亦思科技股份有限公司

Spark培訓課程

卓正欣 Sam Cho
sam@is-land.com.tw

課程大綱

Spark是什麼

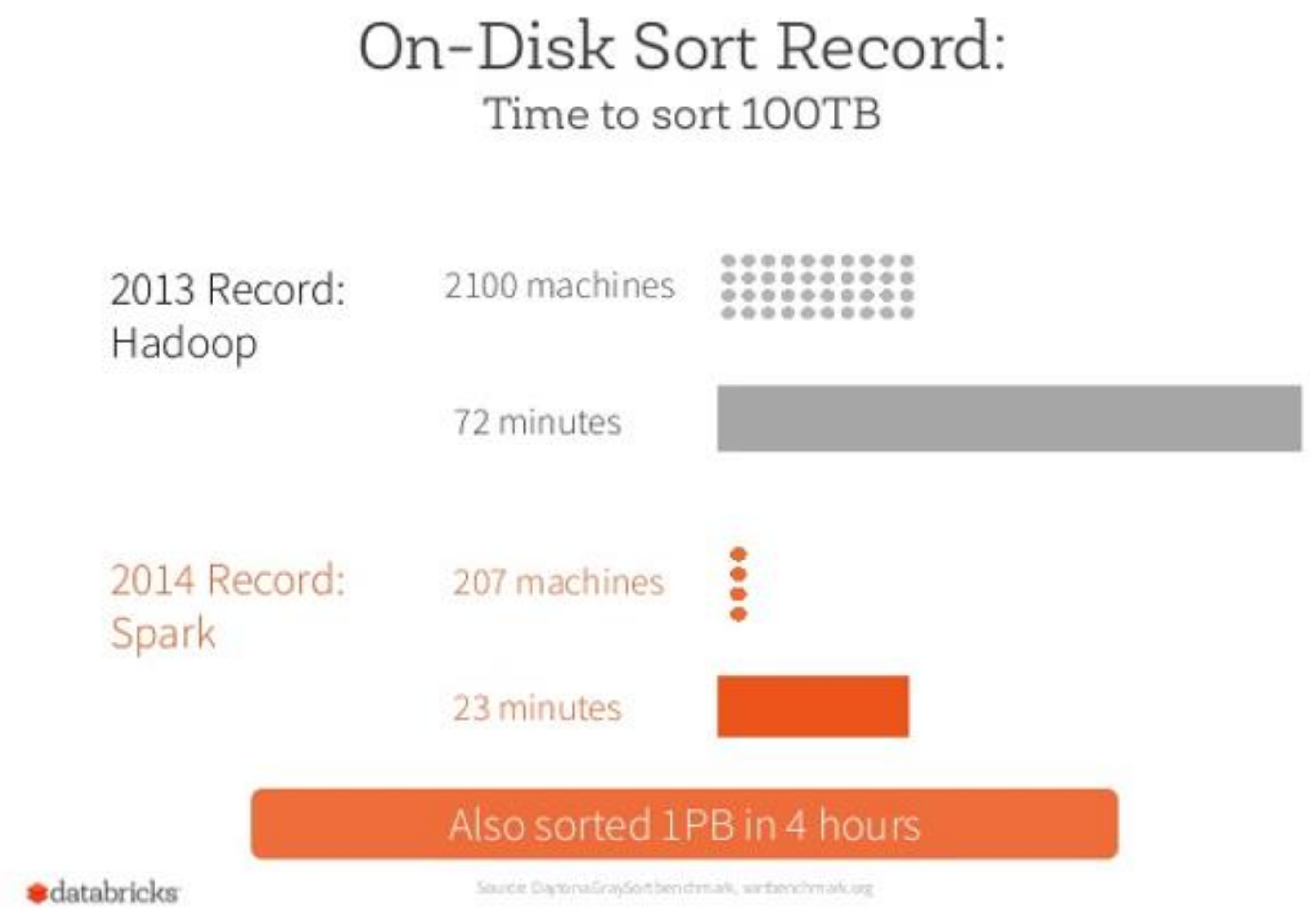
- 輕量級叢集運算工具
  - 執行速度快
  - 容易使用
  - 適用各種平台

Spark特點

- ❑叢集運算平台
- ❑記憶體內計算框架
- ❑提供不同的使用方式
- ❑支援不同的程式語言使用

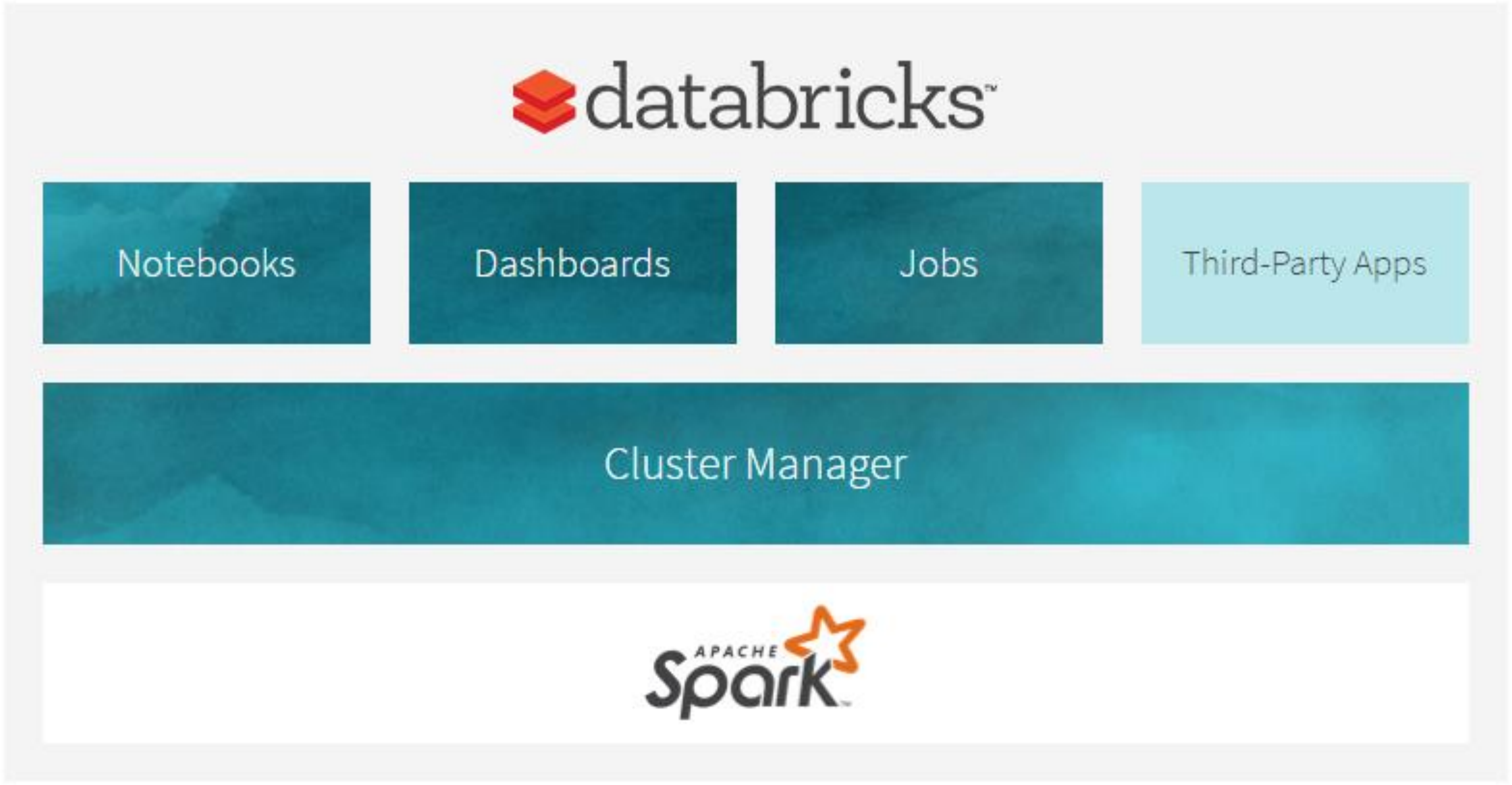
Spark運行速度超越Hadoop

- **Beats the world record for fastest sorting**



叢集運算平台

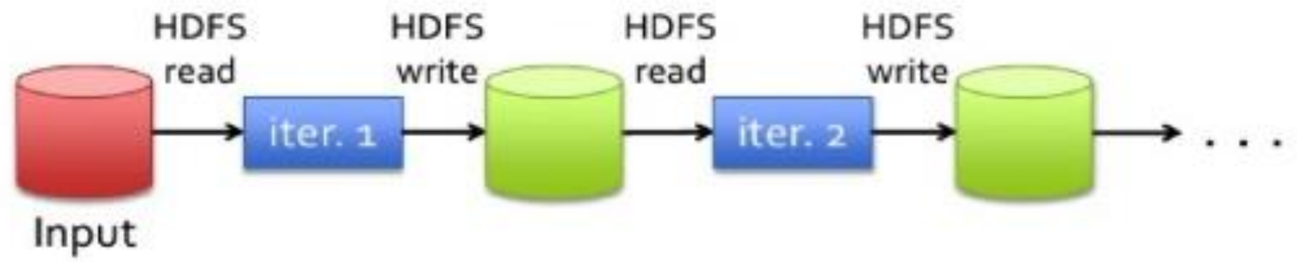
❑Databricks



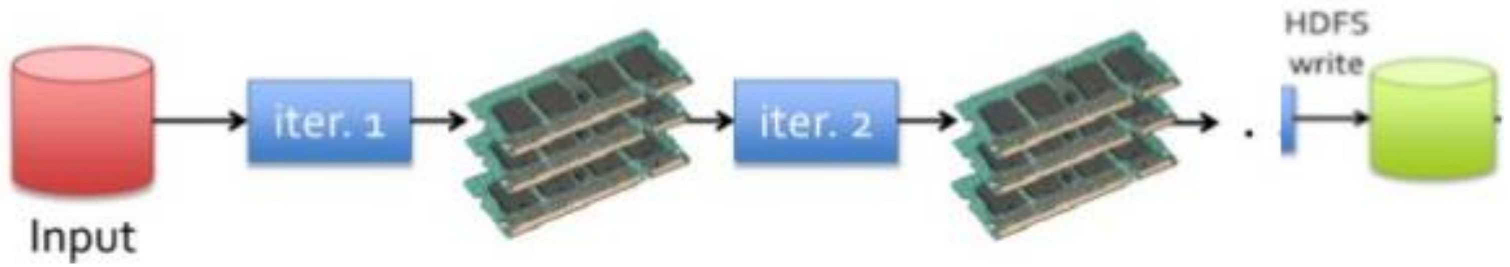
Reference : https://databricks.com/product/databricks

