TRANSFORM
and
ACTION
with

TRANSFORM
and
ACTION
with
Spark

- made it possible to analyze large data sets, but relied heavily on disk storage (rather than memory) for computation

- not a great solution for calculations requiring multiple passes over the same data or many intermediate steps, due to the need to write to and read from the disk between each step

    - difficult to use for interactive data analysis, the main task data scientists need to do

- Hadoop also suffered from suboptimal support for the additional libraries many data scientists needed, such as SQL and machine learning implementations
- Once the cost of RAM (computer memory) started to drop significantly, augmenting or replacing Hadoop by storing data in-memory quickly emerged as an appealing alternative

- made it possible to analyze large data sets, but relied heavily on disk storage (rather than memory) for computation

- not a great solution for calculations requiring multiple passes over the same data or many intermediate steps, due to the need to write to and read from the disk between each step

  - difficult to use for interactive data analysis, the main task data scientists need to do

- Hadoop also suffered from suboptimal support for the additional libraries many data scientists needed, such as SQL and machine learning implementations
- Once the cost of RAM (computer memory) started to drop significantly, augmenting or replacing Hadoop by storing data in-memory quickly emerged as an appealing alternative

- made it possible to analyze large data sets, but relied heavily on disk storage (rather than memory) for computation

- not a great solution for calculations requiring multiple passes over the same data or many intermediate steps, due to **the need to write to and read from the disk between each step**

    - difficult to use for interactive data analysis, the main task data scientists need to do

- Hadoop also suffered from suboptimal support for the additional libraries many data scientists needed, such as SQL and machine learning implementations
- Once the cost of RAM (computer memory) started to drop significantly, augmenting or replacing Hadoop by storing data in-memory quickly emerged as an appealing alternative

- made it possible to analyze large data sets, but relied heavily on disk storage (rather than memory) for computation

- not a great solution for calculations requiring multiple passes over the same data or many intermediate steps, due to the need to write to and read from the disk between each step

    - difficult to use for interactive data analysis, the main task data scientists need to do

- Hadoop also suffered from suboptimal support for the additional libraries many data scientists needed, such as SQL and machine learning implementations
- Once the cost of RAM (computer memory) started to drop significantly, augmenting or replacing Hadoop by storing data in-memory quickly emerged as an appealing alternative

- made it possible to analyze large data sets, but relied heavily on disk storage (rather than memory) for computation

- not a great solution for calculations requiring multiple passes over the same data or many intermediate steps, due to the need to write to and read from the disk between each step

    - difficult to use for interactive data analysis, the main task data scientists need to do

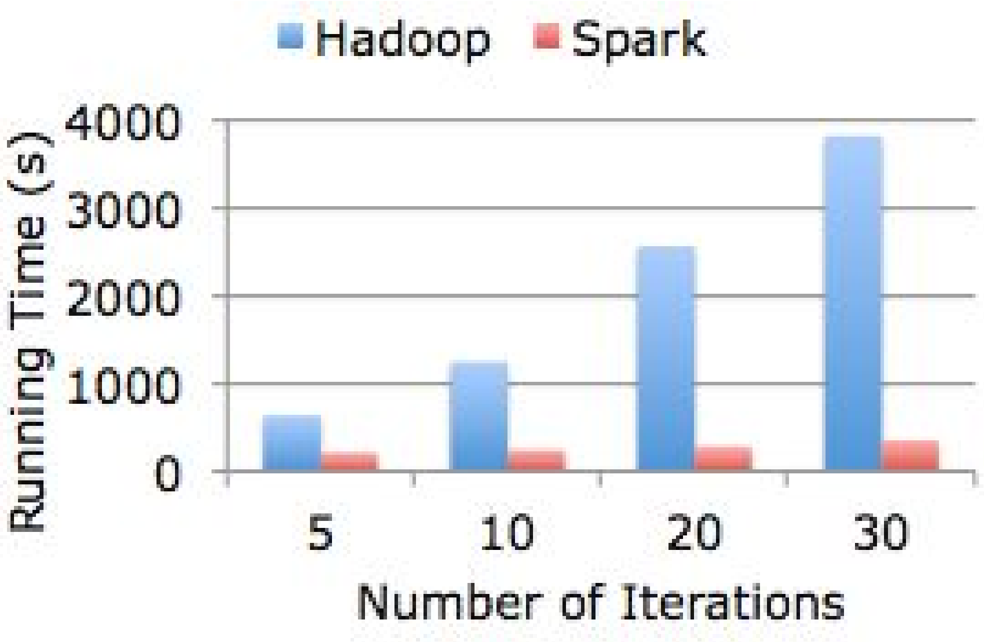
- Hadoop also suffered from suboptimal support for the additional libraries many data scientists needed, such as SQL and machine learning implementations
- Once the cost of RAM (computer memory) started to drop significantly, augmenting or replacing Hadoop by storing data in-memory quickly emerged as an appealing alternative

