

EJERCICIOS DE HADOOP

NOMBRE ALUMNO: TANIA BATISTA

1. Devolver el resultado de ejecutar el siguiente comando (1 punto)
`$HADOOP_HOME/bin/hdfs dfs -cat /outHDFS/*`

RESPUESTA:

```
Esta 1
es 1
línea 1
una 1
```

2. ¿Cuál de los siguientes funcionalidades pertenecen a un NameNode de HDFS? (1 punto)
- a. Transferir bloques de datos de los datanodes a los clientes
 - b. Mantener el árbol del sistema de archivos y los metadatos de todos los ficheros y directorios
 - c. Controlar los procesos de Map Reduce
 - d. Almacenar bloques de datos
 - e. Ninguna de las opciones es correcta

RESPUESTA:

- b. Mantener el árbol del sistema de archivos y los metadatos de todos los ficheros y directorios

3. ¿Cuál de los siguientes funcionalidades pertenece a un DataNode de HDFS? (1 punto)
- a. Mantener el árbol del sistema de archivos y los metadatos de todos los ficheros y directorios.
 - b. Controlar la ejecución de una tarea de mapeo o de reduce individual.
 - c. Gestionar el sistema de espacios de nombres de los archivos.
 - d. Almacenar y recuperar bloques cuando los clientes o el NameNode lo solicita.
 - e. Ninguna de las opciones es correcta.

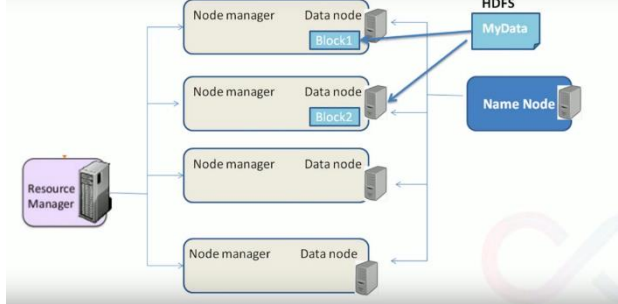
RESPUESTA:

- d. Almacenar y recuperar bloques cuando los clientes o el NameNode lo solicita.

4. ¿Cuál de las siguientes frases es cierta con respecto a YARN? (1 punto)
- a. Implementa un gestor de ficheros para todos los frameworks de Hadoop.
 - b. Permitir acceso a los datos de HDFS a programas que no estén desarrollados en Hadoop.
 - c. Permitir a múltiples Namenodes con sus propios namespaces, compartir el pool de Datanodes.
 - d. Usar el JournalNode para decidir el NameNode activo.
 - e. Ninguna de las anteriores es correcta.

RESPUESTA:

- a. Implementa un gestor de ficheros para todos los frameworks de Hadoop.
(Framework para planificación de tareas y gestión de recursos del cluster)



5. HDFS está diseñado para (1 punto)

- a. Ficheros grandes, acceso continuo a los datos y hardware de grandes prestaciones.
- b. Ficheros pequeños, acceso continuo a los datos y hardware básico.
- c. Ficheros grandes, baja latencia de acceso y hardware básico.
- d. Ficheros grandes, acceso continuo a los datos y hardware básico.
- e. Ninguna de las anteriores es correcta.

RESPUESTA:

- d. Ficheros grandes, acceso continuo a los datos y hardware básico.
(Pero el hardware no debe ser tan básico - veo que Hadoop está diseñado para "industry standard hardware")

6. ¿En cuál de los siguientes escenarios usarías Hadoop? (1 punto)

- a. Analizar los signos vitales de un bebé en tiempo real.
- b. Obtener las tendencias de acciones bursátiles cada minuto.
- c. Procesar un sensor meteorológico para predecir la trayectoria de un huracán.
- d. Procesar billones de mensajes de email para ejecutar análisis de texto.
- e. Ninguna de las anteriores.

RESPUESTA:

- d. Procesar billones de mensajes de email para ejecutar análisis de texto.
(Hadoop va demasiado lento para A y B. Pero quizás se podría usar para C, porque Hadoop es capaz de procesar datos de varios tipos, como quizás los datos de un arduino conectado a sensores meteorológicos. El único problema sería que Hadoop no es bueno para la analítica, ni para acceso random o interactivo. Entonces los datos meteorológicos o los mensajes de email deben ser *analizados* con otro data warehouse.)

7. ¿Cuál de las siguientes frases es cierta? (1 punto)

- a. Hadoop es una nueva tecnología diseñada para reemplazar las bases de datos relacionales.
- b. Hadoop incluye componentes open source y closed source.

- c. Hadoop se puede usar para bigdata, DSS y OLTP.
- d. Todas las anteriores son correctas.
- e. Ninguna de las anteriores es correcta.

RESPUESTA:

- e. Ninguna de las anteriores es correcta.

