PySpark-analysis-file

November 10, 2020

0.1 Migrating from Spark to BigQuery via Dataproc – Part 1

- Part 1: The original Spark code, now running on Dataproc (lift-and-shift).
- Part 2: Replace HDFS by Google Cloud Storage. This enables job-specific-clusters. (cloud-native)
- Part 3: Automate everything, so that we can run in a job-specific cluster. (cloud-optimized)
- Part 4: Load CSV into BigQuery, use BigQuery. (modernize)
- Part 5: Using Cloud Functions, launch analysis every time there is a new file in the bucket. (serverless)

[5]:
```
%%writefile spark_analysis.py

import matplotlib
matplotlib.use('agg')

import argparse
parser = argparse.ArgumentParser()
parser.add_argument("--bucket", help="bucket for input and output")
args = parser.parse_args()

BUCKET = args.bucket
```

```
Overwriting spark_analysis.py
```

0.1.1 Copy data to HDFS

The Spark code in this notebook is based loosely on the code accompanying this post by Dipanjan Sarkar. I am using it to illustrate migrating a Spark analytics workload to BigQuery via Dataproc.

The data itself comes from the 1999 KDD competition. Let's grab 10% of the data to use as an illustration.

0.1.2 Reading in data

The data are CSV files. In Spark, these can be read using textFile and splitting rows on commas.

[6]:
```
%%writefile -a spark_analysis.py

from pyspark.sql import SparkSession, SQLContext, Row
gcs_bucket="qwiklabs-gcp-02-6236be6b5a47"
```

```
spark = SparkSession.builder.appName("kdd").getOrCreate()
sc = spark.sparkContext
# data_file = "hdfs:///kddcup.data_10_percent.gz"
data_file = "gs://"+gcs_bucket+"//kddcup.data_10_percent.gz"
raw_rdd = sc.textFile(data_file).cache()
raw_rdd.take(5)
```

```
Appending to spark_analysis.py
```

[7]:
```
%%writefile -a spark_analysis.py

csv_rdd = raw_rdd.map(lambda row: row.split(","))
parsed_rdd = csv_rdd.map(lambda r: Row(
    duration=int(r[0]),
    protocol_type=r[1],
    service=r[2],
    flag=r[3],
    src_bytes=int(r[4]),
    dst_bytes=int(r[5]),
    wrong_fragment=int(r[7]),
    urgent=int(r[8]),
    hot=int(r[9]),
    num_failed_logins=int(r[10]),
    num_compromised=int(r[12]),
    su_attempted=r[14],
    num_root=int(r[15]),
    num_file_creations=int(r[16]),
    label=r[-1]
    )
)
parsed_rdd.take(5)
```

```
Appending to spark_analysis.py
```

0.1.3 Spark analysis

One way to analyze data in Spark is to call methods on a dataframe.

[8]:
```
%%writefile -a spark_analysis.py

sqlContext = SQLContext(sc)
df = sqlContext.createDataFrame(parsed_rdd)
connections_by_protocol = df.groupBy('protocol_type').count().orderBy('count',␣
 ↪ascending=False)
connections_by_protocol.show()
```

```
Appending to spark_analysis.py
```

Another way is to use Spark SQL

[9]:
```
%%writefile -a spark_analysis.py

df.registerTempTable("connections")
attack_stats = sqlContext.sql("""
                           SELECT
                             protocol_type,
                             CASE label
                               WHEN 'normal.' THEN 'no attack'
                               ELSE 'attack'
                             END AS state,
                             COUNT(*) as total_freq,
                             ROUND(AVG(src_bytes), 2) as mean_src_bytes,
                             ROUND(AVG(dst_bytes), 2) as mean_dst_bytes,
                             ROUND(AVG(duration), 2) as mean_duration,
                             SUM(num_failed_logins) as total_failed_logins,
                             SUM(num_compromised) as total_compromised,
                             SUM(num_file_creations) as total_file_creations,
                             SUM(su_attempted) as total_root_attempts,
                             SUM(num_root) as total_root_acceses
                           FROM connections
                           GROUP BY protocol_type, state
                           ORDER BY 3 DESC
                           """)
attack_stats.show()
```

```
Appending to spark_analysis.py
```

[10]:
```
%%writefile -a spark_analysis.py

ax[0].get_figure().savefig('report.png');
```

```
Appending to spark_analysis.py
```

[11]:
```
%%writefile -a spark_analysis.py

import google.cloud.storage as gcs
bucket = gcs.Client().get_bucket(BUCKET)
for blob in bucket.list_blobs(prefix='sparktodp/'):
    blob.delete()
bucket.blob('sparktodp/report.png').upload_from_filename('report.png')
```

```
Appending to spark_analysis.py
```

[12]:
```
%%writefile -a spark_analysis.py

connections_by_protocol.write.format("csv").mode("overwrite").save(
    "gs://{}/sparktodp/connections_by_protocol".format(BUCKET))
```

```
Appending to spark_analysis.py
```

[13]:
```
BUCKET_list = !gcloud info --format='value(config.project)'
BUCKET=BUCKET_list[0]
print('Writing to {}'.format(BUCKET))
!/opt/conda/anaconda/bin/python spark_analysis.py --bucket=$BUCKET
```

```
Writing to qwiklabs-gcp-02-6236be6b5a47
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
+-------------+------+
|protocol_type| count|
+-------------+------+
|         icmp|283602|
|          tcp|190065|
|          udp| 20354|
+-------------+------+

+-------------+---------+----------+--------------+--------------+-------------+
-------------------+-----------------+--------------------+-------------------+-
-----------------+
|protocol_type|    state|total_freq|mean_src_bytes|mean_dst_bytes|mean_duration|
total_failed_logins|total_compromised|total_file_creations|total_root_attempts|t
otal_root_acceses|
+-------------+---------+----------+--------------+--------------+-------------+
-------------------+-----------------+--------------------+-------------------+-
-----------------+
|         icmp|   attack|    282314|        932.14|           0.0|          0.0|
0|                0|                   0|                0.0|                 0|
|          tcp|   attack|    113252|       9880.38|        881.41|        23.19|
57|             2269|                  76|                1.0|
152|
|          tcp|no attack|     76813|       1439.31|       4263.97|        11.08|
18|             2776|                 459|               17.0|
5456|
|          udp|no attack|     19177|         98.01|         89.89|      1054.63|
0|                0|                   0|                0.0|                 0|
|         icmp|no attack|      1288|         91.47|           0.0|          0.0|
0|                0|                   0|                0.0|                 0|
|          udp|   attack|      1177|          27.5|          0.23|          0.0|
0|                0|                   0|                0.0|                 0|
+-------------+---------+----------+--------------+--------------+-------------+
-------------------+-----------------+--------------------+-------------------+-
-----------------+

Traceback (most recent call last):
  File "spark_analysis.py", line 70, in <module>
```

```
    ax[0].get_figure().savefig('report.png');
NameError: name 'ax' is not defined
```

[14]:
```
!gsutil ls gs://$BUCKET/sparktodp/**
```

```
gs://qwiklabs-gcp-02-6236be6b5a47/sparktodp/spark_analysis.py
```

[15]:
```
!gsutil cp spark_analysis.py gs://$BUCKET/sparktodp/spark_analysis.py
```

```
Copying file://spark_analysis.py [Content-Type=text/x-python]…
/ [1 files][  2.7 KiB/  2.7 KiB]
Operation completed over 1 objects/2.7 KiB.
```