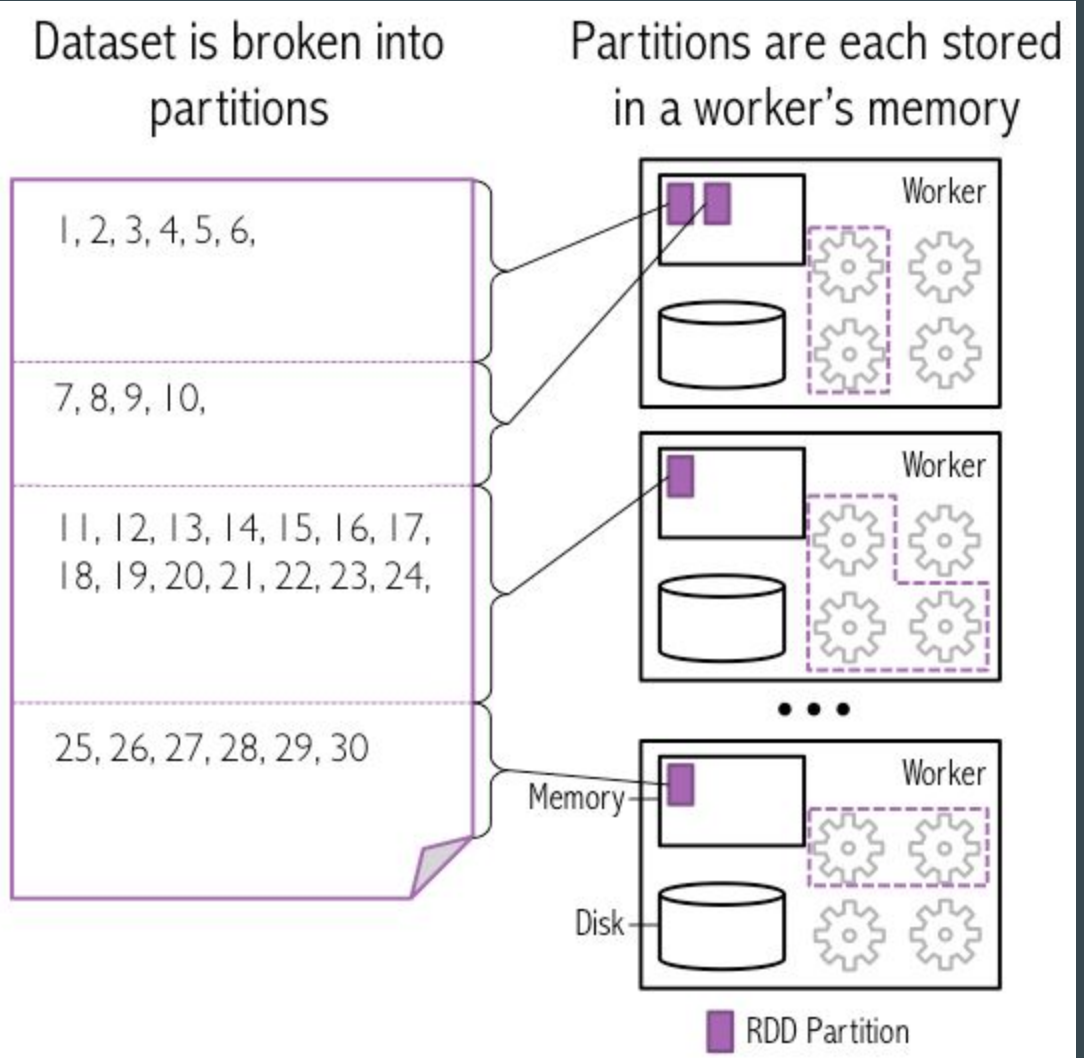
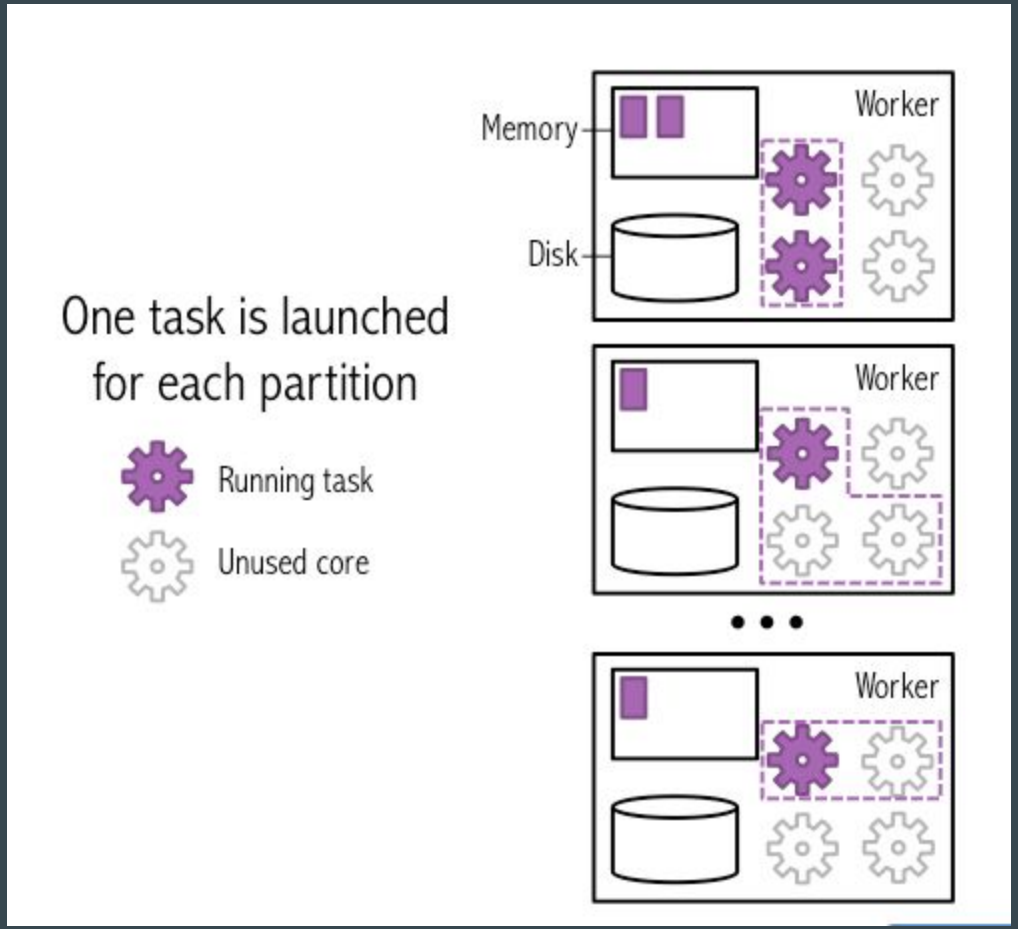
