Introducción a RHadoop: MapReduce

Conociendo a Dana | 08.02.17

Who's **dana**?

Características principales

- 4 nodos
- Cada nodo consta de 1 procesador i5, 8Gb de RAM, 500 Gb de disco duro y GNU/Linux (Debian 8)
- Estructura:
  - 1 nodo permite el acceso al sistema (**dana**)
  - 3 nodos están conectados mediante una red interna

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
351	0	0	351	0	0 B	32 GB	0 B	0	16	0	4	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:2048, vCores:1>	<memory:8192, vCores:4>

Show 20 entries Search:

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
	/default-rack	RUNNING	dana:41073	dana:8042	jue feb 02 15:30:17 +0100 2017		0	0 B	8 GB	0	4	2.7.2
	/default-rack	RUNNING	clust-01:32821	clust-01:8042	jue feb 02 15:30:28 +0100 2017		0	0 B	8 GB	0	4	2.7.2
	/default-rack	RUNNING	clust-03:38879	clust-03:8042	jue feb 02 15:32:24 +0100 2017		0	0 B	8 GB	0	4	2.7.2
	/default-rack	RUNNING	clust-02:46174	clust-02:8042	jue feb 02 15:32:30 +0100 2017		0	0 B	8 GB	0	4	2.7.2

Figure 1:

Nociones básicas (I)

Conexión con el servidor

local:~$ ssh [usuario]@dana.estad.ucm.es

```
jesandubete:~ julioemilio$ ssh julioemilio@dana.estad.ucm.es
Enter passphrase for key '/Users/julioemilio/.ssh/id_rsa':

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

Figure 2:

Nociones básicas (II)

Copiar archivos al servidor

local:~$ scp < *fich.orig* >[usuario]@dana.estad.ucm.es:< *fich.dest* >

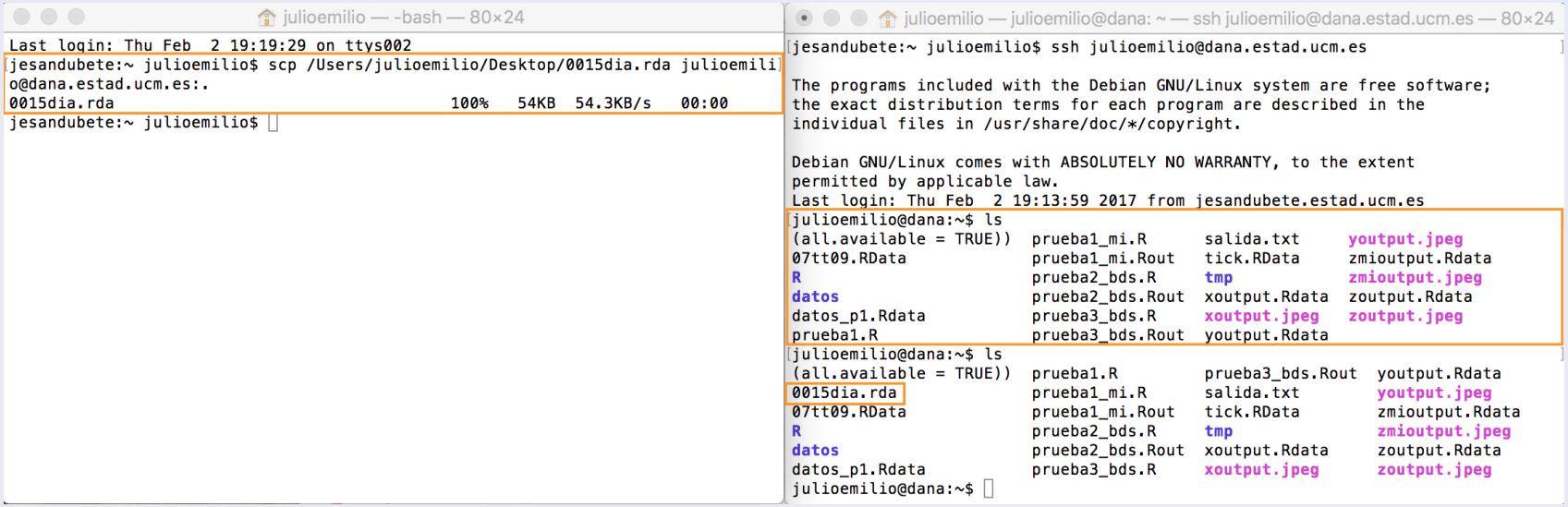

Figure 3: Lista de comandos POSIX

Nociones básicas (III)

Convertir un archivo en hdfs

dana:~$ hadoop fs -< *comando* > < *argumentos* >

