Spark大數據分析與機器學習

國網中心

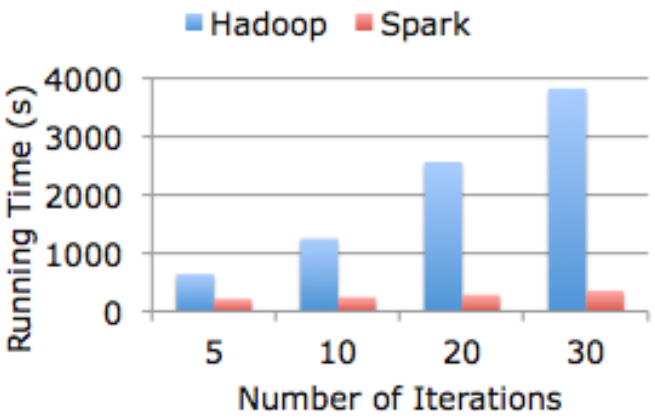
莊家雋 博士

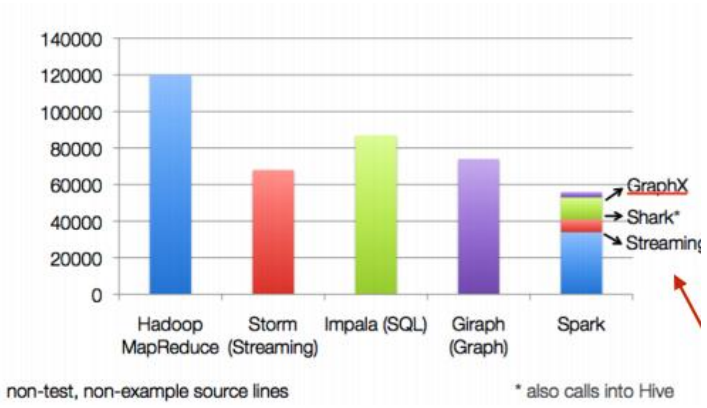
Outline

- What is Spark
  - Lab1: Spark Hello world
- RDD Operations
  - Lab2: Implement K-means using RDD
- DataFrame Operations
  - Lab3: Analysis structured apache log
- First class to Machine learning
- Machine Learning with RDD-based spark.mllib
  - Lab4: K-means using spark mllib
- Machine Learning with dataframe based spark.ml
  - Lab5: K-means using Pipeline

Why Spark

- Compare with Hadoop ecosystem
  - More efficient execution
  - More unified program abstraction
  - More flexible program operation





transformation
map(*func*)
filter(*func*)
flatMap(*func*)
sample(*withReplacement*, *fraction*, *seed*)
union(*otherDataset*)
distinct([*numTasks*]))
groupByKey([*numTasks*])
reduceByKey(*func*, [*numTasks*])
sortByKey([*ascending*], [*numTasks*])
join(*otherDataset*, [*numTasks*])
cogroup(*otherDataset*, [*numTasks*])
cartesian(*otherDataset*)

Spark 版本演進

Version	Original release date	Latest version	Release date
0.5	2012-06-12	0.5.1	2012-10-07
0.6	2012-10-14	0.6.2	2013-02-07[33]
0.7	2013-02-27	0.7.3	2013-07-16
0.8	2013-09-25	0.8.1	2013-12-19
0.9	2014-02-02	0.9.2	2014-07-23
1.0	2014-05-26	1.0.2	2014-08-05
1.1	2014-09-11	1.1.1	2014-11-26
1.2	2014-12-18	1.2.2	2015-04-17
1.3	2015-03-13	1.3.1	2015-04-17
1.4	2015-06-11	1.4.1	2015-07-15
1.5	2015-09-09	1.5.2	2015-11-09
1.6	2016-01-04	1.6.3	2016-11-07
2.0	2016-07-26	2.0.2	2016-11-14
2.1	2016-12-28	2.1.2	2017-10-09
2.2	2017-07-11	2.2.1	2017-12-01
2.3	2018-02-28	2.3.0	2018-02-28
Legend: Old version	Older version, still supported	Latest version	Latest preview version

