ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Συστήματα Ανάλυσης & Διαχείρισης Μεγάλων Δεδομένων

Διδάσκων Καθηγητής
Ι. Κωτίδης

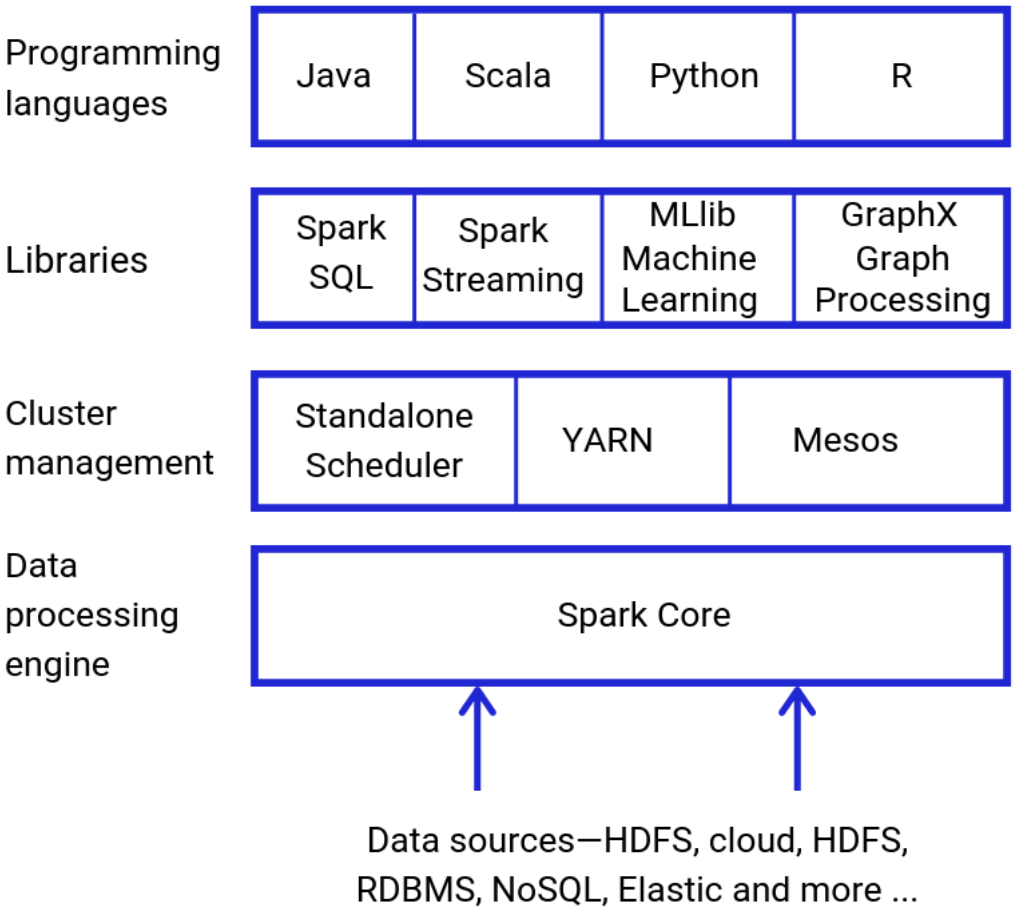
Φροντιστήριο 2

APACHE Spark™

Καπέτης Χρυσόστομος
mkap@aueb.gr

- ✓ Apache Spark is an open-source cluster-computing framework for real time processing developed by the Apache Software Foundation.
- ✓ Spark provides an interface for programming entire clusters with implicit data parallelism and fault-tolerance.
- ✓ It was built on top of Hadoop MapReduce and it extends the MapReduce model to efficiently use more types of computations.