記憶體內計算框架

- HDFS MapReduce

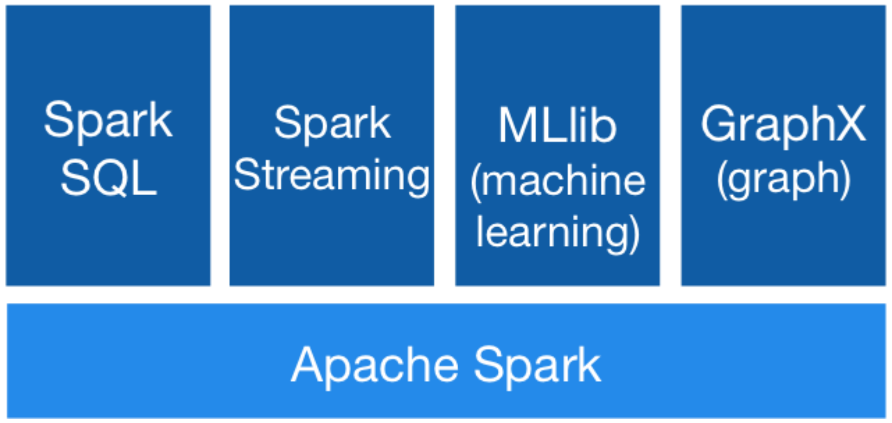


- Spark



提供不同的使用方式

- **Spark SQL**
  - 提供SQL功能查詢結構化資料
- **Spark Streaming**
  - 提供API操作資料流，可以不間斷的處理資料
- **Spark MLlib**
  - 提供類似Machine Learning的功能，在建置好的Spark叢集內進行分析處理
- **Spark GraphX**
  - 提供繪製graph



Reference : http://spark.apache.org/

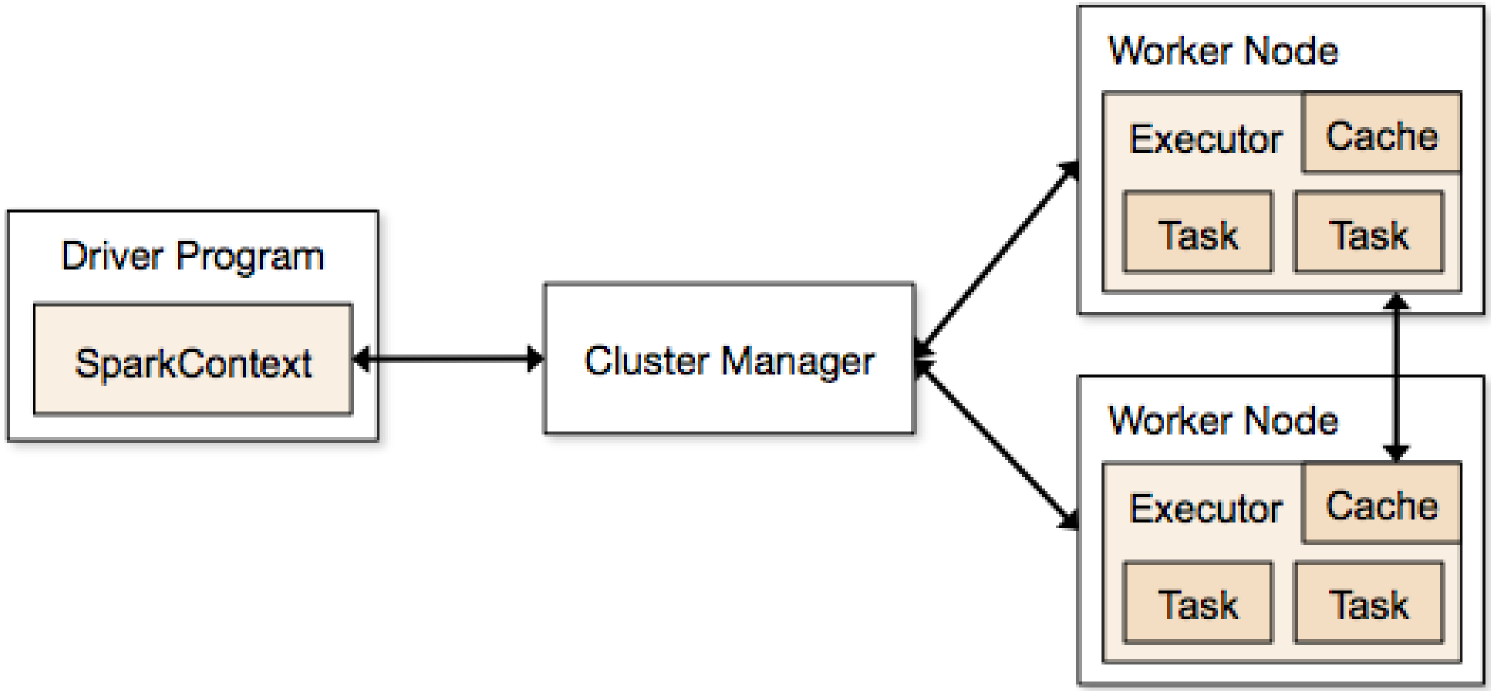
支援不同的語言使用

- ❑Java
- ❑Scala
- ❑Python
- ❑R (Spark 1.4.0 ↑)

Java
Scala
Spark
python
R

Spark叢集模式架構

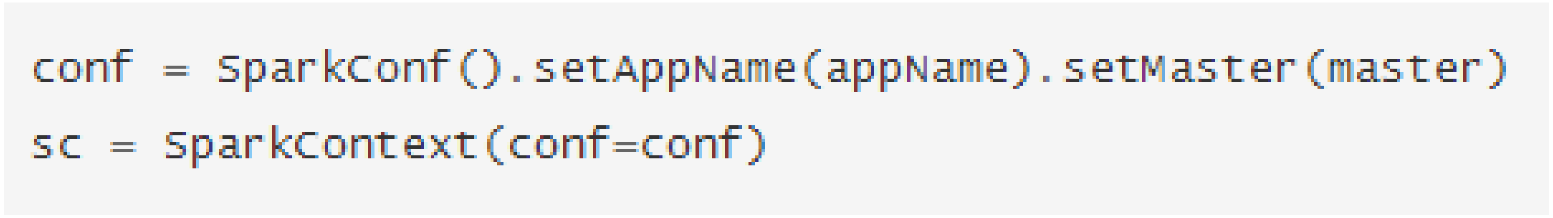
- ❑**Hadoop YARN**
- ❑**Standalone**
- ❑**Apache Mesos**



Reference : http://spark.apache.org/

SparkContext

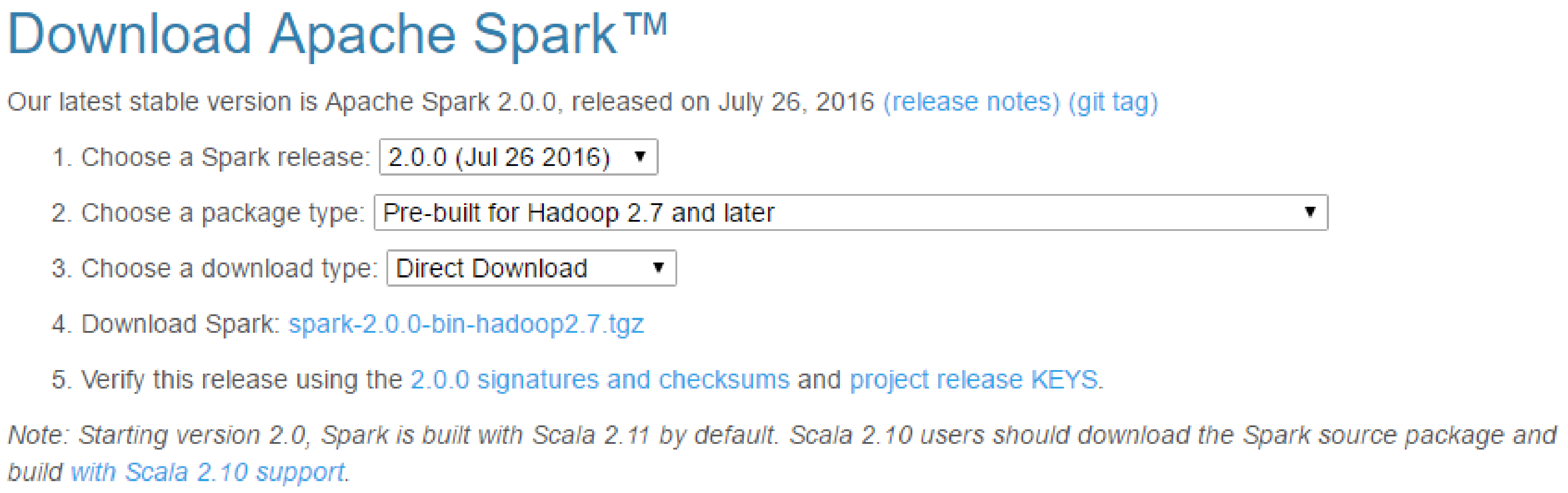
- ❑**Spark program must create a SparkContext object**
  - ❖**SparkContext is a entrance**
- ❑**Create SparkContext**



```
conf = SparkConf().setAppName(appName).setMaster(master)
sc = SparkContext(conf=conf)
```

下載Spark

1. http://spark.apache.org/downloads.html
2. Unzip tgz file


Download Apache Spark™

Our latest stable version is Apache Spark 2.0.0, released on July 26, 2016 (release notes) (git tag)

1. Choose a Spark release: 2.0.0 (Jul 26 2016)
2. Choose a package type: Pre-built for Hadoop 2.7 and later
3. Choose a download type: Direct Download
4. Download Spark: spark-2.0.0-bin-hadoop2.7.tgz
5. Verify this release using the 2.0.0 signatures and checksums and project release KEYS.

Note: Starting version 2.0, Spark is built with Scala 2.11 by default. Scala 2.10 users should download the Spark source package and build with Scala 2.10 support.

