Vaibhav Shekhar Dave
Large Scale Parallel Data Processing
HW-2 Report

Git repo:
https://github.ccs.neu.edu/vaibhavdave5/parallelDataProcessing/tree/master/SocialTraingle

Implementation:

Psuedo Code:

1) Reduce-side Join with Max filter -
https://github.ccs.neu.edu/vaibhavdave5/parallelDataProcessing/blob/master/SocialTraingle/MR-Demo/src/main/java/wc/TriangleCount.java

Let the data set be key value pair for(userID, followerID) represenated by each row.

Map1 for finding Single Path

```
if(userID) < maxFilter && follower <maxFilter){
                                        {
emit( userID, “userid” - “follower”- “from”)
emit( followerID, “userid” - “follower”- “to”)

}
```

Reducer1 for Path2 – Partitioned by keys

Collect all to-s and from in two different lists namely to and from.

As the key is same we the follower info for a particular user in from list
and the information about whom a particular user follows in to List.

```
for (Text fromEdge : from) {
                for (Text toEdge : to) {
                // from is a->b and toEdge is b->c
                // So we want to emit a->c

        context.write(new Text(toEdge.toString().split("-")[0]) ,   // a
        new Text(fromEdge.toString().split("-")[1]));  // c
                }
}
```

Map 2.1 – reiterates through the database to make a to List

```
if(userID) < maxFilter && follower <maxFilter){
for each record emit (follower+"-"+userID, "to")
}
```

Map 2.2 – iterates through the reduce1 output to make a Path2List List
Let the output of reduce1 br in the format (a,b)

for each row
emit(a+"-"+b, from)

Final Reducer: Partitioned by keys

Takes input from output of Map2.1 and Map2.2

For every row

Calculate number of Tos and number of froms

So To.count > 0
There is path between that means we have completed the traingle
and add the number of fromCounts to the main count (global variable).

---

2) Map-Side Join:

https://github.ccs.neu.edu/vaibhavdave5/parallelDataProcessing/blob/master/SocialTraingle/MR-Demo/src/main/java/wc/TwitterRepJoin.java

Map 1: Let the data set be key value pair for(userID, followerID) represenated by each row.

Setup() -

Iterate through all rows and make a HashMap <userId, List<FollwerID>>

For max filter

```
if(userID > max || followerID > max) {
                        // do nothing
}
else{
        map.add(userID, List.add(FollowerID));
}
```

Map

Read each row from input: (userID, followerID)

```
if(userID > max || followerID > max) {
                        // do nothing
}
```

for each userid1 in row find userid1.followers(from map)

for each userid2 in userid1.followers find userid2.followers(from map)

```
if(userid2.followers.contains(userID)){
        count++
}
```

Configuration :

Small Cluster (4 nodes) :

REP-join, MAX = 20000 – Number of triangles 1315197755 time = 58 min

RS-join, MAX = 10000 – Number of triangles 7234833 time = 15 min

Large Cluster (7 nodes) :

REP-join, MAX = 20000 – Number of triangles 1315197755 time = 33 min

RS-join, MAX = 10000 – Number of triangles 7234833 time = 8 min

Output from Twitter Replicated Join:

https://github.ccs.neu.edu/vaibhavdave5/parallelDataProcessing/tree/d2dcb9748b38515cc183d80d00b00bfa47ca60f5/SocialTraingle/MR-Demo/output

Output from RS Join:

https://github.ccs.neu.edu/vaibhavdave5/parallelDataProcessing/tree/d375f9319bb62890a3ee8785998a37d38881022d/SocialTraingle/MR-Demo/output-Triangle

Logs

Twitter Rep Join – Large Cluster

https://s3.amazonaws.com/aws-logs-577453344208-us-east-1/elasticmapreduce/j-2UTY17W9WITC2/hadoop-mapreduce/history/2019/02/10/000000/job_1549838415613_0001-1549838663252-hadoop-TwitterRepJoin-1549840651485-20-0-SUCCEEDED-default-1549838710824.jhist.gz

RSJoin job 1 - Large Cluster

https://s3.amazonaws.com/aws-logs-577453344208-us-east-1/elasticmapreduce/j-2UTY17W9WITC2/hadoop-mapreduce/history/2019/02/10/000000/job_1549838415613_0002-1549841029486-hadoop-RSJoin-1549841186902-20-11-SUCCEEDED-default-1549841035875.jhist.gz

