Triangle counting using Map/Reduce

From paper: Counting Triangles and the Curse of the Last Reducer – by, Siddharth Suri, Sergei Vassilvitskii[1]

Project under the guidance of : Prof. Sharma Chakravarthy
Project by: Rathna Sindura Chikkam
UTA Id: 1001553518
Technologies involved: Python, MapReduce, AWS, MRJob

Objective of the project :

To implement the NodeIterator++ algorithm proposed by the authors using both sequential and Map/Reduce version of algorithms and compare the efficiency of both the approaches.

Problem Statement:

Clustering coefficient of a node in a social network :

$$cc(v) = \frac{|\{(u,w) \in E \mid u \in \Gamma(v) \text{ and } w \in \Gamma(v)\}|}{\binom{d_v}{2}}.$$

- cc(v) can be computed by counting the number of triangles incident on the particular node in the network.
- When the network is huge, we would need a computing system that can still calculate the clustering coefficient in as much less time as possible without running into any memory issues. This calls for the use of Parallel Computing of the graph data, hence the use of Map/Reduce method of programming.

Data Sets Used:

UK Accidents graph data, Light and Weather data pulled from the same dataset.

** Few more data sets were made by pulling the data from Light and Weather data while testing the algorithms. Below is the datasets files index for the same:

Sl. No	File	Corresponding Edges File
1	Light_100	Light_100_edges.txt
2	Light_500	Light_500_edges.txt
3	Light_1000	Light_1000_edges.txt
4	Weather_100	Weather_100_edges.txt
5	Weather_500	Weather_500_edges.txt
6	Weather_1000	Weather_1000_edges.txt

Steps followed to implement this project:

NodeIterator++ algorithm - Sequential Version:

1. Implementation of the **nodeiterator++** algorithm can be found in file: **node_iterator_plus_v5.py** and it returns overall triangle count for a given input file – Light or Weather data files. It also prints the time taken for the algorithm to return the triangle count
2. This algorithm was initially tested with simple graph_sample*.txt files as provided in the code_and_data folder along with this report.
3. "**node_iterator_plus_v6.py**" is a copy of the above file that also prints the vertices and the count of triangles formed over those vertices.
4. **Technology used:** Spyder IDE was used to implement this algorithm in python 3.6.
5. **To run:** make sure that the input data file is in same folder as the **node_iterator_plus_v5.py** file, using Spyder IDE, compile the file by clicking the green play button at the top of the IDE. Now, using the console window, call the "**node_iterator_plus_v5**" function and pass the filename as its parameter.

Please refer the below example:

node_iterator_plus_v5("Light") (or) node_iterator_plus_v5("Light_100")

(or) node_iterator_plus_v5("Weather") (or) node_iterator_plus_v5("Weather_100")

MR-NodeIterator++ algorithm – Comparison with sequential:

1. Implementation of the **MR-nodeiterator++** algorithm can be found in 4 files: Mapper1.py, Reducer1.py, Mapper2.py, and Reducer2.py.
2. Before testing these algorithms in a Hadoop environment, I have tested these locally by combining them in a single script file, MR_Local_testing.py. The results of the mapper and reducer functions for each of the data file it was tested on are as below:

Dataset	Node Size	Edges Count	Sequential Results		MR Results	
			Triangle Count	Execution Time (Sequential) (in sec)	Triangle Count	Execution Time (MR) (in sec)
Light_100	100	61	3	0.140350103	15	0.00799942
	500	2109	10806	4.628014088	1555	1.084049702
	1000	8172	97889	188.3723097	6676	15.62057304
Weather	100	98	8	0.012014866	30	0.007998705
	500	2740	21877	9.43204093	2130	1.691673517
	1000	10381	178693	398.0261416	8691	25.2680366

3. Also, another important thing to notice here is the **Triangle Count being different for both versions** of the algorithm. Reason? MR Algorithm doesn't take into consideration of node degree of all the noticed triangles.
4. This analysis is made under the assumption that the local machine will act as one node/ cluster with 2 mappers and 2 reducers working on the datasets. The datasets are chosen to be with 100, 500 and 1000 vertices based on light_5000 or weather_5000 datasets, because, on local machines, more 1000 nodes takes quite a long time to complete execution.
5. Below is the analysis made on the triangles counted on each node for each data set:

Accident data – Light condition Comparison:

The below comparison is of top 20 values of light conditions data on 100, 500 and 1000 nodes whose triangle count on each node is provided. Turns out, no matter the amount of dataset taken to calculate the triangle count, Light Condition 1 seems to be the reason for accidents on majority of nodes. The triangle count data is obtained by using **nodeiterator++** algorithm (sequential version)

Light_100			**Light_500**			**Light_1000**		
Node	**Triangles on Node**	**Light Condition on Node**	**Node**	**Triangles on Node**	**Light Condition on Node**	**Node**	**Triangles on Node**	**Light Condition on Node**
57	2	4	497	5575	1	999	51761	1
59	2	1	499	5575	1	997	51726	4
61	2	1	489	5566	1	993	51725	1
63	2	1	491	5566	4	995	51725	6
65	2	1	493	5566	4	985	51698	1
67	2	1	495	5566	4	987	51698	1
69	2	1	487	5560	1	989	51698	1
71	2	1	485	5471	1	991	51698	1
73	2	1	479	5467	1	981	51628	1
75	2	1	481	5467	1	983	51628	1
77	2	4	483	5467	4	979	51623	1
79	2	1	500	5231	4	977	50186	1
81	2	4	498	5222	1	975	50177	4
83	2	1	496	5214	1	971	50176	1
85	2	1	494	5206	1	973	50176	4
87	2	1	469	5202	1	969	48985	1
89	2	1	471	5202	4	965	48950	1
91	2	1	473	5202	1	967	48950	1
93	2	4	475	5202	1	957	48949	4
95	2	1	477	5202	1	959	48949	1

Accident data – Weather condition Comparison:

Below is a comparison of top 20 values of weather data on 100, 500 and 1000 nodes whose triangle count on each node is provided. Turns out, no matter the amount of dataset taken to calculate the triangle count, Weather Condition 1 seems to be the reason for accidents on majority of nodes. The triangle count data is obtained by using **nodeiterator++** algorithm (sequential version)

Weather_100			Weather_500			Weather_1000		
Node	Triangles on Node	Weather Condition on node	Node	Triangles on Node	Weather Condition on node	Node	Triangles on Node	Weather Condition on node
5	6	1	500	11939	1	999	90284	1
7	6	1	498	11906	1	997	90236	1
9	6	1	496	11900	1	993	90235	1
11	6	1	494	11894	1	995	90235	8
13	6	1	484	11422	1	985	90212	1
15	6	1	486	11422	9	987	90212	2
17	6	1	488	11422	1	989	90212	1
19	6	1	490	11422	1	991	90212	1
21	6	2	492	11422	1	981	90152	1
23	6	1	480	10853	1	983	90152	1
25	6	1	482	10853	2	979	90143	1
27	6	1	478	10710	1	998	88409	1
29	6	1	476	10709	1	1000	88409	1
31	6	1	462	10708	1	996	88396	1
33	6	1	464	10708	1	988	88388	1
35	6	1	466	10708	1	990	88388	1
37	6	1	468	10708	1	992	88388	1
39	6	1	470	10708	8	994	88388	1
41	6	1	472	10708	1	980	87980	1
43	6	1	474	10708	1	982	87980	1

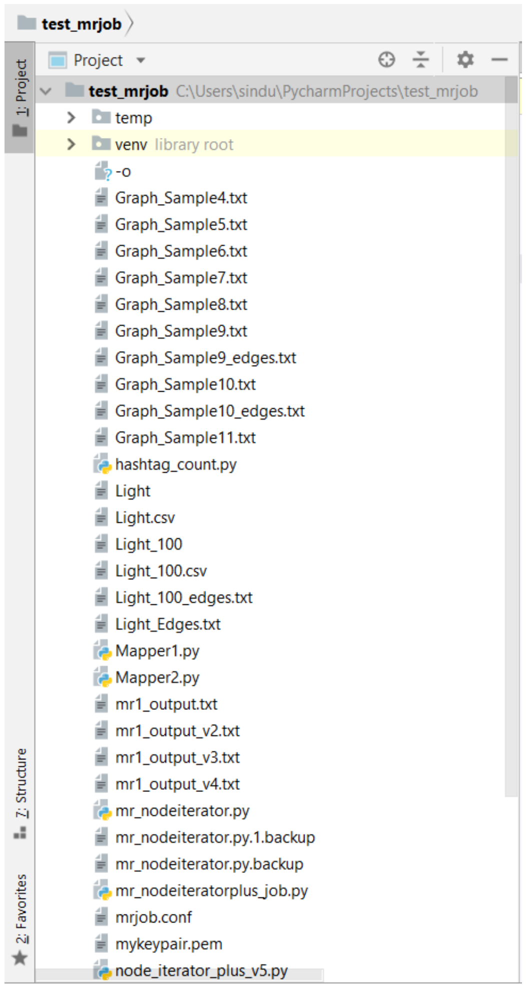
The below files contain the complete output of nodeiterator++ sequential algorithm, displayed in form of node, triangles on node.

