cetic

Your Connection to ICT Research

Machine Learning With Spark

Ons Dridi
R&D Engineer

13 Novembre 2015

FEDER
UNION EUROPEENNE
Wallonie
LE FONDS EUROPEEN DE DEVELOPPEMENT REGIONAL
ET LA WALLONIE INVESTISSENT DANS VOTRE AVENIR.

Centre d'Excellence en Technologies de l'Information et de la Communication

Your Connection to ICT Research

CETIC Presentation

- An applied research centre in the field of ICT

- The knowledge developed by CETIC is made available to companies to help them integrate these technological breakthroughs into their products, processes and services, enabling them to innovate faster, save time and money and develop new markets.

- Three departments:
  Software & System Engineering
  Software & Services Technologies
  Embedded & Communication Systems

Who am I ?

Background:

- Engineering computer science degree
- Master degree in Computer Science and Mathematics applied to Finance and Insurance

Mission:

- Big Data: data analysis

https://www.cetic.be/Ons-Dridi

Plan

- What is Spark ?

  High level Architecture

  How does it Work ?

  RDD and Operations

  Hadoop MapReduce

  DAG (Directed Acyclic Graph)

  Run Mode of Spark

  Programming model

- Machine Learning With Spark

  MLLib Library

  Types of Machine Learning

  ML Architecture

  Comparison with other tools

- Other Applications

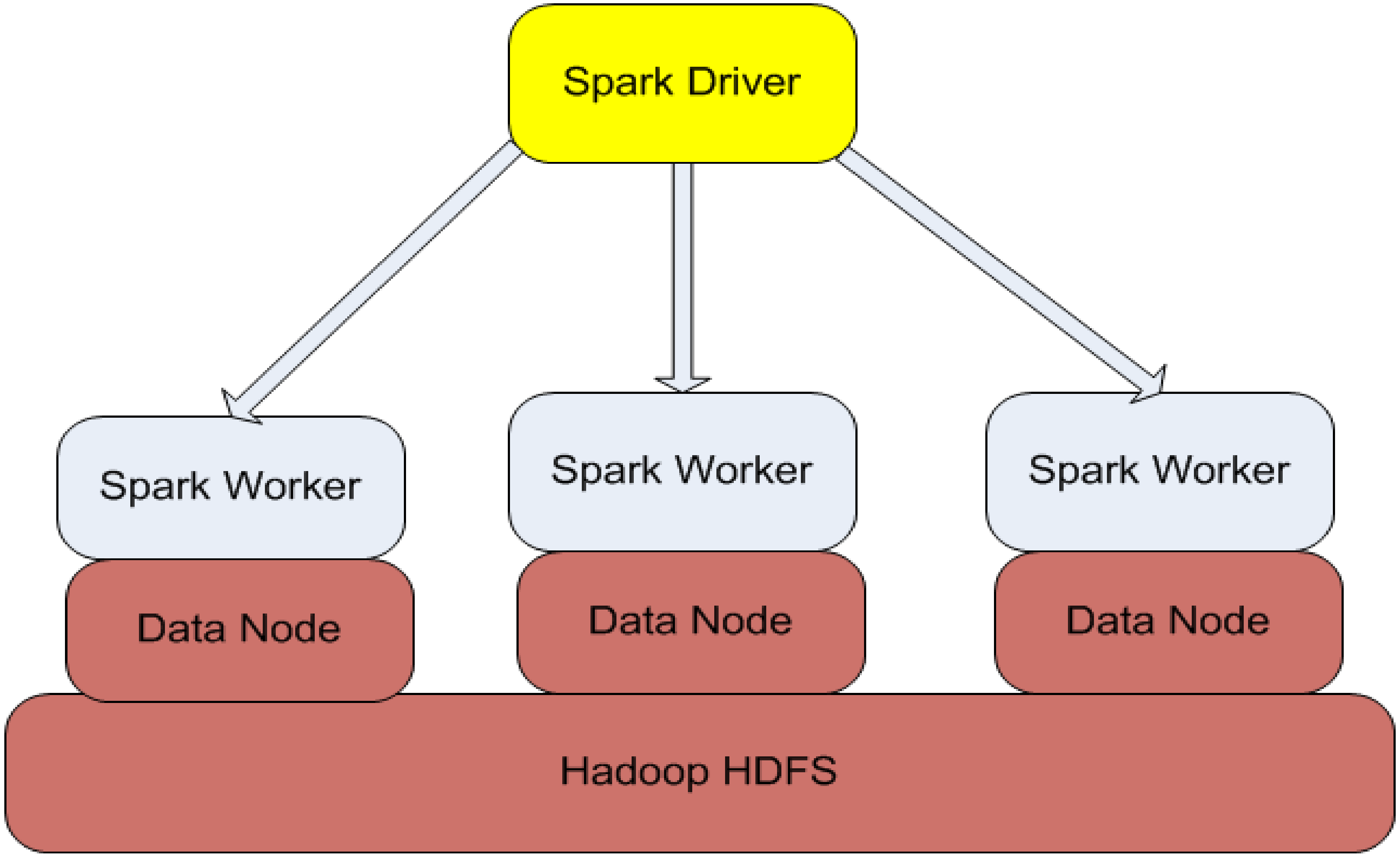
What is Spark ?

- *Definition*

"Apache Spark is an open source big data processing framework built around speed, ease of use, and sophisticated analytics. It was originally developed in 2009 in UC Berkeley's AMPLab, and open sourced in 2010 as an Apache project."

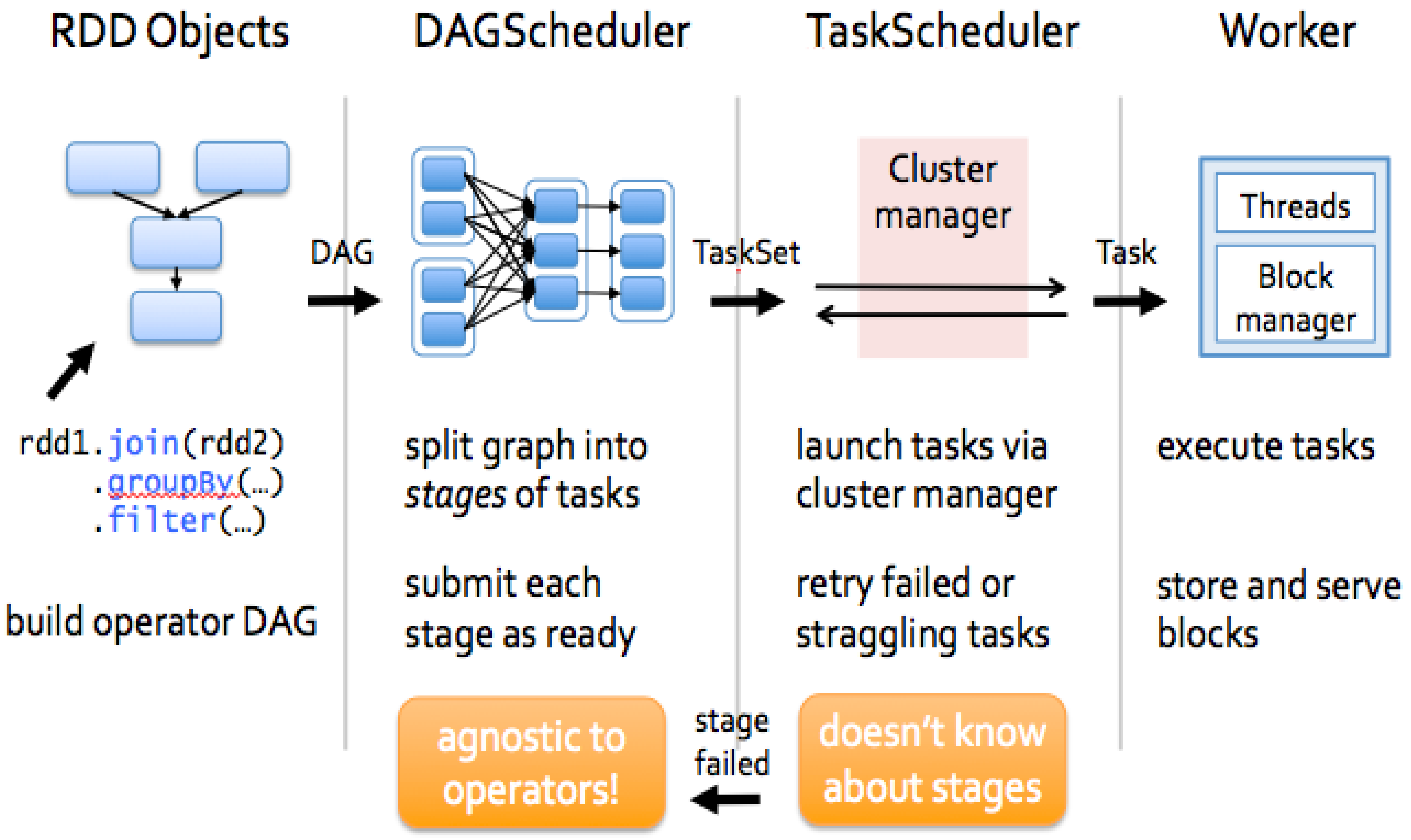
Source : http://www.infoq.com/articles/apache-spark-introduction