```
julioemilio@dana:~$ hadoop fs -put 0015dia.rda
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/opt/hadoop/hadoop-2.7.2/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLogger
Binder.class]
SLF4J: Found binding in [jar:file:/home/opt/tez/tez-0.8.2/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
julioemilio@dana:~$ hadoop fs -ls /user/julioemilio/
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/opt/hadoop/hadoop-2.7.2/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLogger
Binder.class]
SLF4J: Found binding in [jar:file:/home/opt/tez/tez-0.8.2/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 4 items
-rw-r--r--   2 julioemilio supergroup      55554 2017-02-03 12:20 /user/julioemilio/0015dia.rda
```

Figure 4: Lista de comandos HDFS

Nociones básicas (IV)

Comandos elementales HDFS

- $ hadoop fs -put $<$ *fich.orig* $>$ $<$ *fich.HDFS* $>$
- $ hadoop fs -get $<$ *fich.HDFS* $>$ $<$ *fich.orig* $>$
- $ hadoop fs -ls
- $ hadoop fs -cat $<$ *fich.HDFS* $>$
- $ hadoop fs -mkdir $<$ *name.dir* $>$
- $ hadoop fs -rm -r $<$ *name.dir* $>$

Nociones básicas (V)

Copiar archivos desde el servidor

local:~$ scp [usuario]@dana.estad.ucm.es:< *fich.orig* >< *fich.dest* >

```
jesandubete:~ julioemilio$ cd /Users/julioemilio/Desktop/R
jesandubete:R julioemilio$ ls
jesandubete:R julioemilio$ scp julioemilio@dana.estad.ucm.es:0015dia.rda /Users/
julioemilio/Desktop/R
0015dia.rda                                   100%   54KB  54.3KB/s   00:00
jesandubete:R julioemilio$ ls
0015dia.rda
```

```
julioemilio@dana:~$ ls
(all.available = TRUE))  prueba1.R          prueba3_bds.Rout  youtput.Rdata
0015dia.rda              prueba1_mi.R       salida.txt        youtput.jpeg
07tt09.RData             prueba1_mi.Rout    tick.RData        zmioutput.Rdata
R                        prueba2_bds.R      tmp               zmioutput.jpeg
datos                    prueba2_bds.Rout   xoutput.Rdata     zoutput.Rdata
datos_p1.Rdata           prueba3_bds.R      xoutput.jpeg      zoutput.jpeg
```

Figure 5:

Control de ejecución en Hadoop

- OpenVPN + fichero de configuración
- Visionado en streaming: `http://192.168.8.1:8088/`
- Visionado historial: `http://192.168.8.1:19888/`

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
1484045896267_0352	julioemilio	streamjob9185242589144611138.jar	MAPREDUCE	default	Thu Feb 2 12:00:58 +0100 2017	Thu Feb 2 12:01:10 +0100 2017	FINISHED	SUCCEEDED		History
1484045896267_0351	julioemilio	streamjob7465981918549020147.jar	MAPREDUCE	default	Thu Feb 2 11:57:47 +0100 2017	Thu Feb 2 11:58:04 +0100 2017	FINISHED	SUCCEEDED		History
1484045896267_0350	julioemilio	streamjob4269315852951050900.jar	MAPREDUCE	default	Thu Feb 2 11:51:03 +0100 2017	Thu Feb 2 11:53:28 +0100 2017	FINISHED	SUCCEEDED		History
1484045896267_0349	julioemilio	streamjob2065089131964527548.jar	MAPREDUCE	default	Thu Feb 2 11:42:25 +0100 2017	Thu Feb 2 11:44:35 +0100 2017	FINISHED	SUCCEEDED		History

Figure 6:

La función MapReduce vía RHadoop

- RHadoop es una colección de 5 paquetes de R que posibilita al usuario gestionar y analizar un conjunto de datos en Hadoop: `rhdfs`, `rhbase`, `plyrmr`, `rmr2` & `ravro`.
- `rmr2()` permite utilizar la función MapReduce en un entorno Hadoop.
- La función `MapReduce` se caracteriza por:
  - Tiene 2 argumentos: claves y valores.
  - La salida es un par (c,v) aplicando la regla `keyval` predefinida.
  - Los argumentos pueden ser vectores, listas, matrices o data frames.

Ejemplo 1: Aplicar una operación a una secuencia (I)

Dada una secuencia del 1 al 100 se quiere elevar cada número al cuadrado

En *R* el código es,

