CSCI-E63 Big Data Analytics

Section 5– Rahul Joglekar

CSCI-E63, Section 12, 05-02-2015

Learning Objectives …

1	Scala Fundamentals
2	Spark Refresher – RDD Concepts
3	Spark Deployments and Clusters
4	Scala/Spark IDE & Builds (sbt and mvn)

Administriva

- For all Late submissions please be patient, Homeworks are being graded.

- Document your Solutions in details , please see some showcased solutions

- Do NOT fall behind on the homeworks

Scala Programming

Scala

Scala and sbt local Install

Scala and sbt (simple build tool) installation

Mac -

```
$ brew update
$ brew install scala
$ brew install sbt
```

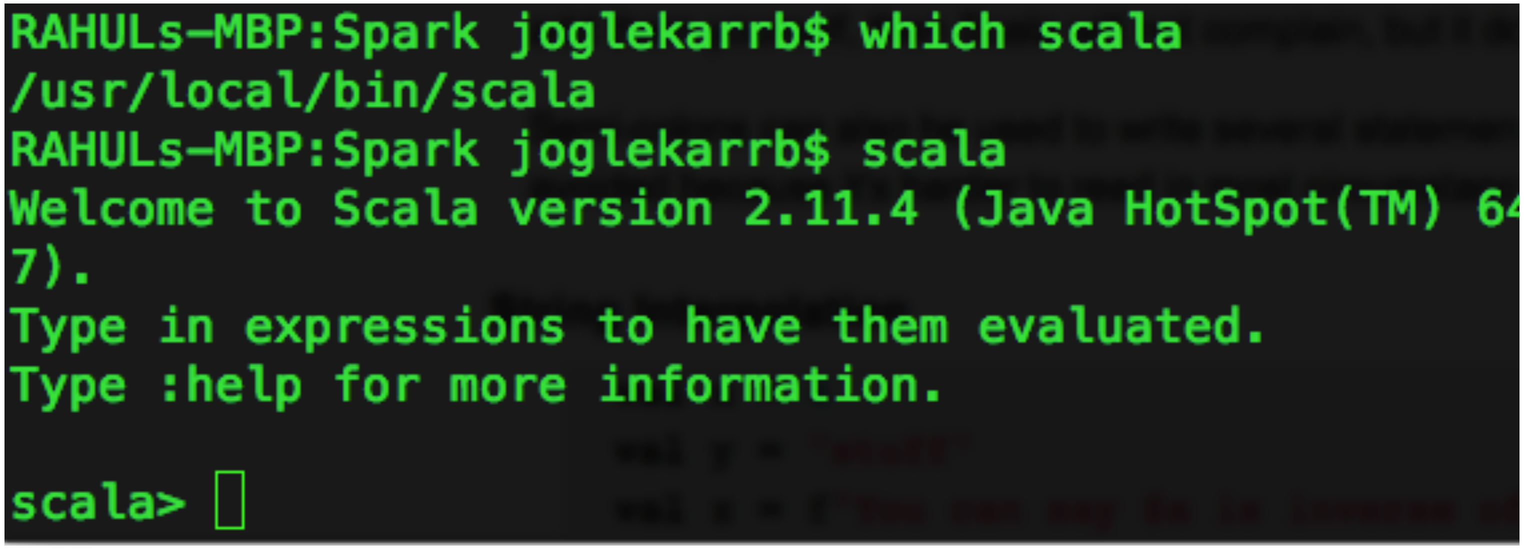
Unix -

Download Scala,Spark and sbt < follow instructions from the lecture>

Scala Basics

- Scala was designed from the beginning as a JVM language. Although it has many features which work differently than Java, and many other features which Java lacks entirely.
  - It compiles to JVM bytecode,
  - deploys as .class or .jar files,
  - runs on any standard JVM,
  - and interoperates with any existing Java classes.

- Scala includes a REPL ("read-eval-print-loop") where we can experiment and test out code. Type – scala to start the REPL



```
RAHULs-MBP:Spark joglekarrb$ which scala
/usr/local/bin/scala
RAHULs-MBP:Spark joglekarrb$ scala
Welcome to Scala version 2.11.4 (Java HotSpot(TM) 64
7).
Type in expressions to have them evaluated.
Type :help for more information.

scala> 
```

Scala Basics - Variables

There are two keywords for declaring variables: val and var.

- Identifiers declared with val cannot be reassigned; this is like a final variable in Java
- Identifiers declared with var may be reassigned.

```
val a = 1
var b = 2
b = 3 // fine
a = 4 // error: reassignment to val
```

- You should (pretty much) exclusively use val

Scala Basics - Style

- Class names start with a capital letter
- Variables and methods start with lowercase letters
- Constants start with capitals
- Everything uses camelCase

Scala - Types

Scala has powerful type inference capabilities:
• In many cases, types do not need to be specified
• However, types may be specified at any time

```
val a = 4          // a: Int = 4
val b: Int = 4     // b: Int = 4
```

This can make complex code more readable, or protect against errors. Types can be specified on any subexpression, not just on variable assignments.

```
val c = (a: Double) + 5          // c: Double = 9.0
```

All types are determined statically during compilation.

Common types include Int, Long, Double, Boolean, String, Char, Unit (“void”)
8

Scala – Collections

The most common collections are **Vector, List (similar to Vector), Map, and Set.**

Vector[T] is a sequence of items of type T. Elements can be accessed by 0-based index, using parens () as the subscript operator:

```
scala> val a = Vector(1,2,3)
a: Vector[Int] = Vector(1, 2, 3)
scala> a(0)
res0: Int = 1
```

Map[K,V] is an associative array or dictionary type mapping elements of type K to elements of type V. Values can be accessed through their keys.

```
scala> val a = Map(1 -> "one", 2 -> "two", 3 -> "three")
a: Map[Int,String] = Map(1 -> one, 2 -> two, 3 -> three)

scala> a(1)
res2: String = one
```

Scala Collections

Set[T] is an unordered collection of items of type T. Since it’s a set, no element can appear more than once. Since there is no order in a Set, elements cannot be accessed by index, but it is possible to check whether an element is in the set.

```
scala> val a = Set(1,2,3,2,3)
a: Set[Int] = Set(1, 2, 3)

scala> a(1)
res1: Boolean = true
```

Scala Imports

Classes, objects, and static methods can all be imported.

Underscore can be used as a wildcard to import everything from a particular context.

```
import scala.collection.immutable.BitSet
import scala.math.log
import scala.math._
```

Scala Immutability

Default collections are immutable: if you use a “write” operation on them, they return a new collection.

```
scala> val x = Vector(1,2)
x: scala.collection.immutable.Vector[Int] = Vector(1, 2)

scala> val y = x :+ 3
y: scala.collection.immutable.Vector[Int] = Vector(1, 2, 3)

scala> x eq y
res22: Boolean = false
```

However, mutable collections are also available:

```
scala> import scala.collection.mutable.ArrayBuffer
import scala.collection.mutable.ArrayBuffer
scala> val a = ArrayBuffer(1,2)
a: scala.collection.mutable.ArrayBuffer[Int] = ArrayBuffer(1, 2)
scala> a += 3
res46: a.type = ArrayBuffer(1, 2, 3) // a is a val; object is mutated in place
```

Scala Control Structures

If – else if --else

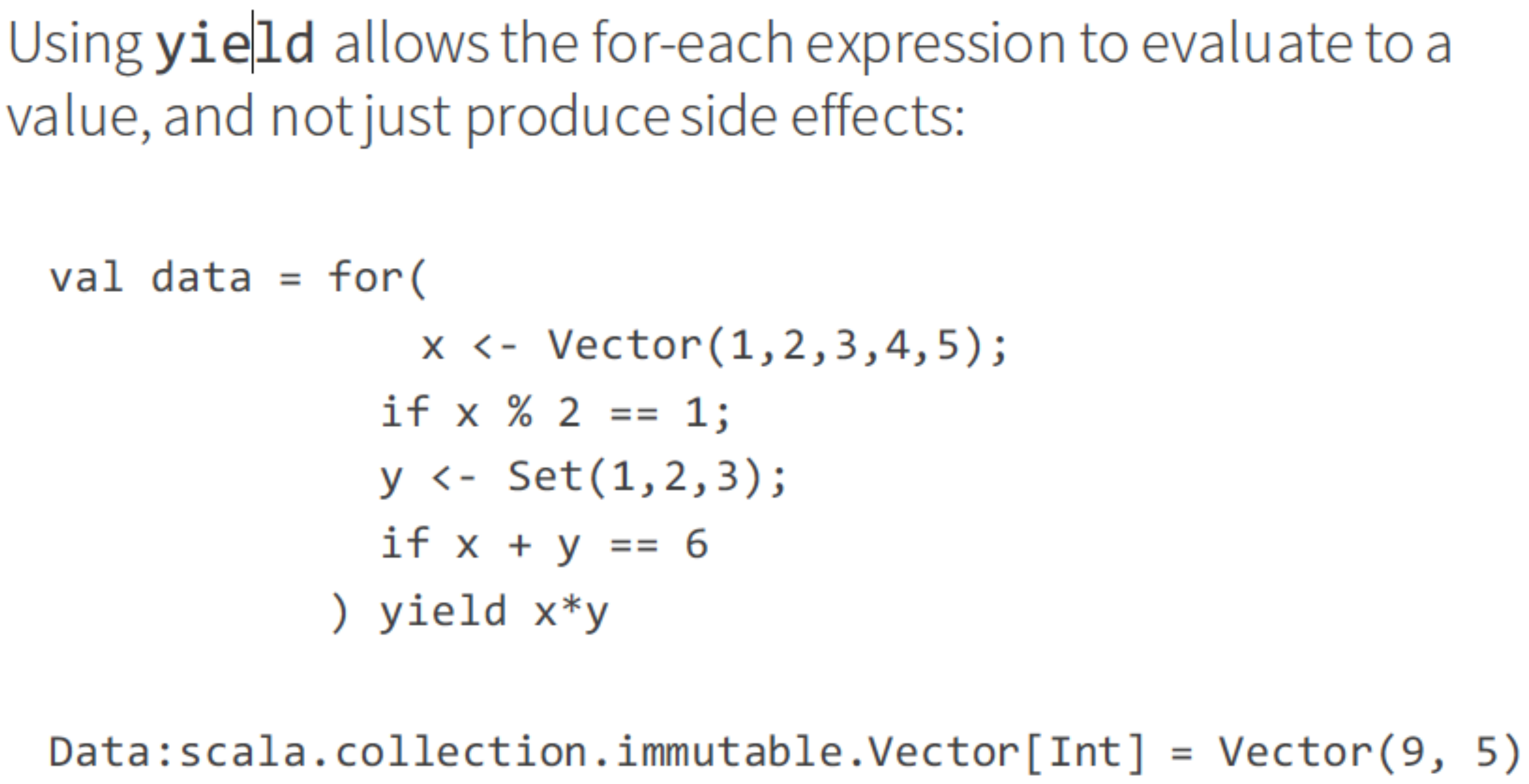
```
val x = 4
if(x > 2)
    println("greater than 2")
else if(x < 4)
    println("less than to 2")
else
    println("equal to 2")
// prints "greater than 2"
```

For Loop

```
for(
    x <- Vector(1,2,3,4,5);
          if x % 2 == 1; //filter out even xs
    y <-  Set(1,2,3); //inner loop over a list
          if x + y == 6
) println (s"x=$x, y=$y")
```

```
// prints:
// x=3, y=3
// x=5, y=1
```

Scala- For Comprehensions



Using `yield` allows the for-each expression to evaluate to a value, and not just produce side effects:

```
val data = for(
                x <- Vector(1,2,3,4,5);
              if x % 2 == 1;
              y <- Set(1,2,3);
              if x + y == 6
            ) yield x*y
```

```
Data:scala.collection.immutable.Vector[Int] = Vector(9, 5)
```

Scala Functions

Functions (often called methods) are defined using the def keyword.

1. Parameter types must be specified
2. Return types are optional: they can be inferred at compile-time
3. Function body should be separated from the signature by an equals sign (unless the return type is Unit)
4. Braces are not needed around a function body of a single expression
5. Parens are not needed in the function signature if there are no params
6. Empty parens means they are optional on the call
7. Function defined without parens, means they are not allowed, so the call looks like a variable access
8. The return keyword is not needed

```
def myadd( i: Int, j: Double) = i+j
```

Scala Functions

```
def mult(i: Int, j: Int): Int = i * j  // return type specified
def add(i: Int, j: Int) = i + j        // no braces needed
def mystring() = "something"           // parentheses option in caller
def mystring2 = "something else"       // no parentheses allowed in call
def doubleSum(i: Int, j: Int) = {      // braces for multiple statements
  val sum = i + j
  sum * 2                              // "return value"
}

mult(2,3)           // res55: Int = 5
add(2,3)            // res48: Int = 5
mystring()          // res50: String = something
mystring            // res51: String = something
mystring2           // res52: String = something else
doubleSum(2,3)      // res53: Int = 10
```

Scala – Function Objects and Lambda

Scala also supports function objects, which have types, and can be assigned:

```
scala> val add = (a:Int, b:Int) => a+b
add: (Int, Int) => Int = <function2>

scala> add(4,5)
res1: Int = 9
```

... and lambdas, or function expressions, which can be used inline without an assignment:

```
scala> List(3,4,5).map(n => n*n) // type is inferred
res3: List[Int] = List(9, 16, 25)
```

Scala Classes

- Classes can be declared using the class keyword.
- Methods are declared with the def keyword.
- Methods and fields are public by default, but can be specified as protected or private.
- Constructor arguments are, by default, private, but can be proceeded by val to be made public.

```
class A(i: Int, val j: Int) {
  val iPlus5 = i + 5
  private[this] val jPlus5 = j + 5 //instance privacy

  def addTo(k: Int) = new A(i + k, j + k)
  def sum = i + j
}
```

Scala Case Classes

Case classes are syntactic sugar for classes with a few methods pre-specified for convenience. These include toString, equals, and hashCode, as well as static methods apply (so that the new keyword is not needed for construction) and unapply (for pattern matching).

```
case class G(i: Int, j: Int) {
  def sum = i + j
}

val g = G(4, 5)    // g: G = G(4,5)
g.sum              // res19: Int = 9
g == G(4,5)        // res21: Boolean = true
```

Scala Tuples

Scala has Tuple types for 1 though 22 elements.

- In a tuple, each element has its own type,
- and each element can be accessed using the ._n syntax, where n is a 1-based index.

```
scala> val a = (1, "second", 3.4)
a: (Int, String, Double) = (1,second,3.4)

scala> a._2
res0: String = second
```

Scala Iterator

An Iterator[T] is a lazy sequence

- It only evaluates its elements once they are accessed
- Iterators can only be traversed one time

Accidentally traversing the same iterator more than once is a common source of bugs. If you want to be able to access the elements more than once, you can always call .toVector to load the entire thing into memory.

```
val a = Iterator(1,2,3)
val b = a.map(x => x + 1) // stage an operation, but don't traverse yet
val c = b.sum             // c: Int = 9
val d = b.mkString(" ")   // d: String = ""

val e = Iterator(1,2,3)
val f = e.map(x => x + 1) // stage an operation, but don't traverse yet
val g = f.toVector        // g: Vector[Int] = Vector(2, 3, 4)
val i = g.mkString(" ")   // i: String = "2 3 4"
```

Scala Pattern Matching

Allows for succinct code and can be used in a variety of situations.

- Many built-in types have pattern-matching behavior defined
- A main use of pattern matching is in match expressions
- Scala also supports conditional, wildcard, and recursive matching

```
val a = Vector(1,2,3)

val sum = a match {
  case Vector(x,y) => x + y
  case Vector(x,y,z) => x + y + z
}
// sum: Int = 6
```

Scala - Functional Programming

- One of the most important characteristics of functional programming is that functions are first-class members of the language. This means that they can be stored in variables and, more importantly, passed as arguments to other functions.
- To facilitate these kinds of uses, Scala has nice syntax for defining anonymous functions. In Scala, the symbol => is used to write lambda functions

```
val add1a = (x: Int) => x + 1
add1a(2)                        // res0: Int = 3
def addSome(f: (Int => Int), i: Int) = f(i) // first arg is a
function
addSome(x => x + 1, 2)                          // res2: Int = 3
addSome(add1a, 2)                               // res3: Int = 3
```

Scala - Functional Programming

Scala also provides the ability to write an underscore (_) as short-hand for x => x (kind of).

```
val add2a: (Int => Int) = _ + 2         // function's type declared
val add2b: (Int => Int) = 2 + _         // function's type declared
add2a(2)                                // res4: Int = 4
add2b(2)                                 // res5: Int = 4
addSome( _+ 2, 2)                       // res6: Int = 4
addSome(add2a, 2)                       // res7: Int = 4
```

Scala - Functional Programming

map: Take a function as an argument and apply it to every element in the collection.

```
Vector(1,2,3).map(x => x + 2)  // same as...
Vector(1,2,3).map(_ + 2)       // res0: Vector[Int] = Vector(3, 4, 5)
```

flatMap: Map a function over the collection and flatten the result

```
Vector(1,2,3).flatMap(n => Vector.fill(n)(s"[$n]"))
// res2: Vector[String] = Vector([1], [2], [2], [3], [3], [3])
```

filter: Remove items for which the given predicate is false

```
Vector(1,2,3).filter(x => x % 2 == 1)   // same as...
Vector(1,2,3).filter(_ % 2 == 1)   // res4: Vector[Int] = Vector(1, 3)
```

Scala Program

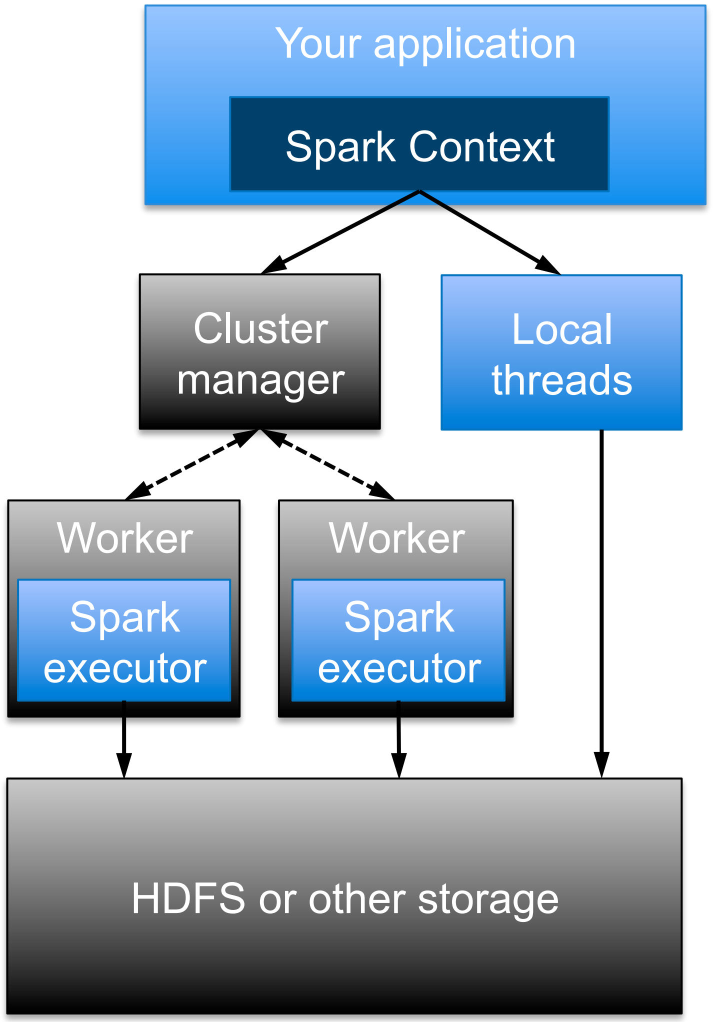
```
import scala.math.random
import org.apache.spark._

object SparkPi {

def main(args: Array[String]) {

        val conf = new SparkConf().setAppName("Spark Pi”).setMaster("local”)
        val spark = new SparkContext(conf)
        val slices = if (args.length > 0) args(0).toInt else 2
        val n = math.min(100000L * slices, Int.MaxValue).toInt // avoid overflow
        val count = spark.parallelize(1 until n, slices).map
                { i =>
                val x = random * 2 - 1
                val y = random * 2 - 1
                if (x*x + y*y < 1) 1
                else 0
                }.reduce(_ + _)

println("Pi is roughly " + 4.0 * count / n)
spark.stop()
}
```

