

APACHE SPARK

SESIÓN 4



UNIÓ EUROPEA

Fons Social Europeu

L'FSE inverteix en el teu futur

¿QUÉ VEREMOS EN ESTA UNIDAD?



Repaso y corrección de ejercicios

Spark Streaming

Spark en Cluster

Spark Streaming



¿QUÉ ES SPARK STREAMING?

Tiempo real

Extensión de spark

Java,Scala,Python

¿APLICACIONES?

Monitorización en tiempo real

Sistemas de seguridad

Detección de fraudes

...

Spark Streaming



Latencia de segundos

Escalable

Tolerante a fallos

Procesamiento de la información una sola vez

Api alto nivel

Spark Streaming

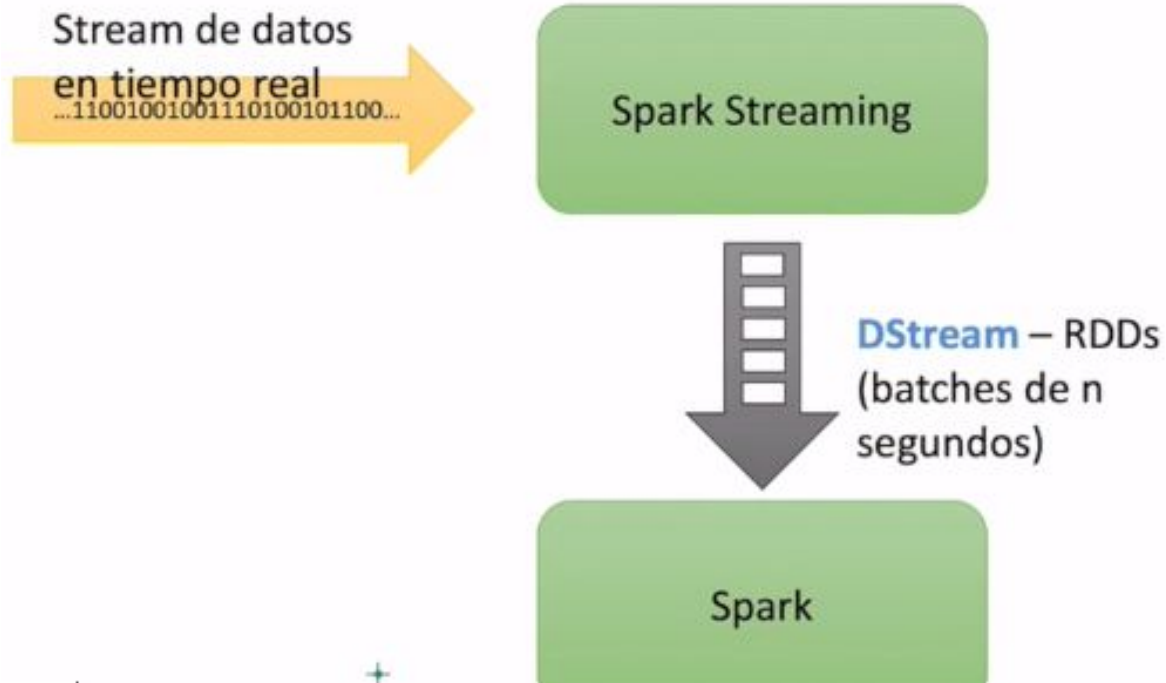
FUNCIONAMIENTO

Divide los datos en bloques de N segundos

Cada bloque se procesa como un RDD



Spark Streaming



Spark Streaming



Trabajando con Spark Streaming

```
StreamingContext (SparkConf conf, Duration batchDuration)
```

```
socketTextStream(String hostname, int port, StorageLevel  
storageLevel)
```


Spark Streaming



```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

ssc = StreamingContext(sc, 2)
lines = ssc.socketTextStream("localhost", 9999)

words = lines.flatMap(lambda line: line.split(" "))

pairs = words.map(lambda word: (word, 1))
wordCounts = pairs.reduceByKey(lambda x, y: x + y)
wordCounts.pprint()

ssc.start() # Start the computation
ssc.awaitTermination() # Wait for the computation to terminate
```


Spark Streaming



```
Time: 1401219545000 ms
```

```
-----  
(23713,2)
```

```
(53,2)
```

```
(24444,2)
```

```
(127,2)
```

```
(93,2)
```

```
...
```

```
-----  
Time: 1401219547000 ms
```

```
-----  
(42400,2)
```

```
(24996,2)
```

```
(97464,2)
```

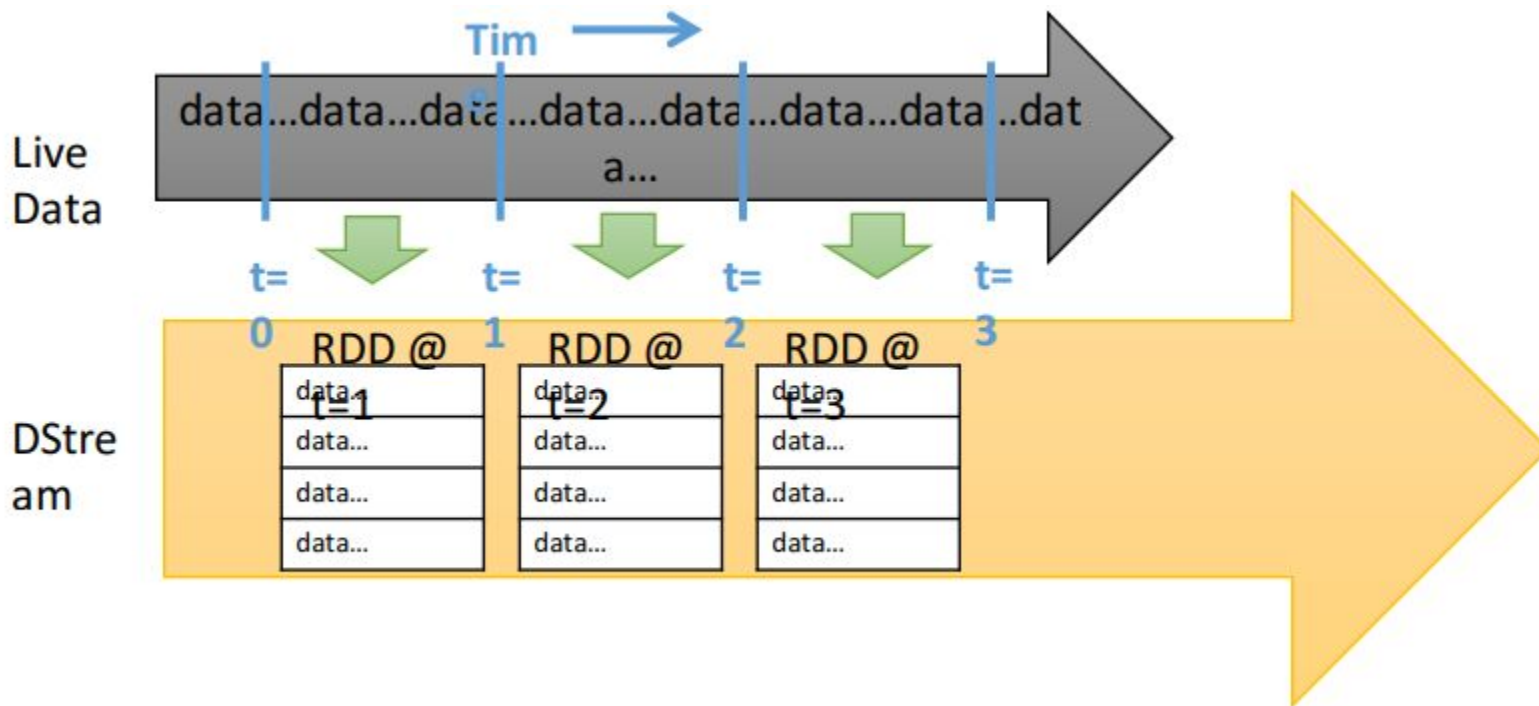
```
(161,2)
```

```
(6011,2)
```

```
..
```

Spark Streaming

DStream



Spark Streaming



Las operaciones que aplicamos sobre DStream se aplican a cada uno de los RDD

2 tipos de Operaciones sobre DStream

- Transformaciones
- Operaciones de salida

Otras funciones

Transform(funcion)

Spark Streaming



OPERACIONES DE SALIDA

Salida a consola

- `Print()`

Salida a fichero

- `saveAsTextFiles`
- `saveAsObjectFiles`

Otras

- `foreachRDD(function,time)`

Spark Streaming



Entrada de datos

- Sockets
- Flume
- Kafka
- RabbitMQ
- Twitter
- Ficheros
- ...

