



PRÁCTICA HADOOP

PROGRAMACIÓN CONCURRENTE Y DISTRIBUIDA



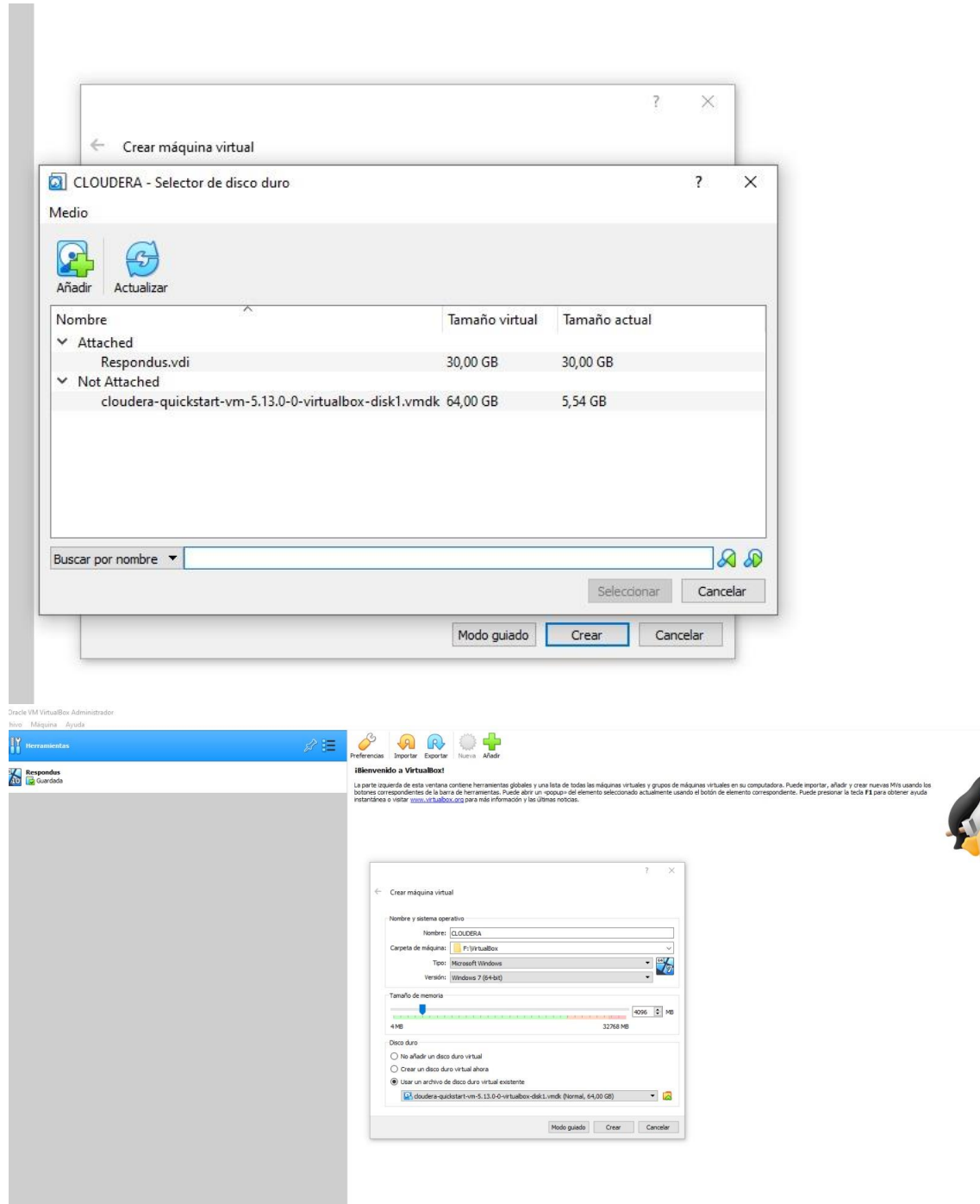
MARCOS ELADIO SOMOZA CORRAL
21711787