如何使用Spark

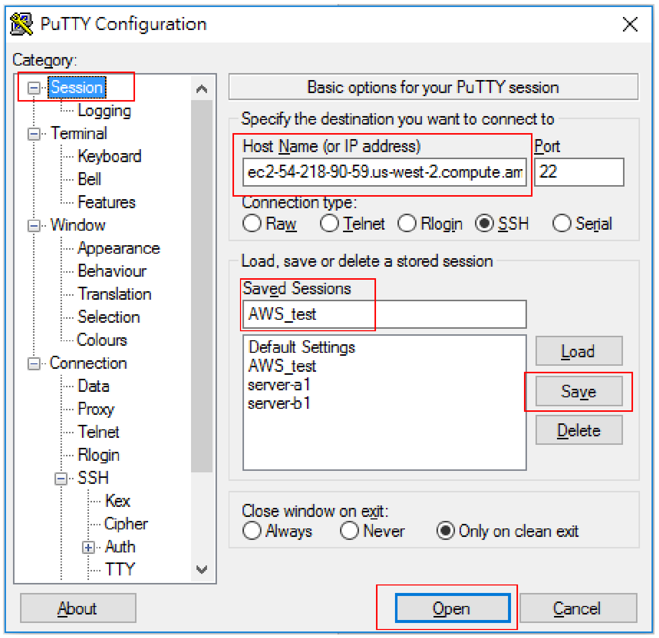
❑Spark CLI (with scala)

```
### Open Your Terminal
Putty SSH to AWS

### change to your spark path and start spark shell
$ cd /home/centos/spark/spark-2.0.0-bin-hadoop2.7/bin
$ ./spark-shell

### try yourself
scala> val input = sc.parallelize(Array(1,2,3))
scala> input.reduce((x,y) => x+y)
```

Is-land

Putty SSH to AWS

- ❑ **Open Putty**
  - ❖ **Session -> Host Name**
    - ▸ **ec2-54-218-90-59.us-west-2.compute.amazonaws.com(更改為你的hostname)**
  - ❖ **Connection -> SSH -> Auth -> Private key**
    - ▸ **Use Is-Land.ppk file (mac : ssh -i Is-Land.pem …)**
  - ❖ **Session -> Saved Sessions**
    - ▸ **Input "AWS_Session"**
    - ▸ **Click Save**
  - ❖ **Click Open**
  - ❖ **User : centos**


PuTTY Configuration

Category: Session, Logging, Terminal, Keyboard, Bell, Features, Window, Appearance, Behaviour, Translation, Selection, Colours, Connection, Data, Proxy, Telnet, Rlogin, SSH, Kex, Cipher, Auth, TTY

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): ec2-54-218-90-59.us-west-2.compute.am

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

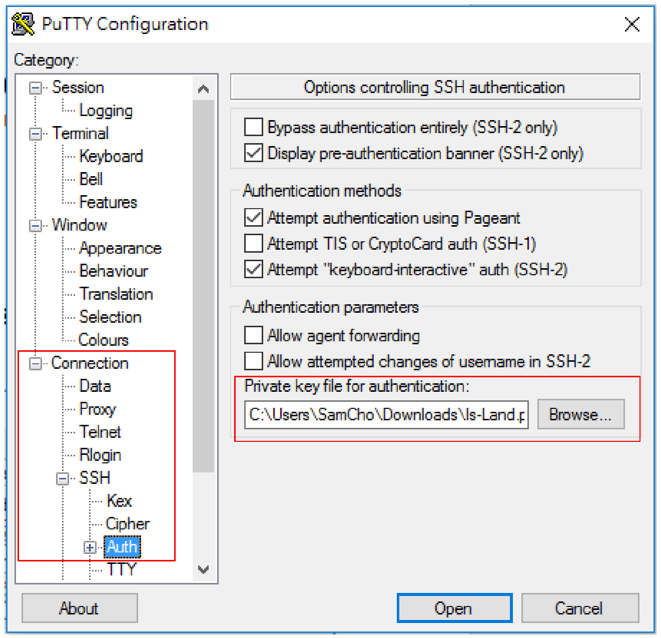
Saved Sessions: AWS_test

Default Settings
AWS_test
server-a1
server-b1

Load | Save | Delete

Close window on exit: Always Never [✓ Only on clean exit]

About | Open | Cancel


PuTTY Configuration

Category: Session, Logging, Terminal, Keyboard, Bell, Features, Window, Appearance, Behaviour, Translation, Selection, Colours, Connection, Data, Proxy, Telnet, Rlogin, SSH, Kex, Cipher, Auth, TTY

Options controlling SSH authentication

- ☐ Bypass authentication entirely (SSH-2 only)
- ☑ Display pre-authentication banner (SSH-2 only)

Authentication methods

- ☑ Attempt authentication using Pageant
- ☐ Attempt TIS or CryptoCard auth (SSH-1)
- ☑ Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

- ☐ Allow agent forwarding
- ☐ Allow attempted changes of username in SSH-2

Private key file for authentication: C:\Users\SamCho\Downloads\Is-Land.p

Browse...

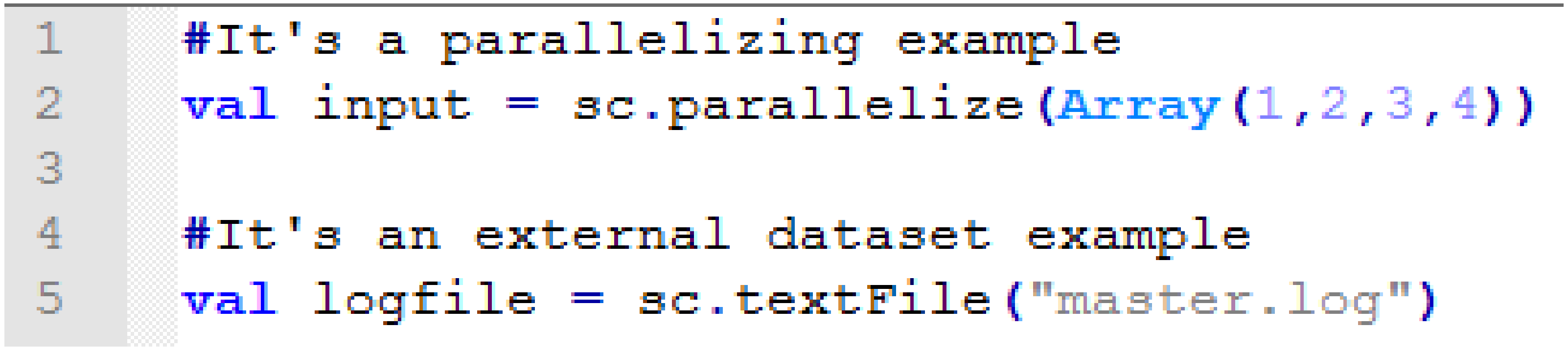
About | Open | Cancel

Spark基本操作物件 - RDD

- Resilient Distributed Datasets (RDDs)
  - Spark revolves around the concept of a resilient distributed dataset (RDD), which is a fault-tolerant collection of elements that can be operated on in parallel.
- Distributed collection of objects on anywhere
- RDDs are the fundamental unit of data in Spark
- Only hold references to objects

建立RDD

- ❑將資料載入到RDD
  - ❖Parallelizing
  - ❖External datasets

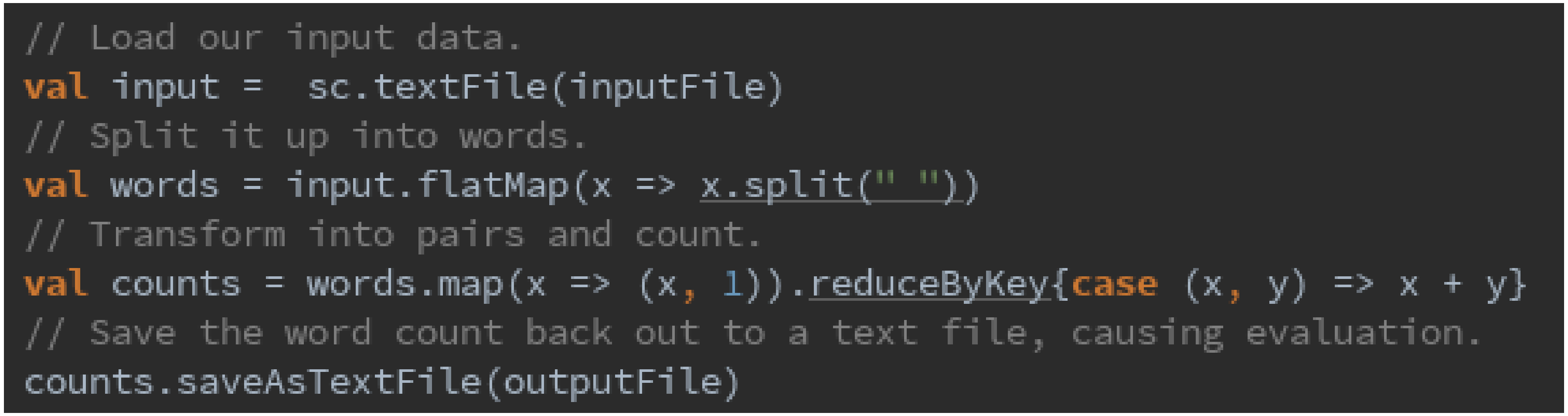


```
#It's a parallelizing example
val input = sc.parallelize(Array(1,2,3,4))

#It's an external dataset example
val logfile = sc.textFile("master.log")
```

RDD操作

❑RDD的操作方式

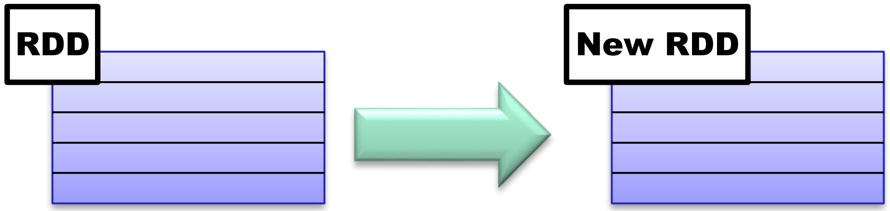
- ❖**Transformations**
- ❖**Actions**



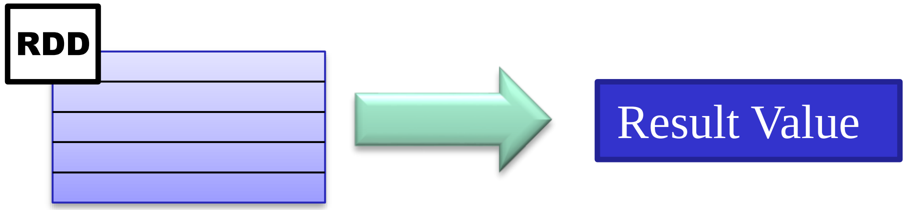
```scala
// Load our input data.
val input =  sc.textFile(inputFile)
// Split it up into words.
val words = input.flatMap(x => x.split(" "))
// Transform into pairs and count.
val counts = words.map(x => (x, 1)).reduceByKey{case (x, y) => x + y}
// Save the word count back out to a text file, causing evaluation.
counts.saveAsTextFile(outputFile)
```

Transformation與Action差別

- ❑ ***Transformation***
  - ❖ **operations on RDDs that return a new RDD**
  - ❖ **"Lazy Evaluation"**
    - ▸ **Transformed RDDs are computed lazily, only when you use them in an action**