Spark Streaming



StreamingContext: Punto de entrada (equivalente a SparkContext pero pasándole el tiempo de batch)

DStream: Conjunto de datos que recogemos de la fuente en streaming

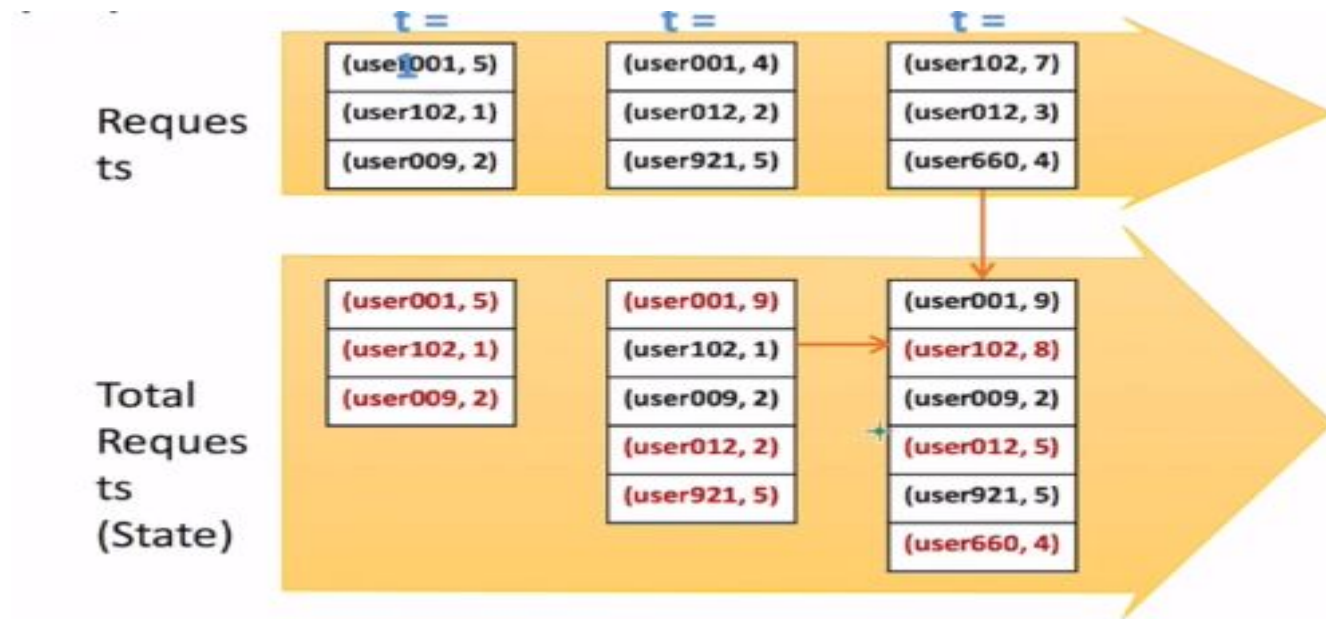
Procesamos los bloques del DStream igual que un RDD

Función start

Función awaitTermination()

Spark Streaming

UpdateStatByKey



Spark Streaming



CHECKPOINT

Es necesario usar checkpoint para poder utilizar UpdateStateByKey

```
ssc.checkpoint("checkpoints")  
totalUserreqs =  
userreqs.updateStateByKey(updateCount)  
totalUserreqs.pprint()
```

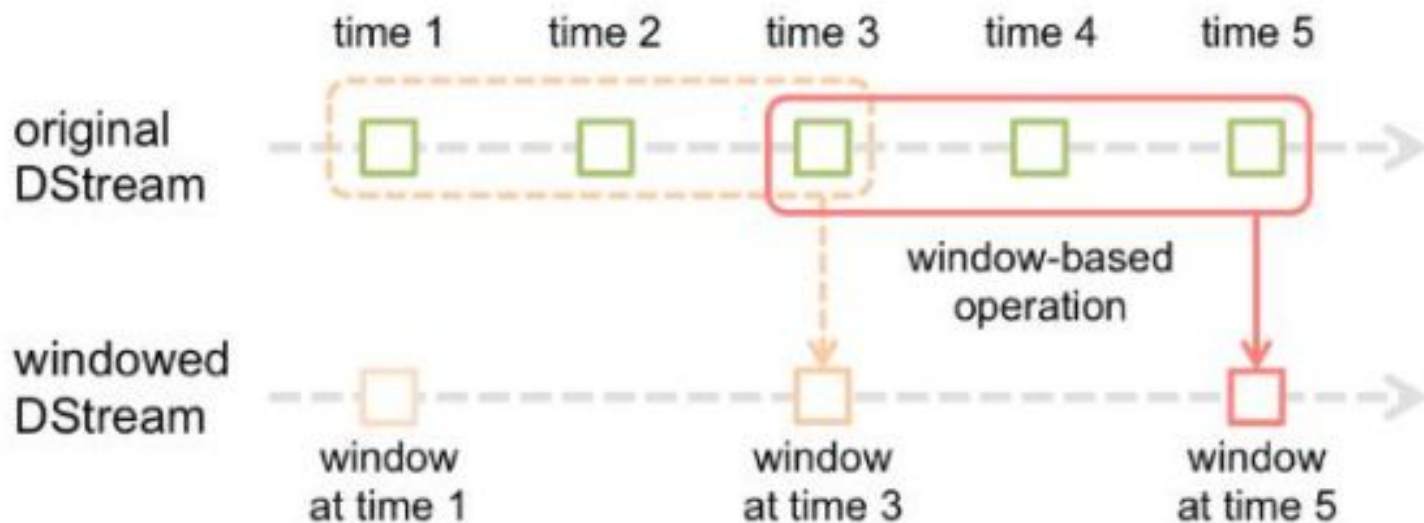
Spark Streaming



```
def updateFunc(new_values, last_sum):  
    return sum(new_values) + (last_sum or 0)
```