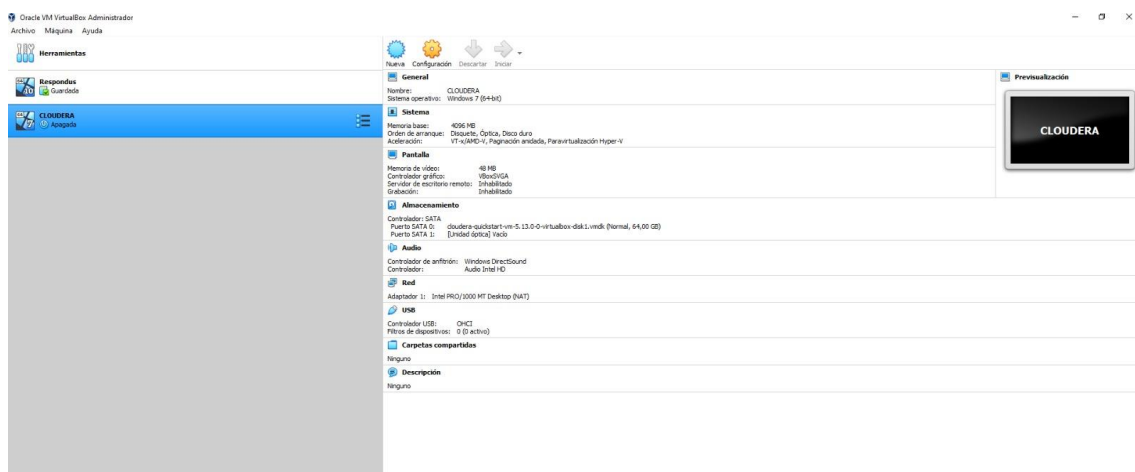
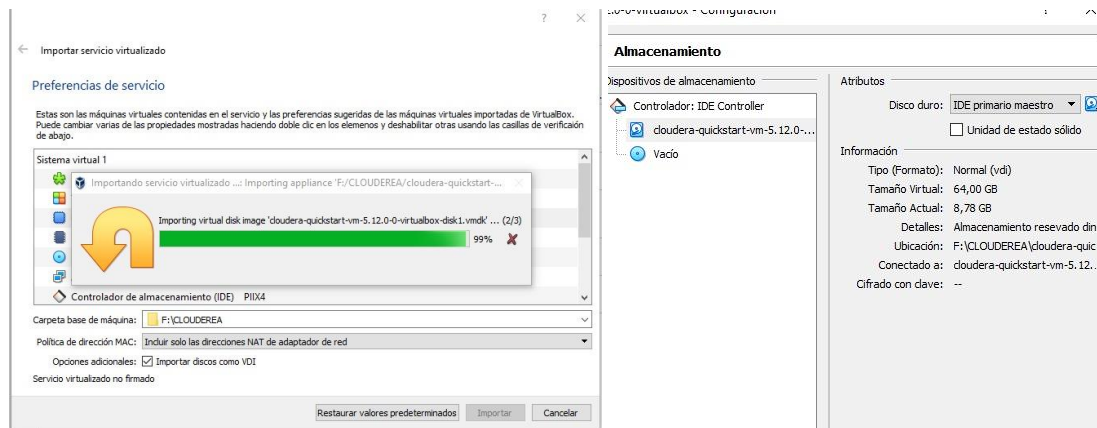
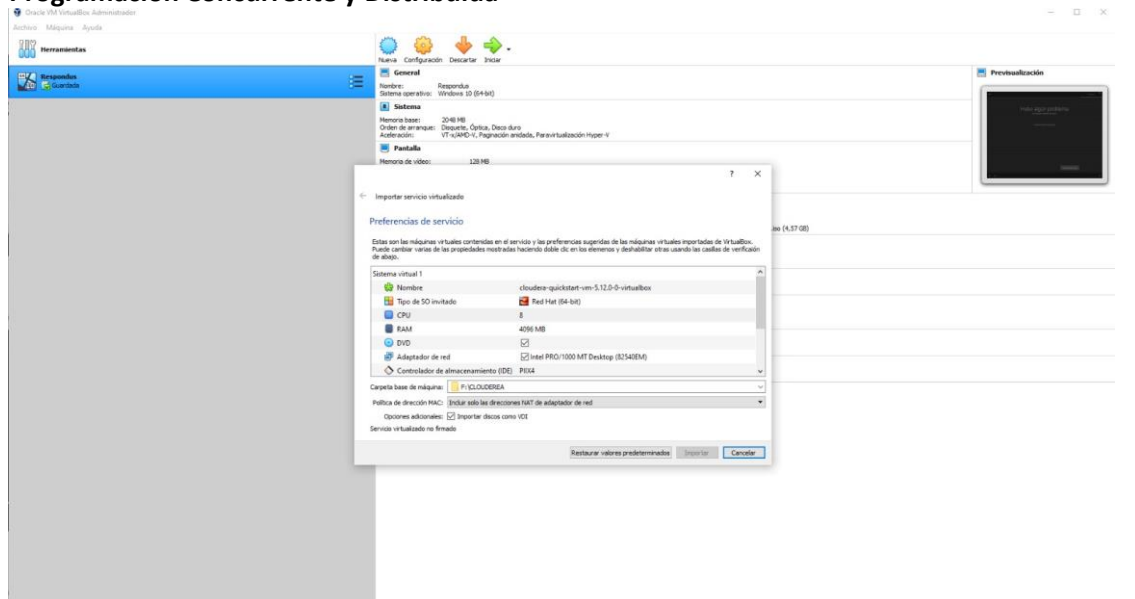
Programación Concurrente y Distribuida