SPARK Ecosystem



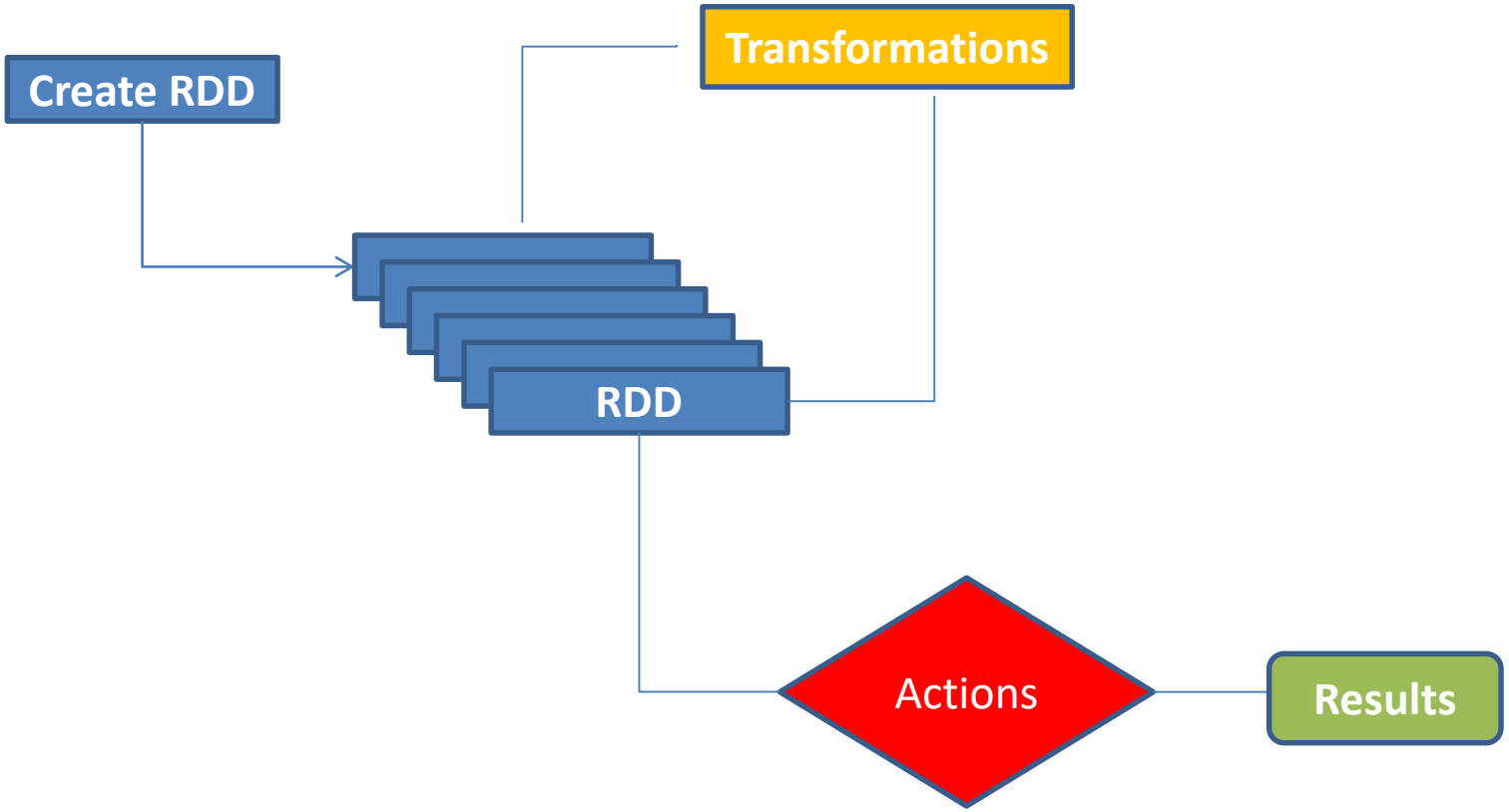
Why Spark?

- ✓ 100x faster than MapReduce for large scale data processing
- ✓ Can be deployed through Mesos, Hadoop via Yarn, or Spark's own cluster manger
- ✓ Simple programming layer provides powerful caching and disk persistence capabilities
- ✓ Can be programmed in Scala, Java, Python and R

Resilient Distributed Data-sets (RDD)

RDDs represent a collection of items distributed across many computer nodes that can be processed in parallel.

Spark's main programming abstraction



Resilient Distributed Datasets (RDD) Creation, Transformations & Actions

Τρείς τρόποι δημιουργίας ενός RDD

- Parallelize

  scala> val myRDD = sc.parallelize(List("spark", "scala", "java"))

- Χρήση ενός συνόλου δεδομένων από ένα εξωτερικό σύστημα αποθήκευσης.

  scala>val textRDD=sc.textFile("/home/lab/myprj/data/persons.txt")

- Δημιουργία ενός νέου RDD από ένα ήδη υπάρχον

  scala>val newRDD=myRDD.filter(x => x.contains("java"))

