Parallel K-Means using Hadoop

Islam Mohamed Mohamed 12
Bahaa Khalf 20

The unparalleled K-Means pseudo-code:

Input:
D \\Set of elements
K \\Number of required clusters
Output:
K \\Set of clusters
K-means algorithm:
Assign initial values for K centroid
repeat
Assign each item in the data set to the closest centroid;
Recalculate the new centroids by taking the mean for each set associated with the old centroid;
until convergence;

MapReduce K-Means algorithm:
Github Link
Code Screen Shots:
Main :

```
public static void main(String[] args) throws Exception {
    run(args);
}
```

```java
public static void run(String[] args) throws Exception {
    if(args.length != 4) {
        System.out.print("Invalid Error!") ;
        System.exit(-1);
    }
    String input = args[0], output = args[1];
    int k = Integer.valueOf(args[2]),  dim = Integer.valueOf(args[3]) ;
    System.out.println("*********************************");
    String centroids = "/irisInput/centers.txt";
    boolean isdone = false;
    double[][] old_centers = new double[k][dim] ;
    int iteration = 1 ;
    while (isdone == false) {
        Job job = Job.getInstance() ;
        Configuration conf = job.getConfiguration() ;

        String path = centroids ;
        Configuration temp = new Configuration();
        FileSystem file = FileSystem.get(URI.create(path), temp);
        Path input_path = new Path(path);
        FSDataInputStream in = file.open(input_path);
        BufferedReader buffer = new BufferedReader(new InputStreamReader(in));
        double[][] new_centers = new double[k][dim] ;
        conf.set("k", Integer.toString(k));
        conf.set("dim", Integer.toString(dim));
        for(int i = 0 ; i < k ; i++) {
            String line = buffer.readLine() ;
            System.out.println("Centroid");
            System.out.println(line);
            int key = Integer.valueOf(line.split("\t")[0]) ;
            String[] center = line.split("\t")[1].split(",") ;
            if(center.length != dim ){
                System.out.print("Invalid Input Length!");
                System.exit(-1);
            }
            for(int j = 0 ; j < dim ; j++) {
                new_centers[key][j] = Double.valueOf(center[j]) ;

            }
            conf.set("centroid" + key, line.split("\t")[1]);
        }
```

```java
        double error = 0 ;
        for(int i=0;i<k;i++){
            for(int j=0;j<dim;j++){
                error += Math.pow(new_centers[i][j] - old_centers[i][j], 2) ;
            }
        }
        double tolerance = 0.000001;
        if(error < tolerance)
            break ;

        job.setJarByClass(KMeans.class);
        job.setJobName("KMeans");
        job.setMapOutputKeyClass(IntWritable.class);
        job.setMapOutputValueClass(Text.class);
        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(Text.class);
        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);
        FileInputFormat.addInputPath(job, new Path(input));
        FileOutputFormat.setOutputPath(job, new Path(output+"_"+Integer.toString(iteration)));
        job.waitForCompletion(true);

        old_centers = new_centers;
        centroids =  output + "_" + Integer.toString(iteration) + "/part-r-00000";
        iteration++ ;
    }

}
```

Map Class :

```java
public static class Map extends Mapper<LongWritable, Text, IntWritable, Text> {

    @Override
    protected void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {

        Configuration conf = (Configuration) context.getConfiguration() ;
        int k = Integer.valueOf(conf.get("k")) ;
        double min = Double.MAX_VALUE ;
        int best = -1 ;
        String[] all_dim = value.toString().split(",") ;
        for ( int i = 0;i < k; i ++) {
            String center = conf.get("centroid" + Integer.toString(i)) ;
            String[] center_values = center.split(",") ;
            double dist = 0 ;
            for(int j = 0; j < all_dim.length - 1; j++) {
                double point = Double.parseDouble(all_dim[j]) ;
                double center_value = Double.parseDouble(center_values[j]) ;
                dist += Math.pow(point - center_value, 2) ;
            }
            dist = Math.sqrt(dist);
            if(dist < min) {
                min = dist;
                best = i;
            }
        }
        context.write(new IntWritable(best) , value);
    }
}
```