High Level Architecture



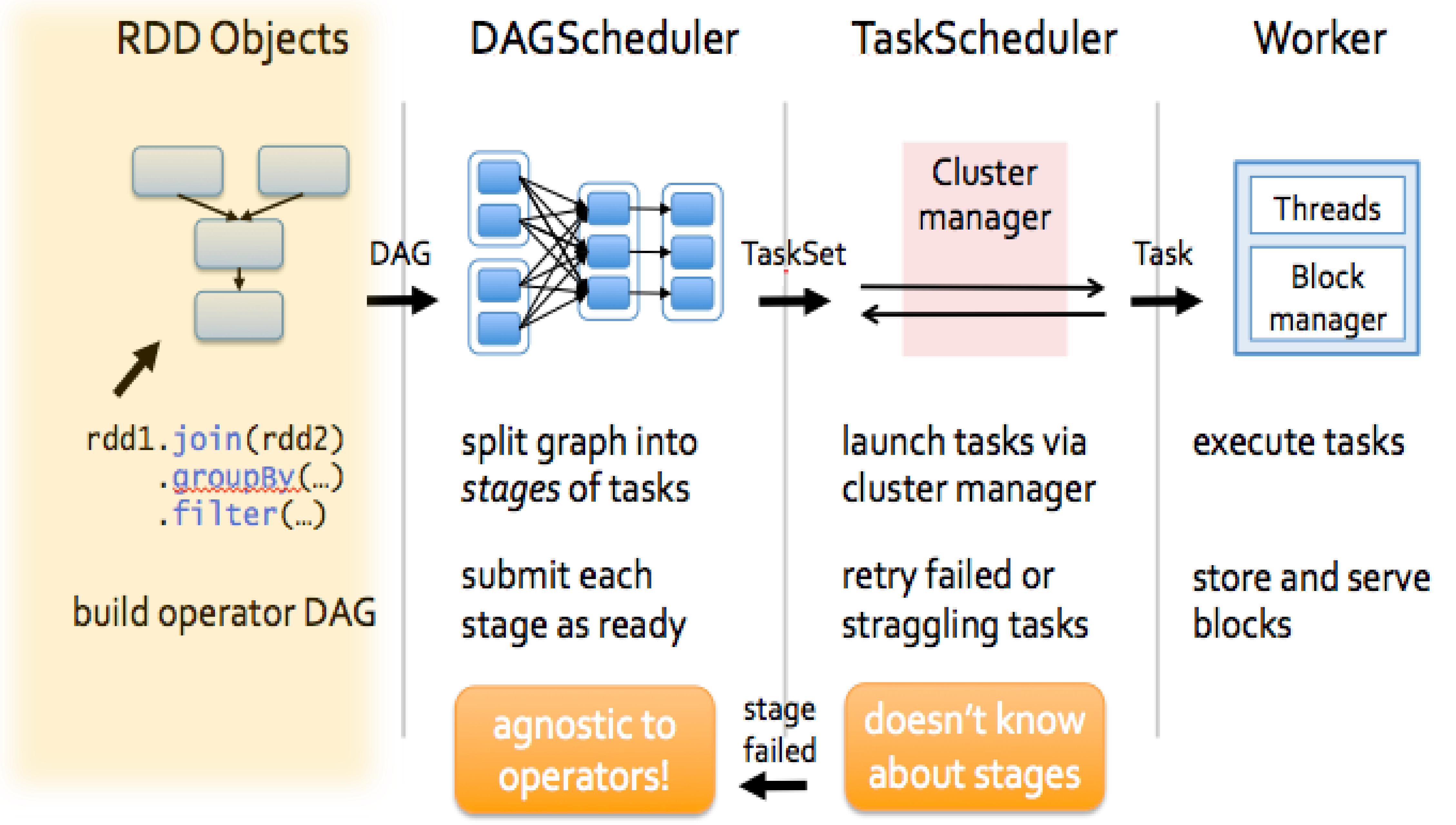
Source : https://bighadoop.wordpress.com/2014/04/03/apache-spark-a-fast-big-data-analytics-engine/

How does it work ?



Source: https://www.sigmoid.com/apache-spark-internals/

RDD and Operations



Source : http://spark.apache.org/docs/0.6.2/api/core/spark/RDD.html

RDD and Operations

- ***Definition***

“A Resilient Distributed Dataset (RDD), the basic abstraction in Spark. Represents an immutable, partitioned collection of elements that can be operated on in parallel. This class contains the basic operations available on all RDD”

Source : http://spark.apache.org/docs/0.6.2/api/core/spark/RDD.html

RDD and Operations

- *Creating RDD:*

  - From existing collection

```
val collection = List ("a", "b", "c", "d")
val rddFromCollection = sc.parallelize (collection)
```

  - From Hadoop-based input sources

```
val rddFromTextFile= sc.textFile (" List ")
```

RDD and Operations

- **Operations:**

  - *Transformations*

    Apply functions to all the records, ex: map, filter, join

```
val sizeFromStringsRDD = rddFromTextFile.map (line => line.size)
```

  - *Actions*

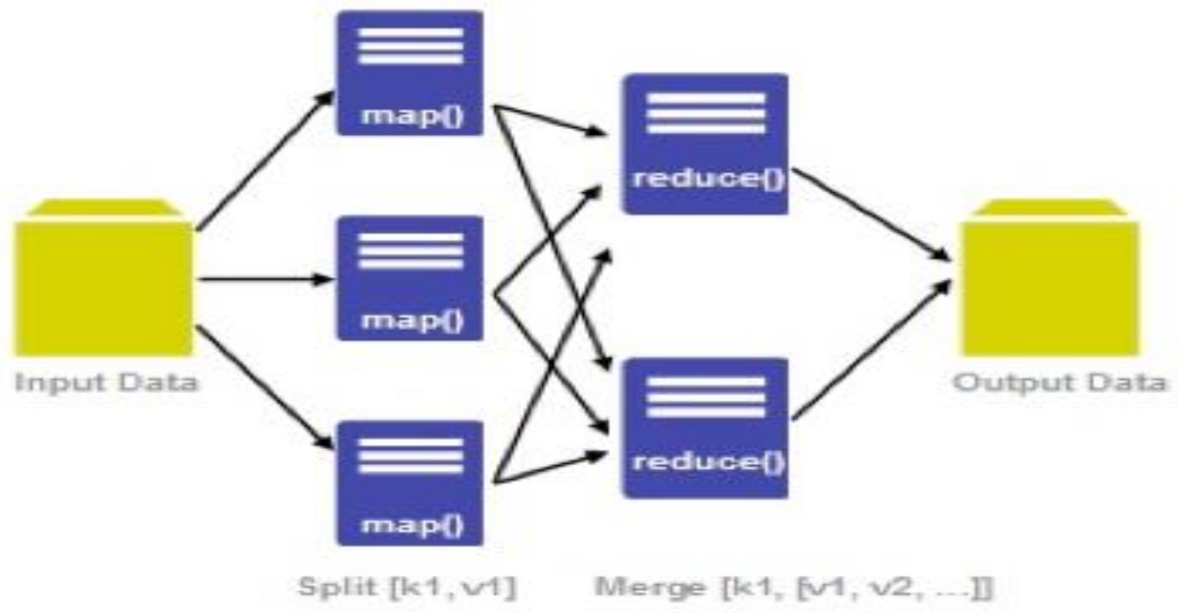
    Apply some computations and return the results to the driver program , ex: reduce, count

```
sizeFromStringsRDD.count
```

Hadoop MapReduce

- ***Definition***

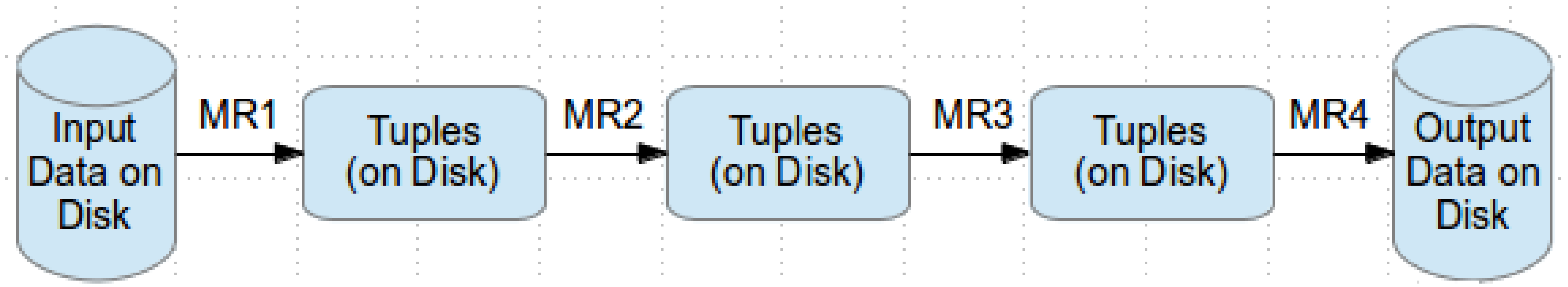
"The term MapReduce actually refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job … and the second is the reduce."



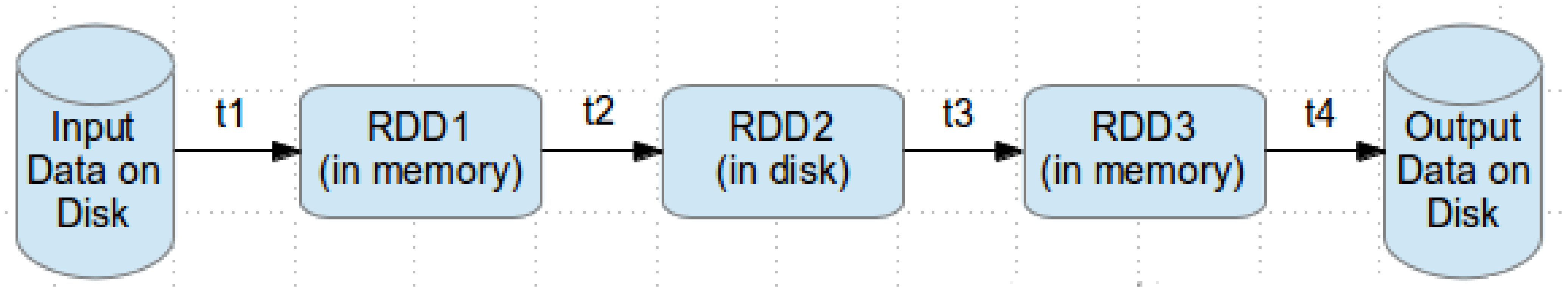
Source: https://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/

RDD and Operations

Case of MapReduce :