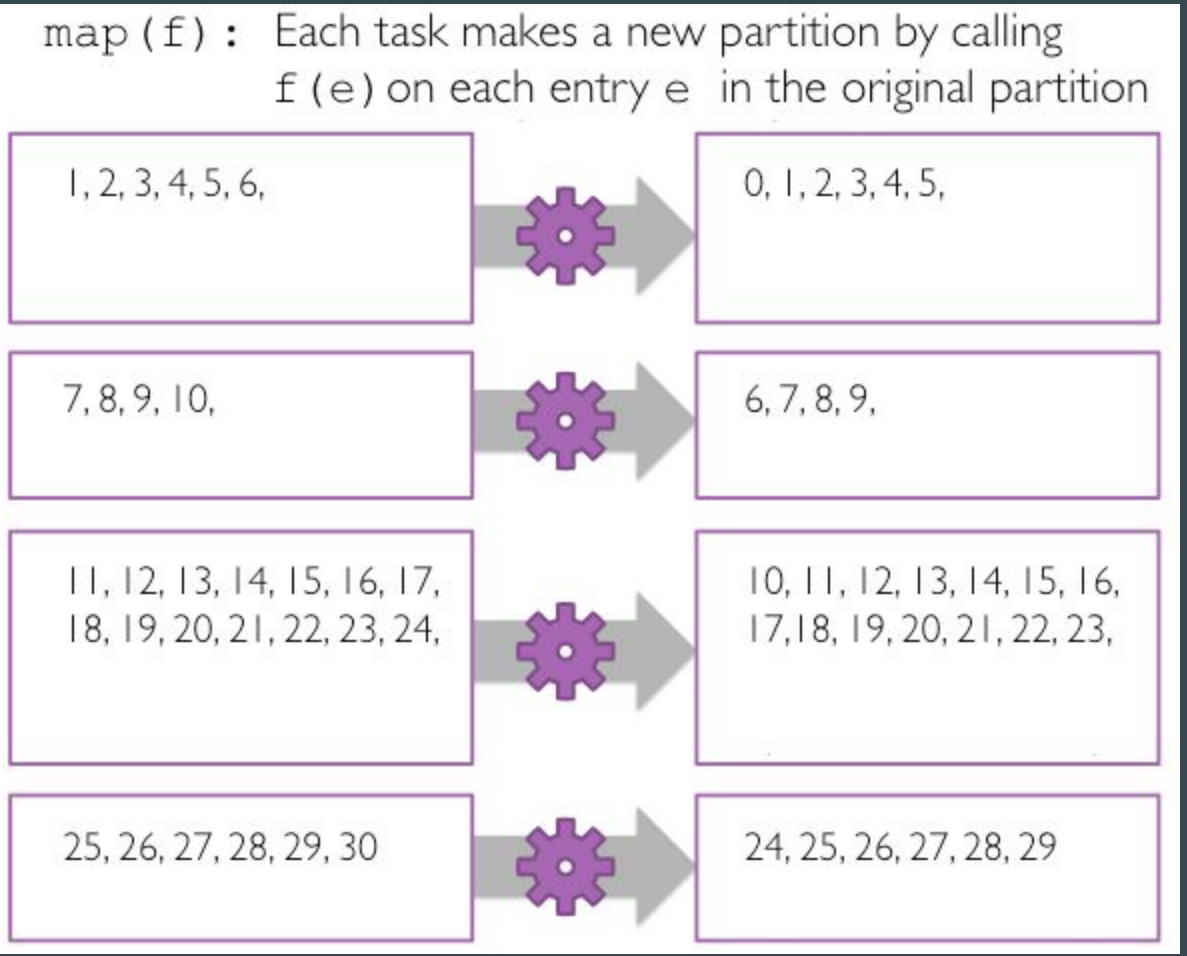
RDD Operations