Explicación:

- a. Hadoop NO puede remplazar las bases de datos relacionales. (Tienen funciones diferentes).
- b. Hadoop incluye componentes open source, pero los componentes "closed source" solo se encuentran con la distribución M5 de MapR. Las otras distribuciones de Hadoop son open source.
- c. Hadoop se puede usar para bigdata y DSS, PERO NO para OLTP. "Hadoop doesn't provide any random access to the data stored in it's file. So we can't use Hadoop as an OLTP database which is characterized by INSERT - UPDATE - DELETE."
- d. Todas las anteriores son correctas

8. Describe con tus palabras (no usar wikipedia ni información por internet) el funcionamiento de los distintos demonios de HDFS (3 puntos)

RESPUESTA: (Mi explicación es en inglés, porque todavía no conozco muy bien el vocabulario técnico en español.)

Hadoop has 5 daemons, which always run in the background: NameNode, Secondary NameNode, DataNode, JobTeacker and TaskTracker. HDFS uses only the first 3.

NameNode: Master node, and directory tree of HDFS. Does not store copies of actual data, instead it just tracks where everything goes. As the highest node in the hierarchy, it a single point of failure - so it needs to be running all the time for the clusters to be found.

Secondary NameNode: It's like a backup node in case the NameNode fails, so that the clusters can still be found. It is not an exact copy of NameNode, because it merges the edits while creating checkpoints. It stores the directory image on a separate machine.

DataNode: this is the node that contains the actual data. It has replicates of the data so that files can be accessed in parallel. (For example, by the MapReduce TaskTrackers)

EJERCICIOS DE HDFS

A. WORD COUNT

Partiendo de las letras de las siguientes canciones

1. Crear ficheros de texto en la máquina ubuntu
2. Subir los ficheros a HDFS
3. Ejecutar un word count y resolver las preguntas del formulario.
 - A. Acceder al jobhistory e indicar los valores de "Status" y "Map Total" del job con el nombre "word count"
 - B. Palabra que se repite más veces
 - C. Palabra que se repite 6 veces
 - D. Cuántas veces aparece la palabra eyes

RESPUESTA 1. Crear ficheros de texto en la máquina ubuntu

Create 3 new files. For each file, copy the song text and paste it, using the nano command to open the text editor.

`nano /home/bigdata/ejemplos/Hadoop/MapReduce/wc-in/song1.txt`

```
bigdata@bigdata:~/hadoop$ nano /home/bigdata/ejemplos/Hadoop/MapReduce/wc-in/song1.txt
```

```
bigdata@bigdata: ~/hadoop
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
GNU nano 2.8.6 Archivo: /home/bigdata/ejemplos/Hadoop/MapReduce/wc-in/song1.txt
NINE INCH NAILS LYRICS "March Of The Pigs"
step right up march push
crawl right up on your knees
please greed feed (no time to hesitate)
I want a little bit I want a piece of it I think he's losing it
I want to watch it come down
don't like the look of it don't like the taste of it don't like the smell of it
I want to watch it come down
all the pigs are all lined up
I give you all that you want
take the skin and peel it back
now doesn't that make you feel better?
shove it up inside surprise! lies
stains like the blood on your teeth
bite chew suck away the tender parts
I want to break it up I want to smash it up I want to fuck it up
I want to watch it come down
maybe afraid of it let's discredit it let's pick away at it
I want to watch it come down
now doesn't that make you feel better?
the pigs have won tonight
now they can all sleep soundly
and everything is all right
```

`nano /home/bigdata/ejemplos/Hadoop/MapReduce/wc-in/song2.txt`

```
bigdata@bigdata:~/hadoop$ nano /home/bigdata/ejemplos/Hadoop/MapReduce/wc-in/song2.txt
```

```
Archivo: /home/bigdata/ejemplos/Hadoop/MapReduce/wc-in/song2.txt
```

```
PINK FLOYD "Pigs (Three Different Ones)"
Big man, pig man, ha ha, charade you are
```

```
nano /home/bigdata/ejemplos/Hadoop/MapReduce/wc-in/song3.txt
```

```
bigdata@bigdata:~/hadoop$ nano /home/bigdata/ejemplos/Hadoop/MapReduce/wc-in/song3.txt
```

```
Archivo: /home/bigdata/ejemplos/Hadoop/MapReduce/wc-in/song3.txt
```

```
DAVE MATTHEWS BAND "Pig"  
Isn't it strange  
How we move our lives for another day
```

RESPUESTA 2. Subir los ficheros a HDFS

Create the directory, carpetaWordCount. (previously done during exercises)

```
$ hdfs dfs -mkdir /user/bigdata/carpetaWordCount
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -mkdir /user/bigdata/carpetaWordCount
```

Upload the text files to the word counter directory

```
$ hdfs dfs -copyFromLocal /home/bigdata/ejemplos/Hadoop/MapReduce/wc-in /user/bigdata/carpetaWordCount
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -copyFromLocal /home/bigdata/ejemplos/Hadoop/MapReduce  
/wc-in /user/bigdata/carpetaWordCount
```

RESPUESTA 3. Ejecutar un word count y resolver las preguntas del formulario.