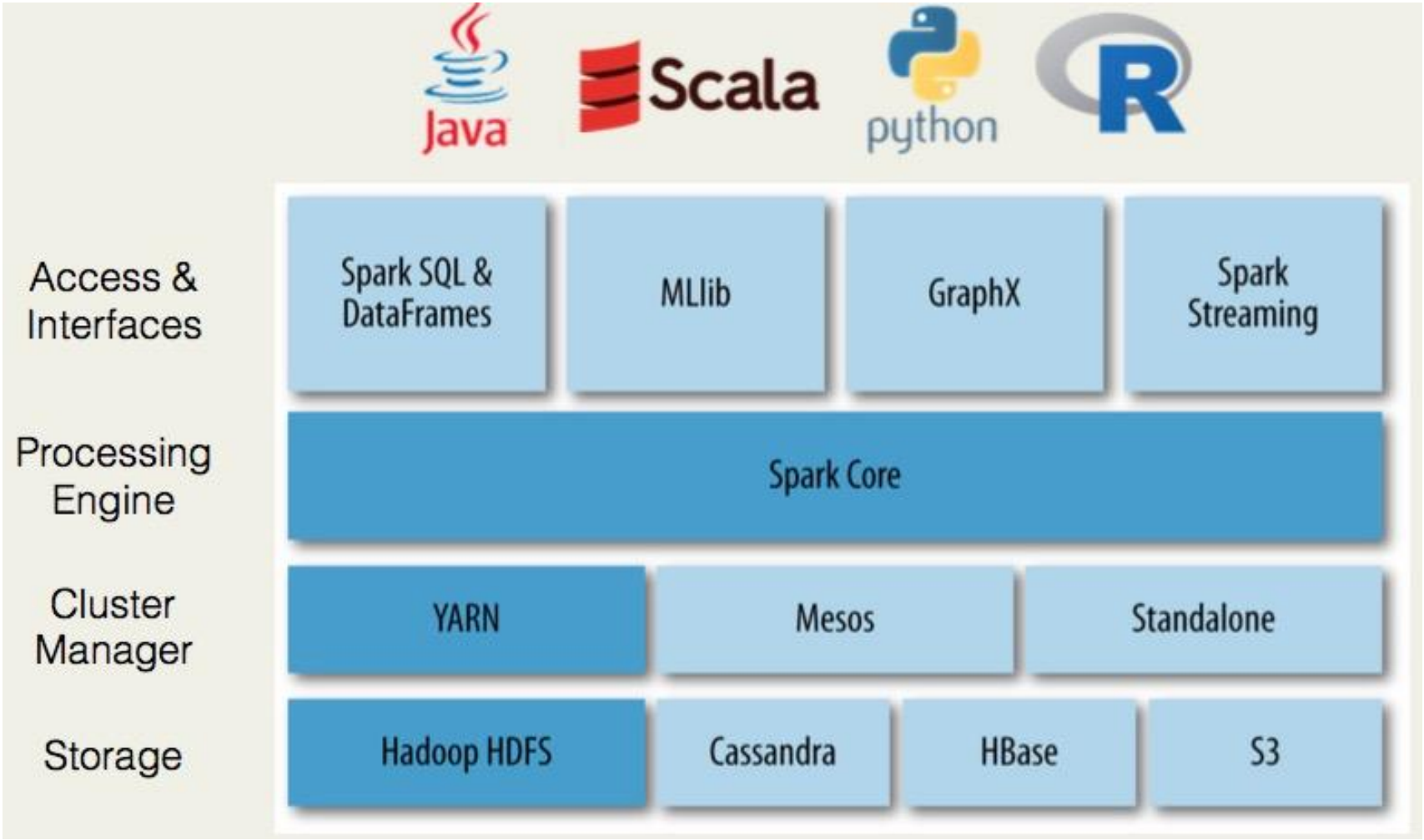
RDD

PySpark

DataFrames

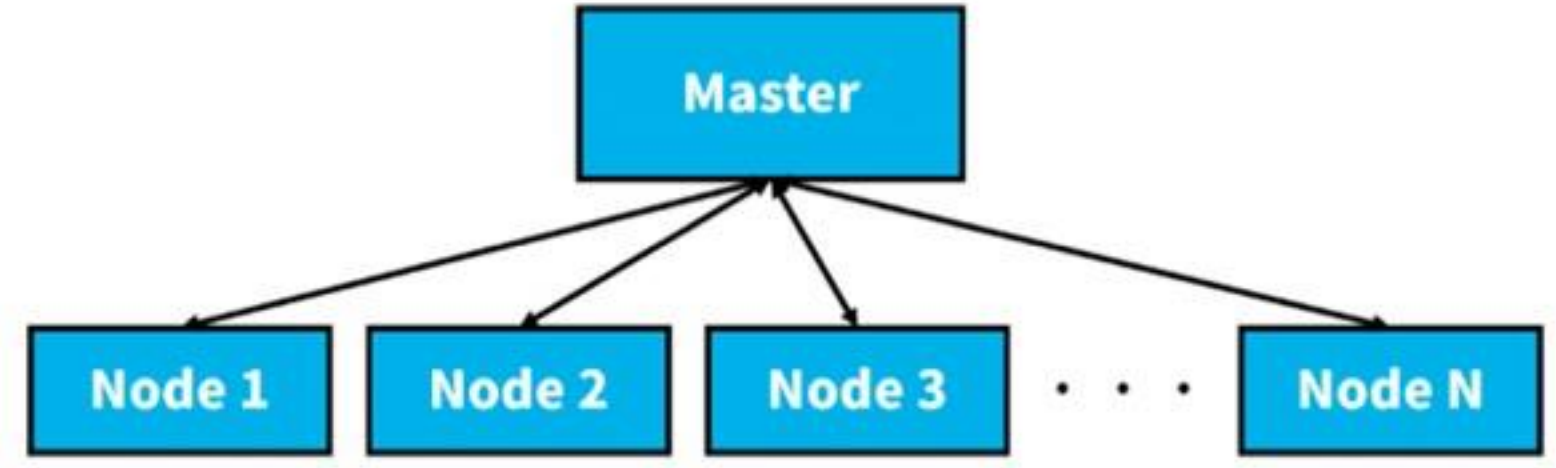
SparkR

Spark Stack


Java
Scala
python
R
Access & Interfaces
Spark SQL & DataFrames
MLlib
GraphX
Spark Streaming
Processing Engine
Spark Core
Cluster Manager
YARN
Mesos
Standalone
Storage
Hadoop HDFS
Cassandra
HBase
S3

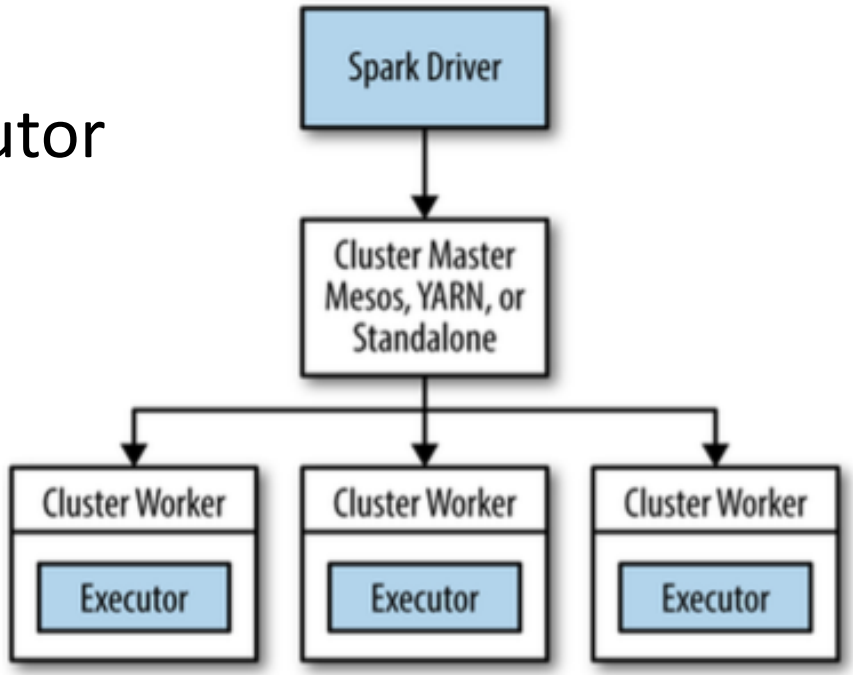
Distributed System

- Master /slave architecture



Spark Runtime Architecture

- Driver
  - Process which has the main() method
  - Convert application to tasks
    - DAGScheduler
  - Scheduling tasks on executor
    - TaskScheduler
    - SchedulerBackend
- Executor
  - Running individual task



Functional Programming 101

- Pass FUNCTION (what to do) as method parameter, rather than OBJECT (what)
  - Python support FP

```
Help on built-in function filter in module __builtin__:

filter(...)
    filter(function or None, sequence) -> list, tuple, or string

    Return those items of sequence for which function(item) is true.  If
    function is None, return the items that are true.  If sequence is a tuple
    or string, return the same type, else return a list.
```

FP in python

- Named function
  - i.e. normal function

```
>>> foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]
>>> def g(x): return x% 3 ==0
...
>>> print filter(g, foo)
[18, 9, 24, 12, 27]
```

- anonymous function
  - functions that are not bound to a name

```
>>> foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]
>>> print filter(lambda x: x%3==0, foo)
[18, 9, 24, 12, 27]
```

Lab 1: Hello World

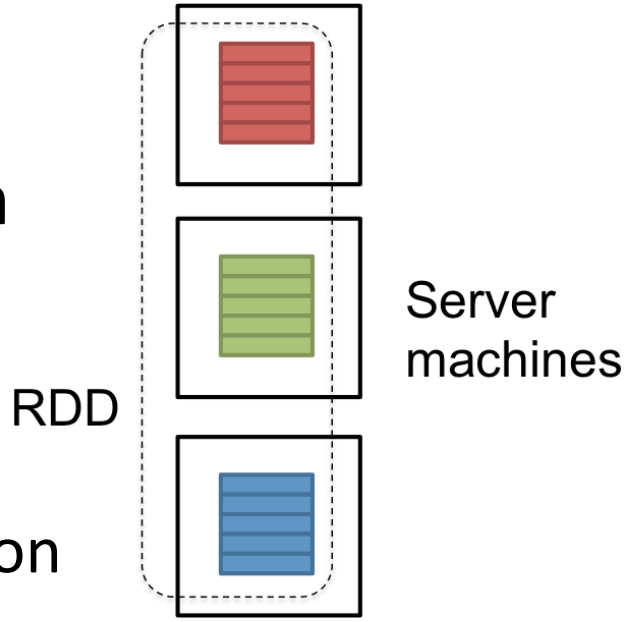
- Run PySpark interactive shell
  - Juypter
    - **>>>** lines = sc.textFile("/opt/spark/README.md")
    - **>>>** pythonLines = lines.filter(lambda line: "Python" in line)
    - **>>>** pythonLines.collect()

Outline

- What is Spark
  - Lab1: Spark Hello world
- RDD Operations
  - Lab2: Implement K-means using RDD
- DataFrame Operations
  - Lab3: Analysis structured apache log
- First class to Machine learning
- Machine Learning with RDD-based spark.mllib
  - Lab4: K-means using spark mllib
- Machine Learning with dataframe based spark.ml
  - Lab5: K-means using Pipeline

RDD Essentials

- Resilient distributed dataset
- Each RDD is split into multiple **partitions**
  - Partitions may exist on different machines
- immutable distributed collection of objects
  - Transform creates new RDD
  - Coarse-grained transformation
- Spark keeps track lineage graph
  - Fast recovery from failure
- Lazy Evaluation
  - Until action is called
- Function is passed into RDD operation



RDD operations

Transformations

- Create a new dataset from and existing one.
- Lazy in nature. They are executed only when some action is performed.
- Example :
  - Map(func)
  - Filter(func)
  - Distinct()

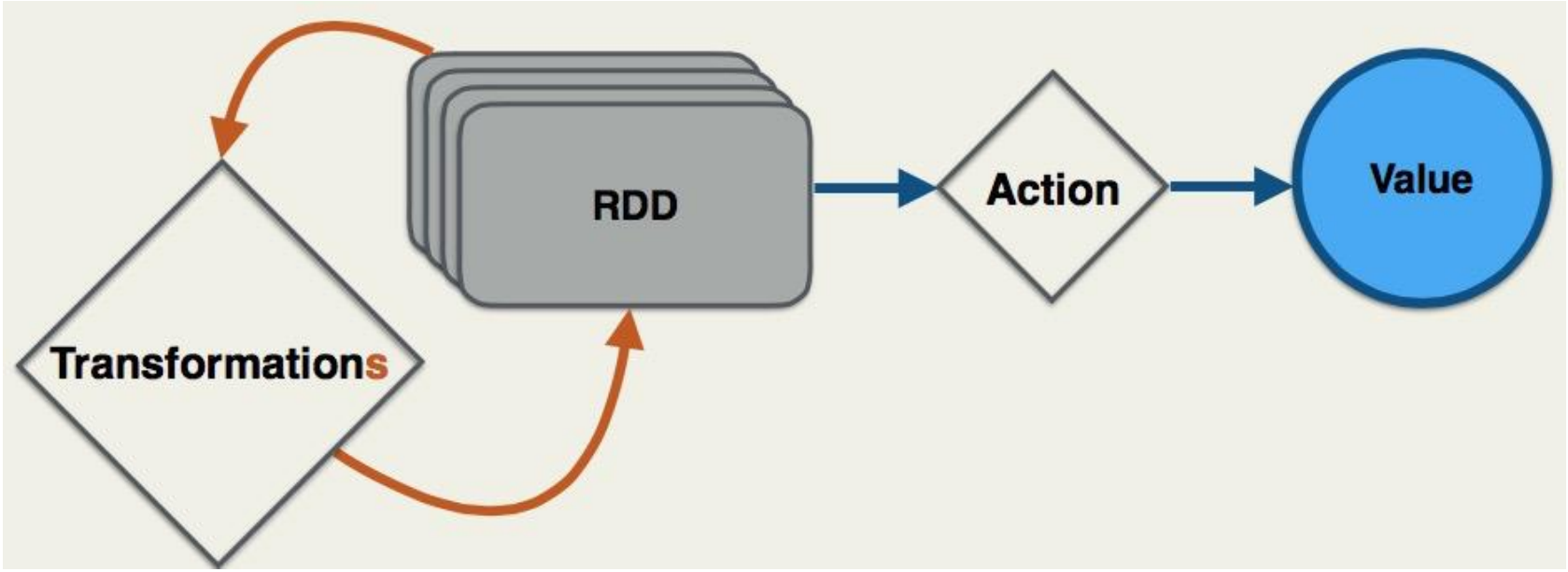
Actions

- Returns to the driver program a value or exports data to a storage system after performing a computation.
- Example:
  - Count()
  - Reduce(funct)
  - Collect
  - Take()