Transformation and Action Zoo

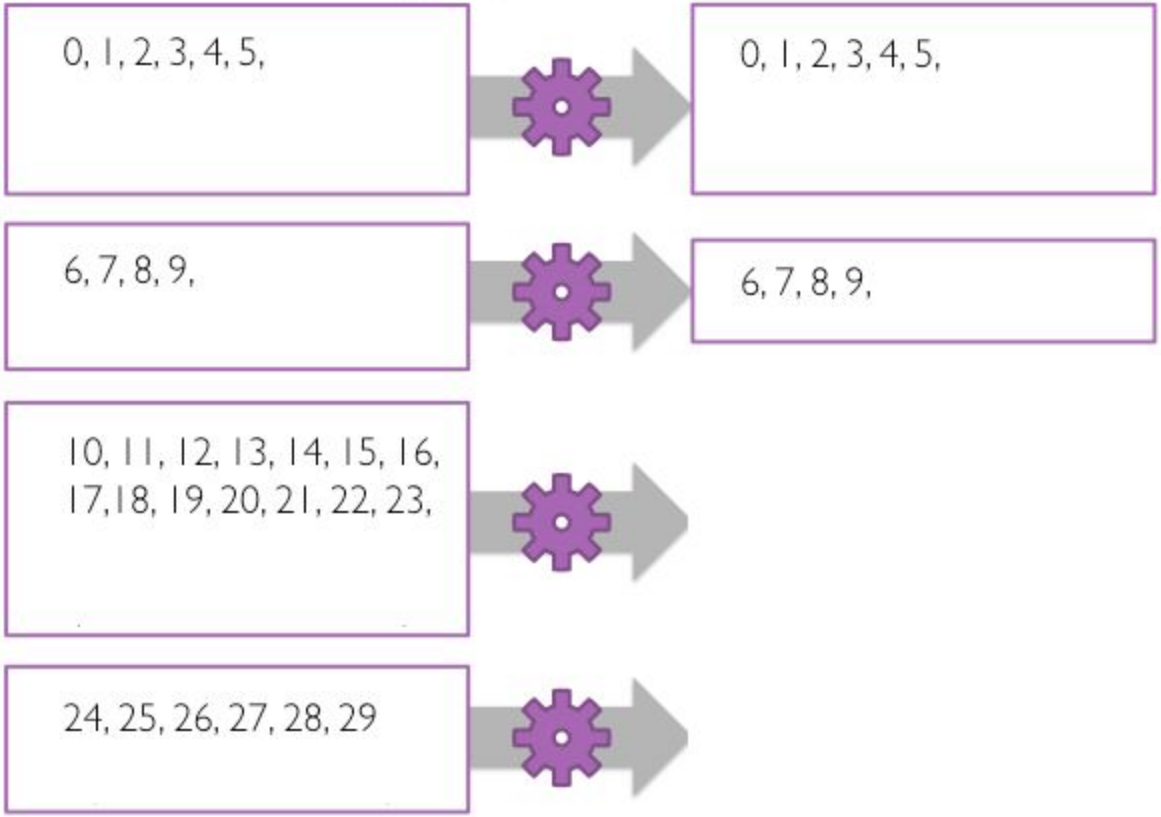

Dataset is broken into partitions
Partitions are each stored in a worker's memory
1, 2, 3, 4, 5, 6,
7, 8, 9, 10,
11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29, 30
Worker
Worker
Worker
