Sravani Beeram

CS6240 Parallel Data Processing using Mapreduce
Homework 2

Pseudo Code - Question : 1

1 No-Combiner :

Mapper

```
map(value) {

//From value consider only records that contain either TMIN or TMAX

    key   = stationId
    temp = temperature_value

    // If record contains TMIN

       value = {temp,0,true}

    // If record contains TMAX

       value = {0,temp,false}

     emit(key,value)

}
```

Reducer

```
reducer(key,value_list){

for (value : value_list){

  // If minTemp
     minTempSum += value.minTemp()
     minCount++
 // else
    maxTempSum += value.maxTemp()
    maxCount++
}

Output = key + (minTempSum/minCount) + (maxTempSum/maxCount)
emit(output,res)        //res is NullWritable
}
```

2 Combiner:

Mapper

```
map(value) {

//From value consider only records that contain either TMIN or TMAX

    key   = stationId
    temp = temperature_value

    // If record contains TMIN

       value = {temp,0,true}

    // If record contains TMAX

       value = {0,temp,false}

     emit(key,value)

}
```

Combiner

```
reduce(key,value_list){

minTempSum = 0
maxTempSum = 0
minCount = 0
maxCOunt = 0

for (value : value_list){

  // If minTemp
     minTempSum += value.minTemp()
     minCount++
 // else
    maxTempSum += value.maxTemp()
    maxCount++
 }
 data = {minTempSum,maxTempSum,minCount,maxCount}
 emit(key,data)
}
```

Reducer

```
reduce(key,value_list){

for (value : value_list){

  // If minTemp
     minTempSum += value.minTemp()
     minCount++
 // else
    maxTempSum += value.maxTemp()
    maxCount++
}

Output = key + (minTempSum/minCount) + (maxTempSum/maxCount)
emit(output,res)        //res is NullWritable
}
```

3 InMapper:

Mapper:

```
setup(){
 //Initialize hashmap
}

 map(value) {

 //From value consider only records that contain either TMIN or TMAX

     key   = stationId
     temp = temperature_value

     // If record contains TMIN

        //if hashmap weather_data already contains key stationId
            //update the temperature and count
        //else
           // value = (minTemp,maxTemp,minCount,maxCount)
           //insert key and value(temp,0,1,0) into hashmap

     // If record contains TMAX

        //if hashmap weather_data already contains key stationId
            //update the temperature and count
        //else
           //insert key and value(0,temp,0,1) into hashmap
   }


 cleanup(){

   for(s : hashmap){
    emit(s,hashmap(s))
   }
 }
```

Reducer

```
reduce(key,value_list){

for (value : value_list){

    minTempSum += value.minTemp()
    minCount       += value.minCount()
    maxTempSum += value.maxTemp()
    maxCount       += value.maxCount()
}

Output = key + (minTempSum/minCount) + (maxTempSum/maxCount)
emit(output,res)       //res is NullWritable
}
```

Pseudo Code - Question : 2

Secondary Sort :

Mapper

```
map(value) {

//From value consider only records that contain either TMIN or TMAX

    temp = temperature_value
   CompositeKey = {stationId,year}

    // If record contains TMIN

       value = {temp,0,1,0}
       emit(CompositeKey,value)

    // If record contains TMAX

       value = {0,temp,0,1}
       emit(CompositeKey,value)

}
```

Partitioner

```
getPartition(CompositeKey,numReduceTasks){
    return key.getStationId().hashCode() % numReduceTasks
}
```

Key Comparator

```
//Sorts in increasing order of stationId first and If the stationId is equal,sorts in decreasing order of year next

compare(CompositeKey k1,CompositeKey k2){

    res= key1.getStationId().compareTo(key2.getStationId())

    if(res == 0) {
        return key1.getYear().compareTo(key2.getYear())
    }
    return res
}
```

Grouping Comparator

```
//Sorts in increasing order of stationId and does not consider year

compare(CompositeKey k1,CompositeKey k2){
  return key1.getStationId().compareTo(key2.getStationId())
}
```

Reducer

```
reduce(CompositeKey,value_list){

    for(value  :value_list){

        if(current_year == prev_year){

            minTempSum += s.getMinTemp()
            minCount   += s.getMinCount()

            maxTempSum += s.getMaxTemp()
            maxCount   += s.getMaxCount()

        }else{
```

```
                prev_year = current_year

                minTempSum = s.getMinTemp()
                minCount = s.getMinCount()

                maxTempSum = s.getMaxTemp()
                maxCount = s.getMaxCount();

            }

         Rec = year + (minTempSum/minCount) + (maxTempSum/maxCount)

         // if arraylist already has a value with year same as in Rec then replace the
            // value with Rec
        // else insert the value into arraylist

        emit(stationId+arraylist,res)    //res is NullWritable
}
```

Comments ::When we call reduce method we sort objects in order of its key(stationId,year).Output through this reduce call would be (stationid,year) values for each record.The output format expected is such that we need to have all stationid's listed for a single year.So, we use a grouping comparator which compares objects on the basis of stationId and groups them together to display year and value

Performance Comparison

Running Times:

Programs	**Run 1**(seconds)	**Run 2**(seconds)
NoCombiner	80	82
Combiner	76	80
InMapper	74	74
Secondary Sort	58	-

1. *Was the Combiner called at all in program Combiner? Was it called more than once per Map task?*

   Yes, Combiner was called in program Combiner. Below data from syslog confirms the same.

   Combine input records=8798241
   Combine output records=223783

   Hadoop doesn't guarantee on how many times a combiner function will be called for each map task key.So, we cannot tell if combiner was called more than once or not.

2. *What difference did the use of a Combiner make in Combiner compared to NoCombiner?*

   In combiner, number of input records in reducer is very less compared to NoCombiner there by decreasing the overhead usage.
   Below data from syslog confirms the same

   No-Combiner : Map output records=8798241
   Reduce input records=8798241

   Combiner : Map output records=8798241
   Reduce input records=223783

3.Was the local aggregation effective in InMapperComb compared to NoCombiner?

Yes,the number of map output records and reduce input records in InMapper decreased significantly compared to the NoCombiner map output records

InMapperCombiner : Map input records=30868726
Map output records=223783

Reduce input records=223783
Reduce output records=14135

No Combiner : Map input records=30868726
Map output records=8798241

Reduce input records=8798241
Reduce output records=14135

4. Which one is better, Combiner or InMapperComb? Briefly justify your answer.

InMapperCombiner : Map input records=30868726
Map output records=223783
Reduce input records=223783

Combiner : Map input records=30868726
Map output records=8798241
Reduce input records=223783

Runtime of InMapper is less compared to Combiner. But, InMapper uses Hashmap it needs more memory compared to Combiner. So if memory is not an issue then InMapper usage is better but if memory is an issue then using Combiner is better.

5. How do the running times and accuracy of these MapReduce programs compare to the sequential implementation of per-station mean temperature?

Sequential implementation took approximately 8.5 seconds to run which is significantly very less compared to the mapreduce programs. As current dataset is small (only 1 GB) and Map reduce programs have overhead as data transfers between map

and reduce which is not the case in sequential.So,sequential running time is very less. But this wouldn’t be case if the data is very huge.If data is very huge then Mapreduce programs run faster compared to sequential.