SPAKH Refresha …

as they say in Boston

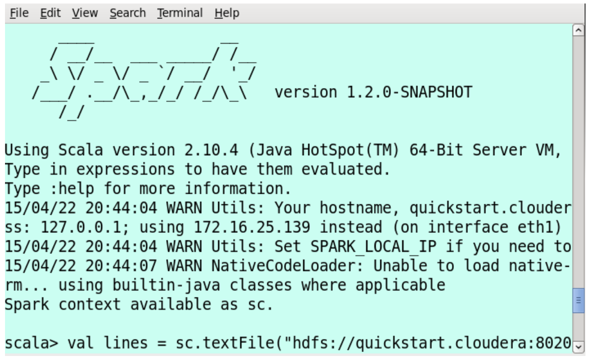
Spark - Software Components

- **Spark client is library in user program (1 instance per app)**
- **Runs tasks locally or on cluster**
  - Mesos, YARN, standalone mode
- **Accesses storage systems via Hadoop InputFormat API**
  - Can use HBase, HDFS, S3, …

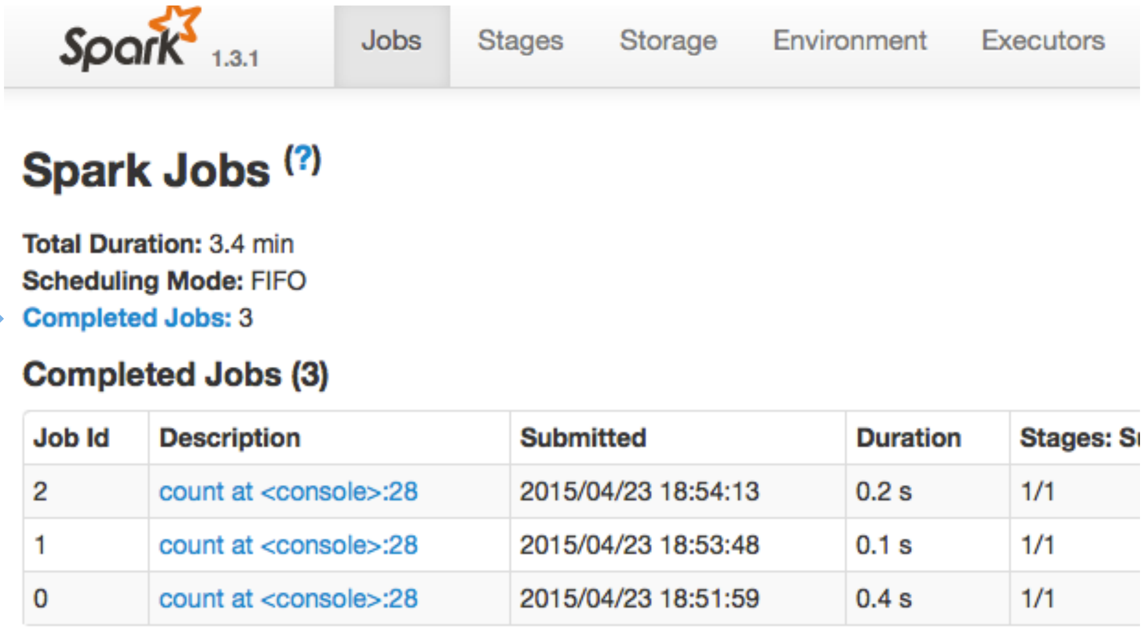


SPARK Programming Model

How does a user program get translated into units of physical execution: *jobs*, *stages*, and *tasks*



```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.2.0-SNAPSHOT
      /_/

Using Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM,
Type in expressions to have them evaluated.
Type :help for more information.
15/04/22 20:44:04 WARN Utils: Your hostname, quickstart.clouder
ss: 127.0.0.1; using 172.16.25.139 instead (on interface eth1)
15/04/22 20:44:04 WARN Utils: Set SPARK_LOCAL_IP if you need to
15/04/22 20:44:07 WARN NativeCodeLoader: Unable to load native-
rm... using builtin-java classes where applicable
Spark context available as sc.

scala> val lines = sc.textFile("hdfs://quickstart.cloudera:8020
```



Spark Jobs (?)

Total Duration: 3.4 min
Scheduling Mode: FIFO
Completed Jobs: 3

Completed Jobs (3)

Job Id	Description	Submitted	Duration	Stages: Su
2	count at <console>:28	2015/04/23 18:54:13	0.2 s	1/1
1	count at <console>:28	2015/04/23 18:53:48	0.1 s	1/1
0	count at <console>:28	2015/04/23 18:51:59	0.4 s	1/1

Life of a SPARK Job

What does a Spark Program Essentially do ?

1. Create some input RDDs from external data or parallelize a collection in your driver program.

```
val myRahulRDD = sc.textFile("hdfs://...")
```

2. Lazily transform them to define new RDDs using transformations like filter()or map()

```
val errorsRDD = myRahulRDD.filter(_.contains("ERROR"))
```

3. Ask Spark to cache() any intermediate RDDs that will need to be reused.

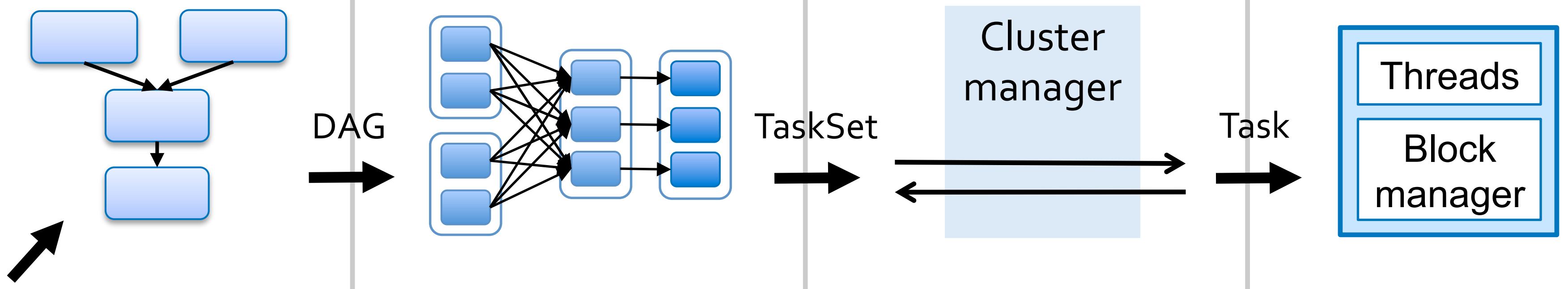
```
errorsRDD.cache()
```

4. Launch actions such as count()and collect()to kick off a parallel computation, which is then optimized and executed by Spark.

```
errorsRDD.count()
```

Job Scheduling Process

RDD Objects | DAGScheduler | TaskScheduler | Worker



```
rdd1.join(rdd2)
     .groupBy(…)
     .filter(…)
```

build operator DAG

split graph into *stages* of tasks

submit each stage as ready

agnostic to operators!

stage failed

launch tasks via cluster manager

retry failed or straggling tasks

doesn't know about stages

execute tasks

store and serve blocks

Spark Programming Model

- **Key idea: *resilient distributed datasets* *(RDDs)***
  - Distributed collections of objects that can be cached in memory across cluster
  - Manipulated through parallel operators
  - Automatically recomputed on failure

- **Programming interface**
  - Functional APIs in Scala, Java, Python
  - Interactive use from Scala shell

RDD’s - Refresher

Its all about the RDD’s

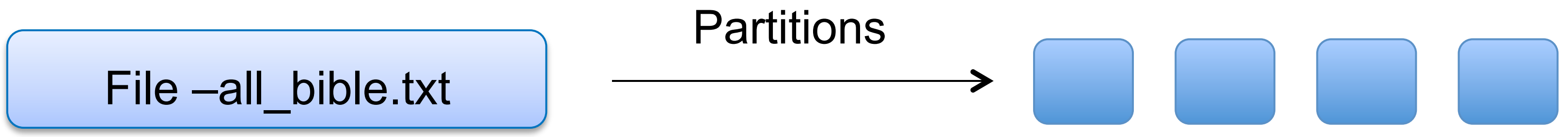
CREATE

TRANSFORM

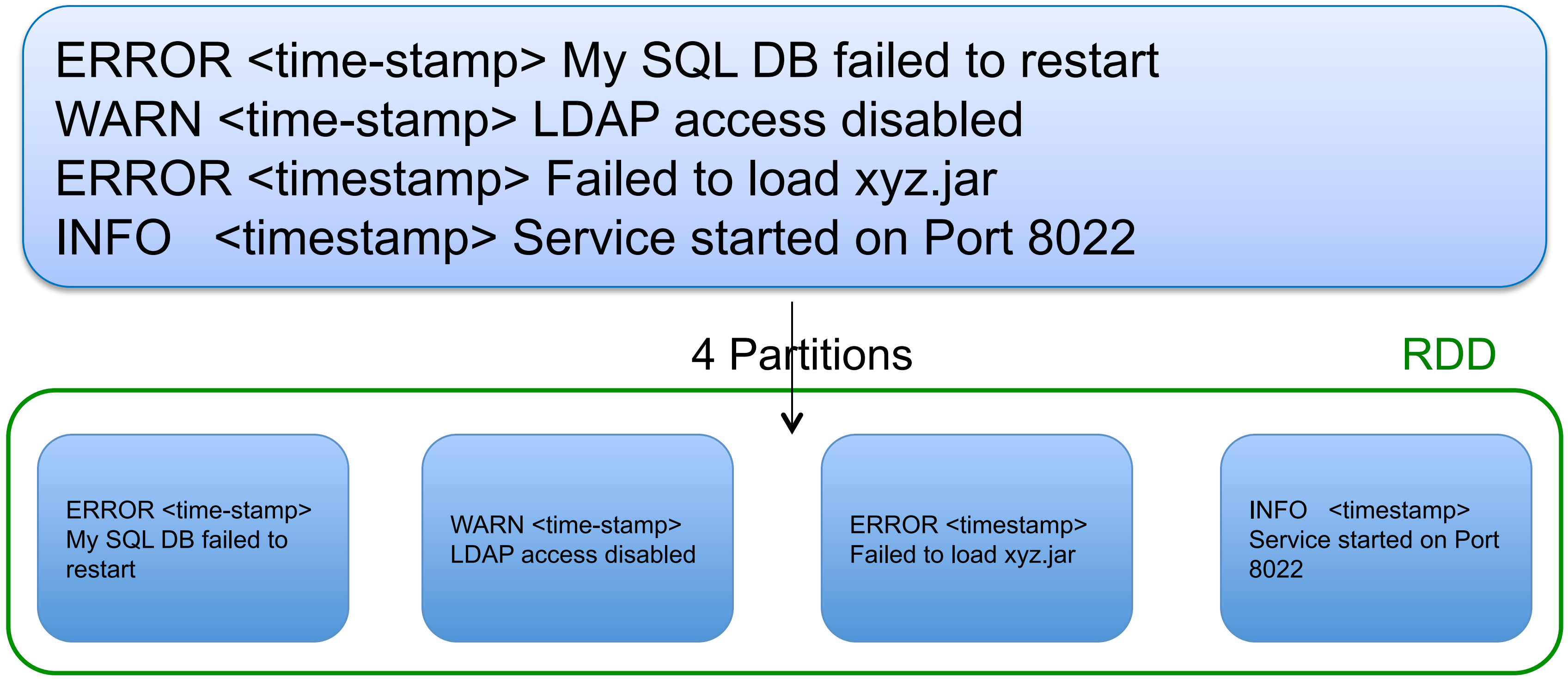
ACTION

File → RDD

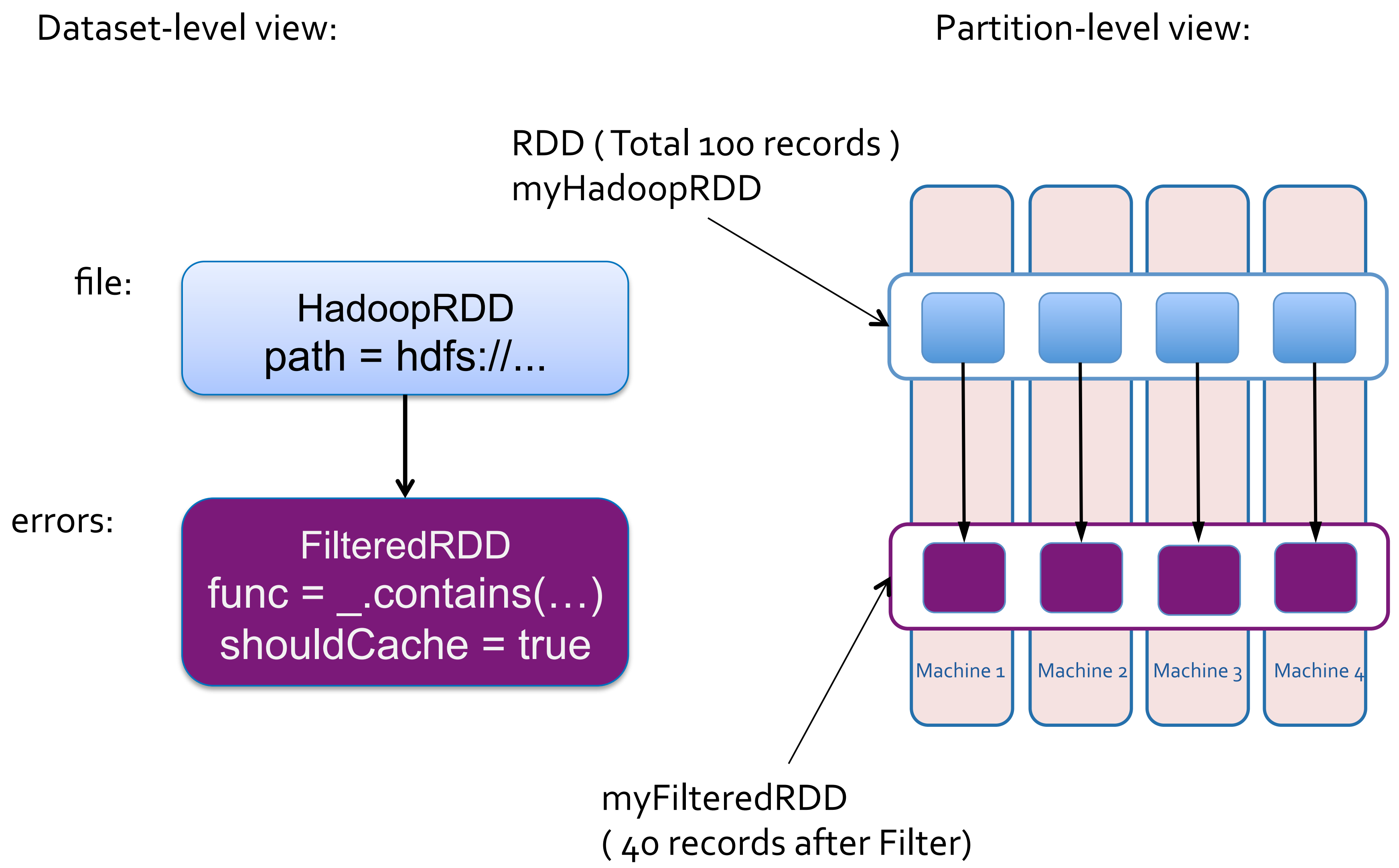
Example 1



Example 2


ERROR <time-stamp> My SQL DB failed to restart
WARN <time-stamp> LDAP access disabled
ERROR <timestamp> Failed to load xyz.jar
INFO <timestamp> Service started on Port 8022
4 Partitions
RDD
ERROR <time-stamp> My SQL DB failed to restart
WARN <time-stamp> LDAP access disabled
ERROR <timestamp> Failed to load xyz.jar
INFO <timestamp> Service started on Port 8022

RDD Dataset and Partition View

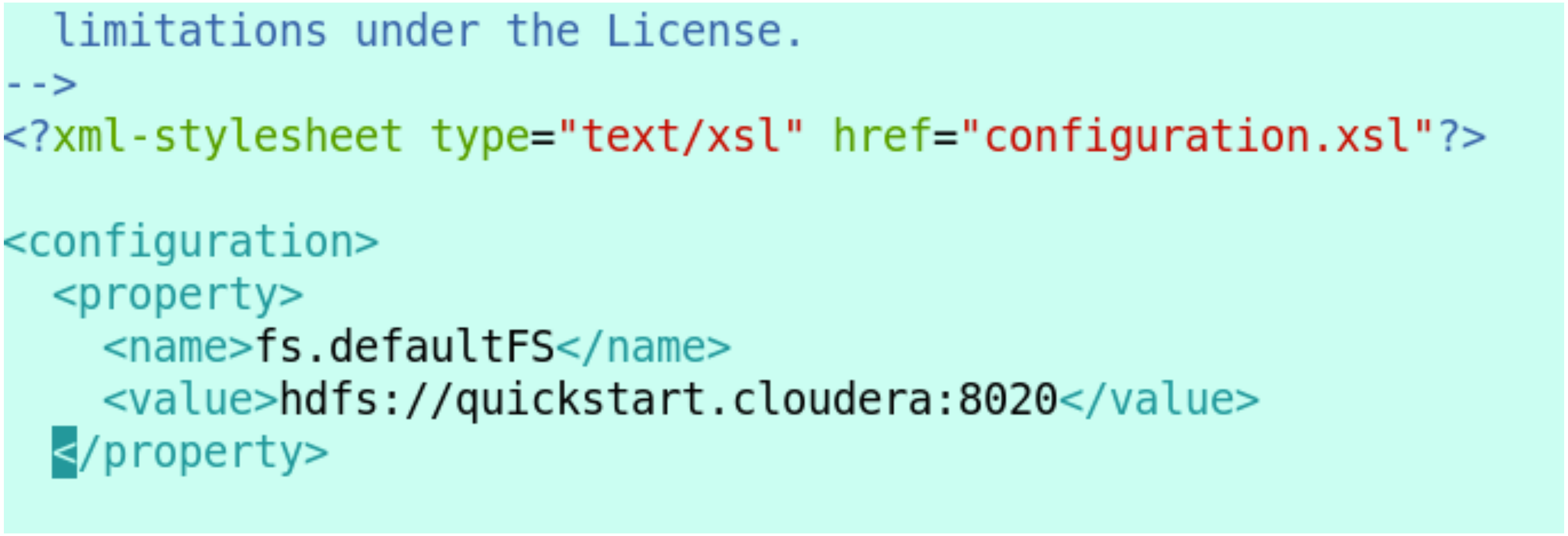


RDD’s - How do we “Create” them ?

```
# Turn a Python collection into an RDD
>sc.parallelize([1, 2, 3])

# Turn a Scala collection into an RDD
>sc.parallelize(List(1, 2, 3))

# Load text file from local FS, HDFS, or S3
>sc.textFile(“file.txt”)
>sc.textFile(“directory/*.txt”)
>sc.textFile(“hdfs://namenode:9000/path/file”)

# Use existing Hadoop InputFormat (Java/Scala only)
>sc.hadoopFile(keyClass, valClass, inputFmt, conf)
```

Create RDD's - Reading from HDFS

HDFS URL - /etc/hadoop/conf.pseudo/core-site.xml



```
  limitations under the License.
-->
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://quickstart.cloudera:8020</value>
  </property>
```

Base RDD

val HadoopLines= sc.textFile("hdfs://quickstart.cloudera:8020/users/cloudera/input/apachelog.txt")

RDD’s - Refresher

TRANSFORM

RDD’s – How do we “Transform” them ?

```
>val nums = sc.parallelize(List(1, 2, 3))

// Pass each element through a function
>val squares = nums.map(x: x*x)   // {1, 4, 9}

// Keep elements passing a predicate
>val even = squares.filter(x => x % 2 == 0) // {4}

// Map each element to zero or more others
>nums.flatMap(x => 0.to(x))
//=> {0, 1, 0, 1, 2, 0, 1, 2, 3}
```

RDD’s - Refresher

ACTION

RDD’s – What “Actions” can we take ?

```
>val nums = sc.parallelize(List(1, 2, 3))

// Retrieve RDD contents as a local collection
>nums.collect() //=> List(1, 2, 3)

// Return first K elements
>nums.take(2)   //=> List(1, 2)

// Count number of elements
>nums.count()   //=> 3

// Merge elements with an associative function
>nums.reduce{case (x, y) => x + y}  //=> 6

// Write elements to a text file
>nums.saveAsTextFile(“hdfs://file.txt”)
```

So what is an RDD after all ?