Memory
Disk
RDD Partition


One task is launched
for each partition
Running task
Unused core
Memory
Disk
Worker
Worker
Worker

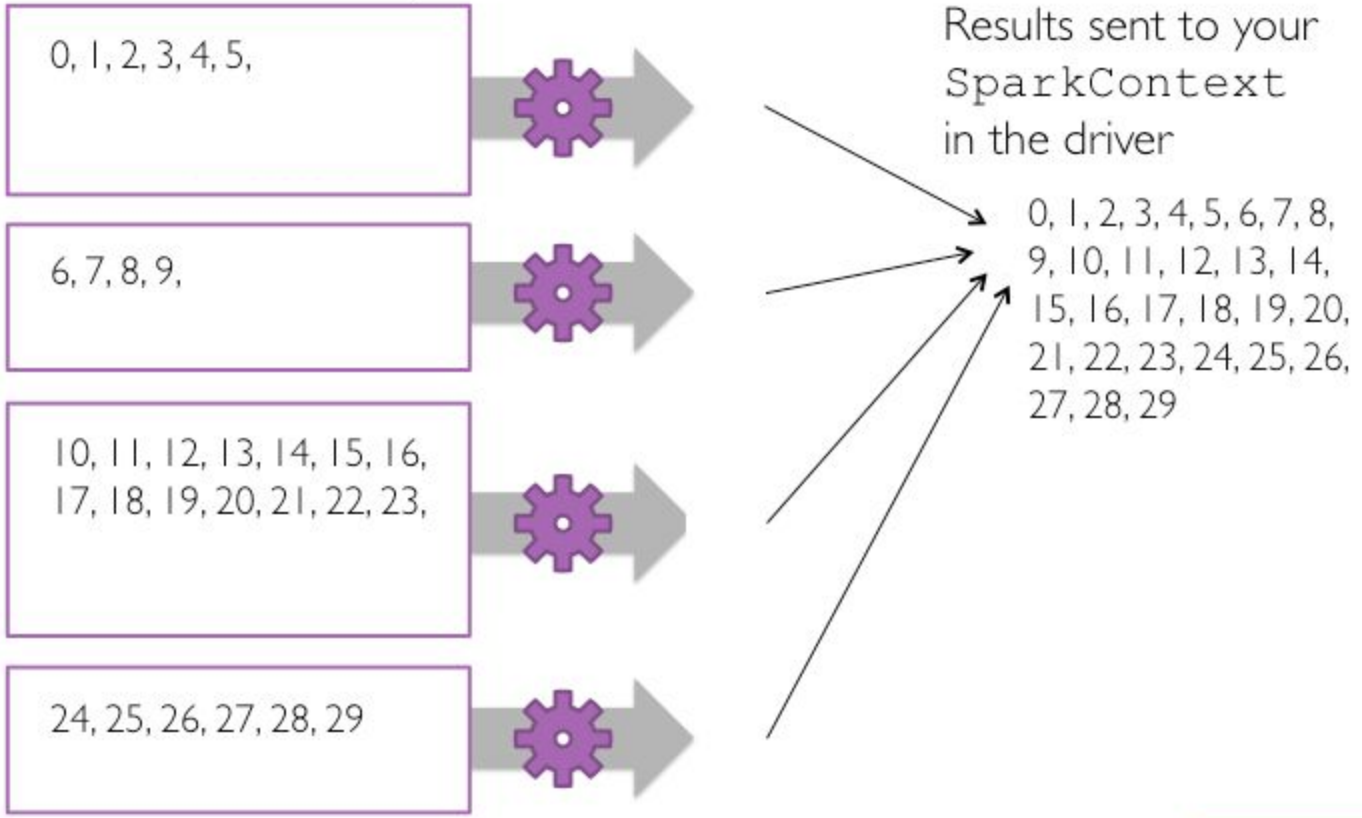
`map(f)`: Each task makes a new partition by calling `f(e)` on each entry `e` in the original partition



