School of Architecture, Computing and Engineering (ACE)

Author: Dr Amin Karami (a.karami@uel.ac.uk) Term 1 19-20

Tutorial 8: RDD Programming on a log file using PySpark

CN7022 - Big Data Analytics

Dr Amin Karami (a.karami@uel.ac.uk)

LEARNING OUTCOMES: After completing this tutorial, you should:

- Have gotten a hands-on experience in loading data from HDFS into Spark
- Be able to analyse log files by PySpark
- Be able to apply descriptive statistics on RDD-based APIs
- Practice Python and Spark commands

Section 1: Working with log files

Step 1: Understanding Data

One of the most common topic in big data analytics domain is processing log files, because they usually have large volume with unstructured format. PySpark is one of the methodologies fits for log file analysing. In this tutorial, we will analyse **GHTorrent** project log files. **GHTorrent** monitors the **Github** public event time line. For each event, it retrieves its contents and their dependencies, exhaustively. It then stores the raw JSON responses to a MongoDB database, while also extracting their structure in a MySQL database. For more information, refer to http://ghtorrent.org/. In this tutorial we will use a portion of this data with the size of **1.13GB**. Since, we cannot open this file directly, we need to understand its format as follows[1]:

```
DEBUG, 2017-03-23T10:02:27+00:00, ghtorrent-40 -- ghtorrent.rb: Repo EFForg/https-everywhere
exists

DEBUG, 2017-03-24T12:06:23+00:00, ghtorrent-49 -- ghtorrent.rb: Repo Shikanime/print exists

INFO, 2017-03-23T13:00:55+00:00, ghtorrent-42 -- api_client.rb: Successful request. URL:
https://api.github.com/repos/CanonicalLtd/maas-docs/issues/365/events?per_page=100,
Remaining: 4943, Total: 88 ms

WARN, 2017-03-23T20:04:28+00:00, ghtorrent-13 -- api_client.rb: Failed request. URL:
https://api.github.com/repos/greatfakeman/Tabchi/commits?sha=Tabchi&per_page=100, Status
code: 404, Status: Not Found, Access: ac6168f8776, IP: 0.0.0.0, Remaining: 3031

DEBUG, 2017-03-23T09:06:09+00:00, ghtorrent-2 -- ghtorrent.rb: Transaction committed (11 ms)
```

The format of this log file is as follows:

[1] source: http://www.gousios.gr/courses/bigdata/assignment-spark-solutions-python.html

1. Logging level, one of DEBUG, INFO, WARN, ERROR (separated by ,)
2. A timestamp (separated by ,)
3. The downloader id, denoting the downloader instance (separated by --)
4. The retrieval stage:
   - event_processing
   - ght_data_retrieval
   - api_client
   - retriever
   - ghtorrent

Step 2: Download and Parsing the data

Download the zipped file from here, then unzip it by this command (`gunzip` command is for opening `.gz` files): `gunzip <path>/ghtorrent-logs.txt.gz`. Afterwards, run Hadoop with `start-all.sh`. Then, put data into HDFS by this command: `hdfs dfs -put <path>/ghtorrent-logs.txt /`

After putting data into HDFS, make sure it is available: `hdfs dfs -ls /`

Go to Spark directory (`cd $SPARK_HOME`) to run Spark engine (`./sbin/start-all.sh`), then type `pyspark` in the terminal to launch PySpark. Open a new notebook and go ahead with the following codes.

