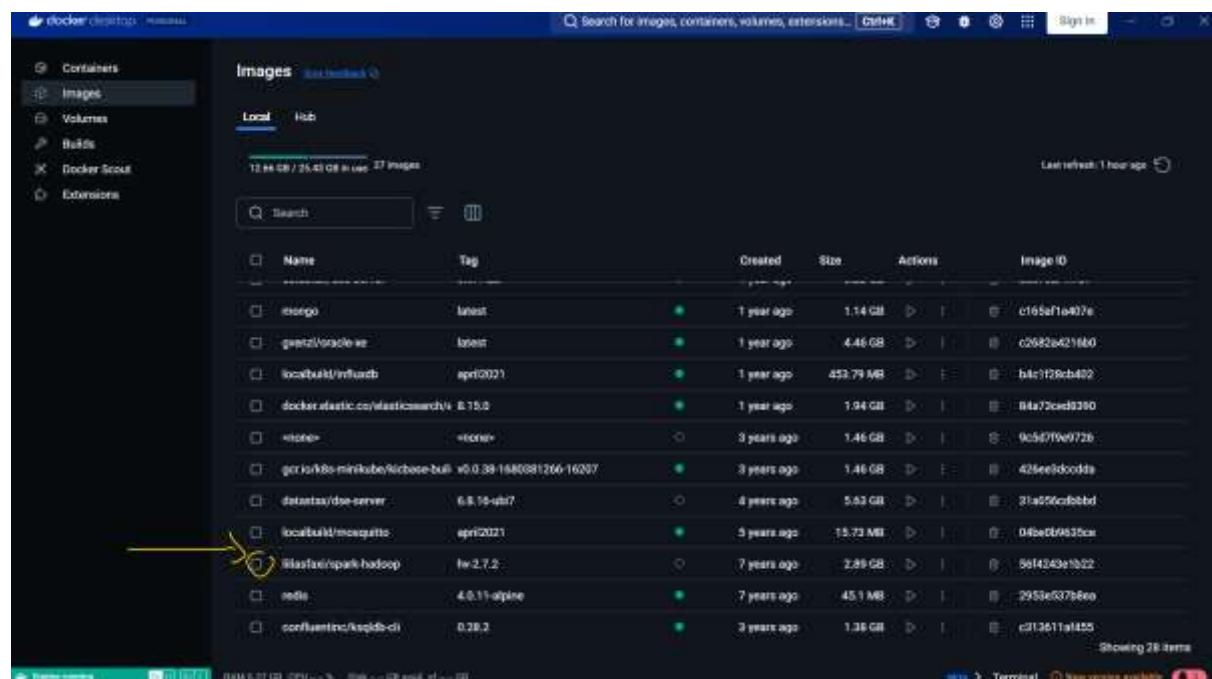


TP 1 BIG data

1. Télécharger l'image docker `liliasfaxy/spark-hadoop:hv-2.7.2` uploadée sur dockerhub.

La commande :

```
docker pull liliasfaxy/spark-hadoop:hv-2.7.2
```



2. Créer les trois conteneurs à partir de l'image téléchargée. Pour cela:

1. Créer un réseau bridge nommé hadoop qui permettra de relier les trois conteneurs.

```
C:\Users\T U F>docker network create --driver bridge hadoop  
b26e85a42a3c85ce123f7e0c2d072301215665ae65d4a84a6e802b2412f76074
```

```
C:\Users\T U F>
```

2. Créer les trois conteneurs à partir de l'image téléchargée. Pour cela:

1. Créer un réseau bridge nommé hadoop qui permettra de relier les trois conteneurs.
2. Créer et lancer les trois conteneurs :
 - Le conteneur hadoop-master expose les ports 50070, 8088 et 16010.
 - Les conteneurs hadoop-slave1 et hadoop-slave2 exposent le port 8042.

