Practical - 6

2CS702 – Big Data Analytics

Harshit Gajipara
19BCE059

Aim:

Analyze impact of different number of mapper and reducer on same definition as practical 4.

Steps:

- Create a new project using Ant in apache netbeans
- Add libraries from below path to library and set jdk as 1.8.0

"D:/BDA/hadoop-3.2.1/share/hadoop/common/hadoop-common-3.2.1.jar"

"D:/BDA/hadoop-3.2.1/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.2.1.jar"

- After saving java files, clean and build the project and run hadoop command after uploading wordfile.txt file.

Hadoop jar "C:\Users\HARSHIT\Documents\NetBeansProjects\practical6\dist\practical6.jar" /prac6/wordfile.txt /prac6/output

- Now run this command to view the output

Hdfs dfs –cat /prac6/output/part-r-00000

Code:

Practical6.java

```
package practical6;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
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 Practical6 {
    public static void main(String[] args) throws
IOException, InterruptedException, ClassNotFoundException {
        if (args.length != 2)
        {
            System.err.println("Please specify the input and
output path");
            System.exit(-1);
        }

        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(Practical6.class);
        job.setMapperClass(map.class);
        //job.setCombinerClass(WCReduce.class);
        job.setReducerClass(reduce.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setNumReduceTasks(0);
        //job.setNumReduceTasks(2);

        FileInputFormat.addInputPath(job, new
Path(args[0]));
        FileOutputFormat.setOutputPath(job, new
Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
```

```
}
```

Map.java

```
package practical6;

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 map extends Mapper<LongWritable, Text, Text,
IntWritable> {
    @Override
    protected void map(LongWritable key, Text value,
Mapper.Context context) throws IOException,
InterruptedException {

        String data[] = value.toString().split(" ");

        for(String x:data)
            context.write(new Text(x), new IntWritable(1));

    }
}
```

Reduce.java

```
package practical6;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class reduce extends Reducer<Text, IntWritable, Text,
IntWritable>{
```

```
    @Override
    protected void reduce(Text key, Iterable<IntWritable>
values, Context context) throws IOException,
InterruptedException {
        int sum = 0;

        for(IntWritable x:values)
        {
            sum += x.get();
        }

        context.write(key, new IntWritable(sum));
    }
}
```

Output:

Number of reducers = 2

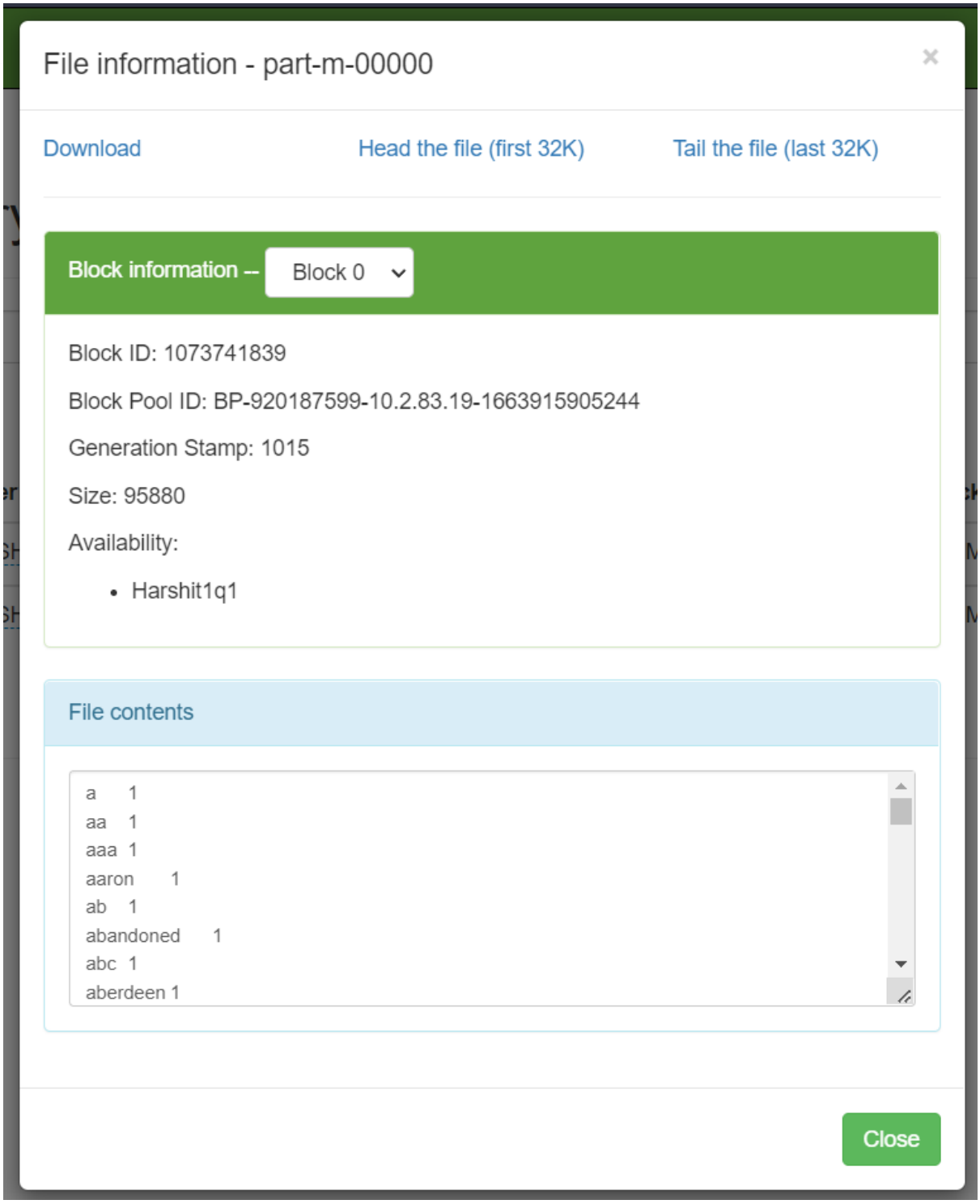
```
Command Prompt

C:\Users\HARSHIT>hdfs dfs -ls /prac6
Found 1 items
-rw-r--r--   3 HARSHIT supergroup      85878 2022-10-21 14:15 /prac6/wordfile.txt

C:\Users\HARSHIT>hadoop jar C:\Users\HARSHIT\Documents\NetBeansProjects\practical6\dist\practical6.jar /prac6/wordfile
.txt /prac6/output
2022-11-25 09:39:38,366 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2022-11-25 09:39:38,466 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2022-11-25 09:39:38,466 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2022-11-25 09:39:38,890 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implemen
t the Tool interface and execute your application with ToolRunner to remedy this.
2022-11-25 09:39:38,968 INFO input.FileInputFormat: Total input files to process : 1
2022-11-25 09:39:39,032 INFO mapreduce.JobSubmitter: number of splits:1
2022-11-25 09:39:39,136 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local469982681_0001
2022-11-25 09:39:39,136 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-11-25 09:39:39,291 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2022-11-25 09:39:39,293 INFO mapreduce.Job: Running job: job_local469982681_0001
2022-11-25 09:39:39,294 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2022-11-25 09:39:39,300 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-11-25 09:39:39,301 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under out
put directory:false, ignore cleanup failures: false
2022-11-25 09:39:39,303 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutp
utCommitter
2022-11-25 09:39:39,828 INFO mapred.LocalJobRunner: Waiting for map tasks
2022-11-25 09:39:39,829 INFO mapred.LocalJobRunner: Starting task: attempt_local469982681_0001_m_000000_0
2022-11-25 09:39:39,855 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-11-25 09:39:39,856 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under out
put directory:false, ignore cleanup failures: false
```

hdfs dfs -cat /prac6/output/part-m-00000

```
two     1
tx      1
ty      1
tyler   1
type    1
types   1
typical 1
typically       1
typing  1
u       1
uc      1
uganda  1
ugly    1
```



Now, we can change number of reducers and accordingly our time for analysis will change. Here, we have tried with number of reducers as 2 and 0, output will be same but time will be different.

Methodology:

Mappers

Hadoop Mapper is a function or task used to process all input records from a file and generate the output that works as input for Reducer.

No. of Mapper= {(total data size)/ (input split size)}

1) By Default, in Hadoop, the input size split is 128 MB. However, we can change the same on reducing the size, the number of mappers will increase.

set mapreduce.input.fileinputformat.split.maxsize=100000;

for example, block size is 100 MB and input of 1 GB, 10 mappers will be used.

2) We can also set the number of mappers as in the driver code

Job.setNumMapTask(5) which sets 5 number of mappers

3) While executing the job

hadoop jar /harshit-starterkit/mapreduce/
mapred.reduce.tasks=10

/user/harshit/input/example/output/mapreduce/example

Reducers

Reducer in Hadoop MapReduce reduces a set of intermediate values which share a key to a smaller set of values. In MapReduce job execution flow, Reducer takes a set of an intermediate key-value pair produced by the mapper as the input.

As given on the website of Hadoop the ideal number of Reducers are (.95 or 1.75) * (nodes * number of mappers)

1) We can also set the number of reducers as

Job.setNumReduceTask(5) which sets 5 number of reducers

2) Using command line

jar word_count.jar com.home.wc.WordCount /input /output \ -D mapred.reduce.tasks = 20 which sets the number of the reducers to 20

Suppose there are 100 reduce slots available in the cluster.

When we consider .95 as the factor, there will be 95 reducers and it means that no task will be waiting. This is to be done when all the tasks take equal number of times.

When we take 1.75 as the factor, 100 tasks will start simultaneously, while 75 will be in the queue. This will allow better load balancing as it would prevent bottlenecks of the jobs.