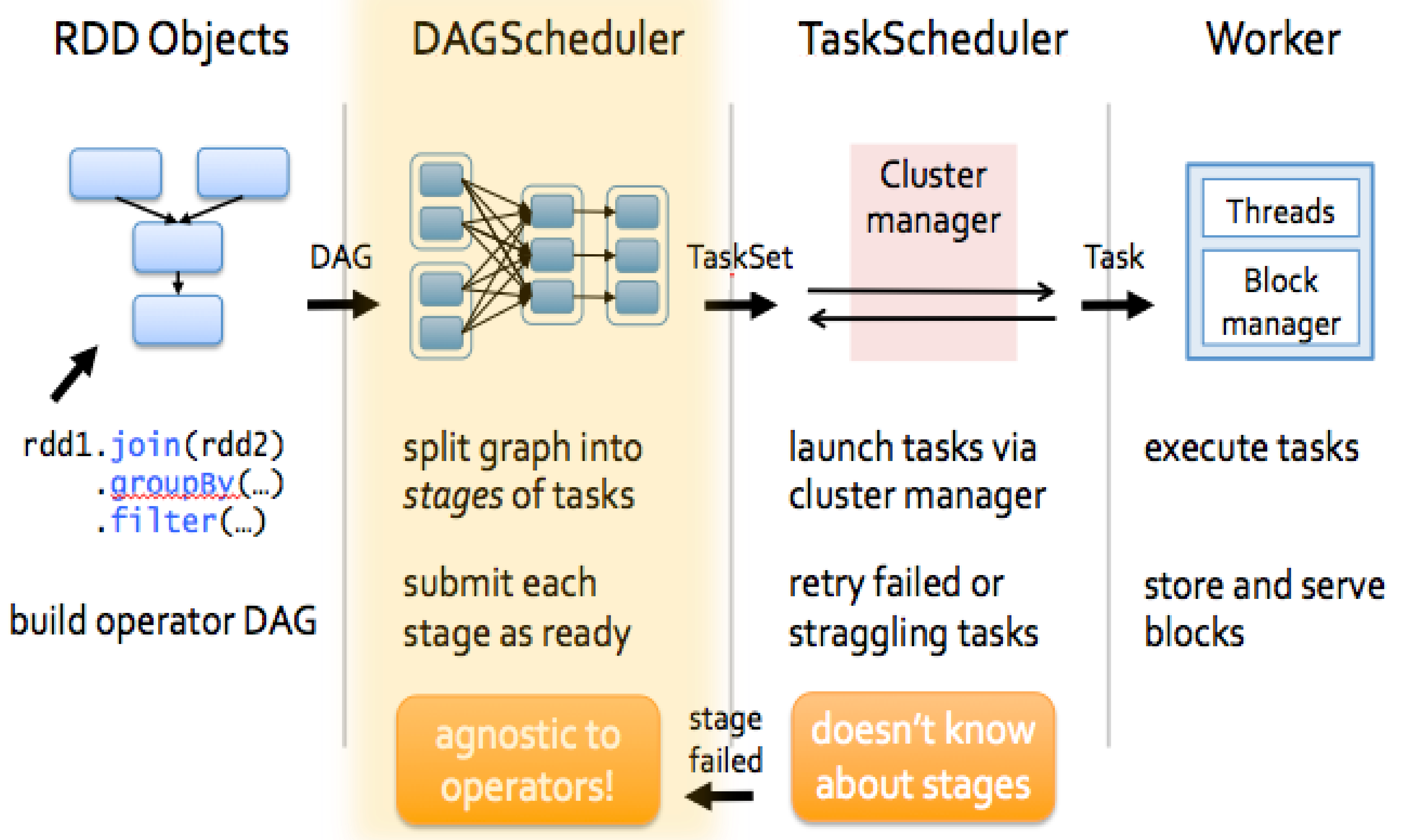


Case of RDD:



Source: http://www.thecloudavenue.com/

DAG (Direct Acyclic Graph)

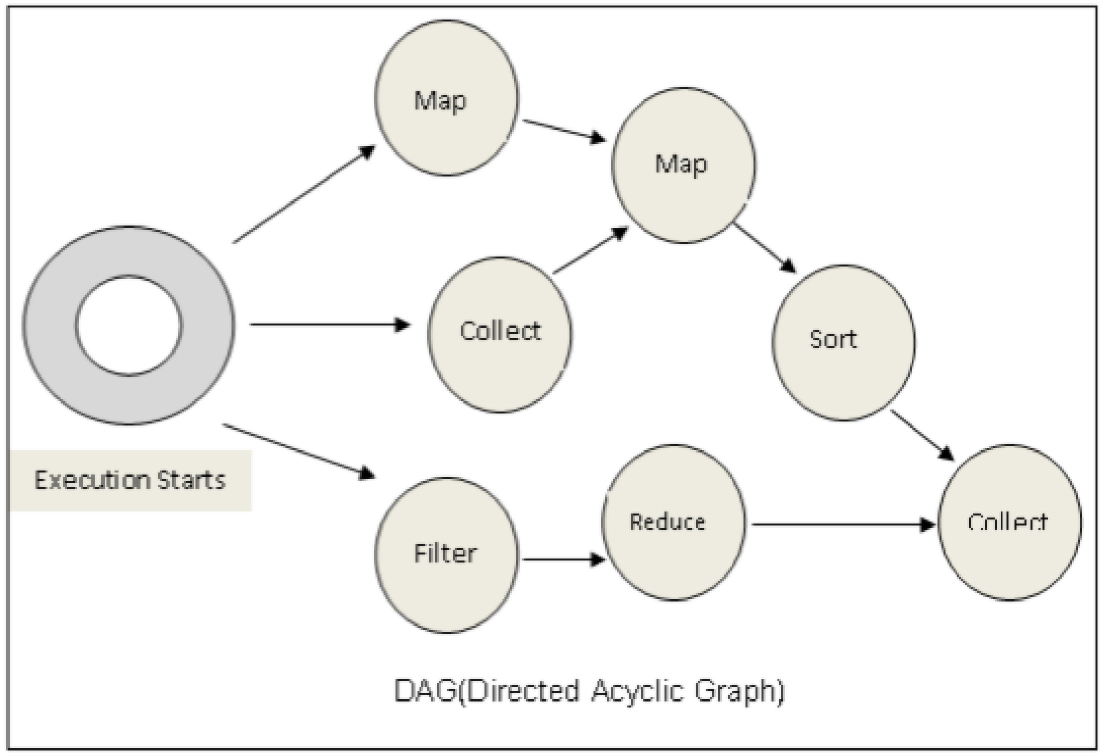


Source : http://spark.apache.org/docs/0.6.2/api/core/spark/RDD.html

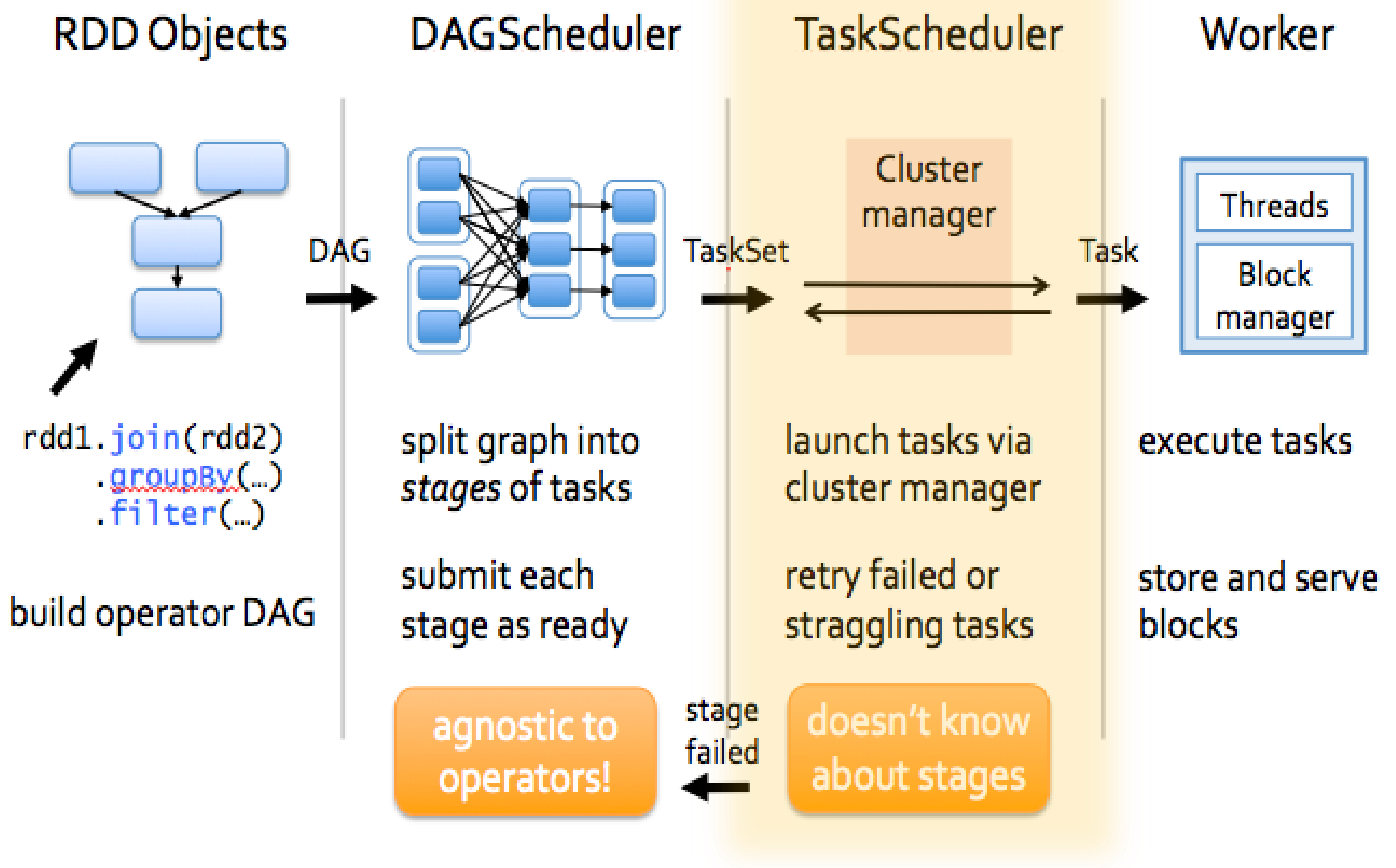
DAG (Direct Acyclic Graph)

- ***Definition :***

"DAG stands for Directed Acyclic Graph, in the present context. It's a DAG of operators. The DAG is optimized by rearranging and combining operators where possible"



Run mode of Spark



Source : http://spark.apache.org/docs/0.6.2/api/core/spark/RDD.html

Run mode of Spark

Spark runs in four modes:

- **Local mode**: run our application locally. This is really useful for debugging, we can step our code line by line with an IDE
- **Cluster Mode:**