RDD Operations: Transformations

- **Filter()**: επιστρέφει ένα νέο RDD το οποίο περιέχει μόνο τα στοιχεία που ικανοποιούν κάποιο κριτήριο.

```
val  nums=sc.parallelize(List (1,2,3,4,5,6,7,8,9,10))
val evens = nums.filter(_%2==0).collect()
```

- **Intersection():** επιστρέφει την τομή δύο RDD. Προσοχή: Τα δύο RDD πρέπει να είναι του ιδίου τύπου.

```
val rdd1 = sc.parallelize(1 to 9)
val rdd2  = sc.parallelize(5 to 15)
rdd1.intersection(rdd2).collect
```

- **Distinct():** επιστρέφει ένα νέο RDD με τα μοναδικά στοιχεία του RDD.

```
val r=sc.parallelize (List(1,1,2,2,2,3,4,5,5))
val s=r.distinct
s.collect
```

Πηγή: https://data-flair.training/blogs/apache-spark-map-vs-flatmap

RDD Operations: Transformations

- **map()**: εφαρμόζει μία λειτουργία σε κάθε στοιχείο του RDD

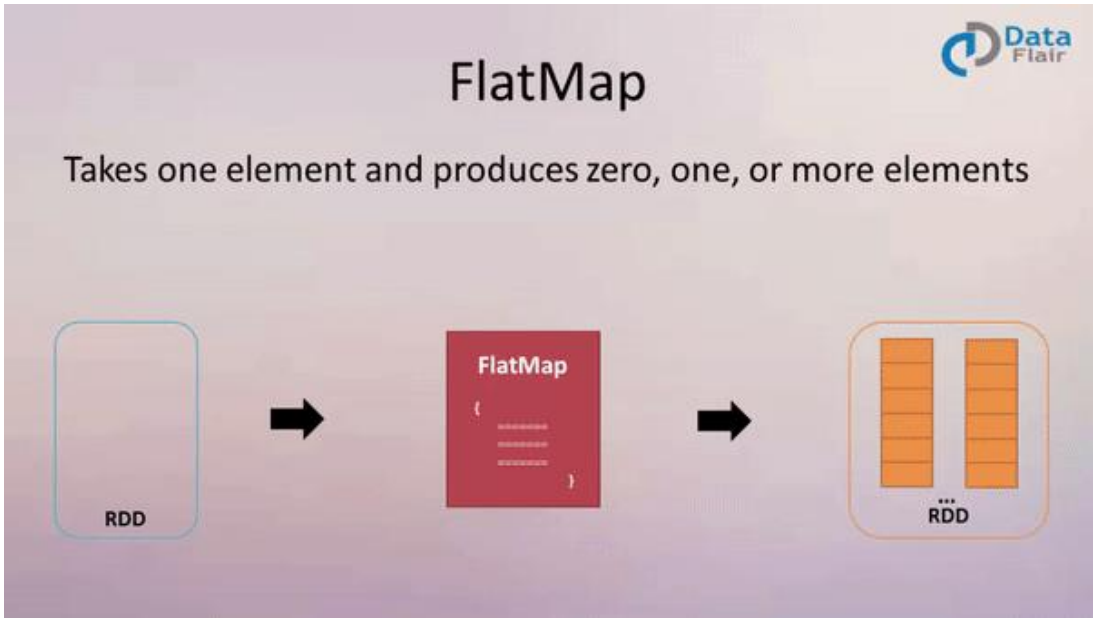
```
val x=sc.parallelize (List("spark", "rdd", "example", "map", "function", "example") )
val y = x.map(x => (x,1))
```

Πηγή: https://data-flair.training/blogs/apache-spark-map-vs-flatmap

RDD Operations: Transformations

- **Flatmap()**: εφαρμόζει μία λειτουργία σε κάθε στοιχείο ενός RDD και επιστρέφει 0, 1 ή περισσότερα στοιχεία.

```
val input =  sc.textFile("/home/lab/data.txt)
val words = input.flatMap(line => line.split(" "))
```



Πηγή: https://data-flair.training/blogs/apache-spark-map-vs-flatmap

RDD Operations: Actions

- Επιστρέφουν ένα αποτέλεσμα κατόπιν εκτέλεσης ενός υπολογισμού στα στοιχεία ενός RDD.