Distributed collection of objects on disk

or

Distributed collection of objects in memory

or

Distributed collection of objects in Cassandra

Internals of the RDD Interface

RDD is an Interface !!! ..

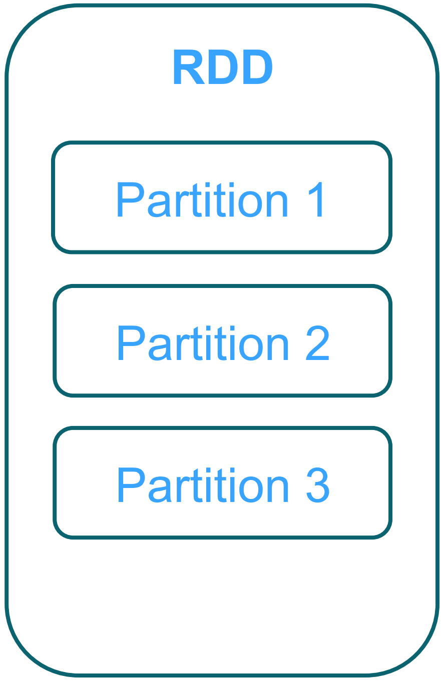
RDD Design Goal:

Support wide array of operators and let users compose them arbitrarily

Don’t want to modify scheduler for each one

How to capture dependencies generically?

1) List of partitions
2) Set of dependencies on parent RDDs
3) Function to compute a partition, given parents
4) Optional partitioning info for k/v RDDs (Partitioner)
5) Optional preferred location info



Types of RDDs

- HadoopRDD
- FilteredRDD
- MappedRDD
- PairRDD
- ShuffledRDD
- UnionRDD
- PythonRDD
- DoubleRDD
- JsonRDD
- JdbcRDD
- SchemaRDD
- VertexRDD
- CassandraRDD

Example: Hadoop RDD

Partitions = 1 per HDFS block (64 or 128 MB)

Dependencies = None

Compute(partition) = read corresponding HDFS block

Preffered Locations(part) = HDFS block location

Partitioner = None

```
> rdd = spark.hadoopFile(“hdfs://click_logs/”)
```

Example: Filtered RDD

Partitions = parent partitions

Dependencies = a single parent

compute(partition) = call parent.compute(partition) and filter

Partitioner = parent partitioner

Preffered Locations(part) = none

```
> filteredRDD = rdd.filter(lambda x: x contains
                    “ERROR”)
```

Example: Joined RDD

Partitions = number chosen by user or heuristics

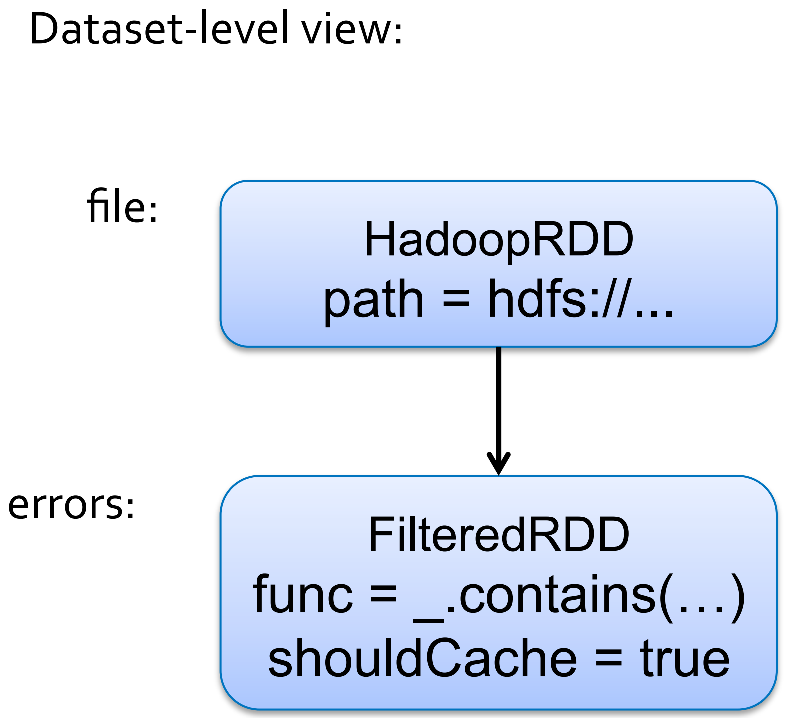
Dependencies = ShuffleDependency on two or more parents

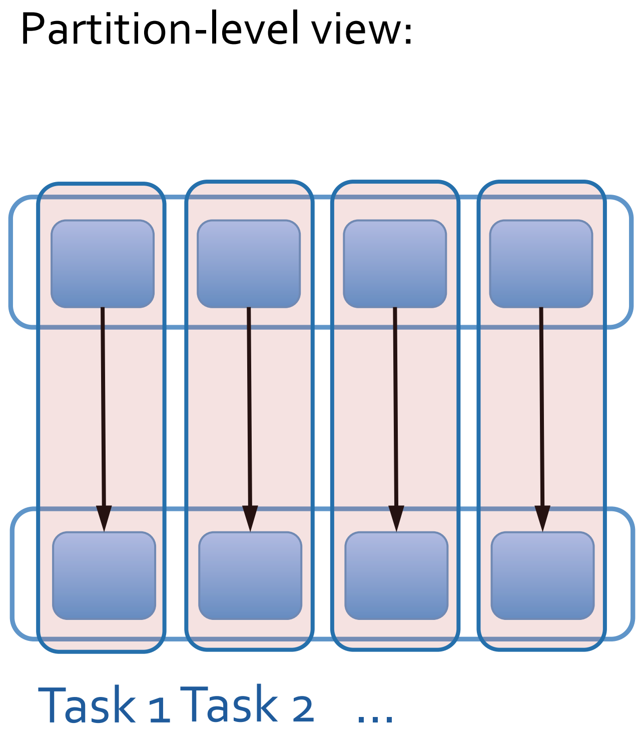
compute(partition) = read and join data from all parents

Partitioner = HashPartitioner(# partitions)

Preffered Locations(part) = none

RDD Graph





First run: data not in cache, so use HadoopRDD's locality prefs (from HDFS)

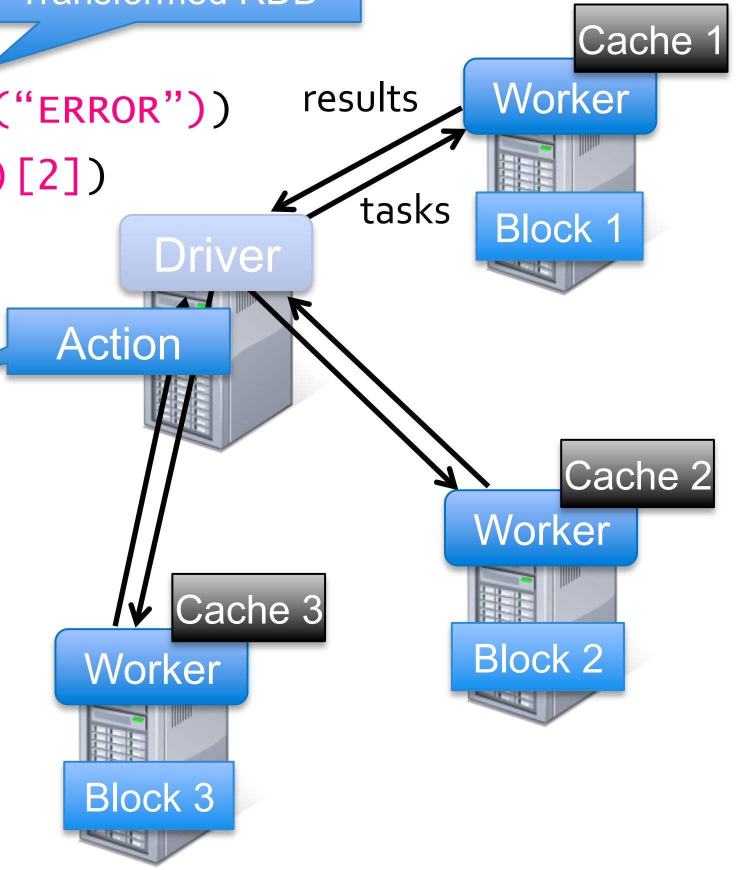
Second run: FilteredRDD is in cache, so use its locations

If something falls out of cache, go back to HDFS

Example: Web Log Mining

Load error messages from a Web log into memory, then interactively search for various patterns

```
lines = sc.textFile("hdfs://...")
errors = lines.filter(lambda x: x.startswith("ERROR"))
messages = errors.map(lambda x: x.split('\t')[2])
messages.cache()

messages.filter(lambda x: "foo" in x).count
messages.filter(lambda x: "bar" in x).count
. . .
```



Example: Web Log Mining

```
lines = sc.textFile(“hdfs://...”)
errors = lines.filter(                        )
messages = errors.map(                        )
messages.cache()

messages.filter(              ).count
messages.filter(              ).count
```

Transformations

```
sc.textFile.filter().map()
```

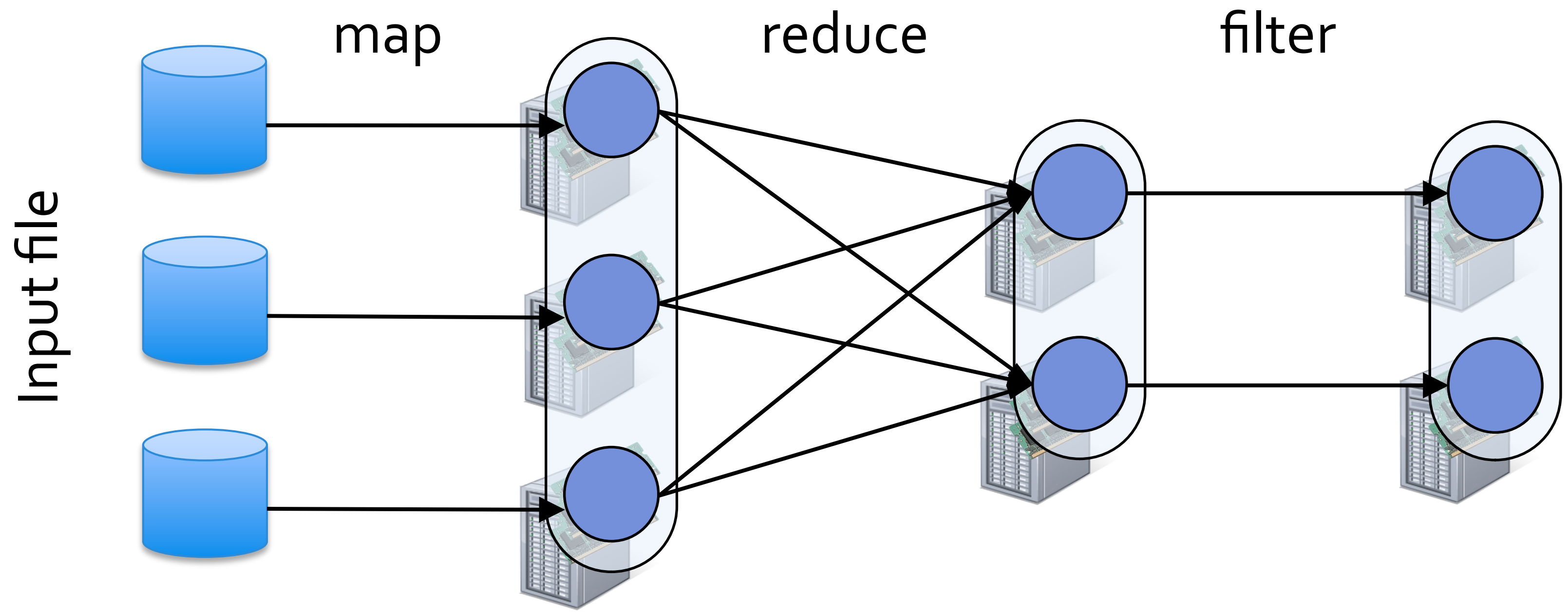
Actions

```
sc.textFile.filter().count()
```

Fault Tolerance

RDDs track *lineage* info to rebuild lost data

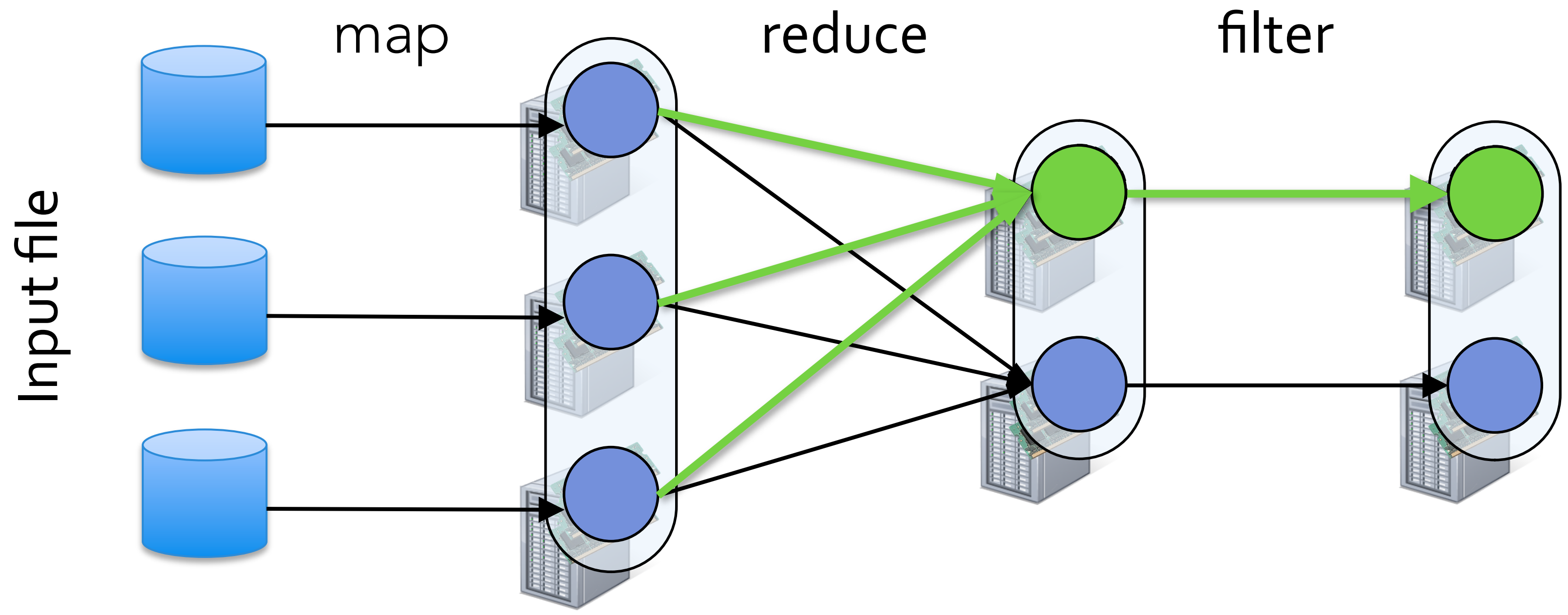
-
  ```
  file.map(lambda rec: (rec.type, 1))
      .reduceByKey(lambda x, y: x + y)
      .filter(lambda (type, count): count > 10)
  ```



Fault Tolerance

RDDs track *lineage* info to rebuild lost data

```
▪ file.map(lambda rec: (rec.type, 1))
        .reduceByKey(lambda x, y: x + y)
        .filter(lambda (type, count): count > 10)
```



RDD Control Methods

RDD.persist(storageLevel)

Store depending on storage level

- Default - MEMORY_ONLY.
- Other options include memory and disk, off-heap

Performed when the RDD is computed.

```
scala> lines.persist(MEMORY_AND_DISK)
```

RDD.unpersist(storageLevel)

```
scala> lines.unpersist()
```

RDD Control Methods

RDD.cache

```
scala> lines.cache()
```

RDD.checkpoint(storageLevel)

- Save to Filesystem
- Forgets the lineage

```
scala> sc.setCheckpointDir("output/checkpoints")
scala> lines.checkpoint
```

DAG Concepts

Jobs: Work required to compute RDD in runJob.

Stages: A wave of work within a job, corresponding to one or more pipelined RDD’s.

Tasks: A unit of work within a stage, corresponding to one RDD partition.

Shuffle: The transfer of data between stages.