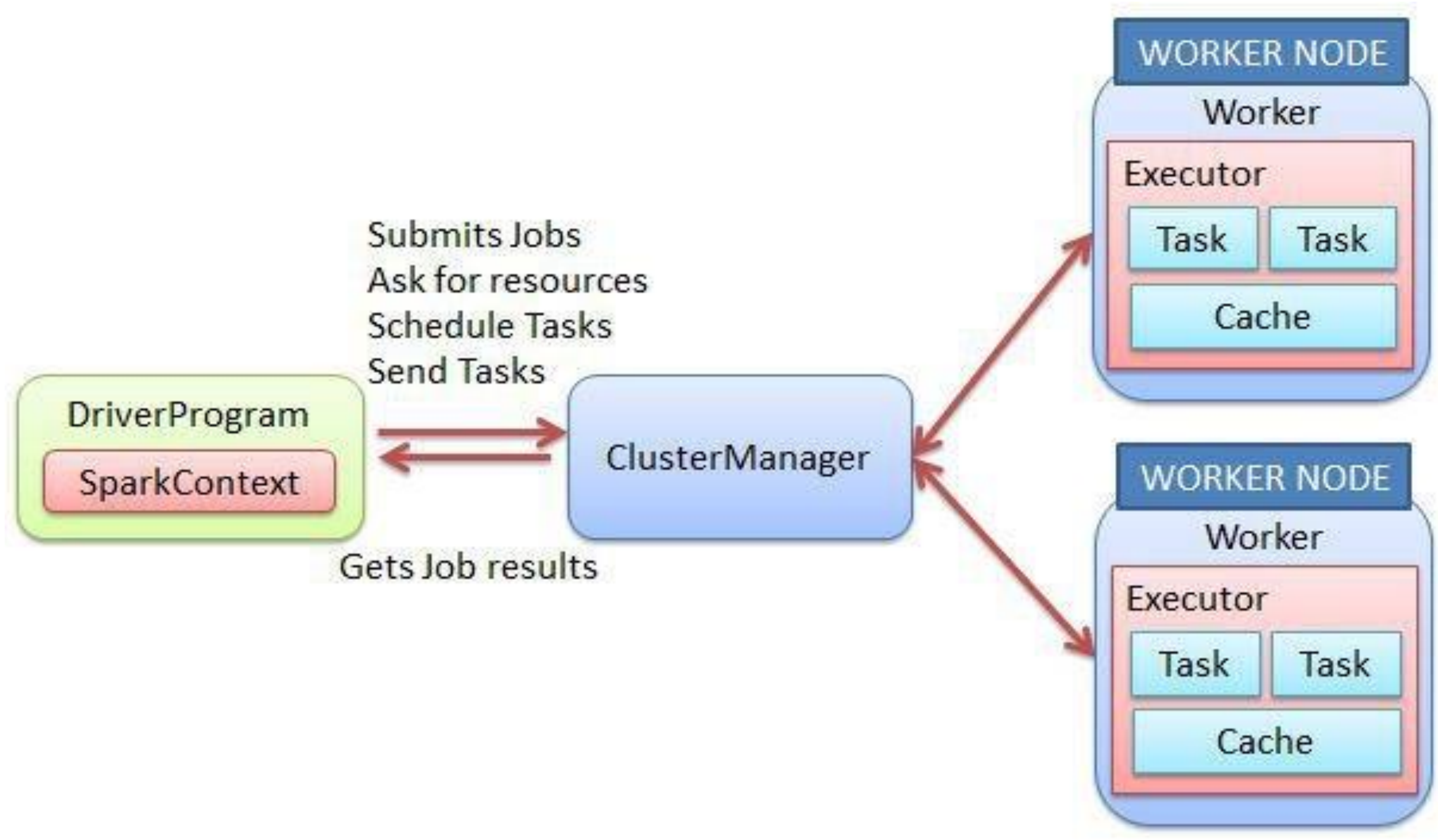
  Standalone mode: we can easily deploy a standalone cluster with very few steps and configurations and then we can play around with it.

  Apache Mesos

  Hadoop Yarn

Run mode of Spark

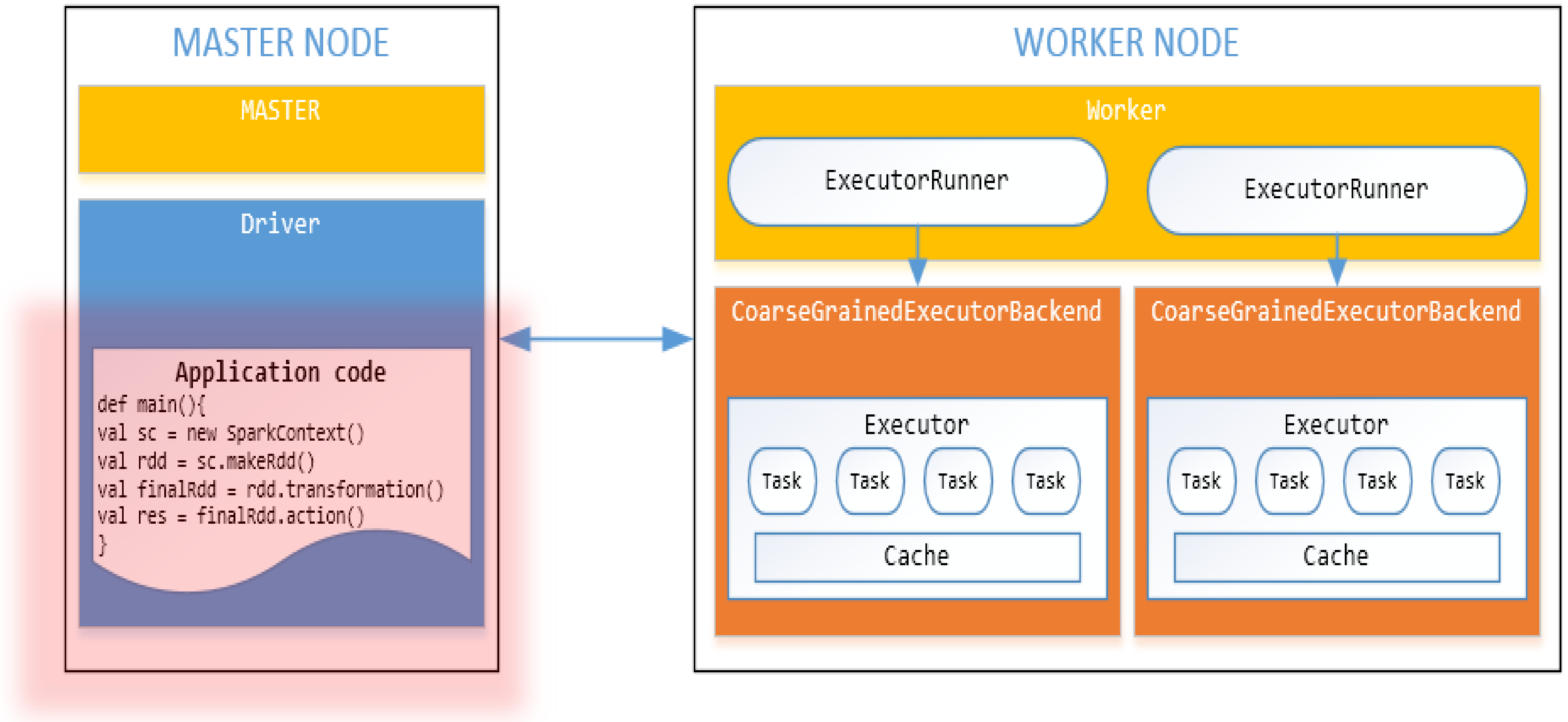
- *Cluster Mode:*



Source: https://trongkhoanguyenblog.wordpress.com

Programming model

- ***Master-Workers:***



```
def main(){
val sc = new SparkContext()
val rdd = sc.makeRdd()
val finalRdd = rdd.transformation()
val res = finalRdd.action()
}
```

Source: https://trongkhoanguyenblog.wordpress.com

Programming model

- *Spark Context and SparkConf:*

  - The starting point of any spark program is *Spark Context*
  - It's initialized with an instance of *SparkConf*
  - Contains various methods to manipulate RDD

- *Initialize SparkContext:*

```
val conf = new SparkConf ()
        .setAppName(" Test Spark ")
                .setMaster(" local[4] " )
val sc= new SparkContext(conf)
```

```
val sc = new SparkContext(" local[4] " , " Test Spark " )
```

Machine Learning With Spark

- ***Definition :***

“Machine learning is a method of data analysis that automates analytical model building. Using algorithms that iteratively learn from data, machine learning allows computers to find hidden insights without being explicitly programmed where to look.“

Source: http://www.sas.com/

Machine Learning With Spark

- *What problems does it solve ?*

  - Marketing
  - Human resources
  - Risk management
  - Health care
  - Travel
  - Education
  - …

Your Connection to ICT Research

Machine Learning With Spark

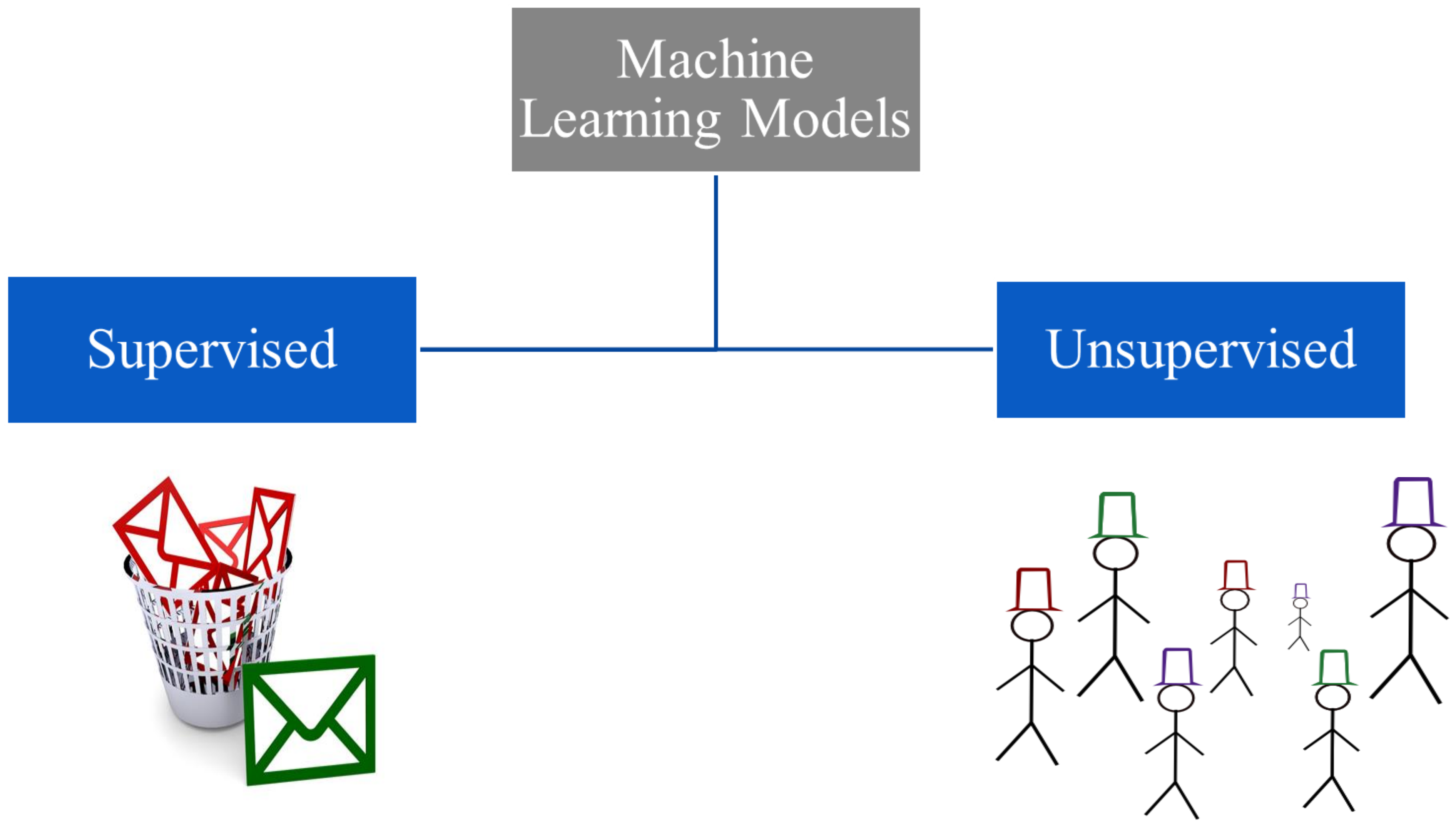
- ***MLLib Library :***

“MLlib is Spark’s scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization Primitives”