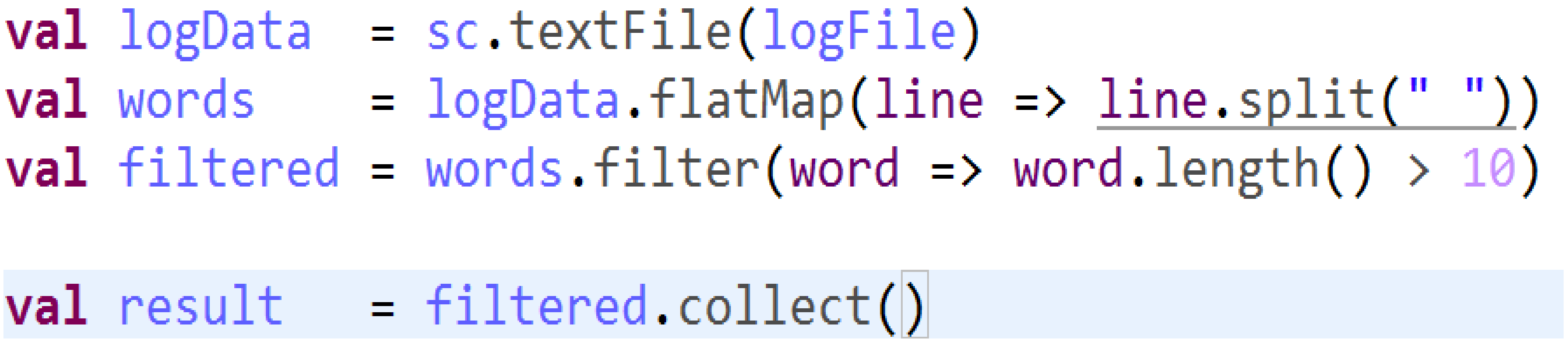


- ❑ **Action**
  - ❖ **Actually do something with dataset**
    - ▸ **Return a final value to the driver program**
    - ▸ **Write data to external storage system**



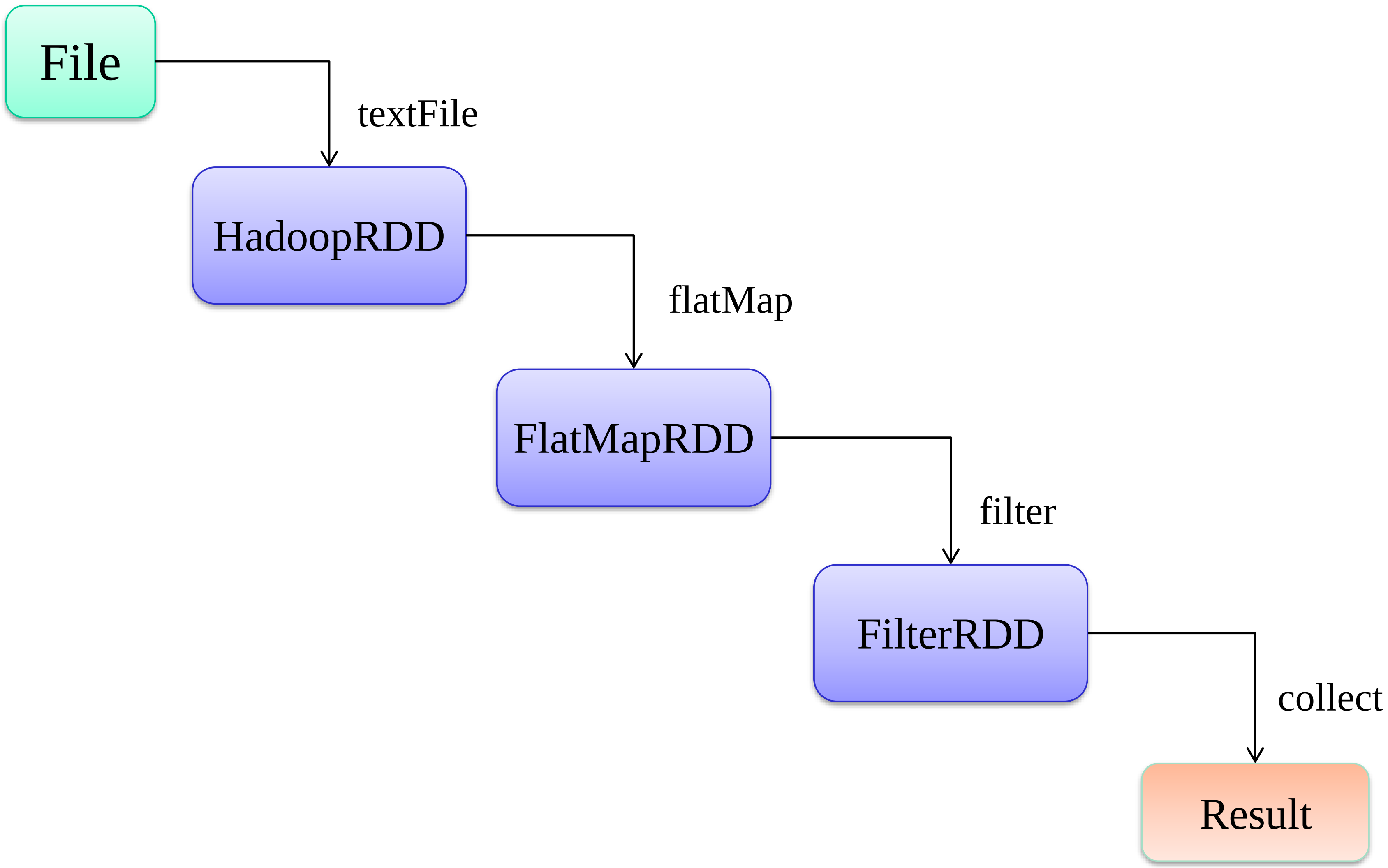
簡單的例子

- ❑ **What we done here**
  - ❖ **load file**
  - ❖ **split line into words by “blank”**
  - ❖ **filter words length > 10**
  - ❖ **print**



```
val logData  = sc.textFile(logFile)
val words    = logData.flatMap(line => line.split(" "))
val filtered = words.filter(word => word.length() > 10)

val result   = filtered.collect()
```

RDD DAG


File
textFile
HadoopRDD
flatMap
FlatMapRDD
filter
FilterRDD
collect
Result

Common RDD Operations

Transformations

- map
- flatMap
- filter
- distinct
- union
- **groupByKey**
- **reduceByKey**
- **join**

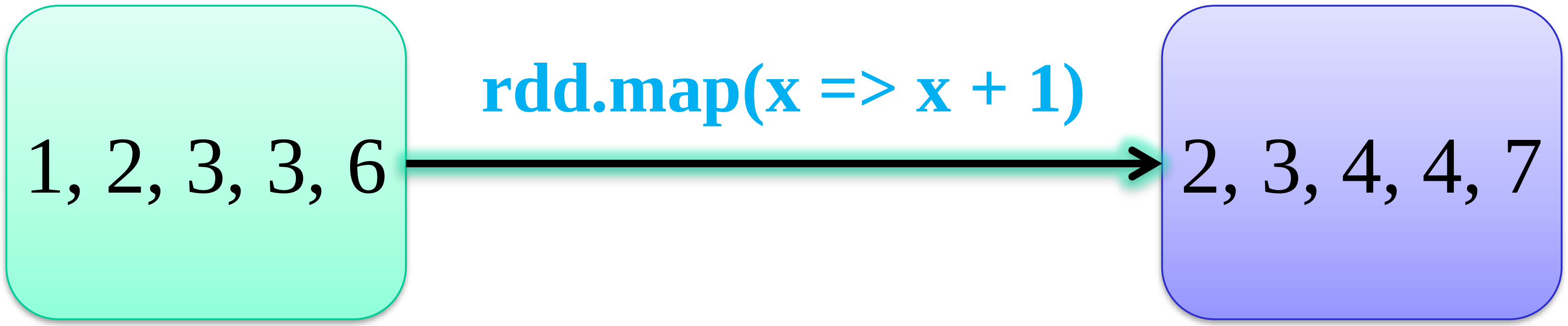
Actions

- collect
- count
- saveAsTextFile

參考：http://spark.apache.org/docs/latest/programming-guide.html

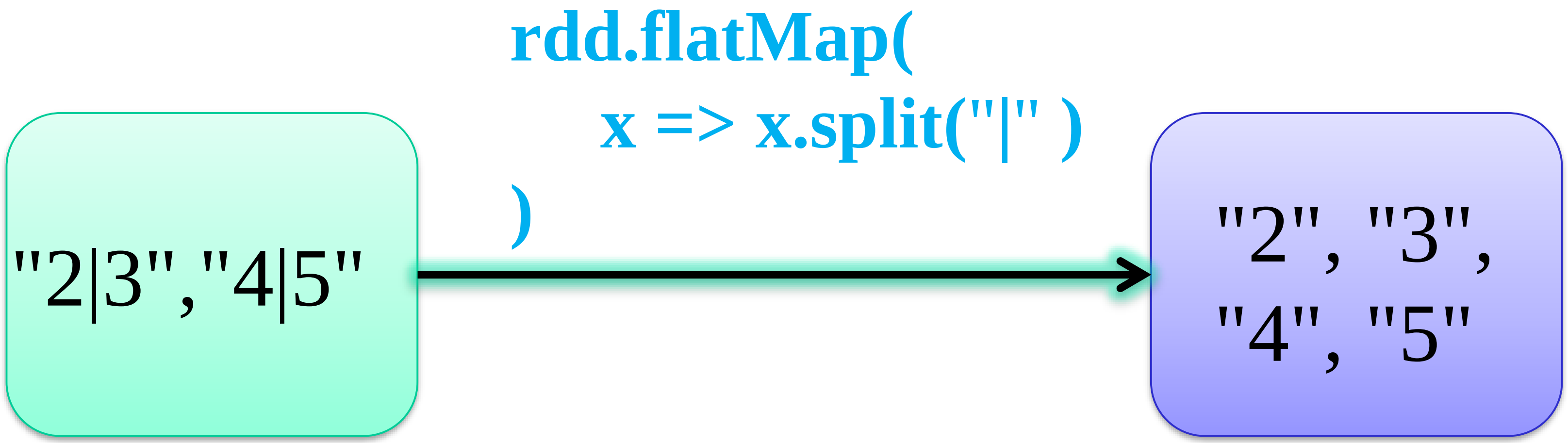
RDD.map()

- ❑**Return a new dataset formed by passing each element of the source through a function**