```
small.ints <- 1:100
salida     <- sapply(small.ints,function(x) x^2)
head(salida, n= 10)
```

```
##   [1]   1   4   9  16  25  36  49  64  81 100
```

En este caso el esquema de **clave-valor** es:

- clave -> *no hay*
- valor -> frecuencia

Ejemplo 1: Aplicar una operación a una secuencia (II)

En *Hadoop* es,

`library(rmr2)` -> Carga la librería

`to.dfs()` -> Envía los datos al sistema HDFS

`from.dfs()` -> Recupera los datos del sistema HDFS

```
library(rmr2)
small.ints <- to.dfs(1:1000)
salida1    <- from.dfs(
                mapreduce(
                   input = small.ints,
                   map = function(k,v) cbind(v,v^2)
                )
)
write(salida1,"salida1.txt")
```

Ejemplo 1: Aplicar una operación a una secuencia (III)

Input

```
jesandubete:~ julioemilio$ scp /Users/julioemilio/Desktop/Ejemplo1.R julioemilio
@dana.estad.ucm.es:.
Ejemplo1.R                                    100%  157     0.2KB/s   00:00
jesandubete:~ julioemilio$
```

```
jesandubete:~ julioemilio$ ssh julioemilio@dana.estad.ucm.es

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Feb  3 10:59:33 2017 from jesandubete.estad.ucm.es
julioemilio@dana:~$ Rscript Ejemplo1.R
```

Figure 7:

Output

```
jesandubete:~ julioemilio$ scp /Users/julioemilio/Desktop/Ejemplo1.R julioemilio
@dana.estad.ucm.es:.
Ejemplo1.R                                    100%  157     0.2KB/s   00:00
jesandubete:~ julioemilio$ scp julioemilio@dana.estad.ucm.es:salida1.txt /Users/
julioemilio/Desktop
salida1.txt                                   100%    0     0.0KB/s   00:00
```

```
julioemilio@dana:~$ ls
(all.available = TRUE))  ejemplo1          prueba3_bds.Rout  youtput.Rdata
0015dia.rda              prueba1.R         salida.txt        youtput.jpeg
07tt09.RData             prueba1_mi.R      salida1.txt       zmioutput.Rdata
Ejemplo1.R               prueba1_mi.Rout   tick.RData        zmioutput.jpeg
R                        prueba2_bds.R     tmp               zoutput.Rdata
```

Figure 8:

Ejemplo 2: Contar números (I)

Generar 32 números provenientes de una $Bin(50, 0.4)$ *y contar las veces que sale cada número*

En *R* el código es,

```
groups <- rbinom(n= 32, size= 50, prob= 0.4)
h      <- tapply(X= groups, INDEX= groups, FUN= length)
h
```