Source: https://spark.apache.org

Machine Learning With Spark

- *Types of Machine Learning system:*



Machine Learning With Spark

- ***Supervised models:***

- Build a model that makes predictions
- The correct classes of the training data are known
- We can validate performance
- Two broad categories:

Classification: assign a class to an observation. e.g.: patient will have a heart attack or not.
Regression: predict a continuous measurement for an observation. e.g.: car prices

- Algorithms:
Regression, Decision Tree, Naive Bayes, SVM, …

Machine Learning With Spark

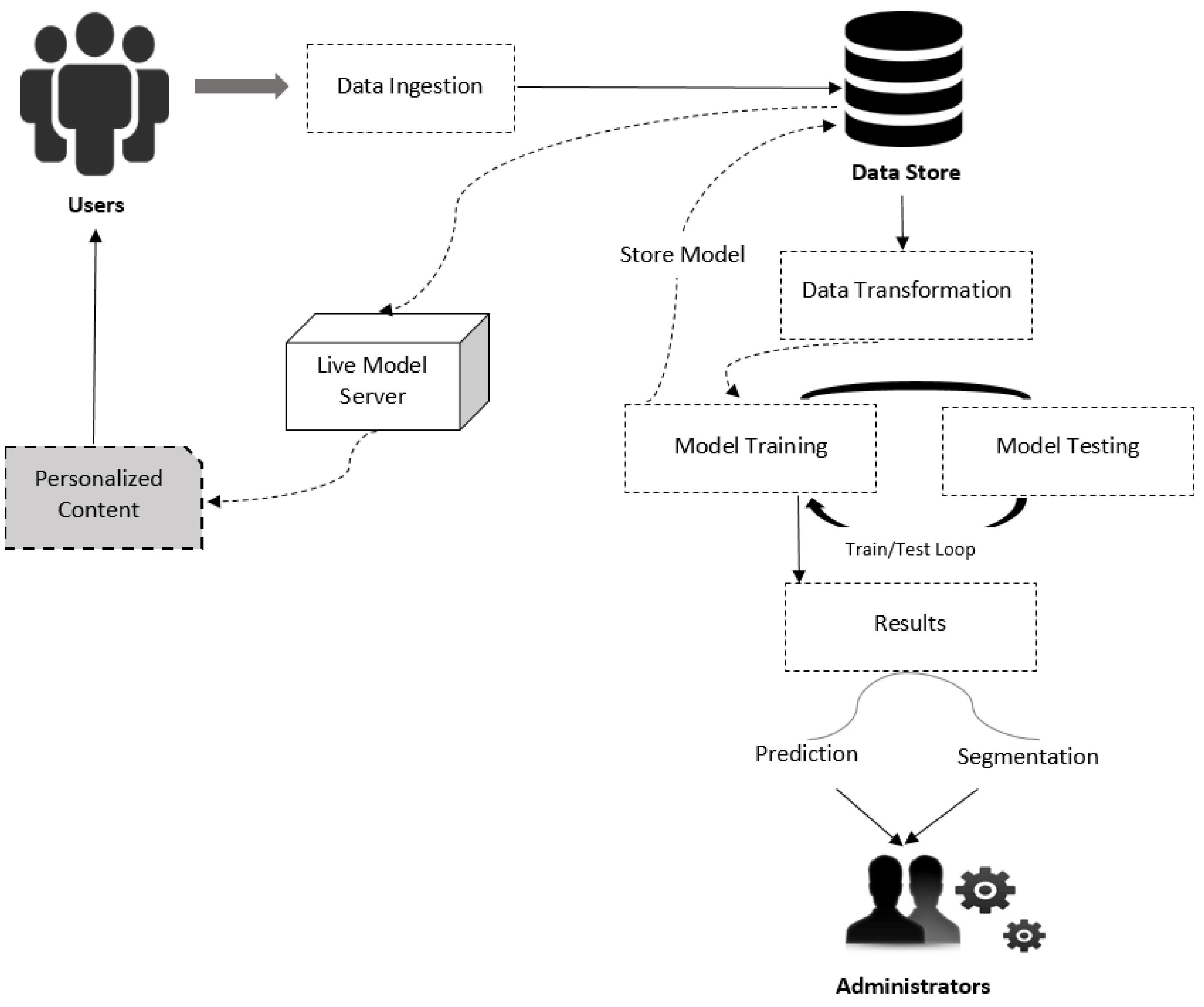
- ***Unsupervised models:***

- Exploratory data analysis to find hidden patterns or grouping in data
- The clusters are modeled using a measure of similarity
- Performance ?

- Algorithms:
    ACP, K-means Clustering , Hierarchical clustering …

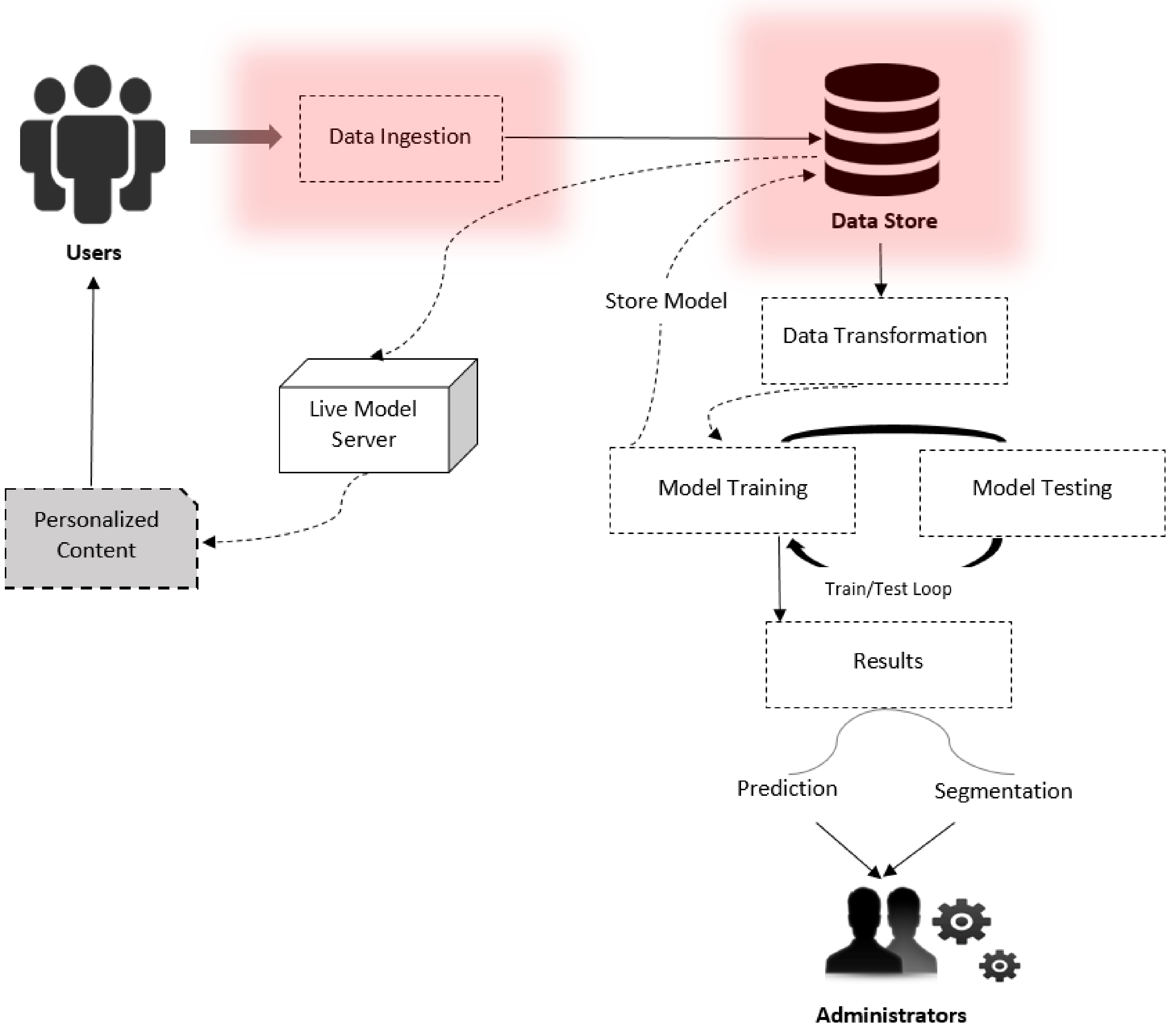
Machine Learning With Spark

- *ML Architecture:*



Machine Learning With Spark

- *Data Ingestion and storage:*



Machine Learning With Spark

- ***Data Ingestion:***

  - Browser, and mobile application event logs or accessing external web APIs

- ***Data Storage:***

  - HDFS, Amazon S3, and other filesystems; SQL databases such as MySQL or PostgreSQL; distributed NoSQL data stores such as HBase, Cassandra, and DynamoDB, …

Machine Learning With Spark

- ***Useful datasets available publicly:***
  - UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/
  - Amazon AWS public datasets: http://aws.amazon. com/publicdatasets/
  - Kaggle: http://www.kaggle.com/competions
  - KDnuggets: http://www.kdnuggets.com/ datasets/index.html

- ***Used dataset:***
  - Training data from the Kaggle: train.tsv
  - Classifying problem : web page is a '*Short-lived*' or '*not*'