El primer paso es instalar una máquina virtual, en mi caso, instalé Oracle VM VirtualBox. Una vez instalada, y teniendo el archivo vmdk de Cloudera procedemos a instalarlo en la máquina virtual. (https://downloads.cloudera.com/demo_vm/virtualbox/cloudera-quickstart-vm-5.12.0-0-virtualbox.zip)





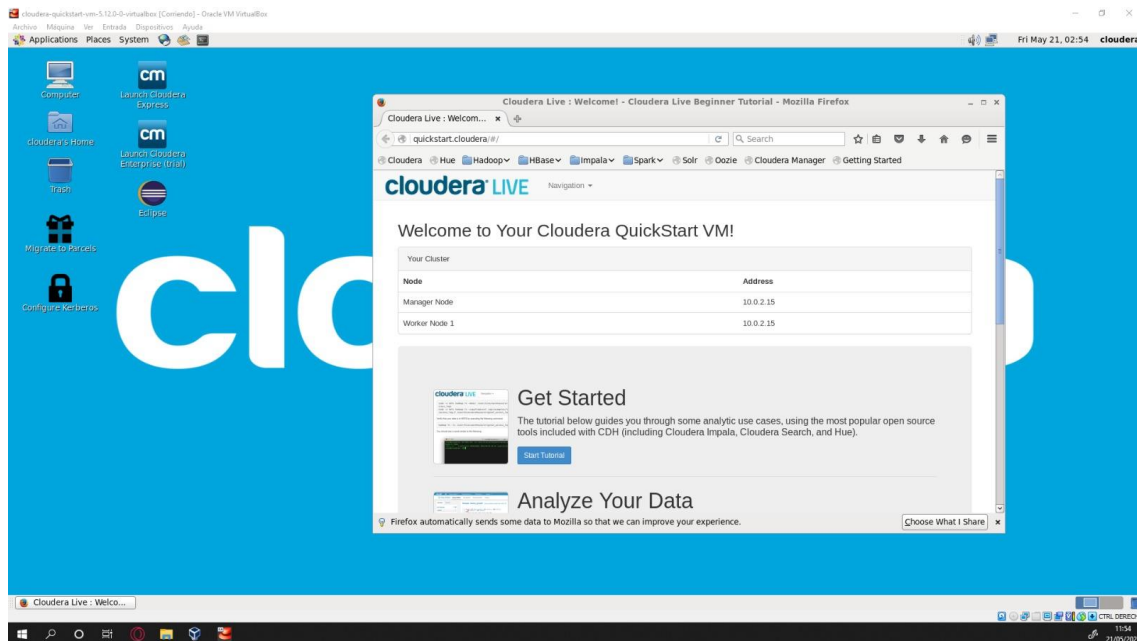
Programación Concurrente y Distribuida





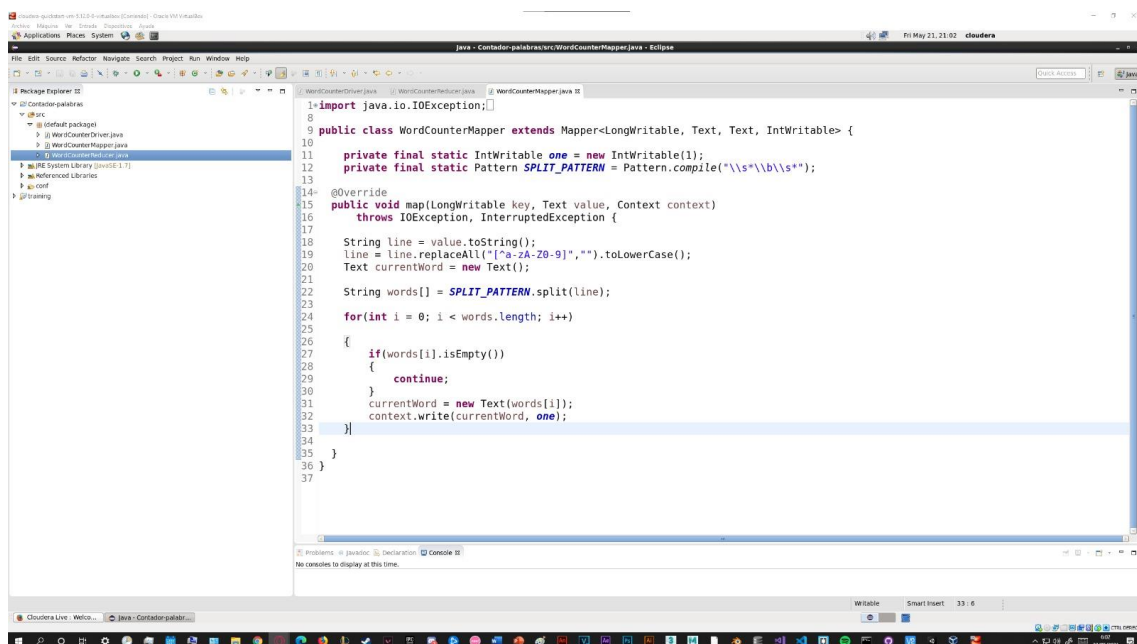
Programación Concurrente y Distribuida

Una vez instalado, iniciamos el sistema de Cloudera.



Ya en Cloudera, abriremos eclipse y crearemos los scripts necesarios para realizar el map reduce en hadoop.

Primero crearemos el script que hará de mapper (WordCounterMapper.java):





Programación Concurrente y Distribuida

Después el script que hará de reducer (WordCounterReducer):

```
1 import java.io.IOException;
2
3 public class WordCounterReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
4
5     @Override
6     public void reduce(Text key, Iterable<IntWritable> values, Context context)
7         throws IOException, InterruptedException {
8
9         int sum = 0;
10        for(IntWritable count : values)
11        {
12            sum += count.get();
13        }
14        context.write(key, new IntWritable(sum));
15    }
16 }
```

