DS 3001: Foundations of Data Science (Spark)

Apdapted from Cloudera and Stanford Univ

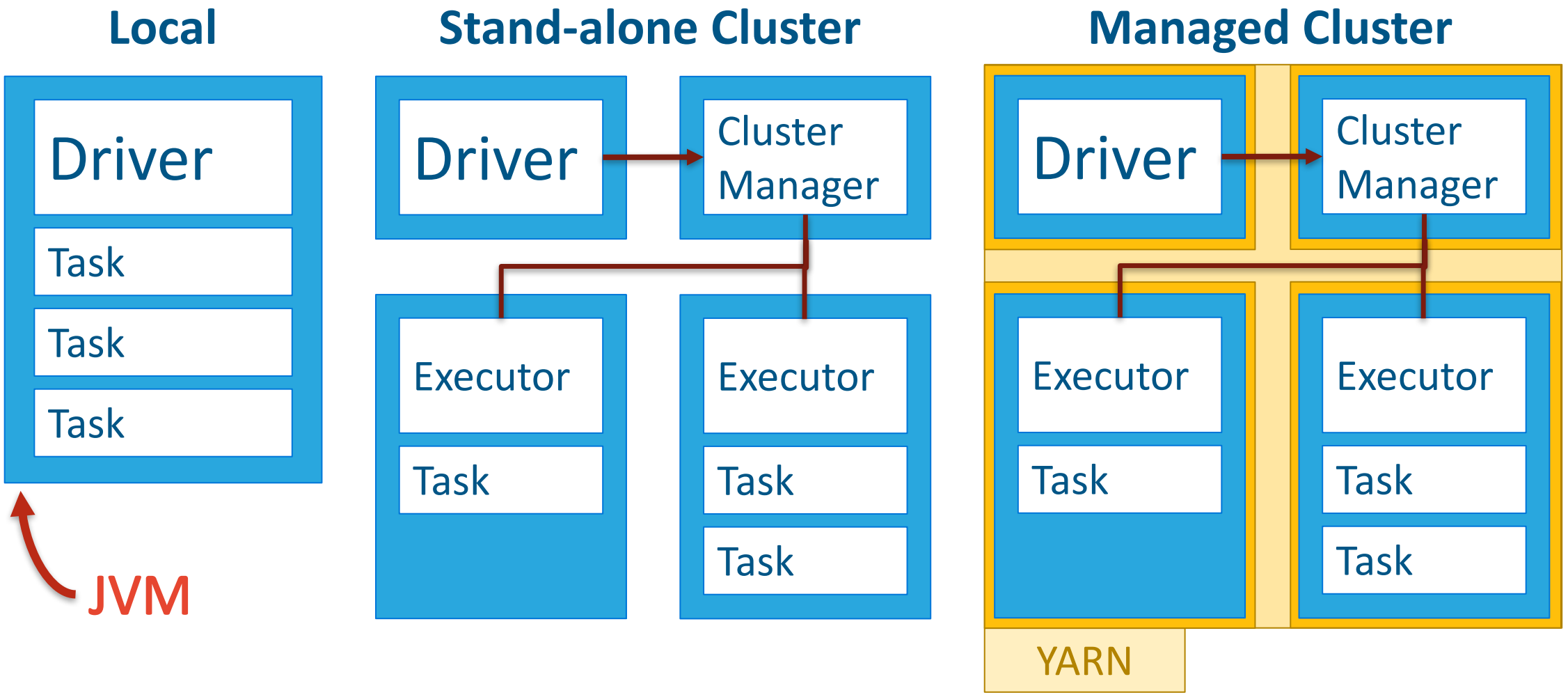
The Plan

- Getting started with Spark
- RDDs
- Commonly useful operations
- Using python
- Using Java
- Using Scala
- Help session

