LogAnalysis

May 15, 2019

```
In [1]: from pyspark import SparkContext, SparkConf
        conf = SparkConf().setAppName("SecondarySort")
        sc = SparkContext(conf=conf)
        sc.version
```

```
Out[1]: '2.4.2'
```

```
In [2]: from pyspark.sql import SparkSession
        spark = SparkSession(sc)
```

```
In [3]: import geoip2.database
        import os
        reader = geoip2.database.Reader(os.getcwd() + '/GeoLite2-Country.mmdb')
        c = reader.country('14.215.177.39')
        print(c.country.names)
```

```
{'de': 'China', 'en': 'China', 'es': 'China', 'fr': 'Chine', 'ja': '', 'pt-BR': 'China', 'ru': '
```

```
In [4]: path = "file://" + os.getcwd() + "/access.log"
        rawData = sc.textFile(path)
        rawData.map(lambda x : x.split(",")) \
                .map(lambda x : (x[0], x[1], x[2])) \
                .toDF(("TimeStamp", "url", "IP_address")) \
                .show(5, False)
```

```
+-------------------+-------------------------------+--------------+
|TimeStamp          |url                            |IP_address    |
+-------------------+-------------------------------+--------------+
|2017-05-11 14:09:14|http://www.ign.com/video/4500  |87.214.232.203|
|2017-05-11 15:25:05|http://www.ign.com/video/14623 |81.45.64.179  |
|2017-05-11 07:50:01|http://www.ign.com/article/17894|80.210.128.135|
|2017-05-11 02:46:43|http://www.ign.com/article/17896|64.35.194.206 |
|2017-05-11 09:30:25|http://www.ign.com/article/17893|222.221.42.166|
+-------------------+-------------------------------+--------------+
only showing top 5 rows
```

```
In [5]: def mapfunc1(line):
            import geoip2.database
            line = line.strip()
            timeStamp, url, IP = tuple(line.split(","))
            elements = url.split("/")
            contentId, contentType = elements[-1], elements[-2]
            try:
                reader = geoip2.database.Reader(os.getcwd() + '/GeoLite2-Country.mmdb')
                c = reader.country(IP)
                c_name = c.country.names['en']
            except:
                c_name = "NotFound"

            return (contentId, contentType, c_name, IP, timeStamp)

In [6]: allData = rawData.map(mapfunc1)
        allData.toDF(("contentId", "contentType", "country", "IPAddress", "timeStamp")) \
                .show(5, False)
```

```
+---------+-----------+-------------+--------------+-------------------+
|contentId|contentType|country      |IPAddress     |timeStamp          |
+---------+-----------+-------------+--------------+-------------------+
|4500     |video      |Netherlands  |87.214.232.203|2017-05-11 14:09:14|
|14623    |video      |Spain        |81.45.64.179  |2017-05-11 15:25:05|
|17894    |article    |Iran         |80.210.128.135|2017-05-11 07:50:01|
|17896    |article    |United States|64.35.194.206 |2017-05-11 02:46:43|
|17893    |article    |China        |222.221.42.166|2017-05-11 09:30:25|
+---------+-----------+-------------+--------------+-------------------+
only showing top 5 rows
```

```
In [7]: def mapfunc2(line):
            line = line.strip()
            timeStamp, url, IP = tuple(line.split(","))
            elements = url.split("/")
            contentId, contentType = elements[-1], elements[-2]
            return (contentId, contentType, 1)

        contentidcount = rawData.map(mapfunc2) \
                .filter(lambda x : x[1] == "video") \
                .map(lambda x: (x[0], x[2])) \
                .reduceByKey(lambda x, y : x + y) \
                .sortBy(lambda x : x[1], False)

        contentidcount.toDF(("contentId", "count")) \
                .show()
```

```
+---------+------+
|contentId| count|
+---------+------+
|    14540|111027|
|     4000| 55734|
|    14704| 55701|
|    14390| 55683|
|    14623| 55621|
|     4600| 55501|
|     4500| 55366|
|    14322| 55102|
+---------+------+
```

```
In [ ]: def mapfunc3(line):
            line = line.strip()
            timeStamp, url, IP = tuple(line.split(","))
            elements = url.split("/")
            contentId, contentType = elements[-1], elements[-2]
            try:
                reader = geoip2.database.Reader(os.getcwd() + '/GeoLite2-Country.mmdb')
                c = reader.country(IP)
                c_name = c.country.names['en']
            except:
                c_name = "NotFound"

            return (contentId, c_name)

        from pyspark.sql import Window
        from pyspark.sql import functions as F
        rawData.map(mapfunc3) \
                .toDF(("contentId", "country")) \
                .groupBy("country", "contentId").count() \
                .select("contentId", "country", "count",  \
                        F.row_number().over(Window.partitionBy("country") \
                                           .orderBy(F.col("count").desc())).alias("rank")) \
                .show(10)
```

```
In [9]:
```

For all results showed above, I run the code on local. But for the last question, the running session takes me 2 hours and still can't get result. Then I run the code on CloudxLab. Though it still run slowly, approximate 30 mins, I got the result below.

```
>>> rawData.map(mapfunc3) \
...         .toDF(("contentId", "country")) \
...         .groupBy("country", "contentId").count() \
...         .select("contentId", "country", "count",  \
...                 F.row_number().over(Window.partitionBy("country") \
...                                    .orderBy(F.col("count").desc())).alias("rank")) \
...         .show(10)
+---------+--------+-----+----+
|contentId| country|count|rank|
+---------+--------+-----+----+
|    14540|    Chad|    2|   1|
|    17897|    Chad|    1|   2|
|    17899|    Chad|    1|   3|
|    17894|    Chad|    1|   4|
|    17891|    Chad|    1|   5|
|    14540|Paraguay|   33|   1|
|    17891|Paraguay|   19|   2|
|     4600|Paraguay|   18|   3|
|    17899|Paraguay|   16|   4|
|    14322|Paraguay|   16|   5|
+---------+--------+-----+----+
only showing top 10 rows
```