RS-Join Complete - Large Cluster

https://s3.console.aws.amazon.com/s3/object/aws-logs-577453344208-us-east-1/elasticmapreduce/j-2UTY17W9WITC2/hadoop-mapreduce/history/2019/02/10/000000/job_1549838415613_0003-1549841188946-hadoop-RSJoin%252BComplete%252BTriangle-1549841504069-47-11-SUCCEEDED-default-1549841195587.jhist.gz?region=us-east-1&tab=overview

Twitter Rep join

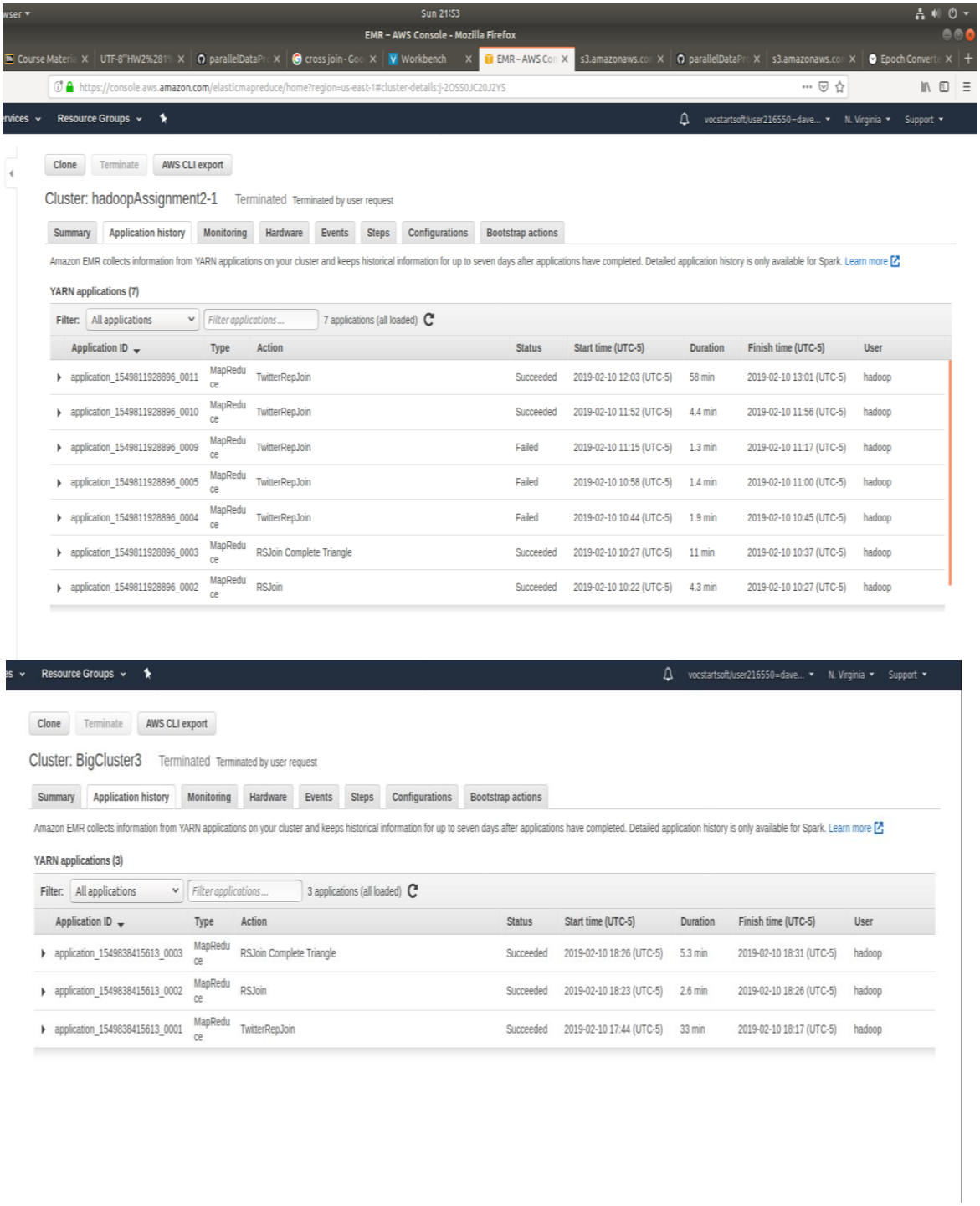
https://s3.amazonaws.com/aws-logs-577453344208-us-east-1/elasticmapreduce/j-2OSS0JC20J2YS/hadoop-mapreduce/history/2019/02/10/000000/job_1549811928896_0011-1549818237189-hadoop-TwitterRepJoin-1549821689429-20-0-SUCCEEDED-default-1549818243398.jhist.gz