Y por último el driver donde estará ubicado el main(), en este controlaremos el Job correspondiente al mapreduce (WordCounterDriver):

```
1 import org.apache.hadoop.io.IntWritable;
2 import org.apache.hadoop.io.Text;
3 import org.apache.hadoop.mapreduce.Job;
4 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
5 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
6 import org.apache.hadoop.util.Tool;
7 import org.apache.hadoop.util.ToolRunner;
8
9 public class WordCounterDriver extends Configured implements Tool {
10
11     public static void main(String[] args) throws Exception {
12
13         int res = ToolRunner.run(new WordCounterDriver(), args);
14         System.exit(res);
15     }
16
17     public int run(String[] args) throws Exception {
18
19         Job job = Job.getInstance(getConf(), "WordCounter");
20         job.setJarByClass(this.getClass());
21
22         FileInputFormat.addInputPath(job, new Path(args[0]));
23         FileOutputFormat.setOutputPath(job, new Path(args[1]));
24
25         job.setMapperClass(WordCounterMapper.class);
26         job.setReducerClass(WordCounterReducer.class);
27
28         job.setOutputKeyClass(Text.class);
29         job.setOutputValueClass(IntWritable.class);
30
31         return job.waitForCompletion(true) ? 0 : 1;
32     }
33 }
```