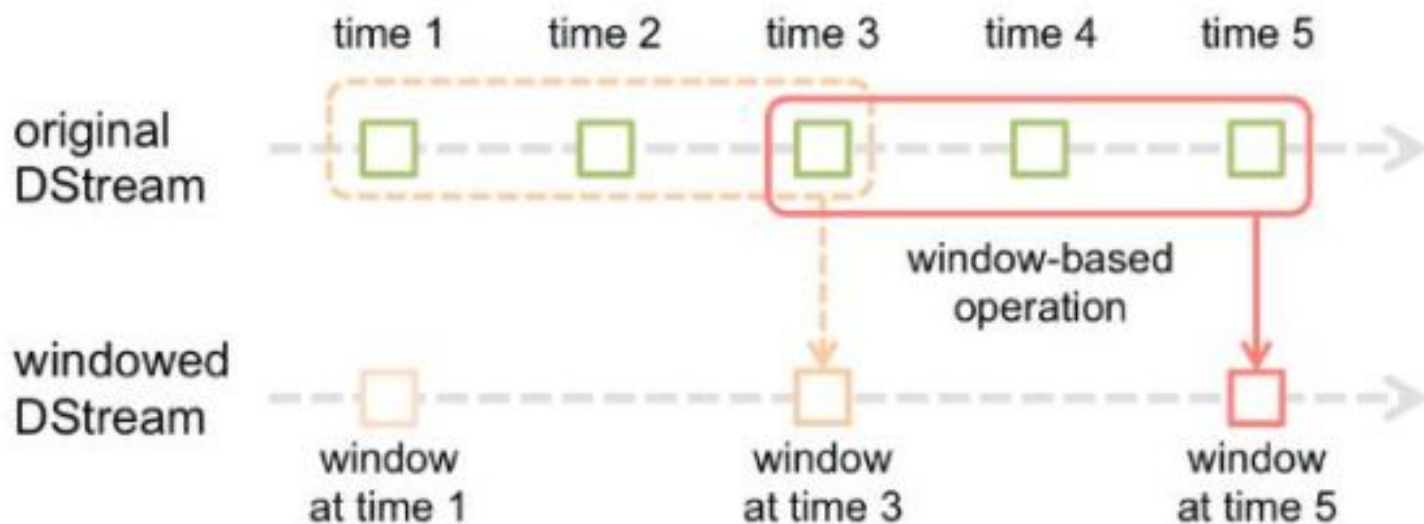
Spark Streaming

`reduceByKeyAndWindow(fn,minutes)`



Spark Streaming

`ReduceByKeyAndWindow(fn,minutes,minutes)`



Spark Streaming



Práctica guiada 1

1. Inicia spark Streaming con 2 hilos (pyspark --master local[2])
2. Abre una terminal en un puerto concreto el que quieras donde escribiremos lo que queramos
3. Paralelamente iniciaremos spark Streaming e iremos recogiendo todo lo que escribimos en la terminal e irá contando las veces que aparece cada palabra.

Práctica Repaso



Práctica guiada 2

Análisis de bizum en tiempo real.

Vamos a simular los bizums que nos llegan a nuestra cuenta y estos, se van a almacenar en un directorio en formato texto.

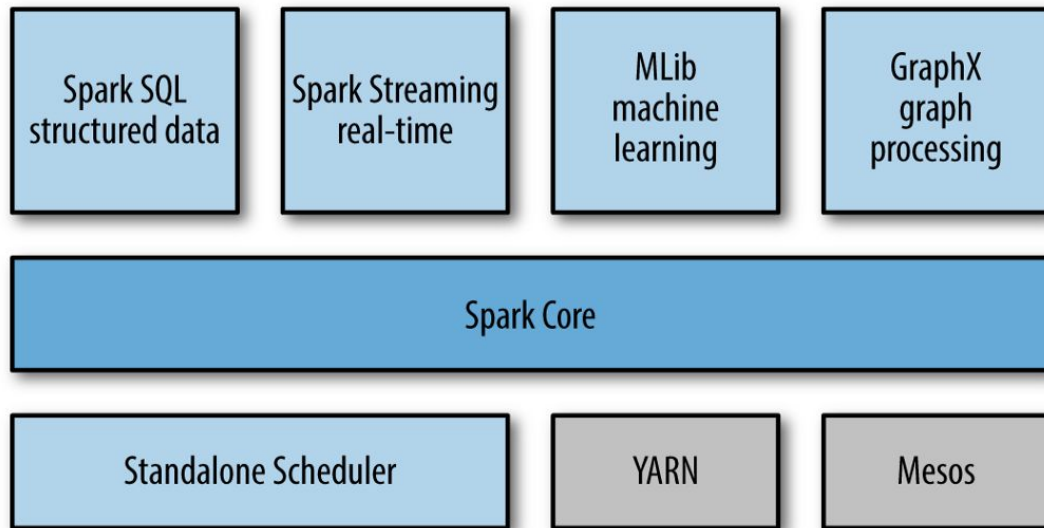
En tiempo real queremos saber cual es el bizum más alto por cada persona y concepto.

SPARK EN CLUSTER

- Spark en local (como hemos trabajado hasta ahora)
- Spark en local (con varios hilos)

Modo cluster:

- Spark Stand Alone
- Apache Hadoop YARM
- Apache Mesos

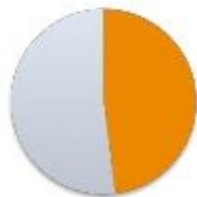


SPARK EN CLUSTER



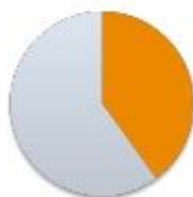
Common Deployment Patterns

Most Common Spark Deployment Environments
(Cluster Managers)



48%

Standalone mode



40%

YARN



11%

Mesos

Source: Spark Survey Report, 2015 (Databricks)

SPARK EN CLUSTER



Ventajas de ejecutar en Cluster

Procesamiento distribuido

Grandes cantidades de datos de forma eficiente

Tolerancia a fallos y escalabilidad

SPARK EN CLUSTER



Nodo: Es una máquina individual que puede ser:

Maestro: Nodos que distribuyen el trabajo

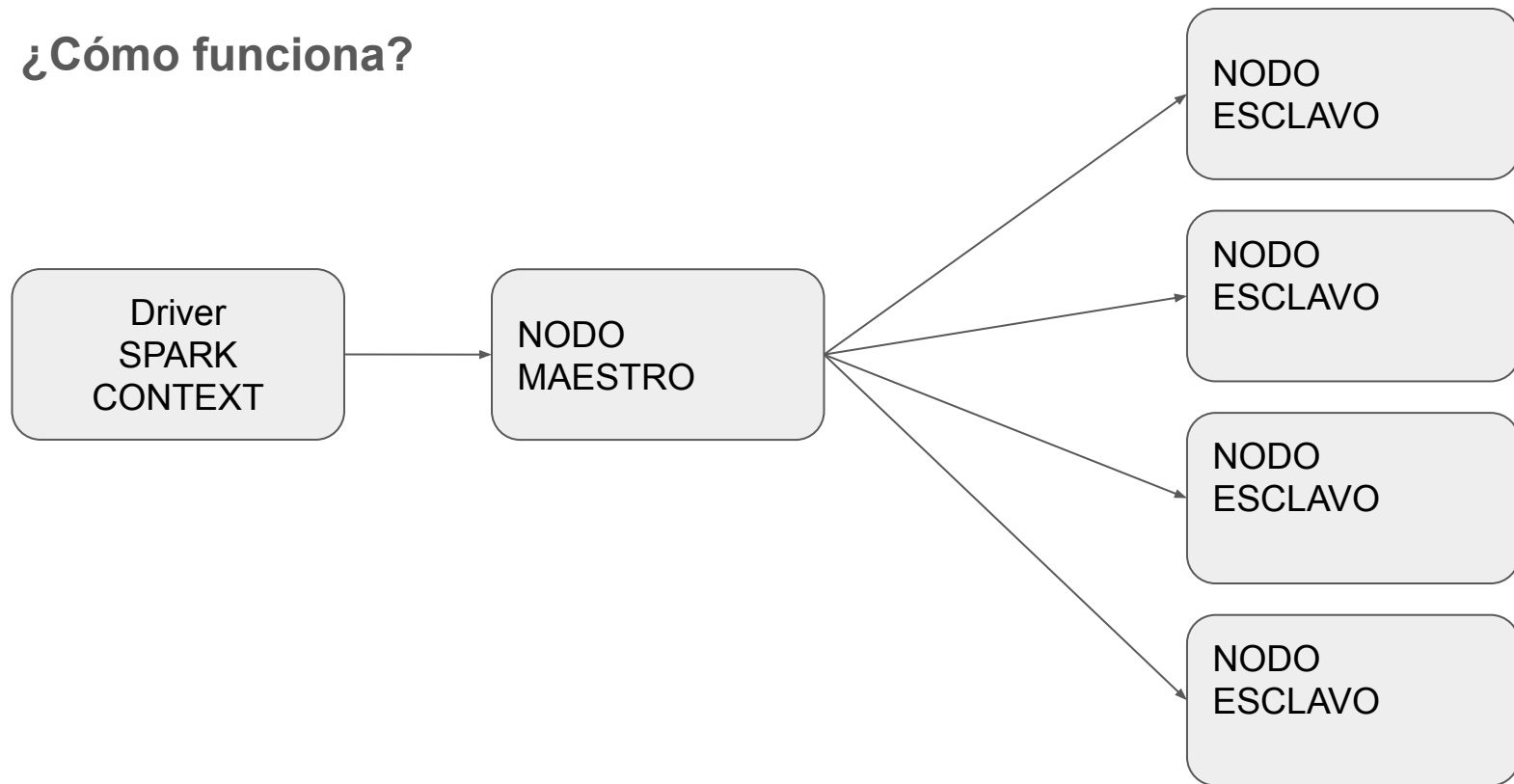
Esclavo o worker: Los que realizan el trabajo

Cluster: Conjunto de nodos

Demonios: Programas que ejecutan los nodos

SPARK EN CLUSTER

¿Cómo funciona?