- made it possible to analyze large data sets, but relied heavily on disk storage (rather than memory) for computation

- not a great solution for calculations requiring multiple passes over the same data or many intermediate steps, due to the need to write to and read from the disk between each step

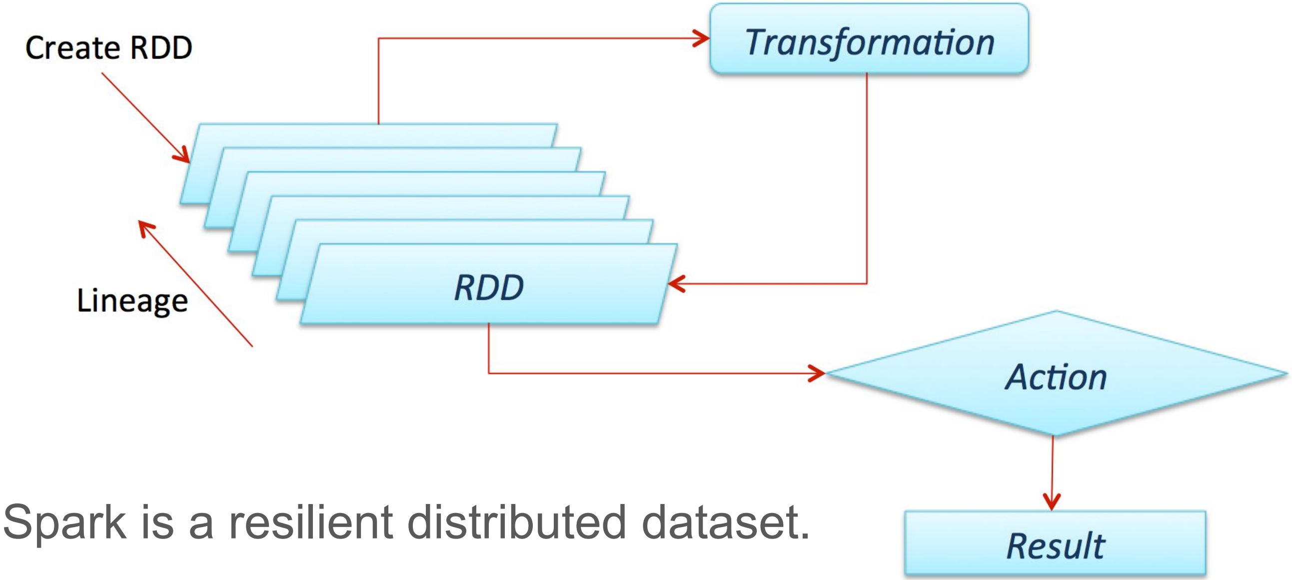
    - difficult to use for interactive data analysis, the main task data scientists need to do

- Hadoop also suffered from suboptimal support for the additional libraries many data scientists needed, such as SQL and machine learning implementations
- Once the cost of RAM (computer memory) started to drop significantly, augmenting or replacing Hadoop by storing data in-memory quickly emerged as an appealing alternative



Spark uses distributed, in-memory data structures to improve speeds for many data processing workloads by several orders of magnitude