- **Reduce():** δέχεται ένα σύνολο στοιχείων από το RDD και παράγει στην έξοδο ένα συγκεντρωτικό αποτέλεσμα του ιδίου τύπου.

```
val a = sc.parallelize (1 to 10)
a.reduce (_ + _)

val names= sc.parallelize(List ("George", "Mary", "Peter") )
names.reduce (_ + _)
names.reduce((n1,n2) => n1+ n2)
```

- **Distinct():** επιστρέφει ένα νέο RDD με τα μοναδικά στοιχεία του RDD.

```
val r=sc.parallelize (List(1,1,2,2,2,3,4,5,5))
val s=r.distinct()
s.collect()
```

Spark Architecture

Worker Node

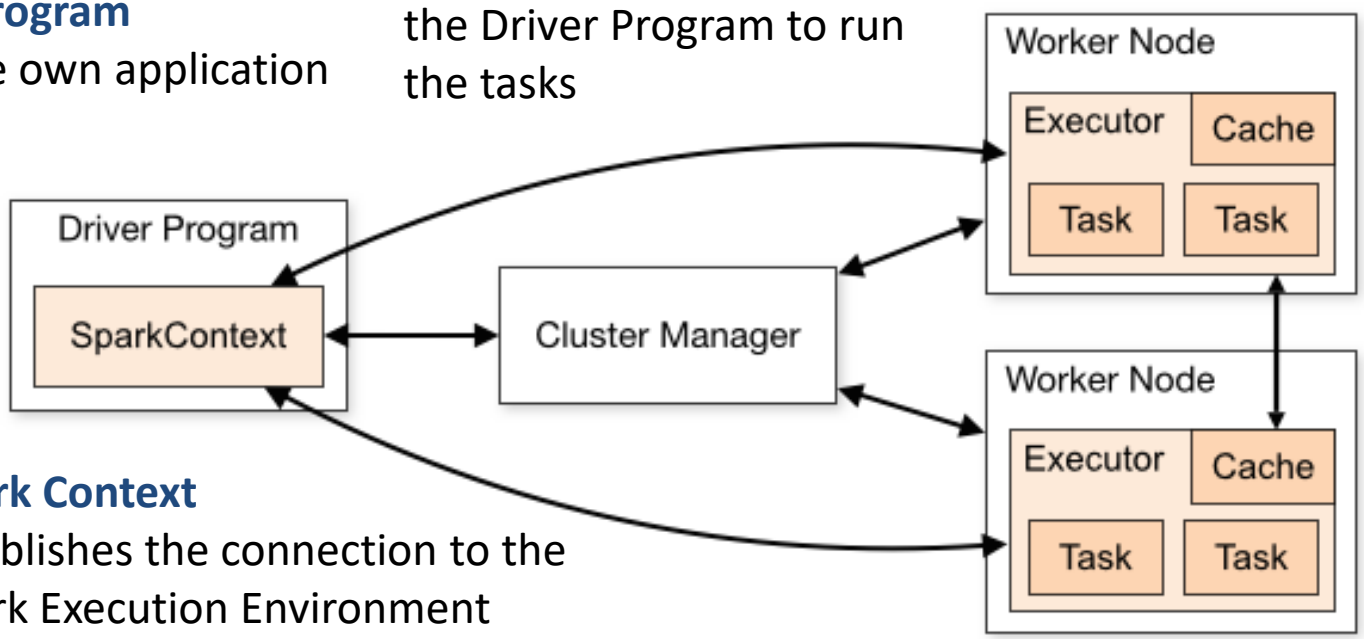
1. Consist of Executors and tasks .
2. Executes the tasks assigned by the Cluster Manager.

