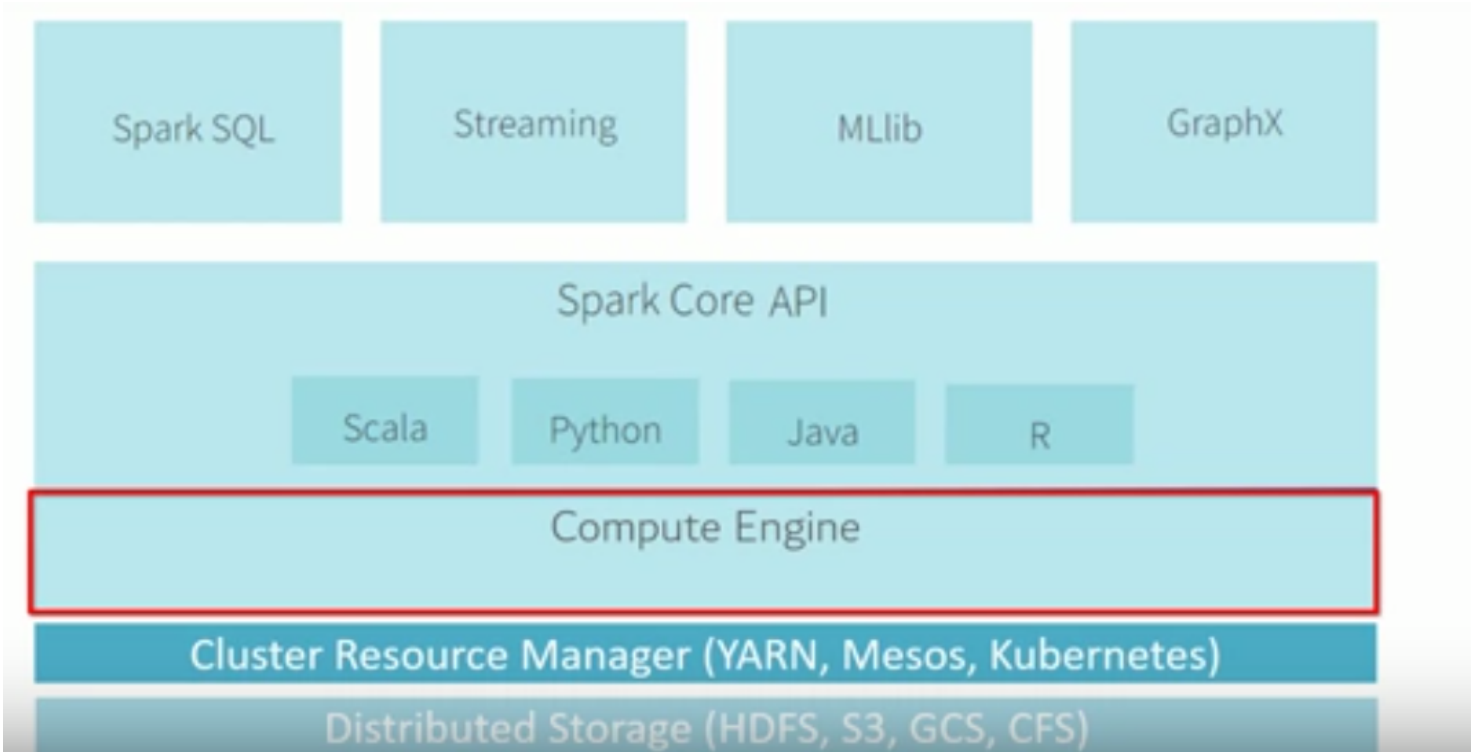
Spark Architecture

Notebook:	Spark		
Created:	5/1/2018 4:38 PM	**Updated:**	5/1/2018 11:31 PM
Author:	ashu41228@gmail.com		

Spark Architecture



Apache Spark™ is a fast and general engine for large-scale data processing.