```
## 14 15 17 18 19 20 21 22 23 25 26 29
##  1  1  4  6  2  2  4  3  2  3  3  1
```

En este caso el esquema de **clave-valor** es:

- clave -> resultados posibles
- valor -> frecuencia

Ejemplo 2: Contar números (II)

En *Hadoop* es,

```
library(rmr2)
groups  <- to.dfs(groups)
salida2 <- from.dfs(
                  mapreduce(
                    input = groups,
                    map = function(., v) keyval(v, 1),
                    reduce = function(k, vv)
                             keyval(k, length(vv))
                  )
)
write(salida2,"salida2.txt")
```

Ejemplo 2: Contar números (III)

Input

```
jesandubete:~ julioemilio$ scp /Users/julioemilio/Desktop/Ejemplo2.R julioemilio
@dana.estad.ucm.es:.
Ejemplo2.R                                    100%  260     0.3KB/s   00:00
jesandubete:~ julioemilio$
```

```
jesandubete:~ julioemilio$ ssh julioemilio@dana.estad.ucm.es

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Feb  3 11:41:11 2017 from jesandubete.estad.ucm.es
julioemilio@dana:~$ Rscript Ejemplo2.R
```

Figure 9:

Output

```
jesandubete:~ julioemilio$ scp julioemilio@dana.estad.ucm.es:salida2.txt /Users/
julioemilio/Desktop
salida2.txt                                   100%    0     0.0KB/s   00:00
jesandubete:~ julioemilio$
```

```
julioemilio@dana:~$ ls
(all.available = TRUE))  ejemplo1          salida.txt     youtput.jpeg
0015dia.rda              prueba1.R         salida1.txt    zmioutput.Rdata
07tt09.RData             prueba1_mi.R      salida2.txt    zmioutput.jpeg
Ejemplo1.R               prueba1_mi.Rout   tick.RData     zoutput.Rdata
Ejemplo2.R               prueba2_bds.R     tmp            zoutput.jpeg
```

Figure 10:

Ejemplo 2: Contar números (IV)

Monitorización de resultados

Counter Group	Counters			
File System Counters	Name	Map	Reduce	Total
	FILE: Number of bytes read	0	1.964	1.964
	FILE: Number of bytes written	248.454	125.151	373.605
	FILE: Number of large read operations	0	0	0
	FILE: Number of read operations	0	0	0
	FILE: Number of write operations	0	0	0
	HDFS: Number of bytes read	1.031	0	1.031
	HDFS: Number of bytes written	0	1.970	1.970
	HDFS: Number of large read operations	0	0	0
	HDFS: Number of read operations	10	3	13
	HDFS: Number of write operations	0	2	2
Job Counters	Name	Map	Reduce	Total
	Data-local map tasks	0	0	2
	Launched map tasks	0	0	2
	Launched reduce tasks	0	0	1
	Total megabyte-milliseconds taken by all map tasks	0	0	7.795.712
	Total megabyte-milliseconds taken by all reduce tasks	0	0	3.233.792
	Total time spent by all map tasks (ms)	0	0	7.613
	Total time spent by all maps in occupied slots (ms)	0	0	7.613
	Total time spent by all reduce tasks (ms)	0	0	3.158
	Total time spent by all reduces in occupied slots (ms)	0	0	3.158
	Total vcore-milliseconds taken by all map tasks	0	0	7.613
	Total vcore-milliseconds taken by all reduce tasks	0	0	3.158
Map-Reduce Framework	Name	Map	Reduce	Total
	Combine input records	0	0	0
	Combine output records	0	0	0
	CPU time spent (ms)	1.100	840	1.940
	Failed Shuffles	0	0	0
	GC time elapsed (ms)	49	15	64
	Input split bytes	178	0	178
	Map input records	3	0	3
	Map output bytes	1.916	0	1.916
	Map output materialized bytes	1.970	0	1.970
	Map output records	21	0	21
	Merged Map outputs	0	2	2
	Physical memory (bytes) snapshot	577.736.704	195.936.256	773.672.960
	Reduce input groups	0	10	10
	Reduce input records	0	21	21
	Reduce output records	0	22	22
	Reduce shuffle bytes	0	1.970	1.970
	Shuffled Maps	0	2	2
	Spilled Records	21	21	42
	Total committed heap usage (bytes)	589.824.000	157.810.688	747.634.688
	Virtual memory (bytes) snapshot	2.115.481.600	1.064.914.944	3.180.396.544
Shuffle Errors	Name	Map	Reduce	Total
	BAD_ID	0	0	0
	CONNECTION	0	0	0
	IO_ERROR	0	0	0
	WRONG_LENGTH	0	0	0
	WRONG_MAP	0	0	0
	WRONG_REDUCE	0	0	0
File Input Format Counters	Name	Map	Reduce	Total
	Bytes Read	853	0	853
File Output Format Counters	Name	Map	Reduce	Total
	Bytes Written	0	1.970	1.970
rmr	Name	Map	Reduce	Total
	reduce calls	0	10	10

Figure 11: `http://192.168.8.1:19888`

Las funciones `to.dfs()` y `from.dfs()`

La conexión entre la memoría y los datos *HDFS* se realiza a través de las funciones:

- `to.dfs()`
- `from.dfs()`

Problema: si el volumen de datos es suficientemente grande puede colapsar el sistema.

Ejemplo 3: Contar palabras (I)

Dado un fichero de texto calcular la frecuencia de las palabras que aparecen

-**Paso 1**: Definir la variable de entorno

-**Paso 2**: Convertir el conjunto de datos a *HDFS*

-**Paso 3**: Ejecutar el script de R en *HADOOP*

-**Paso 4**: Recuperar el fichero de resultados

-**Paso 5**: Monitorizar el rendimiento del proceso

Ejemplo 3: Contar palabras (II)