WordCount Revisited

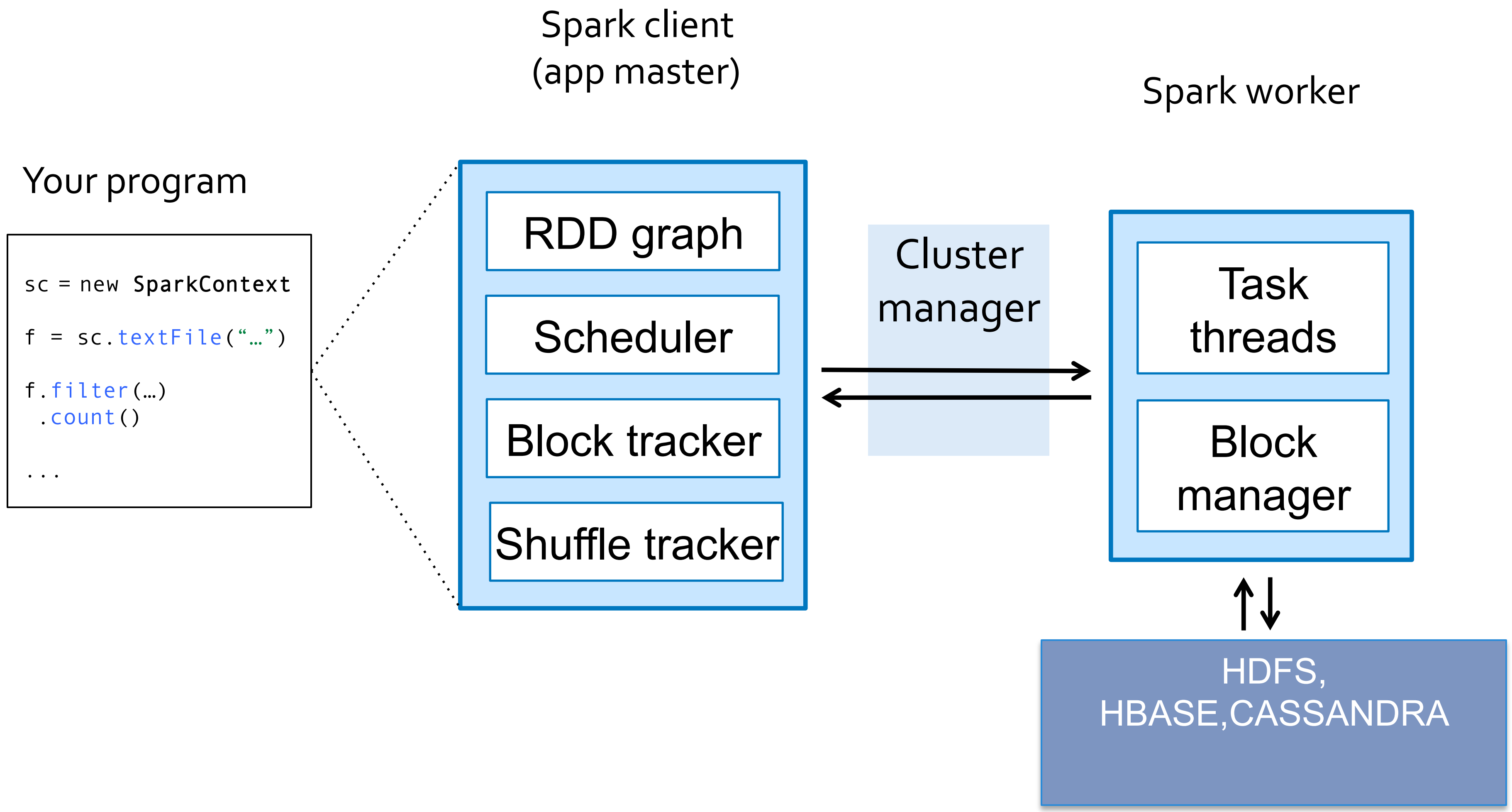
```
val lines= sc.textFile("input")
val words = lines.flatMap(_.split(" "))
val ones = words.map(_ -> 1)
val counts = ones.reduceByKey(_ + _)
val result = counts.collectAsMap()
```

RDD lineage DAG is built on *driver* side with:
-Data Source RDD’s
-Transformation RDD’s

Once an action is triggered on *driver* side, a job is submitted to the *DAG scheduler* of the driver.

Spark Configurations and deployment

A Typical Spark Application



```
sc = new SparkContext
f = sc.textFile("…")
f.filter(…)
 .count()
...
```

Spark Configurations

Option 1 - Changing properties for the SparkConf()

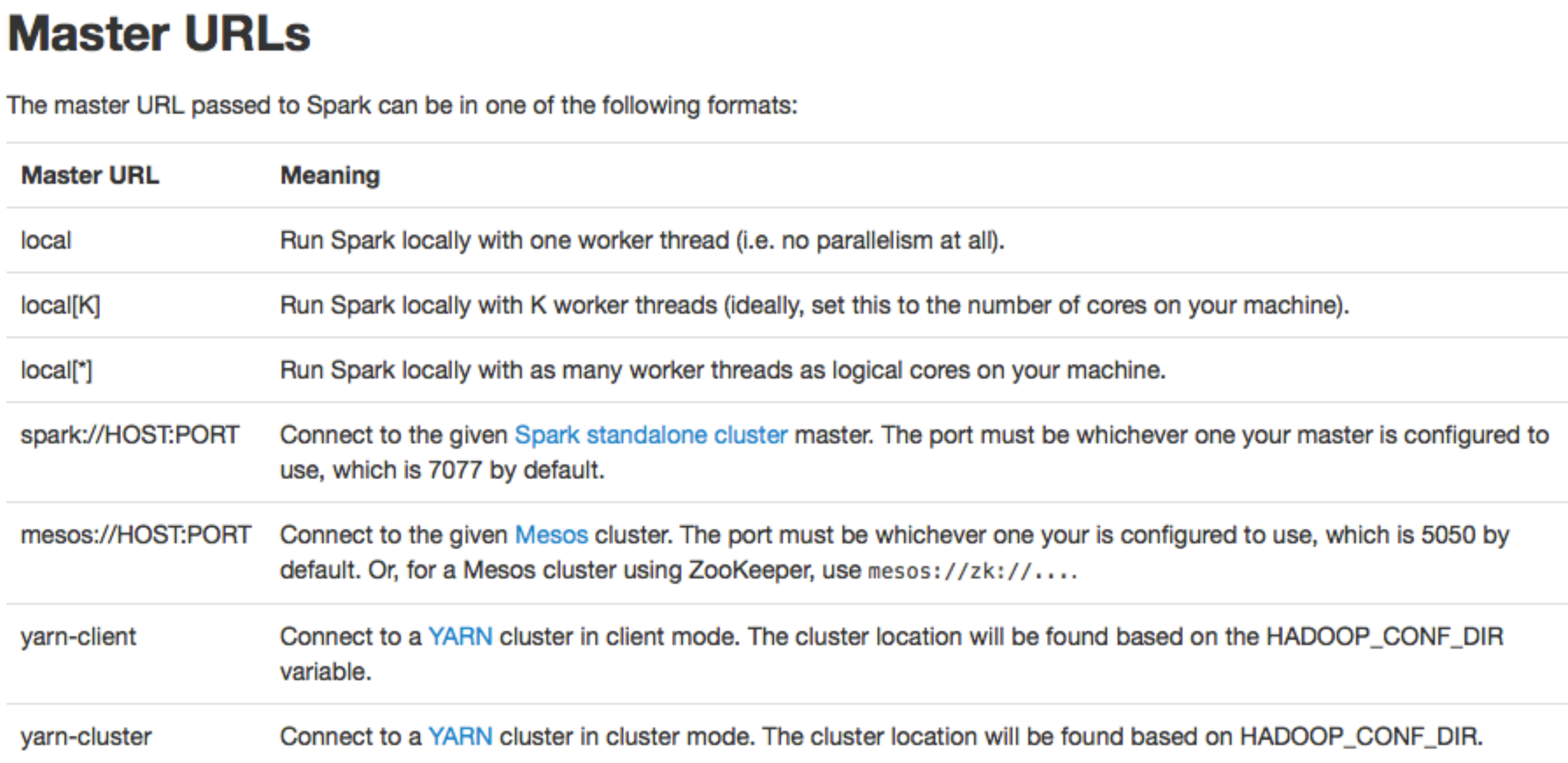
MyScalaApp.scala

```
val conf = new SparkConf()
             .setMaster("local[2]")
             .setAppName("CountingSheep")
             .set("spark.executor.memory", "1g")
val sc = new SparkContext(conf)
```

Option 2 - Supply at runtime

```
./bin/spark-submit --name "My app" --master local[4] --conf spark.shuffle.spill=false --conf
"spark.executor.extraJavaOptions=-XX:+PrintGCDetails -XX:+PrintGCTimeStamps" myApp.jar
```

Spark Master URL’s

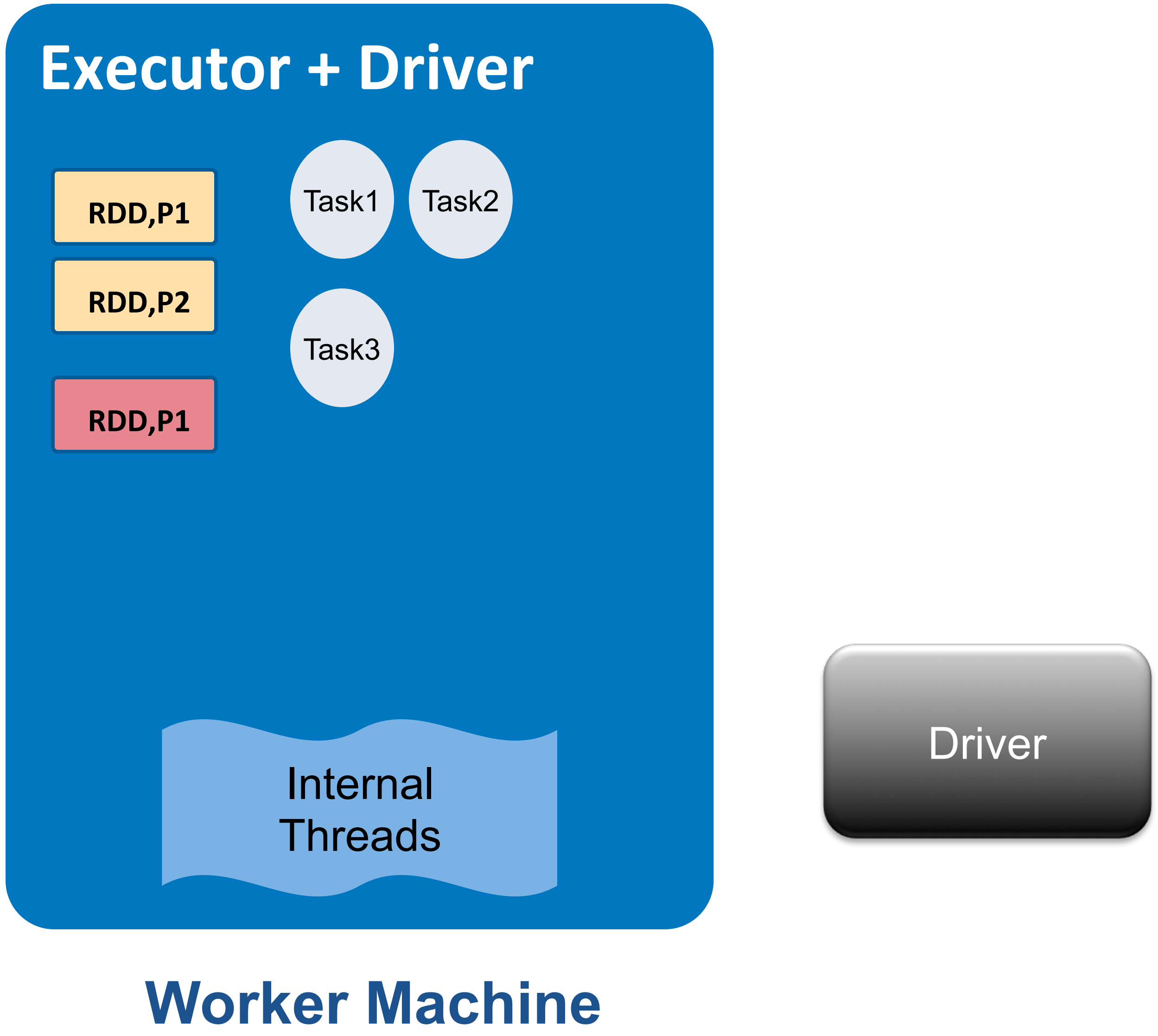


Master URLs

The master URL passed to Spark can be in one of the following formats:

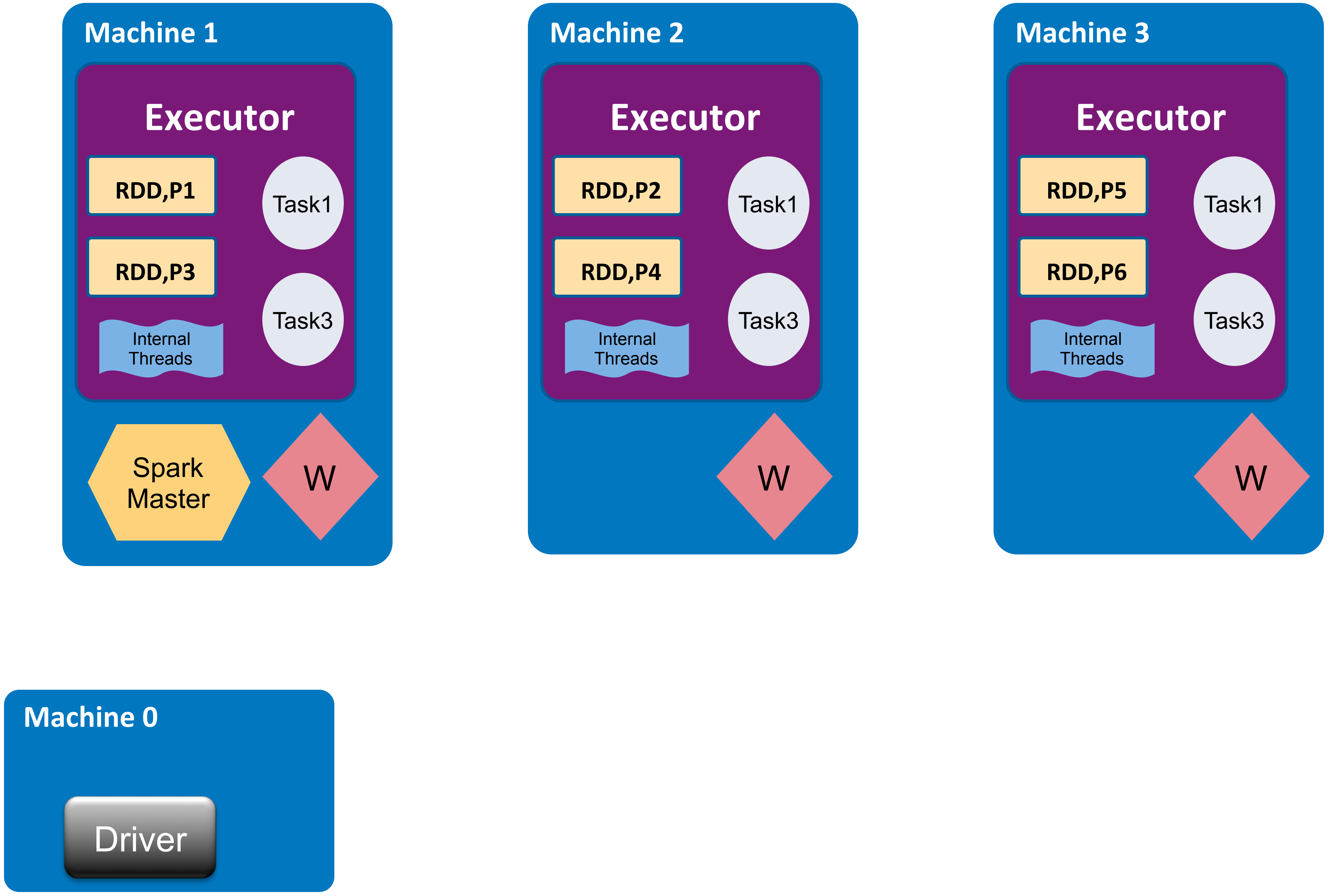
Master URL	Meaning
local	Run Spark locally with one worker thread (i.e. no parallelism at all).
local[K]	Run Spark locally with K worker threads (ideally, set this to the number of cores on your machine).
local[*]	Run Spark locally with as many worker threads as logical cores on your machine.
spark://HOST:PORT	Connect to the given Spark standalone cluster master. The port must be whichever one your master is configured to use, which is 7077 by default.
mesos://HOST:PORT	Connect to the given Mesos cluster. The port must be whichever one your is configured to use, which is 5050 by default. Or, for a Mesos cluster using ZooKeeper, use `mesos://zk://...`.
yarn-client	Connect to a YARN cluster in client mode. The cluster location will be found based on the HADOOP_CONF_DIR variable.
yarn-cluster	Connect to a YARN cluster in cluster mode. The cluster location will be found based on HADOOP_CONF_DIR.