Reduce Class :

```java
public static class Reduce extends Reducer<IntWritable, Text, IntWritable, Text> {
    @Override
    protected void reduce(IntWritable centroid_ind, Iterable<Text> data,Context context)
            throws IOException, InterruptedException {

        Configuration conf = (Configuration) context.getConfiguration() ;
        int dim = Integer.valueOf(conf.get("dim")) ;
        double[] avg = new double[dim] ;
        int count = 0 ;
        for(Text d : data) {
            String[] elements = d.toString().split(",") ;
            for(int i = 0 ; i < elements.length -1  ; i++) {
                double val = Double.valueOf(elements[i]) ;
                avg[i] +=  val  ;
            }
            count++ ;
        }
        for(int i = 0 ; i < dim  ; i++) {
            avg[i] /= count;
        }
        StringBuilder s = new StringBuilder() ;
        for(int i = 0 ; i < avg.length ; i++) {
            s.append(String.valueOf(avg[i])) ;
            if( i != avg.length - 1) s.append(",") ;
        }
        String value = s.toString() ;
        context.write(centroid_ind, new Text(value));
    }
}
```

The challenges faced and how it was solved:

- Getting the initial centers. Solved

we made a file called centers that the user can write the initial centers to use it, then copy the file to Hadoop file systems, and this file should be in a specific form that is key \t The centers

ex:

iris.data | centers.txt

```
0       4.9,3.0,1.4,0.2
1       5.1,3.5,1.4,0.2
2       7.0,3.2,4.7,1.4
```

from command line:

```
[cloudera@quickstart ~]$ hdfs dfs -cat  /irisInput/centers.txt
0       4.9,3.0,1.4,0.2
1       5.1,3.5,1.4,0.2
2       7.0,3.2,4.7,1.4
[cloudera@quickstart ~]$
```

- The output file exists. Solved

rename the output file at every iteration.

in code :

```
FileOutputFormat.setOutputPath(job, new Path(output+"_"+Integer.toString(iteration)));
```

in hadoop files

```
drwxr-xr-x   - cloudera supergroup          0 2022-03-19 05:33 /final_1
drwxr-xr-x   - cloudera supergroup          0 2022-03-19 05:35 /final_2
drwxr-xr-x   - cloudera supergroup          0 2022-03-19 05:37 /final_3

drwxr-xr-x   - cloudera supergroup          0 2022-03-19 04:32 /help_3
drwxr-xr-x   - cloudera supergroup          0 2022-03-19 04:58 /help_3_1
drwxr-xr-x   - cloudera supergroup          0 2022-03-19 04:59 /help_3_2
drwxr-xr-x   - cloudera supergroup          0 2022-03-19 05:00 /help_3_3
drwxr-xr-x   - cloudera supergroup          0 2022-03-19 05:02 /help_3_4
drwxr-xr-x   - cloudera supergroup          0 2022-03-19 05:03 /help_3_5
```

- Get the new centers. Solved

reading them from the previous iteration output file.

```
centroids =  output + "_" + Integer.toString(iteration) + "/part-r-00000";
```

- The biggest challenge was to pass the centroids and parameters to Map and Reduce classes. Solved

solution:

First we tried to make global variables and use these variables inside these classes but the values of the variables vanished inside the classes.