Sl.No	File	Corresponding Output File
1	Light_100	Light_100_TriangleCount.csv
2	Light_500	Light_500_TriangleCount.csv
3	Light_1000	Light_1000_TriangleCount.csv
4	Weather_100	Weather_100_TriangleCount.csv
5	Weather_500	Weather_500_TriangleCount.csv
6	Weather_1000	Weather_1000_TriangleCount.csv

They can be found in either of the Code_and_Data folder or Plotting_Triangle_Count folder. These CSVs are generated in the file, node_iterator_plus_v7.py

MR – Version – attempt on local laptop using MRJob and AWS:

1. Implementation of the **MR-nodeiterator++** algorithm can be found in 4 files: Mapper1.py, Reducer1.py, Mapper2.py, and Reducer2.py.
2. Initially, these mappers and reducers are designed to run in them in a Hadoop environment using AWS. Due to lack of documentation on how to start a Hadoop cluster, SSH into it, and run the mappers and reducers directly using its command prompt, I ended finding a package called MRJob – which provides an interface between AWS Elastic MapReduce (EMR) and the code, right from your laptop.
3. In order to move forward with this approach, this YouTube link[14] was followed: https://www.youtube.com/watch?v=U6Cl7m6gqCI
4. Using PyCharm, I had set up a project on my system, which has the mrjob.conf – a configuration file of MRJob that has details about the AWS account – secret access key and ID, AWS SSH key pem file details, and other configuration details that helps in connecting to the AWS account. It also has the data files, and a temp directory to store the partitions that will be made when the mappers and reducers run using the input file.
5. Below is snapshot of how the project explorer looks like:



```
test_mrjob  C:\Users\sindu\PycharmProjects\test_mrjob
  temp
  venv  library root
  -o
  Graph_Sample4.txt
  Graph_Sample5.txt
  Graph_Sample6.txt
  Graph_Sample7.txt
  Graph_Sample8.txt
  Graph_Sample9.txt
  Graph_Sample9_edges.txt
  Graph_Sample10.txt
  Graph_Sample10_edges.txt
  Graph_Sample11.txt
  hashtag_count.py
  Light
  Light.csv
  Light_100
  Light_100.csv
  Light_100_edges.txt
  Light_Edges.txt
  Mapper1.py
  Mapper2.py
  mr1_output.txt
  mr1_output_v2.txt
  mr1_output_v3.txt
  mr1_output_v4.txt
  mr_nodeiterator.py
  mr_nodeiterator.py.1.backup
  mr_nodeiterator.py.backup
  mr_nodeiteratorplus_job.py
  mrjob.conf
  mykeypair.pem
  node_iterator_plus_v5.py
```

6. Then I import MRJob package, created a virtual environment to create the MRJob for the MR-nodeiterator++ algorithm. The MRJob implementation is in file**: mr_nodeiterator.py**
7. This implementation contains only first mapper and reducer of the algorithm, as I was trying to explore how the MRJob package would work. Also, the kind of structure the MRJob implementation maintains of any map/reduce job, it has one entry point to provide input and one exit point to provide output. Each map/reduce step will be mentioned in a separate MRStep to chain various mappers and reducers. But our algorithm has a

separate input which is a file containing only the edges data (e.g.: Light_100_edges.txt, Weather_500_edges.txt, etc.). Unfortunately, MRJob seemed to not support the aspect of providing another input to the second mapper. Hence, I had created only first part of the algorithm to test and chain the jobs separately using separate MRJobs if the first part of the algorithm implementation returns the expected output.

Execution and issues faced:

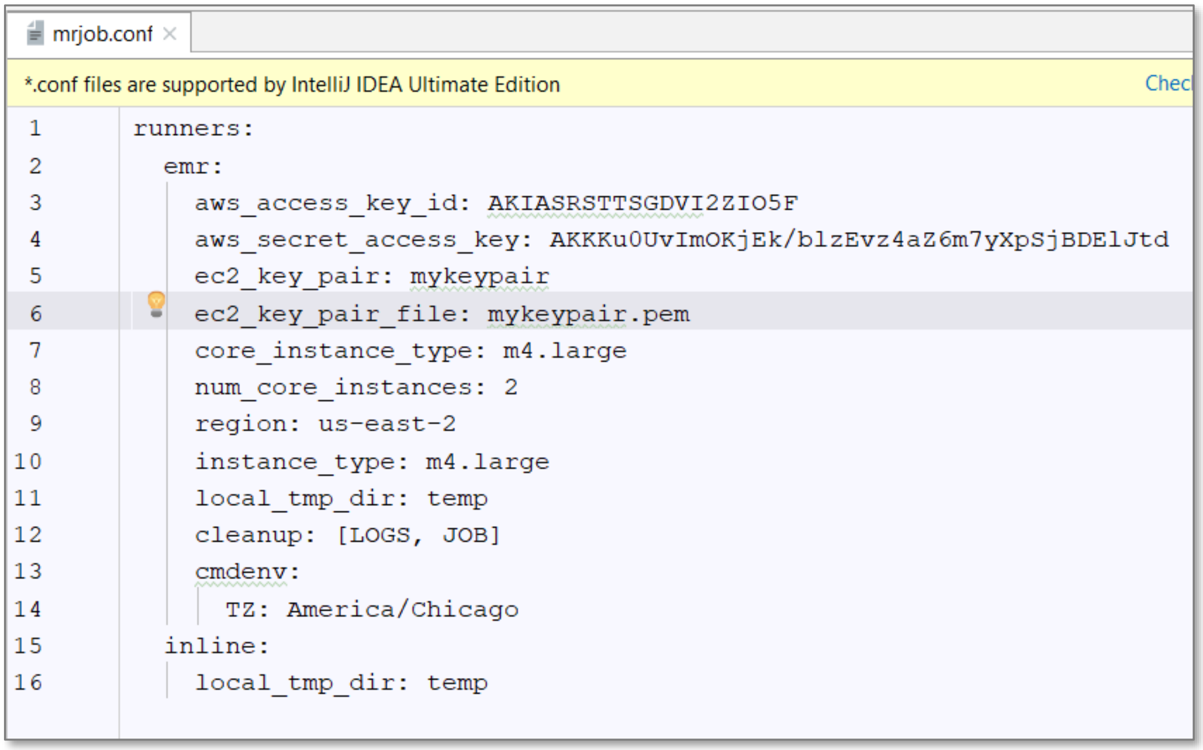
There are ultimately 2 commands that creates a cluster on Amazon's AWS EMR and runs the MRJob to save the output into one of the S3 storage buckets in AWS.