Local Mode

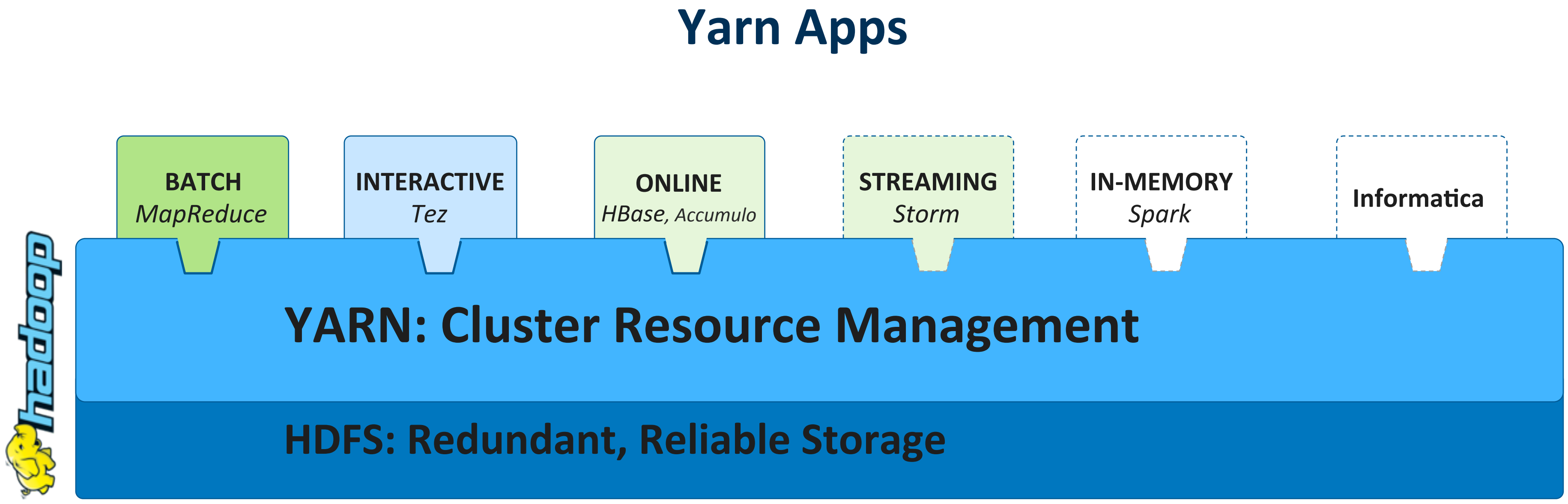


-local
-local[N]
-local[*]

Standalone mode – Spark Cluster



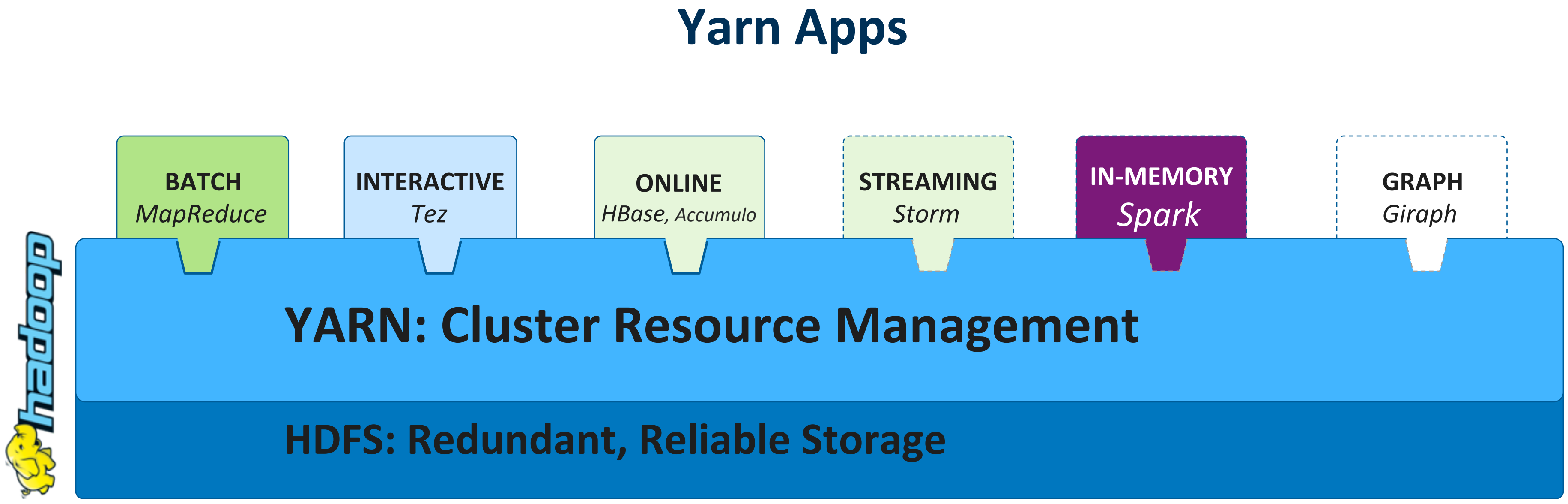
Yarn



In Hadoop 1.0 Mapreduce was the “ONLY” thing

In Hadoop 2.0 Mapreduce is “ONE OF THE” things

SPARK on YARN



Running Spark on YARN

YARN Resource Manager	➔	Spark Master
YARN Node Manager	➔	Spark Workers
YARN Client Mode	➔	Spark Driver runs Locally
YARN Cluster Mode	➔	Spark Driver runs on Application Master

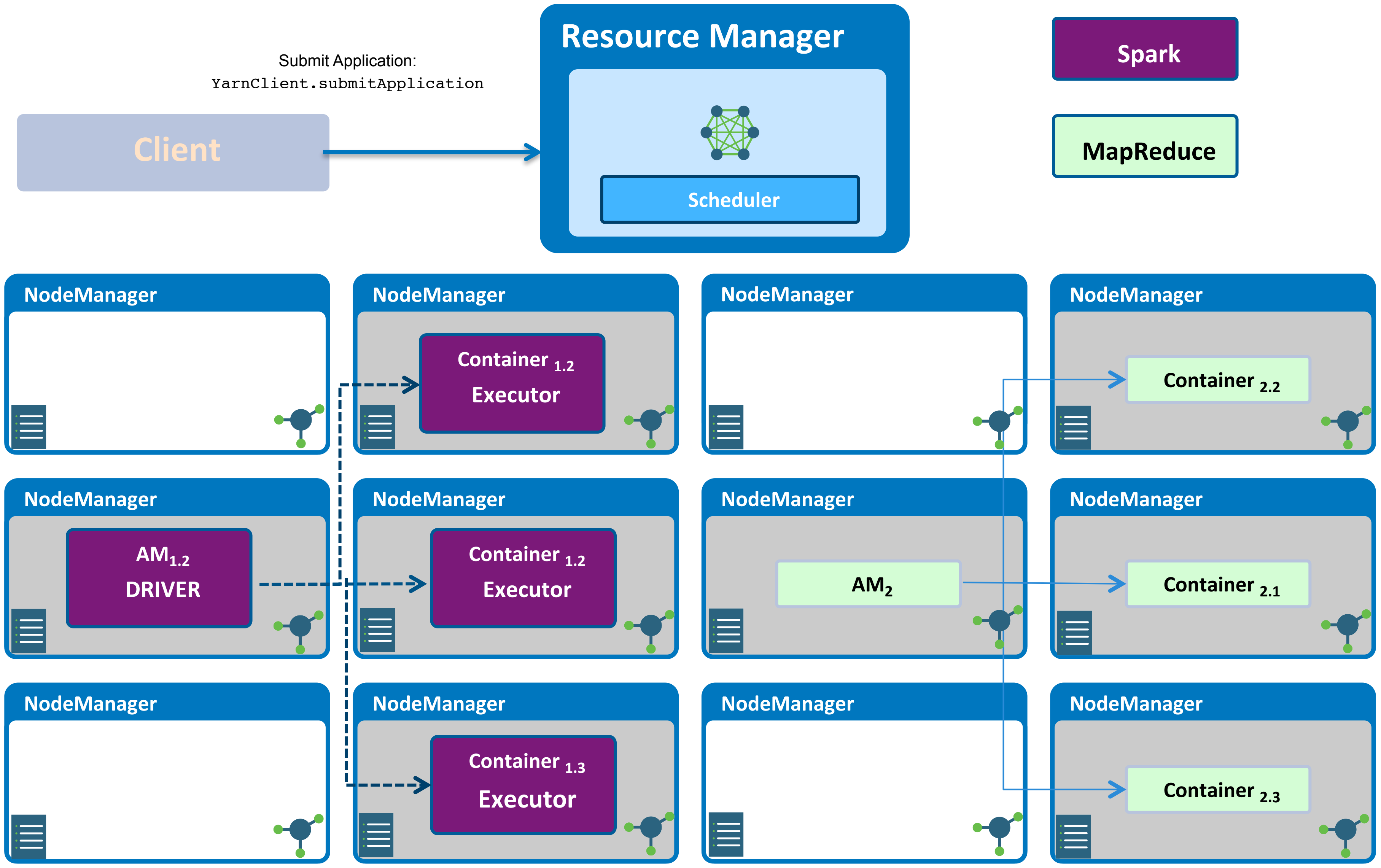
Running Spark in YARN Client Mode

```
spark-submit  \
--class org.apache.spark.examples.SparkPi \
--deploy-mode client \
--master yarn \
$SPARK_HOME/examples/lib/spark-examples_version.jar 10
```

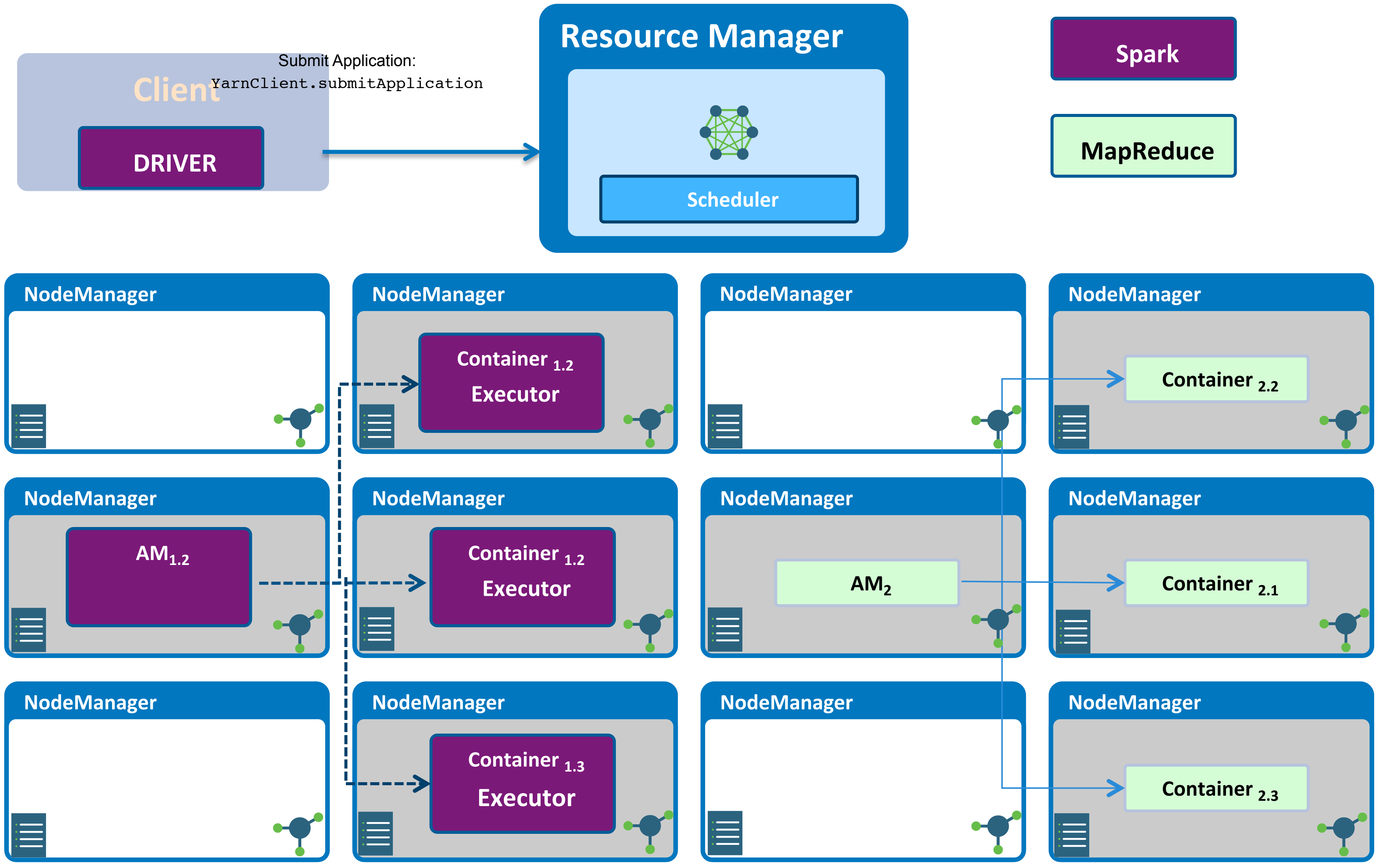
Running SparkPi in YARN Cluster Mode

```
spark-submit  \
--class org.apache.spark.examples.SparkPi \
--deploy-mode cluster \
--master yarn \
$SPARK_HOME/examples/lib/spark-examples_version.jar 10
```

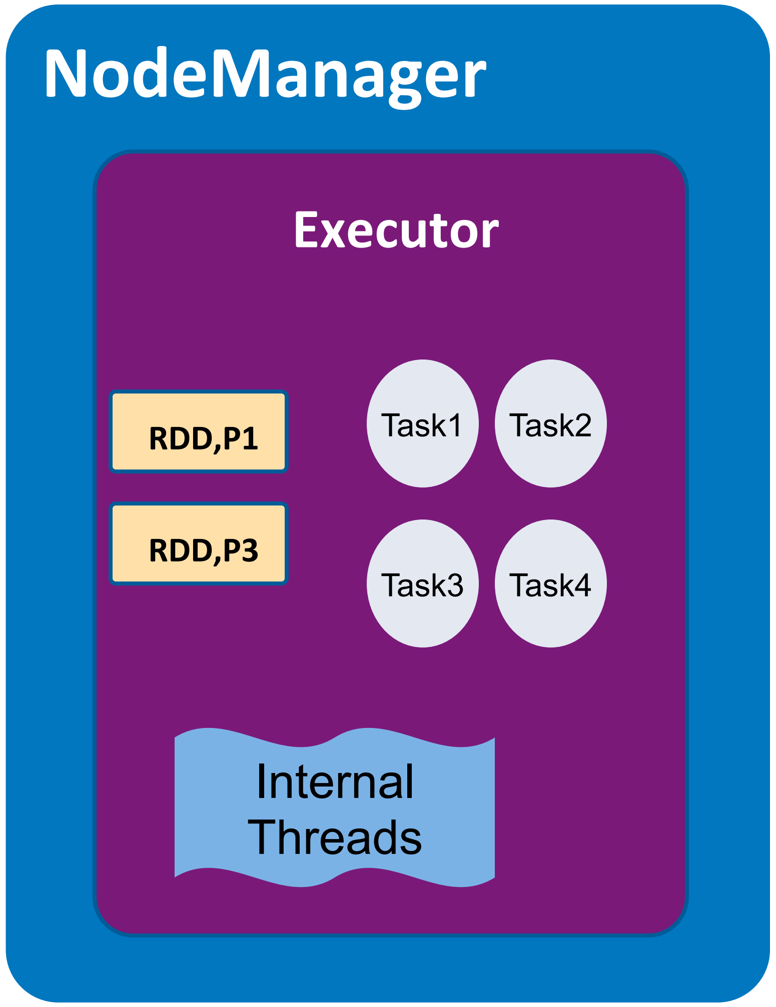
Spark - Yarn Architecture – Cluster Mode



Spark - Yarn Architecture – Client Mode



Yarn Mode - Inside the Executor



Yarn Settings

- YARN settings

--num-executors: controls how many executors will be allocated

--executor-memory: RAM for each executor

--executor-cores: CPU cores for each executor

spark.dynamicAllocation.enabled
spark.dynamicAllocation.minExecutors
spark.dynamicAllocation.maxExecutors
spark.dynamicAllocation.sustainedSchedulerBacklogTimeout (N)
spark.dynamicAllocation.schedulerBacklogTimeout (M)
spark.dynamicAllocation.executorIdleTimeout (K)

Launching Spark Cluster with EC2 tools

a) Download and unzip Spark binary distribution

b) Make sure you have your AWS variables correctly setup

```
export AWS_ACCESS_KEY_ID=
export AWS_SECRET_ACCESS_KEY=
export EC2_CERT= <path-cert.pem>
export EC2_PRIVATE_KEY=<path-pk.pem>
```

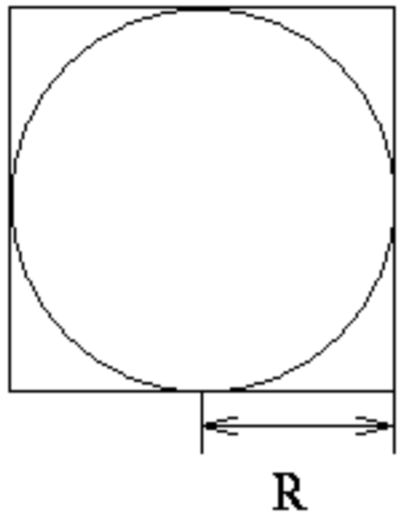
c) Run the spark-ec2 command

```
spark-ec2 -k ec2rbjamazon -i=$AWS_PEM_HOME/ec2rbjamazon.pem
-s3 --region=us-east-1 --instance-type=r3.large
launch MySparkCluster
```

Spark Pi

31415

MonteCarlo method to calculate pi



If assume that the radius of the circle is *R*,

*Area of the circle = Pi * R^2*
and the
*Area of the square = (2 * R)^2 = 4 * R^2*.

If we throw a dart blindly inside of the
square, what is the probability (P) the dart will actually land inside the circle?

P = Area of the circle / Area of the square
= Pi **R^2* / 4**R^2*
= *Pi / 4*

So the chances of hitting the circle are Pi / 4.
In other words, pi = 4 * P

Launching PI example

```
ssh –i  ec2-107-22-91-199.compute-1.amazonaws.com
```

```
spark-submit  --class org.apache.spark.examples.SparkPi
--deploy-mode client --master yarn
/opt/cloudera/parcels/CDH-5.4.0-1.cdh5.4.0.p0.27/jars/spark-examples-1.3.0-
cdh5.4.0-hadoop2.6.0-cdh5.4.0.jar 20
```

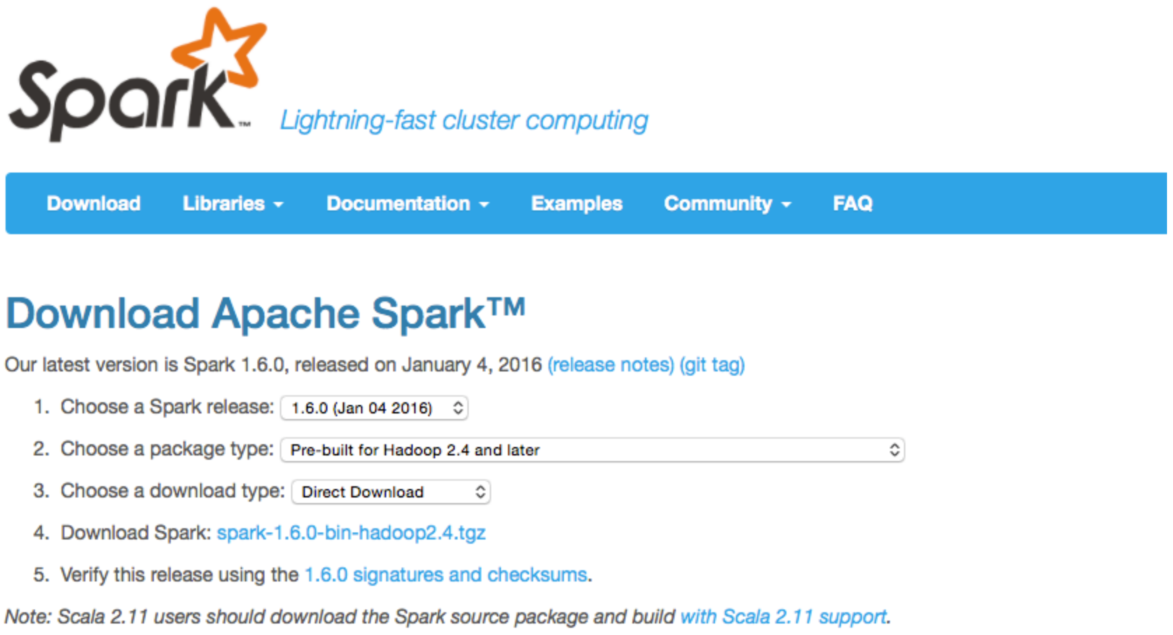
Scala IDE Setup

Scala IDE setup with Eclipse

Eclipse Setup for Spark and Scala

Download Spark binaries –These are needed to be added in Eclipse build path

http://spark.apache.org/downloads.html



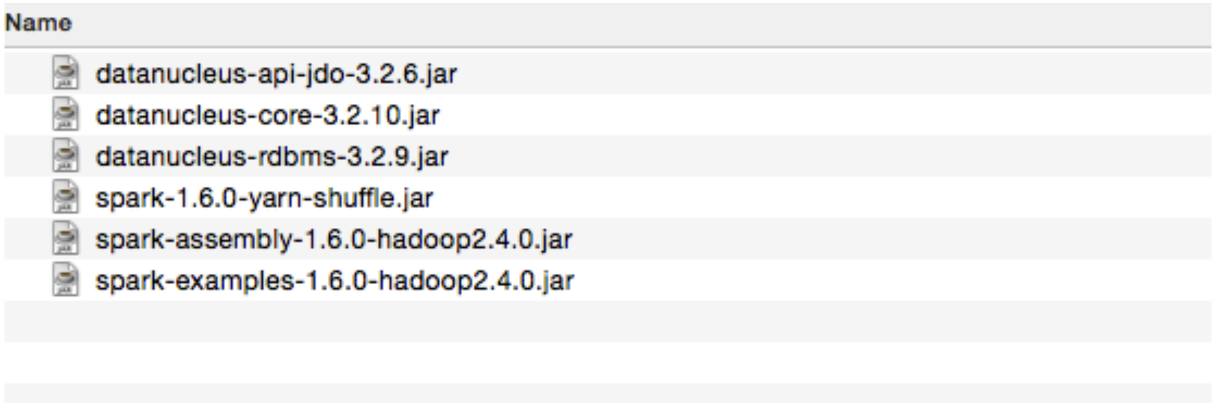
Download Apache Spark™

Our latest version is Spark 1.6.0, released on January 4, 2016 (release notes) (git tag)

1. Choose a Spark release: 1.6.0 (Jan 04 2016)
2. Choose a package type: Pre-built for Hadoop 2.4 and later
3. Choose a download type: Direct Download
4. Download Spark: spark-1.6.0-bin-hadoop2.4.tgz
5. Verify this release using the 1.6.0 signatures and checksums.