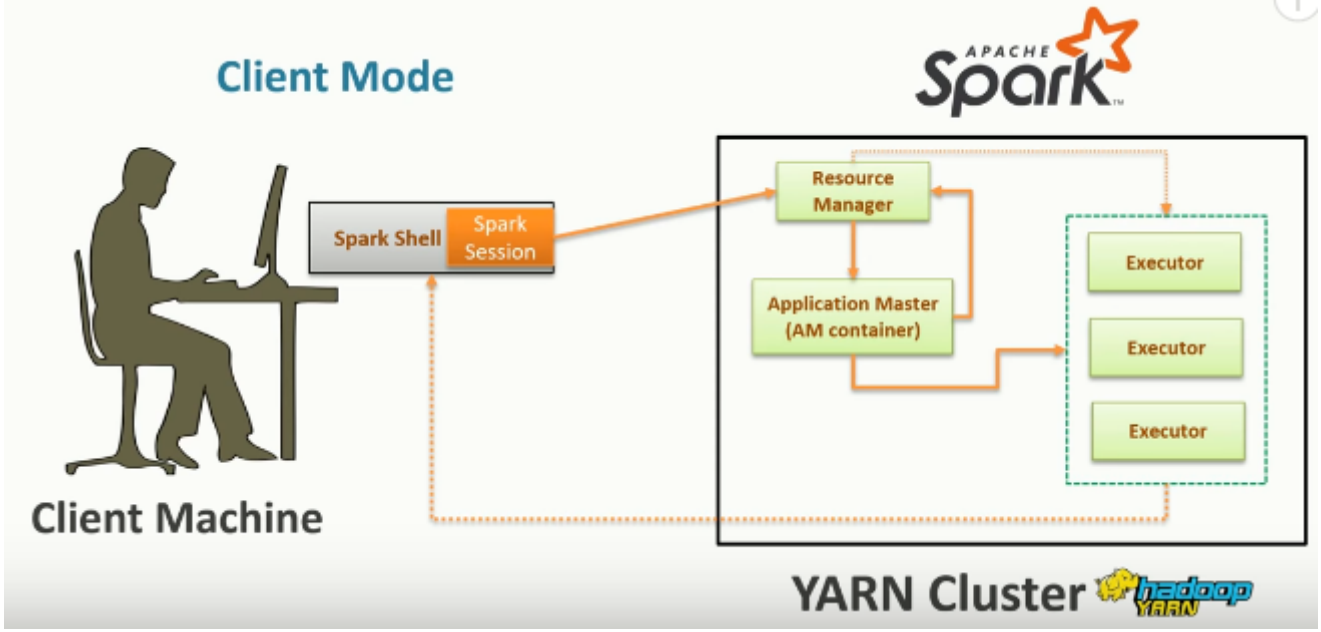
- **How do we execute programs on a Spark Cluster?**
- **Interactive Clients (Scala Shell, Pyspark, Notebooks)**
- **Submit a Job (Spark submit utility)**

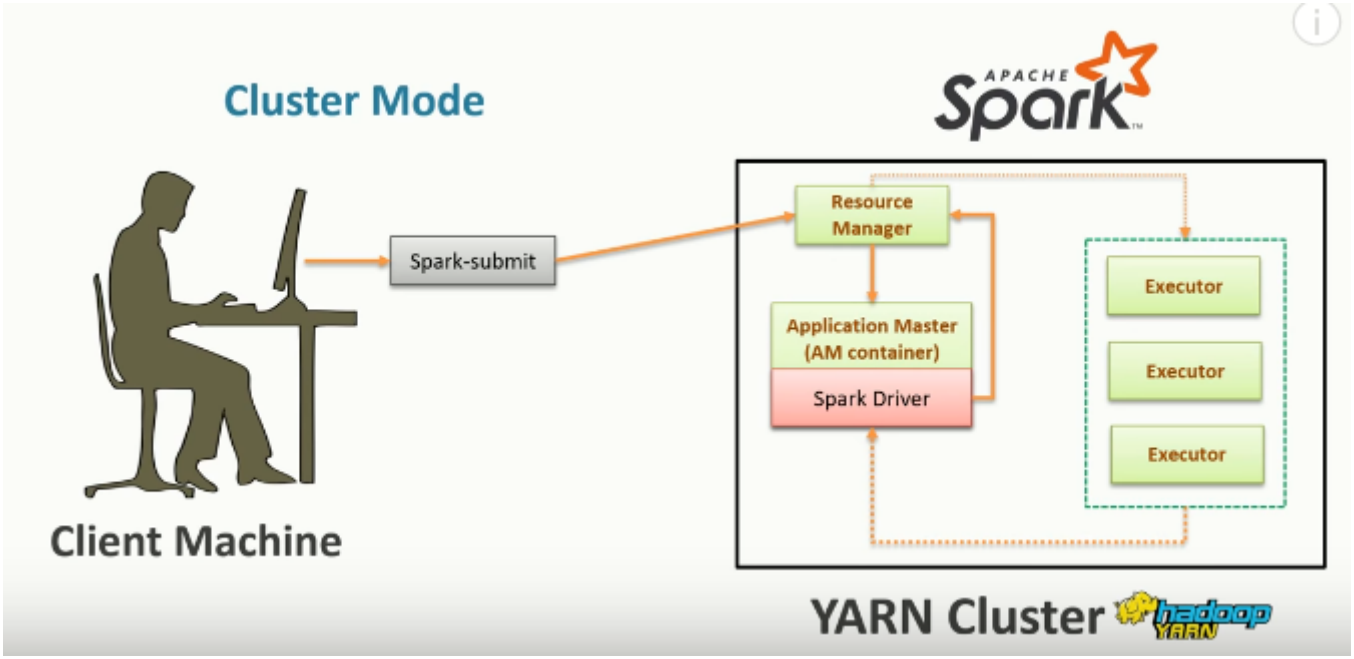
How does the Spark execute our programs on a cluster?

