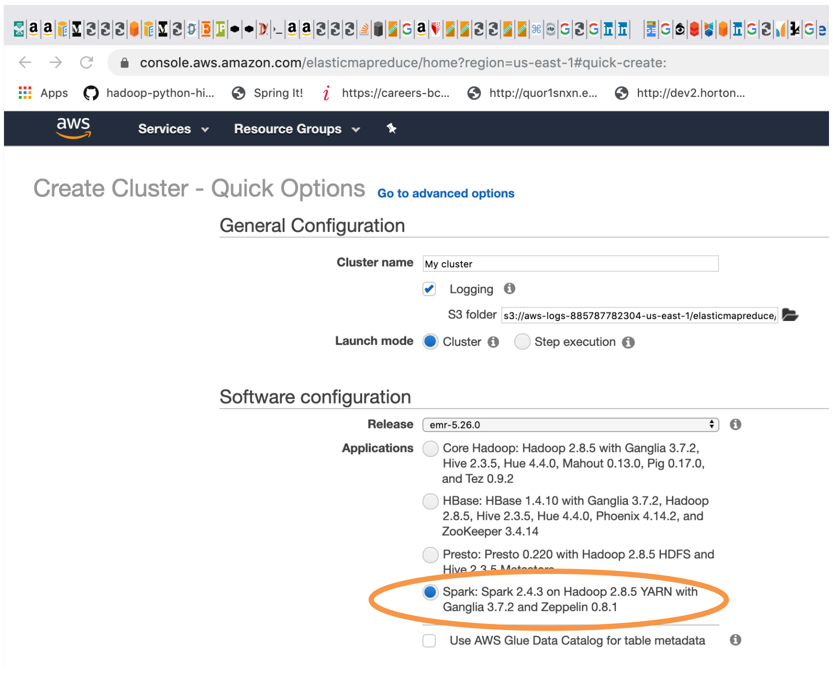
So on one terminal (connected to the EMR master node) you might see:

```
[hadoop@ip-172-31-19-223 ~]$ nc -lk ec2-3-91-10-18.compute-
1.amazonaws.com 3333

this is a test of the the system                <- note
```

And output from the word count program running in the other terminal should look something like:

```
-------------------------------------------

Time: 2019-11-05 21:25:00

-------------------------------------------

(u'a', 1)

(u'this', 1)

(u'is', 1)

(u'test', 1)

(u'the', 2)

(u'of', 1)

(u'system', 1)
```

1) Start up a Hadoop cluster as previously, but instead of choosing the “Core Hadoop” configuration chose the “Spark” configuration (see below), otherwise proceed as before.



console.aws.amazon.com/elasticmapreduce/home?region=us-east-1#quick-create:

Create Cluster - Quick Options — Go to advanced options

General Configuration

Cluster name: My cluster

[✓] Logging

S3 folder: s3://aws-logs-885787782304-us-east-1/elasticmapreduce/

Launch mode: [● Cluster] ○ Step execution

Software configuration

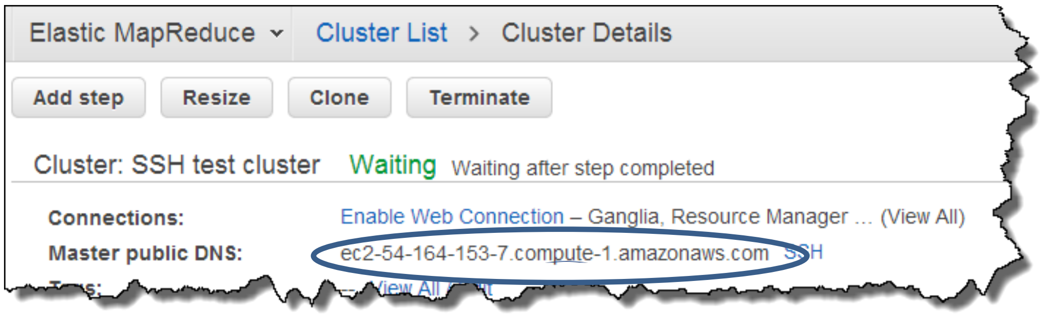
Release: emr-5.26.0

Applications:

- ○ Core Hadoop: Hadoop 2.8.5 with Ganglia 3.7.2, Hive 2.3.5, Hue 4.4.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.2
- ○ HBase: HBase 1.4.10 with Ganglia 3.7.2, Hadoop 2.8.5, Hive 2.3.5, Hue 4.4.0, Phoenix 4.14.2, and ZooKeeper 3.4.14
- ○ Presto: Presto 0.220 with Hadoop 2.8.5 HDFS and Hive 2.3.5 Metastore
- [● Spark: Spark 2.4.3 on Hadoop 2.8.5 YARN with Ganglia 3.7.2 and Zeppelin 0.8.1]

☐ Use AWS Glue Data Catalog for table metadata

2) At a later point in these instructions you will need to use the public DNS name of the master node of your EMR cluster. To retrieve it using the Amazon EMR console
   a) Find the EMR service page.
   b) On the **Cluster List** page, select the link for your cluster.
   c) Note the **Master public DNS** value that appears at the top of the **Cluster Details** page.



Elastic MapReduce › Cluster List › Cluster Details

Add step | Resize | Clone | Terminate

Cluster: SSH test cluster Waiting Waiting after step completed

Connections: Enable Web Connection – Ganglia, Resource Manager ... (View All)

Master public DNS: ec2-54-164-153-7.compute-1.amazonaws.com SSH

3) Download consume.py and log4j.properties files from the assignment to your local PC or MAC
4) There is one item you must change in consume.py. In the following line you must replace <Master public DNS> with your own public DNS name (found as described above)

```
lines = ssc.socketTextStream("<Master public DNS>", 3333)
```

For example:

```
lines = ssc.socketTextStream("ec2-54-164-153-7.compute-1.amazonaws.com”, 3333)
```

5) scp this modified consume.py file to your EMR cluster master node. You may need to answer a security question with “Y/y” or “Yes”.
6) Then scp the file log4j.properties to your EMR cluster master node.
7) Open two terminal sessions to the EMR master node. We will call one the EC2-1 window and the other the EC2-2 window.
8) In the EC2-1 window enter the following:

   ```
   sudo cp ./log4j.properties /etc/spark/conf/log4j.properties
   ```

   This changes the logging properties to turn off “INFO” messages to allow easier viewing of the results of the stream processing job. But it is not something you always want to disable.

9) In the EC2-1 window enter the following command to open a TCP (socket) connection on port 3333

   ```
   nc -lk 3333
   ```

10) In the EC2-2 window enter the following command:

    ```
    spark-submit consume.py
    ```

    This takes a while to start up. So, wait for some messages issued to the console before continuing. Note, when you do this you might see a message beginning with “WARN StreamingContext:...” which you can ignore.

11) Now in the EC2-1 window enter one or more lines of text and press Enter/Return after each one including the last. You should see the word count results scroll by in the EC2-2 window
12) Remember to terminate your EMR instance after you are done!