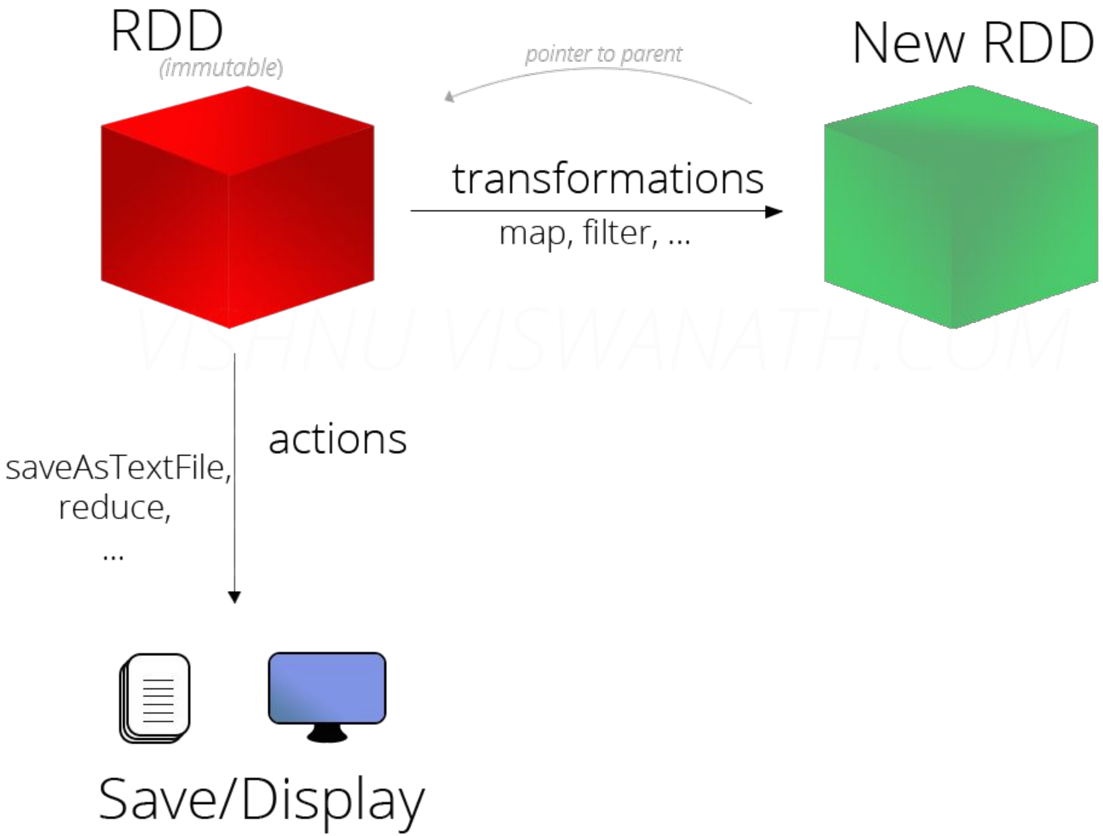
RDD



The core data structure in Spark is a resilient distributed dataset.

RDD is Spark's representation of a data set that's distributed across the RAM, or memory, of a cluster of many machines.

An RDD object is essentially a collection of elements we can use to hold lists of tuples, dictionaries, lists, etc.

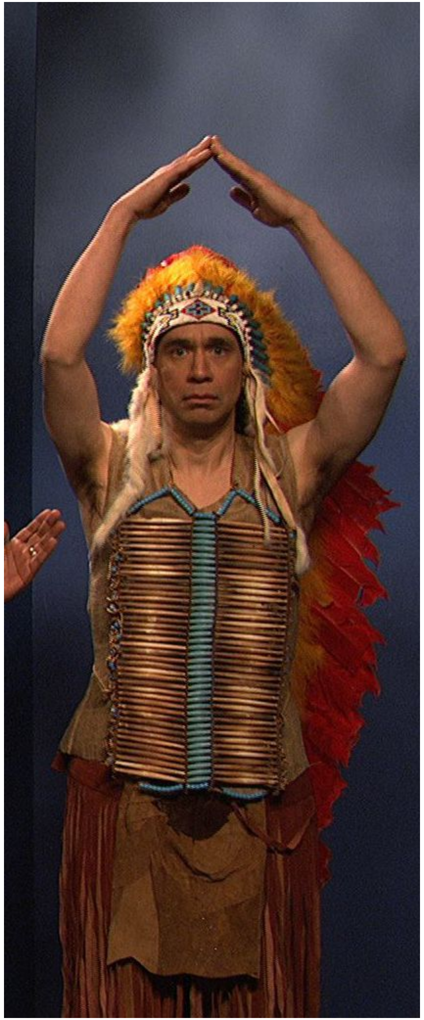

RDD
(immutable)
pointer to parent
New RDD
transformations
map, filter, ...
actions
saveAsTextFile,
reduce,
...
Save/Display