Below command will be used to create a cluster on AWS EMR using the command terminal in MRJob:

Command 1:

```
mrjob create-cluster --conf-path=mrjob.conf
```

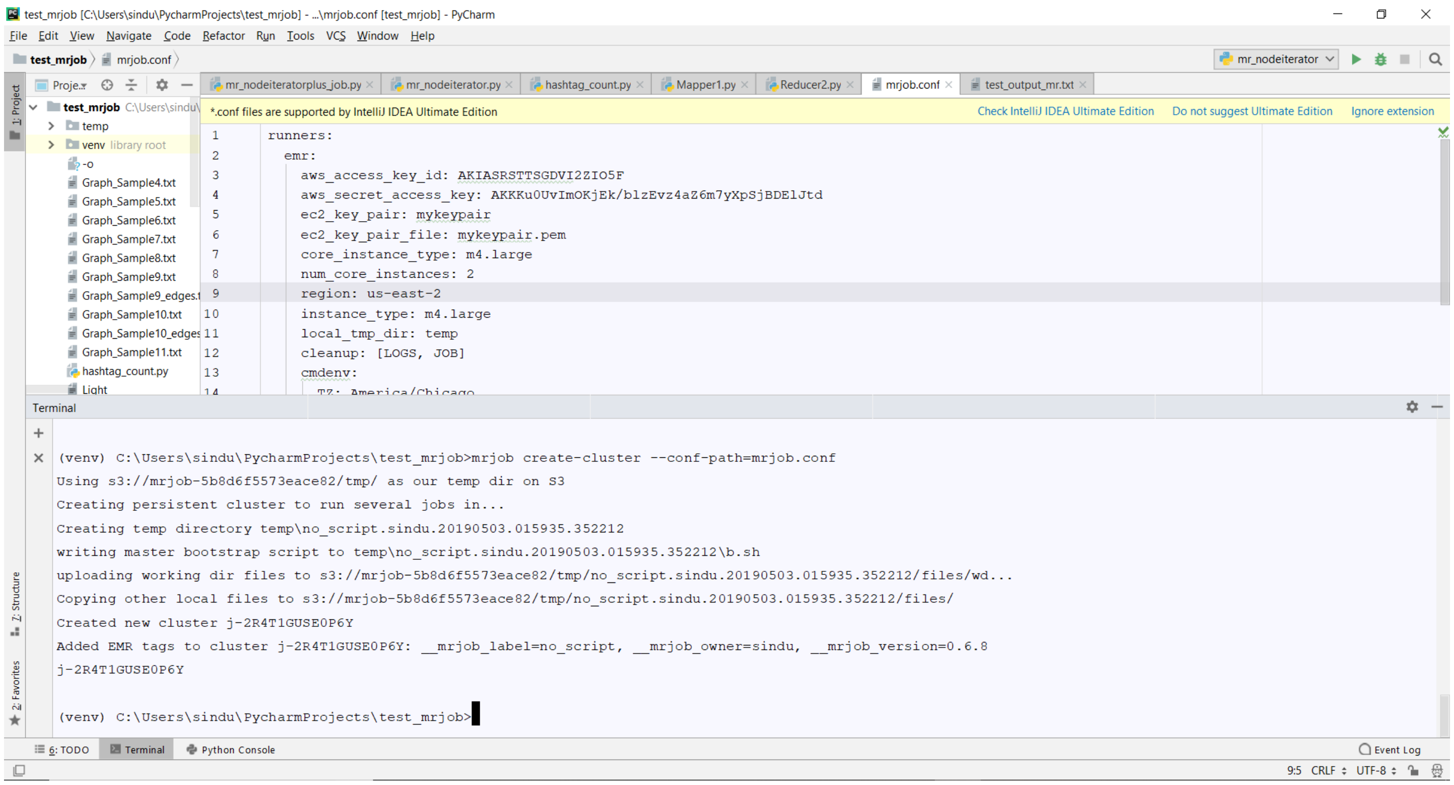
The mrjob.conf file (also included in the code_and_data folder) looks like below:



*.conf files are supported by IntelliJ IDEA Ultimate Edition

```
runners:
  emr:
    aws_access_key_id: AKIASRSTTSGDVI2ZIO5F
    aws_secret_access_key: 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
    ec2_key_pair: mykeypair
    ec2_key_pair_file: mykeypair.pem
    core_instance_type: m4.large
    num_core_instances: 2
    region: us-east-2
    instance_type: m4.large
    local_tmp_dir: temp
    cleanup: [LOGS, JOB]
    cmdenv:
      TZ: America/Chicago
  inline:
    local_tmp_dir: temp
```

As shown in the below screenshot, the cluster is created:

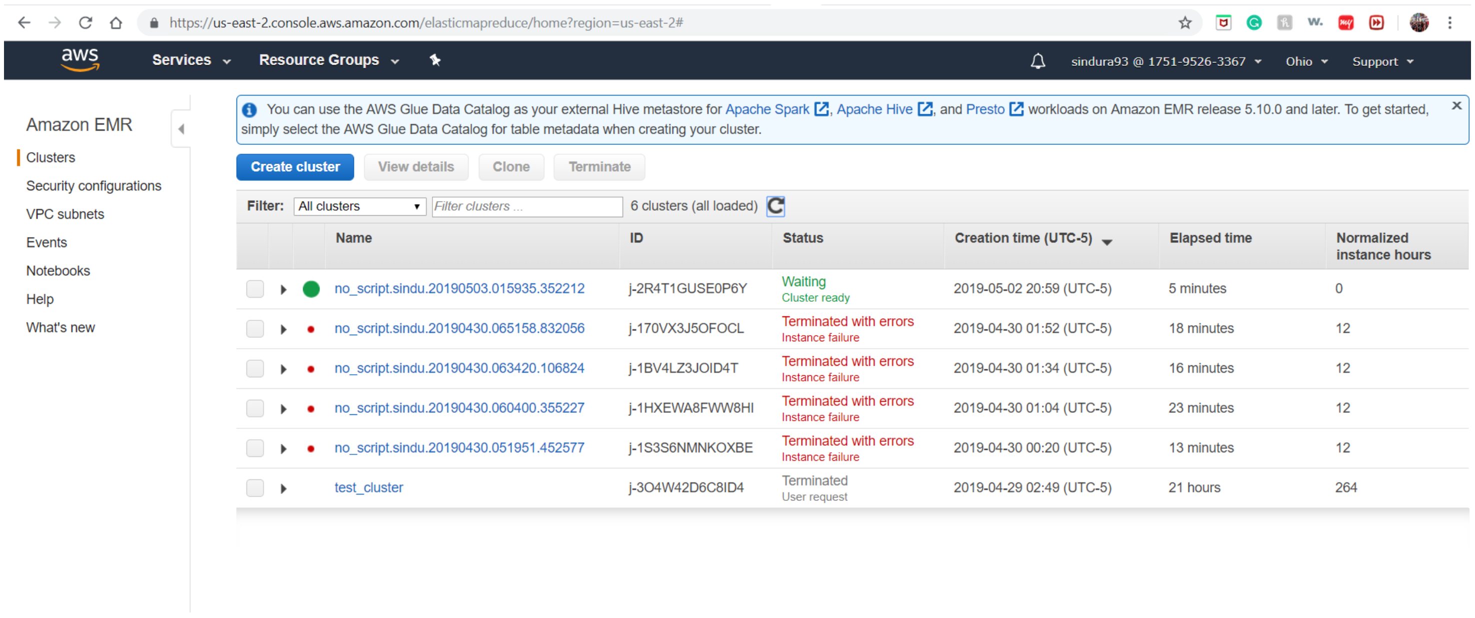


```
runners:
  emr:
    aws_access_key_id: AKIASRSTTSGDVI2ZIO5F
    aws_secret_access_key: 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
    ec2_key_pair: mykeypair
    ec2_key_pair_file: mykeypair.pem
    core_instance_type: m4.large
    num_core_instances: 2
    region: us-east-2
    instance_type: m4.large
    local_tmp_dir: temp
    cleanup: [LOGS, JOB]
    cmdenv:
      TZ: America/Chicago
```

```
(venv) C:\Users\sindu\PycharmProjects\test_mrjob>mrjob create-cluster --conf-path=mrjob.conf
Using s3://mrjob-5b8d6f5573eace82/tmp/ as our temp dir on S3
Creating persistent cluster to run several jobs in...
Creating temp directory temp\no_script.sindu.20190503.015935.352212
writing master bootstrap script to temp\no_script.sindu.20190503.015935.352212\b.sh
uploading working dir files to s3://mrjob-5b8d6f5573eace82/tmp/no_script.sindu.20190503.015935.352212/files/wd...
Copying other local files to s3://mrjob-5b8d6f5573eace82/tmp/no_script.sindu.20190503.015935.352212/files/
Created new cluster j-2R4T1GUSE0P6Y
Added EMR tags to cluster j-2R4T1GUSE0P6Y: __mrjob_label=no_script, __mrjob_owner=sindu, __mrjob_version=0.6.8
j-2R4T1GUSE0P6Y

(venv) C:\Users\sindu\PycharmProjects\test_mrjob>
```

Below picture shows that the cluster, with the id mentioned at the last of the command window in the above image, is created:



https://us-east-2.console.aws.amazon.com/elasticmapreduce/home?region=us-east-2#

You can use the AWS Glue Data Catalog as your external Hive metastore for Apache Spark, Apache Hive, and Presto workloads on Amazon EMR release 5.10.0 and later. To get started, simply select the AWS Glue Data Catalog for table metadata when creating your cluster.