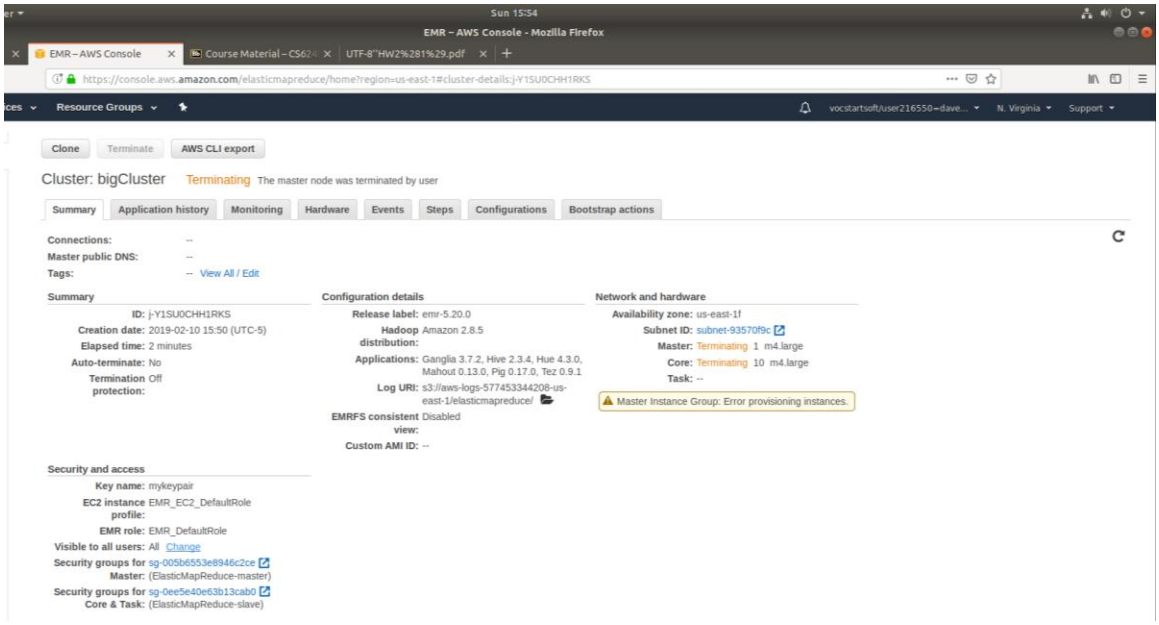
RS Join 1

https://s3.amazonaws.com/aws-logs-577453344208-us-east-1/elasticmapreduce/j-2OSS0JC20J2YS/hadoop-mapreduce/history/2019/02/10/000000/job_1549811928896_0002-1549812171266-hadoop-RSJoin-1549812429287-20-5-SUCCEEDED-default-1549812179685.jhist.gz

RS join Complete:

https://s3.amazonaws.com/aws-logs-577453344208-us-east-1/elasticmapreduce/j-2OSS0JC20J2YS/hadoop-mapreduce/history/2019/02/10/000000/job_1549811928896_0003-1549812432231-hadoop-RSJoin%2BComplete%2BTriangle-1549813064651-44-5-SUCCEEDED-default-1549812438345.jhist.gz