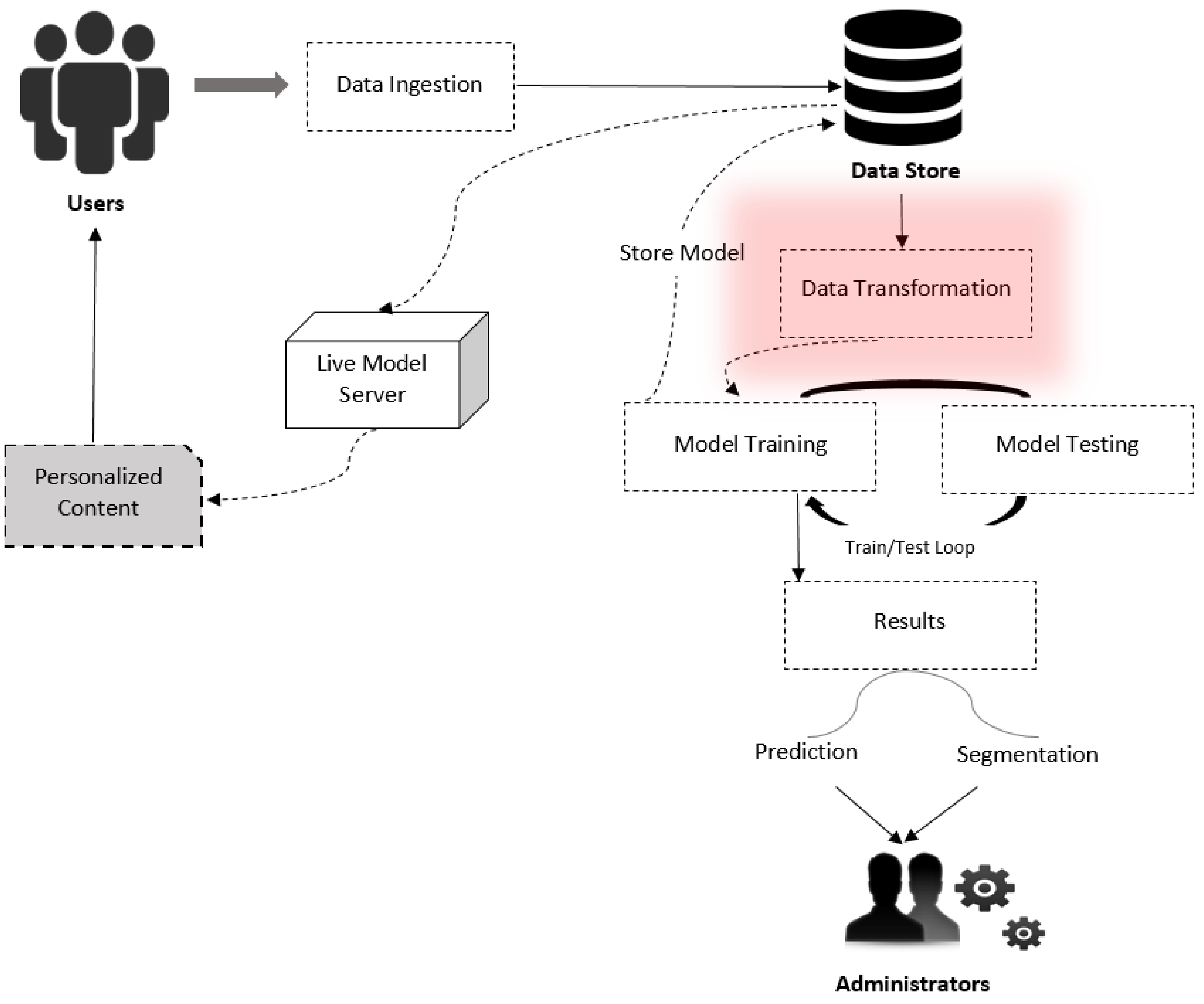
http://www.kaggle.com/c/stumbleupon/data

- Source code on Github

https://github.com/onsDridi/SparkML.git

Machine Learning With Spark

- *Data transformation:*



Machine Learning With Spark

- *Data transformation:*

  - Filter out or remove records with bad or missing values
  - Fill in bad or missing data
  - Apply robust techniques to outliers
  - Apply transformations to potential outliers
  - Extract useful features

Machine Learning With Spark

- *Data transformation:*

  - Remove the first line

```
sed 1d train.tsv > train_noheader.tsv
```

  - Run Spark

```
>./bin/spark-shell --driver-memory 4g
```

  - Create RDD

```
val rawData = sc.textFile("data/train_noheader.tsv")
val records = rawData.map(line => line.split("\t"))
records.first()
```

```
Array[String] = Array("http://www.bloomberg.com/news/2010-12-23/ibm-
predicts-holographic-calls-air-breathing-batteries-by-2015.html", "4042",
...
```

Your Connection to ICT Research

Machine Learning With Spark

- ***Data transformation:***

  - Apply some data cleaning

```
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.linalg.Vectors
    val data = records.map { r =>val trimmed =  r.map(_.replaceAll("\"", ""))
    val label = trimmed(r.size - 1).toInt
    val features = trimmed.slice(4, r.size - 1).map(d => if (d =="?") 0.0 else
    d.toDouble)
    LabeledPoint(label, Vectors.dense(features))}
```

Machine Learning With Spark

- ## *Data transformation:*

- Cache the data

```
data.cache
val  numData = data.count
```

The value is: **7395**

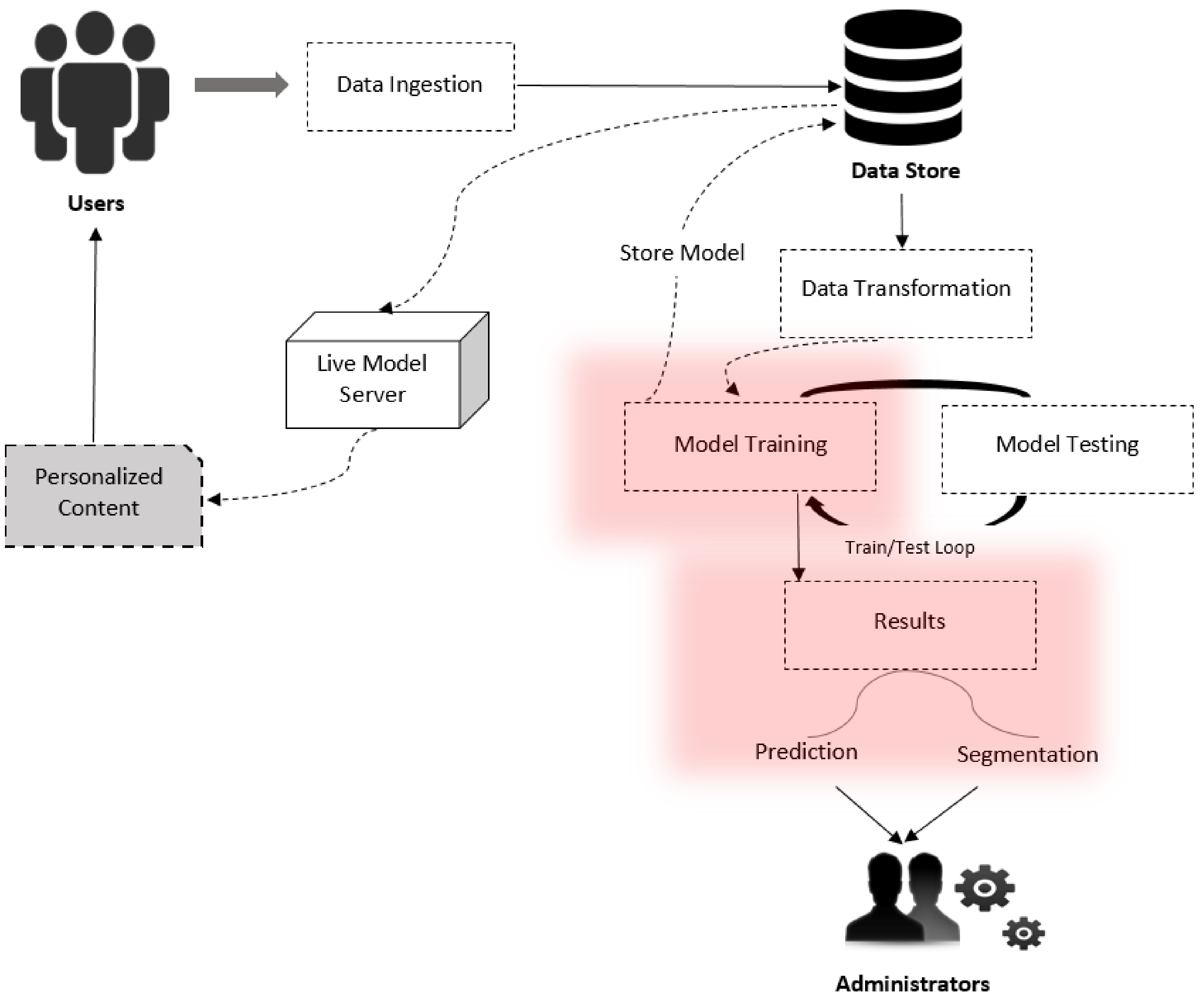
- Replace the negative values by zero

```
val nbData = records.map { r =>
    val trimmed = r.map(_.replaceAll("\"", ""))
    val label = trimmed(r.size - 1).toInt
    val features = trimmed.slice(4, r.size - 1).map(d => if (d == "?")
0.0 else d.toDouble).map(d => if (d < 0) 0.0 else d)
    LabeledPoint(label, Vectors.dense(features))
    }
```

Your Connection to ICT Research

Machine Learning With Spark

- ## *Training Classification Model:*



Your Connection to ICT Research

Machine Learning With Spark

- ***Training Classification model:***

  - Set up input parameters

```
import org.apache.spark.mllib.classification.LogisticRegressionWithSGD
import org.apache.spark.mllib.classification.SVMWithSGD
import org.apache.spark.mllib.classification.NaiveBayes
import org.apache.spark.mllib.tree.DecisionTree
import org.apache.spark.mllib.tree.configuration.Algo
import org.apache.spark.mllib.tree.impurity.Entropy
val numIterations = 10
val maxTreeDepth = 5
```