Cluster Manager

Allocates the resources to the Driver Program to run the tasks

Driver Program

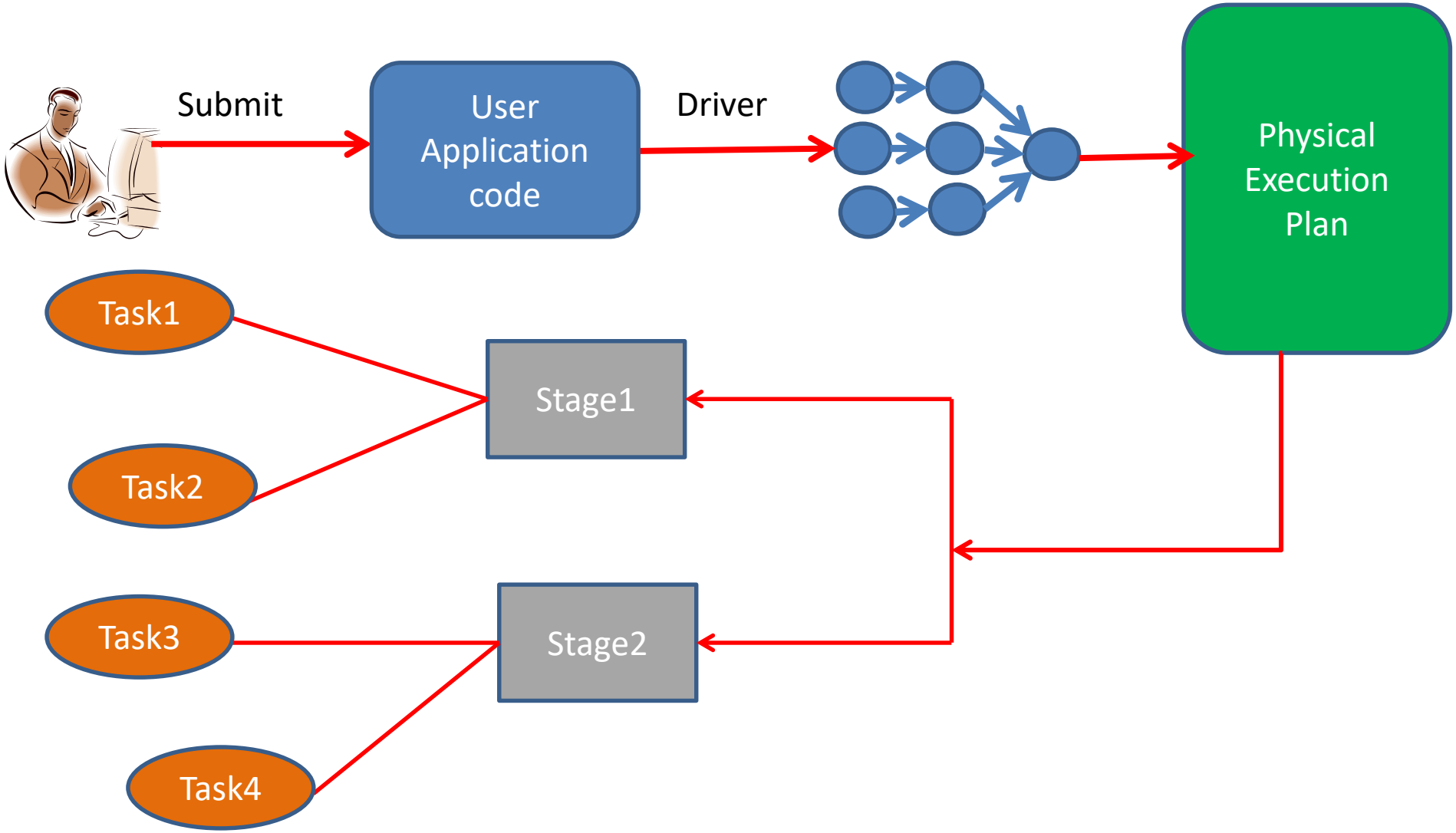
Drive the own application