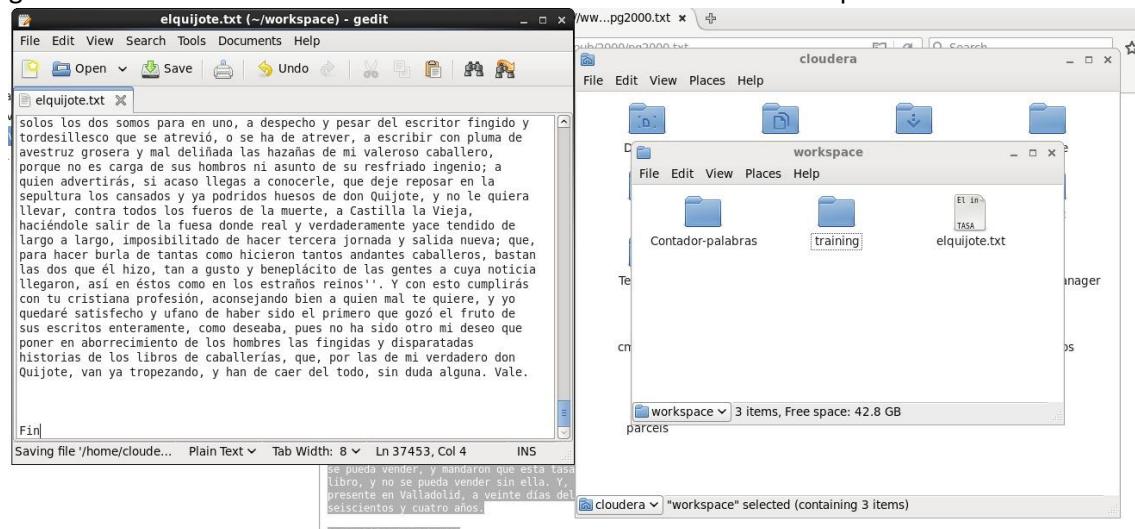


Programación Concurrente y Distribuida

Una vez creados los scripts, deberemos abrir la consola de comandos para configurar hadoop. Empezaremos por crear una carpeta con nuestro usuario y darle permisos con -chow.

```
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ sudo su hdfs
bash-4.1$ hadoop fs -mkdir /user/somozadev
bash-4.1$ hadoop fs -chown cloudera /user/somozadev
bash-4.1$
```

Después, buscaremos el texto al queremos aplicar map reduce (el quijote en este caso) y lo guardaremos en una carpeta local.





Programación Concurrente y Distribuida

Una vez guardado, iremos al directorio donde se encuentre nuestro proyecto de eclipse desde la consola y haremos una build del proyecto llamada "wordcounter.jar".

```
cloudera@quickstart:~/workspace/Contador-palabras
File Edit View Search Terminal Help
[cloudera@quickstart Contador-palabras]$ ls
bin conf elquijote.txt src
[cloudera@quickstart Contador-palabras]$ mkdir -p build
[cloudera@quickstart Contador-palabras]$ ls
bin build conf elquijote.txt src
[cloudera@quickstart Contador-palabras]$ sudo javac -cp /usr/lib/hadoop/*:/usr/lib/hadoop-mapreduce/* src/*.java -d build -Xlint
warning: [path] bad path element "/usr/lib/hadoop-mapreduce/jaxb-api.jar": no such file or directory
warning: [path] bad path element "/usr/lib/hadoop-mapreduce/activation.jar": no such file or directory
warning: [path] bad path element "/usr/lib/hadoop-mapreduce/jsr173 1.0 api.jar": no such file or directory
warning: [path] bad path element "/usr/lib/hadoop-mapreduce/jaxb1-impl.jar": no such file or directory
4 warnings
[cloudera@quickstart Contador-palabras]$ jar -cvf wordcounter.jar -C build/ .
added manifest
adding: WordCounterDriver.class(in = 1712) (out= 866)(deflated 49%)
adding: WordCounterMapper.class(in = 2197) (out= 972)(deflated 55%)
adding: WordCounterReducer.class(in = 1606) (out= 670)(deflated 58%)
```