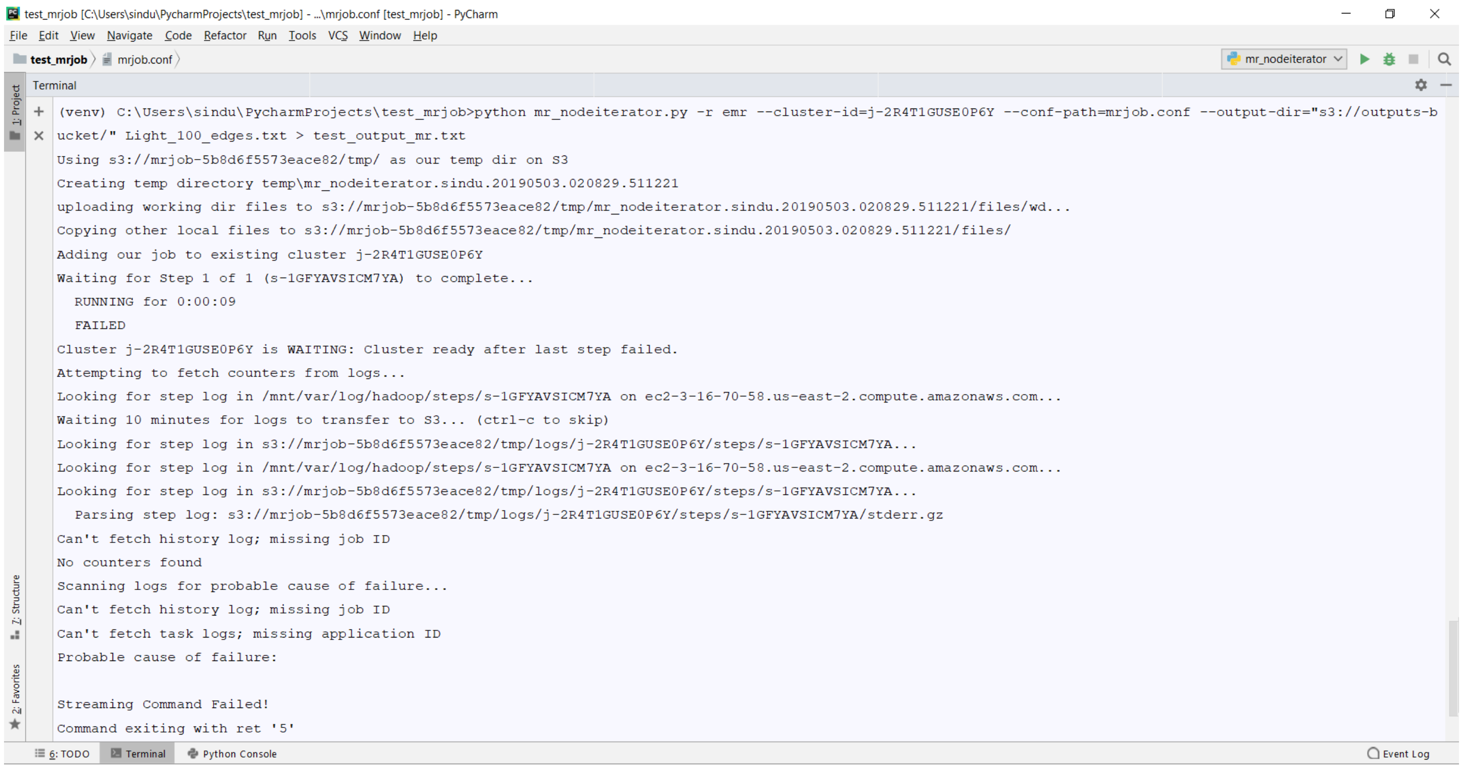
Filter: All clusters — 6 clusters (all loaded)

	Name	ID	Status	Creation time (UTC-5)	Elapsed time	Normalized instance hours
●	no_script.sindu.20190503.015935.352212	j-2R4T1GUSE0P6Y	Waiting Cluster ready	2019-05-02 20:59 (UTC-5)	5 minutes	0
●	no_script.sindu.20190430.065158.832056	j-170VX3J5OFOCL	Terminated with errors Instance failure	2019-04-30 01:52 (UTC-5)	18 minutes	12
●	no_script.sindu.20190430.063420.106824	j-1BV4LZ3JOID4T	Terminated with errors Instance failure	2019-04-30 01:34 (UTC-5)	16 minutes	12
●	no_script.sindu.20190430.060400.355227	j-1HXEWA8FWW8HI	Terminated with errors Instance failure	2019-04-30 01:04 (UTC-5)	23 minutes	12
●	no_script.sindu.20190430.051951.452577	j-1S3S6NMNKOXBE	Terminated with errors Instance failure	2019-04-30 00:20 (UTC-5)	13 minutes	12
	test_cluster	j-3O4W42D6C8ID4	Terminated User request	2019-04-29 02:49 (UTC-5)	21 hours	264

Now we use the below command to run the MRJob:

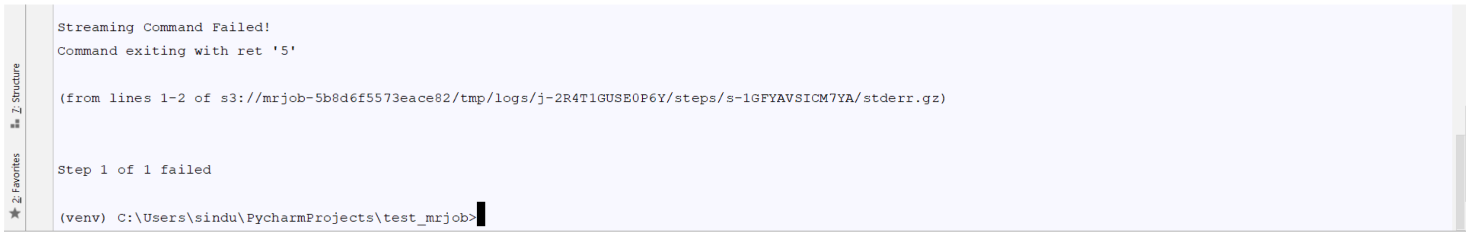
<u>Command 2:</u>

python mr_nodeiterator.py -r emr --cluster-id=j-2R4T1GUSE0P6Y --conf-path=mrjob.conf --output-dir="s3://outputs-bucket/" Light_100_edges.txt > test_output_mr.txt


test_mrjob [C:\Users\sindu\PycharmProjects\test_mrjob] - ...\mrjob.conf [test_mrjob] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
test_mrjob
mrjob.conf
mr_nodeiterator
Terminal
1: Project
7: Structure
2: Favorites
6: TODO
Terminal
Python Console
Event Log

```
(venv) C:\Users\sindu\PycharmProjects\test_mrjob>python mr_nodeiterator.py -r emr --cluster-id=j-2R4T1GUSE0P6Y --conf-path=mrjob.conf --output-dir="s3://outputs-b
ucket/" Light_100_edges.txt > test_output_mr.txt
Using s3://mrjob-5b8d6f5573eace82/tmp/ as our temp dir on S3
Creating temp directory temp\mr_nodeiterator.sindu.20190503.020829.511221
uploading working dir files to s3://mrjob-5b8d6f5573eace82/tmp/mr_nodeiterator.sindu.20190503.020829.511221/files/wd...
Copying other local files to s3://mrjob-5b8d6f5573eace82/tmp/mr_nodeiterator.sindu.20190503.020829.511221/files/
Adding our job to existing cluster j-2R4T1GUSE0P6Y
Waiting for Step 1 of 1 (s-1GFYAVSICM7YA) to complete...
  RUNNING for 0:00:09
  FAILED
Cluster j-2R4T1GUSE0P6Y is WAITING: Cluster ready after last step failed.
Attempting to fetch counters from logs...
Looking for step log in /mnt/var/log/hadoop/steps/s-1GFYAVSICM7YA on ec2-3-16-70-58.us-east-2.compute.amazonaws.com...
Waiting 10 minutes for logs to transfer to S3... (ctrl-c to skip)
Looking for step log in s3://mrjob-5b8d6f5573eace82/tmp/logs/j-2R4T1GUSE0P6Y/steps/s-1GFYAVSICM7YA...
Looking for step log in /mnt/var/log/hadoop/steps/s-1GFYAVSICM7YA on ec2-3-16-70-58.us-east-2.compute.amazonaws.com...
Looking for step log in s3://mrjob-5b8d6f5573eace82/tmp/logs/j-2R4T1GUSE0P6Y/steps/s-1GFYAVSICM7YA...
  Parsing step log: s3://mrjob-5b8d6f5573eace82/tmp/logs/j-2R4T1GUSE0P6Y/steps/s-1GFYAVSICM7YA/stderr.gz
Can't fetch history log; missing job ID
No counters found
Scanning logs for probable cause of failure...
Can't fetch history log; missing job ID
Can't fetch task logs; missing application ID
Probable cause of failure:

Streaming Command Failed!
Command exiting with ret '5'
```