SPARK EN CLUSTER



Iniciar un cluster

url- del cluster

local[*] - Ejecutar localmente con tantos hilos como cores

local[n] - Ejecutar localmente con n hilos worker

local - Ejecutar localmente pero sin procesamiento distribuido (como hasta ahora)

spark-shell --master spark://masternode:7077

SPARK EN CLUSTER



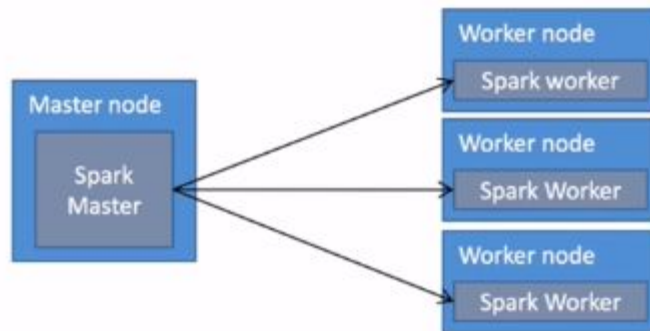
Servicios (demonios) en Spark Standalone

- **Spark master**

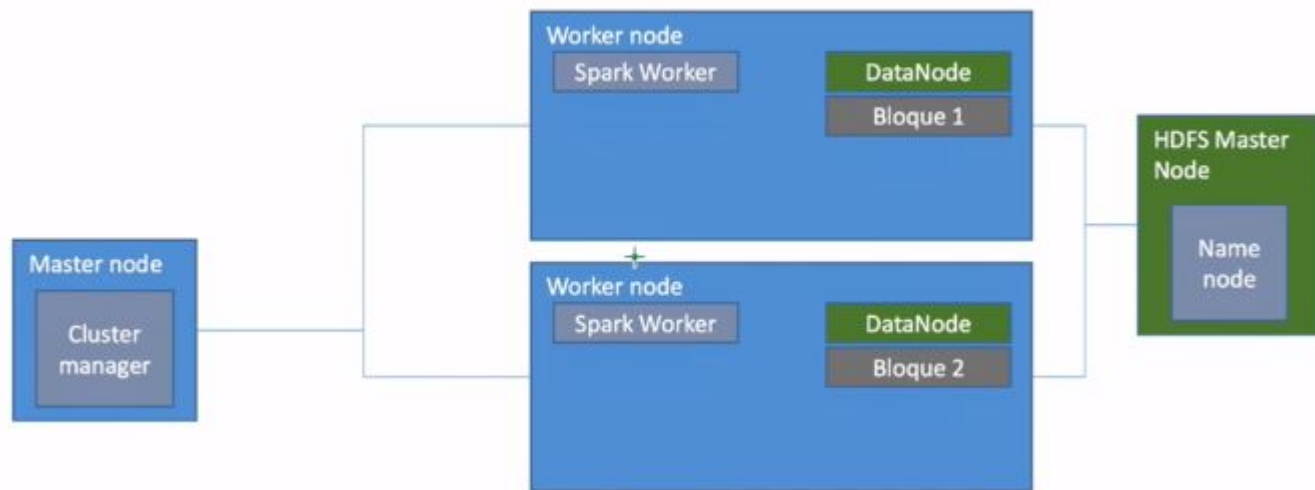
Uno por cluster (Gestiona las aplicaciones y distribuye tareas entre los Spark Workers)

- **Spark Worker**

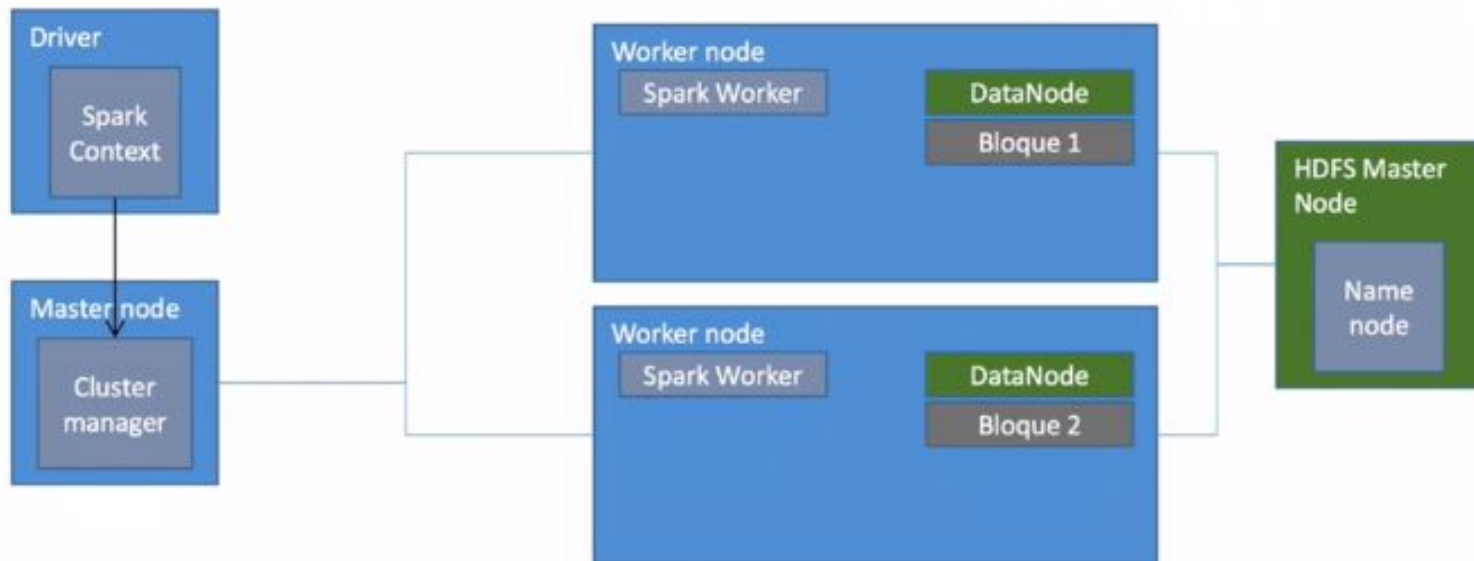
Uno por cada nodo Worker (arranca y monitoriza Executors para las aplicaciones)