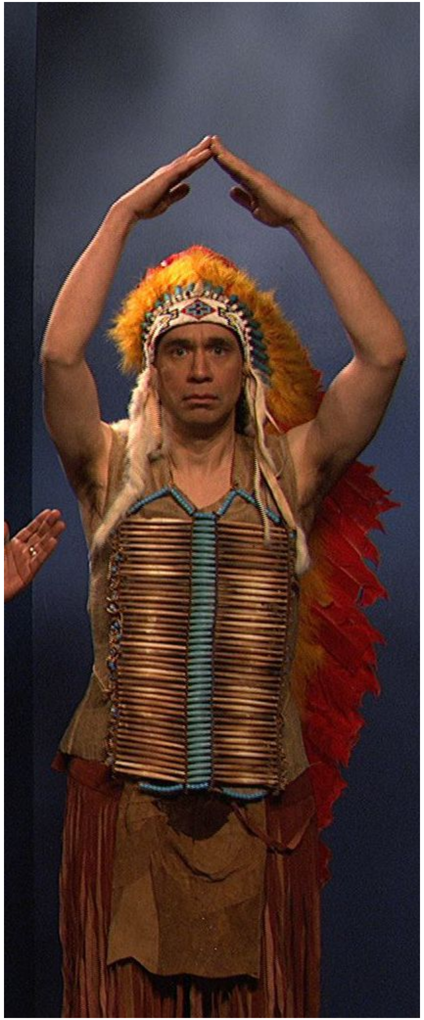
Transformations

- create a new dataset from an existing one
- **map -** a transformation that passes each dataset element through a function and returns a new RDD representing the results
- all transformations in Spark are lazy, in that they do not compute their results right away

Actions

- which return a value to the driver program after running a computation on the dataset
- **reduce** - an action that aggregates all the elements of the RDD using some function and returns the final result to the driver program





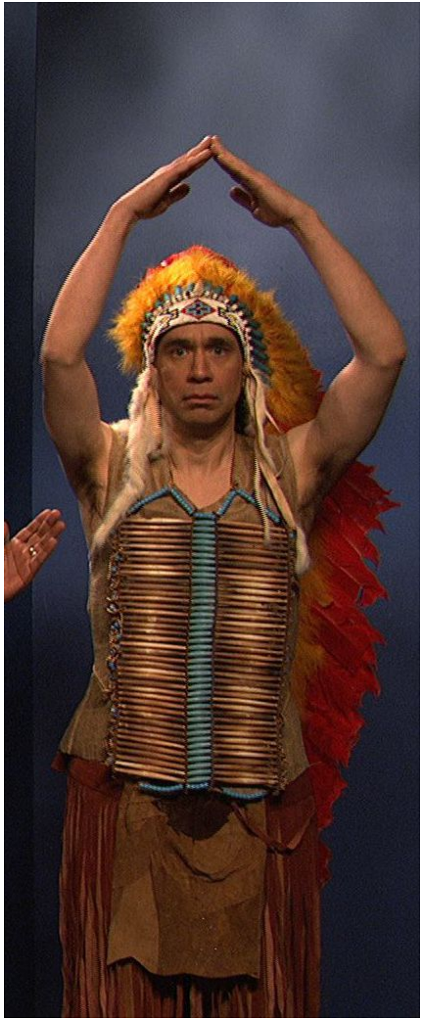
ACTION

reduce
collect
count
first
take
takeSample
takeOrdered
saveAsTextFile
saveAsSequenceFile
saveAsObjectFile
countByKey
foreach


©2015
Neil Keleher
Yoga 4 Smart People
ensational-yoga-poses.com

TRANSFORM

map
filter
flatMap
mapPartitions
mapPartitionsWithIndex
sample
union
intersection
distinct
groupByKey
reduceByKey
aggregateByKey
sortByKey
join
cogroup
cartesian
pipe
coalesce
repartition
repartitionAndSortWithinPartitions