Pour le master on utilise les ports :

50070=>port pour HDFS (Le stockage distribué)

8088=> Port pour YARN (gestion de ressources)

16010 => port pour l'interface HBASE => pour voir l'état de la base de données distribuée

```
C:\Users\T U F>docker run -it --name hadoop-master --hostname hadoop-master --network hadoop -p 50070:50070 -p 50888:8088  
-p 56010:16010 liliasfaxy/spark-hadoop:hv-2.7.2 /bin/bash  
root@hadoop-master:~# |
```

Pour les 2 slaves :

```
Microsoft Windows [Version 10.0.26100.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\T U F>docker run -d --name hadoop-slave1 --hostname hadoop-slave1 ->
--network hadoop -p 8042:8042 liliexfazi/spark-hadoop:hv-2.7.2 /bin/bash -c "sleep infinity"
8fb6819048edca9b30293ff5983b548a88bb73c57e3a3528bd1d796ce5f976
C:\Users\T U F>

Microsoft Windows [Version 10.0.26100.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\T U F>docker run -d --name hadoop-slave2 --hostname hadoop-slave2 ->
--network hadoop -p 8043:8042 liliexfazi/spark-hadoop:hv-2.7.2 /bin/bash -c "sleep infinity"
c238bc9794459b2f238f04128c360ea59d7f2a57a26cb861ee7fc5e3431a9ac+
C:\Users\T U F>
```

C'est eux qu'ils vont executer les taches

le port 8042 => pour voir les ressources pris pour chaque slave et les conteneurs exécutés et les logs de chaque slave

ON VOIT LES CONTENEURS DANS DOCKER :

The screenshot shows the Docker desktop application interface. On the left, there's a sidebar with navigation links: Containers, Images, Volumes, Builds, Docker Scans, and Extensions. The main area is titled 'Containers' and shows a list of currently running containers. Each container entry includes a status icon, the container name, Container ID, Image name, Port(s), CPU usage, Last started time, and Actions (stop, start, logs). A yellow bracket on the left side of the list highlights the first three entries: 'hadoop-master', 'hadoop-slave1', and 'hadoop-slave2'. The 'hadoop-slave1' and 'hadoop-slave2' entries are grouped together by a yellow bracket. At the bottom of the interface, there are tabs for 'Single running' and 'All', along with system status indicators like RAM, CPU, and Disk usage.

3. Entrer dans le conteneur master pour commencer à l'utiliser.

On démarre le serveur ssh dans tous les conteneurs master et slave :

```
root@hadoop-master:~# service ssh start
 * Starting OpenBSD Secure Shell server sshd
root@hadoop-master:~# ./start-hadoop.sh
```

```
C:\Users\T U F>docker exec -it hadoop-slave1 /bin/bash
root@hadoop-slave1:~# service ssh start
 * Starting OpenBSD Secure Shell server sshd
root@hadoop-slave1:~# |
```

```
C:\Users\T U F>docker exec -it hadoop-slave2 /bin/bash
root@hadoop-slave2:~# service ssh start
 * Starting OpenBSD Secure Shell server sshd [ OK ]
root@hadoop-slave2:~# |
```

Puis on lance haddop et Yarn dans le conteneur Master :

```
root@hadoop-master:~# ./start-hadoop.sh

Starting namenodes on [hadoop-master]
hadoop-master: Warning: Permanently added 'hadoop-master,172.27.0.2' (ECDSA) to the list of known hosts.
hadoop-master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-root-namenode-hadoop-master.out
hadoop-slave1: Warning: Permanently added 'hadoop-slave1,172.27.0.3' (ECDSA) to the list of known hosts.
hadoop-slave2: Warning: Permanently added 'hadoop-slave2,172.27.0.4' (ECDSA) to the list of known hosts.
hadoop-slave2: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-hadoop-slave2.out
hadoop-slave1: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-hadoop-slave1.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of known hosts.
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-root-secondarynamenode-hadoop-master.out

starting yarn daemons
resourcemanager running as process 328. Stop it first.
hadoop-slave2: Warning: Permanently added 'hadoop-slave2,172.27.0.4' (ECDSA) to the list of known hosts.
hadoop-slave1: Warning: Permanently added 'hadoop-slave1,172.27.0.3' (ECDSA) to the list of known hosts.
hadoop-slave1: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-hadoop-slave1.out
hadoop-slave2: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-hadoop-slave2.out

root@hadoop-master:~# |
```

Premiers pas avec Hadoop :

- Créer un répertoire dans HDFS, appelé input.

```
root@hadoop-master:~# hdfs dfs -ls /
root@hadoop-master:~# hdfs dfs -mkdir /input
root@hadoop-master:~# hdfs dfs -ls /
Found 1 items
drwxr-xr-x  - root supergroup          0 2025-11-28 19:57 /input
root@hadoop-master:~# |
```

- Nous allons utiliser le fichier **purchases.txt** comme entrée pour le traitement MapReduce. Ce fichier se trouve déjà sous le répertoire principal de votre machine master.
- Charger le fichier **purchases** dans le répertoire input que vous avez créé.

Afficher le contenu du répertoire input.

