

Assignment - 4

- Aim:- Design & develop a distributed application to find the coolest/hottest year from available weather data. Process it using MapReduce.

- Theory:-

Q.1. List & explain commands used for execution of any mapreduce program.

→ ① Download the input file/dataset and store this on HDFS.

Syntax: `hdfs dfs -copyFromLocal /path/to/input /path/HDFS`

Example: `hdfs dfs -copyFromLocal /Downloads/input.txt /PVG/input.txt`

Alternatively, we can use the `-put` command to transfer file from local system to HDFS.

② Create output directory in HDFS

Syntax: `hdfs dfs -mkdir /path`

Example: `hdfs dfs -mkdir /PVG/output`

③ Create source code. Write the code for Mapper, Reducer & Driver classes.

④ Create JAR file.

For Eclipse IDE, go to Java project^{package} you are working on
→ Export → JAR file → select Java project → Finish select
export destination → Finish

After this step, jar file will be created in Downloads Folder.

⑤ Execute jar file using hadoop jar command

syntax: `hadoop jar /path/to/jar/file Driver class /path/to/
input/file /path/to/output/directory`

Example: `hadoop jar /home/hduser/words.jar Words.Driver
/PVG/input.txt /PVG/output`

⑥ Display output using cat command

syntax: `hdfs dfs -cat /path/to/file`

Example: `hdfs dfs -cat /PVG/output/part-*`

Q.2 Explain following classes.

① Job

→ ① The Hadoop job class represents the unit of work that is sent to the mapreduce programming model.

② It allows user to configure the job, control its execution and query the state. User creates application, describes the job & then submits job & monitors its progress.

③ `Job job = Job.getInstance();`

`Job job = new Job(new Configuration(), "Name");`

② File Input Format

→ ① `FileInputFormat` is the base class for all File based Input formats. This provides a generic implementation of `getSplits()`.

② `FileInputFormat` is used to provide input file to Hadoop job.

③ `FileInputFormat.addInputPath(job, new Path("..."));`

③ FileOutputFormat

→ ① FileOutputFormat is the base class for OutputFormats that read from file systems.

② FileOutputFormat.addOutputPath(job, new Path("..."));

④ FileStatus

→ ① FileStatus is an interface that represents the client side information for a file.

② Output of fs.listStatus(path) returns an array of FileStatus objects in which arr[0] is success_file & arr[2] contains file information.

③ This information includes name, size, path, block size, permissions etc.

⑤ LongWritable

→ ① LongWritable class in Hadoop wraps the Java Long class which is used for storing large numbers.

② LongWritable is a serializable class & is similar to IntWritable in terms of implementation.

- Conclusion:- Thus we have successfully implemented MapReduce for Weather Data.

Driver class : Driver.java

```
package weather;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Driver {

    @SuppressWarnings("deprecation")
    public static void main(String[] args) throws Exception{
        //creating object of configuration class
        Configuration c = new Configuration();

        //Assigning job to new configuration object
        Job job = new Job(c);

        //setting jar class
        job.setJarByClass(weather.Driver.class);

        job.setMapperClass(weather.TempMapper.class);

        job.setReducerClass(weather.TempReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        //Adding a Path to the list of inputs
        FileInputFormat.addInputPath(job, new Path(args[0]));

        //Setting the Path of the output directory
        FileOutputFormat.setOutputPath(job,new Path(args[1]));

        //wait till job is completed
        job.waitForCompletion(true);

        //file system object
        FileSystem fs = FileSystem.get(c);

        FileStatus[] status = fs.listStatus(new
Path("hdfs://localhost:9000"+args[1]));
        FSDataInputStream fd = fs.open(status[1].getPath());

        String str = fd.readLine();

        float max = Integer.MIN_VALUE, min = Integer.MAX_VALUE, temp;
        String minYear = null, maxYear = null;

        int pos = 0;

        while(str != null) {

            String [] parts = str.split("\t");
            temp = Integer.parseInt(parts[1]);

            if(pos % 2 == 0)
            {
                if(temp > max)
                {
                    max = temp;
                    maxYear = parts[0];
                }
            }
            else {
                if(temp < min)
                {
                    min = temp;
                    minYear = parts[0];
                }
            }
            pos++;
            str = fd.readLine();
        }

        System.out.println("Max Temp: " + max + " in " + maxYear);
        System.out.println("Min Temp: " + min + " in " + minYear);
    }
}
```

```

        }
    }

    pos++;

    str = fd.readLine();
}

System.out.println("Maximum temperature : " + max/10 + " in the year " +
maxYear);
System.out.println("Minimum temperature : " + min/10 + " in the year " +
minYear);

}
}

```

Mapper class : TempMapper.java

```

package weather;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class TempMapper extends Mapper<LongWritable,Text,Text,IntWritable>
{
    public void map(LongWritable key,Text value,Context context) throws
IOException,InterruptedException {

        String line = value.toString();
        String year = line.substring(15, 19);
        int temp = 9999;

        if (line.charAt(87)=='+') {
            temp = Integer.parseInt(line.substring(88, 92));
        } else {
            temp = Integer.parseInt(line.substring(87, 92));
        }

        if (temp != 9999) {
            context.write(new Text(year), new IntWritable(temp));
        }
    }
}

```

Reducer class : TempReducer.java

```

package weather;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class TempReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {

        int max = -9999;
        int min = 9999;
    }
}