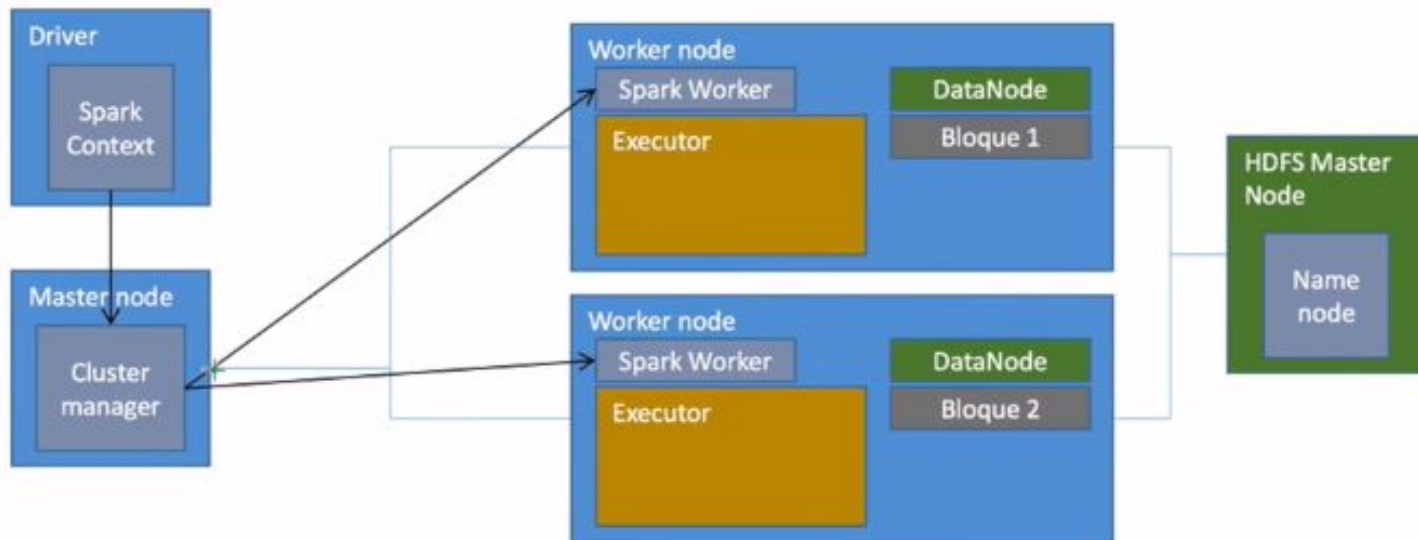
SPARK EN CLUSTER



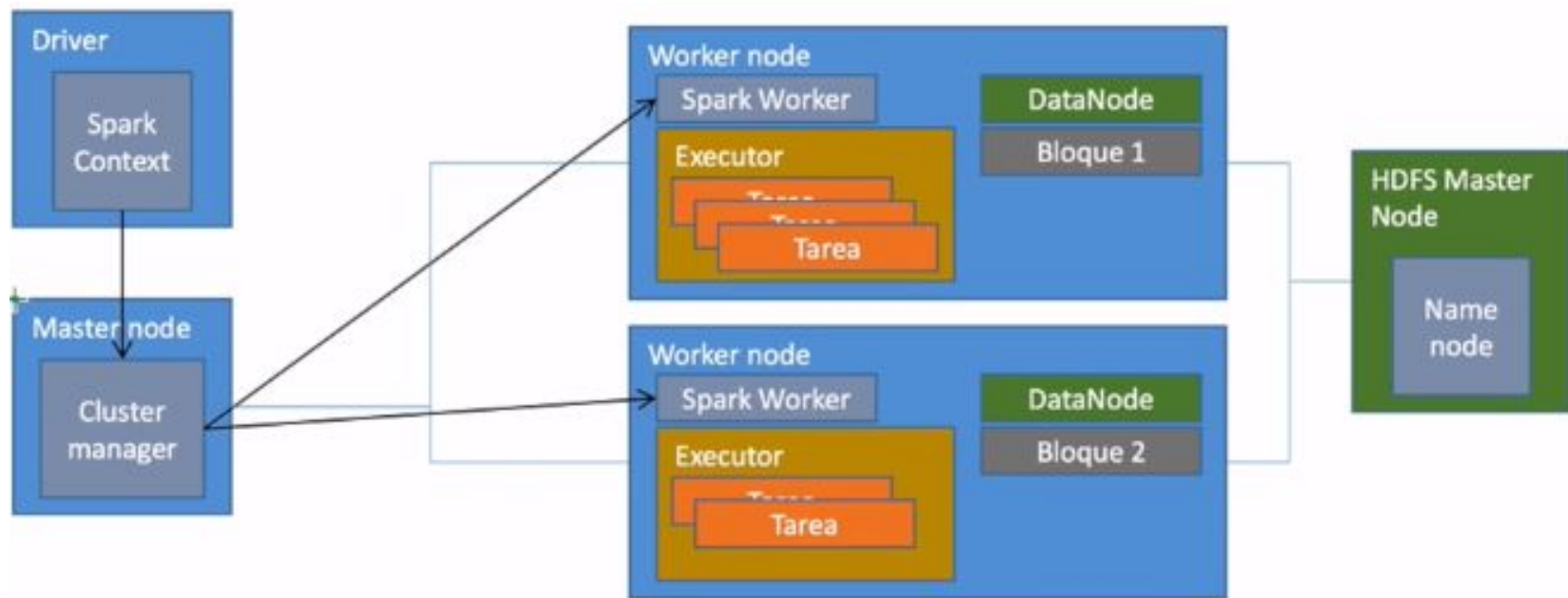
SPARK EN CLUSTER



SPARK EN CLUSTER



SPARK EN CLUSTER



SPARK EN CLUSTER



Spark Master at spark://172.17.0.2:7077

URL: spark://172.17.0.2:7077

REST URL: spark://172.17.0.2:6066 (cluster mode)

Workers: 3

Cores: 6 Total, 0 Used

Memory: 19.9 GB Total, 0.0 B Used

Applications: 0 Running, 3 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20160105084533-172.17.0.3-8888	172.17.0.3:8888	ALIVE	2 (0 Used)	6.6 GB (0.0 B Used)
worker-20160105084952-172.17.0.4-8888	172.17.0.4:8888	ALIVE	2 (0 Used)	6.6 GB (0.0 B Used)
worker-20160105085704-172.17.0.5-8888	172.17.0.5:8888	DEAD	2 (0 Used)	6.6 GB (0.0 B Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
----------------	------	-------	-----------------	----------------	------	-------	----------

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20160105085716-0002	Spark shell	6	512.0 MB	2016/01/05 08:57:16	root	FINISHED	13 min
app-20160105085023-0001	Spark shell	4	512.0 MB	2016/01/05 08:50:23	root	FINISHED	6.4 min
app-20160105084640-0000	Spark shell	2	512.0 MB	2016/01/05 08:46:40	root	FINISHED	3.0 min

SPARK EN CLUSTER



Application: Spark shell

ID: app-20160105085716-0002

Name: Spark shell

User: root

Cores: Unlimited (6 granted)

Executor Memory: 512.0 MB

Submit Date: Tue Jan 05 08:57:16 UTC 2016

State: FINISHED

[Application Detail UI](#)

Executor Summary

ExecutorID	Worker	Cores	Memory	State	Logs
------------	--------	-------	--------	-------	------

Removed Executors

ExecutorID	Worker	Cores	Memory	State	Logs
2	worker-20160105084952-172.17.0.4-8888	2	512	KILLED	stdout stderr
1	worker-20160105085704-172.17.0.5-8888	2	512	KILLED	stdout stderr
0	worker-20160105084533-172.17.0.3-8888	2	512	KILLED	stdout stderr