```
Sys.setenv("HADOOP_HOME"="/srv/nfs4/opt/hadoop/
           hadoop-2.7.2")
library(rmr2)

# Función map

map <- function(k,lines) {
                words.list <- strsplit(lines, '\\s')
                words <- unlist(words.list)
                return(keyval(words, 1))}

# Función reduce

reduce <- function(word, counts) {
                  keyval(word, sum(counts))}
```

Ejemplo 3: Contar palabras (III)

```
# Función wordcount
wordcount <- function (input, output=NULL) {
                       mapreduce(input=input,
                                 output=output,
                                 input.format="text",
                                 map=map,
                                 reduce=reduce)}

# Definir ruta
hdfs.root <- '/user/julioemilio'
hdfs.data <- file.path(hdfs.root, 'quijote.txt')
hdfs.out <- file.path(hdfs.root, 'out')
out <- wordcount(hdfs.data, hdfs.out)
```

Ejemplo 3: Contar palabras (IV)

```
# Obtener resultados
results <- from.dfs(out)
results.df <- data.frame(word=results[[1]],
                         count=results[[2]])
save(results.df,file = "salida3.RData")
```

Ejemplo 3: Contar palabras (V)

Input

```
eduroam228153:~ julioemilio$ scp /Users/julioemilio/Desktop/ejemplo3.R julioemil
io@dana.estad.ucm.es:.
ejemplo3.R                                    100% 1033     1.0KB/s   00:00
eduroam228153:~ julioemilio$ scp /Users/julioemilio/Desktop/quijote.txt julioemi
lio@dana.estad.ucm.es:.
quijote.txt                                   100% 8360     8.2KB/s   00:00
eduroam228153:~ julioemilio$
```

