Case Study: Log Parsing

Domain: Telecom

A telecom software provider is building an application to monitor different telecom components in the production environment. For monitoring purpose, the application relies on log files by parsing the log files and looking for potential warning or exceptions in the logs and reporting them.

The Dataset contains the log files from different components used in the overall telecom application.

Tasks: The volume of data is quite large. As part of the R&D team, you are building a solution on spark to load and parse the multiple log files and then arranging the error and warning by the timestamp.

1. **Load file as a text file in spark**

   access_logs_DF = spark.read.text("/user/edureka_524533/Datasets/access.clean.log")

2. **Find out how many 404 HTTP codes are in access logs.**

   Count404 = result.where(result['StatusCode']=='404').count()

   Count404

   227101

3. **Find out which URLs are broken.**

   None

4. **Verify there are no null columns in the original dataset.**

   # Verify there are no null columns in the original dataset.

   bad_rows_df = result.filter(result['Host'].isNull()|
   result['Timestamp'].isNull() |
   result['Method'].isNull() |
   result['StatusCode'].isNull() |
   result['Bytes'].isNull()|
   result['Date'].isNull())

   bad_rows_df.count()

   0

```
3]: bad_rows_df.show()
+-----+---------+----+----+-----+------+---+----------+
|value|Timestamp|Date|Host|Bytes|Method|URL|StatusCode|
+-----+---------+----+----+-----+------+---+----------+
+-----+---------+----+----+-----+------+---+----------+
```

5. **Replace null values with constants such as 0**

   Previous returned zero, thus none replaced with 0

6. **Parse timestamp to readable date.**

   #Extract time stamps

   ts_pattern = r'\d{2}/\w{3}/\d{4}:\d{2}:\d{2}:\d{2}'

   result = access_logs_DF.withColumn('Timestamp', regexp_extract(col('value'), ts_pattern, 0))

   dt_pattern = r'\d{2}/\w{3}/\d{4}'

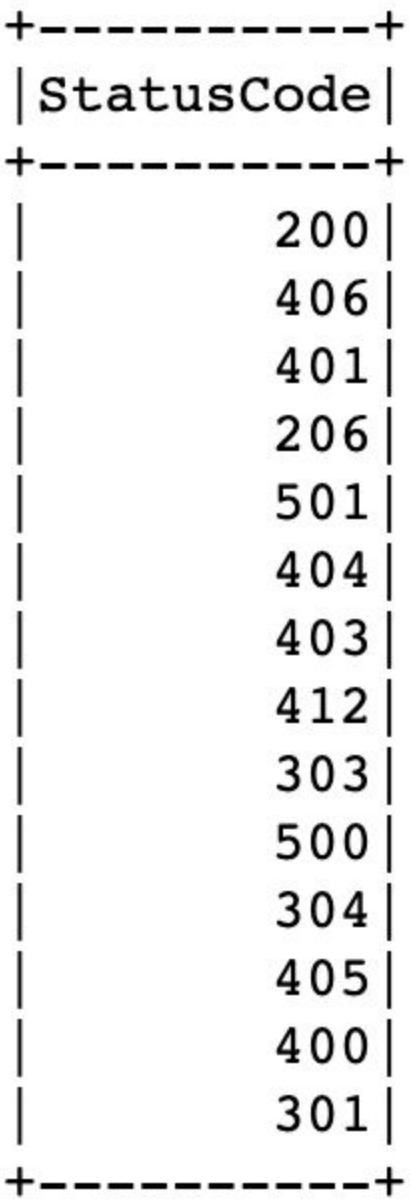
   result = result.withColumn('Date', regexp_extract(col('TimeStamp'), dt_pattern, 0))

7. **Describe which HTTP status values appear in data and how many.**

   StatusCodeDF = result.select(result['StatusCode'])

   StatusCodeDF=StatusCodeDF.dropDuplicates()

   StatusCodeDF.show()



StatusCode
200
406
401
206
501
404
403
412
303
500
304
405
400
301

StatusCodeDF.count()