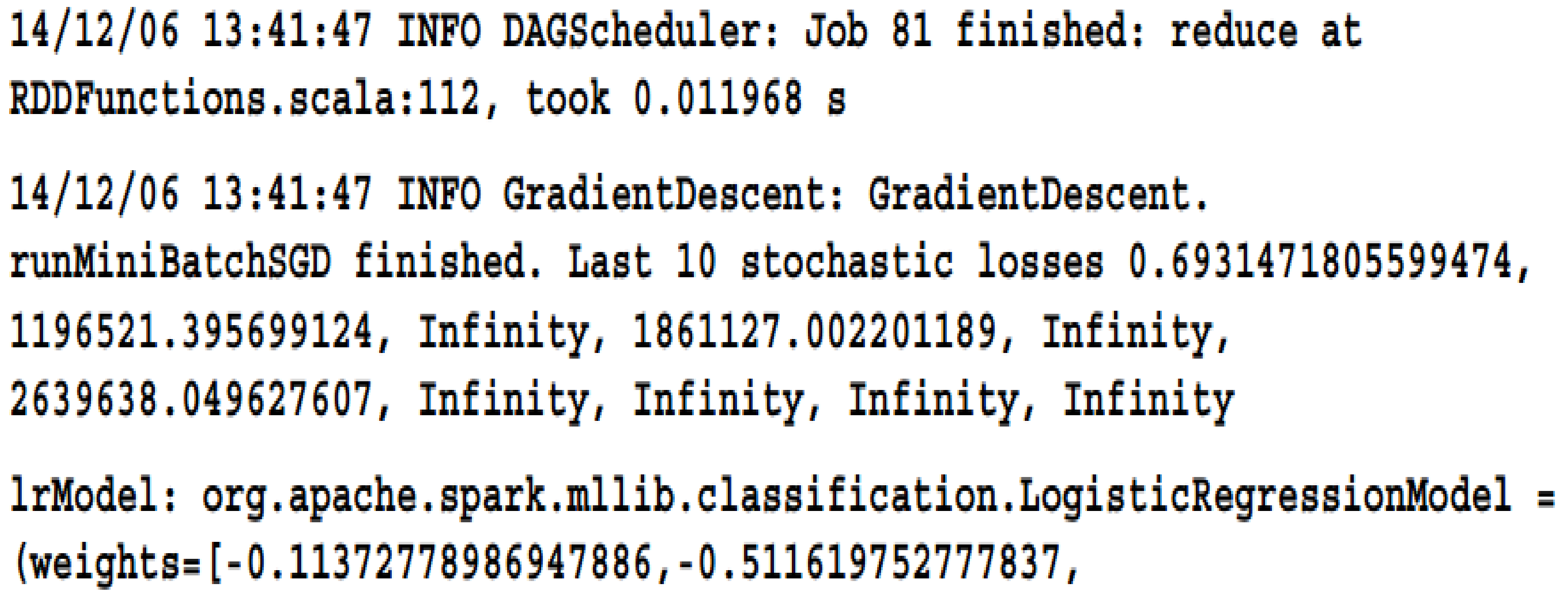
Your Connection to ICT Research

Machine Learning With Spark

- ***Training Classification model:***

  - Train logistic regression

```
val lrModel = LogisticRegressionWithSGD.train(data, numIterations)
```



```
14/12/06 13:41:47 INFO DAGScheduler: Job 81 finished: reduce at
RDDFunctions.scala:112, took 0.011968 s

14/12/06 13:41:47 INFO GradientDescent: GradientDescent.
runMiniBatchSGD finished. Last 10 stochastic losses 0.6931471805599474,
1196521.395699124, Infinity, 1861127.002201189, Infinity,
2639638.049627607, Infinity, Infinity, Infinity, Infinity

lrModel: org.apache.spark.mllib.classification.LogisticRegressionModel =
(weights=[-0.11372778986947886,-0.511619752777837,
```

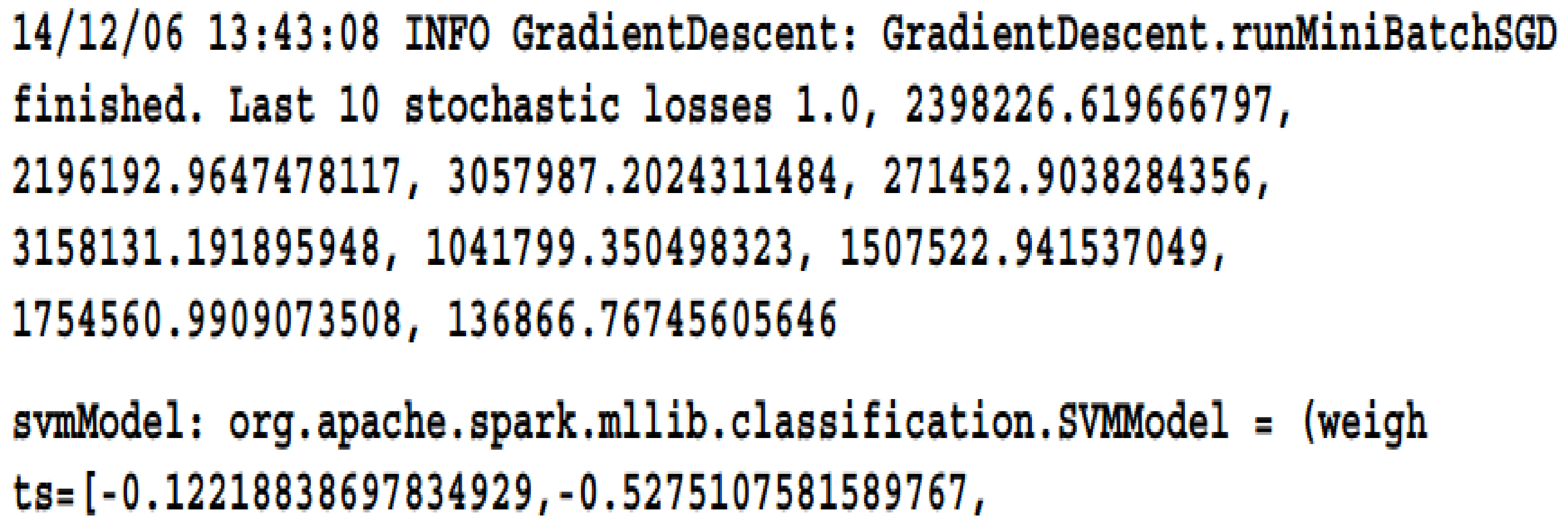
Your Connection to ICT Research

Machine Learning With Spark

- ## *Training Classification model:*

  - Train an SVM model

```
val svmModel = SVMWithSGD.train(data, numIterations)
```



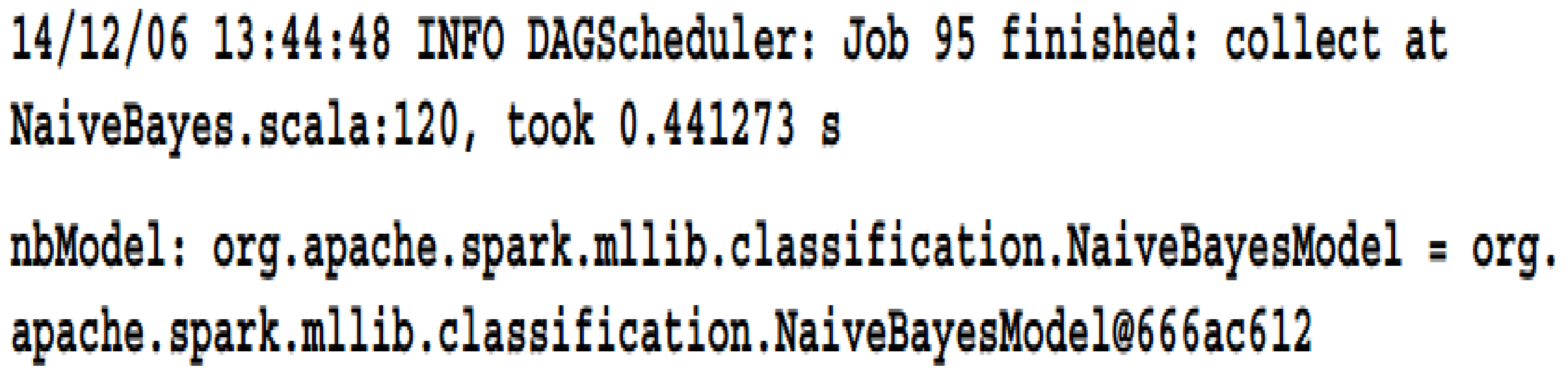
```
14/12/06 13:43:08 INFO GradientDescent: GradientDescent.runMiniBatchSGD
finished. Last 10 stochastic losses 1.0, 2398226.619666797,
2196192.9647478117, 3057987.2024311484, 271452.9038284356,
3158131.191895948, 1041799.350498323, 1507522.941537049,
1754560.9909073508, 136866.76745605646

svmModel: org.apache.spark.mllib.classification.SVMModel = (weigh
ts=[-0.12218838697834929,-0.5275107581589767,
```

Machine Learning With Spark

- ***Training Classification model:***

  - Train the naïve Bayes model

```
val nbModel = NaiveBayes.train(nbData)
```



```
14/12/06 13:44:48 INFO DAGScheduler: Job 95 finished: collect at
NaiveBayes.scala:120, took 0.441273 s

nbModel: org.apache.spark.mllib.classification.NaiveBayesModel = org.
apache.spark.mllib.classification.NaiveBayesModel@666ac612
```

Your Connection to ICT Research

Machine Learning With Spark

- ***Training Classification model:***

  - Train the Decision tree model

```
val dtModel = DecisionTree.train(data, Algo.Classification, Entropy,
maxTreeDepth)
```

```
14/12/06 13:46:03 INFO DAGScheduler: Job 104 finished: collectAsMap at
DecisionTree.scala:653, took 0.031338 s
...
  total: 0.343024
  findSplitsBins: 0.119499
  findBestSplits: 0.200352
  chooseSplits: 0.199705
dtModel: org.apache.spark.mllib.tree.model.DecisionTreeModel =
DecisionTreeModel classifier of depth 5 with 61 nodes
```

Machine Learning With Spark

- *Generate Predictions*

- Use the logistic regression model: only the first feature

```
val dataPoint = data.first
val prediction = lrModel.predict(dataPoint.features)
```