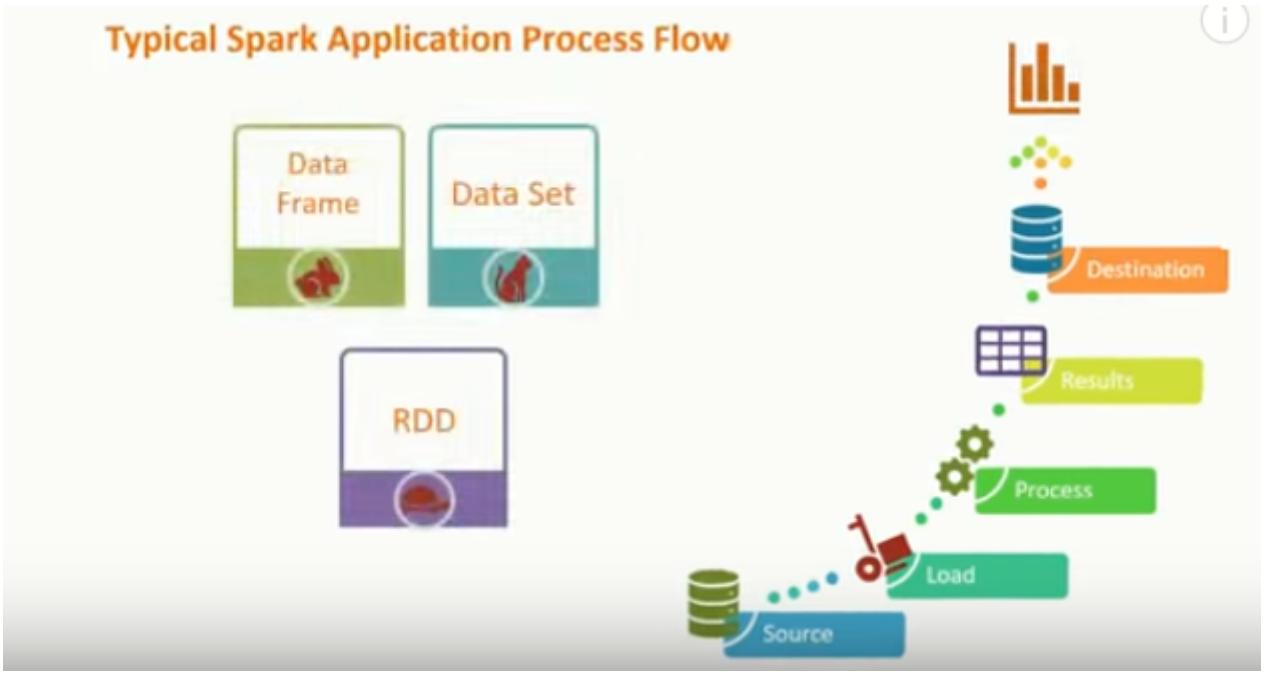
- Spark creates one driver and a bunch of executors for each application.