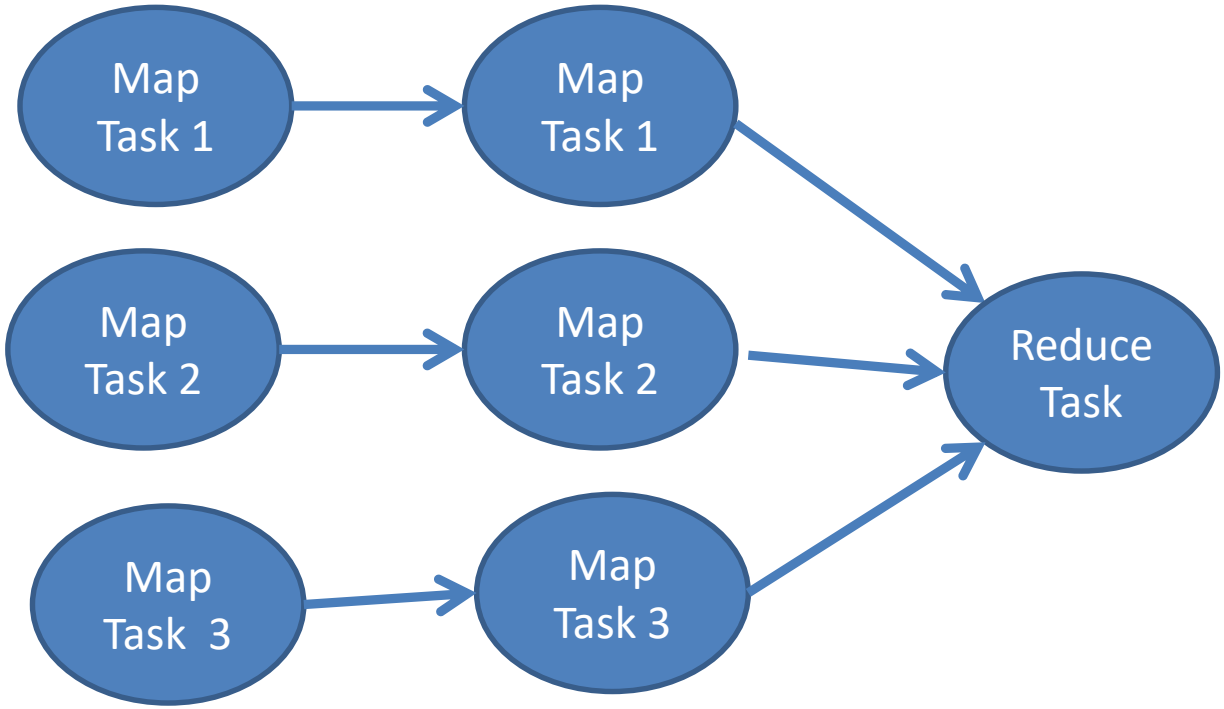
Spark Context

Establishes the connection to the Spark Execution Environment

Directed Acyclic Graph DAG in Apache Spark

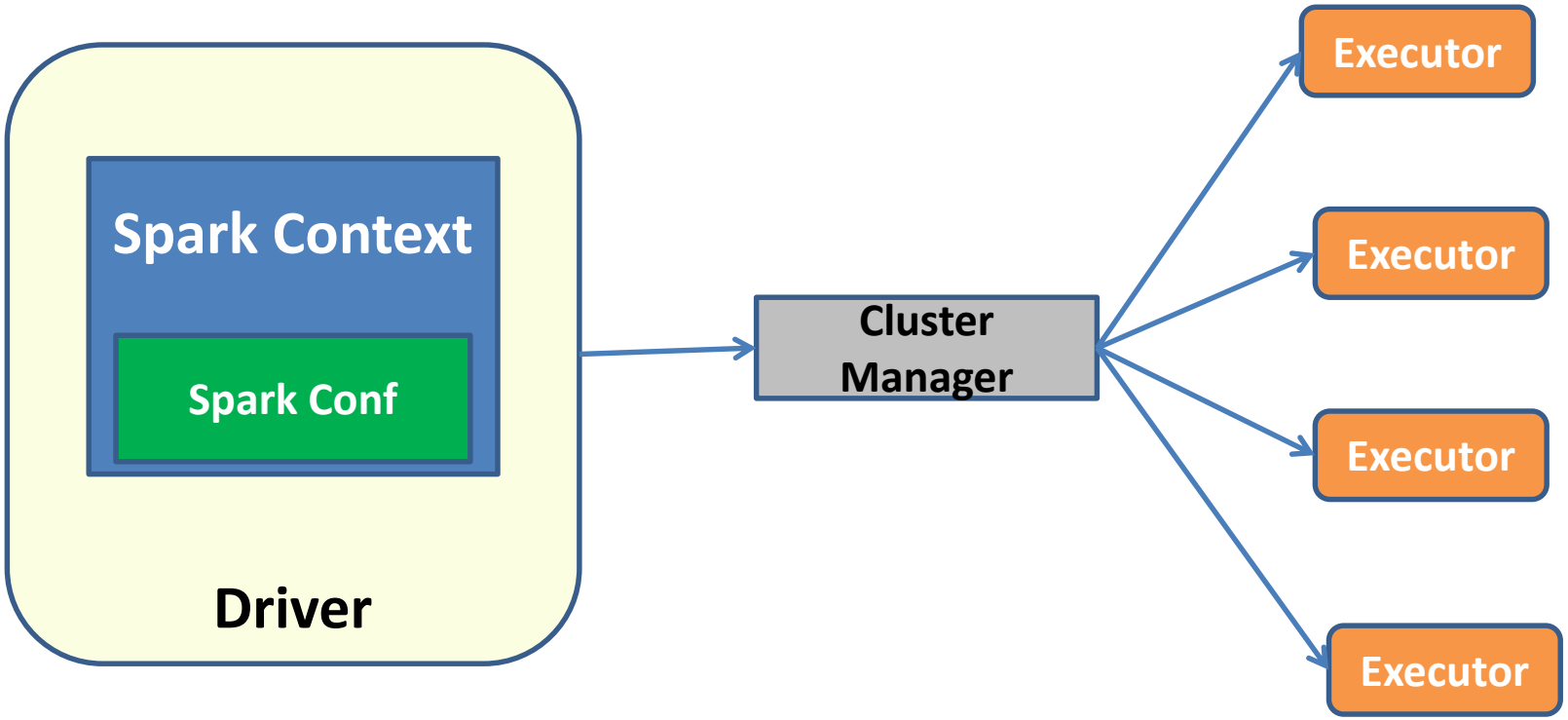


