Execution

1. Setup: Load Java and download Spark binary

Load JAVA (jdk)

```
smodule load openjdk/1.8.0_202-b08
```

Download Spark binary in home directory

```
wget https://ftp.wayne.edu/apache/spark/spark-2.4.7/spark-2.4.7-bin-hadoop2.7.tgz
tar xvf spark-2.4.7-bin-hadoop2.7.tgz
mv spark-2.4.7-bin-hadoop2.7 spark
```

2. Copy the required folder (PA3) to Discovery HPC cluster
3. Enter into PA3 and run below commands

To run kmeans.py

```
srun -n 1 ~/spark/bin/spark-submit kmeans.py data.txt means.txt
```

To run trianglecounting.py

```
srun -n 1 ~/spark/bin/spark-submit trianglecounting.py p2p-Gnutella06.txt
```

To run trianglecounting_with_nodes.py

```
srun -n 1 ~/spark/bin/spark-submit trianglecounting_with_nodes.py p2p-Gnutella06.txt
```

Code difference in trianglecounting.py and trianglecounting_with_nodes.py

The algorithm used in trianglecounting_with_nodes.py is slightly different from trianglecounting.py. In trianglecounting_with_nodes.py, the nodes with no outgoing neighbours are not considered or ignored in traingle counting and are directly set to 0 count in the final step using a Map Reduce using the Nodes List collected in the begining. However, the trianglecounting.py implementation computes the triangle count for each vertes without using the nodes list. So all the edge cases (or conditions) are handled in the mapToCluster function as well the countOnCluster function. The need to use the final MapReduce to coun the traingles with the nodes list is completely avoided or unnecessary.