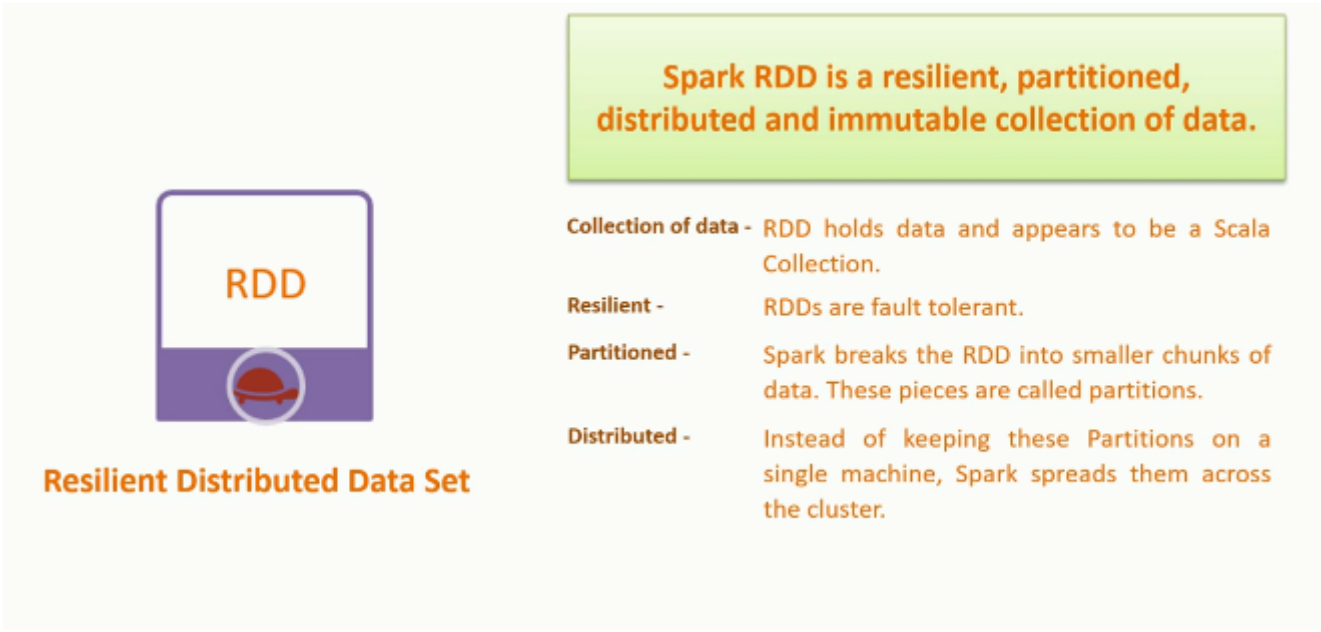
- Spark offers two deployment modes for an application.
  - Client Mode - Driver on client machine and Executors on cluster
  - Cluster Mode - Driver and executors on cluster