Execute a word count:

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar wordcount /user/bigdata/carpetaWordCount/wc-in/  
/user/bigdata/carpetaWordCount/wc-out
```

```
bigdata@bigdata:~/hadoop$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce  
-examples-2.9.0.jar wordcount /user/bigdata/carpetaWordCount/wc-in/ /user/bigdata/carpetaWordCount/wc-out
```

..wait for it to apply mapreduce

```
18/02/21 16:10:50 INFO client.RMPProxy: Connecting to ResourceManager at /0.0.0.0:8032  
18/02/21 16:10:51 INFO input.FileInputFormat: Total input files to process : 5  
18/02/21 16:10:51 INFO mapreduce.JobSubmitter: number of splits:5  
18/02/21 16:10:51 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
```

```

Job Counters
  Killed map tasks=1
  Launched map tasks=5
  Launched reduce tasks=1
  Data-local map tasks=5
  Total time spent by all maps in occupied slots (ms)=103024
  Total time spent by all reduces in occupied slots (ms)=4803
  Total time spent by all map tasks (ms)=103024
  Total time spent by all reduce tasks (ms)=4803
  Total vcore-milliseconds taken by all map tasks=103024
  Total vcore-milliseconds taken by all reduce tasks=4803
  Total megabyte-milliseconds taken by all map tasks=105496576
  Total megabyte-milliseconds taken by all reduce tasks=4918272
Map-Reduce Framework
  Map input records=140
  Map output records=870
  Map output bytes=7769
  Map output materialized bytes=4912
  Input split bytes=666
  Combine input records=870
  Combine output records=426
  Reduce input groups=369
  Reduce shuffle bytes=4912
  Reduce input records=426
  Reduce output records=369
  Spilled Records=852
  Shuffled Maps =5
  Failed Shuffles=0
  Merged Map outputs=5
  GC time elapsed (ms)=1837
  CPU time spent (ms)=3020
  Physical memory (bytes) snapshot=1387282432
  Virtual memory (bytes) snapshot=13658439680
  Total committed heap usage (bytes)=901644288
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=4293
File Output Format Counters
  Bytes Written=2837

```

*hdfs dfs -cat /user/bigdata/carpetaWordCount/wc-out/**

```

bigdata@bigdata:~/hadoop$ hdfs dfs -cat /user/bigdata/carpetaWordCount/wc-out/*
"March  1
"Pig"    1
"Pigs   1
"keep   1
(Three  1
(no     1
.....!.....!.....!.....!      1
All     1
Almost  2
And     7
BAND    1
Big     1

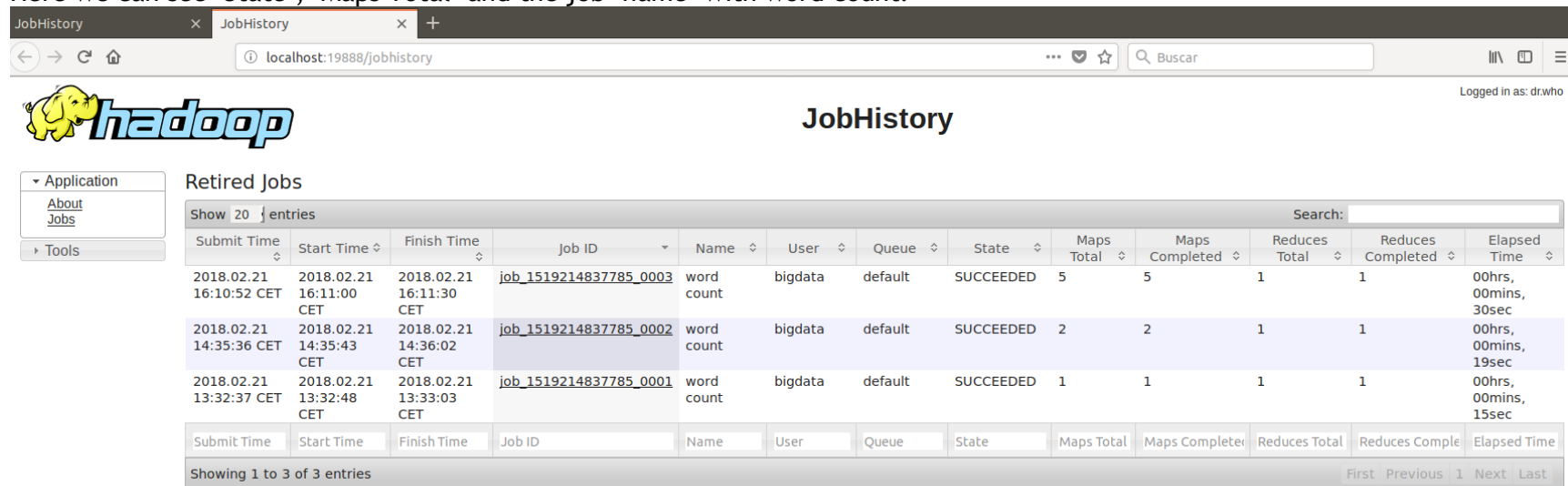
```

A. Acceder al jobhistory e indicar los valores de "Status" y "Map Total" del job con el nombre "word count"

Check that everything is running correctly:

```
bigdata@bigdata:~/hadoop$ jps
22497 SecondaryNameNode
18084 ResourceManager
18230 NodeManager
22102 NameNode
22281 DataNode
16811 JobHistoryServer
25148 Jps
```

To see details about jobs in MapReduce that have been executed, we go to <http://localhost:19888/jobhistory>
Here we can see "State", "Maps Total" and the job "name" with word count.



The screenshot shows the Hadoop JobHistory web interface. The browser address bar displays `localhost:19888/jobhistory`. The page title is "JobHistory". On the left, there is a sidebar with "Application" and "Tools" links. The main content area is titled "Retired Jobs" and displays a table of job details. The table has columns for Submit Time, Start Time, Finish Time, Job ID, Name, User, Queue, State, Maps Total, Maps Completed, Reduces Total, Reduces Completed, and Elapsed Time. Three jobs are listed, all with the name "word count" and state "SUCCEEDED".

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed	Elapsed Time
2018.02.21 16:10:52 CET	2018.02.21 16:11:00 CET	2018.02.21 16:11:30 CET	job_1519214837785_0003	word count	bigdata	default	SUCCEEDED	5	5	1	1	00hrs, 00mins, 30sec
2018.02.21 14:35:36 CET	2018.02.21 14:35:43 CET	2018.02.21 14:36:02 CET	job_1519214837785_0002	word count	bigdata	default	SUCCEEDED	2	2	1	1	00hrs, 00mins, 19sec
2018.02.21 13:32:37 CET	2018.02.21 13:32:48 CET	2018.02.21 13:33:03 CET	job_1519214837785_0001	word count	bigdata	default	SUCCEEDED	1	1	1	1	00hrs, 00mins, 15sec

Showing 1 to 3 of 3 entries

Use the `-cat` file function to create a new file named "songs"

```
hdfs dfs -cat /user/bigdata/carpetaWordCount/wc-out/* > songs.txt
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -cat /user/bigdata/carpetaWordCount/wc-out/* > songs.txt
```

B. Palabra que se repite más veces:

Apply "sort", one of the programs suggested under [hadoop-mapreduce-examples-2.9.0.jar](#)

n = return results in numerical order

r = return results in descending order

k 2 = which column we are applying this to

```
sort: A map/reduce program that sorts the data written by the random writer.
bigdata@bigdata:~/hadoop$ sort --help
-n, --numeric-sort          compare according to string numerical value
-r, --reverse               reverse the result of comparisons
    --sort=PALABRA          ordena de acuerdo con PALABRA:
                             general-numeric -g, human-numeric -h, month -M,
                             numeric -n, random -R, version -V
-k, --key=CLAVEDEF          ordena de acuerdo con una clave, CLAVEDEF establece
                             lugar y tipo
```

\$ cat songs.txt | sort -nrk 2

```
bigdata@bigdata:~/hadoop$ cat songs.txt | sort -nrk 2
the      33
it       24
a        22
your     18
to       15
of       15
```

Respuesta: THE aparece 33 veces

C. Palabra que se repite 6 veces

Apply "grep", one of the programs suggested under [hadoop-mapreduce-examples-2.9.0.jar](#)

```
grep: A map/reduce program that counts the matches of a regex in the input.
bigdata@bigdata:~/hadoop$ grep --help
Uso: grep [OPCIÓN]... PATRÓN [ARCHIVO]...
Search for PATTERN in each FILE.
Example: grep -i 'hello world' menu.h main.c
```


grep 6 songs.txt

```
bigdata@bigdata:~/hadoop$ grep 6 songs.txt
You      6
charade  6
ha       6
ha,      6
laugh    6
our      6
we       6
```

It looks like "ha" and "you" appear more than 6 times, under different punctuation and capitalization.