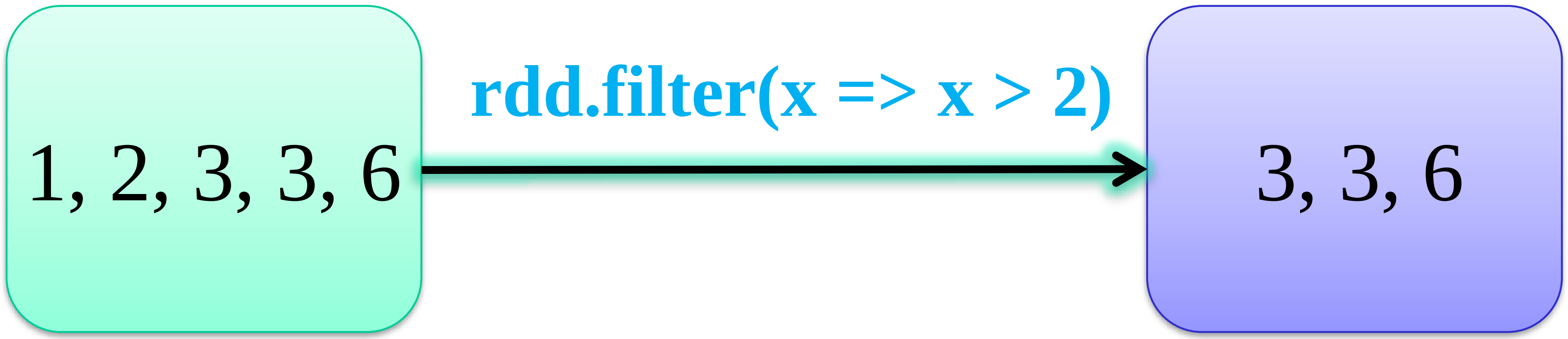
RDD.flatMap()

❑**Similar to map, but each input item can be mapped to 0 or more output items, but all data only in one collection**



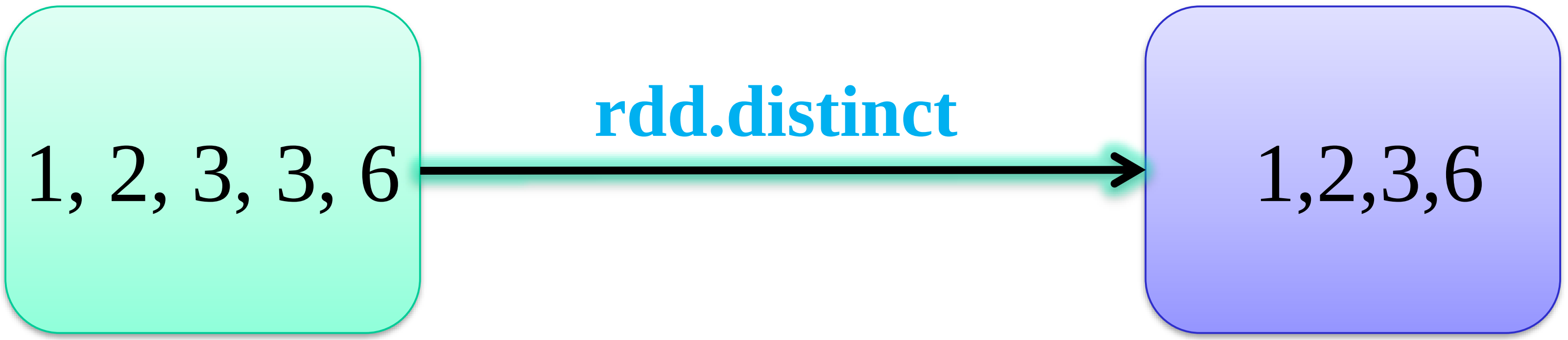
RDD.filter()

- Return a new dataset formed by selecting those elements of the source on which function returns true



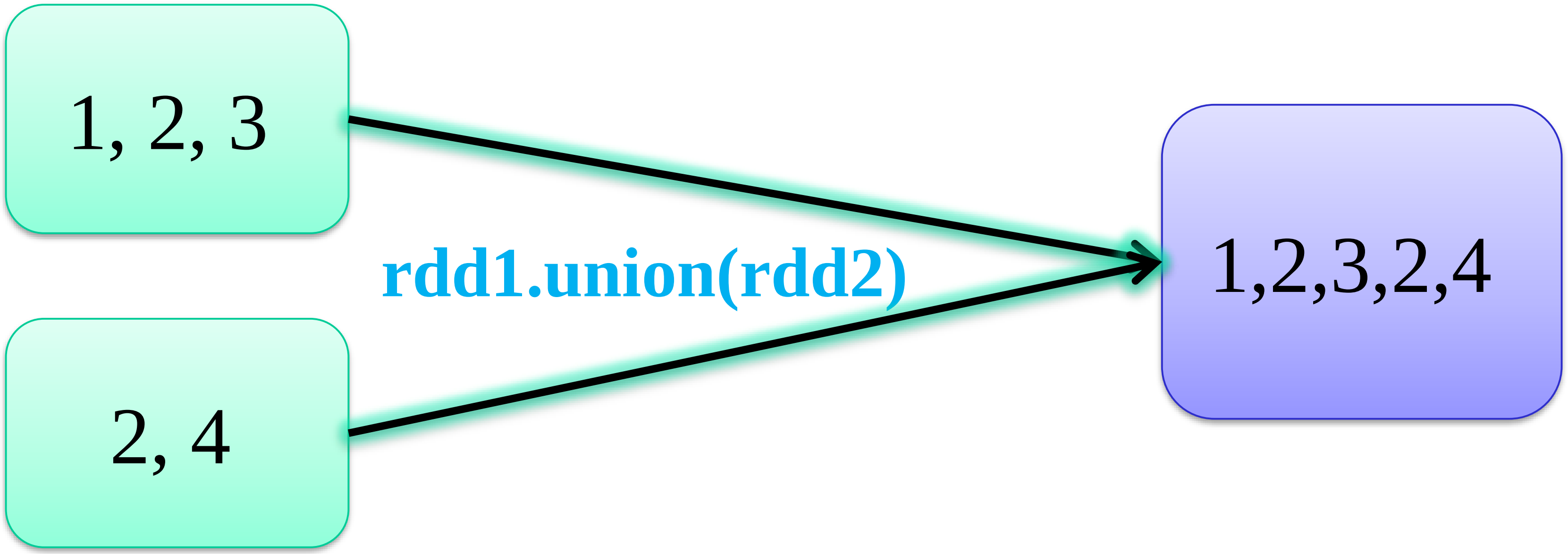
RDD.distinct()

- ❑Return a new RDD containing the distinct elements in this RDD



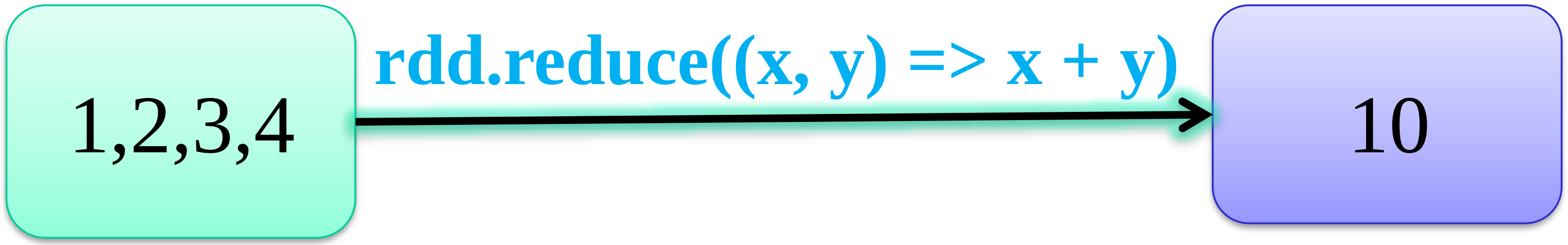
RDD.union()

- Return a new dataset that contains the union of the elements in the source dataset and the argument.

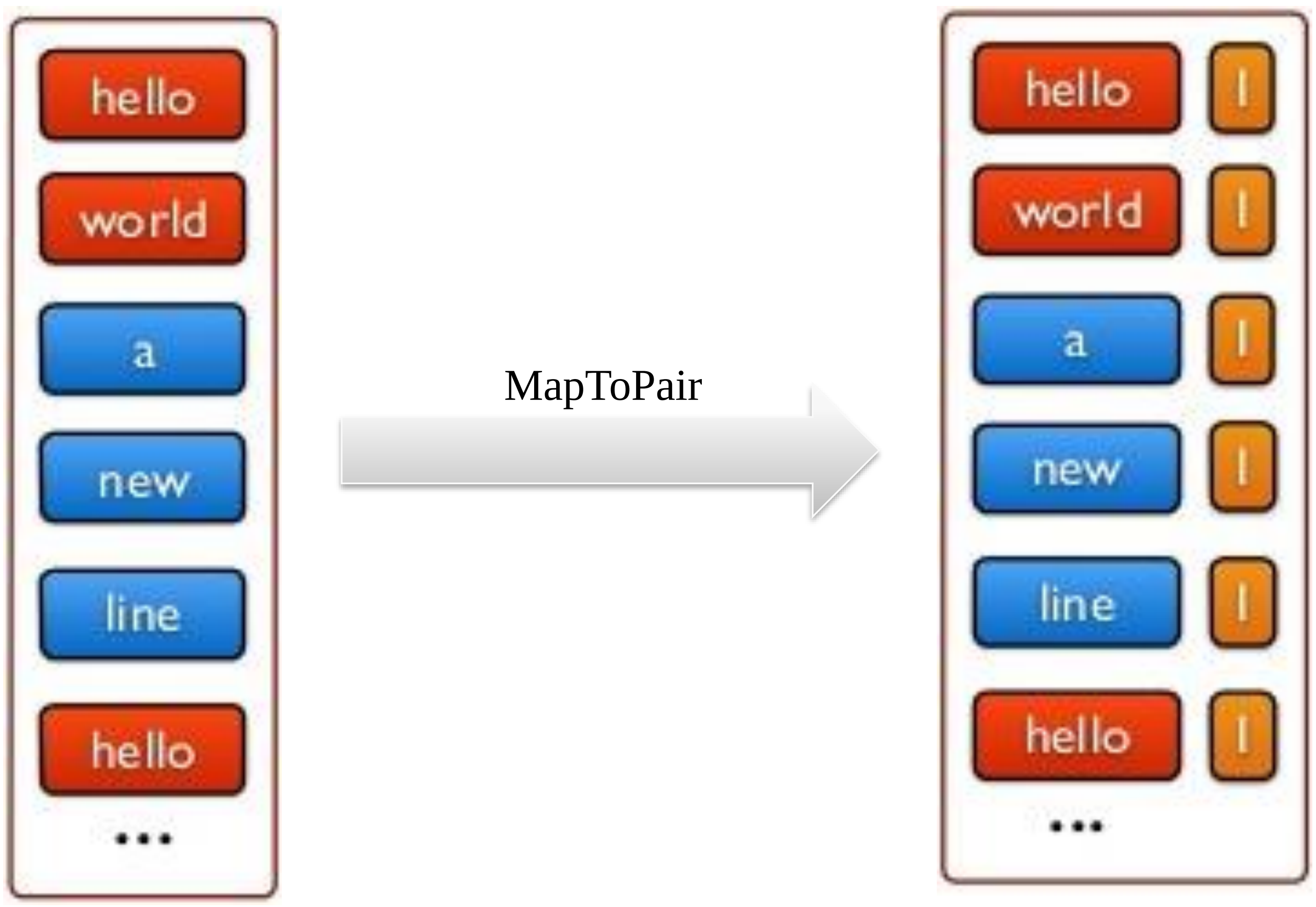


RDD.reduce()