8. **Display as chart the above stat in chart in Zeppelin notebook**

   Not sure as not taught by the trainer

9. **How many unique hosts are there in the entire log and their average request**

   HostsDF = result.select(result['Host'],result['Bytes'])

   HostsDF = HostsDF.groupby('Host').agg(F.mean('Bytes'),F.count('Bytes').alias('AverageRequests'))

HostsDF.show()

```
In [61]: HostsDF.show()
```

```
+--------------+------------------+--------------+
|          Host|        avg(Bytes)|AverageRequests|
+--------------+------------------+--------------+
|   46.72.177.4|            4378.5|             8|
| 194.48.218.78|            4378.5|             2|
| 31.181.253.16|            4378.5|             2|
|  37.112.46.76|            4378.5|             2|
| 95.107.90.225|            4378.5|             2|
|  5.138.58.118|            4378.5|             2|
|95.188.228.228|            4378.5|             2|
|  66.7.119.112|          887508.0|             1|
| 145.255.2.176|            4378.5|             4|
| 176.59.208.95|            4378.5|             2|
| 62.133.162.65|            4378.5|             4|
| 95.29.129.235|            4378.5|             2|
|  66.249.64.64|10022.774193548386|            41|
| 207.46.13.165| 8839.333333333334|             6|
| 180.76.15.162|29545.735294117647|            75|
|  37.139.52.40|         1396107.0|            16|
| 89.144.209.67| 53847.52173913043|            26|
|23.106.216.107|         1507464.0|             3|
|  195.20.125.6|            4378.5|            18|
| 92.113.63.101|            4378.5|             6|
+--------------+------------------+--------------+
only showing top 20 rows
```

10. Create a spark-submit application for the same and print the findings in the log

Module : mod5cs2.py

Screen Shots:

```
19/07/10 16:24:28 INFO scheduler.DAGScheduler: Job 0 finished: count at NativeMetho
19/07/10 16:24:28 INFO codegen.CodeGenerator: Code generated in 7.800413 ms
Number of null columns
0
19/07/10 16:24:28 INFO datasources.FileSourceStrategy: Pruning directories with:
19/07/10 16:24:28 INFO datasources.FileSourceStrategy: Post-Scan Filters:
19/07/10 16:24:28 INFO datasources.FileSourceStrategy: Output Data Schema: struct<
```

```
19/07/10 16:24:32 INFO codegen.CodeGenerator: Code generated in 7.210169 
19/07/10 16:24:32 INFO storage.BlockManagerInfo: Removed broadcast_5_piece
+----------+
|StatusCode|
+----------+
|       200|
|       406|
|       401|
|       206|
|       501|
|       404|
|       403|
|       412|
|       303|
|       500|
|       304|
|       405|
|       400|
|       301|
+----------+

19/07/10 16:24:32 INFO storage.BlockManagerInfo: Removed broadcast_5_piece
19/07/10 16:24:32 INFO storage.BlockManagerInfo: Removed broadcast_6_piece
```

```
19/07/10 16:24:35 INFO scheduler.DAGScheduler: Job 6 finished: count at Na
Number of unique status codes
14
19/07/10 16:24:35 INFO spark.ContextCleaner: Cleaned accumulator 5246
19/07/10 16:24:35 INFO spark.ContextCleaner: Cleaned accumulator 5253
19/07/10 16:24:35 INFO spark.ContextCleaner: Cleaned accumulator 5260
19/07/10 16:24:35 INFO spark.ContextCleaner: Cleaned accumulator 5238
```

```
19/07/10 16:24:39 INFO scheduler.DAGScheduler: Job 7 finished: showString at NativeMet
19/07/10 16:24:39 INFO codegen.CodeGenerator: Code generated in 6.766376 ms
+--------------+------------------+---------------+
|          Host|        avg(Bytes)|AverageRequests|
+--------------+------------------+---------------+
|   46.72.177.4|            4378.5|              8|
| 194.48.218.78|            4378.5|              2|
| 31.181.253.16|            4378.5|              2|
|  37.112.46.76|            4378.5|              2|
| 95.107.90.225|            4378.5|              2|
|  5.138.58.118|            4378.5|              2|
|95.188.228.228|            4378.5|              2|
|  66.7.119.112|          887508.0|              1|
| 145.255.2.176|            4378.5|              4|
| 176.59.208.95|            4378.5|              2|
| 62.133.162.65|            4378.5|              4|
| 95.29.129.235|            4378.5|              2|
|  66.249.64.64|10022.774193548386|             41|
| 207.46.13.165| 8839.333333333334|              6|
| 180.76.15.162|29545.735294117647|             75|
|  37.139.52.40|         1396107.0|             16|
| 89.144.209.67| 53847.52173913043|             26|
|23.106.216.107|         1507464.0|              3|
|  195.20.125.6|            4378.5|             18|
| 92.113.63.101|            4378.5|              6|
+--------------+------------------+---------------+
only showing top 20 rows

19/07/10 16:24:39 INFO datasources.FileSourceStrategy: Pruning directories with:
```

```
19/07/10 16:24:42 INFO scheduler.DAGScheduler: Job 8 finished: count at NativeMethodAccessorImpl.java:0, took 2.533479 s
Number of Unique Hosts
40836
19/07/10 16:24:42 INFO datasources.FileSourceStrategy: Pruning directories with:
19/07/10 16:24:42 INFO datasources.FileSourceStrategy: Post-Scan Filters: (regexp_extract(value#0, \s(\d{3})\s, 1) = 404)
```

```
19/07/10 16:24:43 INFO scheduler.DAGScheduler: Job 9 finished: count at NativeMethodAcces
Total 404 HTTP codes
227101
19/07/10 16:24:44 INFO spark.SparkContext: Invoking stop() from shutdown hook
19/07/10 16:24:44 INFO cluster.YarnClientSchedulerBackend: Interrupting monitor thread
19/07/10 16:24:44 INFO cluster.YarnClientSchedulerBackend: Shutting down all executors
```