Una vez creada la build, la ejecutaremos desde hadoop, pasando como parámetros el texto del quijote (previamente guardado en /user/somozadev/input) y el directorio destino donde queremos guardar el resultado del mapreduce (/user/somozadev/wordcounter/output).

```
[cloudera@quickstart Contador-palabras]$ sudo hadoop jar wordcounter.jar WordCounterDriver /user/somozadev/wordcounter/input /user/somozadev/wordcounter/output
21/05/21 21:20:28 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
21/05/21 21:20:36 INFO input.FileInputFormat: Total input paths to process : 1
21/05/21 21:20:37 INFO mapreduce.JobSubmitter: number of splits:1
21/05/21 21:20:39 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1621590485686_0001
21/05/21 21:20:51 INFO impl.YarnClientImpl: Submitted application application_1621590485686_0001
21/05/21 21:20:52 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1621590485686_0001/
21/05/21 21:20:52 INFO mapreduce.Job: Running job: job_1621590485686_0001
21/05/21 21:21:24 INFO mapreduce.Job: Job job_1621590485686_0001 running in uber mode : false
21/05/21 21:21:24 INFO mapreduce.Job: map 0% reduce 0%
21/05/21 21:21:32 INFO mapreduce.Job: map 100% reduce 0%
21/05/21 21:21:39 INFO mapreduce.Job: map 100% reduce 100%
21/05/21 21:21:41 INFO mapreduce.Job: Job job_1621590485686_0001 completed successfully
21/05/21 21:21:41 INFO mapreduce.Job: Counters: 49
File System Counters
  FILE: Number of bytes read=1020407
  FILE: Number of bytes written=3890793
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=2141662
  HDFS: Number of bytes written=1690873
  HDFS: Number of read operations=6
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
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)=6207
  Total time spent by all reduces in occupied slots (ms)=3757
  Total time spent by all map tasks (ms)=6207
  Total time spent by all reduce tasks (ms)=3757
  Total vcore-milliseconds taken by all map tasks=6207
```




Programación Concurrente y Distribuida

```
File Edit View Search Terminal Help
21/05/21 21:20:36 INFO input.FileInputFormat: Total input paths to process : 1
21/05/21 21:20:37 INFO mapreduce.JobSubmitter: number of splits:1
21/05/21 21:20:39 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1621590485686_0001
21/05/21 21:20:51 INFO impl.YarnClientImpl: Submitted application application_1621590485686_0001
21/05/21 21:20:52 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1621590485686_0001/
21/05/21 21:20:52 INFO mapreduce.Job: Running job: job_1621590485686_0001
21/05/21 21:21:24 INFO mapreduce.Job: Job job_1621590485686_0001 running in uber mode : false
21/05/21 21:21:24 INFO mapreduce.Job: map 0% reduce 0%
21/05/21 21:21:32 INFO mapreduce.Job: map 100% reduce 0%
21/05/21 21:21:39 INFO mapreduce.Job: map 100% reduce 100%
21/05/21 21:21:41 INFO mapreduce.Job: Job job_1621590485686_0001 completed successfully
21/05/21 21:21:41 INFO mapreduce.Job: Counters: 49
  File System Counters
    FILE: Number of bytes read=1820407
    FILE: Number of bytes written=3890793
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=2141662
    HDFS: Number of bytes written=1690873
    HDFS: Number of read operations=6
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
  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)=6207
    Total time spent by all reduces in occupied slots (ms)=3757
    Total time spent by all map tasks (ms)=6207
    Total time spent by all reduce tasks (ms)=3757
    Total vcore-milliseonds taken by all map tasks=6207
    Total vcore-milliseonds taken by all reduce tasks=3757
    Total megabyte-milliseonds taken by all map tasks=6355968
    Total megabyte-milliseonds taken by all reduce tasks=3847168
  Map-Reduce Framework
    Map input records=37453
    Map output records=31619
    Map output bytes=1757163
    Map output materialized bytes=1820407
    Input split bytes=143
    Combine input records=0
    Combine output records=0
    Reduce input groups=31393
    Reduce shuffle bytes=1820407
    Reduce input records=31619
    Reduce output records=31393
    Spilled Records=63238
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=591
    CPU time spent (ms)=5980
    Physical memory (bytes) snapshot=658481152
    Virtual memory (bytes) snapshot=3170938880
    Total committed heap usage (bytes)=649068544
  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=2141519
  File Output Format Counters
    Bytes Written=1690873
[cloudera@quickstart Contador-palabras]$ ss
```

Y como se puede ver en la captura anterior, se ha realizado el mapreduce de forma correcta. Se conecta al resourcesManager, ejecuta el trabajo y empiezan los procesos de map y reduce. Dado que solo tenemos un único modo, los trabajos tienen que hacerse secuencialmente (se ve en la captura que primero se realizan el 0% de las tareas map y 0% de las tareas reduce. Luego pasa a 100% de tareas map y 0% de tareas reduce y finalmente 100% de cada). Es decir, se trata de una pseudo distribución.