Persistence

- For caching datasets in-memory for future operations.
- Option to store on disk or RAM or mixed (Storage Level).
- Example:
  - Persist()
  - Cache()

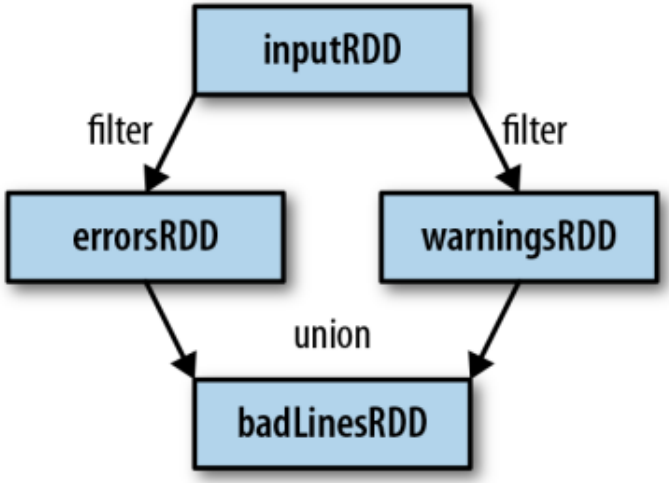
RDD transformation & Action



Lineage Graph

- Think of each RDD as consisting of instructions on how to compute the data through transformations.

```
errorsRDD = inputRDD.filter(lambda x: "error" in x)
warningsRDD = inputRDD.filter(lambda x: "warning" in x)
badLinesRDD = errorsRDD.union(warningsRDD)
```



Cache() or Persist()

- Each time call a action, the entire RDD must be computed “from scratch”
  - Cache/persist the computed lineage result
  - For iterative algorithm, keep temporary in memory can improve performance
    - The reason for in-memory computing

RDD Type

- Basic RDD[T]
  - considers each data item as a single value
- PairRDDs[K,V]
  - each data item containing key/value pairs.

http://blog.csdn.net/pelick/article/details/44922619
http://lxw1234.com/archives/2015/07/363.htm
http://homepage.cs.latrobe.edu.au/zhe/ZhenHeSparkRDDAPIExamples.html

Create RDD

- 從集合建立RDD
  - sc.parallelize([3,1, 2, 5, 5])
  - sc.parallelize(["Apple", "Orange”, "Banana", "Grape", "Apple”])

- 從檔案建立RDD
  - sc.textFile("file://path/to/file")

Basic RDD Transformation (part)

- RDD.map(func)

Table 3-2. Basic RDD transformations on an RDD containing {1, 2, 3, 3}

Function name	Purpose	Example	Result
`map()`	Apply a function to each element in the RDD and return an RDD of the result.	`rdd.map(x => x + 1)`	`{2, 3, 4, 4}`

Basic RDD Transformation (part)

- RDD.filter(func)

Table 3-2. Basic RDD transformations on an RDD containing {1, 2, 3, 3}

Function name	Purpose	Example	Result
`filter()`	Return an RDD consisting of only elements that pass the condition passed to `filter()`.	`rdd.filter(x => x != 1)`	`{2, 3, 3}`

Basic RDD Transformation (part)

- RDD.flatMap(func)

Table 3-2. Basic RDD transformations on an RDD containing {1, 2, 3, 3}

Function name	Purpose	Example	Result
`flatMap()`	Apply a function to each element in the RDD and return an RDD of the contents of the iterators returned. Often used to extract words.	`rdd.flatMap(x => x.to(3))`	`{1, 2, 3, 2, 3, 3, 3}`

Basic RDD Action (part)

- RDD.collect()

Table 3-4. Basic actions on an RDD containing {1, 2, 3, 3}

Function name	Purpose	Example	Result
`collect()`	Return all elements from the RDD.	`rdd.collect()`	`{1, 2, 3, 3}`

  - collect() will attempt to copy every single element in the RDD onto the single driver program, and then run out of memory and crash.

Basic RDD Action (part)

- RDD.reduce(func)

Table 3-4. Basic actions on an RDD containing {1, 2, 3, 3}

Function name	Purpose	Example	Result
`reduce(func)`	Combine the elements of the RDD together in parallel (e.g., `sum`).	`rdd.reduce((x, y) => x + y)`	9

Basic RDD Action (part)

- RDD.saveAsTextFile()
  - 若路徑為本地路徑，則只會存在executor所在機器上

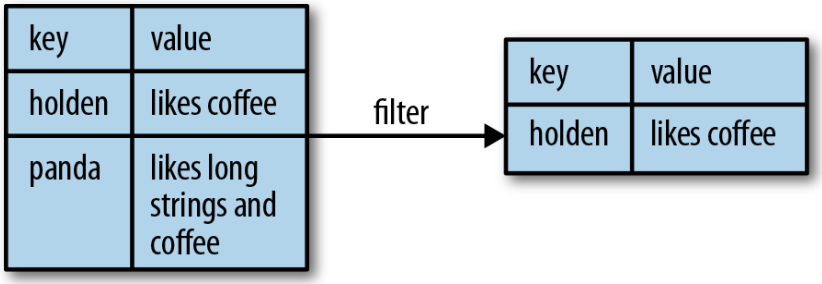
RDD Type

- Basic RDD[T]
  - considers each data item as a single value
- PairRDDs[K,V]
  - each data item containing key/value pairs.

http://blog.csdn.net/pelick/article/details/44922619
http://lxw1234.com/archives/2015/07/363.htm
http://homepage.cs.latrobe.edu.au/zhe/ZhenHeSparkRDDAPIExamples.html

PairRDD Transformation

- PairRDD is also a RDD
  - RDD.filter(func)
  - RDD.map(func)
  - RDD.flatMap(func)
  - …



PairRDD Transformation

- RDD.mapValues(func)

Table 4-1. Transformations on one pair RDD (example: {(1, 2), (3, 4), (3, 6)})

Function name	Purpose	Example	Result
`mapValues(func)`	Apply a function to each value of a pair RDD without changing the key.	`rdd.mapValues(x => x+1)`	`{(1, 3), (3, 5), (3, 7)}`

PairRDD Transformation

- RDD.reduceByKey(func)

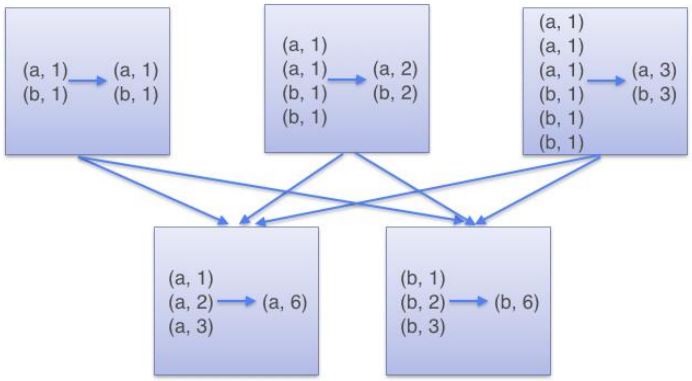


Table 4-1. Transformations on one pair RDD (example: {(1, 2), (3, 4), (3, 6)})

Function name	Purpose	Example	Result
`reduceByKey(func)`	Combine values with the same key.	`rdd.reduceByKey( (x, y) => x + y)`	`{(1, 2), (3, 10)}`

PairRDD Action

- RDD.countByKey()

