Homework2-Section1-Yexin Wang

https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-2

Problem Analysis (20 points total)

Pseudo-code

```
// for each tuple in the input (id1, id2)
// for id2
map(input key, input value)
  split value to get id2
  emit(id2, 1)

reduce(input key, Iterable values)
  total = 0
  for each val in values
      total += val
  emit(key, total)

// for id1, same as above
map(input key, input value)
  split value to get id1
  emit(id1, 1)

reduce(input key, Iterable values)
  total = 0
  for each val in values
      total += val
  emit(key, total)

// for path2 mapper and reducer, we pass the previous results
// as inputs
map(input key, input value)
  split the input to a tuple(id, number)
  emit(id, number)
```

```
reduce(input key, Iterable values)
// check if there are two inputs for the key
      if (values.length == 2)
          add the product of two values to the global counter
```

The output of the above is 953,138,453,592. The input edges.csv has 85331845 records. Assume the records of input and output files have the same bytes per record, we can estimate the Volume.

	RS join input	RS join shuffled	RS join output	Rep join input	Rep join file cache	Rep join output
Step1(join of Edges with itself)	85,331,845 1.32GB	170,663,690 2.64GB	953,138,453,592 14744GB	Same as RS Join	Number of machines * 85,331,845 Number of machines *1.32GB	<953,223,785,437 <14746GB
Step2(join of Path2 with Edges)	953,223,785,437 14746GB	953,223,785,437 14746GB	<953,223,785,437 <14746GB	0 0GB	0 0GB	0 0GB

(I am using markdown so it is hard to generate table, so I use word to draw the table and screenshot it. Sorry for the inconvenience.)

Since I do the step2 with step 1 in Rep Join, there is no actual step2 in it, thus no estimate either. The exact cardinality of path2 is 953,138,453,592.

Join Implementation (48 points total)

RS Join Pseudo-code with Max Filter

```
// for generate path2
map(input key, input value)
  split value in (id1, id2)
  if (id1 <= MAX and id2 <= MAX)
      // flag the input
```

```
        emit(id1, value + "first")
        emit(id2, value + "second")

reduce(input key, Iterable values)
  for each val(id1, id2, flag) in values
        if (flag is "first")
            add val to first_list
        if (flag is "second")
            add val to second_list
  for f in first_list
        for s in second_list
            if (f.id2 != s.id1)
                emit(null, (s.id1, s.di2, f.id1))

//mapper for path2 file
map(input key, input value)
  // (start, mid, end)
  split value in (id1, id2, id3)
  emit((id3, id1), value + "path2")

//mapper for edges file
map(input key, input value)
  split value in (id1, id2)
  if (id1 <= MAX and id2 <= MAX)
        emit(value, value + "edge")

// reducer for trangle, get input from the above mappers
reduce (input key, Iterable values)
  path2Count = 0
  edgesCount = 0
  for (val : values)
        if (val.length == 3)
            edgesCount++
        else
            path2Count++
  emit(COUNTER.increment(edgesCount * path2Count)
```

Rep Join Pseudo-code with Max Filter

```
// maper for file cache
map (input key, input value)
  split value in (id1, id2)
  if (id1 <= MAX and id2 <= MAX)
      emit(null, value)

// mapper for calculate triangle
setup()
  read records from file cache and construct a hashmap map
  with id1 as key, all id2 as a list as value

map (input key, input value)
  split value in (id1, id2)

  // for each (id1, id2), we get the list of id2, for each
  // id3 in that list, we get the list of id3 and see if id1
  // in that list of id3

  if (id1 <= MAX and id2 <= MAX)
      if (map.contains(id2))
          for (id3 in map.get(id2))
              if (map.contains(id3))
                  if (map.get(id3).contains(id1))
                      emit(COUNTER.increment(1)
```

Result Table

For each Triangle (a, b, c), my programs count (a, b, c), (c, a, b) and (b, c, a). So the distinct number of Triangles should be COUNTER / 3. I calculate it from the Triangle counter number from syslog.

36,089,721 / 3 = 12,029,907

Configuration	Small Cluster Result	Large Cluster Result
RS-join, MAX = 50000	Running time: 47 min Triangle count: 12,029,907.	Running time: 23 min Triangle count: 12,029,907.
Rep-join, MAX = 50000	Running time: 37 min Triangle count: 12,029,907.	Running time: 37 min Triangle count: 12,029,907.

links

log file:

- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-2/tree/master/MR-Demo-RSJoin/6%20m4.large/log
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-2/tree/master/MR-Demo-RSJoin/11%20m4.large/log
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-2/tree/master/MR-Demo-RepJoin/6%20m4.large/log
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-2/tree/master/MR-Demo-RepJoin/11%20m4.large

output file:

Since we use the global counter, the output files are empty and meaningless.