Client Mode

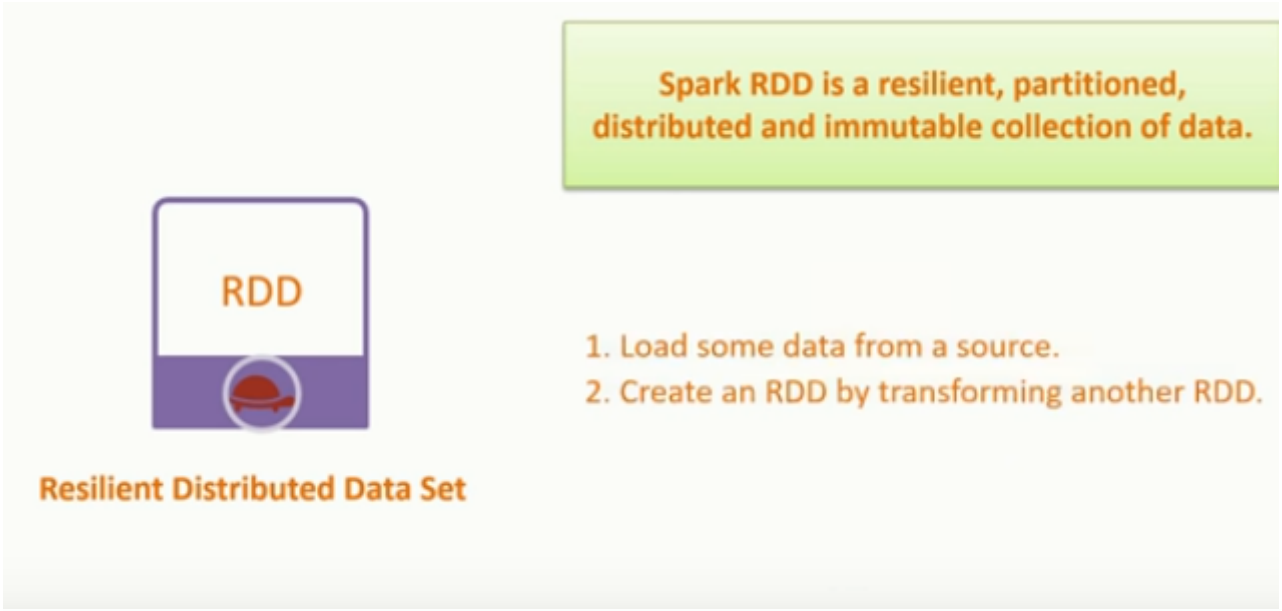


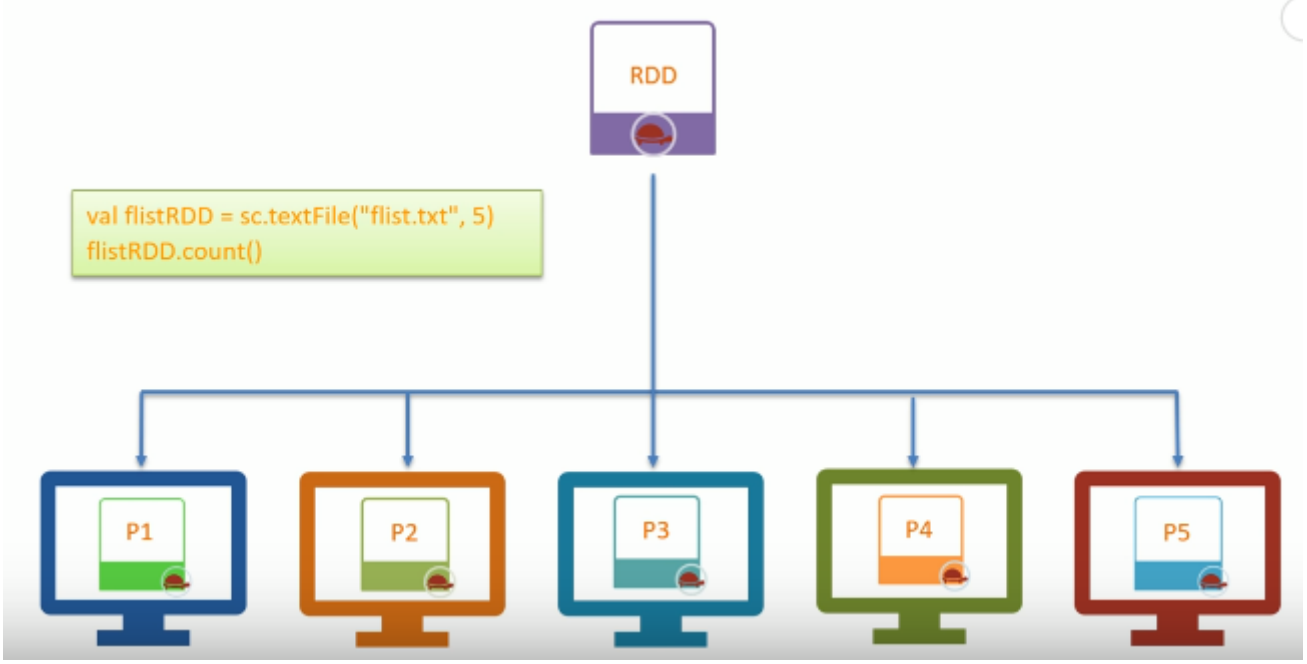

Cluster Mode
APACHE Spark
Spark-submit
Resource Manager
Application Master (AM container)
Spark Driver
Executor
Executor
Executor
Client Machine
YARN Cluster
hadoop YARN


Typical Spark Application Process Flow
Data Frame
Data Set
RDD
Destination
Results
Process
Load
Source



How to create an RDD





File will be split into 5 partitions and will be processed on 5 different Executers. The number of Partitions is equal to the number of executers

Transformations are lazy they don't perform computation until an action method is invoked its like pig syntax when we do dump then only operation perform in DAG manner