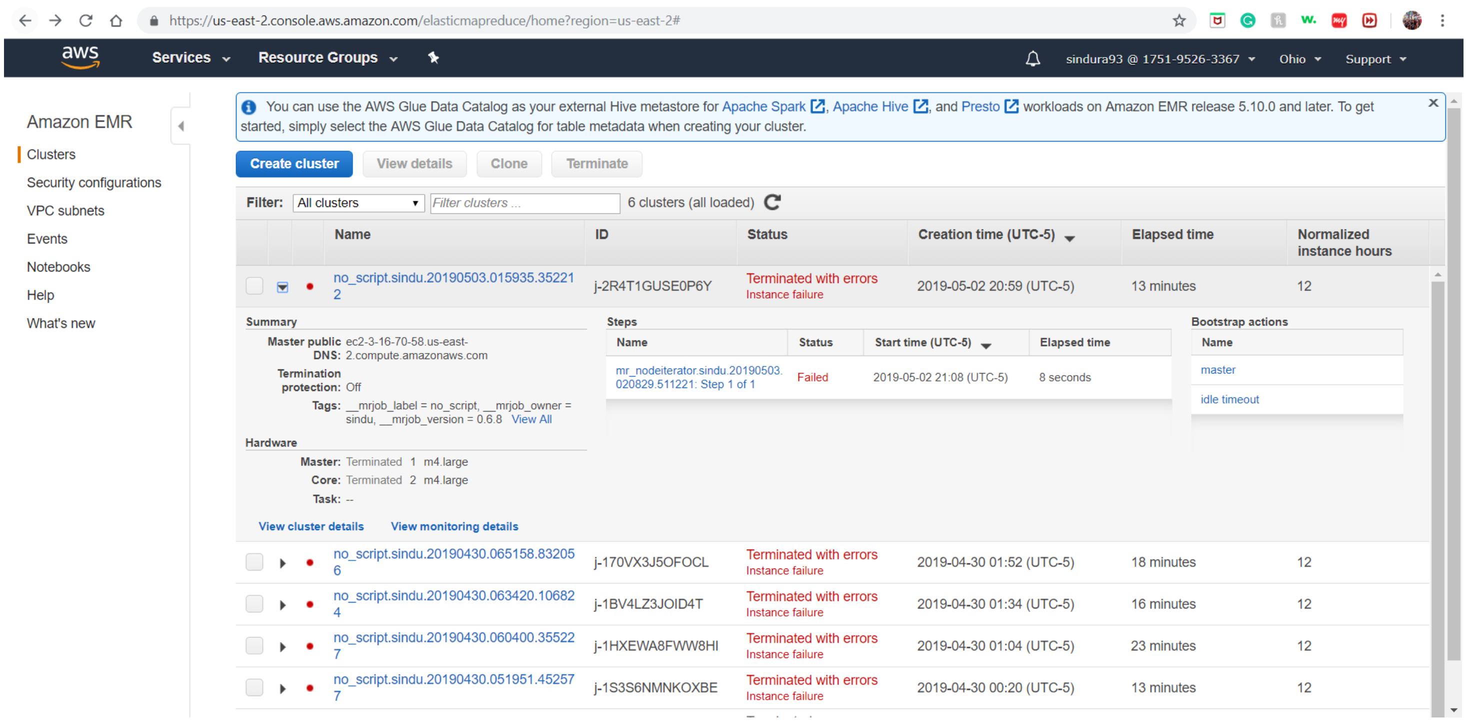
```
Streaming Command Failed!
Command exiting with ret '5'

(from lines 1-2 of s3://mrjob-5b8d6f5573eace82/tmp/logs/j-2R4T1GUSE0P6Y/steps/s-1GFYAVSICM7YA/stderr.gz)


Step 1 of 1 failed

(venv) C:\Users\sindu\PycharmProjects\test_mrjob>
```

As mentioned in the command window, the job fails to run every single time and created cluster terminates on itself as shown in the below image:



You can use the AWS Glue Data Catalog as your external Hive metastore for Apache Spark, Apache Hive, and Presto workloads on Amazon EMR release 5.10.0 and later. To get started, simply select the AWS Glue Data Catalog for table metadata when creating your cluster.

Name	ID	Status	Creation time (UTC-5)	Elapsed time	Normalized instance hours
no_script.sindu.20190503.015935.352212	j-2R4T1GUSE0P6Y	Terminated with errors Instance failure	2019-05-02 20:59 (UTC-5)	13 minutes	12
no_script.sindu.20190430.065158.832056	j-170VX3J5OFOCL	Terminated with errors Instance failure	2019-04-30 01:52 (UTC-5)	18 minutes	12
no_script.sindu.20190430.063420.106824	j-1BV4LZ3JOID4T	Terminated with errors Instance failure	2019-04-30 01:34 (UTC-5)	16 minutes	12
no_script.sindu.20190430.060400.355227	j-1HXEWA8FWW8HI	Terminated with errors Instance failure	2019-04-30 01:04 (UTC-5)	23 minutes	12
no_script.sindu.20190430.051951.452577	j-1S3S6NMNKOXBE	Terminated with errors Instance failure	2019-04-30 00:20 (UTC-5)	13 minutes	12

Summary

Master public DNS: ec2-3-16-70-58.us-east-2.compute.amazonaws.com
Termination protection: Off
Tags: __mrjob_label = no_script, __mrjob_owner = sindu, __mrjob_version = 0.6.8 View All

Hardware

Master: Terminated 1 m4.large
Core: Terminated 2 m4.large
Task: --

View cluster details View monitoring details

Steps

Name	Status	Start time (UTC-5)	Elapsed time
mr_nodeiterator.sindu.20190503.020829.511221: Step 1 of 1	Failed	2019-05-02 21:08 (UTC-5)	8 seconds

Bootstrap actions

Name
master
idle timeout

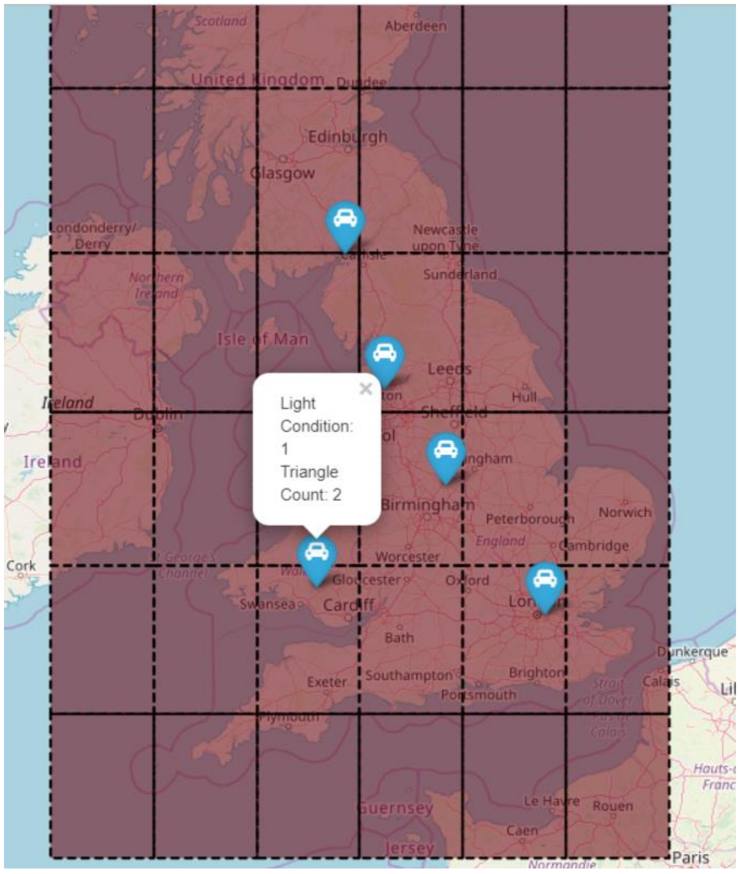
There are barely any fixes available on the internet to know why this is happening or how to fix this issue with the AWS EMR.