The value is:

prediction: Double = 1.0

```
val trueLabel = dataPoint.label
```

The value is:

trueLabel = Double = 0.0

This is wrong ! ✗

Machine Learning With Spark

- *Generate Predictions*

- Use the logistic regression model: RDD vector

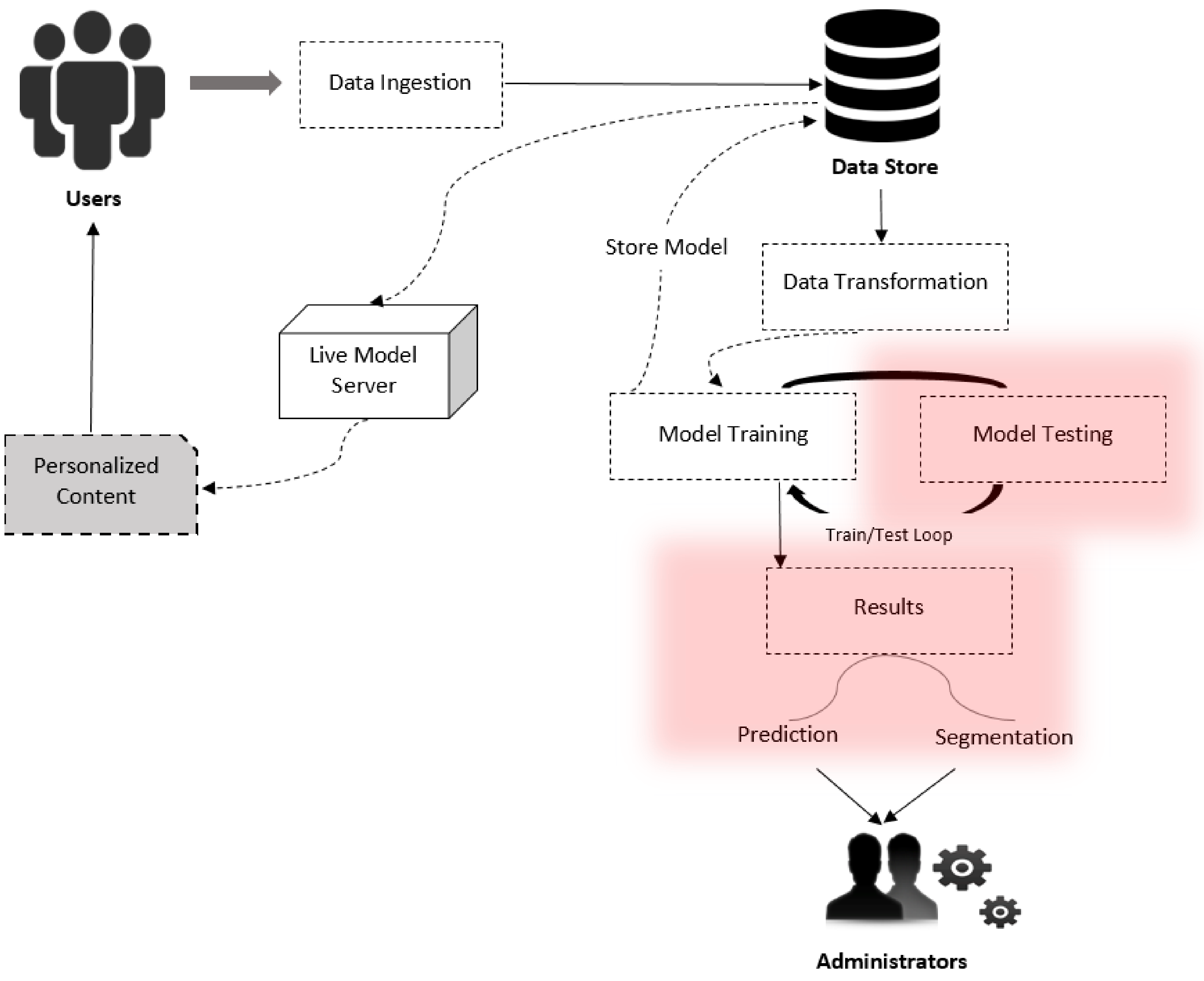
```
val predictions = lrModel.predict(data.map(lp => lp.features))
predictions.take(5)
```

The output is:

Array[Double] = Array(1.0, 1.0, 1.0, 1.0, 1.0)

Machine Learning With Spark

- *Evaluate Performance*



Machine Learning With Spark

- *Evaluate Performance*

- Accuracy and prediction error

```
val lrTotalCorrect = data.map { point => if (lrModel.predict(point.features)
    == point.label) 1 else 0 }.sum
val lrAccuracy = lrTotalCorrect / data.count
```

The output is:

```
lrAccuracy: Double = 0.5146720757268425
```

Not excellent !

Machine Learning With Spark

• *Improving model performance*

- Feature standardization, Additional features, Correct form of data
- Feature standardization

  Matrix of vectors using the *RowMatrix* class

```
import org.apache.spark.mllib.linalg.distributed.RowMatrix
  val vectors = data.map(lp => lp.features)
  val matrix = new RowMatrix(vectors)
  val matrixSummary = matrix.computeColumnSummaryStatistics()
```

```
println(matrixSummary.mean)
```

```
println(matrixSummary.min)
```

```
println(matrixSummary.variance)
```

Machine Learning With Spark

- *Improving model performance*

- Feature standardization

StandardScaler ou *Normalizer*

```
import org.apache.spark.mllib.feature.StandardScaler
val scaler = new StandardScaler(withMean = true, withStd = true).fit(vectors)
val scaledData = data.map(lp => LabeledPoint(lp.label,scaler.transform(lp.features)))
```

Your Connection to ICT Research

Machine Learning With Spark

- *Improving model performance*

  - Feature standardization

```
val lrModelScaled = LogisticRegressionWithSGD.train(scaledData,
 numIterations)
val lrTotalCorrectScaled = scaledData.map { point =>
    if (lrModelScaled.predict(point.features) == point.label) 1 else
0 }.sum
val lrAccuracyScaled = lrTotalCorrectScaled / numData
```

The output is:

lrAccuracyScaled: Double = 62.0419%

It’s better !

Your Connection to ICT Research

Machine Learning With Spark

• *Text Mining*

Text processing is too complex for two reasons:

- Text and language have an inherent structure
- The effective dimensionality of text data is extremely large

Possibilities with MLLib:

- Term weighting schemes, Feature hashing , tokenization, Removing stop words , Excluding terms based on frequency, stemming , etc.

Your Connection to ICT Research

Comparison with other tools

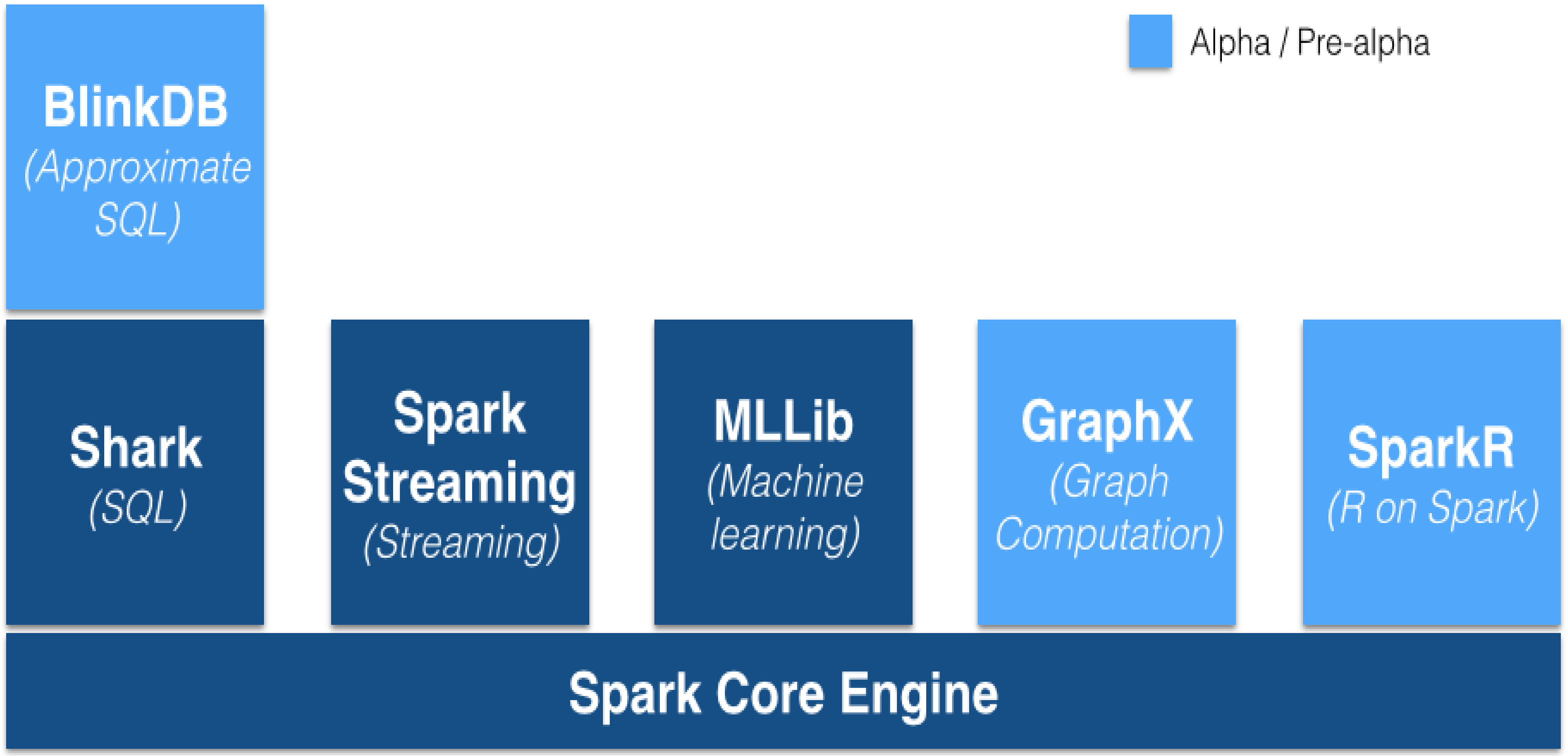
Spark MLLib and Apache Mahout:

- In case of Mahout it is Hadoop MapReduce and in case of MLib it is RDD
- Algorithms with Spark MLLib are more developed

Spark MLLib and R project:

- Spark includes SparkR library
- Working with R is not so fast

Other applications



Source: http://www.jorditorres.org/spark-ecosystem/

References

- Machine Learning With Spark Nick Pentreath
- spark.apache.org
- https://bighadoop.wordpress.com/2014/04/03/apache-spark-a-fast-big-data-analytics-engine/
- https://www.sigmoid.com/apache-spark-internals/
- http://www.jorditorres.org/spark-ecosystem/
- http://www.sas.com/
- https://trongkhoanguyenblog.wordpress.com

Source: http://www.jorditorres.org/spark-ecosystem/

cetic

Your Connection to ICT Research

Thank you

CETIC

Aéropôle de Charleroi-Gosselies
Rue des Frères Wright, 29/3
B-6041 Gosselies
info@cetic.be

www.cetic.be