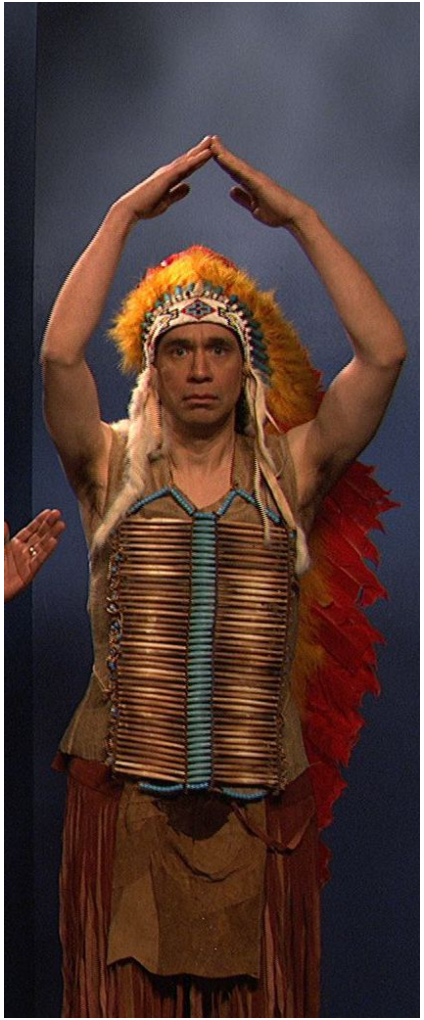


ACTION

reduce
collect
count
first
take
takeSample
takeOrdered
saveAsTextFile
saveAsSequenceFile
saveAsObjectFile
countByKey
foreach

TRANSFORM

map
filter
flatMap
mapPartitions
mapPartitionsWithIndex
sample
union
intersection
distinct
groupByKey
reduceByKey
aggregateByKey
sortByKey
join
cogroup
cartesian
pipe
coalesce
repartition
repartitionAndSortWithinPartitions