```

```
for(IntWritable value : values) {

    if(value.get() < min)
        min = value.get();
    if(value.get() > max)
        max = value.get();

}

context.write(key, new IntWritable(max));
context.write(key, new IntWritable(min));

}
```

Output Screenshots

```
No. of occurrences : 63
hduser@omkar-VirtualBox:~$ hadoop jar /home/hduser/Downloads/temp.jar weather.Driver /Weather/input-dataset /Weather/Weather_output_2/
21/05/22 16:55:01 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
21/05/22 16:55:03 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
21/05/22 16:55:03 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
21/05/22 16:55:03 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy th
21/05/22 16:55:04 INFO input.FileInputFormat: Total input files to process : 20
21/05/22 16:55:04 INFO mapreduce.JobSubmitter: number of splits:20
21/05/22 16:55:05 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local86310316_0001
21/05/22 16:55:06 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
21/05/22 16:55:06 INFO mapreduce.Job: Running job: job_local86310316_0001
21/05/22 16:55:06 INFO mapred.LocalJobRunner: OutputCommitter set in config null
21/05/22 16:55:06 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
21/05/22 16:55:06 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup_temporary folders under output directory:false, ignore cleanup failures: false
21/05/22 16:55:06 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
21/05/22 16:55:06 INFO mapred.LocalJobRunner: Waiting for map tasks
21/05/22 16:55:06 INFO mapred.LocalJobRunner: Starting task: attempt_local86310316_0001_m_000000_0
21/05/22 16:55:06 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
21/05/22 16:55:07 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup_temporary folders under output directory:false, ignore cleanup failures: false
21/05/22 16:55:07 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
21/05/22 16:55:07 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/Weather/input-dataset/1914:0+1222602
21/05/22 16:55:07 INFO mapreduce.Job: Job job_local86310316_0001 running in uber mode : false
21/05/22 16:55:07 INFO mapreduce.Job: map 0% reduce 0%
21/05/22 16:55:07 INFO mapred.MapTask: (EQUATOR) 0 kvl 26214396(104857584)
21/05/22 16:55:07 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
21/05/22 16:55:07 INFO mapred.MapTask: soft limit at 83886080
21/05/22 16:55:07 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
21/05/22 16:55:07 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
21/05/22 16:55:07 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
21/05/22 16:55:08 INFO mapred.LocalJobRunner:
21/05/22 16:55:08 INFO mapred.MapTask: Starting flush of map output
21/05/22 16:55:08 INFO mapred.MapTask: Spilling map output
21/05/22 16:55:08 INFO mapred.MapTask: bufstart = 0; bufend = 78696; bufvoid = 104857600
21/05/22 16:55:08 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 26179424(104717696); length = 34973/6553600
21/05/22 16:55:08 INFO mapred.MapTask: Finished spill 0
21/05/22 16:55:08 INFO mapred.Task: Task:attempt_local86310316_0001_m_000000_0 is done. And is in the process of committing
21/05/22 16:55:08 INFO mapred.LocalJobRunner: map
21/05/22 16:55:08 INFO mapred.Task: Task 'attempt_local86310316_0001_m_000000_0' done.
21/05/22 16:55:08 INFO mapred.LocalJobRunner: Finishing task: attempt_local86310316_0001_m_000000_0
21/05/22 16:55:09 INFO mapred.LocalJobRunner: Starting task: attempt_local86310316_0001_m_000001_0
21/05/22 16:55:09 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
21/05/22 16:55:09 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup_temporary folders under output directory:false, ignore cleanup failures: false
21/05/22 16:55:09 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
21/05/22 16:55:09 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/Weather/input-dataset/1913:0+1222051
21/05/22 16:55:09 INFO mapred.MapTask: (EQUATOR) 0 kvl 26214396(104857584)
21/05/22 16:55:09 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
21/05/22 16:55:09 INFO mapred.MapTask: soft limit at 83886080
21/05/22 16:55:09 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
```

```
21/05/22 16:55:25 INFO mapred.LocalJobRunner: Finishing task: attempt_local86310316_0001_r_000000_0
21/05/22 16:55:25 INFO mapred.LocalJobRunner: reduce task executor complete.
21/05/22 16:55:26 INFO mapreduce.Job: map 100% reduce 100%
21/05/22 16:55:26 INFO mapreduce.Job: Job job_local86310316_0001 completed successfully
21/05/22 16:55:26 INFO mapreduce.Job: Counters: 35

File System Counters
  FILE: Number of bytes read=3649600
  FILE: Number of bytes written=31355690
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=248529156
  HDFS: Number of bytes written=380
  HDFS: Number of read operations=526
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=23

Map-Reduce Framework
  Map input records=142622
  Map output records=142462
  Map output bytes=1282158
  Map output materialized bytes=1567202
  Input split bytes=2260
  Combine input records=0
  Combine output records=0
  Reduce input groups=20
  Reduce shuffle bytes=1567202
  Reduce input records=142462
  Reduce output records=40
  Spilled Records=284924
  Shuffled Maps =20
  Failed Shuffles=0
  Merged Map outputs=20
  GC time elapsed (ms)=2558
  Total committed heap usage (bytes)=3855314944

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=19630411
File Output Format Counters
  Bytes Written=380

Maximum temperature : 37.8 in the year 1919
Minimum temperature : -47.8 in the year 1917
```