- **Aggregate the elements of the dataset using a function. The function should be commutative and associative so that it can be computed correctly in parallel**

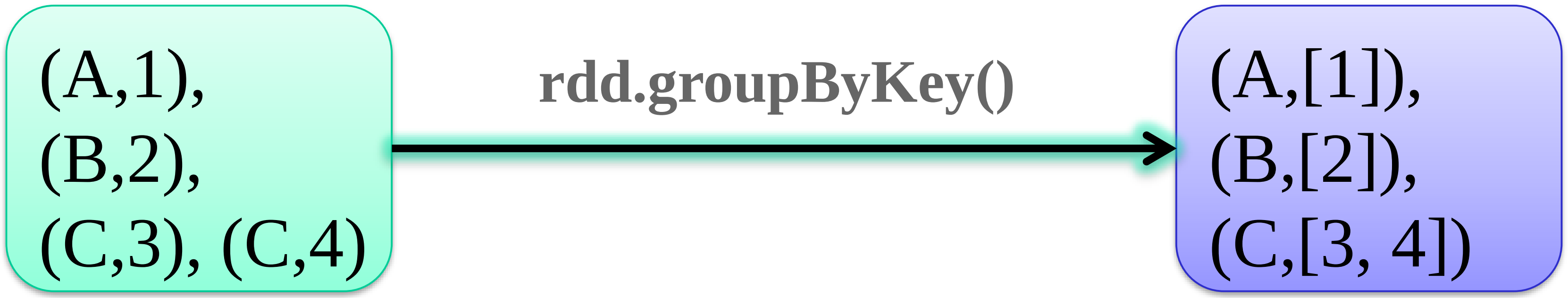


Key-Value觀念


hello
world
a
new
line
hello
...
MapToPair
hello 1
world 1
a 1
new 1
line 1
hello 1
...

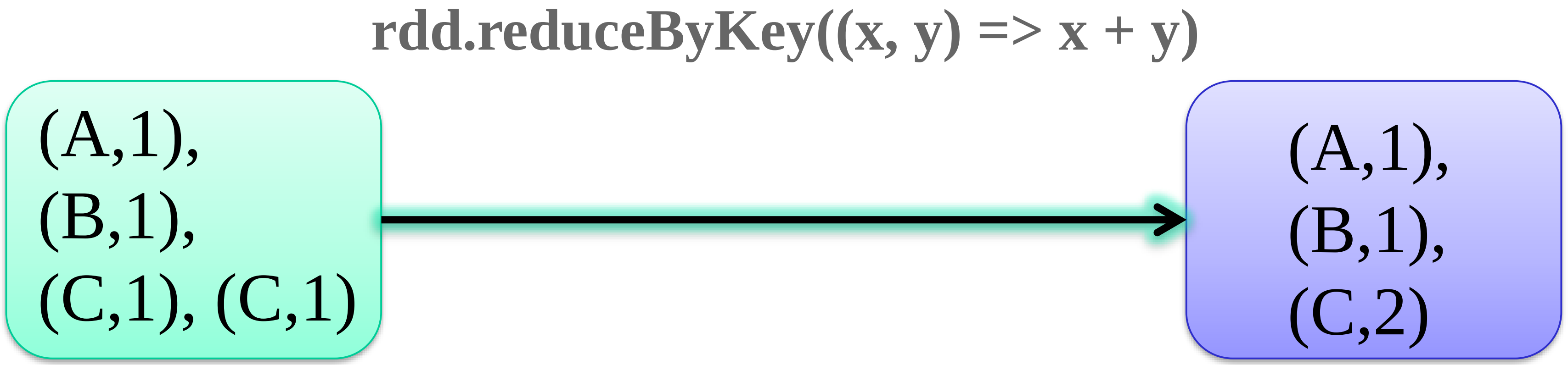
Pair RDD - RDD.groupByKey()

- **When called on a dataset of (K, V) pairs, returns a dataset of (K, Iterable<V>) pairs**



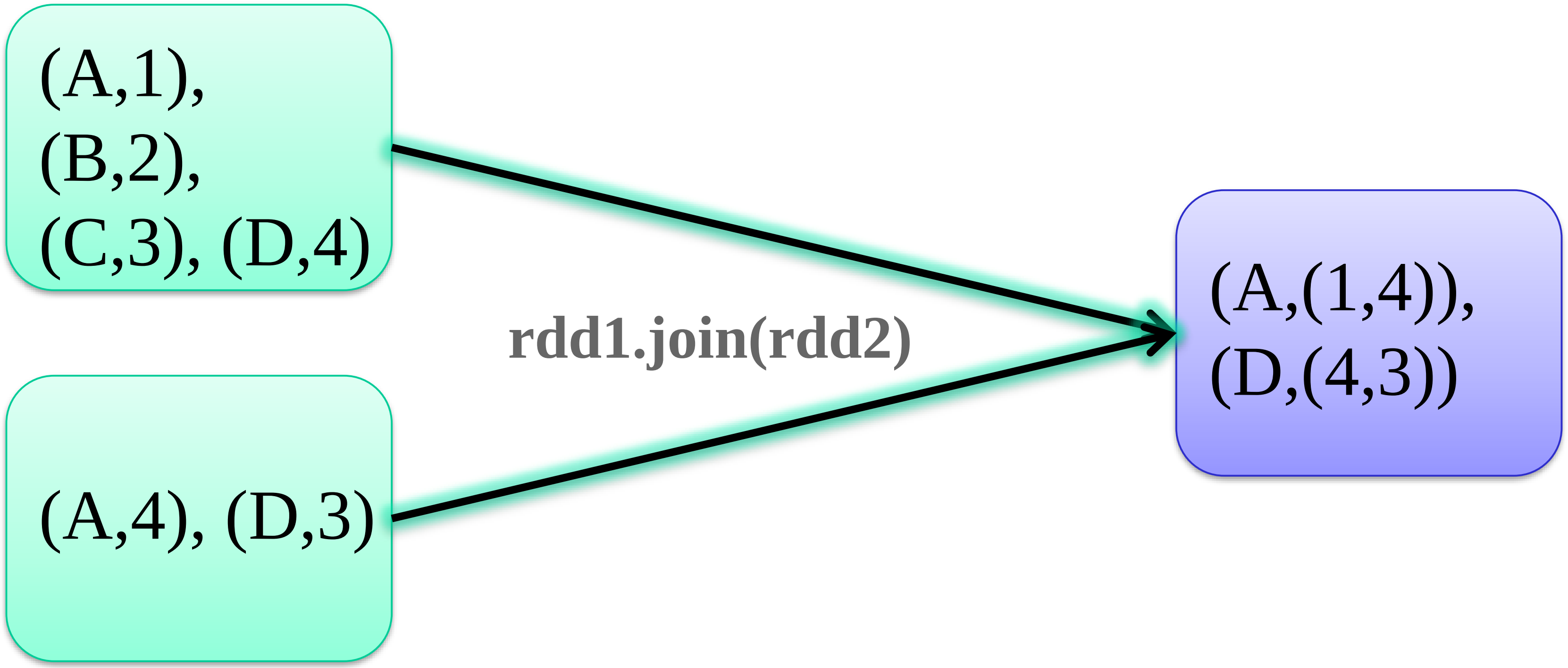
Pair RDD - RDD.reduceByKey()

- ❑Returns a dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function



Pair RDD - RDD.join()

- When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key



Actions

- collect - 把RDD的內容輸出到螢幕上
  - Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data
- count - 計算RDD的內容筆數
  - Return the number of elements in the dataset
- saveAsTextFile(path) - RDD存成檔案
  - Write the elements of the dataset as a text file (or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system. Spark will call toString on each element to convert it to a line of text in the file

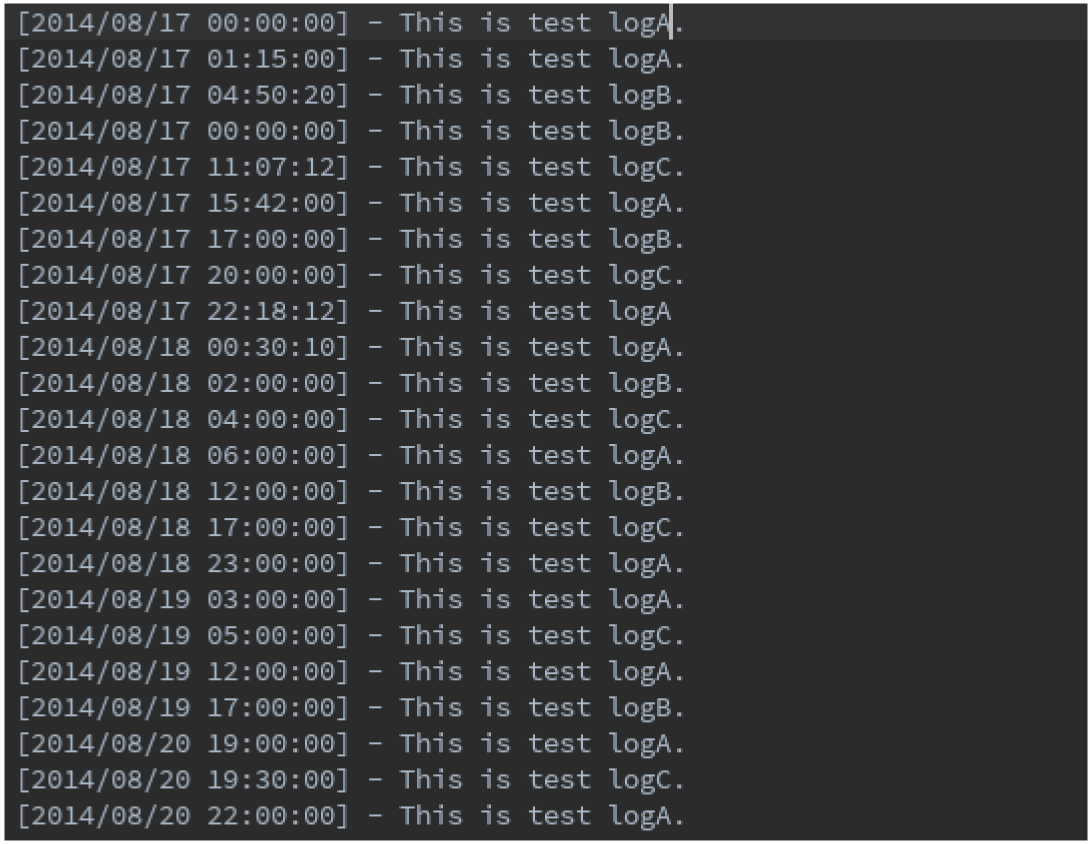
Hello World !