```
ejemplo3.R              prueba3_bds.Rout  youtput.Rdata
ejemplo4.R              quijote.txt       youtput.jpeg
julioemilio@dana:~$ hadoop fs -put quijote.txt
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/opt/hadoop/hadoop-2.7.2/share/hadoop/com
mon/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/opt/tez/tez-0.8.2/lib/slf4j-log4j12-1.7.
10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
julioemilio@dana:~$ hadoop fs -ls
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/opt/hadoop/hadoop-2.7.2/share/hadoop/com
mon/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/opt/tez/tez-0.8.2/lib/slf4j-log4j12-1.7.
10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 4 items
-rw-r--r--   2 julioemilio supergroup   20509529 2017-02-02 19:28 07tt09.RData
drwxr-xr-x   - julioemilio supergroup          0 2016-09-23 14:25 R
drwxr-xr-x   - julioemilio supergroup          0 2017-02-06 15:45 datos
-rw-r--r--   2 julioemilio supergroup       8360 2017-02-08 10:07 quijote.txt
```

Figure 12:

Ejemplo 3: Contar palabras (VI)

Output

```
julioemilio@dana:~$ Rscript ejemplo3.R -r hadoop
Durante la inicializaci?n - Mensajes de aviso perdidos
Setting LC_CTYPE failed, using "C"
Loading required package: methods
Please review your hadoop settings. See help(hadoop.settings)
Mensajes de aviso perdidos
S3 methods 'gorder.default', 'gorder.factor', 'gorder.data.frame', 'gorder.matrix', 'gorder.raw' were declared in NAMESPACE but not found
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/opt/hadoop/hadoop-2.7.2/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/opt/tez/tez-0.8.2/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
packageJobJar: [/tmp/hadoop-unjar863977881797762049/] [] /tmp/streamjob186468569860853025.jar tmpDir=null
17/02/08 10:14:46 INFO client.RMProxy: Connecting to ResourceManager at /192.168.134.1:8032
17/02/08 10:14:46 INFO client.RMProxy: Connecting to ResourceManager at /192.168.134.1:8032
17/02/08 10:14:47 INFO mapred.FileInputFormat: Total input paths to process : 1
17/02/08 10:14:48 INFO mapreduce.JobSubmitter: number of splits:2
17/02/08 10:14:48 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1486543458613_0001
17/02/08 10:14:48 INFO impl.YarnClientImpl: Submitted application application_1486543458613_0001
17/02/08 10:14:48 INFO mapreduce.Job: The url to track the job: http://dana:8088/proxy/application_1486543458613_0001/
17/02/08 10:14:48 INFO mapreduce.Job: Running job: job_1486543458613_0001
17/02/08 10:14:54 INFO mapreduce.Job: Job job_1486543458613_0001 running in uber mode : false
17/02/08 10:14:54 INFO mapreduce.Job:  map 0% reduce 0%
17/02/08 10:15:02 INFO mapreduce.Job:  map 100% reduce 0%
17/02/08 10:15:08 INFO mapreduce.Job:  map 100% reduce 100%
17/02/08 10:15:08 INFO mapreduce.Job: Job job_1486543458613_0001 completed successfully
17/02/08 10:15:08 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=64586
                FILE: Number of bytes written=498432
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=12734
                HDFS: Number of bytes written=47488
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        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)=11358
                Total time spent by all reduces in occupied slots (ms)=2808
                Total time spent by all map tasks (ms)=11358
                Total time spent by all reduce tasks (ms)=2808
                Total vcore-milliseconds taken by all map tasks=11358
                Total vcore-milliseconds taken by all reduce tasks=2808
                Total megabyte-milliseconds taken by all map tasks=11630592
                Total megabyte-milliseconds taken by all reduce tasks=2875392
        Map-Reduce Framework
                Map input records=142
                Map output records=925
                Map output bytes=62712
                Map output materialized bytes=64592
                Input split bytes=194
```

Ejemplo 3: Contar palabras (VII)

Output

```
eduroam228153:~ julioemilio$ scp julioemilio@dana.estad.ucm.es:salida3.RData /Us
ers/julioemilio/Desktop
salida3.RData                                 100% 5405     5.3KB/s   00:00
eduroam228153:~ julioemilio$
```