SPARK EN CLUSTER



Gestor de recursos

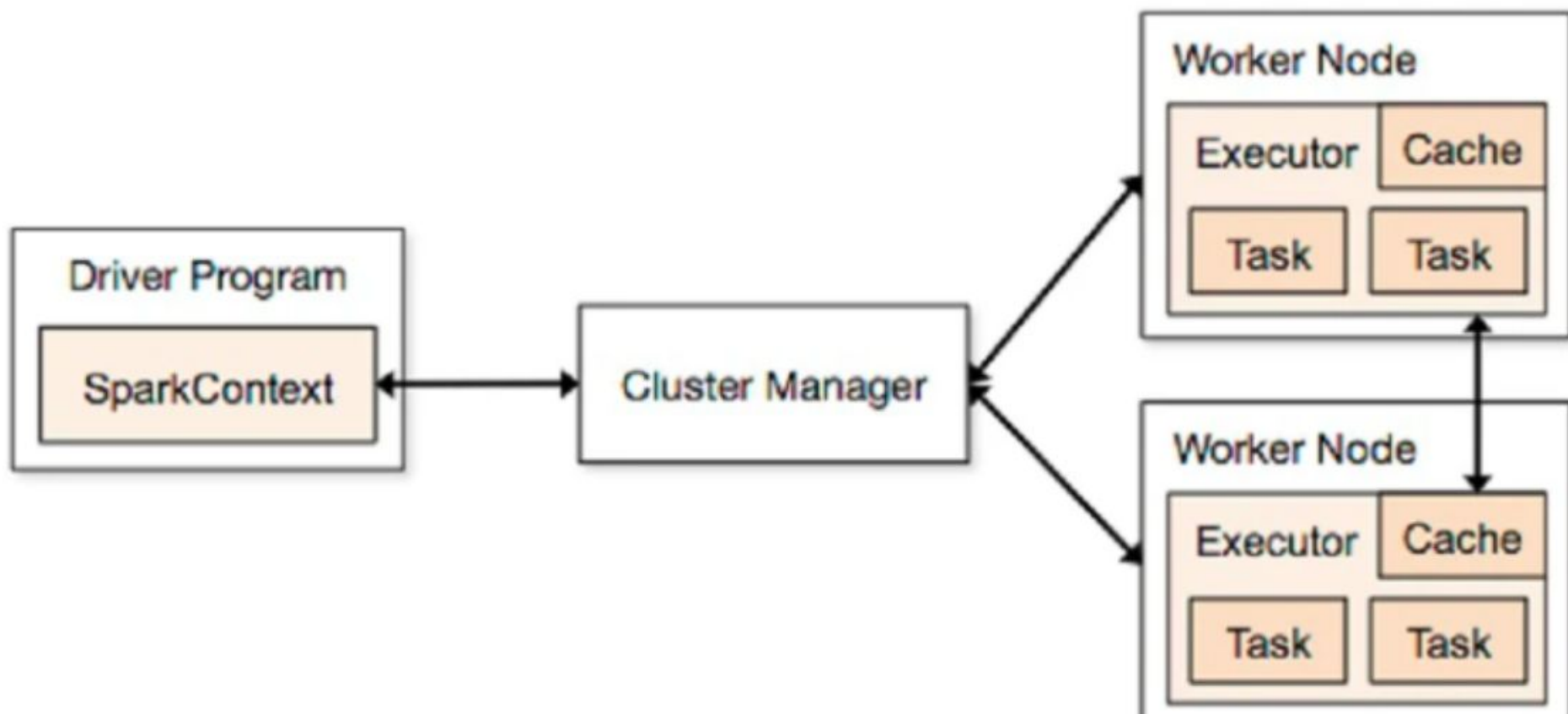
Hadoop YARM

- Es el más común
- Permite compartir recursos del cluster con otras aplicaciones(hive,impala,pig..)