❑WordCount in Spark API (Use Scala)

```
val textFile = sc.textFile("/home/centos/spark/log.txt")
//split each document into words
val tokenized = textFile.flatMap(line => line.split(" "))
//count the occurrence of each word
val wordCounts = tokenized.map(word => (word,1))
val result = wordCounts.reduceByKey(_ + _)
```

實作練習

- ❑現在有一文字log檔，每行開頭標有紀錄該log的時間
  - ❖檔案：/home/centos/spark/log.txt
  - ❖目標：找出在8/17與8/20兩天，"logA"關鍵字出現的次數
    - ▸Hint : use "contain()" function



```
[2014/08/17 00:00:00] - This is test logA.
[2014/08/17 01:15:00] - This is test logA.
[2014/08/17 04:50:20] - This is test logB.
[2014/08/17 00:00:00] - This is test logB.
[2014/08/17 11:07:12] - This is test logC.
[2014/08/17 15:42:00] - This is test logA.
[2014/08/17 17:00:00] - This is test logB.
[2014/08/17 20:00:00] - This is test logC.
[2014/08/17 22:18:12] - This is test logA
[2014/08/18 00:30:10] - This is test logA.
[2014/08/18 02:00:00] - This is test logB.
[2014/08/18 04:00:00] - This is test logC.
[2014/08/18 06:00:00] - This is test logA.
[2014/08/18 12:00:00] - This is test logB.
[2014/08/18 17:00:00] - This is test logC.
[2014/08/18 23:00:00] - This is test logA.
[2014/08/19 03:00:00] - This is test logA.
[2014/08/19 05:00:00] - This is test logC.
[2014/08/19 12:00:00] - This is test logA.
[2014/08/19 17:00:00] - This is test logB.
[2014/08/20 19:00:00] - This is test logA.
[2014/08/20 19:30:00] - This is test logC.
[2014/08/20 22:00:00] - This is test logA.
```

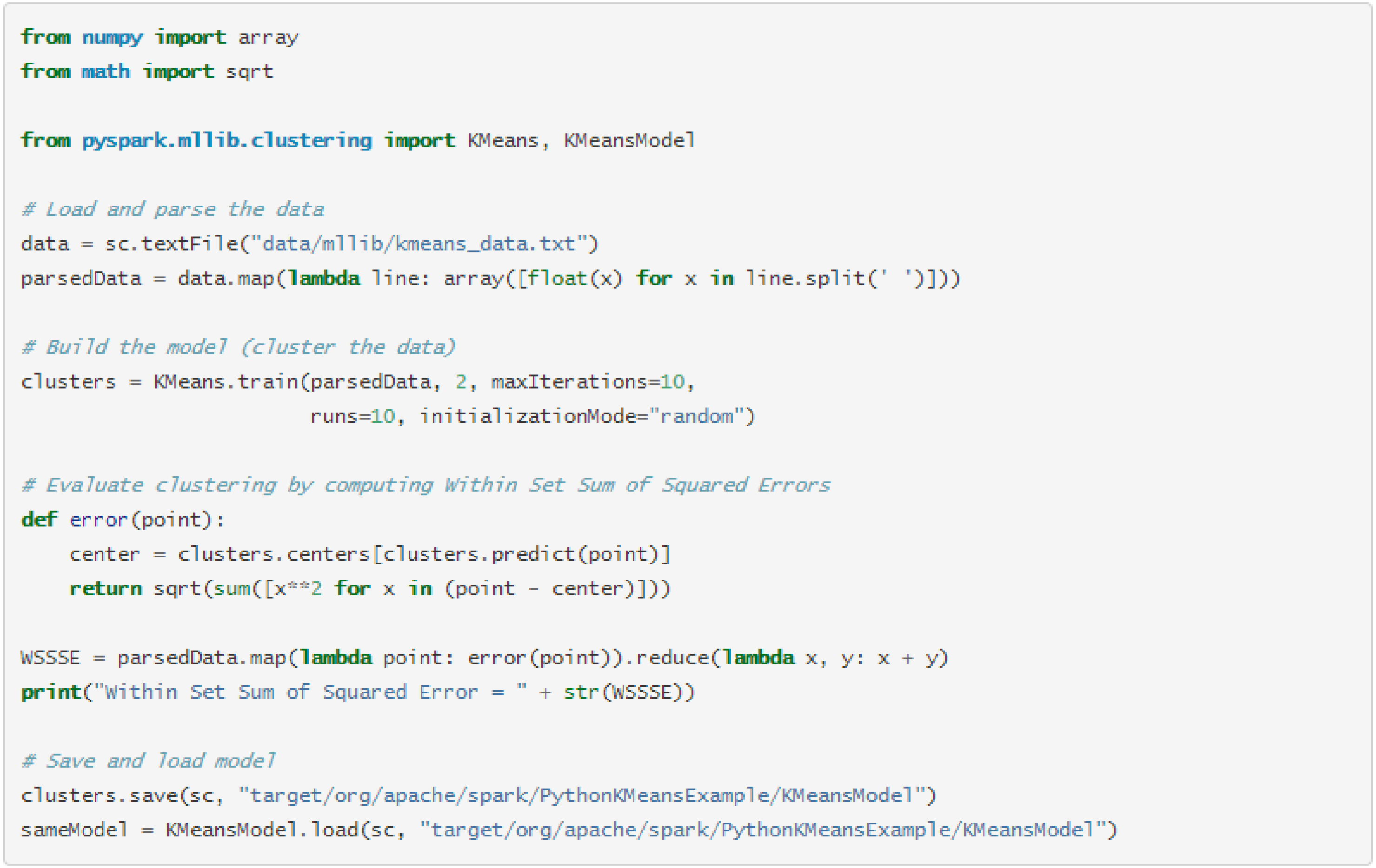
完整程式碼參考

❑利用Spark RDD

```
// you can save these content as a file, and run spark-shell then load the file
// scala> :load {FILE_PATH}

val inputFile = sc.textFile("/home/centos/spark/log.txt")
val dateString_1 = "2014/08/17"
val dateString_2 = "2014/08/20"
val keyWord = "logA"

// 作法 A : filter data with dateString condition, and use <"logA", 1> pair to calculate
val contain_1 = inputFile.filter(line => line.contains(dateString_1))
val contain_2 = inputFile.filter(line => line.contains(dateString_2))
val contains_word = contain_1.union(contain_2)
val words = contains_word.flatMap(line => line.split(" "))
val counts = words.filter(word => word.contains(keyWord)).map(word => (word,1)).reduceByKey(_ + _)
// output the result to screen
counts.collect()
// or, you can save result to file
// counts.saveAsTextFile("/home/centos/spark/output.txt")

// 作法 B : filter data with all condition, and use "count" Action to count
inputFile.filter(line => line.contains(dateString_1) | line.contains(dateString_2)).filter(_.contains(keyWord)).count
```

Spark MLlib

- ❑MLlib is Apache Spark's scalable machine learning library
- ❑KMeans example (python)



```python
from numpy import array
from math import sqrt

from pyspark.mllib.clustering import KMeans, KMeansModel

# Load and parse the data
data = sc.textFile("data/mllib/kmeans_data.txt")
parsedData = data.map(lambda line: array([float(x) for x in line.split(' ')]))

# Build the model (cluster the data)
clusters = KMeans.train(parsedData, 2, maxIterations=10,
                        runs=10, initializationMode="random")

# Evaluate clustering by computing Within Set Sum of Squared Errors
def error(point):
    center = clusters.centers[clusters.predict(point)]
    return sqrt(sum([x**2 for x in (point - center)]))

WSSSE = parsedData.map(lambda point: error(point)).reduce(lambda x, y: x + y)
print("Within Set Sum of Squared Error = " + str(WSSSE))

# Save and load model
clusters.save(sc, "target/org/apache/spark/PythonKMeansExample/KMeansModel")
sameModel = KMeansModel.load(sc, "target/org/apache/spark/PythonKMeansExample/KMeansModel")
```

reference: http://spark.apache.org/docs/latest/mllib-clustering.html

Spark SQL

- ❑Spark SQL brings native support for SQL to Spark and it focus a structured data processing
- ❑Manipulation DataFrames
  - ❖Distributed collection of data organized into named columns
  - ❖Conceptually equivalent to a table in a relational database
- ❑Features :
  - ❖Import relational data from external source
  - ❖Run SQL over imported data or RDD
  - ❖Easy write RDDs out to Hive or Parquet files

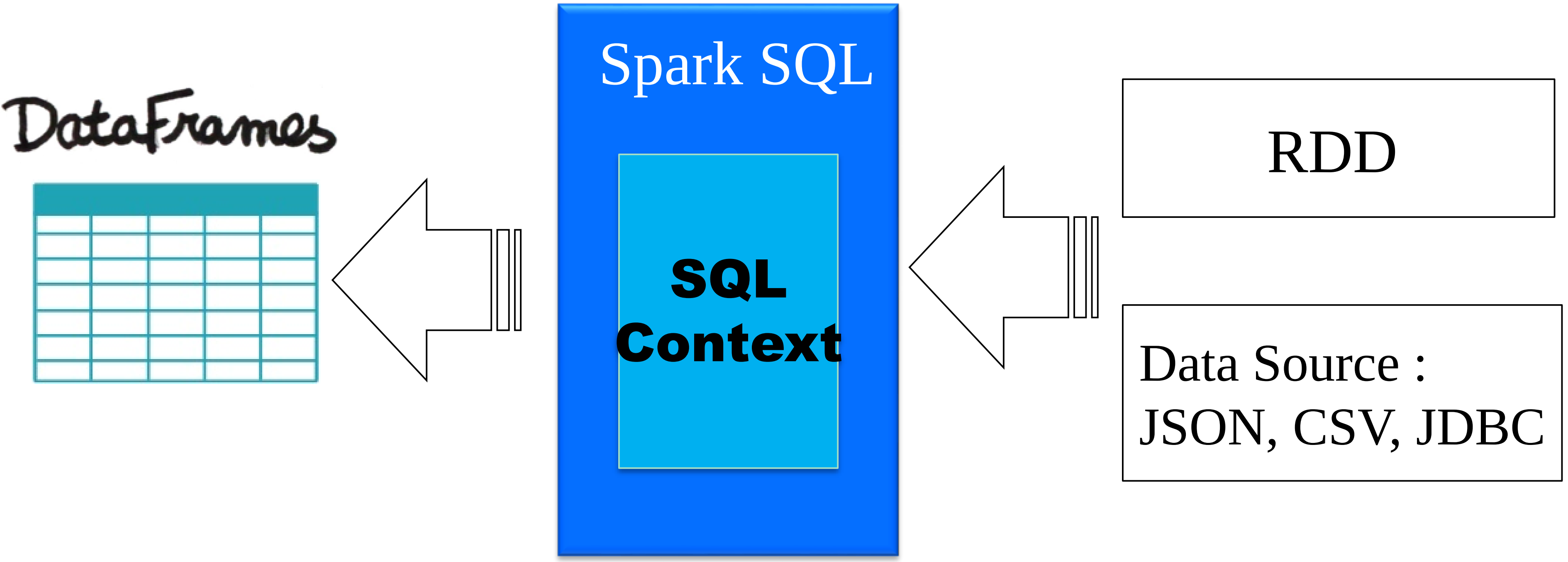
Interact with Spark SQL

- SQL
- DataFrames
  - Distributed collection of data organized into named columns
  - Conceptually equivalent to a table in a relational database
- Datasets