Spark Job :

App ID	App Name	Attempt ID	Started	Spark User	Last Updated	Event Log
application_1528714825862_134748	Module 5 SparkSession take		2019-07-10 16:24:05	edureka_524533	2019-07-10 16:24:11	Download

Spark Jobs (?)

User: edureka_524533
Total Uptime: 38 s
Scheduling Mode: FIFO
Completed Jobs: 10

▸ Event Timeline

Completed Jobs (10)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
9	count at NativeMethodAccessorImpl.java:0	2019/07/10 16:24:42	2 s	2/2	5/5
8	count at NativeMethodAccessorImpl.java:0	2019/07/10 16:24:39	3 s	3/3	205/205
7	showString at NativeMethodAccessorImpl.java:0	2019/07/10 16:24:35	4 s	2/2	5/5
6	count at NativeMethodAccessorImpl.java:0	2019/07/10 16:24:32	3 s	3/3	205/205
5	showString at NativeMethodAccessorImpl.java:0	2019/07/10 16:24:31	0.5 s	1/1 (1 skipped)	75/75 (4 skipped)
4	showString at NativeMethodAccessorImpl.java:0	2019/07/10 16:24:30	0.7 s	1/1 (1 skipped)	100/100 (4 skipped)
3	showString at NativeMethodAccessorImpl.java:0	2019/07/10 16:24:30	0.2 s	1/1 (1 skipped)	20/20 (4 skipped)
2	showString at NativeMethodAccessorImpl.java:0	2019/07/10 16:24:30	61 ms	1/1 (1 skipped)	4/4 (4 skipped)
1	showString at NativeMethodAccessorImpl.java:0	2019/07/10 16:24:28	2 s	2/2	5/5
0	count at NativeMethodAccessorImpl.java:0	2019/07/10 16:24:17	11 s	2/2	5/5