Word Count Program below

Scala

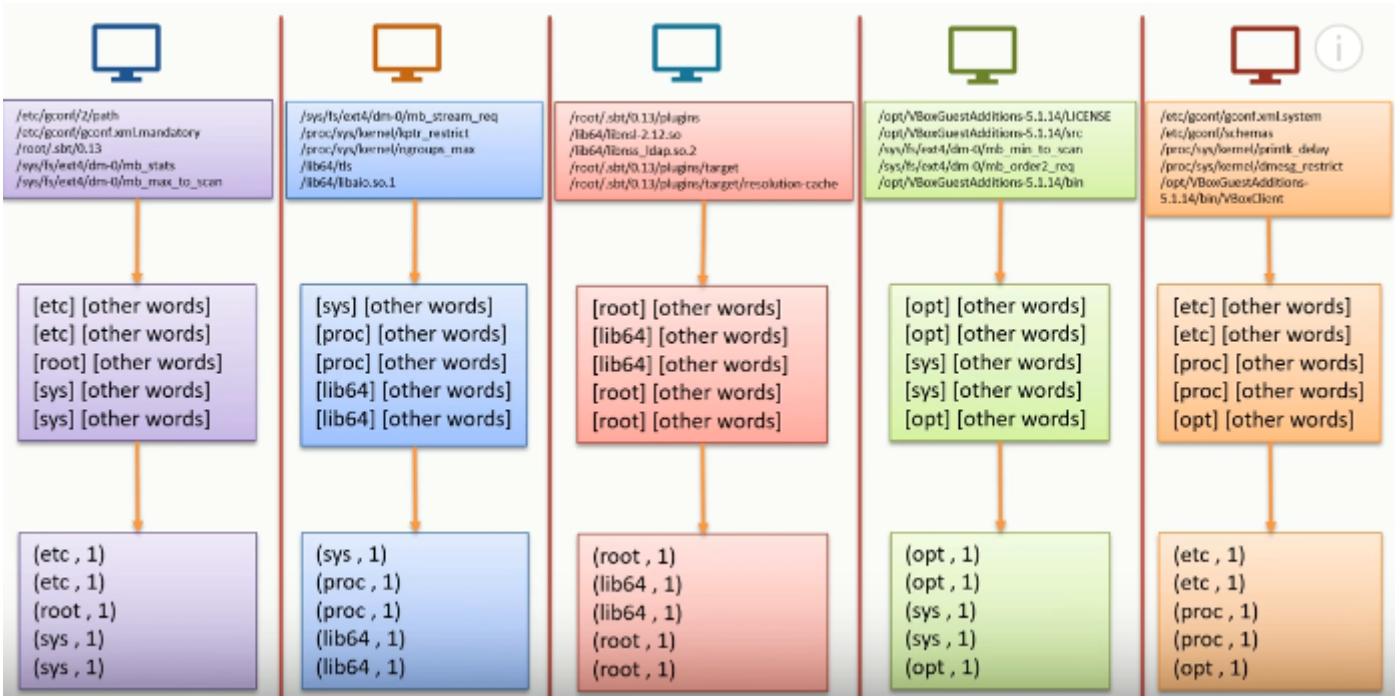
```
val flistRDD = sc.textFile("/user/prashant/flist.txt", 5)
val arrayRDD = flistRDD.map(x=> x.split("/"))
val kvRDD = arrayRDD.map(a => (a(0),1))
val fcountRDD= kvRDD.reduceByKey((x,y)=> x+y)
fcountRDD.collect()
```