Initializing Spark SQL

- SQLContext

```
JavaSparkContext sc = ...; // An existing JavaSparkContext.
SQLContext sqlContext = new org.apache.spark.sql.SQLContext(sc);
```



SQL操作例子 (scala)

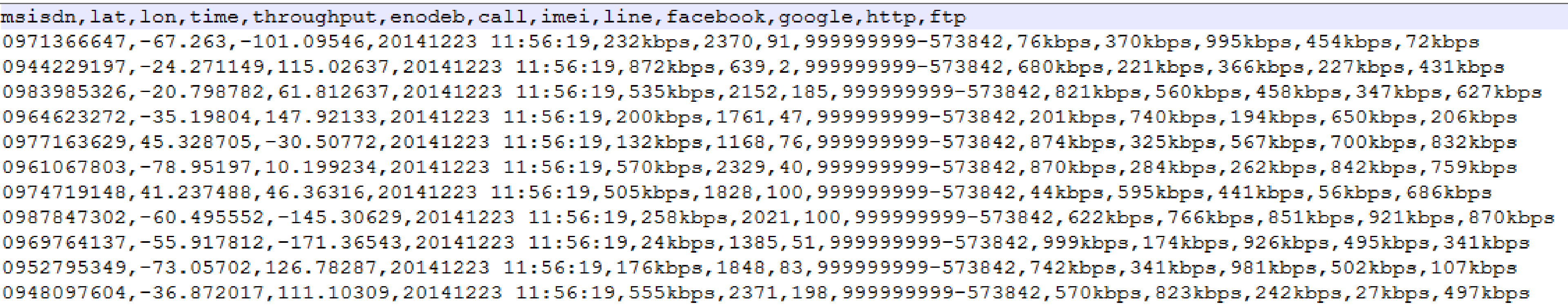
```
### Start your spark shell (ctrl + c could quit the terminal)
$ cd /home/centos/spark/spark-2.0.0-bin-hadoop2.7/bin
$ ./spark-shell

### Method A : Convert your input data RDD to DataFrame
scala> val input = sc.textFile("/home/centos/spark/log.txt")
scala> case class Log(datetime: String, info: String)
scala> val logtable = input.map(_.split(" ")).map(log => Log(log(0),log(6))).toDF
### register "LogTable"
scala> logtable.registerTempTable("LogTable")
scala> sql("select * from LogTable").show

### Method B : Use csv class to convert csv file to Dataframe
scala> import org.apache.spark.sql.SQLContext
scala> val sqlContext = new SQLContext(sc)
scala> sqlContext.load("com.databricks.spark.csv", Map("path"->
   "/home/centos/spark/datafile.txt","header"->"true")).registerTempTable("Datafile")
scala> sqlContext.sql("select count(*) from Datafile").show
```

SQL操作例子 (python)

```
### Start your spark shell (ctrl + d will quit)
$ cd /home/centos/spark/spark-2.0.0-bin-hadoop2.7/bin
$ ./pyspark

### Method A : Convert your input data RDD to DataFrame
>>> from pyspark.sql import Row
>>> input = sc.textFile("/home/centos/spark/log.txt")
>>> parts = input.map(lambda line : line.split(" "))
>>> log = parts.map(lambda p : Row(datetime=p[0], info=p[6]))
>>> logTable = spark.createDataFrame(log)
### register "LogTable"
>>> logTable.registerTempTable("LogTable")
>>> spark.sql("select * from LogTable").show()

### Method B : Use csv class to convert csv file to Dataframe
>>> from pyspark.sql import SQLContext
>>> sqlContext.read.format("com.databricks.spark.csv")
.options(header="true").load("/home/centos/spark/datafile.txt").registerTempTable("Datafile")
>>> results = spark.sql("select count(*) from Datafile").show()
```

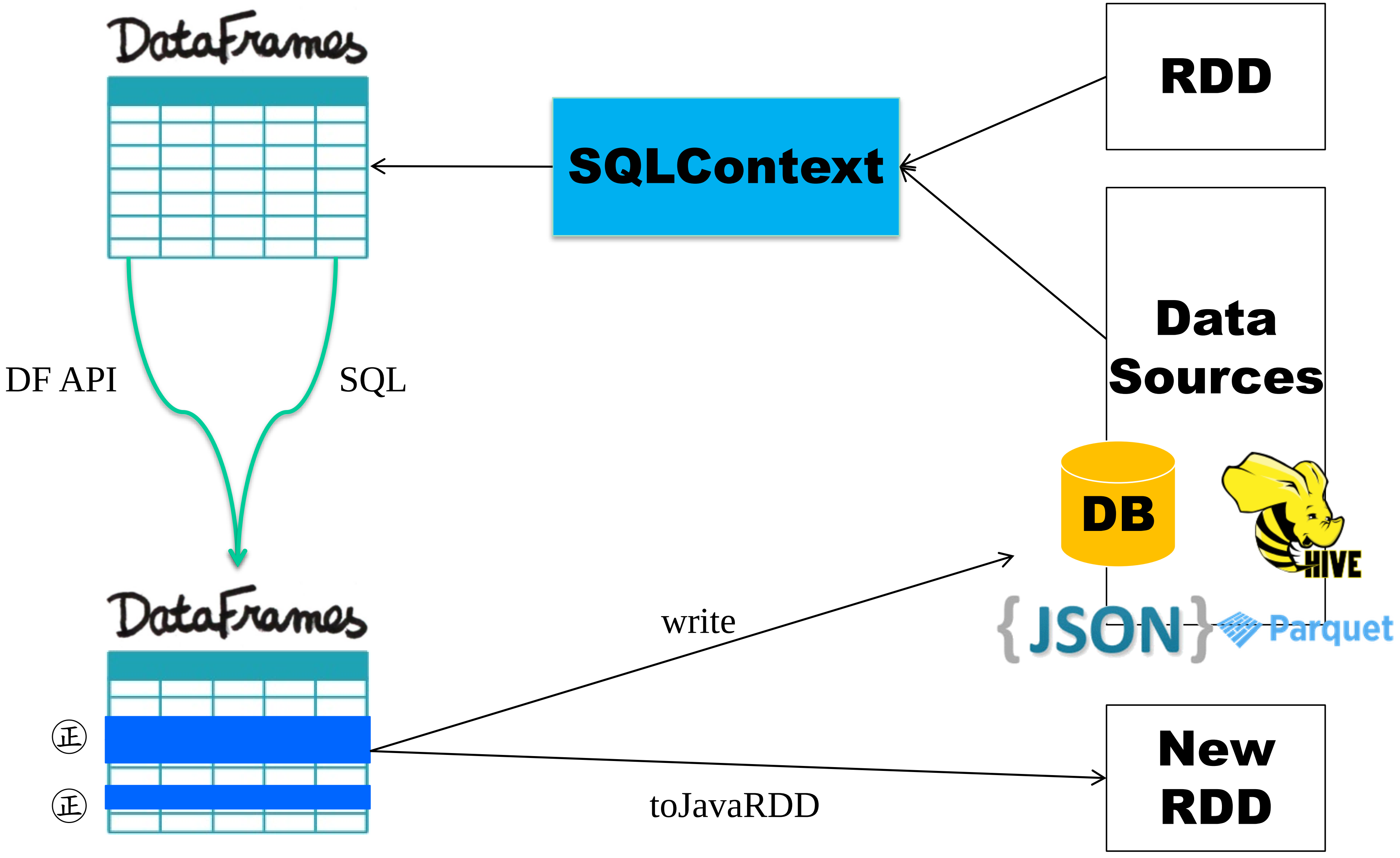
課後練習

- 請以datafile.txt資料分析
  - (電話號碼)msisdn=0987782022的line使用狀況
    -
  - facebook使用量在300kbps ~ 500kbps的人數
    -
  - throughput大於500kbps的總http使用量
    -



```
msisdn,lat,lon,time,throughput,enodeb,call,imei,line,facebook,google,http,ftp
0971366647,-67.263,-101.09546,20141223 11:56:19,232kbps,2370,91,999999999-573842,76kbps,370kbps,995kbps,454kbps,72kbps
0944229197,-24.271149,115.02637,20141223 11:56:19,872kbps,639,2,999999999-573842,680kbps,221kbps,366kbps,227kbps,431kbps
0983985326,-20.798782,61.812637,20141223 11:56:19,535kbps,2152,185,999999999-573842,821kbps,560kbps,458kbps,347kbps,627kbps
0964623272,-35.19804,147.92133,20141223 11:56:19,200kbps,1761,47,999999999-573842,201kbps,740kbps,194kbps,650kbps,206kbps
0977163629,45.328705,-30.50772,20141223 11:56:19,132kbps,1168,76,999999999-573842,874kbps,325kbps,567kbps,700kbps,832kbps
0961067803,-78.95197,10.199234,20141223 11:56:19,570kbps,2329,40,999999999-573842,870kbps,284kbps,262kbps,842kbps,759kbps
0974719148,41.237488,46.36316,20141223 11:56:19,505kbps,1828,100,999999999-573842,44kbps,595kbps,441kbps,56kbps,686kbps
0987847302,-60.495552,-145.30629,20141223 11:56:19,258kbps,2021,100,999999999-573842,622kbps,766kbps,851kbps,921kbps,870kbps
0969764137,-55.917812,-171.36543,20141223 11:56:19,24kbps,1385,51,999999999-573842,999kbps,174kbps,926kbps,495kbps,341kbps
0952795349,-73.05702,126.78287,20141223 11:56:19,176kbps,1848,83,999999999-573842,742kbps,341kbps,981kbps,502kbps,107kbps
0948097604,-36.872017,111.10309,20141223 11:56:19,555kbps,2371,198,999999999-573842,570kbps,823kbps,242kbps,27kbps,497kbps
```

That's figure it again



Q & A