Top 5 Triangle Count Visualizations

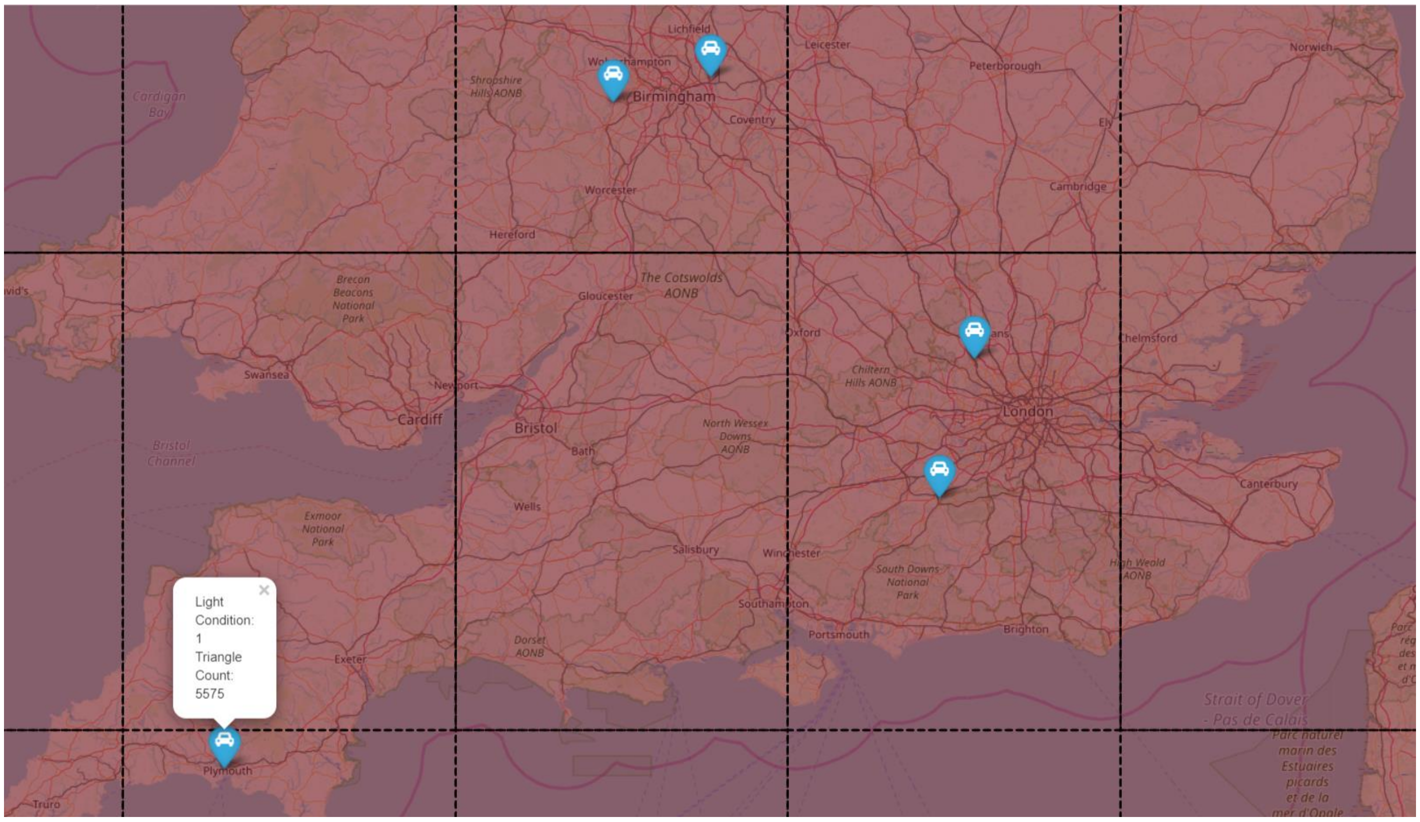
Used Folium package available on python platform to analyze the top 5 nodes with highest triangles incident on them, based on light conditions. The below are the results. All these visualizations are generated as html pages to zoom-in and inspect each node. They are submitted along with other deliverables in the folder **'Plotting_Triangle_Count'**. For based on light condition datasets, code for the below visualizations can be found in "**Top 5 - Based on Light Conditions - V2.ipynb**". For based on light condition datasets, code for the below visualizations can be found in "**Top 5 - Based on Weather Conditions - V2.ipynb**". Below are the CSVs, as mentioned above, that were used to create these visualizations:

Sl. No	Dataset	Corresponding Triangle Count File
1	Light_100	Light_100_TriangleCount.csv
2	Light_500	Light_500_TriangleCount.csv
3	Light_1000	Light_1000_TriangleCount.csv
4	Weather_100	Weather_100_TriangleCount.csv
5	Weather_500	Weather_500_TriangleCount.csv
6	Weather_1000	Weather_1000_TriangleCount.csv