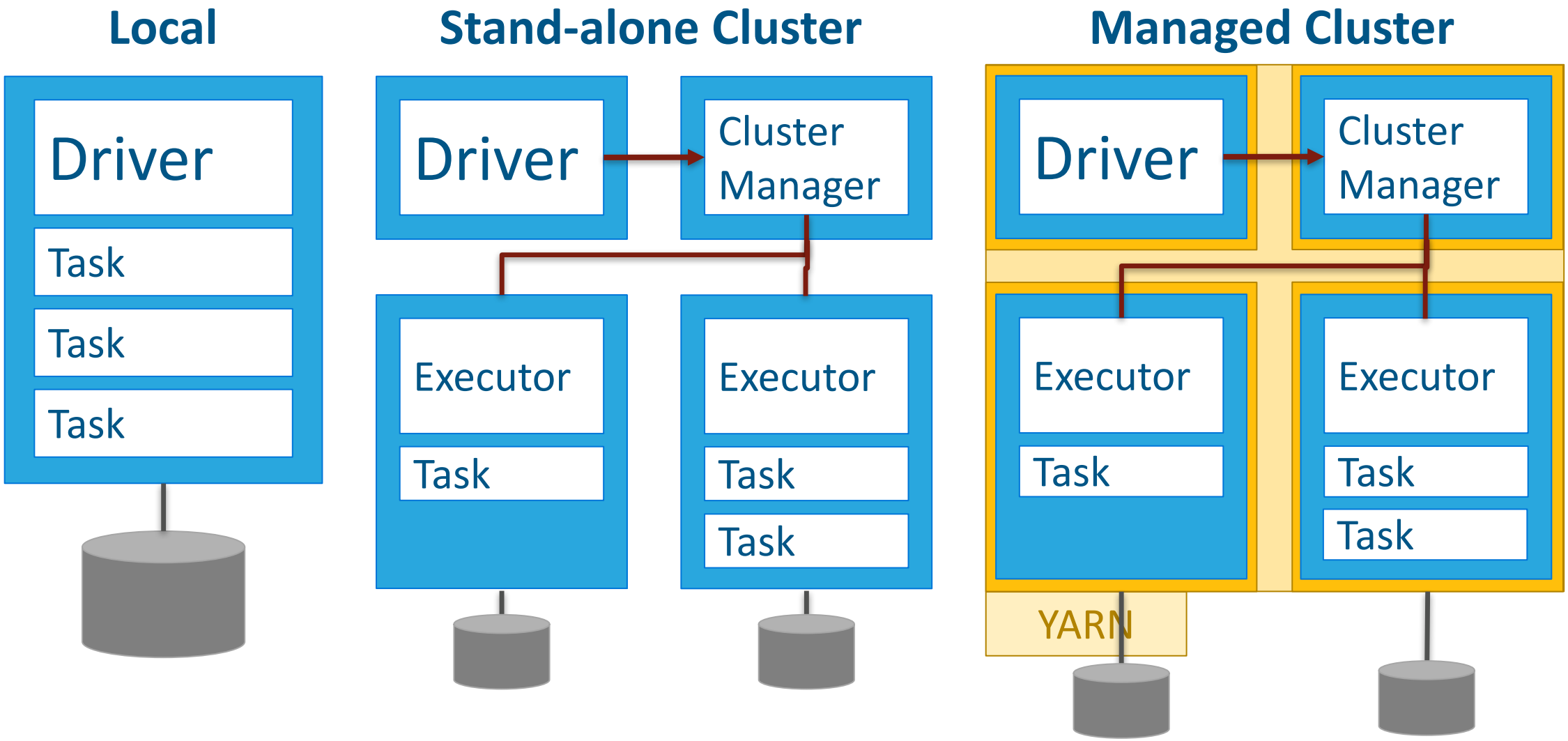
Go download Spark now.
(Version 2.2.1 for Hadoop 2.7)

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

Three Deployment Options



Three Deployment Options



Resilient Distributed Datasets

- Spark is RDD-centric
- RDDs are immutable
- RDDs are computed lazily
- RDDs can be cached
- RDDs know who their parents are
- RDDs that contain only tuples of two elements are “pair RDDs”

Useful RDD Actions

- take(n) – return the first n elements in the RDD as an array.
- collect() – return all elements of the RDD as an array. Use with caution.
- count() – return the number of elements in the RDD as an int.
- saveAsTextFile('path/to/dir') – save the RDD to files in a directory. Will create the directory if it doesn't exist and will fail if it does.
- foreach(func) – execute the function against every element in the RDD, but don't keep any results.

Useful RDD Operations

map()

Apply an operation to every element of an RDD and return a new RDD that contains the results

```
>>> data = sc.textFile('path/to/file')
>>> data.take(3)
[u'Apple,Amy', u'Butter,Bob', u'Cheese,Chucky']
>>> data.map(lambda line: line.split(',')).take(3)
[[u'Apple', u'Amy'], [u'Butter', u'Bob'], [u'Cheese', u'Chucky']]
```

flatMap()

Apply an operation to every element of an RDD and return a new RDD that contains the results after dropping the outermost container

```
>>> data = sc.textFile('path/to/file')
>>> data.take(3)
[u'Apple,Amy', u'Butter,Bob', u'Cheese,Chucky']
>>> data.flatMap(lambda line: line.split(',')).take(6)
[u'Apple', u'Amy', u'Butter', u'Bob', u'Cheese', u'Chucky']
```

mapValues()

Apply an operation to the value of every element of an RDD and return a new RDD that contains the results. Only works with pair RDDs

```
>>> data = sc.textFile('path/to/file')
>>> data = data.map(lambda line: line.split(','))
>>> data = data.map(lambda pair: (pair[0], pair[1]))
>>> data.take(3)
[(u'Apple', u'Amy'), (u'Butter', u'Bob'), (u'Cheese', u'Chucky')]
>>> data.mapValues(lambda name: name.lower()).take(3)
[(u'Apple', u'amy'), (u'Butter', u'bob'), (u'Cheese', u'chucky')]
```

flatMapValues()