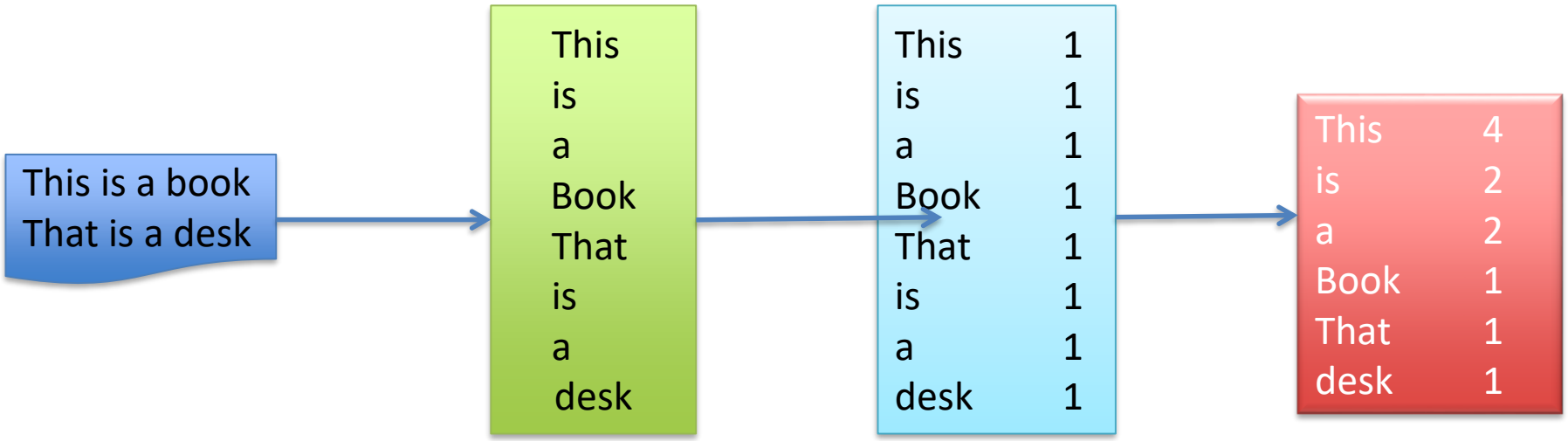
Table 4-3. Actions on pair RDDs (example ({(1, 2), (3, 4), (3, 6)}))

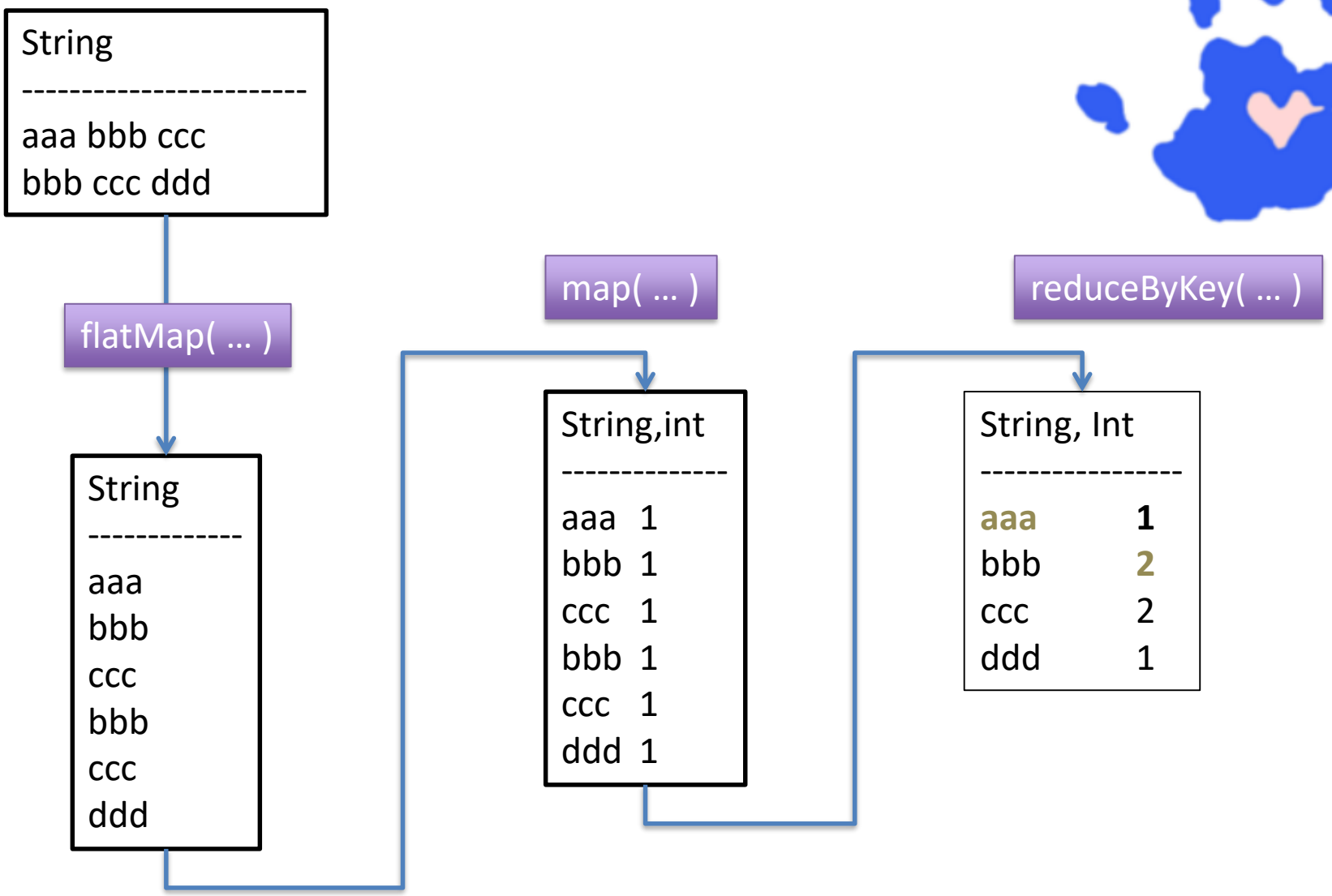
Function	Description	Example	Result
`countByKey()`	Count the number of elements for each key.	`rdd.countByKey()`	`{(1, 1), (3, 2)}`

PairRDD Action

- RDD.collectAsMap()
  - Return the key-value pairs in this RDD to the master as a Map.
  - 如果RDD中一個key對應到多個value，後面的value會覆蓋前面的value，因此最終得到的map一個key只會對到一個value

