Practical - 7: Implement any one of the analytic algorithm using mapreduce by handling larger datasets in main memory.

- K-means Clustering

By Divya Mahur (18bce106)

➔ **`java` file**

```
package com.code.dezyre;


import java.io.IOException;
import java.util.*;
import java.io.*;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.filecache.DistributedCache;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.mapred.Reducer;
```

```
@SuppressWarnings("deprecation")
public class KMeans {
    public static String OUT = "outfile";
    public static String IN = "inputlarger";
    public static String CENTROID_FILE_NAME = "/centroid.txt";
    public static String OUTPUT_FILE_NAME = "/part-00000";
    public static String DATA_FILE_NAME = "/data.txt";
    public static String JOB_NAME = "KMeans";
    public static String SPLITTER = "\t| ";
    public static List<Double> mCenters = new ArrayList<Double>();


    /*
     * In Mapper class we are overriding configure function. In this we
are
     * reading file from Distributed Cache and then storing that into
instance
     * variable "mCenters"
     */
    public static class Map extends MapReduceBase implements
            Mapper<LongWritable, Text, DoubleWritable, DoubleWritable> {
        @Override
        public void configure(JobConf job) {
            try {
                // Fetch the file from Distributed Cache Read it and store
the
                // centroid in the ArrayList
                Path[] cacheFiles =
DistributedCache.getLocalCacheFiles(job);
                if (cacheFiles != null && cacheFiles.length > 0) {
```

```
                    String line;
                    mCenters.clear();
                    BufferedReader cacheReader = new BufferedReader(
                            new FileReader(cacheFiles[0].toString()));
                    try {
                        // Read the file split by the splitter and store
it in
                        // the list
                        while ((line = cacheReader.readLine()) != null) {
                            String[] temp = line.split(SPLITTER);
                            mCenters.add(Double.parseDouble(temp[0]));
                        }
                    } finally {
                        cacheReader.close();
                    }
                }
            } catch (IOException e) {
                System.err.println("Exception reading DistribtuedCache: "
+ e);
            }
        }

        /*
         * Map function will find the minimum center of the point and emit
it to
         * the reducer
         */
        @Override
        public void map(LongWritable key, Text value,
                OutputCollector<DoubleWritable, DoubleWritable> output,
```

```
                Reporter reporter) throws IOException {
            String line = value.toString();
            double point = Double.parseDouble(line);
            double min1, min2 = Double.MAX_VALUE, nearest_center =
mCenters
                    .get(0);
            // Find the minimum center from a point
            for (double c : mCenters) {
                min1 = c - point;
                if (Math.abs(min1) < Math.abs(min2)) {
                    nearest_center = c;
                    min2 = min1;
                }
            }
            // Emit the nearest center and the point
            output.collect(new DoubleWritable(nearest_center),
                    new DoubleWritable(point));
        }
    }

    public static class Reduce extends MapReduceBase implements
            Reducer<DoubleWritable, DoubleWritable, DoubleWritable, Text>
{

        /*
         * Reduce function will emit all the points to that center and
calculate
         * the next center for these points
         */
        @Override
```

```
        public void reduce(DoubleWritable key, Iterator<DoubleWritable>
values,
                OutputCollector<DoubleWritable, Text> output, Reporter
reporter)
                throws IOException {
            double newCenter;
            double sum = 0;
            int no_elements = 0;
            String points = "";
            while (values.hasNext()) {
                double d = values.next().get();
                points = points + " " + Double.toString(d);
                sum = sum + d;
                ++no_elements;
            }

            // We have new center now
            newCenter = sum / no_elements;

            // Emit new center and point
            output.collect(new DoubleWritable(newCenter), new
Text(points));
        }
    }

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

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

```
        IN = args[0];
        OUT = args[1];
        String input = IN;
        String output = OUT + System.nanoTime();
        String again_input = output;

        // Reiterating till the convergence
        int iteration = 0;
        boolean isdone = false;
        while (isdone == false) {
            JobConf conf = new JobConf(KMeans.class);
            if (iteration == 0) {
                Path hdfsPath = new Path(input + CENTROID_FILE_NAME);
                // upload the file to hdfs. Overwrite any existing copy.
                DistributedCache.addCacheFile(hdfsPath.toUri(), conf);
            } else {
                Path hdfsPath = new Path(again_input + OUTPUT_FILE_NAME);
                // upload the file to hdfs. Overwrite any existing copy.
                DistributedCache.addCacheFile(hdfsPath.toUri(), conf);
            }

            conf.setJobName(JOB_NAME);
            conf.setMapOutputKeyClass(DoubleWritable.class);
            conf.setMapOutputValueClass(DoubleWritable.class);
            conf.setOutputKeyClass(DoubleWritable.class);
            conf.setOutputValueClass(Text.class);
            conf.setMapperClass(Map.class);
            conf.setReducerClass(Reduce.class);
```

```
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf,
                new Path(input + DATA_FILE_NAME));
        FileOutputFormat.setOutputPath(conf, new Path(output));

        JobClient.runJob(conf);

        Path ofile = new Path(output + OUTPUT_FILE_NAME);
        FileSystem fs = FileSystem.get(new Configuration());
        BufferedReader br = new BufferedReader(new InputStreamReader(
                fs.open(ofile)));
        List<Double> centers_next = new ArrayList<Double>();
        String line = br.readLine();
        while (line != null) {
            String[] sp = line.split("\t| ");
            double c = Double.parseDouble(sp[0]);
            centers_next.add(c);
            line = br.readLine();
        }
        br.close();