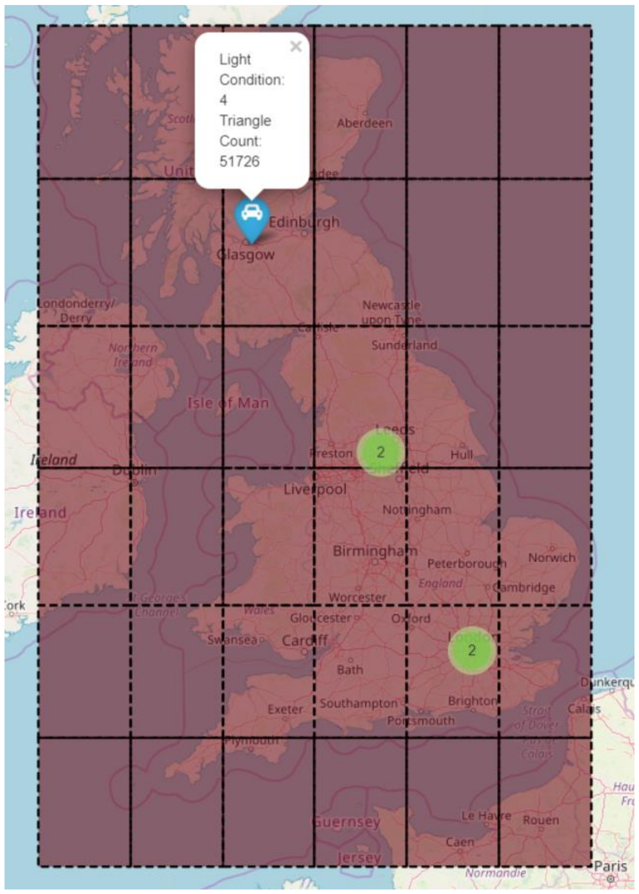
Based on Light Datasets



1. Above is the visualization, made for top 5 triangles incident on nodes of 100 nodes data set, Light_100

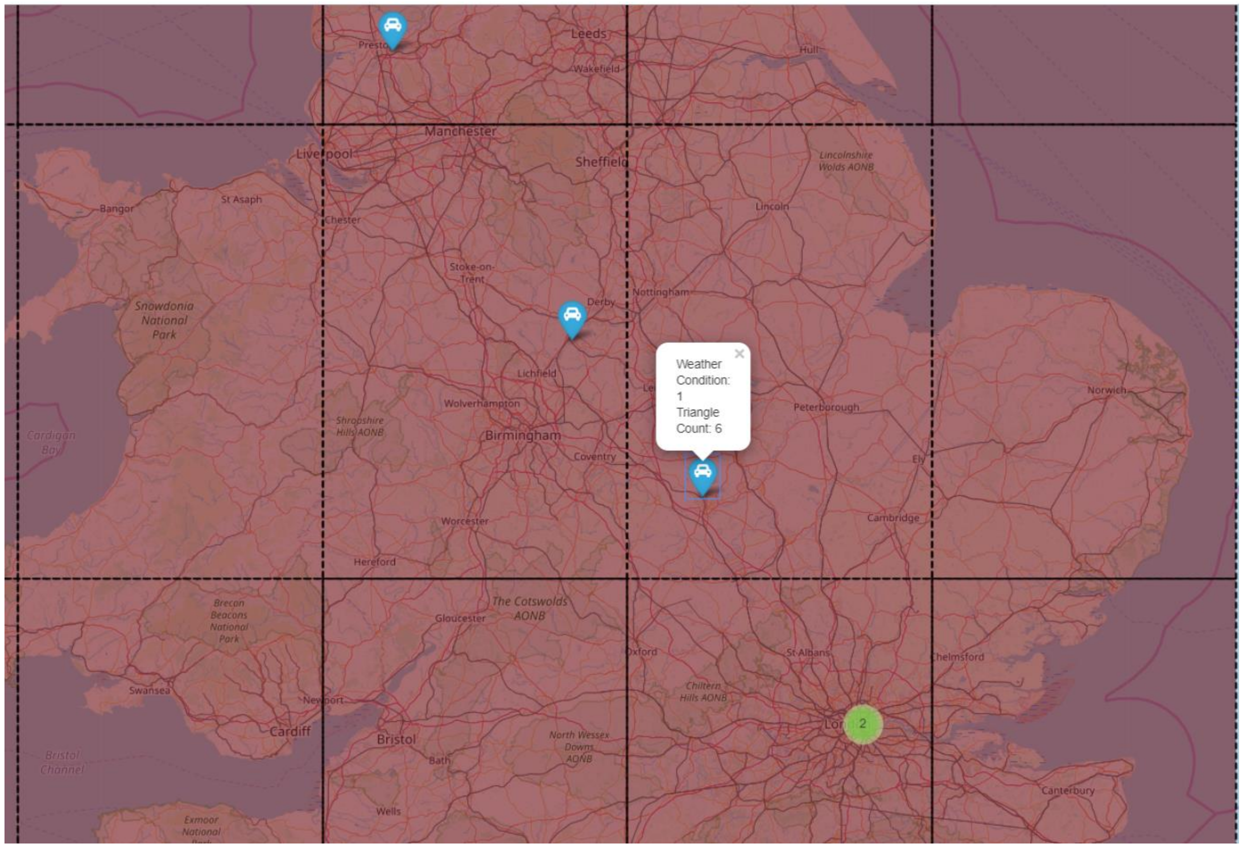


2. Above is the visualization, made for top 5 triangles incident on nodes of 500 nodes data set, Light_500

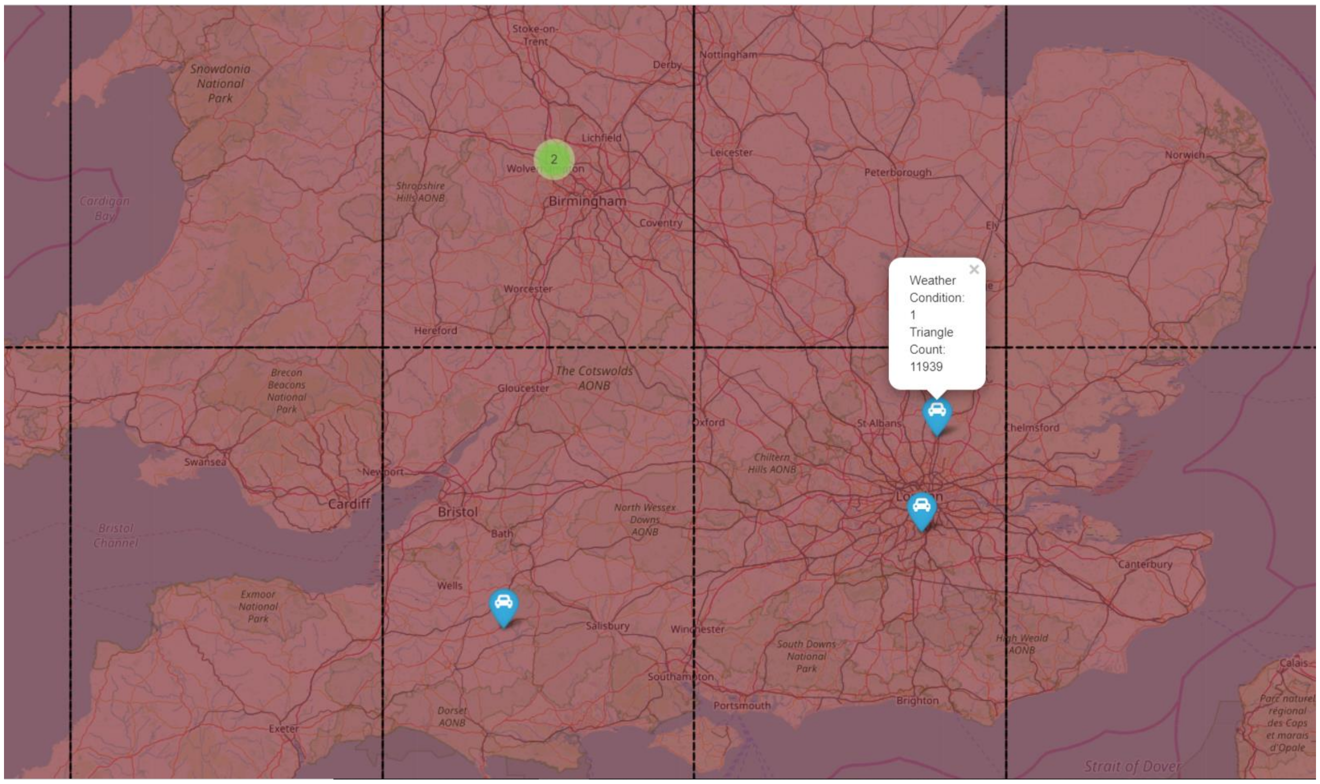


3. Above is the visualization, made for top 5 triangles incident on nodes of 1000 nodes data set, Light_1000

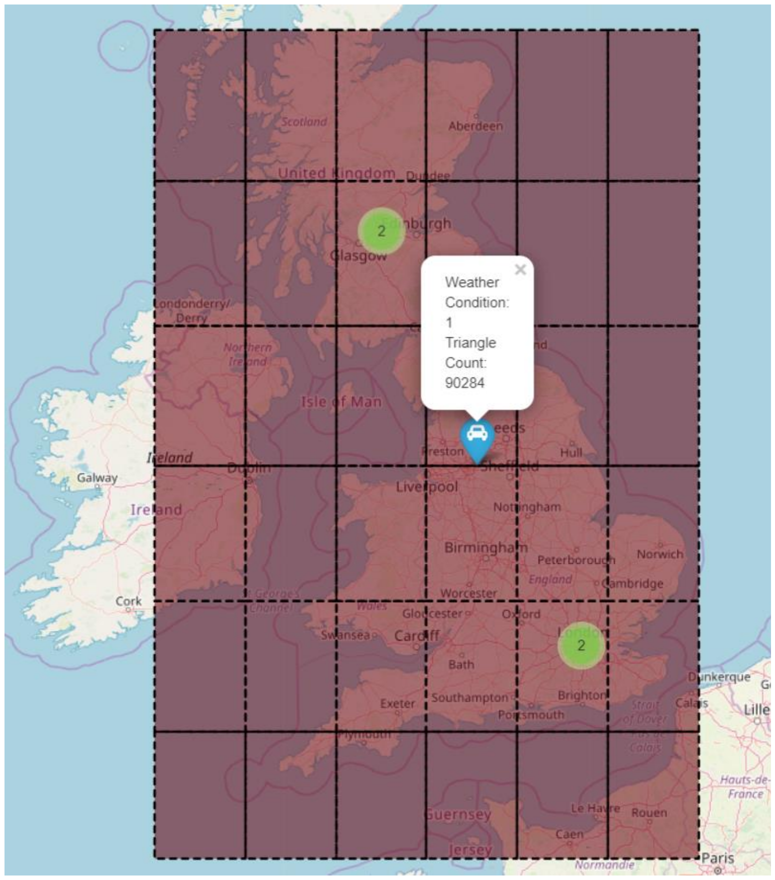
Based on Weather Datasets



1. Above is the visualization, made for top 5 triangles incident on nodes of 100 nodes data set, Weather_100



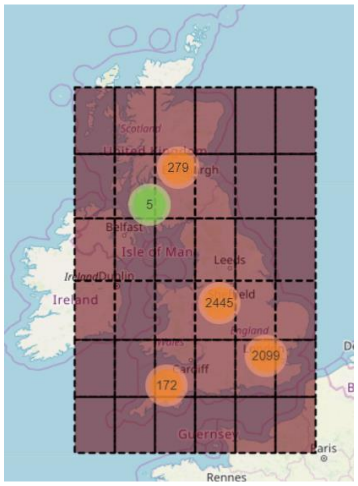
2. Above is the visualization, made for top 5 triangles incident on nodes of 500 nodes data set, Weather_500