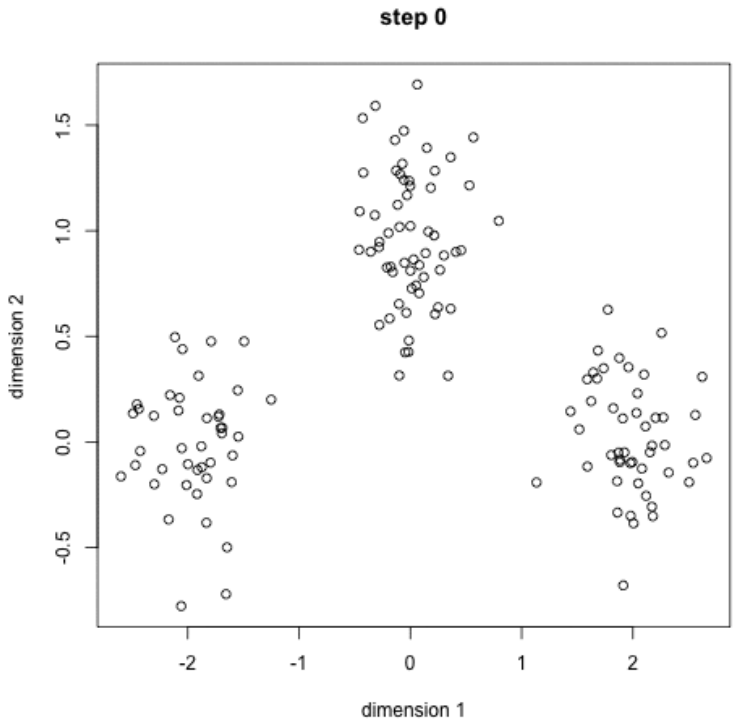
Example: Word Count

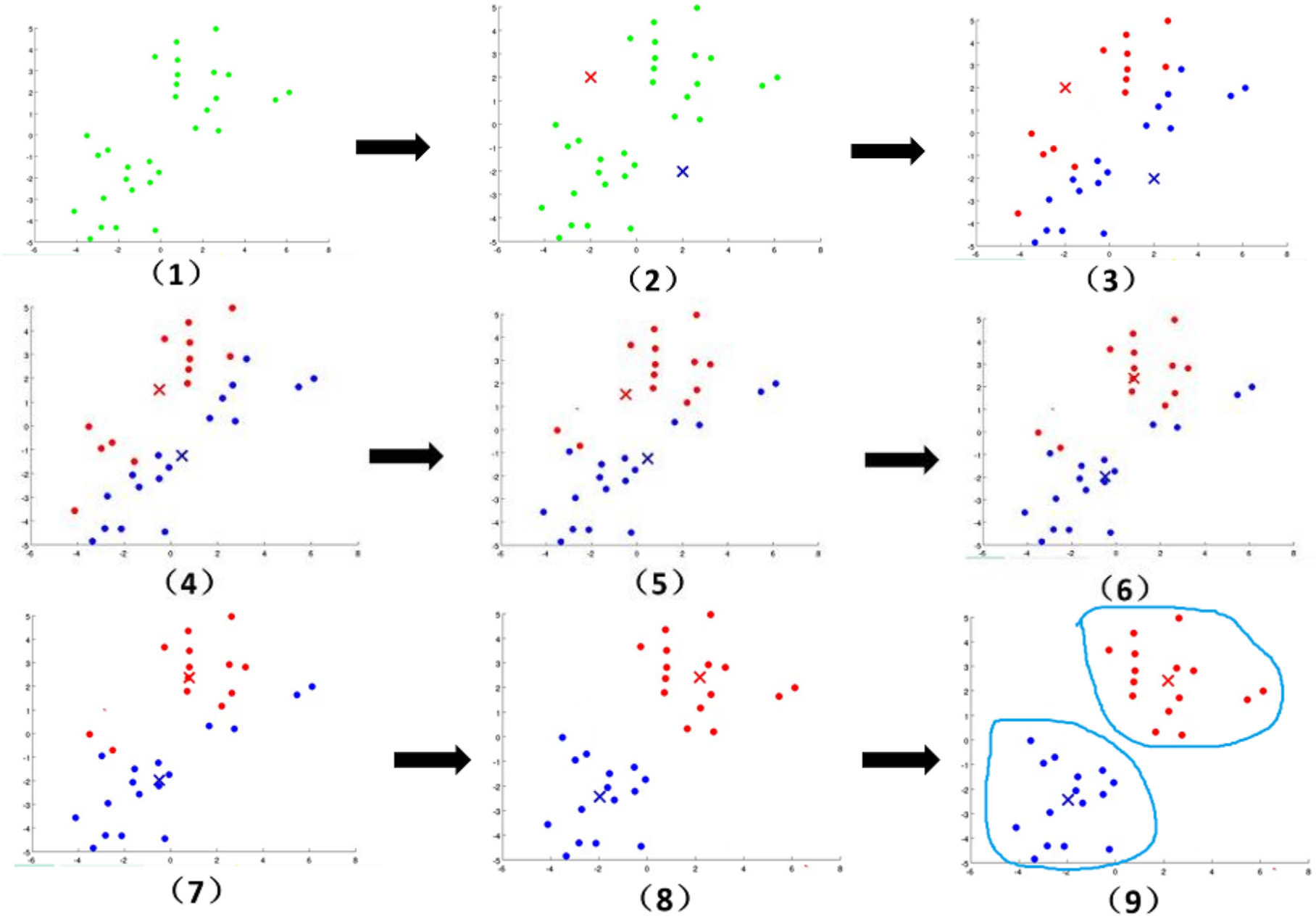



String
aaa bbb ccc
bbb ccc ddd
flatMap(…)
String
aaa
bbb
ccc
bbb
ccc
ddd
map(…)
String,int
aaa 1
bbb 1
ccc 1
bbb 1
ccc 1
ddd 1
reduceByKey(…)
String, Int
aaa 1
bbb 2
ccc 2
ddd 1

Lab 2: K-means

- 隨機選取資料組中的k筆資料當作初始群中心u_1~u_k
- 計算每個資料xi 對應到最短距離的群中心 (固定 ui 求解所屬群 Si)
- 利用目前得到的分類重新計算群中心 (固定 Si 求解群中心 ui)
- 重複step 2,3直到收斂 (達到最大疊代次數 or 群心中移動距離很小)




(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)

```
// Add in new data, one at a time, recalculating centroids with each new one.
while(!finish) {
    //Clear cluster state
    clearClusters();

    List lastCentroids = getCentroids();

    //Assign points to the closer cluster
    assignCluster();

    //Calculate new centroids.
    calculateCentroids();

    iteration++;

    List currentCentroids = getCentroids();

    //Calculates total distance between new and old Centroids
    double distance = 0;
    for(int i = 0; i &lt; lastCentroids.size(); i++) {
        distance += Point.distance(lastCentroids.get(i),currentCentroids.get(i));
    }
    System.out.println("##################");
    System.out.println("Iteration: " + iteration);
    System.out.println("Centroid distances: " + distance);
    plotClusters();

    if(distance == 0) {
        finish = true;
    }
```

初始
Centroids
0: (2, 2)
1: (5, 5)

np.Vector

(1, 1)
(1, 2)
(10, 11)

map(…)

Integer, [np.Vector, 1]
---
0 [(1, 1), 1]
0 [(1, 2), 1]
1 [(10, 11), 1]

reduceByKey(…)

Integer, [np.Vector, int]
---
0 (2, 3), 2
1 (10, 11), 1

map(…)

JavaPairRDD<Integer, Vector>
---
0 (1, 1.5)
1 (10, 11)

重新設定
Centroids
0: (1, 1.5)
1: (10, 11)

Outline

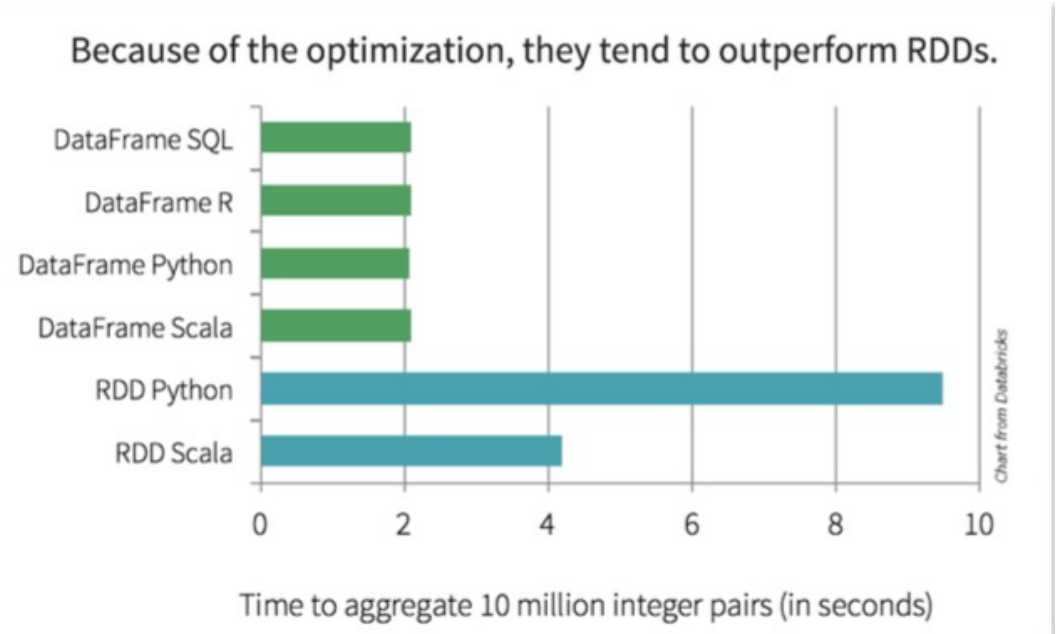
- What is Spark
  - Lab1: Spark Hello world
- RDD Operations
  - Lab2: Implement K-means using RDD
- DataFrame Operations
  - Lab3: Analysis structured apache log
- First class to Machine learning
- Machine Learning with RDD-based spark.mllib
  - Lab4: K-means using spark mllib
- Machine Learning with dataframe based spark.ml
  - Lab5: K-means using Pipeline

RDD v.s. DataFrame v.s. SparkSQL

- RDD:
  - 可直接讀取任何文字檔，需再進行加工
  - 需透過RDD operation進行程式開發
  - http://spark.apache.org/docs/2.1.0/api/python/pyspark.html#pyspark.RDD
- DataFrame:
  - 必須讀取結構化資料，讀進來後可以直接操作
  - 仍需使用DataFrame api進行操作
  - http://spark.apache.org/docs/2.1.0/api/python/pyspark.sql.html#pyspark.sql.DataFrame

RDD v.s. DataFrame v.s. SparkSQL

- Saprk SQL
  - 由DataFrame產生tempTable
  - Spark 2.0 後，支援SQL:2003全部語法
- Performance



Creating DataFrames

- From RDD

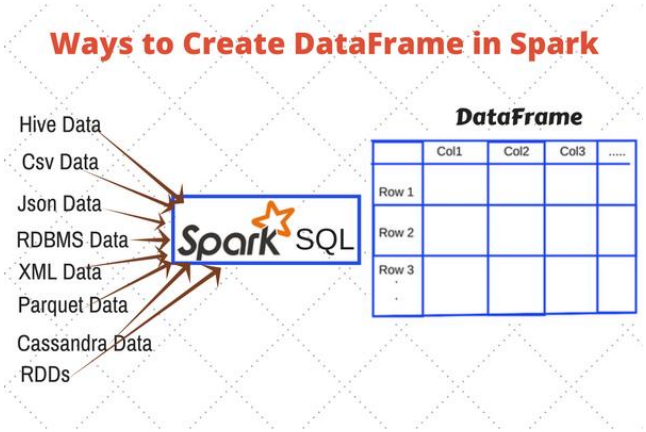
```
# Load a text file and convert each line to a Row.
lines = sc.textFile("examples/src/main/resources/people.txt")
parts = lines.map(lambda l: l.split(","))
people = parts.map(lambda p: Row(name=p[0], age=int(p[1])))

# Infer the schema, and register the DataFrame as a table.
schemaPeople = spark.createDataFrame(people)
```