+et

On telecharge le fichier et le transmettre vers le master

```
C:\Users\T U F>docker cp "C:\Users\T U F\Downloads\purchases.txt.gz" hadoop-master:/root/
Successfully copied 38.5MB to hadoop-master:/root/
C:\Users\T U F>
```

On copie le fichier dans le repertoire input crée :

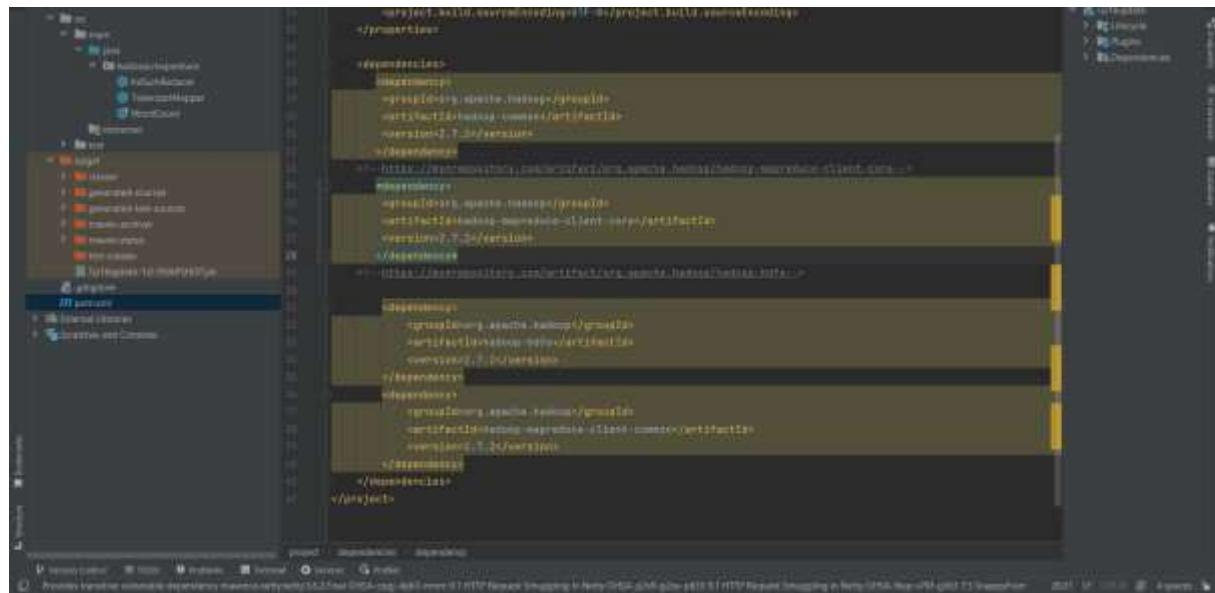
```
root@hadoop-master:~# ls
hdfs purchases.txt purchases2.txt run-wordcount.sh start-hadoop.sh start-kafka-zookeeper.sh
root@hadoop-master:~# ls /root
hdfs purchases.txt purchases.txt.gz purchases2.txt run-wordcount.sh start-hadoop.sh start-kafka-zookeeper.sh
root@hadoop-master:~# hdfs dfs -put -f /root/purchases.txt.gz /input/
root@hadoop-master:~# hdfs dfs -ls /input
Found 1 items
-rw-r--r-- 2 root supergroup 38454568 2025-11-28 20:09 /input/purchases.txt.gz
root@hadoop-master:~#
```

- Afficher les dernières lignes du fichier **purchases**.

```
root@hadoop-master:~# hdfs dfs -cat /input/purchases.txt.gz | gunzip -c | head -n 10
2012-01-01 09:00 San Jose Men's Clothing 214.05 Amex
2012-01-01 09:00 Fort Worth Women's Clothing 153.57 Visa
2012-01-01 09:00 San Diego Music 66.08 Cash
2012-01-01 09:00 Pittsburgh Pet Supplies 493.51 Discover
2012-01-01 09:00 Omaha Children's Clothing 235.63 MasterCard
2012-01-01 09:00 Stockton Men's Clothing 247.18 MasterCard
2012-01-01 09:00 Austin Cameras 379.6 Visa
2012-01-01 09:00 New York Consumer Electronics 296.8 Cash
2012-01-01 09:00 Corpus Christi Toys 25.38 Discover
2012-01-01 09:00 Fort Worth Toys 213.88 Visa
cat: Unable to write to output stream.
root@hadoop-master:~#
```