1. Practice the following codes (read comments to understand the codes) in PySpark:

```
# Phase 1: Load Data
# -----------------------------
# for parsing rdd rows
# Columns:
# 0: logging level, 1: timestamp, 2: downloader id,
# 3: retrieval stage, 4: Action?
def myParse(line):
    line = line.replace(' -- ', ', ')
    line = line.replace('.rb: ', ', ')
    line = line.replace(', ghtorrent-', ', ')
    return line.split(', ', 4)
```

```
# Load Data from HDFS and put it into a RDD

def getRDD():
    txtfile=sc.textFile("hdfs://localhost:9000/ghtorrent-logs.txt",4)
    rdd=txtfile.map(myParse)
    return rdd
```

```
# Call getRDD() to make RDD and cache it into Memory
rowRDD = getRDD().cache()
```

```
print(rowRDD.count())   # count the number of lines/records in RDD
```

```
9669788
```

```
print(rowRDD.take(2))  # print the frist two items
```

```
[['INFO', '2017-03-22T20:11:49+00:00', '31', 'ghtorrent', 'Added pullreq_commit 244eeac28bf419642d5d5c3b388bd2
999c8c72e6 to tgstation/tgstation -> 25341'], ['DEBUG', '2017-03-23T11:15:14+00:00', '30', 'retriever', 'Commi
t mzvast/FlappyFrog -> 80bf5c5fde7be6274a2721422f4d9a773583f73c exists']]
```

```
# Find the number of "WARN" messages
WarnCount = rowRDD.filter(lambda x:x[0]=="WARN")
WarnCount.count()
```

```
132158
```

2. How many repositories where processed in total? Use the `api_client` lines only.

```python
import itertools
# Add repositories as column 5

# rewrite with split, and use only api_client
def parseRepos(x):
    try:
        # Filter for repos by looking for it in url
        # For instance:
        # Successful request. URL: https://api.github.com/repos/CanonicalLtd/maas-docs
        # /issues/365/events?per_page=100, Remaining: 4943, Total: 88 ms
        # Should return "CanonicalLtd/maas-docs/maas-docs"
        split = x[4].split('/')[4:6]
        joinedSplit = '/'.join(split)
        result = joinedSplit.split('?')[0]
    except:
        result = ''
    x.append(result)
    return x


# Filters out rows without enough elements
filteredRDD = rowRDD.filter(lambda x: len(x) == 5)

# Only look at api_client calls
apiRDD = filteredRDD.filter(lambda x: x[3] == "api_client")

# Add another column with the repo if can find one, otherwise ''
reposRDD = apiRDD.map(parseRepos)
```

```python
# Filter out rows without repo
removedEmpty = reposRDD.filter(lambda x: x[5] != '')

# Group by repo and count
uniqueRepos = removedEmpty.groupBy(lambda x: x[5])
print(uniqueRepos.count())
```

```
78588
```

3. Which client did the most HTTP requests?

```python
# Group by, count and find max

usersHTTP = apiRDD.groupBy(lambda x: x[2])
usersHTTPSum = usersHTTP.map(lambda x: (x[0], x[1].__len__()))
print(usersHTTPSum.max(key=lambda x: x[1]))
```

```
('13', 135978)
```

4. What is the most active hour of day?

```python
# Get hour of the day from timestamp and add it
def appendAndReturn(x, toAdd):
    x.append(toAdd)
    return x

# Split date to hour only
onlyHours = filteredRDD.map(lambda x: appendAndReturn(x, x[1].split('T', 1)[1].split(':', 1)[0]))

# Group by, count, find max
groupOnlyHours = onlyHours.groupBy(lambda x: x[5])
hoursCount = groupOnlyHours.map(lambda x: (x[0], x[1].__len__()))
print(hoursCount.max(key=lambda x: x[1]))
```

```
('10', 2662487)
```

5. What is the most active repository (use messages from the `ghtorrent.rb` layer only)?

```
# Group by, count, find max
activityRepos = removedEmpty.groupBy(lambda x: x[5])
countActivityRepos = activityRepos.map(lambda x: (x[0], x[1].__len__()))
print(countActivityRepos.max(key=lambda x: x[1]))
```

```
('greatfakeman/Tabchi', 79524)
```

Section 2: Statistics on RDD-based API

(1) Go to the Spark MLlib web-link https://spark.apache.org/docs/2.2.0/mllib-statistics.html and work on basic descriptive analysis on RDD-based applications with PySpark.

(2) You can also work on “Descriptive Statistics.pdf” and “Advanced Analysis.pdf” added at Week 7 in Moodle, and find out how we are able to employ them in PySpark. They are basically available in statistics library in Python.