- From CSV/JSON … etc.

```
>>> df = spark.read.csv('python/test_support/sql/ages.csv')
>>> df.dtypes
[('_c0', 'string'), ('_c1', 'string')]
```

```
>>> df1 = spark.read.json('python/test_support/sql/people.json')
>>> df1.dtypes
[('age', 'bigint'), ('name', 'string')]
>>> rdd = sc.textFile('python/test_support/sql/people.json')
>>> df2 = spark.read.json(rdd)
>>> df2.dtypes
[('age', 'bigint'), ('name', 'string')]
```



DataFrames API

- Select

```
>>> df.select('*').collect()
[Row(age=2, name=u'Alice'), Row(age=5, name=u'Bob')]
>>> df.select('name', 'age').collect()
[Row(name=u'Alice', age=2), Row(name=u'Bob', age=5)]
>>> df.select(df.name, (df.age + 10).alias('age')).collect()
[Row(name=u'Alice', age=12), Row(name=u'Bob', age=15)]
```

- orderby

```
>>> df.sort(df.age.desc()).collect()
[Row(age=5, name=u'Bob'), Row(age=2, name=u'Alice')]
>>> df.sort("age", ascending=False).collect()
[Row(age=5, name=u'Bob'), Row(age=2, name=u'Alice')]
>>> df.orderBy(df.age.desc()).collect()
[Row(age=5, name=u'Bob'), Row(age=2, name=u'Alice')]
>>> from pyspark.sql.functions import *
>>> df.sort(asc("age")).collect()
[Row(age=2, name=u'Alice'), Row(age=5, name=u'Bob')]
>>> df.orderBy(desc("age"), "name").collect()
[Row(age=5, name=u'Bob'), Row(age=2, name=u'Alice')]
>>> df.orderBy(["age", "name"], ascending=[0, 1]).collect()
[Row(age=5, name=u'Bob'), Row(age=2, name=u'Alice')]
```

- groupBy

```
>>> df.groupBy().avg().collect()
[Row(avg(age)=3.5)]
>>> sorted(df.groupBy('name').agg({'age': 'mean'}).collect())
[Row(name=u'Alice', avg(age)=2.0), Row(name=u'Bob', avg(age)=5.0)]
>>> sorted(df.groupBy(df.name).avg().collect())
[Row(name=u'Alice', avg(age)=2.0), Row(name=u'Bob', avg(age)=5.0)]
>>> sorted(df.groupBy(['name', df.age]).count().collect())
[Row(name=u'Alice', age=2, count=1), Row(name=u'Bob', age=5, count=1)]
```

- filter

```
>>> df.filter(df.age > 3).collect()
[Row(age=5, name=u'Bob')]
>>> df.where(df.age == 2).collect()
[Row(age=2, name=u'Alice')]
```

```
>>> df.filter("age > 3").collect()
[Row(age=5, name=u'Bob')]
>>> df.where("age = 2").collect()
[Row(age=2, name=u'Alice')]
```

- join

```
>>> df.join(df2, df.name == df2.name, 'outer').select(df.name, df2.height).collect()
[Row(name=None, height=80), Row(name=u'Bob', height=85), Row(name=u'Alice', height=None)]
```

```
>>> df.join(df2, 'name').select(df.name, df2.height).collect()
[Row(name=u'Bob', height=85)]
```

Running SQL Queries

- Register DataFrames as a table first
  - <DF_NAME>.registerTempTable("<TABLE_NAME>")
- Run query
  - Ver. 2.0 : spark.sql("SELECT ….")
  - Ver. 1.6: sqlContext.sql(…)
- Result is returned as a DataFrame.

```
# Global temporary view is tied to a system preserved database `global_temp`
spark.sql("SELECT * FROM global_temp.people").show()
```

```
teenagers = sqlContext.sql("SELECT name FROM people WHERE age >= 13 AND age <= 19")
```

Lab3 Apache log analysis:

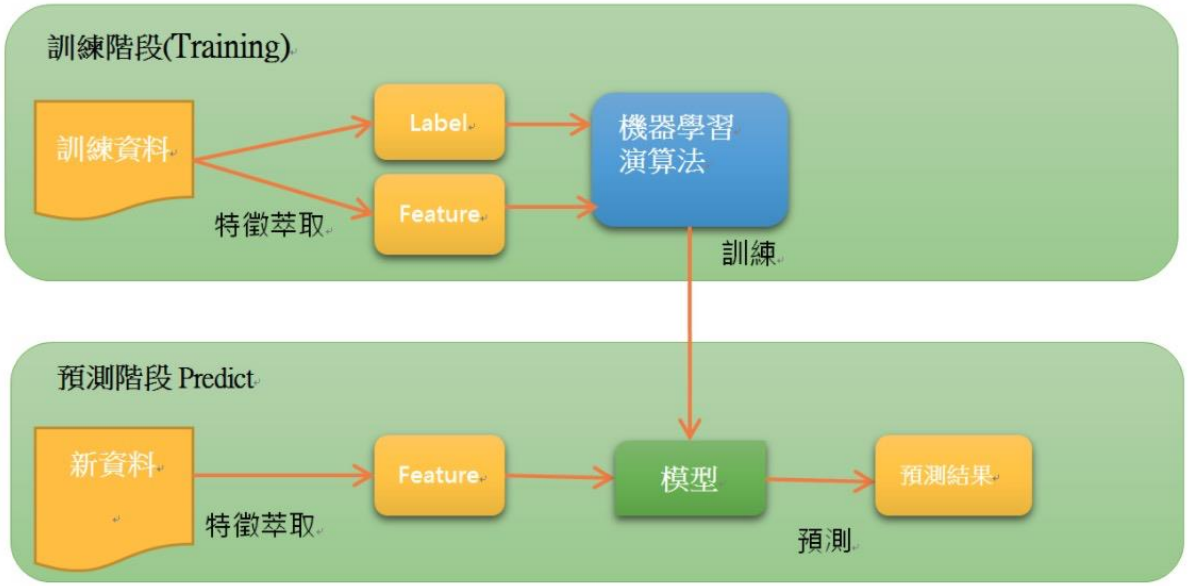
- 分別使用RDD、dataframe、spark sql分析下列三種情況：
  - 算出status為304共有幾筆
  - 在status為304的log裡，找出不同path的count
  - 在status為304的log裡，找出不同host的count

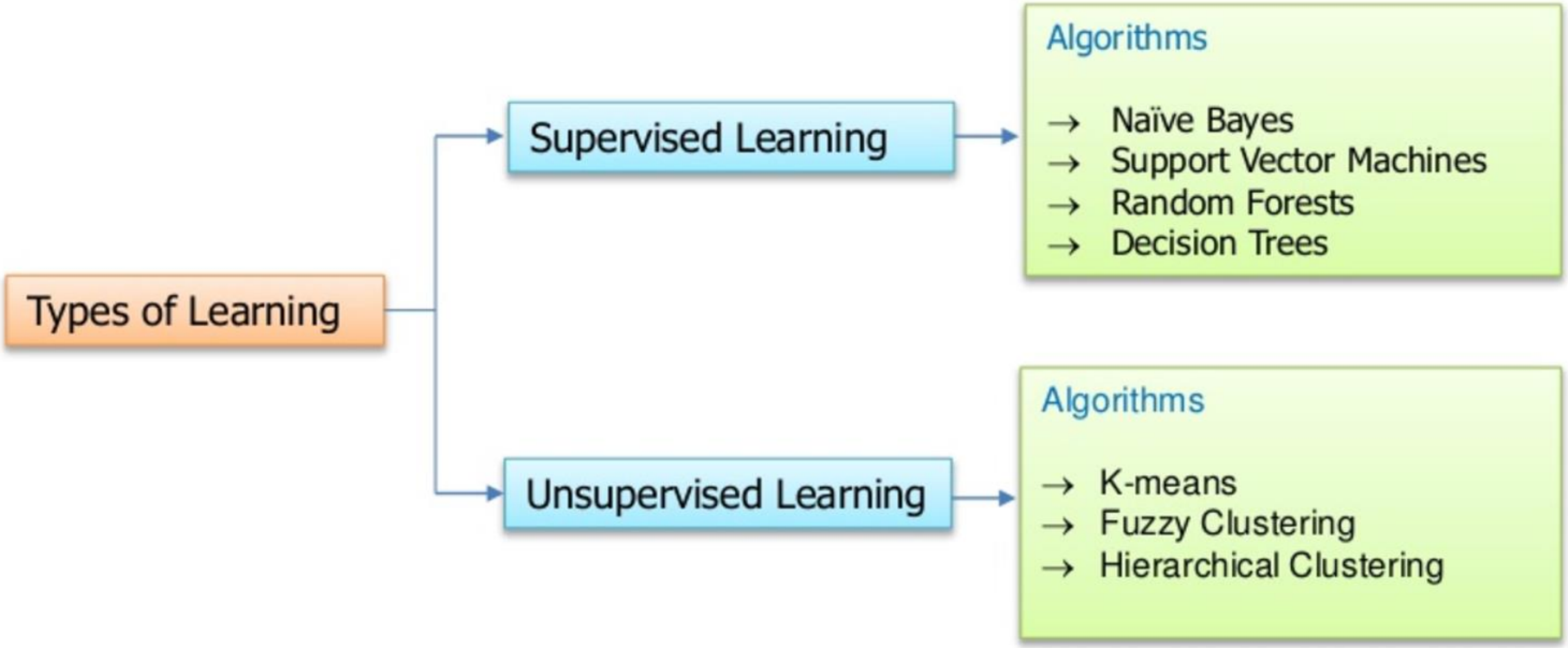
Outline

- What is Spark
  - Lab1: Spark Hello world
- RDD Operations
  - Lab2: Implement K-means using RDD
- DataFrame Operations
  - Lab3: Analysis structured apache log
- First class to Machine learning
- Machine Learning with RDD-based spark.mllib
  - Lab4: K-means using spark mllib
- Machine Learning with dataframe based spark.ml
  - Lab5: K-means using Pipeline