MAP REDUCE PROCESS :

1/ dans le fichier pom.xml on ajoute les dépendances demandées



- Créer la classe **TokenizerMapper** qui représente la classe MAP.

```
package hadoop.mapreduce;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
public class TokenizerMapper extends Mapper<Text, Text, IntWritable> {
    //Mapper : le type de la clé d'entrée (l'est un fichier texte) n'est l'offre de la ligne dans le fichier.
    //Mapper : le type de la valeur d'entrée (les lignes en texte).
    //Mapper : le type de la clé de sortie (les mots).
    //Mapper : le type de la valeur de sortie (les 1 pour compter chaque mot).
    public void map(Text key, Text value, Context context) throws IOException, InterruptedException {
        StringTokenizer itr = new StringTokenizer(value.toString());
        while(itr.hasMoreTokens()){
            String word = itr.nextToken();
            context.write(word, new IntWritable(1));
        }
    }
}
```

- Créez la classe **IntSumReducer** qui représente la classe REDUCE.

```
package hadoop.mapreduce;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
public class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    //Reduce : le type de la clé de sortie (la sort).
    //Reduce : le type des valeurs associées (qui sont des réduits).
    //Reduce : le type de la clé de sortie.
    //Reduce : le type de la valeur de sortie.
    protected 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));
    }
}
```

- Enfin, créez la classe **WordCount** qui représente la classe Driver.

```

WordCount.java
1 package org.apache.hadoop.mapreduce;
2 import org.apache.hadoop.conf.Configuration;
3 import org.apache.hadoop.fs.Path;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.mapreduce.Job;
6 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
7 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
8 import org.apache.hadoop.mapreduce.lib.reduce.IntSumReducer;
9 import org.apache.hadoop.mapreduce.lib.reduce.WordCountMapper;
10
11 public class WordCount {
12     public static void main(String[] args) throws Exception {
13         // Vérification du nombre d'arguments : 2 obligatoires (input + output)
14         if (args.length != 2) System.err.println("Usage: WordCount <input path> <output path>");
15         // Créeation de la configuration Hadoop
16         Configuration conf = new Configuration();
17         // Spécifie que les Reduce sont utilisés pour la combinaison
18         Job job = Job.getInstance(conf, "Word count");
19         // Spécifie la classe combiner IntSumReducer
20         job.setCombinerClass(IntSumReducer.class);
21         // Spécifie la classe Mapper WordCountMapper
22         job.setMapperClass(WordCountMapper.class);
23         // Spécifie la classe Reducer IntSumReducer
24         job.setReducerClass(IntSumReducer.class);
25         // Spécifie la clé de combiner IntSumReducer
26         job.setKeyClass(IntWritable.class);
27         // Spécifie la valeur de combiner IntWritable
28         job.setOutputValueClass(IntWritable.class);
29         // Spécifie le nom de sortie du répertoire
30         FileInputFormat.addInputPath(job, new Path(args[0]));
31         // Spécifie le nom du répertoire de sortie (output)
32         FileOutputFormat.setOutputPath(job, new Path(args[1]));
33         // Lance le job en exécutant la job
34         System.exit(job.waitForCompletion(true));
35     }
36 }

```

Dans votre projet IntelliJ :

- Générer le fichier **Jar** de l'application.
- Copier le fichier **Jar** créé dans le conteneur **master**.
- Revenir au shell du conteneur master et lancer le job map reduce sur le fichier purchases.txt que vous aviez préalablement chargé dans le répertoire input de HDFS..
- Afficher le contenu du fichier généré.

On fait:

Tp1bigdata/target/Tp1bigdata-1.0-SNAPSHOT.jar

Le fichier Jar est généré:

```

WordCount.java
1 package org.apache.hadoop.mapreduce;
2 import org.apache.hadoop.conf.Configuration;
3 import org.apache.hadoop.fs.Path;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.mapreduce.Job;
6 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
7 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
8 import org.apache.hadoop.mapreduce.lib.reduce.IntSumReducer;
9 import org.apache.hadoop.mapreduce.lib.reduce.WordCountMapper;
10
11 public class WordCount {
12     public static void main(String[] args) throws Exception {
13         // Vérification du nombre d'arguments : 2 obligatoires (input + output)
14         if (args.length != 2) System.err.println("Usage: WordCount <input path> <output path>");
15         // Créeation de la configuration Hadoop
16         Configuration conf = new Configuration();
17         // Créeation d'un job MapReduce à partir de la configuration
18         Job job = Job.getInstance(conf, "Word count");
19         // Spécifie la classe principale (celle qui contient le main)
20         job.setJarByClass(WordCount.class);
21         // Spécifie la classe Mapper utilisée
22         job.setMapperClass(WordCountMapper.class);
23         // Ici on réutilise le Reduce comme Combiner
24         job.setCombinerClass(IntSumReducer.class);
25         // Spécifie la classe Reducer utilisée
26         job.setReducerClass(IntSumReducer.class);
27         // Spécifie la clé de combiner IntSumReducer
28         job.setKeyClass(IntWritable.class);
29         // Spécifie la valeur de combiner IntWritable
30         job.setOutputValueClass(IntWritable.class);
31         // Spécifie le nom de sortie du répertoire
32         FileInputFormat.addInputPath(job, new Path(args[0]));
33         // Spécifie le nom du répertoire de sortie (output)
34         FileOutputFormat.setOutputPath(job, new Path(args[1]));
35         // Lance le job en exécutant la job
36         System.exit(job.waitForCompletion(true));
37     }
38 }

```

On copie le JAR vers le master :

```
PS C:\Users\T U F\IdeaProjects\Tpibigdata> docker cp target/Tplibigdata-1.0-SNAPSHOT.jar hadoop-master:/root/
Successfully copied 6.66kB to hadoop-master:/root/
PS C:\Users\T U F\IdeaProjects\Tpibigdata>
```

Maintenant on lance notre jib dans le conteneur laster

```
root@hadoop-master:~# ls
Tpibigdata-1.0-SNAPSHOT.jar  purchases.txt  purchases2.txt  start-hadoop.sh
[root@hadoop-master ~]# purchases.txt.gz run-mapreduce.sh start-hadoop-zookeeper.sh
root@hadoop-master:~# hadoop jar Tplibigdata-1.0-SNAPSHOT.jar hadoop.mapreduce /input/purchases.txt.gz /output
Exception in thread "main" java.lang.ClassNotFoundException: hadoop.mapreduce
    at java.net.URLClassLoader$1.run(URLClassLoader.java:362)
    at java.net.URLClassLoader.findClass(URLClassLoader.java:361)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
    at java.lang.Class.forName0(Native Method)
    at java.lang.Class.forName(Class.java:348)
    at org.apache.hadoop.util.RunJar.run(RunJar.java:214)
    at org.apache.hadoop.util.RunJar.main(RunJar.java:176)
root@hadoop-master:~# hadoop jar Tplibigdata-1.0-SNAPSHOT.jar hadoop.mapreduce.WordCount /input/purchases.txt.gz /output
25/11/28 21:19:43 INFO client.RMProxy: Connecting to ResourceManager at hadoop-master/172.27.0.2:8882
25/11/28 21:19:43 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
25/11/28 21:19:43 INFO input.FileInputFormat: Total input paths to process : 1
25/11/28 21:19:44 INFO mapreduce.JobSubmitter: number of splits:1
25/11/28 21:19:44 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1764358966858_0001
25/11/28 21:19:44 INFO impl.YarnClientImpl: Submitted application application_1764358966858_0001
25/11/28 21:19:44 INFO mapreduce.Job: The url to track the job: http://hadoop-master:8888/proxy/application_1764358966858_0001/
25/11/28 21:19:45 INFO mapreduce.Job: Running job: job_1764358966858_0001
25/11/28 21:19:51 INFO mapreduce.Job: Job job_1764358966858_0001 running in uber mode : false
25/11/28 21:19:51 INFO mapreduce.Job: map 0% reduce 0%
25/11/28 21:20:02 INFO mapreduce.Job: map 16% reduce 0%
25/11/28 21:20:05 INFO mapreduce.Job: map 23% reduce 0%
25/11/28 21:20:08 INFO mapreduce.Job: map 31% reduce 0%
25/11/28 21:20:11 INFO mapreduce.Job: map 38% reduce 0%
25/11/28 21:20:14 INFO mapreduce.Job: map 45% reduce 0%
25/11/28 21:20:17 INFO mapreduce.Job: map 52% reduce 0%
25/11/28 21:20:21 INFO mapreduce.Job: map 60% reduce 0%
25/11/28 21:20:30 INFO mapreduce.Job: map 67% reduce 0%
25/11/28 21:20:35 INFO mapreduce.Job: map 100% reduce 0%
25/11/28 21:20:39 INFO mapreduce.Job: map 100% reduce 100%
25/11/28 21:20:39 INFO mapreduce.Job: Job job_1764358966858_0001 completed successfully
25/11/28 21:20:39 INFO mapreduce.Job: Counters: 49
  File System Counters
    FILE: Number of bytes read=990951
    FILE: Number of bytes written=7837876
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
  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)=30933
    Total time spent by all reduces in occupied slots (ms)=1770
    Total time spent by all map tasks (ms)=38933
    Total time spent by all reduce tasks (ms)=3778
    Total vcore-milliseconds taken by all map tasks=38933
    Total vcore-milliseconds taken by all reduce tasks=3778
    Total megabyte-milliseconds taken by all map tasks=31675392
    Total megabyte-milliseconds taken by all reduce tasks=1828672
  Map-Reduce Framework
    Map input records=4138476
    Map output records=27982895
    Map bytes=323244588
    Map output materialized bytes=647726
    Input split bytes=111
    Combine input records=28681122
    Combine output records=569279
    Reduce input groups=51853
    Reduce shuffle bytes=647726
    Reduce input records=51853
    Reduce output records=51853
    Spilled Records=688332
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=285
    CPU time spent (ms)=35588
    Physical memory (bytes) snapshot=853177344
    Virtual memory (bytes) snapshot=2980143664
    Total committed heap usage (bytes)=327680000
```

```
root@hadoop-master:~# ./start-hadoop-zookeeper.sh
FILE: Number of write operations=0
HDFS: Number of bytes read=38845681
HDFS: Number of bytes written=990951
HDFS: Number of read operations=0
HDFS: Number of large read operations=0
HDFS: Number of write operations=0
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)=30933
Total time spent by all reduces in occupied slots (ms)=1770
Total time spent by all map tasks (ms)=38933
Total time spent by all reduce tasks (ms)=3778
Total vcore-milliseconds taken by all map tasks=38933
Total vcore-milliseconds taken by all reduce tasks=3778
Total megabyte-milliseconds taken by all map tasks=31675392
Total megabyte-milliseconds taken by all reduce tasks=1828672
Map-Reduce Framework
Map input records=4138476
Map output records=27982895
Map bytes=323244588
Map output materialized bytes=647726
Input split bytes=111
Combine input records=28681122
Combine output records=569279
Reduce input groups=51853
Reduce shuffle bytes=647726
Reduce input records=51853
Reduce output records=51853
Spilled Records=688332
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=285
CPU time spent (ms)=35588
Physical memory (bytes) snapshot=853177344
Virtual memory (bytes) snapshot=2980143664
Total committed heap usage (bytes)=327680000
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=38454568
File Output Format Counters
    Bytes Written=499048
root@hadoop-master:~# |
```