Programación Concurrente y Distribuida

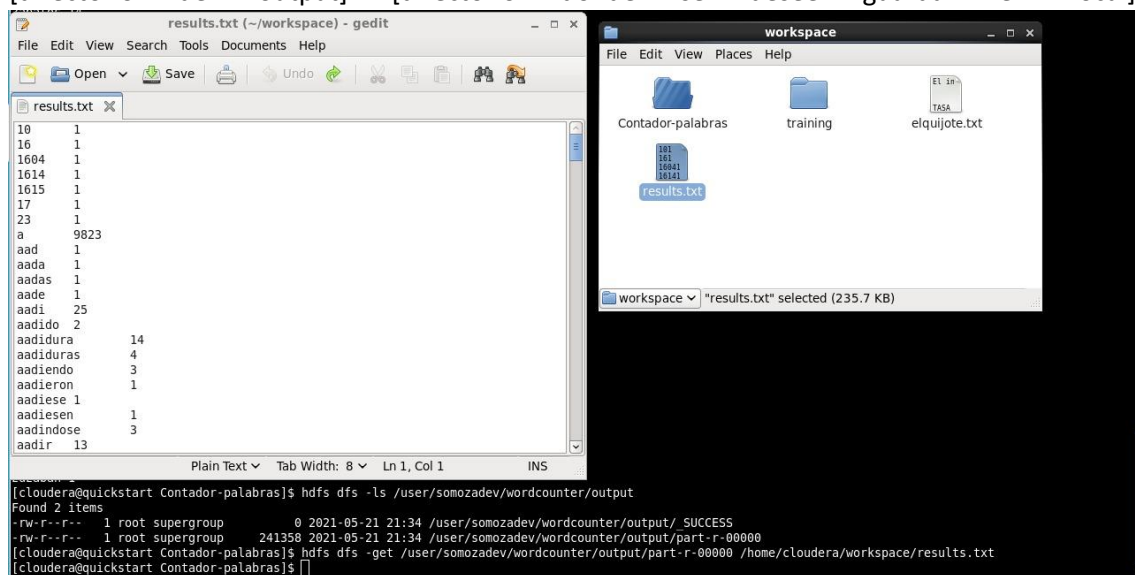
Finalmente, para ver el output podemos usar el comando `hadoop fs -cat` con el directorio para ver el contenido.

```
File Edit View Search Terminal Help
zafio 2
zaga 9
zagal 7
zagala 1
zagalas 6
zagalejas 1
zagales 4
zaharea 1
zaheriendo 1
zahor 1
zahrda 1
zal 2
zaleas 1
zalemas 3
zamarro 1
zamora 1
zamorana 1
zamoranas 1
zamorano 1
zampoas 1
zanahorias 1
zancadilla 2
zancajos 1
zancas 3
zanja 1
zanoguera 1
zapateadores 1
zapatear 1
zapateo 1
zapatero 3
zapatetas 2
zapaticos 1
zapatilla 1
zapatillas 2
zapato 8
zapatos 14
zaque 3
zaques 2
zaquizam 1
zaragoza 21
zarandajas 4
zarpasen 1
zarzas 3
zarzo 1
zas 1
zelador 1
zeuxis 1
znganos 1
zoca 1
zocodover 2
zodacos 1
zolo 1
zoltans 1
zonzorino 1
zopiro 1
zoraída 78
zoroastes 1
zorostrica 1
zorra 1
zorras 1
zorruna 1
zuecos 1
zulema 1
zumban 1
zurdo 2
zurrrn 1
zuzaban 1
[cloudera@quickstart Contador-palabras]$
```