```
bigdata@bigdata:~/hadoop$ grep "you" songs.txt
you      14
you're   10
your     18
```

Respuesta: charade, laugh, our, we

D. Cuántas veces aparece la palabra eyes

grep "eyes" songs.txt

```
bigdata@bigdata:~/hadoop$ grep "eyes" songs.txt
eyes     3
eyes,    1
```

Respuesta: 4

B. SUDOKU

4. Ejecutar el programa sudoku con los siguientes parámetros:

- Programa: sudoku

```
sudoku: A sudoku solver.
```

\$ sudo apt install sudoku

```
bigdata@bigdata:~/hadoop$ sudo apt install sudoku
[sudo] password for bigdata:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  sudoku
```

- Path fichero

En la ruta del fichero habrá que indicar la ruta en la que se encuentre un fichero que tendrá la siguiente información:

```
? 4 6 3 ? ? 8 2 5
? ? ? ? ? 2 4 ? ?
8 2 ? ? 6 ? ? 7 ?
7 ? ? 4 ? ? ? ? 2
3 8 ? ? ? ? 6 ? 9
? 6 ? 2 8 ? ? ? ?
? 7 ? ? ? 5 ? 3 ?
5 3 8 ? ? 7 ? ? ?
? 9 ? ? ? ? ? ? 6
```

\$ nano sudoku.txt

```
bigdata@bigdata:~/hadoop$ nano sudoku.txt
```

```
GNU nano 2.8.6                               Archivo: sudoku.txt
? 4 6 3 ? ? 8 2 5
? ? ? ? ? 2 4 ? ?
8 2 ? ? 6 ? ? 7 ?
7 ? ? 4 ? ? ? ? 2
3 8 ? ? ? ? 6 ? 9
? 6 ? 2 8 ? ? ? ?
? 7 ? ? ? 5 ? 3 ?
5 3 8 ? ? 7 ? ? ?
? 9 ? ? ? ? ? ? 6
```

Indicar todos los resultados del sudoku de ejemplo

Take the wordcount command, and change it to sudoku:

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar wordcount /user/bigdata/carpetasWordCount/wc-in/ /user/bigdata/carpetasWordCount/wc-out
```

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar sudoku /home/bigdata/hadoop/sudoku.txt
```

```
bigdata@bigdata:~/hadoop$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar sudoku /home/bigdata/hadoop/sudoku.txt
Solving /home/bigdata/hadoop/sudoku.txt
1 4 6 3 7 9 8 2 5
9 5 7 8 1 2 4 6 3
8 2 3 5 6 4 9 7 1
7 1 5 4 9 6 3 8 2
3 8 2 7 5 1 6 4 9
4 6 9 2 8 3 5 1 7
6 7 1 9 4 5 2 3 8
5 3 8 6 2 7 1 9 4
2 9 4 1 3 8 7 5 6
```

C. PI

5. Ejecutar el programa "pi" con los siguientes parámetros de entrada e indicar el valor devuelto.

- Programa: pi
- N° de mapeos: 2

- Nº de muestras por mapeo: 4

\$ sudo apt install pi

```
bigdata@bigdata:~/hadoop$ sudo apt install pi
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libc6
Se instalarán los siguientes paquetes NUEVOS:
  libc6 pi
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 142 no actualizados.
Se necesita descargar 467 kB de archivos.
Se utilizarán 1.533 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S
Des:1 http://es.archive.ubuntu.com/ubuntu artful/universe amd64 libc6 amd64 1.3.4-2 [460 kB]
Des:2 http://es.archive.ubuntu.com/ubuntu artful/universe amd64 pi amd64 1.3.4-2 [6.068 B]
Descargados 467 kB en 0s (850 kB/s)
Seleccionando el paquete libc6 previamente no seleccionado.
(Leyendo la base de datos ... 172152 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../libc6_1.3.4-2_amd64.deb ...
Desempaquetando libc6 (1.3.4-2) ...
Seleccionando el paquete pi previamente no seleccionado.
Preparando para desempaquetar .../archives/pi_1.3.4-2_amd64.deb ...
Desempaquetando pi (1.3.4-2) ...
Configurando libc6 (1.3.4-2) ...
Configurando pi (1.3.4-2) ...
Procesando disparadores para libc-bin (2.26-0ubuntu2.1) ...
Procesando disparadores para man-db (2.7.6.1-2) ...
```

\$ pi --help

```
bigdata@bigdata:~/hadoop$ pi --help
Usage: pi [digits]
Compute decimal Archimedes' constant Pi to arbitrary accuracy.
```

Take the wordcount command, and change it to pi:

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar wordcount /user/bigdata/carpetaWordCount/wc-in/ /user/bigdata/carpetaWordCount/wc-out
```