Note: Scala 2.11 users should download the Spark source package and build with Scala 2.11 support.

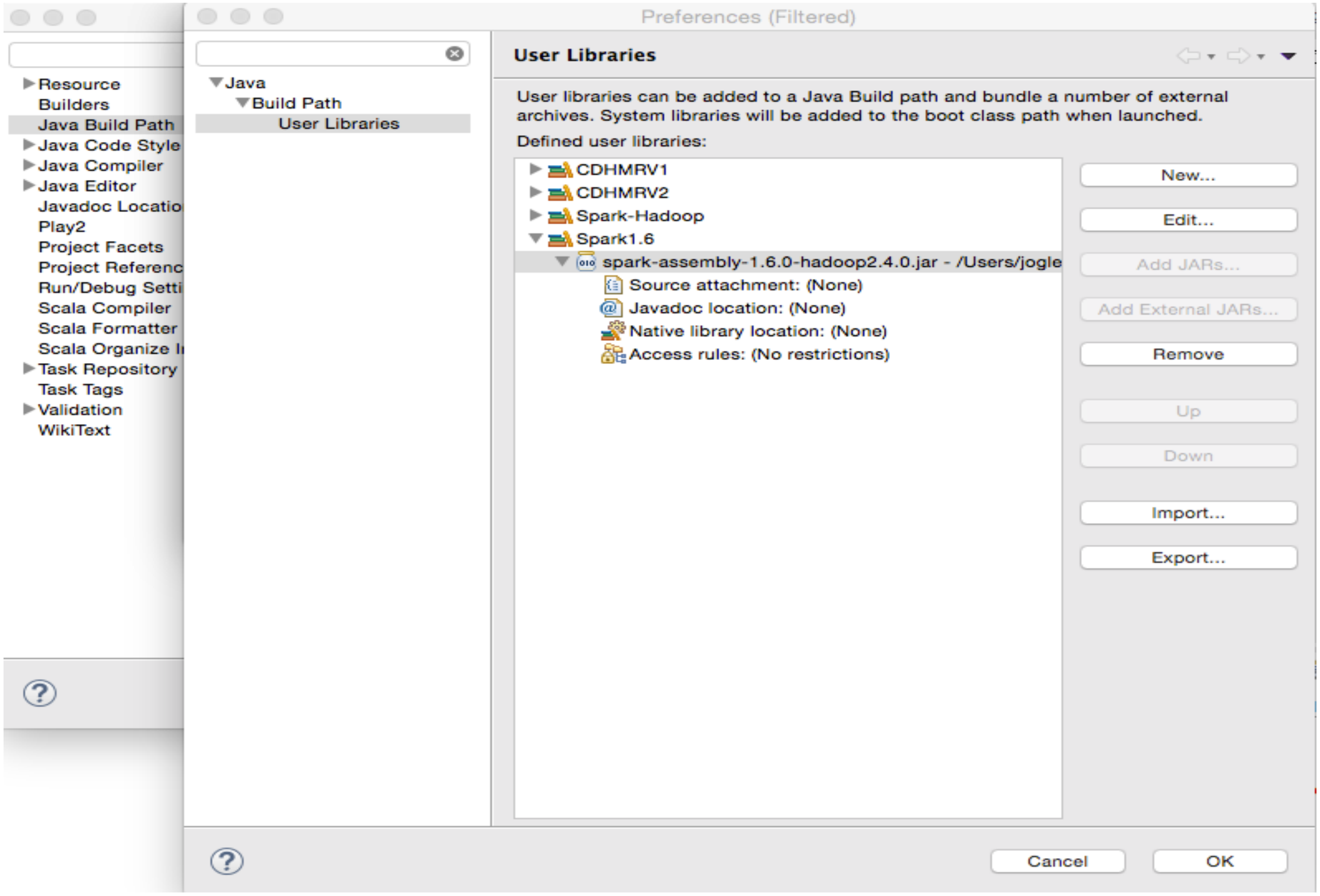
"lib" Contains these Jar files



Name
datanucleus-api-jdo-3.2.6.jar
datanucleus-core-3.2.10.jar
datanucleus-rdbms-3.2.9.jar
spark-1.6.0-yarn-shuffle.jar
spark-assembly-1.6.0-hadoop2.4.0.jar
spark-examples-1.6.0-hadoop2.4.0.jar

Eclipse Setup for Spark and Scala

Create a new User Library Spark 1.6



Preferences (Filtered)

User Libraries

User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.

Defined user libraries:

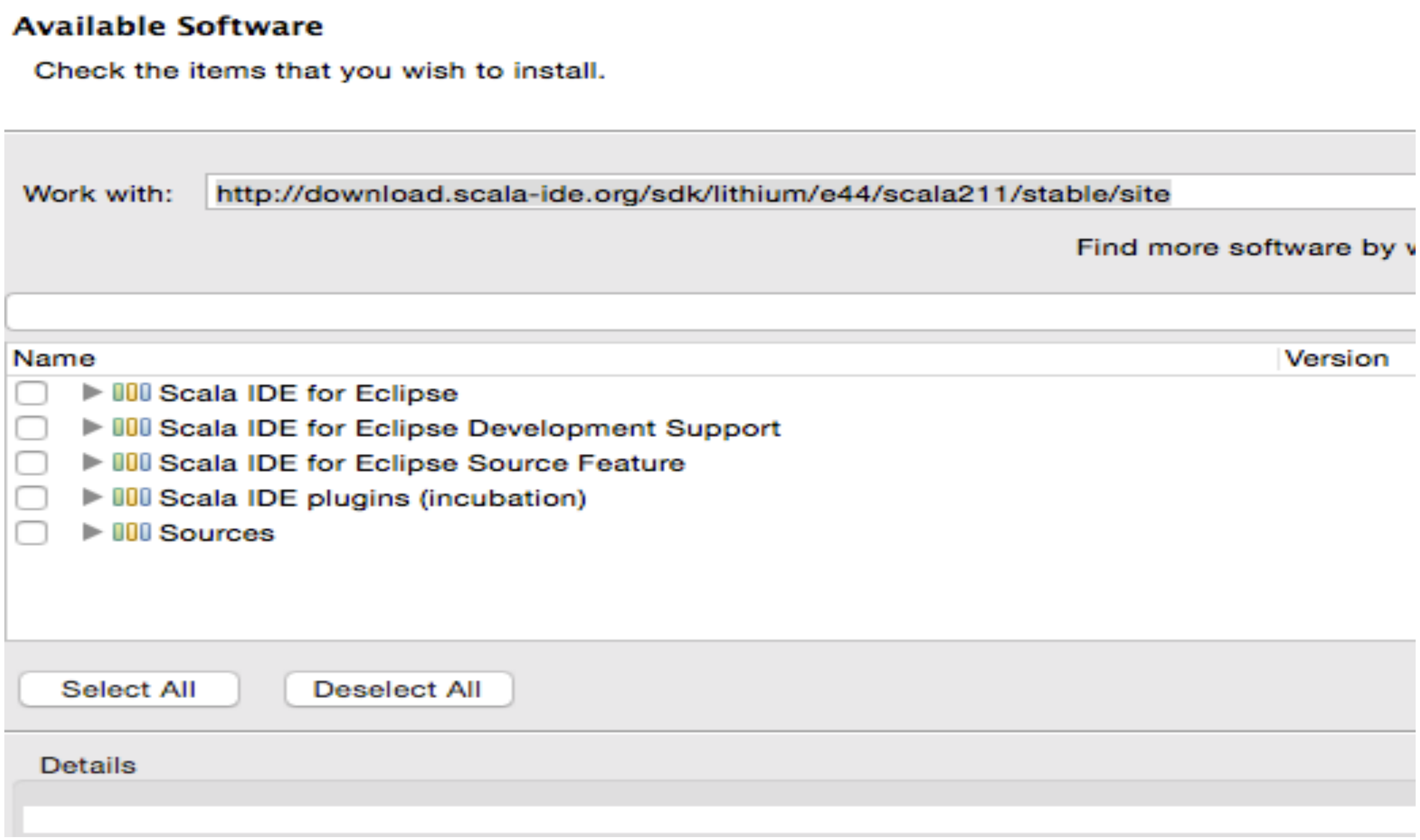
- CDHMRV1
- CDHMRV2
- Spark-Hadoop
- Spark1.6
  - spark-assembly-1.6.0-hadoop2.4.0.jar - /Users/jogle
    - Source attachment: (None)
    - Javadoc location: (None)
    - Native library location: (None)
    - Access rules: (No restrictions)

Eclipse Setup for Spark and Scala

Add Scala Plugin to Eclipse

Within Eclipse Help → Install New Software…

http://download.scala-ide.org/sdk/lithium/e44/scala211/stable/site



Available Software

Check the items that you wish to install.

Work with: http://download.scala-ide.org/sdk/lithium/e44/scala211/stable/site

Find more software by w

Name	Version
☐ ▶ Scala IDE for Eclipse	
☐ ▶ Scala IDE for Eclipse Development Support	
☐ ▶ Scala IDE for Eclipse Source Feature	
☐ ▶ Scala IDE plugins (incubation)	
☐ ▶ Sources	

Eclipse Setup for Spark and Scala - Scala IDE

Scala IDE Use the URL’s below

Eclipse Luna

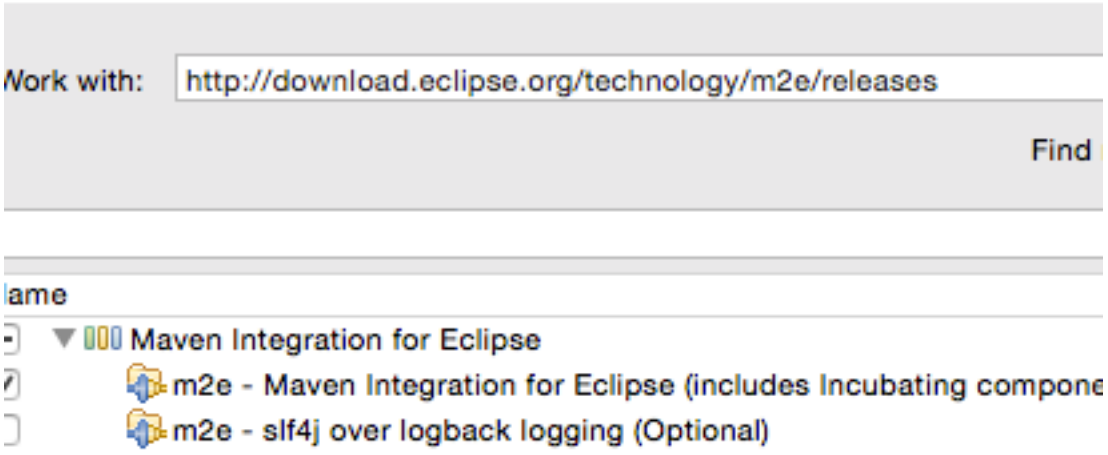
http://download.scala-ide.org/sdk/lithium/e44/scala211/stable/site

Eclipse Juno/Kepler

http://download.scala-ide.org/sdk/lithium/e38/scala211/stable/site

Eclipse Setup - M2Eclipse and M2Eclipse-Scala

Make sure you have M2Eclipse , download if you don't .



Work with: http://download.eclipse.org/technology/m2e/releases

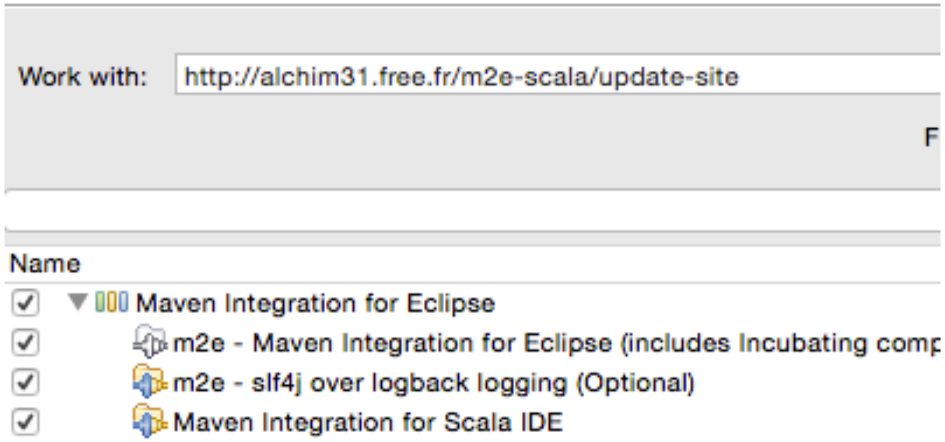
Find

Name

- Maven Integration for Eclipse
  - ✓ m2e - Maven Integration for Eclipse (includes Incubating compone
  - m2e - slf4j over logback logging (Optional)

Scala needs another plugin M2Eclipse-Scala

http://alchim31.free.fr/m2e-scala/update-site



Work with: http://alchim31.free.fr/m2e-scala/update-site

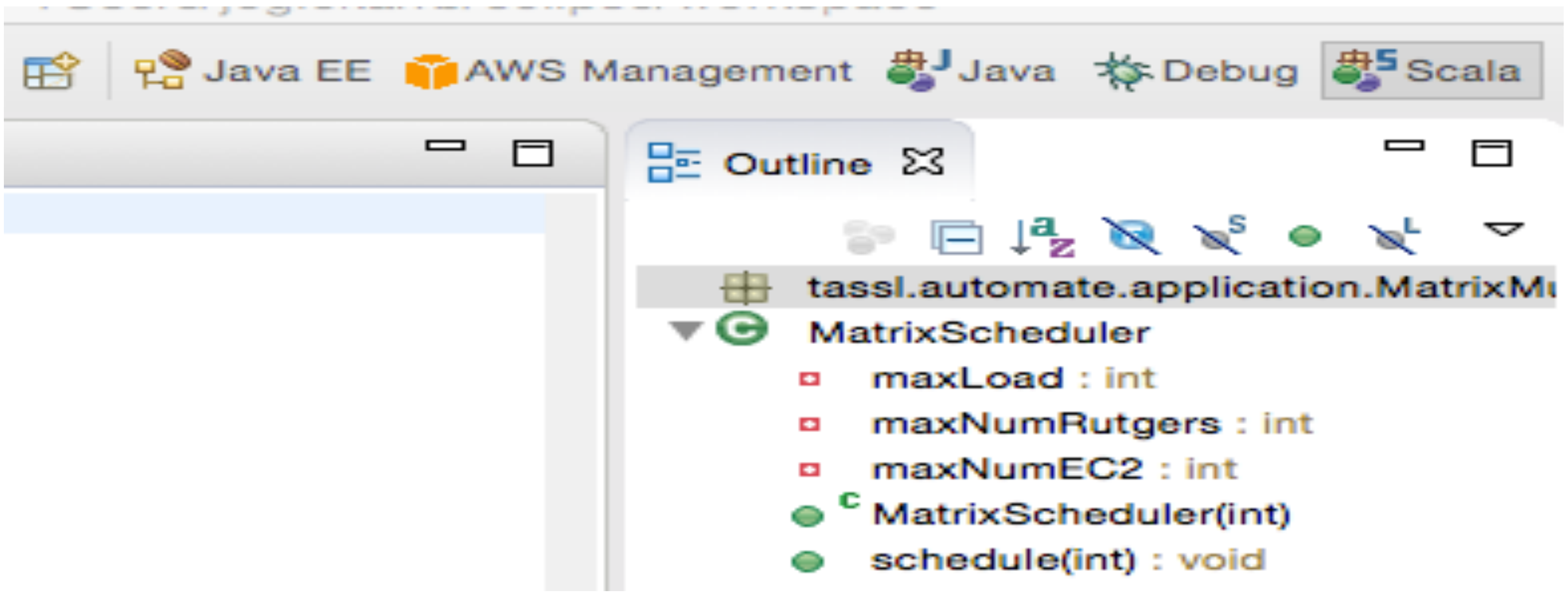
F

Name

- ✓ Maven Integration for Eclipse
  - ✓ m2e - Maven Integration for Eclipse (includes Incubating comp
  - ✓ m2e - slf4j over logback logging (Optional)
  - ✓ Maven Integration for Scala IDE

First Scala Project in Eclipse

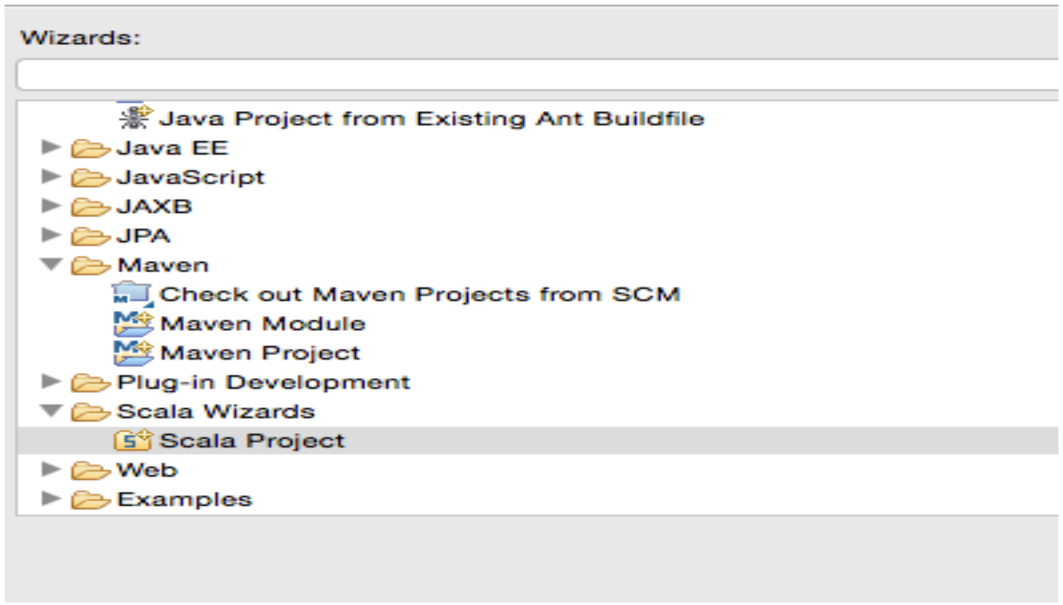
Switch Eclipse Perspective to Scala



```
tassl.automate.application.MatrixM
MatrixScheduler
    maxLoad : int
    maxNumRutgers : int
    maxNumEC2 : int
    MatrixScheduler(int)
    schedule(int) : void
```

First Scala Project in Eclipse

Within Eclipse File→ New Project → Scala Project



Wizards:

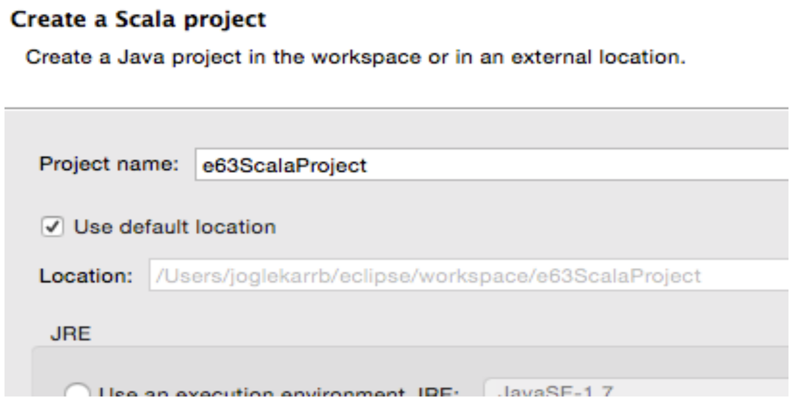
Java Project from Existing Ant Buildfile
▶ Java EE
▶ JavaScript
▶ JAXB
▶ JPA
▼ Maven
Check out Maven Projects from SCM
Maven Module
Maven Project
▶ Plug-in Development
▼ Scala Wizards
Scala Project
▶ Web
▶ Examples

File → New –> Scala Project

Next src → New Package → call it edu.hu.e63

Click on package → new Scala Class

Next → Name your project and you will see a scala project in the explorer



Create a Scala project

Create a Java project in the workspace or in an external location.