FINAL RESULT :

On execute :

```
hdfs dfs -cat /output/part-r-00000 | head -n 50
```

```
root@hadoop-master:~# hdfs dfs -ls /output
Found 2 items
-rw-r--r--  2 root supergroup          0 2025-11-28 21:20 /output/_SUCCESS
-rw-r--r--  2 root supergroup  499048 2025-11-28 21:20 /output/part-r-00000
root@hadoop-master:~# hdfs dfs -cat /output/part-r-00000 | head -n 50
0          86
0.01       78
0.02       83
0.03       80
0.04       89
0.05       82
0.06       81
0.07       70
0.08       85
0.09       84
0.1        81
0.11       87
0.12       97
0.13       83
0.14       75
0.15       90
0.16       86
0.17      102
0.18       78
0.19       78
0.2        76
0.21       87
0.22       89
0.23       91
0.24       94
0.25       81
0.26       79
0.27       78
0.28       69
0.29       83
0.3        104
0.31       89
0.32       95
0.33       79
0.34       70
0.35       88
0.36       79
0.37       86
0.38       76
0.39       72
0.4        75
0.41       86
0.42       89
0.43      102
0.44       80
0.45       78
0.46       78
0.47       88
0.48       88
```

```
0.17    102
0.18    78
0.19    78
0.2     76
0.21    87
0.22    89
0.23    91
0.24    94
0.25    81
0.26    79
0.27    78
0.28    69
0.29    83
0.3     104
0.31    89
0.32    95
0.33    79
0.34    70
0.35    88
0.36    79
0.37    86
0.38    76
0.39    72
0.4     75
0.41    86
0.42    89
0.43    102
0.44    80
0.45    78
0.46    78
0.47    88
0.48    88
0.49    92
cat: Unable to write to output stream.
root@hadoop-master:~# |
```