Sun 21:53
EMR – AWS Console - Mozilla Firefox
https://console.aws.amazon.com/elasticmapreduce/home?region=us-east-1#cluster-details:j-2OSS0JC20J2YS
Resource Groups
vocstartsoft/user216550=dave...
N. Virginia
Support
Clone
Terminate
AWS CLI export
Cluster: hadoopAssignment2-1
Terminated
Terminated by user request
Summary
Application history
Monitoring
Hardware
Events
Steps
Configurations
Bootstrap actions
Amazon EMR collects information from YARN applications on your cluster and keeps historical information for up to seven days after applications have completed. Detailed application history is only available for Spark. Learn more
YARN applications (7)
Filter:
All applications
Filter applications...
7 applications (all loaded)
Application ID
Type
Action
Status
Start time (UTC-5)
Duration
Finish time (UTC-5)
User
application_1549811928896_0011
MapReduce
TwitterRepJoin
Succeeded
2019-02-10 12:03 (UTC-5)
58 min
2019-02-10 13:01 (UTC-5)
hadoop
application_1549811928896_0010
MapReduce
TwitterRepJoin
Succeeded
2019-02-10 11:52 (UTC-5)
4.4 min
2019-02-10 11:56 (UTC-5)
hadoop
application_1549811928896_0009
MapReduce
TwitterRepJoin
Failed
2019-02-10 11:15 (UTC-5)
1.3 min
2019-02-10 11:17 (UTC-5)
hadoop
application_1549811928896_0005
MapReduce
TwitterRepJoin
Failed
2019-02-10 10:58 (UTC-5)
1.4 min
2019-02-10 11:00 (UTC-5)
hadoop
application_1549811928896_0004
MapReduce
TwitterRepJoin
Failed
2019-02-10 10:44 (UTC-5)
1.9 min
2019-02-10 10:45 (UTC-5)
hadoop
application_1549811928896_0003
MapReduce
RSJoin Complete Triangle
Succeeded
2019-02-10 10:27 (UTC-5)
11 min
2019-02-10 10:37 (UTC-5)
hadoop
application_1549811928896_0002
MapReduce
RSJoin
Succeeded
2019-02-10 10:22 (UTC-5)
4.3 min
2019-02-10 10:27 (UTC-5)
hadoop
Resource Groups
vocstartsoft/user216550=dave...
N. Virginia
Support
Clone
Terminate
AWS CLI export
Cluster: BigCluster3
Terminated
Terminated by user request
Summary
Application history
Monitoring
Hardware
Events
Steps
Configurations
Bootstrap actions
Amazon EMR collects information from YARN applications on your cluster and keeps historical information for up to seven days after applications have completed. Detailed application history is only available for Spark. Learn more
YARN applications (3)
Filter:
All applications
Filter applications...
3 applications (all loaded)
Application ID
Type
Action
Status
Start time (UTC-5)
Duration
Finish time (UTC-5)
User
application_1549838415613_0003
MapReduce
RSJoin Complete Triangle
Succeeded
2019-02-10 18:26 (UTC-5)
5.3 min
2019-02-10 18:31 (UTC-5)
hadoop
application_1549838415613_0002
MapReduce
RSJoin
Succeeded
2019-02-10 18:23 (UTC-5)
2.6 min
2019-02-10 18:26 (UTC-5)
hadoop
application_1549838415613_0001
MapReduce
TwitterRepJoin
Succeeded
2019-02-10 17:44 (UTC-5)
33 min
2019-02-10 18:17 (UTC-5)
hadoop



Tried using 11 nodes but I got this error so my small cluster has 4 nodes and large has 7 nodes.

Analysis:

Show the MapReduce pseudo-code for the program you used to determine the cardinality (and maybe data volume) of Path2. If you did not use a program, show the steps of the analysis you performed to estimate the number?

1) Psuedo Code

Mapper
Let each row of the record be (userid , followerID)

Emit(userid, from)

Emit(followid, to)

Reducer
Count the number of from c
Count the number of to d

Mul = c*d
Total count(global var) =+ mul

Implementation:

https://github.ccs.neu.edu/vaibhavdave5/parallelDataProcessing/blob/master/SocialTraingle/MR-Demo/src/main/java/wc/Path2JoinRSCardinality.java

2) Show the table with all 12 cardinality and all 12 volume estimates for the two join steps and RS-join vs. Rep-join. If you merge the two steps into one for one or both join types, state so clearly in the report. Then you only have to report the corresponding input/shuffle/file cache/output numbers for the merged program

	RS Join input	RS join Shuffle	RS join Output	Rep Join input	Rep join File Cache	Rep join Output
Step 1	Full input of Edges.csv 1319496574 bytes	1356928 Bytes 88571 records	1302114799 Bytes 4387766 records	1319496574 bytes	9130999808 bytes	1315197755 bytes
Step 2	4387766 records 1302114799 Bytes	26397 records 23200210944	**201519176792 Records** **Ran on whole edges.csv**	Merged step 1 and 2	Merged step 1 and 2	Merged step 1 and 2

3) For cardinality of Path2 I have the following code and output

Output:
https://github.ccs.neu.edu/vaibhavdave5/parallelDataProcessing/blob/915bb51456220b3d589d923521becd5b172cf214/SocialTraingle/MR-Demo/output/part-r-00000

Code:
https://github.ccs.neu.edu/vaibhavdave5/parallelDataProcessing/blob/915bb51456220b3d589d923521becd5b172cf214/SocialTraingle/MR-Demo/src/main/java/wc/Path2JoinRSCardinality.java