Directed Acyclic Graph DAG in Apache Spark



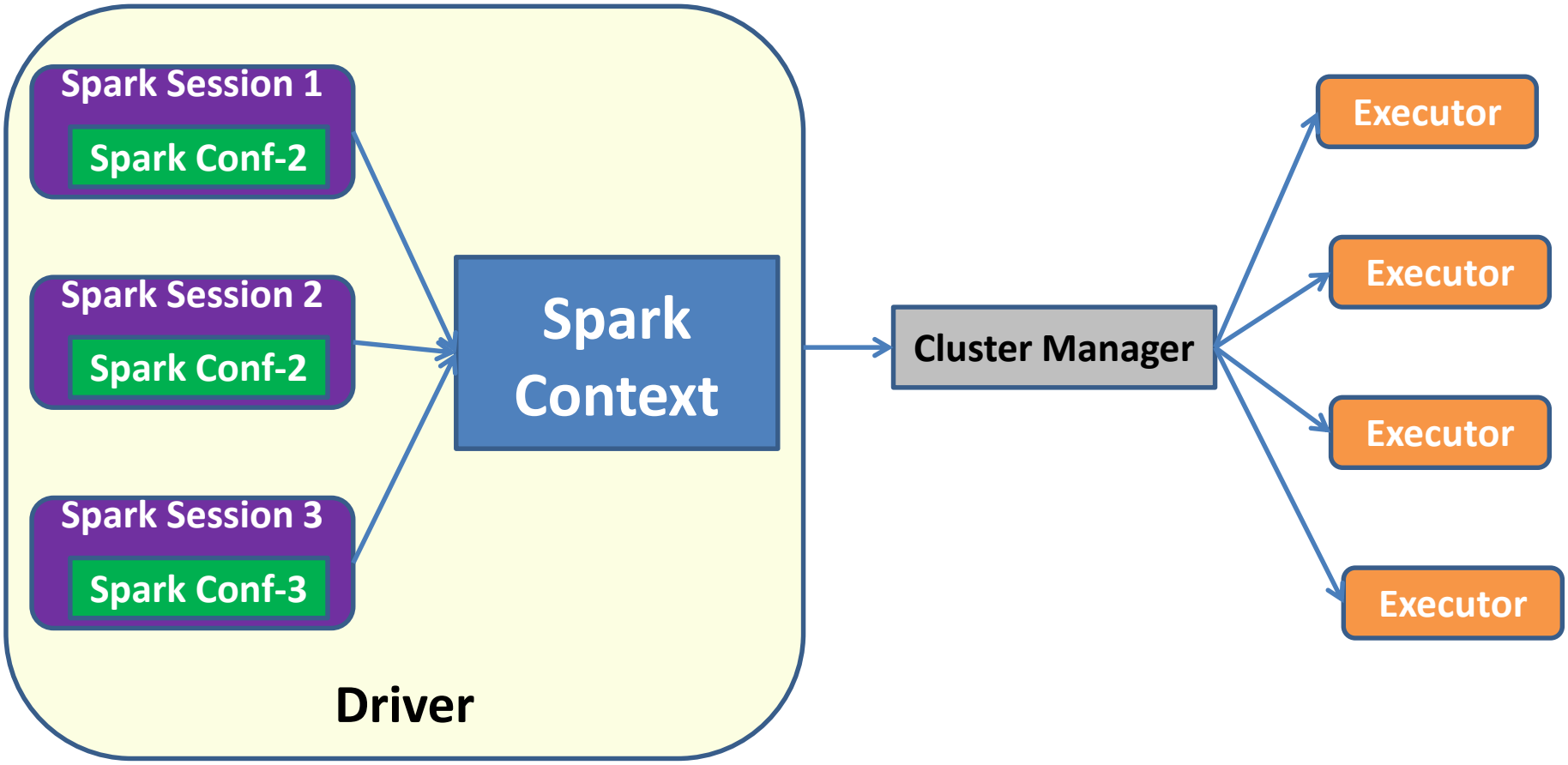
- ✓Nodes represents executable tasks
- ✓Edges are task dependencies