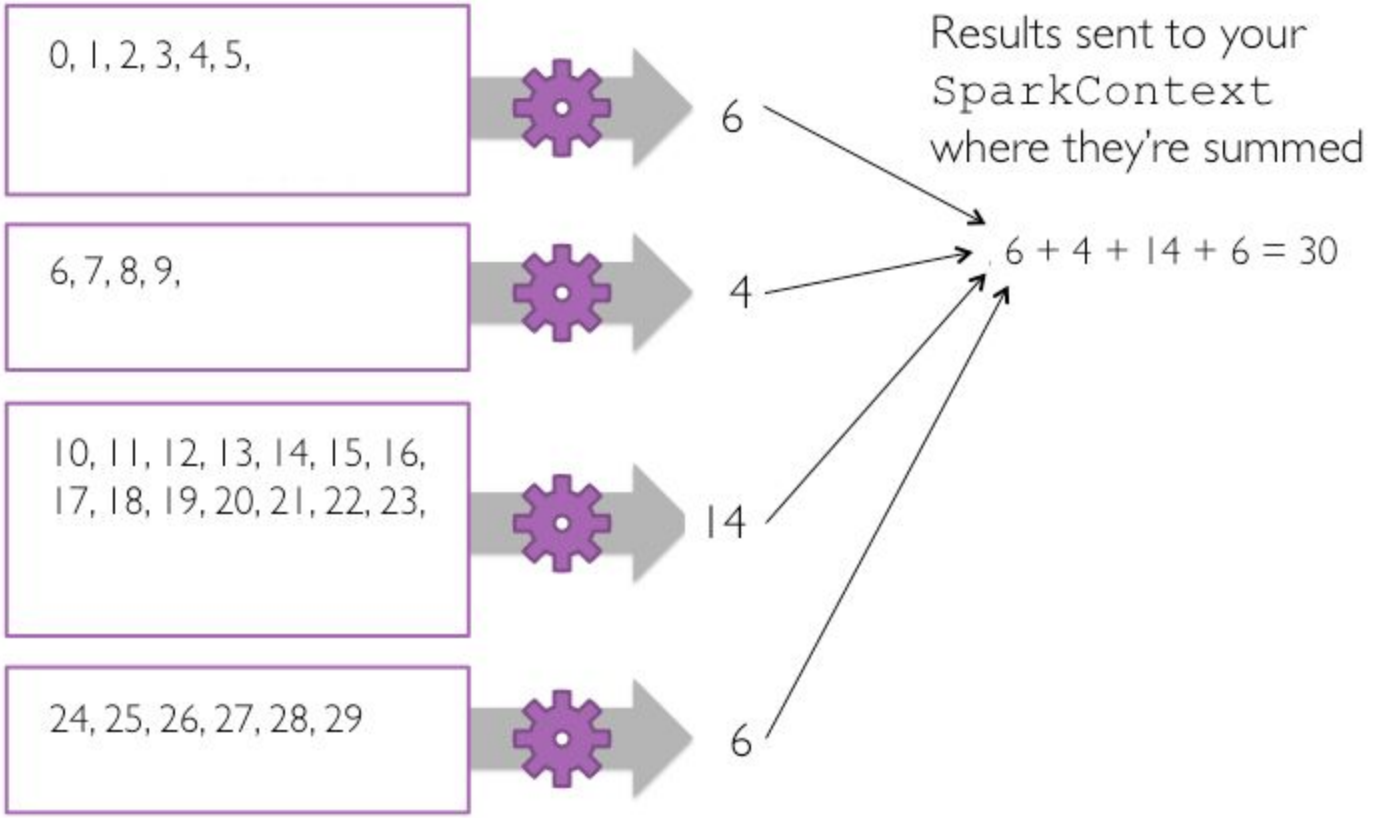
`filter()`: Each task makes a new partition with the entries in the original partition that have a value less than 10