        String prev;
        if (iteration == 0) {
            prev = input + CENTROID_FILE_NAME;
        } else {
            prev = again_input + OUTPUT_FILE_NAME;
```

```
            }
            Path prevfile = new Path(prev);
            FileSystem fs1 = FileSystem.get(new Configuration());
            BufferedReader br1 = new BufferedReader(new InputStreamReader(
                    fs1.open(prevfile)));
            List<Double> centers_prev = new ArrayList<Double>();
            String l = br1.readLine();
            while (l != null) {
                String[] sp1 = l.split(SPLITTER);
                double d = Double.parseDouble(sp1[0]);
                centers_prev.add(d);
                l = br1.readLine();
            }
            br1.close();

            // Sort the old centroid and new centroid and check for
convergence
            // condition
            Collections.sort(centers_next);
            Collections.sort(centers_prev);

            Iterator<Double> it = centers_prev.iterator();
            for (double d : centers_next) {
                double temp = it.next();
                if (Math.abs(temp - d) <= 0.1) {
                    isdone = true;
                } else {
                    isdone = false;
                    break;
```

```
                    }

                }

                ++iteration;

                again_input = output;

                output = OUT + System.nanoTime();

                for (double d : centers_next)

                {

                    System.out.println(d);

                }

            }

}

}
```

```
Administrator: Command Prompt
C:\hadoop>hadoop jar KMeans.jar com.code.dezyre.KMeans /prac7/input /prac7/output
2021-11-16 11:14:29,224 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2021-11-16 11:14:29,410 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2021-11-16 11:14:30,189 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2021-11-16 11:14:30,483 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/tirth/.staging/job_1637039175847_0010
2021-11-16 11:14:31,094 INFO mapred.FileInputFormat: Total input files to process : 1
2021-11-16 11:14:31,535 INFO mapreduce.JobSubmitter: number of splits:2
2021-11-16 11:14:31,885 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1637039175847_0010
2021-11-16 11:14:31,888 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-11-16 11:14:32,253 INFO conf.Configuration: resource-types.xml not found
2021-11-16 11:14:32,254 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-11-16 11:14:32,375 INFO impl.YarnClientImpl: Submitted application application_1637039175847_0010
2021-11-16 11:14:32,468 INFO mapreduce.Job: The url to track the job: http://DESKTOP-LMJLBO6:8088/proxy/application_1637039175847_0010/
2021-11-16 11:14:32,471 INFO mapreduce.Job: Running job: job_1637039175847_0010
2021-11-16 11:14:46,742 INFO mapreduce.Job: Job job_1637039175847_0010 running in uber mode : false
2021-11-16 11:14:46,744 INFO mapreduce.Job:  map 0% reduce 0%
2021-11-16 11:14:58,033 INFO mapreduce.Job:  map 100% reduce 0%
2021-11-16 11:15:08,166 INFO mapreduce.Job:  map 100% reduce 100%
2021-11-16 11:15:09,188 INFO mapreduce.Job: Job job_1637039175847_0010 completed successfully
2021-11-16 11:15:09,357 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=162006
                FILE: Number of bytes written=1035551
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=58283
                HDFS: Number of bytes written=72042
                HDFS: Number of read operations=11
                HDFS: Number of large read operations=0
```

```
2021-11-16 11:14:32,471 INFO mapreduce.Job: Running job: job_1637039175847_0010
2021-11-16 11:14:46,742 INFO mapreduce.Job: Job job_1637039175847_0010 running in uber mode : false
2021-11-16 11:14:46,744 INFO mapreduce.Job:  map 0% reduce 0%
2021-11-16 11:14:58,033 INFO mapreduce.Job:  map 100% reduce 0%
2021-11-16 11:15:08,166 INFO mapreduce.Job:  map 100% reduce 100%
2021-11-16 11:15:09,188 INFO mapreduce.Job: Job job_1637039175847_0010 completed successfully
2021-11-16 11:15:09,357 INFO mapreduce.Job: Counters: 54
        File System Counters
```

```
                GC time elapsed (ms)=253
                CPU time spent (ms)=5824
                Physical memory (bytes) snapshot=821387264
                Virtual memory (bytes) snapshot=1247543296
                Total committed heap usage (bytes)=726138880
                Peak Map Physical memory (bytes)=312598528
                Peak Map Virtual memory (bytes)=468795392
                Peak Reduce Physical memory (bytes)=233177088
                Peak Reduce Virtual memory (bytes)=391741440
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=58095
        File Output Format Counters
                Bytes Written=72042
12368.836
16017.089333333333
21241.047
```

```
                CPU time spent (ms)=5230
                Physical memory (bytes) snapshot=819335168
                Virtual memory (bytes) snapshot=1222639616
                Total committed heap usage (bytes)=691535872
                Peak Map Physical memory (bytes)=303476736
                Peak Map Virtual memory (bytes)=438587392
                Peak Reduce Physical memory (bytes)=239972352
                Peak Reduce Virtual memory (bytes)=396460032
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=58095
        File Output Format Counters
                Bytes Written=72042
12368.836
16017.089333333333
21241.047
```