\$ **hadoop jar \$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar pi 2 4**

```
bigdata@bigdata:~/hadoop$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar pi 2 4
Number of Maps = 2
Samples per Map = 4
Wrote input for Map #0
Wrote input for Map #1
Starting Job
18/02/21 17:50:45 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
18/02/21 17:50:46 INFO input.FileInputFormat: Total input files to process : 2
18/02/21 17:50:46 INFO mapreduce.JobSubmitter: number of splits:2
18/02/21 17:50:47 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated.
18/02/21 17:50:47 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1519214837785_0004
18/02/21 17:50:47 INFO impl.VarnClientImpl: Submitted application application_1519214837785_0004
18/02/21 17:50:48 INFO mapreduce.Job: The url to track the job: http://bigdata:8088/proxy/application_1519214837785_0004
18/02/21 17:50:48 INFO mapreduce.Job: Running Job: job_1519214837785_0004
18/02/21 17:50:56 INFO mapreduce.Job: Job job_1519214837785_0004 running in uber mode : false
18/02/21 17:50:56 INFO mapreduce.Job: map 0% reduce 0%
18/02/21 17:51:06 INFO mapreduce.Job: map 100% reduce 0%
18/02/21 17:51:12 INFO mapreduce.Job: map 100% reduce 100%
18/02/21 17:51:12 INFO mapreduce.Job: Job job_1519214837785_0004 completed successfully
18/02/21 17:51:12 INFO mapreduce.Job: Counters: 49
File System Counters
  FILE: Number of bytes read=50
  FILE: Number of bytes written=606414
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=534
  HDFS: Number of bytes written=215
  HDFS: Number of read operations=11
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=3
Job Counters
  Launched map tasks=2
  Launched reduce tasks=1
  Data-local map tasks=2
  Total time spent by all maps in occupied slots (ms)=15445
  Total time spent by all reduces in occupied slots (ms)=3759
  Total time spent by all map tasks (ms)=15445
  Total time spent by all reduce tasks (ms)=3759
  Total vcore-milliseonds taken by all map tasks=15445
  Total vcore-milliseonds taken by all reduce tasks=3759
  Total megabyte-milliseonds taken by all map tasks=15815680
  Total megabyte-milliseonds taken by all reduce tasks=3849216
Map-Reduce Framework
  Map input records=2
  Map output records=4
  Map output bytes=36
  Map output materialized bytes=56
  Input split bytes=298
  Combine input records=0
  Combine output records=0
  Reduce input groups=2
  Reduce shuffle bytes=56
  Reduce input records=4
  Reduce output records=0
  Spilled Records=8
  Shuffled Maps =2
  Failed Shuffles=0
  Merged Map outputs=2
  GC time elapsed (ms)=370
  CPU time spent (ms)=1400
  Physical memory (bytes) snapshot=663872768
  Virtual memory (bytes) snapshot=6832381952
  Total committed heap usage (bytes)=408158208
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=236
File Output Format Counters
  Bytes Written=97
Job Finished in 27.925 seconds
Estimated value of PI is 3.500000000000000000000000
```

C. PI

6. Ejecutar el programa "pi" con los siguientes parámetros de entrada e indicar el valor devuelto.

- Programa: pi
- N° de mapeos: 5
- N° de muestras por mapeo: 10

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar pi 5 10
```

```
bigdata@bigdata:~/hadoop$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar pi 5 10
```

```
Number of Maps = 5  
Samples per Map = 10  
Wrote input for Map #0  
Wrote input for Map #1  
Wrote input for Map #2  
Wrote input for Map #3  
Wrote input for Map #4
```

```
Map-Reduce Framework  
  Map input records=5  
  Map output records=10  
  Map output bytes=90  
  Map output materialized bytes=140  
  Input split bytes=745  
  Combine input records=0  
  Combine output records=0  
  Reduce input groups=2  
  Reduce shuffle bytes=140  
  Reduce input records=10  
  Reduce output records=0  
  Spilled Records=20  
  Shuffled Maps =5  
  Failed Shuffles=0  
  Merged Map outputs=5  
  GC time elapsed (ms)=1996  
  CPU time spent (ms)=2590  
  Physical memory (bytes) snapshot=1328648192  
  Virtual memory (bytes) snapshot=13658439680  
  Total committed heap usage (bytes)=901644288  
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=590  
File Output Format Counters  
  Bytes Written=97  
Job Finished in 38.752 seconds  
Estimated value of PI is 3.28000000000000000000000000000000
```

EJERCICIOS DE MAP REDUCE

1. Escribe las traducciones de la palabra **PIG**.

I followed the steps from your MapReduce example (WordCount) as well as the 8 steps for this exercise. ("Ejercicio diccionario 1/3")

Crear nuestro fichero Dictionary.java

Partiendo de los siguientes pantallazos o del documento de ejemplo, crear un fichero llamado Dictionary.java

Ensure that all the services are running properly

start-dfs.sh

start-yarn.sh

jps

```
bigdata@bigdata:~/hadoop$ jps
22497 SecondaryNameNode
18084 ResourceManager
2309 Jps
18230 NodeManager
22102 NameNode
22281 DataNode
16811 JobHistoryServer
```

Create a Dictionary.java file

cd /home/bigdata/ejemplos/Hadoop/MapReduce

mkdir ejemploDictionary

cd ejemploDictionary

sudo nano Dictionary.java

```
bigdata@bigdata:~/hadoop$ cd /home/bigdata/ejemplos/Hadoop/MapReduce
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce$ mkdir ejemploDictionary
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce$ cd ejemploDictionary
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$ sudo nano Dictionary.java
[sudo] password for bigdata:
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$
```

Compilar el fichero

Al compilarlo, os aparecerá un mensaje de un método deprecated. Revisar el código y rectificarlo para compilarlo correctamente.

Copy and paste the text from the Dictionary.java.docx file