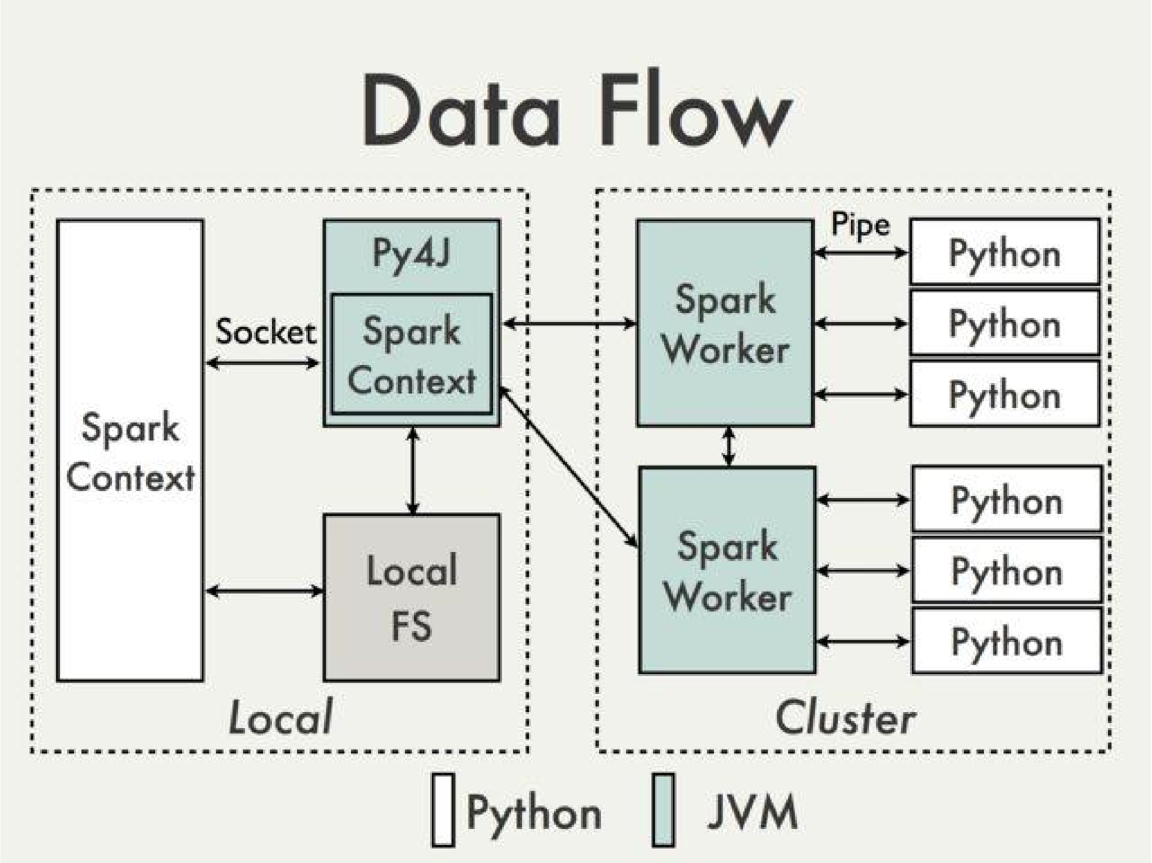


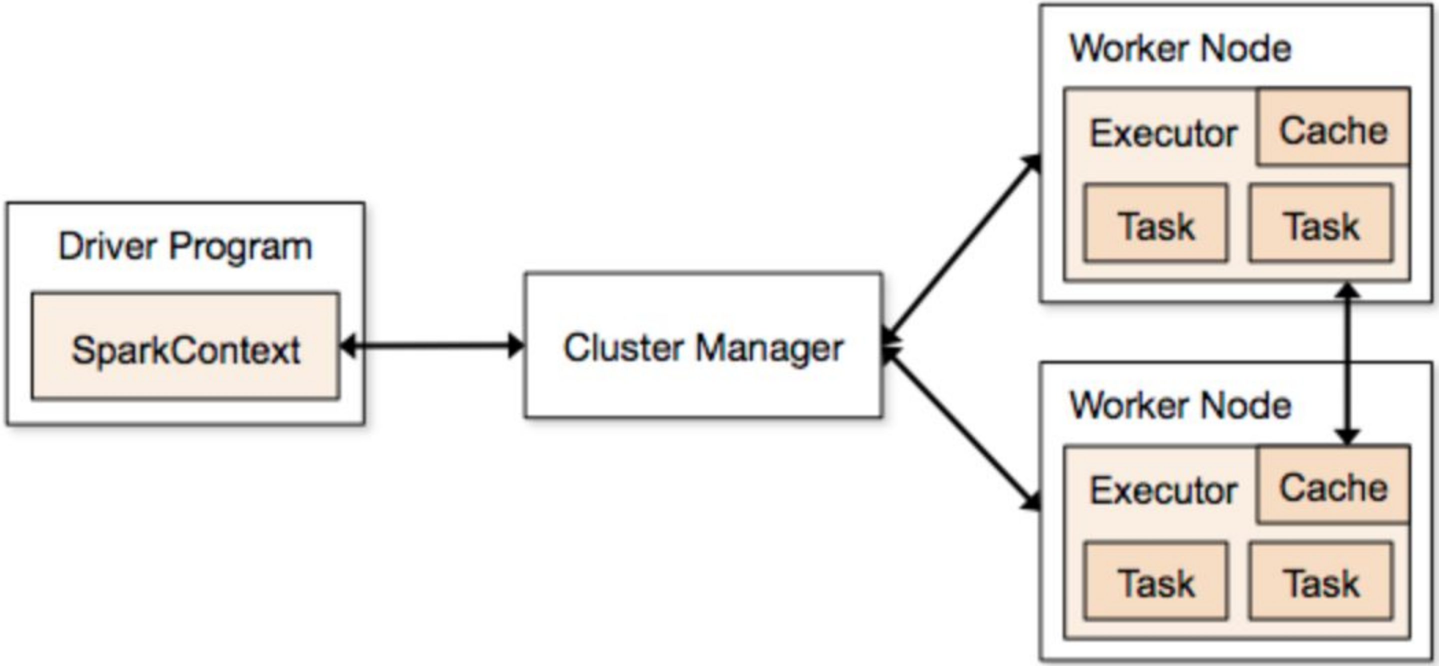

©2015
Neil Keleher
Yoga 4 Smart People
sensational-yoga-poses.com

PySpark

PySpark is built on top of Spark's Java API.

Data is processed in Python and cached / shuffled in the JVM





In Spark, the SparkContext object manages the connection to the clusters, and coordinates the running of processes on those clusters https://github.com/gSchool/DSI_Lectures/blob/master/spark/ryan_henning/Intro%20to%20Spark.pdf

https://spark.apache.org/docs/1.1.1/api/python/pyspark.rdd.RDD-class.html#take