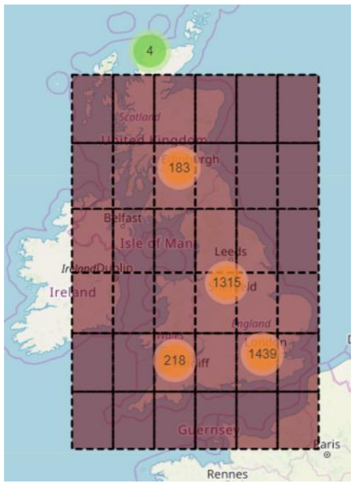
3. Above is the visualization, made for top 5 triangles incident on nodes of 1000 nodes data set, Weather_1000

Visual comparisons created to view light and weather data of accident data

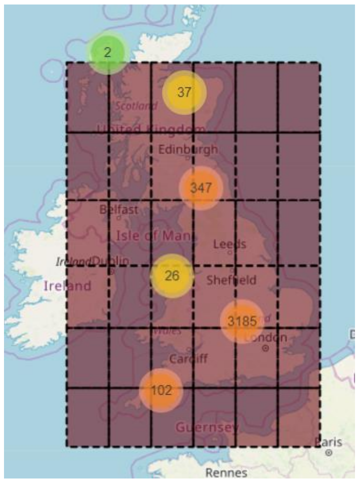
Comparisons are made by using the UK accidents data to see geographically which areas of UK are affected by accidents due to which of Light or Weather conditions. Used Folium package available on python platform to analyze how much of UK is affected by either of poor light or weather conditions. The below are the results:



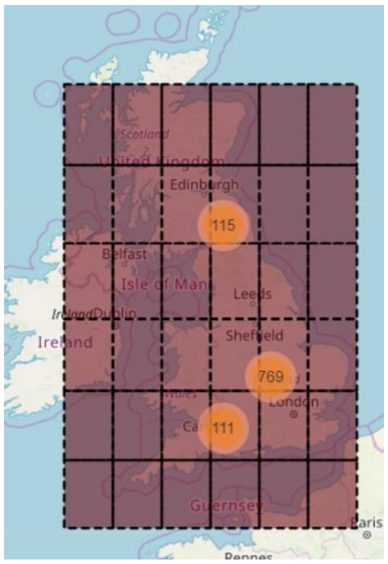
All the accident data



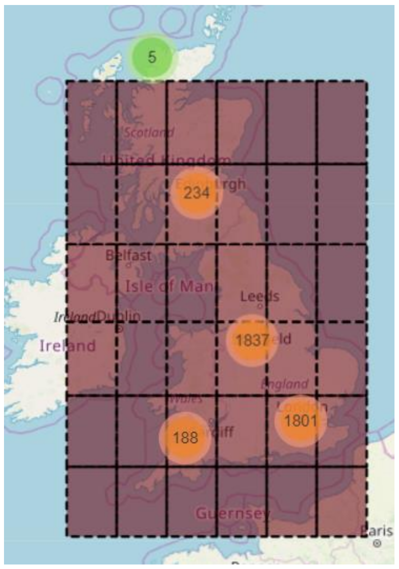
Data with same Light and Weather condition rating



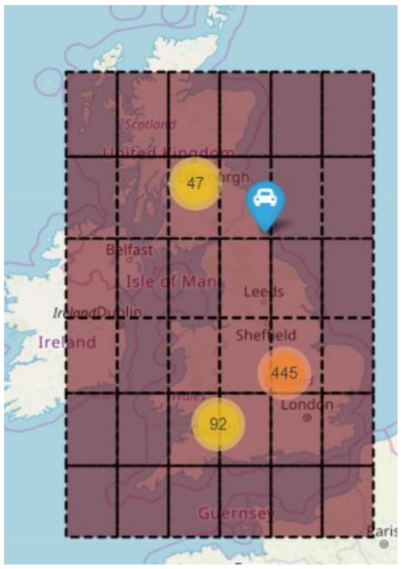
Data with 1st worst Light condition rating



Data with 2nd worst
Light condition rating



Data with 1st worst
Weather condition rating



Data with 2nd worst
Weather condition rating

These snapshots of the visualizations are taken from HTML files generated using folium package. Code can be found in an IPython Notebook: **advdb-plotting.ipynb**

Code and the visualization HTMLs generated, can be found in the folder, "Plotting_Triangle_Count"

Final Analysis on visualizing the triangle count:

- For all the light_100, light_500, light_1000 datasets, at least 90% of the accidents have occurred due to the Light Condition = 1, which means during the day light – in day time.
- Similarly, for all the weather_100, weather_500, weather_1000 datasets, at least 90% of the accidents have occurred due to the weather Condition = 1, which means when the weather was perfectly fine, with no high winds at all!
- Interesting things to note from each of the Light datasets is that:
  - For Light_100 dataset: the top 5 nodes with triangles incident on them appear in mid to southern region of UK.
  - For Light_500 dataset: the top 5 nodes with highest triangles incident on them appear in far southern region of UK, with highest count in Plymouth region (triangle_count = 5575, light_condition = 1)
  - For Light_1000 dataset: the top 5 nodes with highest triangles incident on them appear in surrounding areas of Glasgow, Sheffield and London areas, with highest being in Manchester close by Sheffield area (triangle_count = 51761, light_condition = 1)
- Interesting things to note from each of Weather datasets is that:
  - For Weather_100 dataset: the top 5 nodes with triangles incident on them appear in the southern region of UK, with 2 nodes within London (triangle_count = 6*2, weather_condition = 1)

- For Weather_500 dataset: the top 5 nodes with highest triangles incident on them appear in far southern region of UK, with highest count in London region, close to Waltham Cross (triangle_count = 11939, weather_condition = 1)
- For Weather_1000 dataset: the top 5 nodes with highest triangles incident on them appear in surrounding areas of Glasgow, Sheffield and London areas, with highest being in Manchester close by Sheffield area (triangle_count = 90284, weather_condition = 1)

➢ For both the light and weather data set, also the overall UK accidents dataset, they all seem to have high number of accidents in the mid-east to south and south east region of UK, as clusters shown in the graphs tend to form a strong hub.

Project Conclusion:

As per the goal of the project, I was able to create and run both the algorithms locally and the results seemed impressive. I was able to run only 100, 500, 1000 nodes worth of data as the MRJobs created to run the mappers and reducers could not instantiate and run an AWS Elastic MapReduce cluster properly. The cluster kept dying with a static error that had no explanation – please refer to error in the screenshots above.

It took quite some time to understand that the algorithms are supposed to provide different outputs. Whether using MRJob or trying to implement the Mappers and Reducers in GCP or AWS, there isn't much documentation available to know how to do them. Referenced articles were of great help!

References:

1. Suri, Siddharth & Vassilvitskii, Sergei. (2011). Counting triangles and the curse of the last reducer. Proceedings of the 20th International Conference on World Wide Web, WWW 2011. 607-614. 10.1145/1963405.1963491.
2. https://csci8980bigdataalgo.files.wordpress.com/2013/09/lecture_01_sergei-vassilvitskii_intro-and-model.pdf
3. http://archive.dimacs.rutgers.edu/Workshops/Parallel/slides/suri.pdf
4. https://www2.cs.duke.edu/courses/fall15/compsci590.4/assignment3/MapReduceTutorial.pdf
5. https://snap.stanford.edu/class/cs341-2013/downloads/amazon-emr-tutorial.pdf
6. http://megasthenis.github.io/repository/Hadoop-WNCG-Intro.pdf
7. https://en.wikipedia.org/wiki/Clustering_coefficient
8. https://en.wikipedia.org/wiki/Social_network
9. https://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/
10. https://www.jpytr.com/post/analysinggeographicdatawithfolium/
11. AWS Elastic Map Reduce using MRJob: https://www.youtube.com/watch?v=U6Cl7m6gqCI