What is Machine learning

- 透過演算法，使用歷史資料進行訓練，訓練完成後會產生模型。未來當有新的資料，我們可以使用訓練產生的模型進行預測。
- https://www.youtube.com/watch?v=ty-kTUzMnjk




Types of Learning
Supervised Learning
Algorithms
→ Naïve Bayes
→ Support Vector Machines
→ Random Forests
→ Decision Trees
Unsupervised Learning
Algorithms
→ K-means
→ Fuzzy Clustering
→ Hierarchical Clustering

Regression & Classification & Clustering (定義)

- Supervised learning
  - 已知的一些資料輸入項目後，能夠透過模型與對應關係的建構，得到可以預期或是有預測能力的特定資料輸出
  - Regression
  - Classification
- Unsupervised learning
  - 給定相關資料輸入之後，透過適當的資料處理，讓資料替自己說話。資料輸出的情況是無法預測。
  - Clustering

Regression & Classification & Clustering (用途)

- Supervised learning
  - Regression
    - 預測股價
  - Classification
    - 判斷是否為垃圾郵件
- Unsupervised learning
  - Clustering
    - 市場客戶分群

Regression & Classification & Clustering (差異)

- Supervised learning
  - Regression
    - 預測結果是連續
  - Classification
    - 預測結果是離散
- Unsupervised learning
  - Clustering
    - 將一堆物件有相同特性的分成一群

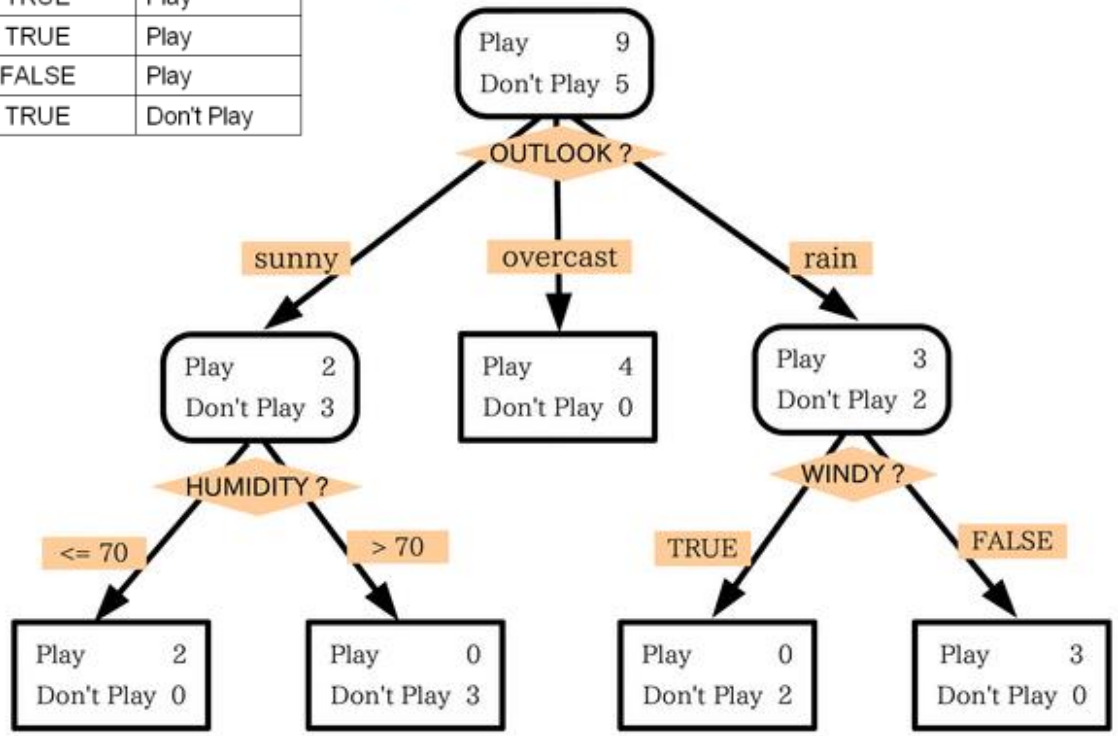
Regression & Classification & Clustering (範例演算法)

- Supervised learning
  - Regression
    - Decision tree
    - 預測腳踏車每小時的租借量
  - Classification
    - Decision tree
    - 預測網頁是暫時的或是長青的
- Unsupervised learning
  - Clustering
    - K-means (Labs)

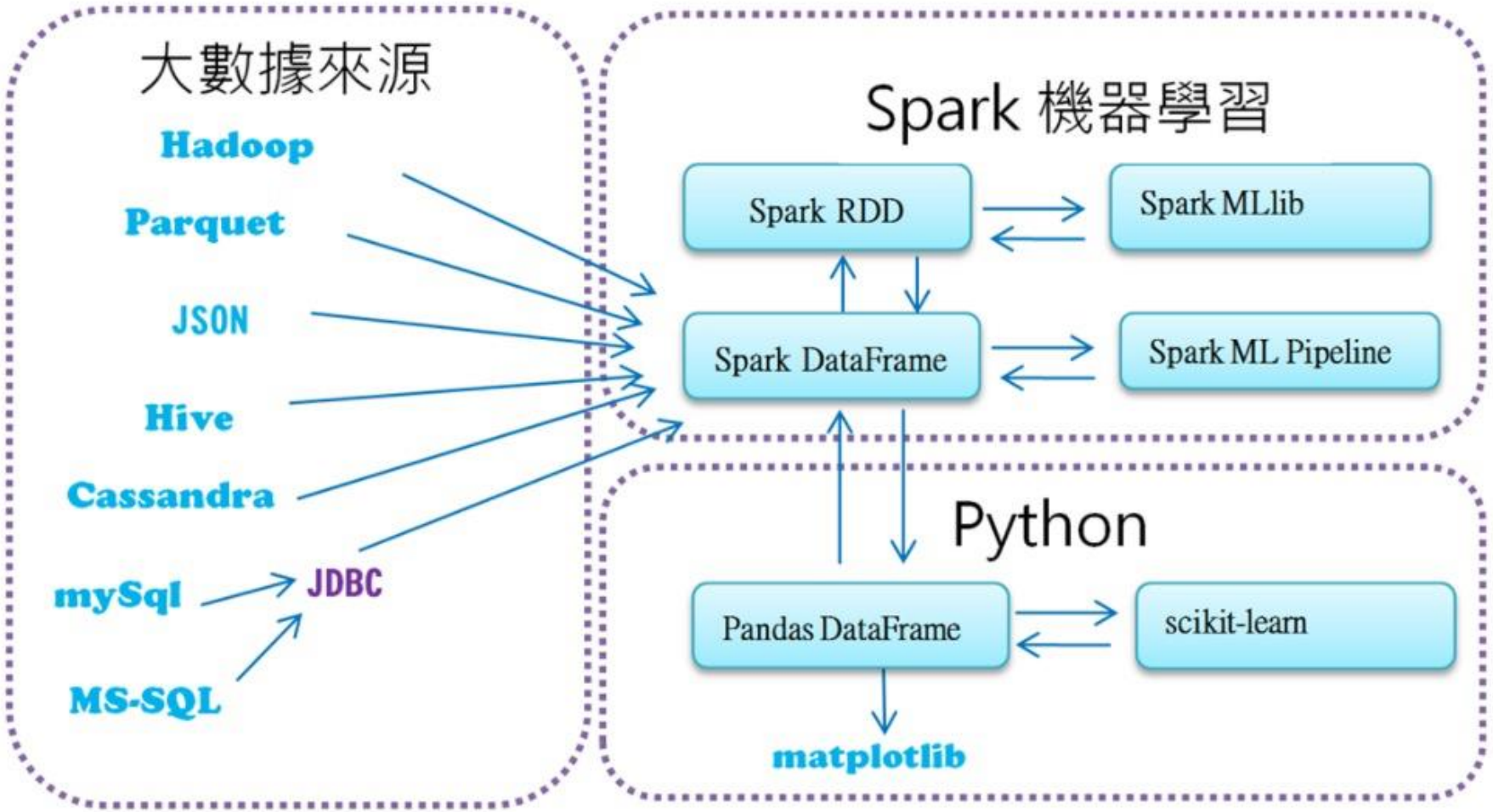
Play golf dataset

Independent variables				Dep. var
OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY
sunny	85	85	FALSE	Don't Play
sunny	80	90	TRUE	Don't Play
overcast	83	78	FALSE	Play
rain	70	96	FALSE	Play
rain	68	80	FALSE	Play
rain	65	70	TRUE	Don't Play
overcast	64	65	TRUE	Play
sunny	72	95	FALSE	Don't Play
sunny	69	70	FALSE	Play
rain	75	80	FALSE	Play
sunny	75	70	TRUE	Play
overcast	72	90	TRUE	Play
overcast	81	75	FALSE	Play
rain	71	80	TRUE	Don't Play