```

GNU nano 2.8.6                               Archivo: Dictionary.java

import java.io.IOException;
import java.util.StringTokenizer;

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.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.KeyValueTextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Dictionary {

    public static class WordMapper
        extends Mapper<Text, Text, Text, Text>{

        private Text word = new Text();

        public void map(Text key, Text value, Context context
            ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString(), ",");
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(key,word);
            }
        }
    }
}

```

Crear el .jar

Inside the ejemploDictionary file, create a dictionary.jar file with the classes

```

javac -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-2.9.0.jar:$HADOOP_HOME/share/hadoop/common/lib/hadoop-annotations-2.9.0.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.9.0.jar Dictionary.java
jar cf /home/bigdata/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary.jar Dictionary.class

```

```

bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$ javac -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-2.9.0.jar:$HADOOP_HOME/share/hadoop/
/lib/hadoop-annotations-2.9.0.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.9.0.jar Dictionary.java
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$ jar cf /home/bigdata/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary.jar Dictionary.class

```

Check that we do indeed have dictionary.jar and its components inside the directory.

ls /home/bigdata/ejemplos/Hadoop/MapReduce/ejemploDictionary

```

bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$ ls /home/bigdata/ejemplos/Hadoop/MapReduce/ejemploDictionary
Dictionary$AllTranslationsReducer.class Dictionary.class dictionary.jar Dictionary.java Dictionary$WordMapper.class

```

Read the permissions for dictionary.jar

ls -lrt /home/bigdata/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary.jar

```

bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$ ls -lrt /home/bigdata/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary.jar
-rw-r--r-- 1 bigdata bigdata 1276 feb 22 00:12 /home/bigdata/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary.jar

```


Descargar el fichero

Lo primero que haremos es descargarnos los diferentes diccionarios a una nueva carpeta (Emplear por ejemplo comando wget)

wget <http://www.ilovelanguages.com/IDP/files/Spanish.txt>

wget <http://www.ilovelanguages.com/IDP/files/Italian.txt>

wget <http://www.ilovelanguages.com/IDP/files/French.txt>

wget <http://www.ilovelanguages.com/IDP/files/German.txt>

Download all 4 of the dictionaries by pasting the code links:

```
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$ wget http://www.ilovelanguages.com/IDP/files/Spanish.txt
Guardando como: "Spanish.txt"

Spanish.txt                               100%[=====]

2018-02-22 00:56:30 (294 KB/s) - "Spanish.txt" guardado [171564/171564]
```

Create a new directory, dictionary-in-local, where the text file will go

mkdir /home/bigdata/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary-in-local/

```
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$ mkdir /home/bigdata/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary-in-local/
```

Check that we have the HDFS folder called dictionary-in-local

```
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$ ls
Dictionary$AllTranslationsReducer.class  dictionary-in-local  Dictionary.java      French.txt  Italian.txt
Dictionary.class                         dictionary.jar        Dictionary$WordMapper.class  German.txt  Spanish.txt
```

Merge de ficheros

Como Hadoop trabaja mejor con ficheros grandes, haremos un merge de los 4 ficheros para obtener un único fichero diccionario.

Por ejemplo se puede emplear el comando CAT redirigiendo la salida a un nuevo fichero llamado dictionary.txt

Put the 4 text files inside of 1 text file called dictionary.txt. (Then put that inside the directory dictionary-in-local)

cat Spanish.txt Italian.txt French.txt German.txt > dictionary.txt

```
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$ cat Spanish.txt Italian.txt French.txt German.txt > dictionary.txt
```

```
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$ mv dictionary.txt dictionary-in-local
```

Check that the file contains the 4 sets of keys and values

```
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$ cd dictionary-in-local
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary-in-local$ ls
dictionary.txt
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary-in-local$ nano dictionary.txt
```

Here you can see the end of the English-Spanish dictionary, and the beginning of the English-Italian

```

zoologist      el zoo/logo[Noun]
zoology la zoologi/a[Noun]
zoom      pasar zumbando[Verb]
zounds    ca/spita
zucchini   el calabaci/n[Noun]
#####
#Copyright 1999 The Internet Dictionary Project/Tyler Chambers
#http://www.june29.com/IDP/
#This file is free to use and modify. Thank you for using the IDP.
#
#Approximately 2990 entries. 9/21/97
#Approximately 3386 entries. 1/7/98
#Approximately 3790 entries. 3/8/98
#Approximately 5150 entries. 2/19/99
#####
a      un, uno, una[Article]
a      uno, una, un
a      un,uno,una,un'[Article]
aardvark      oritteropo
aardvarks     oritteropi
ab      da[Preposition]
aback      di dietro
aback      essere sorpreso/essere preso alla sprovvista
abacterial    abatterico
abacus        abaco
abacuses      abachi[Noun]
abacuses      abaci

```

Get rid of the previous language texts

rm Spanish.txt Italian.txt German.txt French.txt