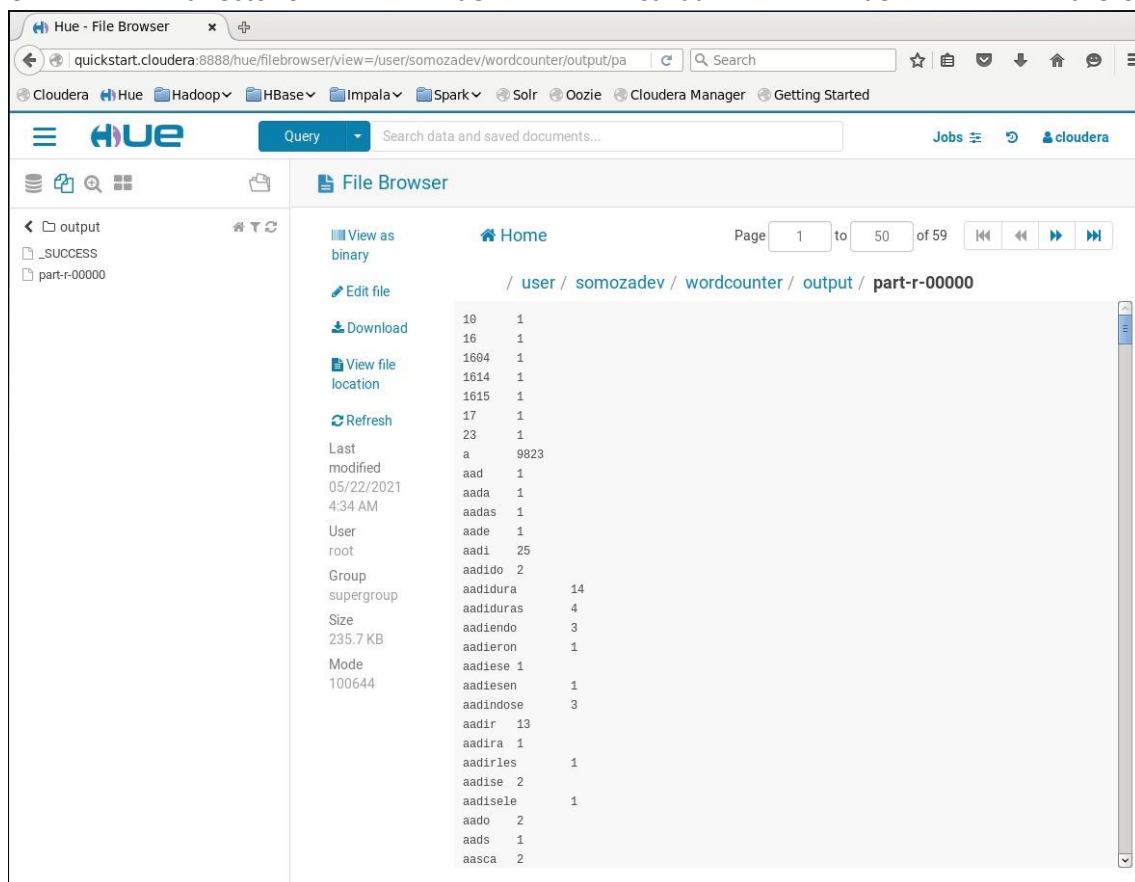


Programación Concurrente y Distribuida

De todas formas, este output no se encuentra de manera local en nuestro equipo, para poder traerlo a nuestra máquina debemos ubicarlo en el servidor hadoop y usar `hdfs dfs -get [directorio del output] [directorio donde se desee guardar en local]`.



O si se prefiere, acceder al hue en nuestro navegador (iniciando sesión con usr: cloudera y pass: cloudera) donde podemos acceder a los directorios hdfs y encontraremos de una manera visual el directorio de salida del fichero.





Programación Concurrente y Distribuida

Finalmente, si se desearía volver a realizar la ejecución de la aplicación, se deberá eliminar el directorio de salida para que no haya problemas a la hora de sobrescribir en él.

```
[cloudera@quickstart ~]$ sudo hadoop fs -rm -r -f /user/somozadev/wordcounter/output  
Deleted /user/somozadev/wordcounter/output  
[cloudera@quickstart ~]$
```