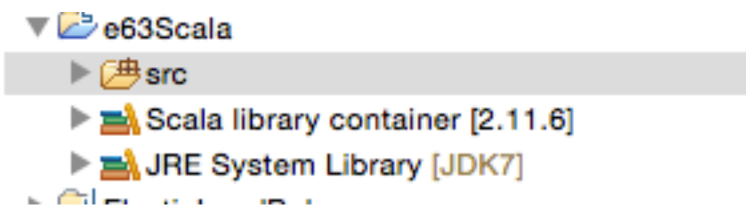
Project name: e63ScalaProject

☑ Use default location

Location: /Users/joglekarrb/eclipse/workspace/e63ScalaProject

JRE

○ Use an execution environment JRE: JavaSE-1.7

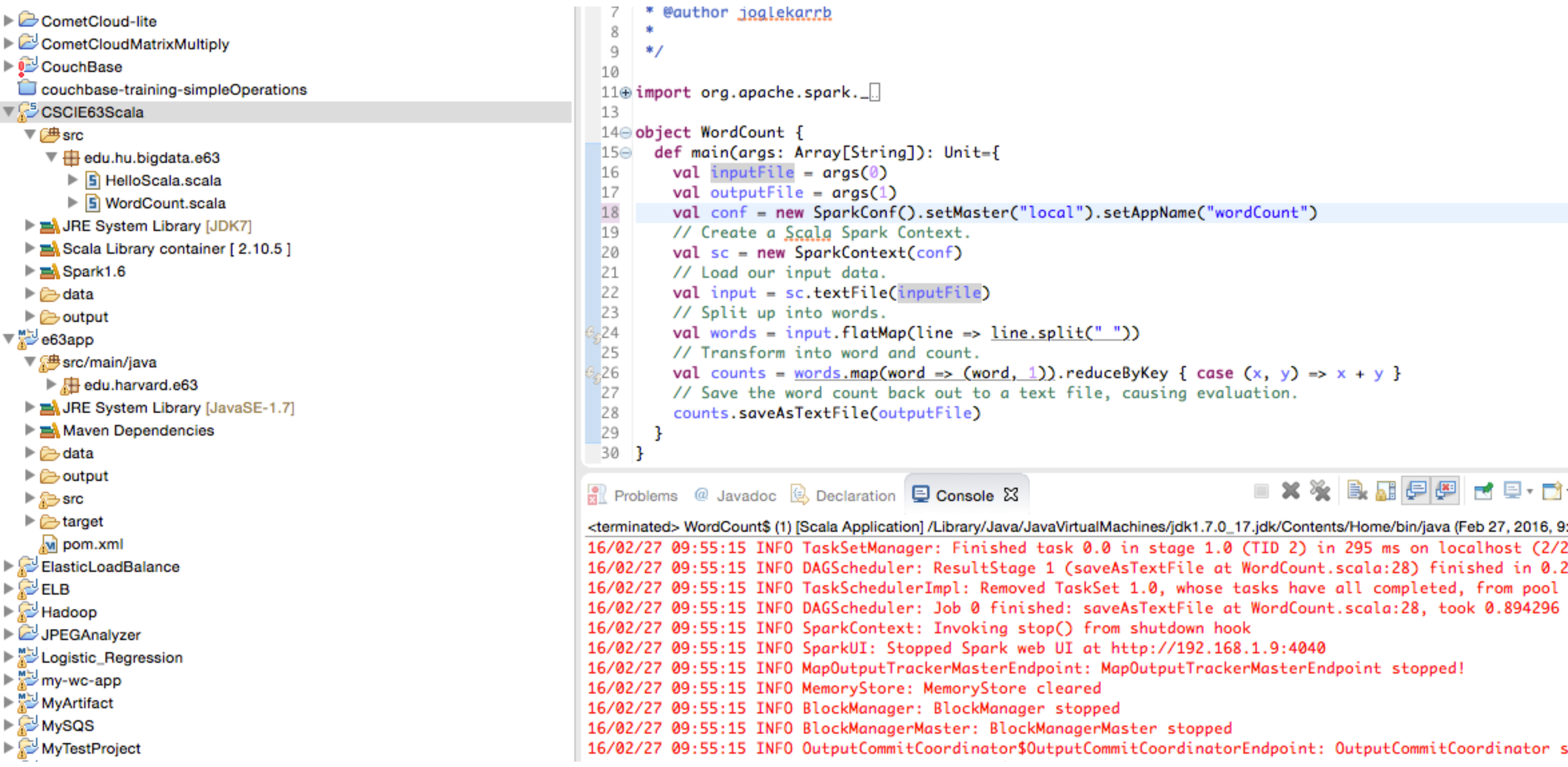


▼ e63Scala
▶ src
▶ Scala library container [2.11.6]
▶ JRE System Library [JDK7]

First Scala Project in Eclipse

Within Eclipse Project you just created
Src → New Package → call it edu.hu.e63

Click on package → new Scala Class



```
CometCloud-lite
CometCloudMatrixMultiply
CouchBase
couchbase-training-simpleOperations
CSCIE63Scala
  src
    edu.hu.bigdata.e63
      HelloScala.scala
      WordCount.scala
  JRE System Library [JDK7]
  Scala Library container [ 2.10.5 ]
  Spark1.6
  data
  output
e63app
  src/main/java
    edu.harvard.e63
  JRE System Library [JavaSE-1.7]
  Maven Dependencies
  data
  output
  src
  target
  pom.xml
ElasticLoadBalance
ELB
Hadoop
JPEGAnalyzer
Logistic_Regression
my-wc-app
MyArtifact
MySQS
MyTestProject
```

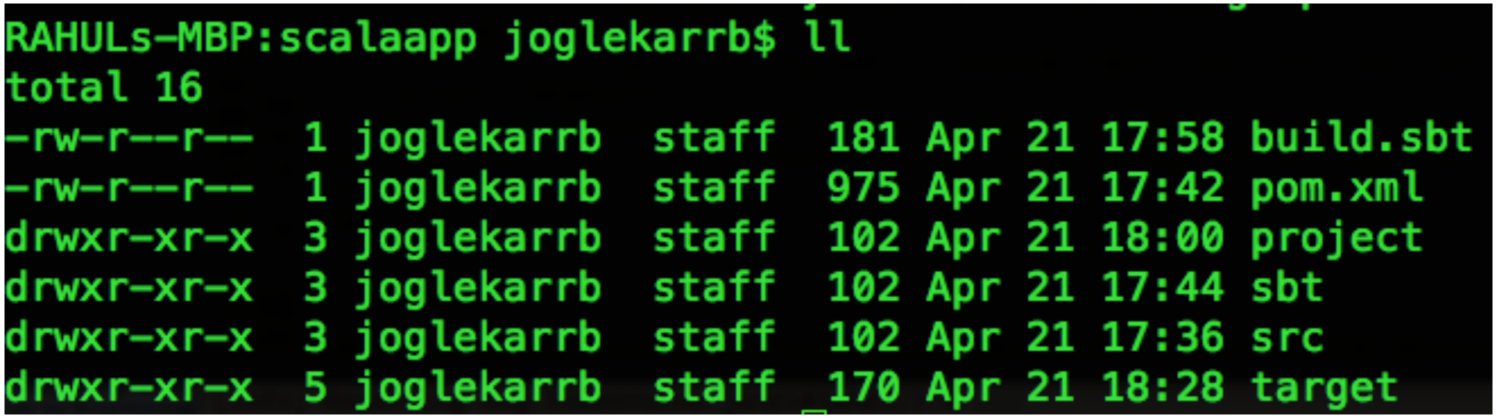
```scala
 * @author joglekarrb
 *
 */

import org.apache.spark._

object WordCount {
  def main(args: Array[String]): Unit={
    val inputFile = args(0)
    val outputFile = args(1)
    val conf = new SparkConf().setMaster("local").setAppName("wordCount")
    // Create a Scala Spark Context.
    val sc = new SparkContext(conf)
    // Load our input data.
    val input = sc.textFile(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 { case (x, y) => x + y }
    // Save the word count back out to a text file, causing evaluation.
    counts.saveAsTextFile(outputFile)
  }
}
```

Problems Javadoc Declaration Console

```
<terminated> WordCount$ (1) [Scala Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_17.jdk/Contents/Home/bin/java (Feb 27, 2016, 9:
16/02/27 09:55:15 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 2) in 295 ms on localhost (2/2
16/02/27 09:55:15 INFO DAGScheduler: ResultStage 1 (saveAsTextFile at WordCount.scala:28) finished in 0.2
16/02/27 09:55:15 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
16/02/27 09:55:15 INFO DAGScheduler: Job 0 finished: saveAsTextFile at WordCount.scala:28, took 0.894296
16/02/27 09:55:15 INFO SparkContext: Invoking stop() from shutdown hook
16/02/27 09:55:15 INFO SparkUI: Stopped Spark web UI at http://192.168.1.9:4040
16/02/27 09:55:15 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
16/02/27 09:55:15 INFO MemoryStore: MemoryStore cleared
16/02/27 09:55:15 INFO BlockManager: BlockManager stopped
16/02/27 09:55:15 INFO BlockManagerMaster: BlockManagerMaster stopped
16/02/27 09:55:15 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator s
```

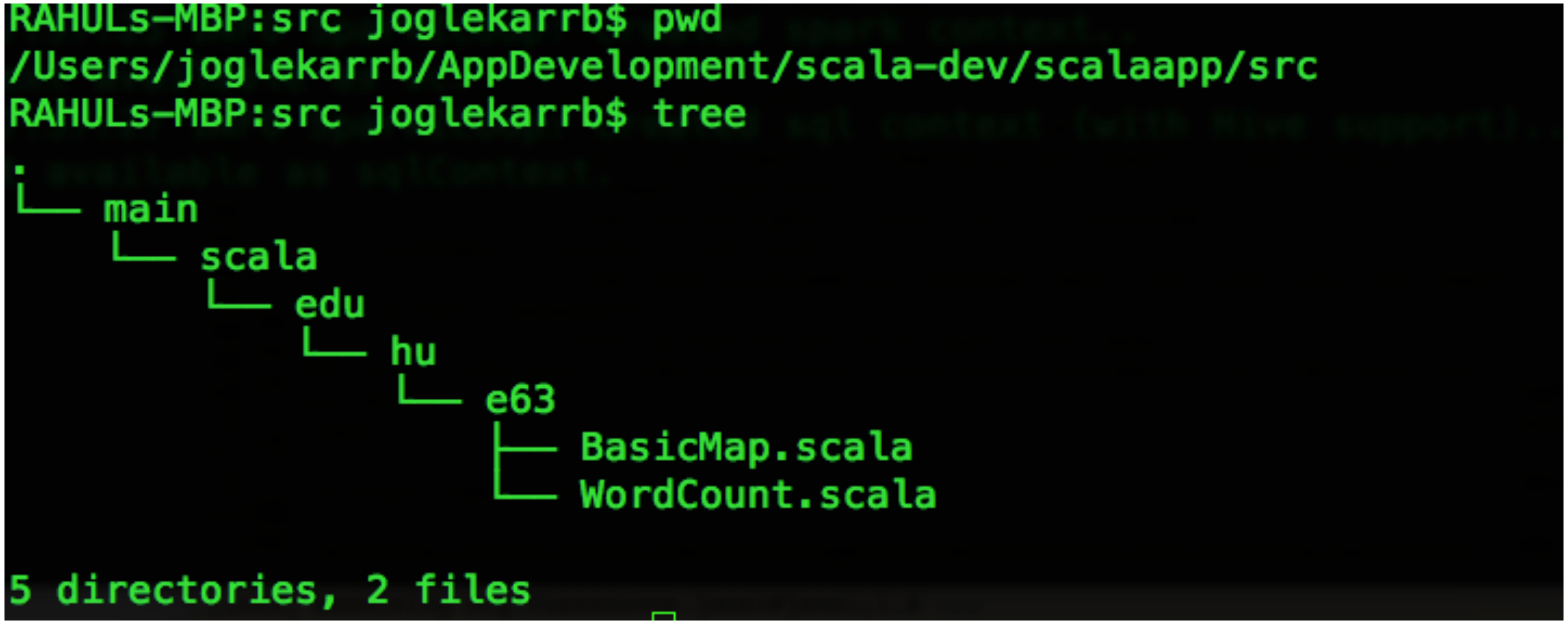
Scala building with sbt

Project Structure



```
RAHULs-MBP:scalaapp joglekarrb$ ll
total 16
-rw-r--r--  1 joglekarrb  staff  181 Apr 21 17:58 build.sbt
-rw-r--r--  1 joglekarrb  staff  975 Apr 21 17:42 pom.xml
drwxr-xr-x  3 joglekarrb  staff  102 Apr 21 18:00 project
drwxr-xr-x  3 joglekarrb  staff  102 Apr 21 17:44 sbt
drwxr-xr-x  3 joglekarrb  staff  102 Apr 21 17:36 src
drwxr-xr-x  5 joglekarrb  staff  170 Apr 21 18:28 target
```

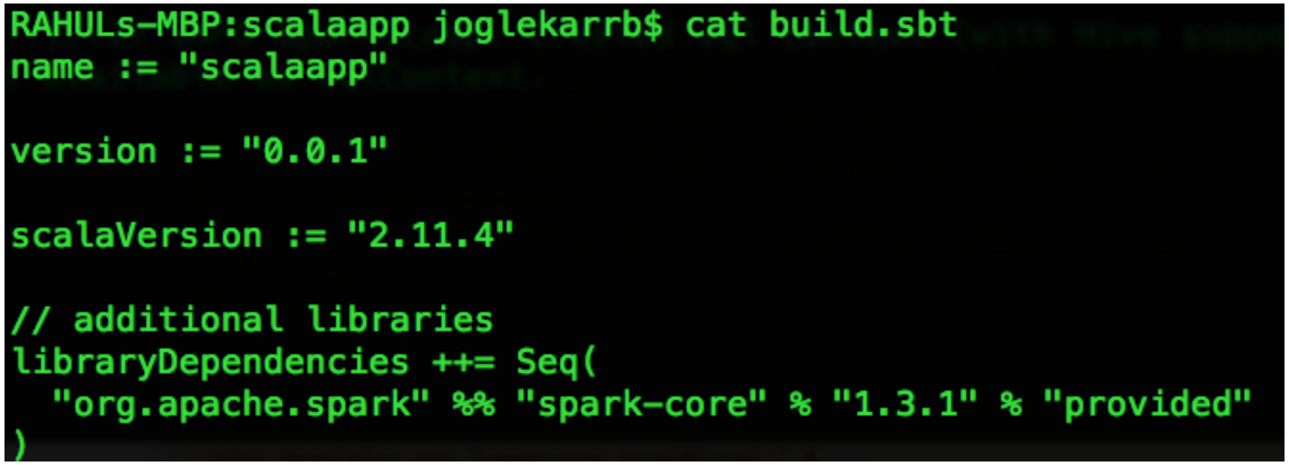
Src directory needs to be setup correctly



```
RAHULs-MBP:src joglekarrb$ pwd
/Users/joglekarrb/AppDevelopment/scala-dev/scalaapp/src
RAHULs-MBP:src joglekarrb$ tree
.
└── main
    └── scala
        └── edu
            └── hu
                └── e63
                    ├── BasicMap.scala
                    └── WordCount.scala

5 directories, 2 files
```

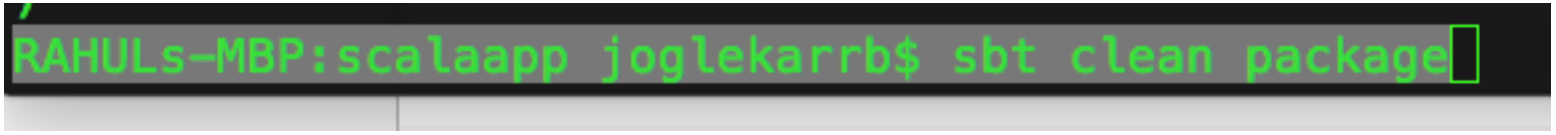
Build

build.sbt - Notice the blank line after every statement



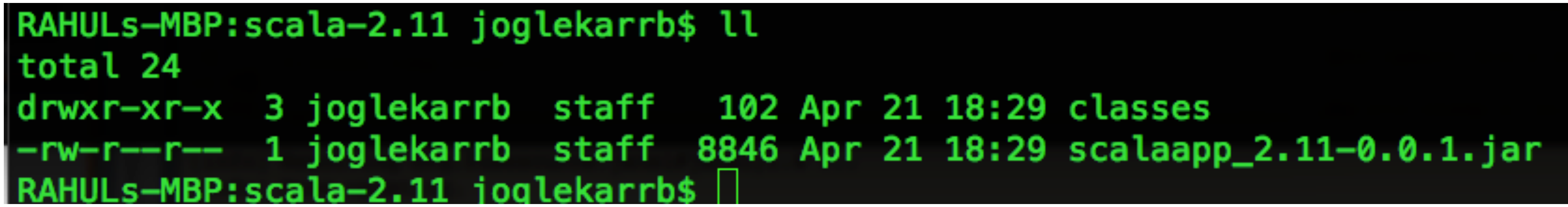
```
RAHULs-MBP:scalaapp joglekarrb$ cat build.sbt
name := "scalaapp"

version := "0.0.1"

scalaVersion := "2.11.4"

// additional libraries
libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % "1.3.1" % "provided"
)
```