SparkContext




```
val conf = new SparkConf().setAppName("wordCount").setMaster("local[1]")
val sparkContext = new SparkContext(conf)
```

SparkContext vs SparkSession




```
val spark = SparkSession.builder()
                .master("local[1]")
                .appName("wordCount")
               .getOrCreate();
```

Word Count Example

```
/* wordCount.scala */
import org.apache.spark._
import org.apache.spark.SparkContext._

object wordCount {
   def main(args: Array[String]) {
     val inputFile = args(0)
     val outputFile = args(1)
     val conf = new SparkConf().setAppName("wordCount")
     // Create a Scala Spark Context.
     val sc = new SparkContext(conf)
     // Load our input data.
     //val input =  sc.textFile(inputFile)
     val input =  sc.textFile("file:///home/lab/myprj/wordCount/"+inputFile)
     // Split up into words.
     val words = input.flatMap(line => line.split(" "))
     // Transform into word and count.
     val counts = words.map(word => (word, 1)).reduceByKey ( (x, y) => x + y)
     // Save the word count back out to a text file, causing evaluation.
     counts.saveAsTextFile("file:///home/lab/myprj/wordCount/"+outputFile)
   }
}
```

Resilient Distributed Datasets (RDD)

- Θεμελιώδη δομή δεδομένων του SPARK
- Κύρια προγραμματιστική διεπαφή (API).
- Είναι μία αμετάβλητη συλλογή στοιχείων των δεδομένων εισόδου.
- Κάθε RDD χωρίζεται σε λογικά τμήματα (partitions) τα οποία κατανέμονται στους κόμβους ενός cluster και μπορούν να επεξεργαστούν παράλληλα .

DataFrame

- Ένα DataFrame είναι μια αμετάβλητη συλλογή δεδομένων όπως και ένα RDD.
- Τα δεδομένα ενός DataFrame οργανώνονται σε στήλες κάθε μία εκ των οποίων φέρει ένα όνομα όπως ένας πίνακας μιας σχεσιακής βάσης δεδομένων.
- Ένα DataFrame παρέχει μια προγραμματιστική επαφή API για την επεξεργασία των κατανεμημένων δεδομένων.
- Επιτρέπει στους προγραμματιστές να επιβάλουν μια δομή σε μία κατανεμημένη συλλογή δεδομένων (higher level abstraction).
- Η εκτέλεση επερωτήσεων σε ένα DataFrame βελτιστοποιείται αυτόματα από τον query optimizer του spark ο οποίος καλείται Catalyst.

DataSet

- Type Safe
- Καλύτερη απόδοση
- Καλύτερη διαχείριση μνήμης

1. Δημιουργία δοκιμαστικών δεδομένων

```
case class Employ(name: String, age: Int, id: Int, department: String)
val empData = Seq(Employ("A", 24, 132, "HR"), Employ("B", 26, 131, "Engineering"), Employ("C", 25, 135, "Data Science"))
```

2. Δημιουργία DataFrame και DataSet

```
val empRDD = spark.sparkContext.makeRDD(empData)
val empDataFrame = empRDD.toDF()
val empDataset = empRDD.toDS()
```

3. DataSet

```
val empDatasetResult = empDataset.filter(employ => employ.age > 24)
```

4. DataFrame

```
val empDatasetResult = empDataframe.filter(employ => employ.age > 24)
```

******ERROR******

```
val empDataFrameResult = empDataFrame.filter(employ => employ.getAs[Int]("age") > 24)
```

DataFrame vs DataSet

	SQL	DataFrames	DataSets
Syntax Errors	Runtime	Compile Time	Compile Time
Analysis Errors	Runtime	runtime	Compile Time

Only Dataframes for Python and R.