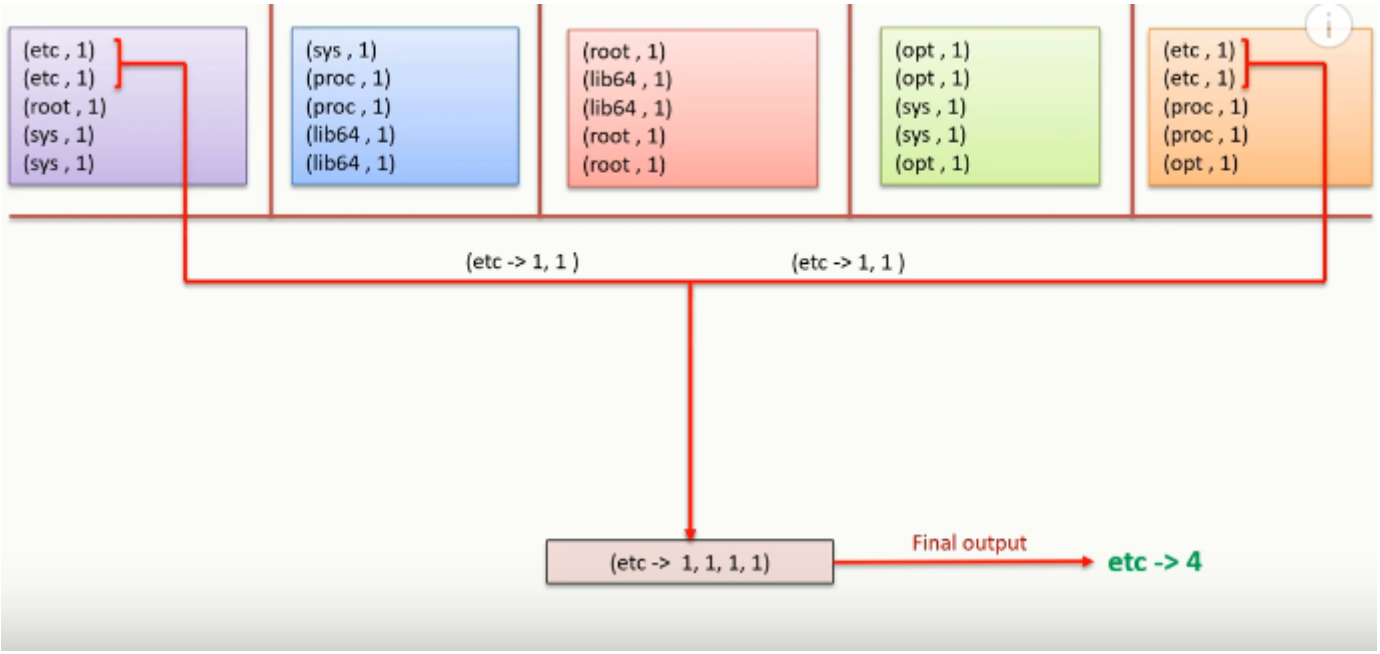
Python

```
flistRDD = sc.textFile("/user/prashant/flist.txt", 5)
arrayRDD = flistRDD.map(lambda x: x.split("/"))
kvRDD = arrayRDD.map(lambda a: (a[0],1))
fcountRDD= kvRDD.reduceByKey(lambda x,y: x+y)
fcountRDD.collect()
```