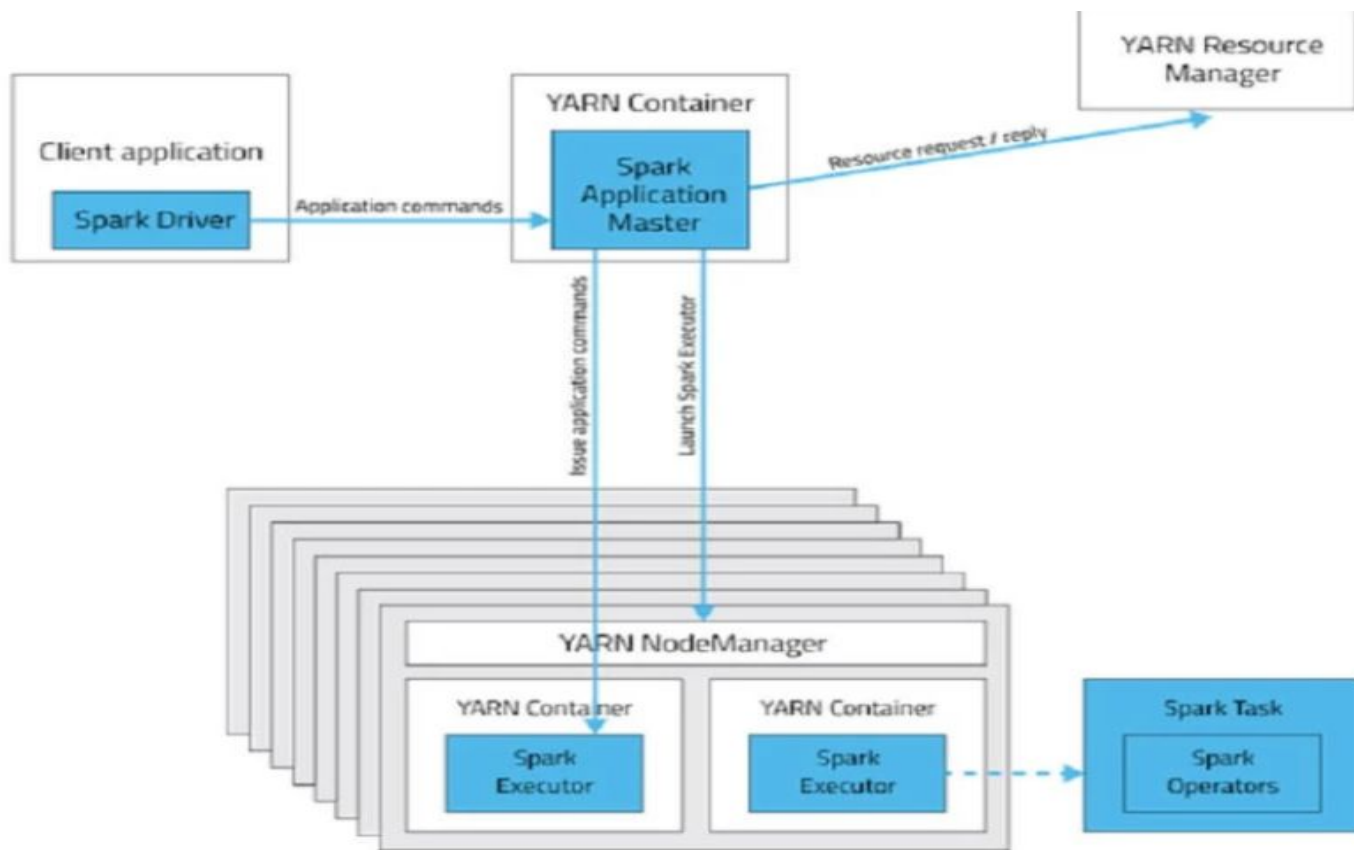
Apache MESOS

- Fue la primera plataforma que soportaba spark
- No se usa casi hoy en dia

SPARK EN CLUSTER

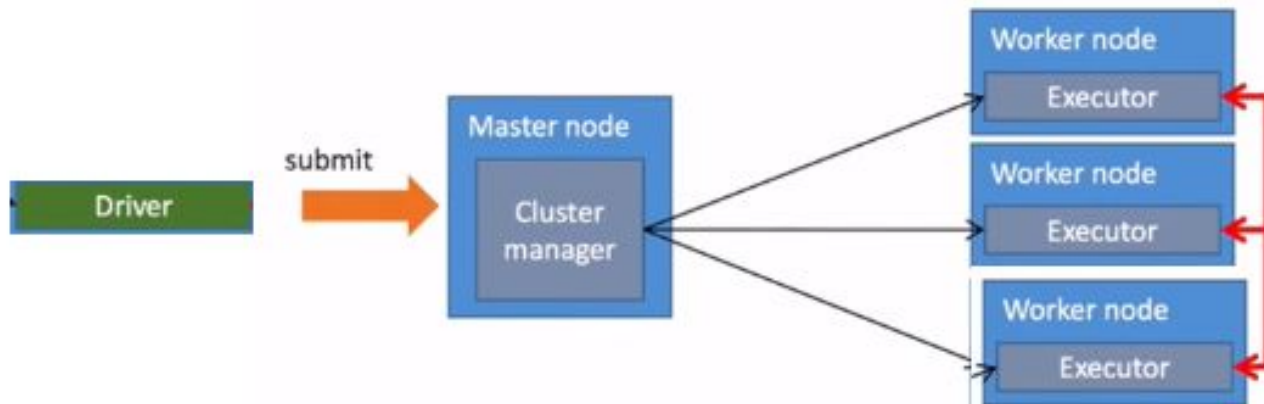


SPARK EN CLUSTER

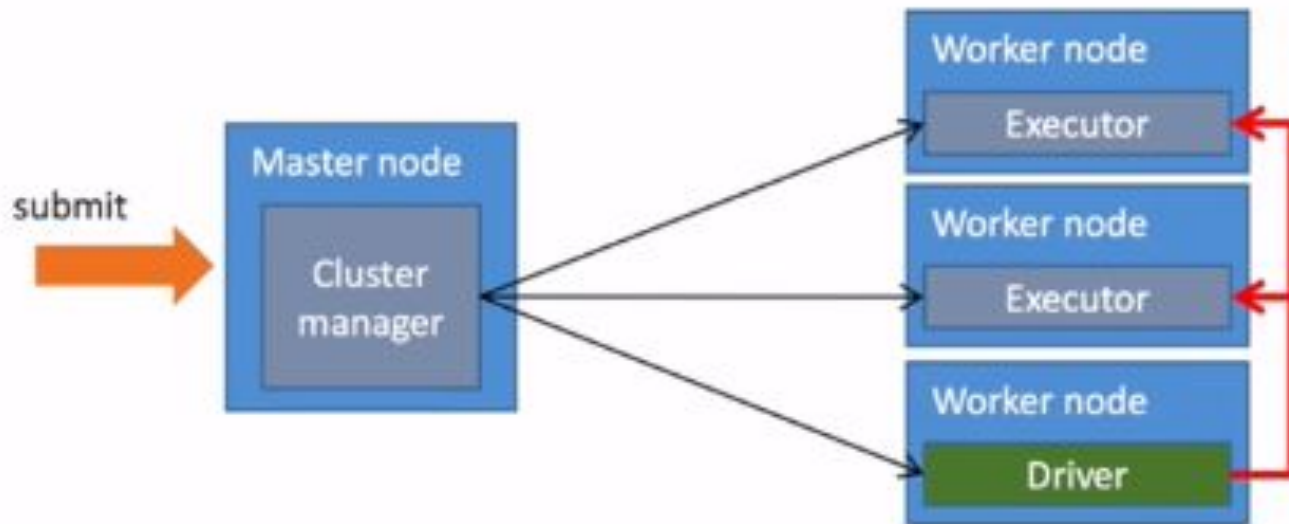


SPARK EN CLUSTER

Ejecución del driver



SPARK EN CLUSTER



PARALELIZACIÓN



Master node

Worker node

- Executor
 - Task
- RDD Distribuido entre los nodos
- Distribución del dataset automática

PARALELIZACIÓN



¿Como se particiona un RDD que viene de fichero?

`sc.textfile(fichero)` -> particiona por defecto

`sc.textfile(fichero,num)`-> elegimos nosotros las particiones

Múltiples ficheros

`sc.textfile(/*)`

`sc.wholeTextFiles(ruta)`

PARALELIZACIÓN



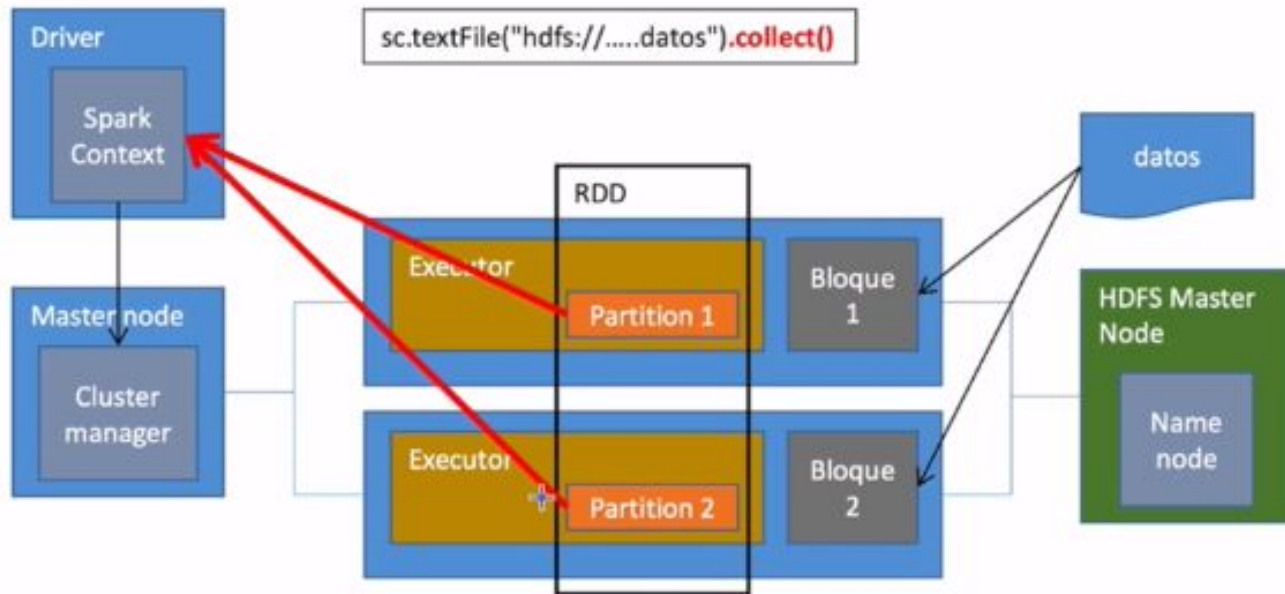
Operaciones sobre RDD

- La gran mayoría

Operaciones sobre particiones

- `foreachpartition`
- `mappartition`
- `mappartitionwithindex`

PARALELIZACIÓN



PARALELIZACIÓN



Funciones que trabajan en particiones

MAP

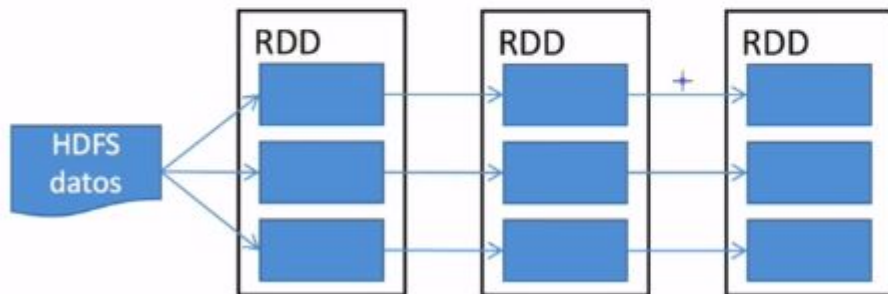
FLATMAP

...

Trabajan independientemente en su partición

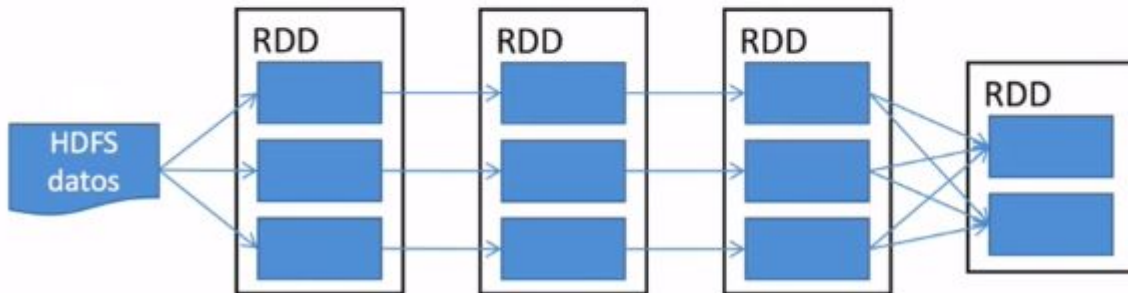
PARALELIZACIÓN

```
avglens = sc.textFile(file) \  
  .flatMap(lambda linea: linea.split()) \  
  .map(lambda palabra: (palabra[0],len(palabra))) \  
  .reduceByKey(lambda a,b: a+b)
```