Then, we used the configuration class to pass the parameters to the Map and Reduce classes by using the function set(String key, String value) then we use the function get(String key) to get the parameters when we need it.

```
conf.set("k", Integer.toString(k));
conf.set("dim", Integer.toString(dim));

int k = Integer.valueOf(conf.get("k")) ;

int dim = Integer.valueOf(conf.get("dim")) ;
```

<u>Unknown behavior:</u>

- Inside the reduce function we tried to loop on the dimensions readed from the file by using dim as a stop condition in for loop but it was giving that we try

to read empty string instead of string which can't be converted to double so we tried to use elements.length-1 as a stop condition and the program worked, and this behavior was also inside the map function.
The weird thing that when we tried to print dim and elements.length-1 in the output file both of them was equal to 4

- if we use dim which is 4 :

```
for(Text d : data) {
    String[] elements = d.toString().split(",") ;
    for(int i = 0 ; i < dim  ; i++) {
        double val = Double.valueOf(elements[i]) ;
        avg[i] +=  val  ;
    }
    count++ ;
}
```

  - it shows this error :

```
22/03/19 12:00:38 INFO mapreduce.Job:  map 0% reduce 0%
22/03/19 12:00:52 INFO mapreduce.Job:  map 100% reduce 0%
22/03/19 12:01:07 INFO mapreduce.Job: Task Id : attempt_1646456123277_0093_r_000
000_0, Status : FAILED
Error: java.lang.NumberFormatException: empty String
        at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:10
20)
        at java.lang.Double.valueOf(Double.java:504)
        at kmeans.KMeans$Reduce.reduce(KMeans.java:64)
        at kmeans.KMeans$Reduce.reduce(KMeans.java:1)
        at org.apache.hadoop.mapreduce.Reducer.run(Reducer.java:171)
        at org.apache.hadoop.mapred.ReduceTask.runNewReducer(ReduceTask.java:627
)
        at org.apache.hadoop.mapred.ReduceTask.run(ReduceTask.java:389)
        at org.apache.hadoop.mapred.YarnChild$2.run(YarnChild.java:164)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:415)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInforma
tion.java:1917)
```

- if we use elements.length -1, it works well .

The evaluation results:

MapReduce is likely to be less efficient on a single Node.
In other words, if you are using a single Node, it's good to use a custom solution instead of MapReduce to achieve higher performance.
But if you want to add more nodes and create a cluster, then MapReduce will have better performance.

```
22/03/19 05:58:20 INFO mapreduce.Job:  map 100% reduce 100%
22/03/19 05:58:21 INFO mapreduce.Job: Job job_1646456123277_0092 completed successfully
22/03/19 05:58:21 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=5463
                FILE: Number of bytes written=299739
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=4667
                HDFS: Number of bytes written=218
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=21459
                Total time spent by all reduces in occupied slots (ms)=22304
                Total time spent by all map tasks (ms)=21459
                Total time spent by all reduce tasks (ms)=22304
                Total vcore-milliseconds taken by all map tasks=21459
                Total vcore-milliseconds taken by all reduce tasks=22304
                Total megabyte-milliseconds taken by all map tasks=21974016
                Total megabyte-milliseconds taken by all reduce tasks=22839296
        Map-Reduce Framework
                Map input records=151
                Map output records=151
                Map output bytes=5155
                Map output materialized bytes=5463
                Input split bytes=116
                Combine input records=0
                Combine output records=0
                Reduce input groups=3
                Reduce shuffle bytes=5463
                Reduce input records=151
                Reduce output records=3
                Spilled Records=302
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=691
                CPU time spent (ms)=3640
                Physical memory (bytes) snapshot=359329792
```

using 2 centers and 4 dimensions :

```
[cloudera@quickstart ~]$ hdfs dfs -cat /help_3_5/part-r-00000
0       4.912962962962963,3.298148148148148,1.5333333333333339,0.28333333333333316
1       6.301030927835049,2.88659793814433,4.95876288659794,1.6958762886597938
```

using 3 centers and 4 dimensions

```
[cloudera@quickstart ~]$ hdfs dfs -cat /final_3/part-r-00000
0       4.527272727272727,2.7590909090909093,1.6909090909090907,0.3272727272727273
1       5.172727272727273,3.618181818181819,1.487878787878788,0.2757575757575758
2       6.314583333333331,2.8958333333333326,4.973958333333333,1.703125
```