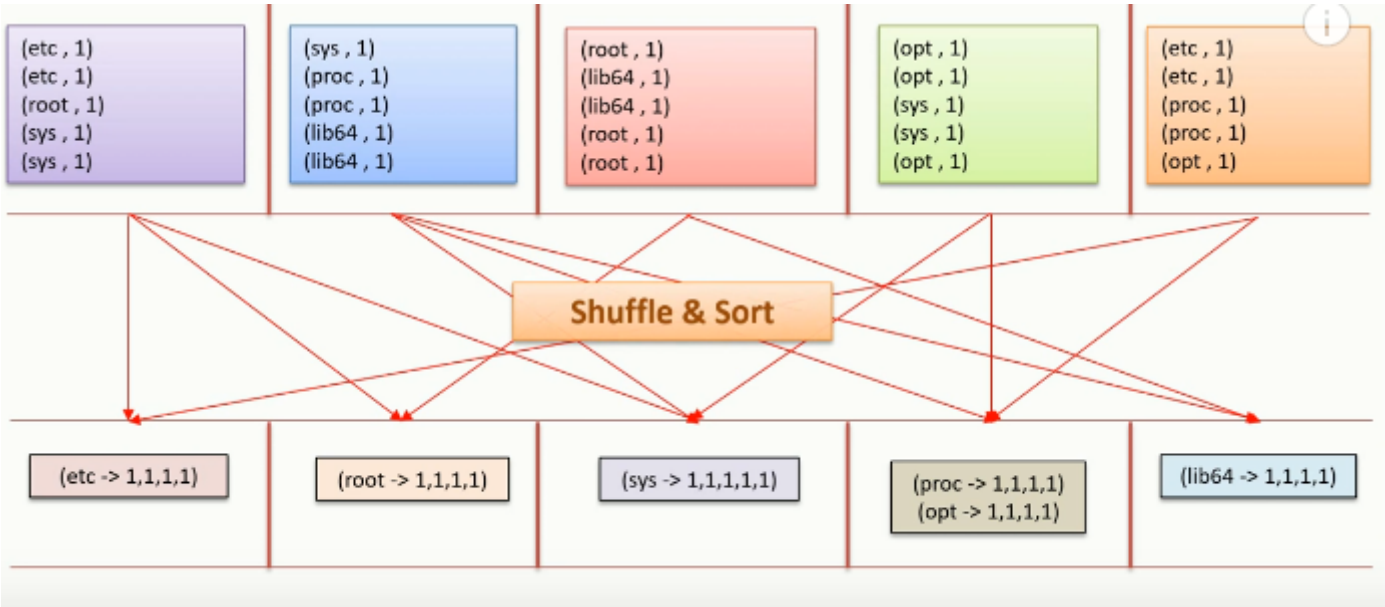
Sample from the data file

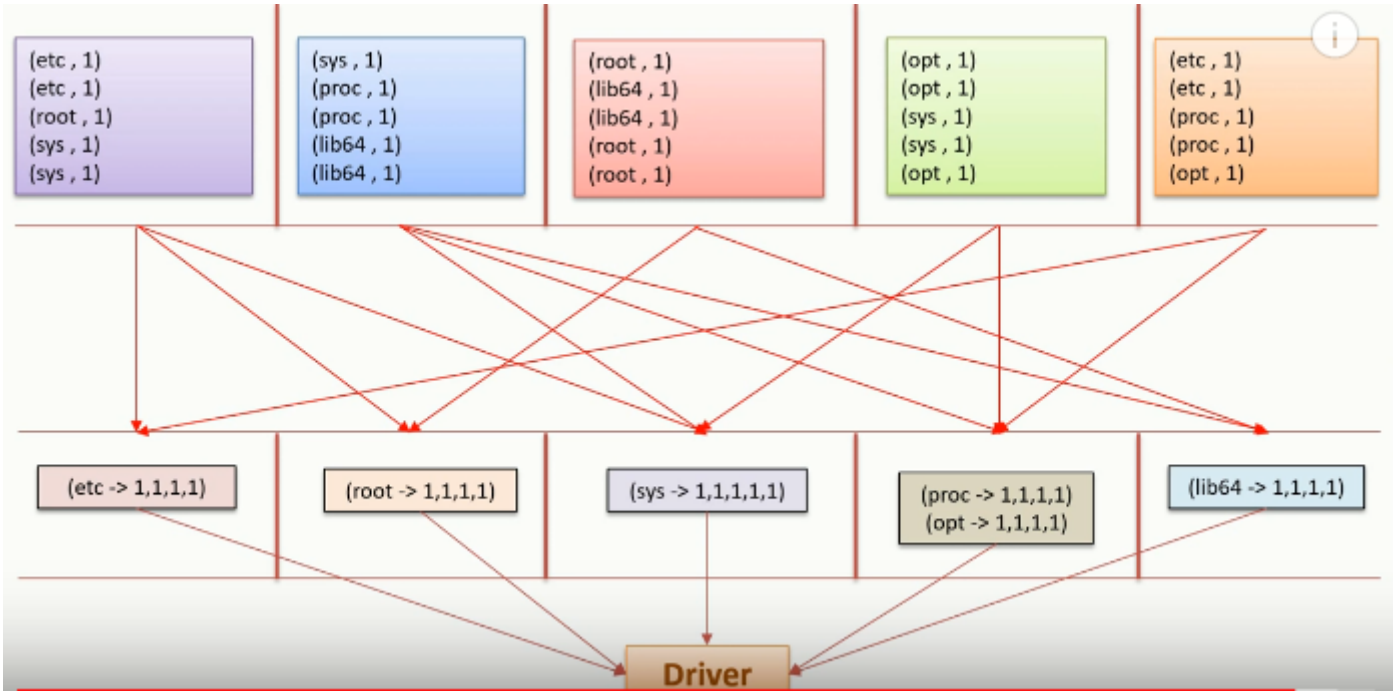
```
/etc/abrt
/etc/abrt/plugins
/etc/abrt/plugins/CCpp.conf
/etc/reader.conf
/etc/fonts
/etc/fonts/fonts.conf
/etc/fonts/fonts.dtd
```





The Above operation is performed by below : re-partitioning is done based on shuffle and sort maintained by spark internally




(etc , 1)
(etc , 1)
(root , 1)
(sys , 1)
(sys , 1)
(sys , 1)
(proc , 1)
(proc , 1)
(lib64 , 1)
(lib64 , 1)
(root , 1)
(lib64 , 1)
(root , 1)
(root , 1)
(opt , 1)
(opt , 1)
(sys , 1)
(sys , 1)
(opt , 1)
(etc , 1)
(etc , 1)
(proc , 1)
(proc , 1)
(opt , 1)
(etc -> 1,1,1,1)
(root -> 1,1,1,1)
(sys -> 1,1,1,1,1)
(proc -> 1,1,1,1)
(opt -> 1,1,1,1)
(lib64 -> 1,1,1,1)
Driver