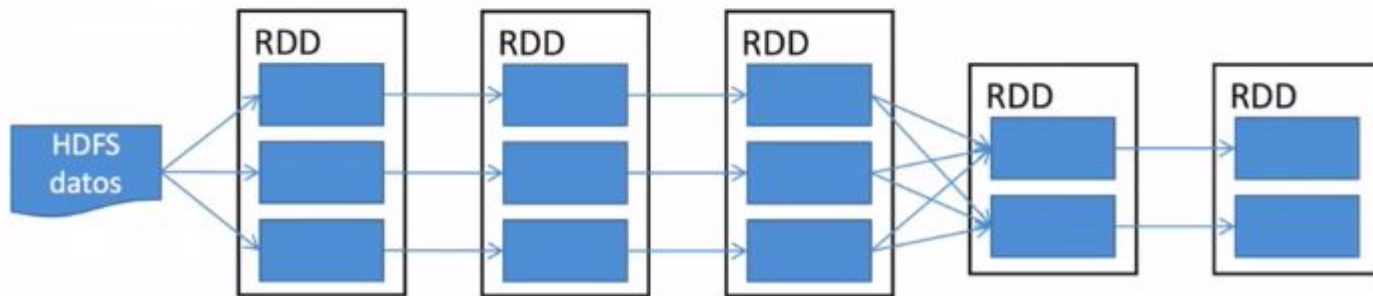
PARALELIZACIÓN

```
avgLens = sc.textFile(file) \  
  .flatMap(lambda linea: linea.split()) \  
  .map(lambda palabra: (palabra[0],len(palabra))) \  
  .groupByKey() \  
+
```

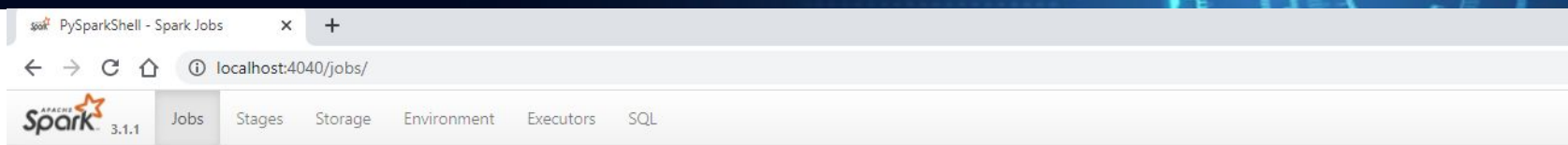


PARALELIZACIÓN

```
avglens = sc.textFile(file) \  
  .flatMap(lambda linea: linea.split()) \  
  .map(lambda palabra: (palabra[0],len(palabra))) \  
  .groupByKey() \  
  .map(lambda (k,valores): \  
    (k, sum(valores)/len(valores)))
```



Interfaz de usuario (Web)



Spark Jobs (?)

User: fedev

Total Uptime: 21,2 h

Scheduling Mode: FIFO

Failed Jobs: 1

▼ Event Timeline

☐ Enable zooming



▼ Failed Jobs (1)

Page: 1

Job Id ▼	Description	Submitted	Duration	Stages: Succeeded/Total
0	count (*) table...	2021-05-14 17:15:04	2 s	0/1 (1 failed)

Interfaz de usuario (Web)



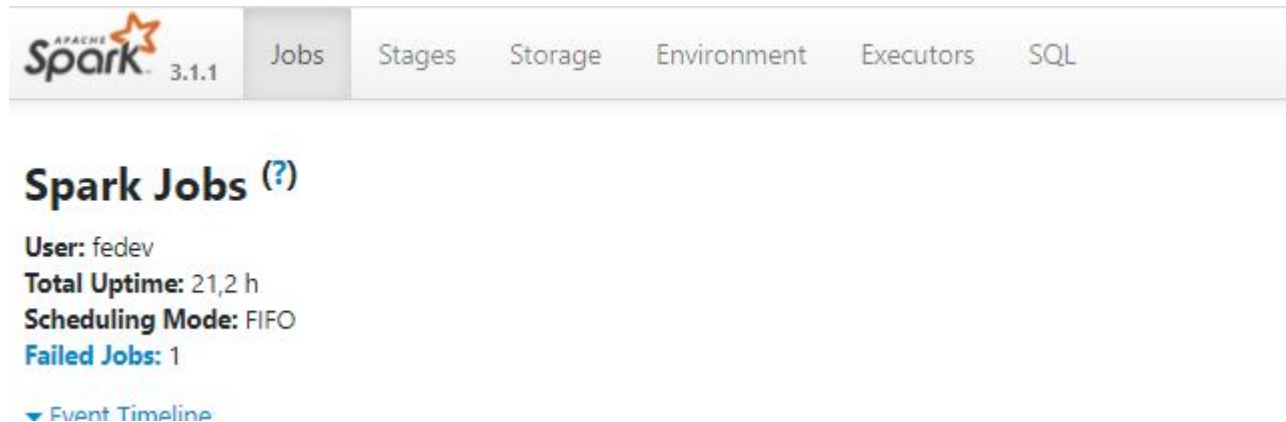
Pestañas de la interfaz

- Trabajos de Spark
- Etapas
- Tareas
- Almacenamiento
- Ambiente
- Ejecutores
- SQL

Acceso: localhost:4040

Interfaz de usuario (Web)

Spark Jobs




The screenshot displays the Apache Spark 3.1.1 web interface. At the top, there is a navigation bar with the following tabs: Jobs (selected), Stages, Storage, Environment, Executors, and SQL. Below the navigation bar, the main content area shows the 'Spark Jobs' section with a help icon (?). The status information provided is as follows:

- User:** fedev
- Total Uptime:** 21,2 h
- Scheduling Mode:** FIFO
- Failed Jobs:** 1

At the bottom of the section, there is a link labeled 'Event Timeline' with a downward-pointing triangle icon.

Interfaz de usuario (Web)

Stages

 3.1.1

[Jobs](#) **[Stages](#)** [Storage](#) [Environment](#) [Executors](#) [SQL](#)

Stages for All Jobs

Completed Stages: 3
Failed Stages: 7

▼ Completed Stages (3)

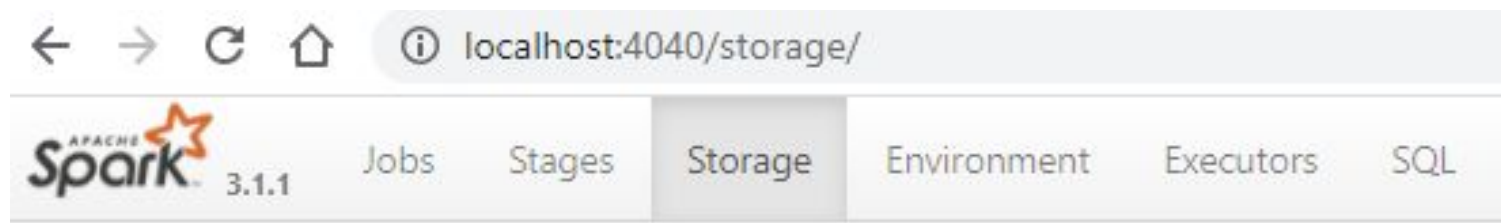
Page:

Stage Id ▼	Description	
9	collect at <stdin>:1	+details 2
8	collect at <stdin>:1	+details 2
3	collect at <stdin>:1	+details 2

Page:

Interfaz de usuario (Web)

Storage



Storage

Interfaz de usuario (Web)

Environment



Environment

- ▶ [Runtime Information](#)
- ▶ [Spark Properties](#)
- ▶ [Resource Profiles](#)
- ▶ [Hadoop Properties](#)
- ▼ [System Properties](#)

Name
SPARK_SUBMIT
awt.toolkit
file.encoding

Interfaz de usuario (Web)

Executors



Executors

[▶ Show Additional Metrics](#)

Summary

	▲ RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks
Active(1)	0	164.3 KiB / 366.3 MiB	0.0 B	8	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0
Total(1)	0	164.3 KiB / 366.3 MiB	0.0 B	8	0

Executors

Show entries

Executor ID	▲ Address	Status	▲ RDD Blocks	Storage Memory	Disk Used	Cores
driver	192.168.1.65:57811	Active	0	164.3 KiB / 366.3 MiB	0.0 B	8

Interfaz de usuario (Web)

SQL



SQL