

EX:NO:05	SALES COUNT PROGRAM USING MAPREDUCE
DATE:	

AIM:

To implement a MapReduce program in Hadoop to count sales records per country from a given dataset.

PROCEDURE:

Step 1:

Before running the MapReduce program, start the necessary Hadoop services.

```
(durgah@Kali)-[/usr/local/hadoop/sbin]
$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as durgah in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [Kali]
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Starting resourcemanager
Starting nodemanagers

(durgah@Kali)-[/usr/local/hadoop/sbin]
$ jps
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
13408 ResourceManager
13142 SecondaryNameNode
13527 NodeManager
12808 NameNode
12920 DataNode
13946 Jps
```

Step 2: Prepare the Sales Data (Input File)

Create a text file named sales.txt containing sales records. Each row represents a sale with fields like date, product, cost, payment type, country.

Example Content (sales.txt):

```
2024-04-01,TV,1000,Credit Card,USA
2024-04-01,Phone,800,Debit Card,India
2024-04-01,Laptop,1200,Cash,India
2024-04-01,Tablet,500,Credit Card,Canada
2024-04-01,Camera,700,Cash,USA
```

Step 3: Upload Data to HDFS

```
(durgah@Kali)-[~]
$ sudo mkdir MapReduceTut
[sudo] password for durgah:

(durgah@Kali)-[~]
$ sudo chmod -R 777 MapReduceTut
```

- `hdfs dfs -mkdir /mapreduce_input_sales`
 - The first command creates a directory in HDFS
- `hdfs dfs -put sales.txt /mapreduce_input_sales/`
 - The second command uploads the sales data file.

Step 4: Write the MapReduce Code

Mapper Code (SalesMapper.java)

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class SalesMapper extends Mapper<Object, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text country = new Text();
    public void map(Object key, Text value, Context context) throws IOException,
    InterruptedException {
        String[] fields = value.toString().split(",");
        if (fields.length == 5) { // Ensure correct format
            country.set(fields[4].trim()); // Extract country
            context.write(country, one); // Emit (Country, 1)
        }
    }
}
```

Reducer Code (SalesReducer.java)

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class SalesReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
    InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```

Driver Code (SalesDriver.java)

```
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 SalesDriver {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
```

```
Job job = Job.getInstance(conf, "Sales Count");
```

```
job.setJarByClass(SalesDriver.class);
job.setMapperClass(SalesMapper.class);
job.setReducerClass(SalesReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

