Problem Analysis:

1. *Cardinality of Path2 (MapReduce Pseudocode):*

```
int counter = 0

map(String input_key, String input_value):
        // input_key: document name
        // input_value: document contents
        rowValues = value.split(“,”)
        follower, followed = rowValues[0], rowValues=[1]

        outkey = follower
        outvalue = “O” + followed
        EmitIntermediate(outkey, outvalue);

        outkey = followed
        outvalue = “I” + follower
        EmitIntermediate(outkey, outvalue);

reduce(String intermediate_key, Iterator intermediate_values):
        // intermediate _key: userID
        // intermediate_values: a list of followers count associated with userID
        outList = []
        inList = []
        for each val in intermediate_values:
                if val[0] == ‘O’:
                        outList.add(val[1:])
                else if val[0] == ‘I’:
                        inList.add(val[1:])
        int outSize = outList.size()
        int inSize = inList.size()
        counter += outSize * inSize
```

2. *Cardinality and Volume Estimates for the two join steps using RS-join vs. Rep-join*
   Note: Merged the two steps into one for Rep join

	RS join input	**RS join shuffled**	**RS join output**	**Rep join input**	**Rep join file cache**	**Rep join output**
Step 1 (join of Edges with itself)	*Cardinality:* 85331845 *Volume:* 1319281825 bytes	*Cardinality:* 170663690 *Volume:* 2638563650 bytes	*Cardinality:* 201519176792 *Volume: 1007595883960* bytes	-	-	-
Step 2 (join of Path2 with Edges)	*Cardinality:* 201604508637 *Volume:* 1008915165785 bytes	*Cardinality:* 201604508637 *Volume:* 1008915165785 bytes	*Cardinality:* 201604508637 *Volume:* 1008915165785 bytes	*Cardinality:* 85331845 *Volume:* 1319281825 bytes	*Cardinality:* 85331845 *Volume:* 1319281825 bytes	*Cardinality:* 201604508637 *Volume:* 1008915165785 bytes

3. *Cardinality of Path2:*

PATH2_COUNT = 201519176792, exact cardinality obtained using the MapReduce program from above.

Join Implementation:

RS Join (MapReduce Pseudocode):

```
int MAX_FILTER = 50000; int TRIANGLE_COUNT = 0

map1(String input_key, String input_value):
        // input_key: document name
        // input_value: document contents
        rowValues = value.split(“,”)
        follower, user = rowValues[0], rowValues=[1]
        if follower < MAX_FILTER and user < MAX_FILTER:
                outkey = user
                outvalue = “F” + follower
                EmitIntermediate(outkey, outvalue);
                outkey = follower
                outvalue = “T” + user
                EmitIntermediate(outkey, outvalue);

reduce1(String intermediate_key, Iterator intermediate_values):
        // intermediate _key: userID
        // intermediate_values: a list of followers count associated with userID
        fromList = []
        toList = []
        populate(fromList, toList, intermediate_values)
        if fromList not empty and toList not empty:
                for each f in fromList:
                        for each t in toList:
                                outkey = f
                                outvalue = “R” + t
                                EmitIntermediate(outkey, outvalue);

map2(String input_key, String input_value):
        // input_key: document name
        // input_value: document contents
        rowValues = value.split(“,”)
        follower, user = rowValues[0], rowValues=[1]
        if user[0] == ‘R’:
                outkey = follower
                outvalue = “T” + user
        else:
                outkey = user
                outvalue = “F” + follower
        EmitIntermediate(outkey, outvalue);

reduce2(String intermediate_key, Iterator intermediate_values):
        // intermediate _key: userID
        // intermediate_values: a list of followers count associated with userID
        fromList = []
        toList = []
        populate(fromList, toList, intermediate_values)
        for each searchVal in toList:
                if searchVal in fromList:
                        TRIANGLE_COUNT += 1

populate(fromList, toList, intermediate_values):
        for each val in intermediate_values:
        if val[0] == ‘F’:
                fromList.add(val[1:])
        else if val[0] == ‘T’:
                toList.add(val[1:])
```

Rep Join (MapReduce Pseudocode):

```
int MAX_FILTER = 50000; int TRIANGLE_COUNT = 0

class Mapper:
        H = new hashMap
        setup() :
                // Load data set from the distributed file cache into H
                for each tuple row in Distributed File Cache:
                        follower = row[0]
                        user = row[1]
                        if follower < MAX_FILTER and user < MAX_FILTER:
                                if user in H:
                                        H.get(user).add(follower)
                                else:
                                        hashSet = new hashSet
                                        hashSet.add(follower)
                                        H.put(user, hashSet)

        map(String input key, String input_value):
                rowValues = value.split(“,”)
                follower, user = rowValues[0], rowValues=[1]
                if follower < MAX_FILTER and user < MAX_FILTER:
                        connections = H.get(follower)
                        if connections is not empty:
                                for each c in connections:
                                        connectionsOfC = H.get(c)
                                        if connectionsOfC is not empty & connectionsOfC contains user:
                                                TRIANGLE_COUNT += 1
```

Results with the corresponding MAX value on AWS on the full Twitter edges dataset:

Configuration	**Small Cluster Result**	**Large Cluster Result**
RS-join, MAX = `50000` `(aws)`	Running time: 35 minutes Triangle count: 12029907	Running time: 23 minutes, Triangle count: 12029907
Rep-join, MAX = `10000` `(local machine)`	Running time: 17 minutes Triangle count: 520296	Running time: 17 minutes Triangle count: 520296

Side Notes:

No output files have been created for this assignment, the triangle count is reported at the bottom of the respective syslog.

Please take a look at the log file, *syslog_rep_join.txt* generated from running the Rep-join code on AWS. Running the Rep-join on AWS returns the TRIANGLE_COUNT = 0, one of the reasons might be how distributed caching works on AWS. I tried looking at various documentations online, and maybe using the -cacheFile flag when setting up the EMR cluster could have been a possible solution. Nevertheless, in order to verify if the MapReduce program was functional, I ran the code for Rep-join on my local machine. The screenshot below illustrates the result with MAX_FILTER = 10000.

```
20/02/14 00:58:56 INFO mapreduce.Job: Counters: 16
        File System Counters
                FILE: Number of bytes read=80270308389
                FILE: Number of bytes written=53203820680
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
        Map-Reduce Framework
                Map input records=85331845
                Map output records=0
                Input split bytes=6240
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=14246
                Total committed heap usage (bytes)=8848932864
        edu.neu.ccs.UserTriangleCounterRep$EnumCounter
                TRIANGLE_COUNT=1560888
        File Input Format Counters
                Bytes Read=1319441569
        File Output Format Counters
                Bytes Written=320
20/02/14 00:58:56 INFO ccs.UserTriangleCounterRep:
TRIANGLE_COUNT = 520296
```