Dependent variable: PLAY



Python + Spark + ML



Spark Machine Learning Library

- MLlib RDD-based API is in maintenance mode
- ML DataFrame-based API is primary API
  - Pipeline: 建立ML的工作流程

	Spark.mllib	Spark.ml
Since	Ver. 0.8~	Ver. 1.2 ~
Datatype	RDD	DataFrame
API	contains the original API built on top of RDDs.	higher-level API built on top of DataFrames
優點	發展較早，演算法較多	結構化資料，資料操作較簡單

Outline

- What is Spark
  - Lab1: Spark Hello world
- RDD Operations
  - Lab2: Implement K-means using RDD
- DataFrame Operations
  - Lab3: Analysis structured apache log
- First class to Machine learning
- Machine Learning with RDD-based spark.mllib
  - Lab4: K-means using spark mllib
- Machine Learning with dataframe based spark.ml
  - Lab5: K-means using Pipeline

RDD-based mllib缺點

- 需要針對讀進來的RDD資料進行前處理
- 不同演算法的RDD[T]都不同
  - DecisionTree: RDD[LabeledPoint]
  - Kmean: RDD[array]

- 不同的演算法所用的訓練與預測api皆不同
  - DecisionTree.trainClassifier()
  - DecisionTree.trainRegressor()
  - Kmeans().train()

LabelPoint

- 透過RDD.map()產生出演算法所需要用的資料結構

Lab4: k-means

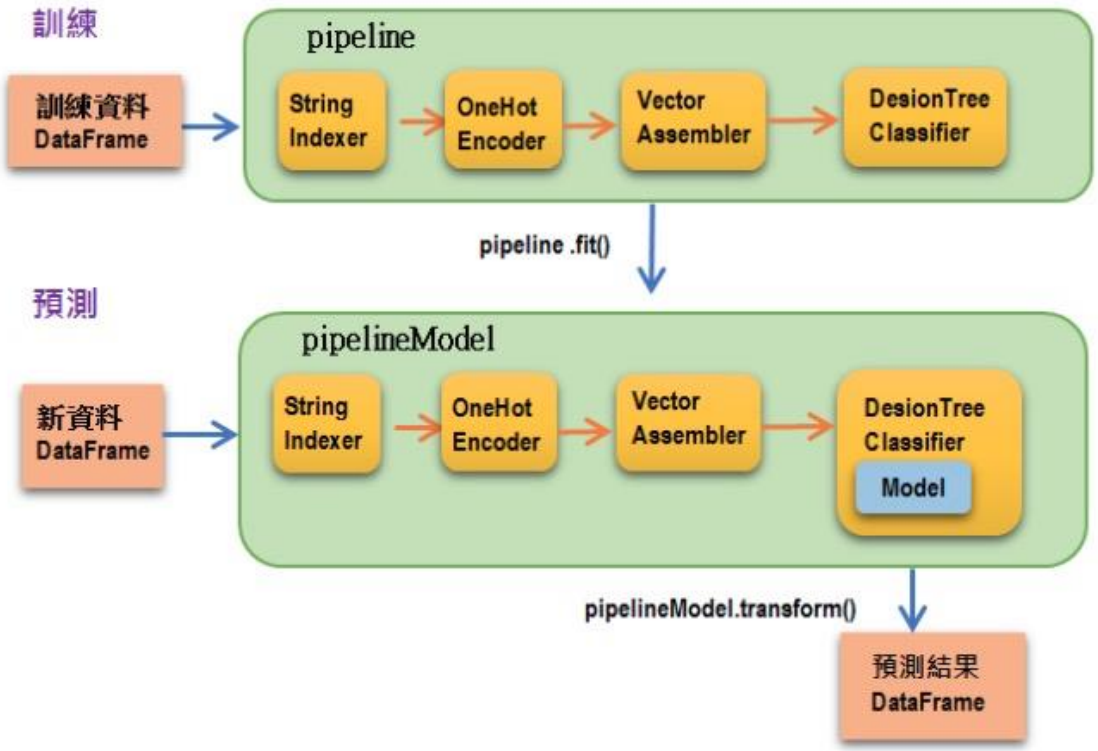
- 使用map()將RDD[tuple]轉化成RDD[array]
  - RDD[(x,y)] → RDD[[x,y]]
- 使用Kmean.train()進行訓練

Outline

- What is Spark
  - Lab1: Spark Hello world
- RDD Operations
  - Lab2: Implement K-means using RDD
- DataFrame Operations
  - Lab3: Analysis structured apache log
- First class to Machine learning
- Machine Learning with RDD-based spark.mllib
  - Lab4: K-means using spark mllib
- Machine Learning with dataframe based spark.ml
  - Lab5: K-means using Pipeline

Spark.ML

- 以DataFrame為基礎的機器學習模組
  - Spark DataFrame: 受Pandas啟發的資料處理架構
  - Spark ML pipeline: 受Scikit-learn啟發的ML架構



資料前處理工具

- 不是每一種前處理都需要，視資料狀況混合使用
- StringIndexer
  - 將文字欄位，轉成數字
  - 男→0，女→1
- OneHotEncoder
  - 將數字欄位轉成多個欄位的向量
  - 0→[1,0]，1→[0,1]
- VectorAssembler
  - 將多個欄位組成一個特徵向量
  - 男，180cm → [1,0][180] → [1,0,180]

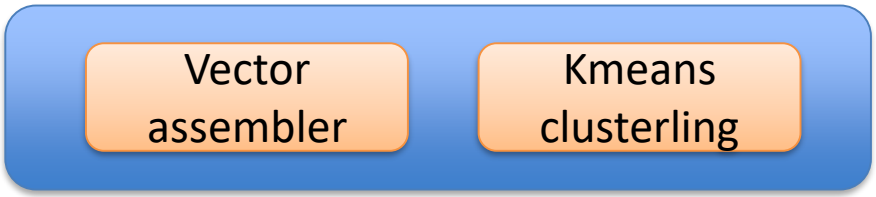
Pipeline

- 提供統一的訓練與預測語法
- pipeline使用fit()進行訓練，產生model
- model使用transform進行預測

Lab5: k-means pipeline

- 只需要vector assembler

```
+-------------------+--------------------+
|                  x|                   y|
+-------------------+--------------------+
|-1.0789262655979264|-0.29058890178264374|
|-1.5631374488292438|-0.01310586267402436|
|-1.2299535180277972|-0.10703168514429007|
|-1.2867842461628365| 0.45799739022887676|
|-2.9858188383793522|-0.15985872797137646|
+-------------------+--------------------+
```




```
+-------------------+--------------------+----------------------------------------+
|x                  |y                   |features                                |
+-------------------+--------------------+----------------------------------------+
|-1.0789262655979264|-0.29058890178264374|[-1.0789262655979264,-0.29058890178264374]|
|-1.5631374488292438|-0.01310586267402436|[-1.5631374488292438,-0.01310586267402436]|
|-1.2299535180277972|-0.10703168514429007|[-1.2299535180277972,-0.10703168514429007]|
|-1.2867842461628365|0.45799739022887676 |[-1.2867842461628365,0.45799739022887676] |
|-2.9858188383793522|-0.15985872797137646|[-2.9858188383793522,-0.15985872797137646]|
+-------------------+--------------------+----------------------------------------+
```

```
+-------------------+--------------------+----------------------------------------+----------+
|x                  |y                   |features                                |prediction|
+-------------------+--------------------+----------------------------------------+----------+
|-1.0789262655979264|-0.29058890178264374|[-1.0789262655979264,-0.29058890178264374]|0         |
|-1.5631374488292438|-0.01310586267402436|[-1.5631374488292438,-0.01310586267402436]|0         |
|-1.2299535180277972|-0.10703168514429007|[-1.2299535180277972,-0.10703168514429007]|0         |
|-1.2867842461628365|0.45799739022887676 |[-1.2867842461628365,0.45799739022887676] |0         |
|-2.9858188383793522|-0.15985872797137646|[-2.9858188383793522,-0.15985872797137646]|0         |
```