Step 5: Compile and Create a JAR File

```
(durgah@Kali)-[/usr/local/hadoop/sbin]
$ hadoop jar ProductSalePerCountry.jar SalesCountry.SalesCountryDriver /inputMapReduce /mapreduce_output_sales

Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
JAR does not exist or is not a normal file: /usr/local/hadoop/sbin/ProductSalePerCountry.jar

(durgah@Kali)-[/usr/local/hadoop/sbin]
$ ls -lh ProductSalePerCountry.jar

ls: cannot access 'ProductSalePerCountry.jar': No such file or directory

(durgah@Kali)-[/usr/local/hadoop/sbin]
$ cd ~/MapReduceTut

(durgah@Kali)-[~/MapReduceTut]
$ hadoop com.sun.tools.javac.Main SalesCountryDriver.java SalesMapper.java SalesCountryReducer.java

Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true

(durgah@Kali)-[~/MapReduceTut]
$ jar -cvf ProductSalePerCountry.jar *.class

Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
added manifest
adding: SalesCountryDriver.class(in = 1374) (out= 749)(deflated 45%)
adding: SalesCountryReducer.class(in = 1605) (out= 669)(deflated 58%)
adding: SalesMapper.class(in = 1691) (out= 701)(deflated 58%)
```

```
(durgah@Kali)-[~/MapReduceTut]
$ hadoop jar ProductSalePerCountry.jar SalesCountryDriver /inputMapReduce /mapreduce_output_sales

Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
2025-04-03 12:58:58,303 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2025-04-03 12:58:58,418 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2025-04-03 12:58:58,419 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2025-04-03 12:58:58,574 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2025-04-03 12:58:58,940 INFO input.FileInputFormat: Total input files to process : 1
2025-04-03 12:58:58,984 INFO mapreduce.JobSubmitter: number of splits:1
2025-04-03 12:58:59,217 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1559068103_0001
2025-04-03 12:58:59,217 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-04-03 12:58:59,371 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2025-04-03 12:58:59,372 INFO mapreduce.Job: job_local1559068103_0001
2025-04-03 12:58:59,458 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2025-04-03 12:58:59,494 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
2025-04-03 12:58:59,496 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2025-04-03 12:58:59,497 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2025-04-03 12:58:59,501 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2025-04-03 12:58:59,604 INFO mapred.LocalJobRunner: Waiting for map tasks
2025-04-03 12:58:59,605 INFO mapred.LocalJobRunner: Starting task: attempt_local1559068103_0001_m_000000_0
2025-04-03 12:58:59,651 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
2025-04-03 12:58:59,655 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2025-04-03 12:58:59,659 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2025-04-03 12:58:59,698 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
2025-04-03 12:58:59,712 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/inputMapReduce/sales.csv:0+57
2025-04-03 12:58:59,825 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2025-04-03 12:58:59,825 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2025-04-03 12:58:59,825 INFO mapred.MapTask: soft limit at 8386080
2025-04-03 12:58:59,825 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2025-04-03 12:58:59,825 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2025-04-03 12:58:59,860 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputTBuffer
2025-04-03 12:59:00,401 INFO mapreduce.Job: Job job_local1559068103_0001 running in uber mode : false
2025-04-03 12:59:00,408 INFO mapreduce.Job: map 0% reduce 0%
2025-04-03 12:59:00,518 INFO mapred.LocalJobRunner:
2025-04-03 12:59:00,528 INFO mapred.MapTask: Starting flush of map output
2025-04-03 12:59:00,529 INFO mapred.MapTask: Spilling map output
2025-04-03 12:59:00,529 INFO mapred.MapTask: bufstart = 0; bufend = 47; bufvoid = 104857600
2025-04-03 12:59:00,529 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 26214380(104857520); length = 17/6553600
2025-04-03 12:59:00,573 INFO mapred.MapTask: Finished spill 0
2025-04-03 12:59:00,636 INFO mapred.Task: Task:attempt_local1559068103_0001_m_000000_0 is done. And is in the process of committing
2025-04-03 12:59:00,657 INFO mapred.LocalJobRunner: map
```

Step 6: Run the MapReduce Job and Verify the output

```
File Actions Edit View Help
HDFS: Number of bytes read=114
HDFS: Number of bytes written=23
HDFS: Number of read operations=15
HDFS: Number of large read operations=0
HDFS: Number of write operations=4
HDFS: Number of bytes read erasure-coded=0
Map-Reduce Framework
  Map input records=5
  Map output records=5
  Map output bytes=47
  Map output materialized bytes=63
  Input split bytes=111
  Combine input records=0
  Combine output records=0
  Reduce input groups=3
  Reduce shuffle bytes=63
  Reduce input records=5
  Reduce output records=3
  Spilled Records=10
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=42
  Total committed heap usage (bytes)=413138944
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=57
File Output Format Counters
  Bytes Written=23

(durgah@Kali)-[~/MapReduceTut]
$ hdfs dfs -ls /mapreduce_output_sales
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Found 2 items
-rw-r--r-- 3 durgah supergroup          0 2025-04-03 12:59 /mapreduce_output_sales/_SUCCESS
-rw-r--r-- 3 durgah supergroup        23 2025-04-03 12:59 /mapreduce_output_sales/part-r-000000

(durgah@Kali)-[~/MapReduceTut]
$ hdfs dfs -cat /mapreduce_output_sales/part-r-000000
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Canada 1
India 2
USA 2
```

RESULT:

Thus, the implementation of MapReduce program in Hadoop to count sales records per country from a given dataset is done successfully.