Apply an operation to the value of every element of an RDD and return a new RDD that contains the results after removing the outermost container. Only works with pair RDDs

```
>>> data = sc.textFile('path/to/file')
>>> data = data.map(lambda line: line.split(','))
>>> data = data.map(lambda pair: (pair[0], pair[1])).take(3)
>>> data.take(3)
[(u'Apple', u'Amy'), (u'Butter', u'Bob'), (u'Cheese', u'Chucky')]
>>> data.flatMapValues(lambda name: name.lower()).take(3)
[(u'Apple', u'a'), (u'Apple', u'm'), (u'Apple', u'y')]
```

filter()

Return a new RDD that contains only the elements that pass a filter operation

```
>>> import re
>>> data = sc.textFile('path/to/file')
>>> data.take(3)
[u'Apple,Amy', u'Butter,Bob', u'Cheese,Chucky']
>>> data.filter(lambda line: re.match(r'^[AEIOU]', line)).take(3)
[u'Apple,Amy', u'Egg,Edward', u'Oxtail,Oscar']
```

groupByKey()

Apply an operation to the value of every element of an RDD and return a new RDD that contains the results after removing the outermost container. Only works with pair RDDs

```
>>> data = sc.textFile('path/to/file')
>>> data = data.map(lambda line: line.split(','))
>>> data = data.map(lambda pair: (pair[0], pair[1]))
>>> data.take(3)
[(u'Apple', u'Amy'), (u'Butter', u'Bob'), (u'Cheese', u'Chucky')]
>>> data.groupByKey().take(1)
[(u'Apple', <pyspark.resultiterable.ResultIterable object at 0x102ed1290>)]
>>> for pair in data.groupByKey().take(1):
...   print "%s:%s" % (pair[0], ",".join([n for n in pair[1]]))
Apple:Amy,Adam,Alex
```

reduceByKey()

Combine elements of an RDD by key and then apply a reduce operation to pairs of keys until only a single key remains. Return the result in a new RDD.

```
>>> data = sc.textFile('path/to/file')
>>> data = data.map(lambda line: line.split(','))
>>> data = data.map(lambda pair: (pair[0], pair[1]))
>>> data.take(3)
[(u'Apple', u'Amy'), (u'Butter', u'Bob'), (u'Cheese', u'Chucky')]
>>> data.reduceByKey(lambda v1, v2: v1 + ":" + v2).take(1)
[(u'Apple', u'Amy:Alex:Adam')]
```

sortBy()

Sort an RDD according to a sorting function and return the results in a new RDD.

```
>>> data = sc.textFile('path/to/file')
>>> data = data.map(lambda line: line.split(','))
>>> data = data.map(lambda pair: (pair[0], pair[1]))
>>> data.take(3)
[(u'Apple', u'Amy'), (u'Butter', u'Bob'), (u'Cheese', u'Chucky')]
>>> data.sortBy(lambda pair: pair[1]).take(3)
[(u'Avocado', u'Adam'), (u'Anchovie', u'Alex'), (u'Apple', u'Amy')]
```

sortByKey()

Sort an RDD according to the natural ordering of the keys and return the results in a new RDD.

```
>>> data = sc.textFile('path/to/file')
>>> data = data.map(lambda line: line.split(','))
>>> data = data.map(lambda pair: (pair[0], pair[1]))
>>> data.take(3)
[(u'Apple', u'Amy'), (u'Butter', u'Bob'), (u'Cheese', u'Chucky')]
>>> data.sortByKey().take(3)
[(u'Apple', u'Amy'), (u'Anchovie', u'Alex'), (u'Avocado', u'Adam')]
```

subtract()

Return a new RDD that contains all the elements from the original RDD that do not appear in a target RDD.

```
>>> data1 = sc.textFile('path/to/file1')
>>> data1.take(3)
[u'Apple,Amy', u'Butter,Bob', u'Cheese,Chucky']
>>> data2 = sc.textFile('path/to/file2')
>>> data2.take(3)
[u'Wendy', u'McDonald,Ronald', u'Cheese,Chucky']
>>> data1.subtract(data2).take(3)
[u'Apple,Amy', u'Butter,Bob', u'Dinkel,Dieter']
```

join()

Return a new RDD that contains all the elements from the original RDD joined (inner join) with elements from the target RDD.

```
>>> data1 = sc.textFile('path/to/file1').map(lambda line: line.split(',')).map(lambda
pair: (pair[0], pair[1]))
>>> data1.take(3)
[(u'Apple', u'Amy'), (u'Butter', u'Bob'), (u'Cheese', u'Chucky')]
>>> data2 = sc.textFile('path/to/file2').map(lambda line: line.split(',')).map(lambda
pair: (pair[0], pair[1]))
>>> data2.take(3)
[(u'Doughboy', u'Pilsbury'), (u'McDonald', u'Ronald'), (u'Cheese', u'Chucky')]
>>> data1.join(data2).collect()
[(u'Cheese', (u'Chucky', u'Chucky'))]
>>> data1.fullOuterJoin(data2).take(2)
[(u'Apple',(u'Amy', None)), (u'Cheese', (u'Chucky', u'Chucky'))]
```

cloudera

Thank you