Run the build



```
RAHULs-MBP:scalaapp joglekarrb$ sbt clean package
```

Notice the scalaapp_2.11-0.0.1.jar was created



```
RAHULs-MBP:scala-2.11 joglekarrb$ ll
total 24
drwxr-xr-x  3 joglekarrb  staff   102 Apr 21 18:29 classes
-rw-r--r--  1 joglekarrb  staff  8846 Apr 21 18:29 scalaapp_2.11-0.0.1.jar
RAHULs-MBP:scala-2.11 joglekarrb$
```

Maven

Maven is a Java build tool. Top level Apache Project active since 2001

Maven project structure

- target: Default work directory
- src: All project source files go in this directory
- src/main: All sources that go into primary artifact
- src/test: All sources contributing to testing project
- src/main/java: All java source files
- src/main/webapp: All web source files
- src/main/resources: All non compiled source files
- src/test/java: All java test source files
- src/test/resources: All non compiled test source files

Maven POM

- **Stands for Project Object Model**
- **Describes a project**
  - Name and Version
  - Artifact Type
  - Source Code Locations
  - Dependencies
  - Plugins
  - Profiles (Alternate build configurations)
- **Uses XML by Default**
  - Not the way Ant uses XML

POM - GAV

- **Maven uniquely identifies a project using:**
  - groupID: Arbitrary project grouping identifier (no spaces or colons)
    - Usually loosely based on Java package
  - artfiactId: Arbitrary name of project (no spaces or colons)
  - version: Version of project
    - Format {Major}.{Minor}.{Maintanence}
    - Add '-SNAPSHOT ' to identify in development
- **GAV Syntax: groupId:artifactId:version**

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>edu.harvard.e63</groupId>
    <artifactId>MyWordCountApp</artifactId>
    <version>1.0</version>
</project>
```

POM – Dependencies and Packaging

- **Dependencies - GAV**
- **Build type identified using the "packaging" element**
- **Tells Maven how to build the project**
- **Example packaging types:**
  - pom, jar, war, ear, custom
  - Default is jar

```
<project>
    ...
    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
</project>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
 <modelVersion>4.0.0</modelVersion>
    <groupId>edu.harvard.e63</groupId>
    <artifactId>MyWordCountApp</artifactId>
    <packaging>jar</packaging>
</project>
```

Maven Repositories

- **Dependencies are downloaded from repositories**
  - Via http
- **Downloaded dependencies are cached in a local repository**
  - Usually found in ${user.home}/.m2/repository
- **Repository follows a simple directory structure**
  - {groupId}/{artifactId}/{version}/{artifactId}-{version}.jar
  - groupId '.' is replaced with '/'
- **Maven Central is primary community repo**
  - http://repo1.maven.org/maven2

Generate using Archtypes

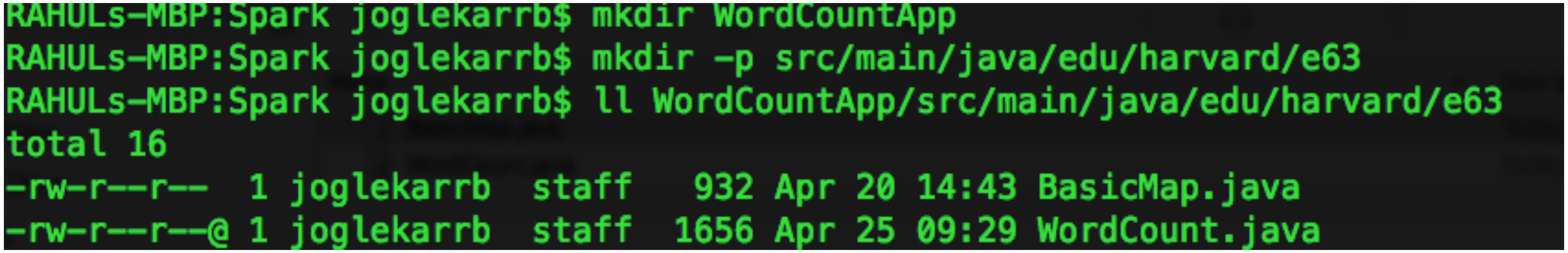
Java Project with Maven

```
mvn archetype:generate -
DgroupId=edu.harvard.e63 -
DartifactId=my-wc-app -
DarchetypeArtifactId=maven-archetype-
quickstart -DinteractiveMode=false
```

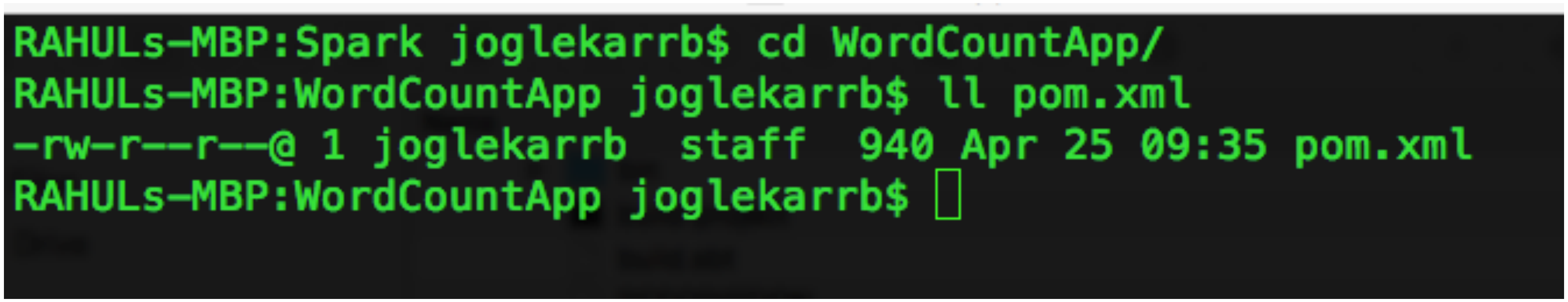
Scala Project with Maven

```
mvn archetype:generate -B \
-DarchetypeGroupId=net.alchim31.maven \
-DarchetypeArtifactId=scala-archetype-simple \
-DarchetypeVersion=1.5 \
-DgroupId=com.company -DartifactId=MyWordCount \
-Dversion=0.1-SNAPSHOT \
-Dpackage=edu.harvard.e63
```

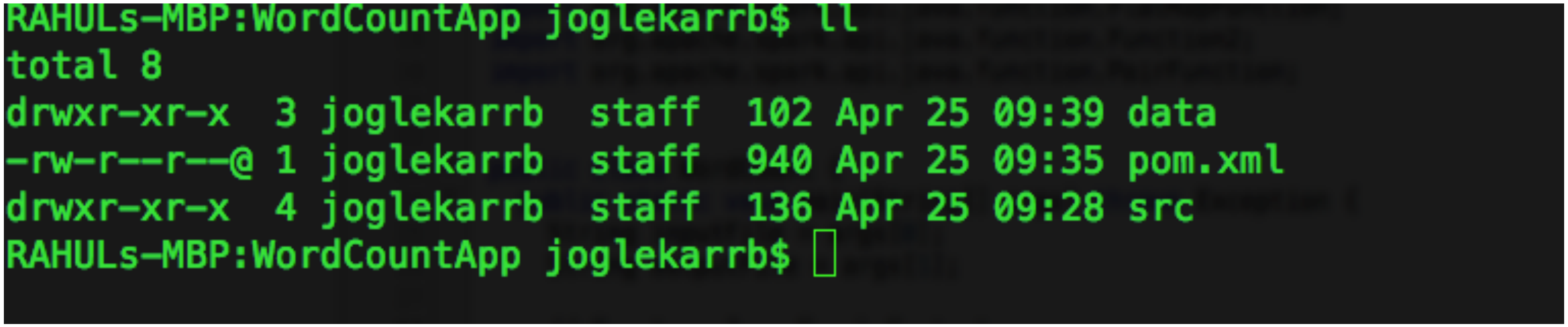
Maven – Java Build



```
RAHULs-MBP:Spark joglekarrb$ mkdir WordCountApp
RAHULs-MBP:Spark joglekarrb$ mkdir -p src/main/java/edu/harvard/e63
RAHULs-MBP:Spark joglekarrb$ ll WordCountApp/src/main/java/edu/harvard/e63
total 16
-rw-r--r--  1 joglekarrb  staff   932 Apr 20 14:43 BasicMap.java
-rw-r--r--@ 1 joglekarrb  staff  1656 Apr 25 09:29 WordCount.java
```

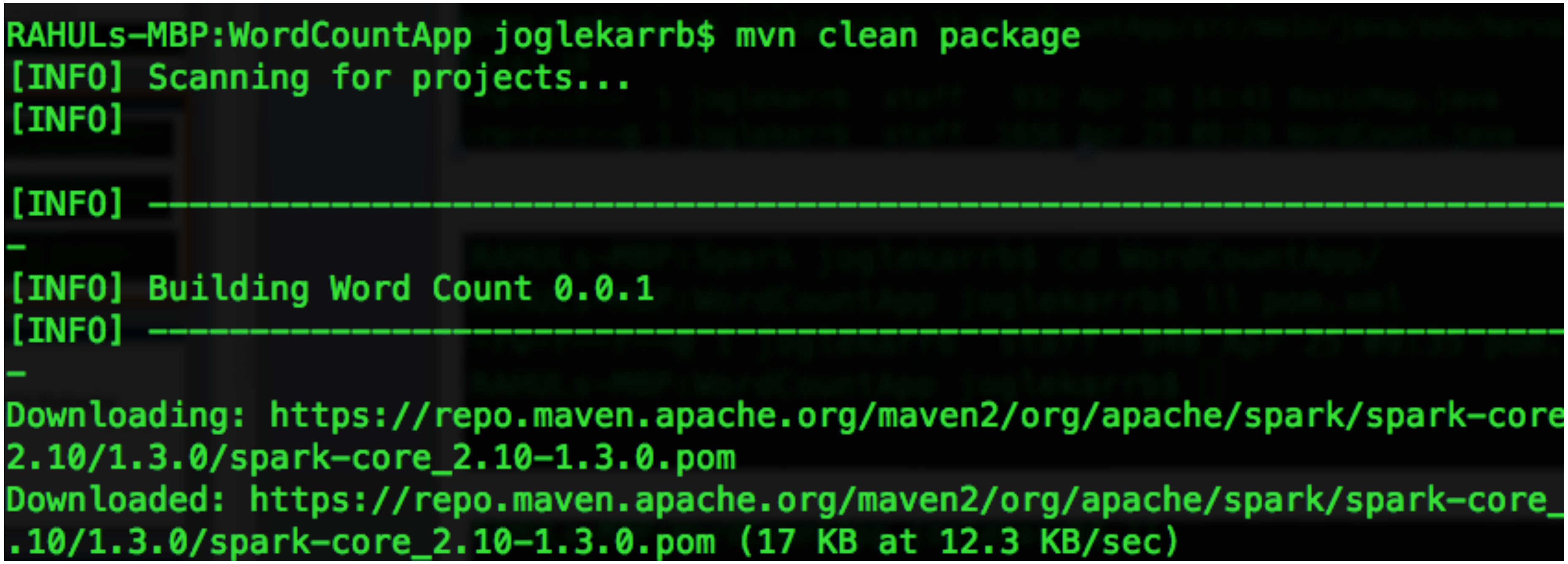


```
RAHULs-MBP:Spark joglekarrb$ cd WordCountApp/
RAHULs-MBP:WordCountApp joglekarrb$ ll pom.xml
-rw-r--r--@ 1 joglekarrb  staff  940 Apr 25 09:35 pom.xml
RAHULs-MBP:WordCountApp joglekarrb$
```

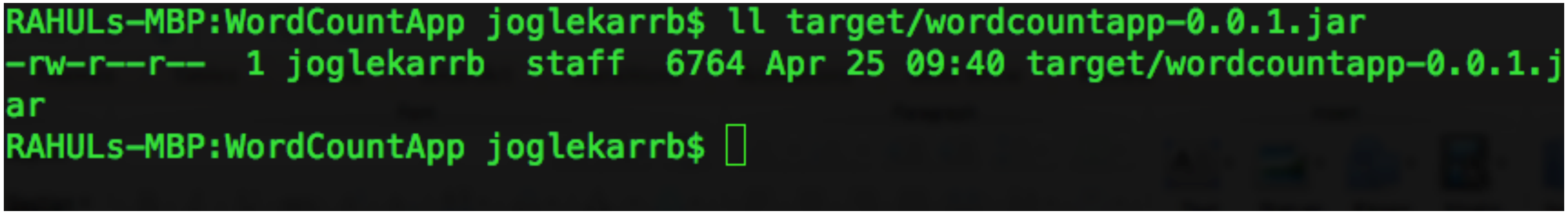


```
RAHULs-MBP:WordCountApp joglekarrb$ ll
total 8
drwxr-xr-x  3 joglekarrb  staff  102 Apr 25 09:39 data
-rw-r--r--@ 1 joglekarrb  staff  940 Apr 25 09:35 pom.xml
drwxr-xr-x  4 joglekarrb  staff  136 Apr 25 09:28 src
RAHULs-MBP:WordCountApp joglekarrb$
```

Java Builds with Maven



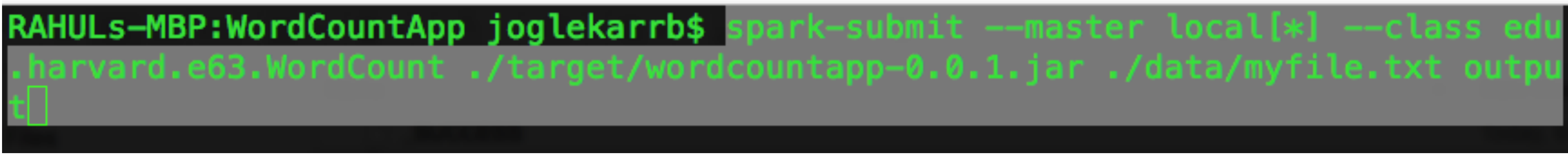
```
RAHULs-MBP:WordCountApp joglekarrb$ mvn clean package
[INFO] Scanning for projects...
[INFO]

[INFO] ------------------------------------------------------------------------
-
[INFO] Building Word Count 0.0.1
[INFO] ------------------------------------------------------------------------
-
Downloading: https://repo.maven.apache.org/maven2/org/apache/spark/spark-core
2.10/1.3.0/spark-core_2.10-1.3.0.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/spark/spark-core_
.10/1.3.0/spark-core_2.10-1.3.0.pom (17 KB at 12.3 KB/sec)
```



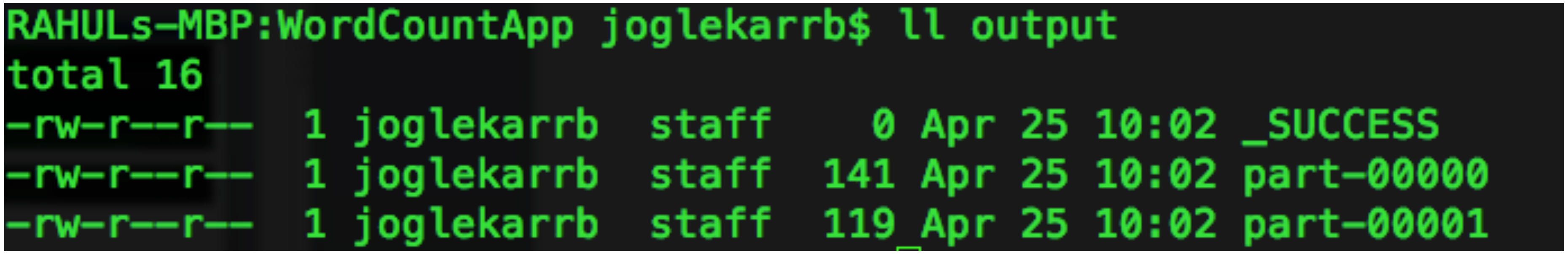
```
RAHULs-MBP:WordCountApp joglekarrb$ ll target/wordcountapp-0.0.1.jar
-rw-r--r--  1 joglekarrb  staff  6764 Apr 25 09:40 target/wordcountapp-0.0.1.j
ar
RAHULs-MBP:WordCountApp joglekarrb$
```

Running the Job

spark-submit --master local[*] --class edu.harvard.e63.WordCount ./target/
e63app-0.0.1.jar ./data/all_bible.txt output



```
RAHULs-MBP:WordCountApp joglekarrb$ spark-submit --master local[*] --class edu
.harvard.e63.WordCount ./target/wordcountapp-0.0.1.jar ./data/myfile.txt outpu
t
```



```
RAHULs-MBP:WordCountApp joglekarrb$ ll output
total 16
-rw-r--r--  1 joglekarrb  staff    0 Apr 25 10:02 _SUCCESS
-rw-r--r--  1 joglekarrb  staff  141 Apr 25 10:02 part-00000
-rw-r--r--  1 joglekarrb  staff  119 Apr 25 10:02 part-00001
```

Notice - master local[*]
Vs
master local[1]

What is the difference ?

References

http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-12.pdf

http://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf

http://blog.cloudera.com/blog/2014/09/how-to-translate-from-mapreduce-to-apache-spark

http://www.cs.berkeley.edu/~matei/papers/2010/hotcloud_spark.pdf

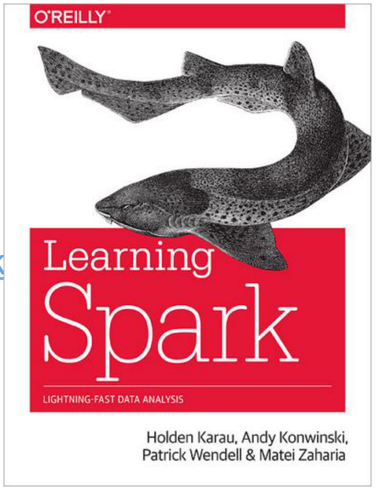
http://shop.oreilly.com/product/0636920028512.do

https://spark.apache.org/docs/latest/

https://spark.apache.org/examples.html

Databricks - https://www.youtube.com/channel/UC3q8O3Bh2Le8Rj1-Q-_UUbA

http://www.dhgarrette.com/nlpclass/notes/


O'REILLY
Learning Spark
LIGHTNING-FAST DATA ANALYSIS
Holden Karau, Andy Konwinski, Patrick Wendell & Matei Zaharia

Thank You