```
julioemilio@dana:~$ ls
(all.available = TRUE))  prueba1_mi.Rout   salida2.txt     youtput.jpeg
R                        prueba2_bds.R     salida3.RData   zmioutput.Rdata
datos                    prueba2_bds.Rout  tmp             zmioutput.jpeg
ejemplo2.R               prueba3_bds.R     xoutput.Rdata   zoutput.Rdata
ejemplo3.R               prueba3_bds.Rout  xoutput.jpeg    zoutput.jpeg
ejemplo4.R               quijote.txt       youtput.Rdata
```

Figure 14:

Ejemplo 3: Contar palabras (VIII)

Output

```
> load("/Users/julioemilio/Desktop/salida3.RData")
> head(results.df,20)
   word count
1     Y     3
2     a    19
3     e     1
4     o     9
5     y    77
6    AL     1
7    DE     2
8    EL     2
9    El     6
10   En     2
11   La     2
12   Y,     1
```

Ejemplo 3: Contar palabras (IX)

Monitorización de resultados

	Name	Map	Reduce	Total
File System Counters	FILE: Number of bytes read	0	64.586	64.586
	FILE: Number of bytes written	310.798	187.634	498.432
	FILE: Number of large read operations	0	0	0
	FILE: Number of read operations	0	0	0
	FILE: Number of write operations	0	0	0
	HDFS: Number of bytes read	12.734	0	12.734
	HDFS: Number of bytes written	0	47.488	47.488
	HDFS: Number of large read operations	0	0	0
	HDFS: Number of read operations	6	3	9
	HDFS: Number of write operations	0	2	2
	Name	**Map**	**Reduce**	**Total**
Job Counters	Data-local map tasks	0	0	2
	Launched map tasks	0	0	2
	Launched reduce tasks	0	0	1
	Total megabyte-milliseconds taken by all map tasks	0	0	11.630.592
	Total megabyte-milliseconds taken by all reduce tasks	0	0	2.875.392
	Total time spent by all map tasks (ms)	0	0	11.358
	Total time spent by all maps in occupied slots (ms)	0	0	11.358
	Total time spent by all reduce tasks (ms)	0	0	2.808
	Total time spent by all reduces in occupied slots (ms)	0	0	2.808
	Total vcore-milliseconds taken by all map tasks	0	0	11.358
	Total vcore-milliseconds taken by all reduce tasks	0	0	2.808
	Name	**Map**	**Reduce**	**Total**
Map-Reduce Framework	Combine input records	0	0	0
	Combine output records	0	0	0
	CPU time spent (ms)	900	730	1.630
	Failed Shuffles	0	0	0
	GC time elapsed (ms)	18	18	36
	Input split bytes	194	0	194
	Map input records	142	0	142
	Map output bytes	62.712	0	62.712
	Map output materialized bytes	64.592	0	64.592
	Map output records	925	0	925
	Merged Map outputs	0	2	2
	Physical memory (bytes) snapshot	595.664.896	185.204.736	780.869.632
	Reduce input groups	0	647	647
	Reduce input records	0	925	925
	Reduce output records	0	750	750
	Reduce shuffle bytes	0	64.592	64.592
	Shuffled Maps	0	2	2
	Spilled Records	925	925	1.850
	Total committed heap usage (bytes)	526.385.152	157.810.688	684.195.840
	Virtual memory (bytes) snapshot	2.105.495.552	1.069.142.016	3.174.637.568
	Name	**Map**	**Reduce**	**Total**
Shuffle Errors	BAD_ID	0	0	0
	CONNECTION	0	0	0
	IO_ERROR	0	0	0
	WRONG_LENGTH	0	0	0
	WRONG_MAP	0	0	0
	WRONG_REDUCE	0	0	0
	Name	**Map**	**Reduce**	**Total**
File Input Format Counters	Bytes Read	12.540	0	12.540
	Name	**Map**	**Reduce**	**Total**
File Output Format Counters	Bytes Written	0	47.488	47.488
	Name	**Map**	**Reduce**	**Total**
rmr	reduce calls	0	647	647

Figure 16: `http://192.168.8.1:19888`