```

bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$ rm Spanish.txt Italian.txt German.txt French.txt
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$ ls
Dictionary$AllTranslationsReducer.class Dictionary.class dictionary.jar Dictionary$WordMapper.class -in-local
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary$

```

Subir el fichero a HDFS

Empleando alguno de los comandos de subida a HDFS [mover el fichero a una nueva carpeta](#) en la ruta /user/bigdata/mapreduce poniéndole el nombre diccionario.txt.

We need dictionary-in-local inside HDFS, so create a folder in user/bigdata. Load the files into this HDFS folder.

hdfs dfs -put /home/bigdata/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary-in-local/* /user/bigdata/dictionary-with-java

```

bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary-in-local$ hdfs dfs -put /home/bigdata/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary-in-local/
* /user/bigdata/dictionary-with-java

```

Confirm that it loaded into /user/bigdata correctly

hdfs dfs -ls /user/bigdata/dictionary-with-java

```

bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary-in-local$ hdfs dfs -ls /user/bigdata/dictionary-with-java
-rw-r--r-- 1 bigdata supergroup 598677 2018-02-22 12:10 /user/bigdata/dictionary-with-java

```

Create an output folder, dictionary-java-output, where all the jar-executed results will be outputted to:

hadoop jar /home/bigdata/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary.jar Dictionary dictionary-with-java dictionary-java-output

```
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary-in-local$ hadoop jar /home/bigdata/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary.jar Dictionary
ry dictionary-with-java dictionary-java-output
18/02/22 12:18:47 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
18/02/22 12:18:48 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner
to remedy this.
18/02/22 12:18:48 INFO input.FileInputFormat: Total input files to process : 1
18/02/22 12:18:48 INFO mapreduce.JobSubmitter: number of splits:1
18/02/22 12:18:48 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
18/02/22 12:18:49 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1519214837785_0008
18/02/22 12:18:49 INFO impl.YarnClientImpl: Submitted application application_1519214837785_0008
18/02/22 12:18:49 INFO mapreduce.Job: The url to track the job: http://bigdata:8088/proxy/application_1519214837785_0008/
18/02/22 12:18:49 INFO mapreduce.Job: Running job: job_1519214837785_0008
18/02/22 12:19:00 INFO mapreduce.Job: Job job_1519214837785_0008 running in uber mode : false
18/02/22 12:19:00 INFO mapreduce.Job:  map 0% reduce 0%
18/02/22 12:19:08 INFO mapreduce.Job:  map 100% reduce 0%
18/02/22 12:19:15 INFO mapreduce.Job:  map 100% reduce 100%
18/02/22 12:19:17 INFO mapreduce.Job: Job job_1519214837785_0008 completed successfully
18/02/22 12:19:17 INFO mapreduce.Job: Counters: 49
```

Ejecutar el proceso map reduce.

En este ejemplo, como hemos visto no hay parámetros de entrada ya que vienen indicados en el código.

Confirm that the files are loaded.

hdfs dfs -ls /user/bigdata/*

```
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary-in-local$ hdfs dfs -ls /user/bigdata/*
Found 2 items
drwxr-xr-x - bigdata supergroup 0 2018-02-21 16:06 /user/bigdata/carpetaWordCount/wc-in
drwxr-xr-x - bigdata supergroup 0 2018-02-21 16:11 /user/bigdata/carpetaWordCount/wc-out
Found 1 items
-rw-r--r-- 1 bigdata supergroup 0 2018-02-22 01:19 /user/bigdata/dictionary-in-java/dictionary.txt
Found 2 items
-rw-r--r-- 1 bigdata supergroup 0 2018-02-22 12:19 /user/bigdata/dictionary-java-output/_SUCCESS
-rw-r--r-- 1 bigdata supergroup 525276 2018-02-22 12:19 /user/bigdata/dictionary-java-output/part-r-00000
-rw-r--r-- 1 bigdata supergroup 598677 2018-02-22 12:19 /user/bigdata/dictionary-with-java
```

hdfs dfs -cat /user/bigdata/dictionary-java-output/part-r-00000 > mapReducedDictionary.txt

```
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary-in-local$ hdfs dfs -cat /user/bigdata/dictionary-java-output/part-r-00000 > mapReducedDictionary.txt
```

Devolver traducción de pig

grep pig mapReducedDictionary.txt

```
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary-in-local$ grep pig mapReducedDictionary.txt
pig | cochon[Noun]|Schwein (n)|el chancho[Noun]|el puerco
```

Resultado:

```
bigdata@bigdata:~/ejemplos/Hadoop/MapReduce/ejemploDictionary/dictionary-in-local$ grep pig mapReducedDictionary.txt
pig | cochon[Noun]|Schwein (n)|el chancho[Noun]|el puerco
```