`collect()`: Gathers the entries from all partitions into the driver



`count()`: Each task counts the entries in one partition



Operations on RDD

Transformations and Actions

Once created, RDDs offer two types of operations:

- Transformations (akin to map)
- Actions (akin to reduce)

Laziness

Transformation

Transformations construct a new RDD from a previous one

Examples

- Applying functions on each element of an RDD
- Filtering data that matches a predicate

Action

Actions compute a result based on an RDD

- Result is either
  - returned to the driver program, or
  - saved to an external storage system

Examples

- return the first element of an RDD
- return the RDD as a Python object

Laziness

Why is it useful?

```
lines = sc.textFile("large_textfile")
lines.first()
```

Don’t need to read the whole file to extract the first line.

Exploring RDDs

Hands-on Activity

Spark_Activities_02_Transformations_and_Actions.ipynb

Activity 2

Transformations

List of common Transformations

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

Spark_Activities_02_Transformations_and_Actions.ipynb

Activity 3

map()

Elementwise mapping

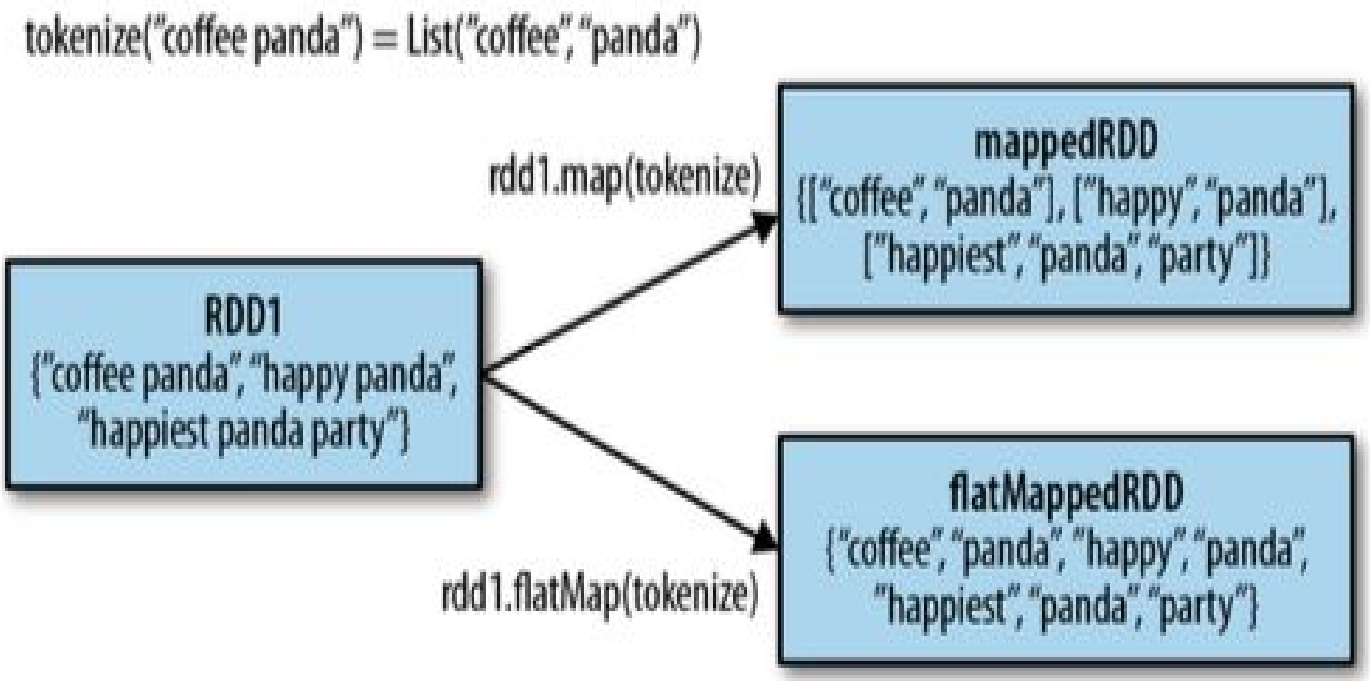
Can be performed on all RDDs

- Applies given function to each element of an RDD (maps existing value to new values) and returns the new RDD
- Return type doesn't have to be the same as input type

flatMap()

multiple output elements for each input element

- Instead of returning a single element, it returns an iterator with our return values
- Rather than producing an RDD of iterators, it returns an RDD that consists of the elements from all of the iterators

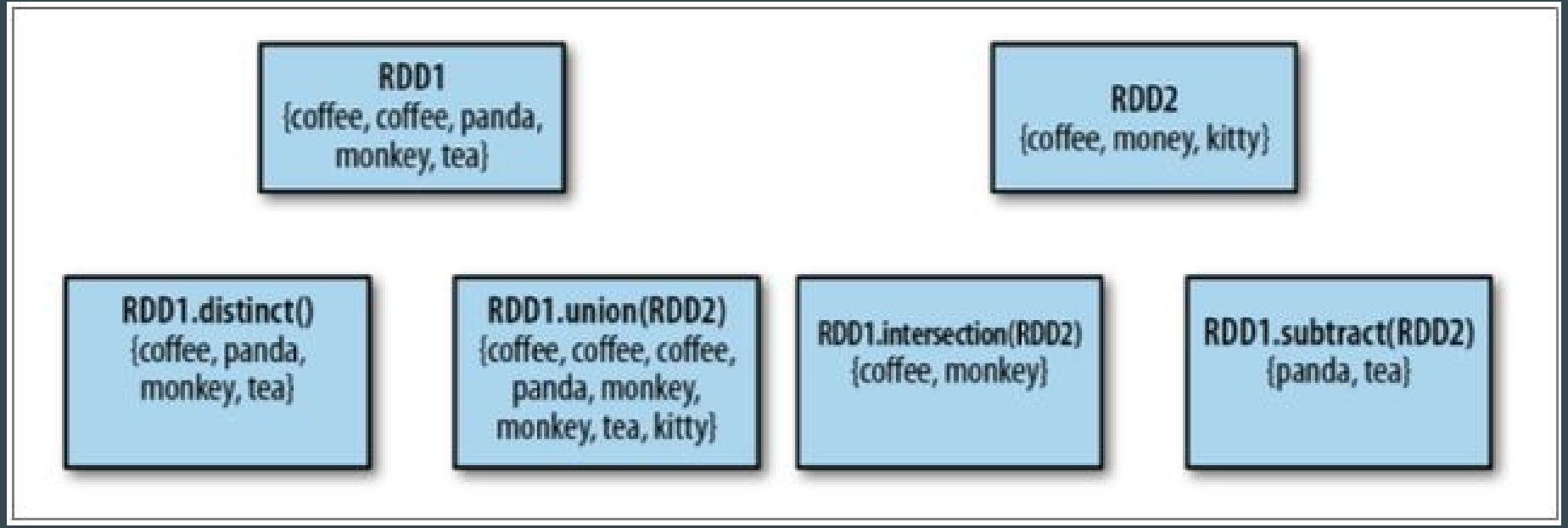

tokenize("coffee panda") = List("coffee", "panda")
rdd1.map(tokenize)
mappedRDD
{["coffee", "panda"], ["happy", "panda"],
["happiest", "panda", "party"]}
RDD1
{"coffee panda", "happy panda",
"happiest panda party"}
rdd1.flatMap(tokenize)
flatMappedRDD
{"coffee", "panda", "happy", "panda",
"happiest", "panda", "party"}

Set Operations

- Both RDD have to be of the same type
- The RDDs don’t have to ve sets themselves

- distinct()
- union()
- intersection()
- subtract()

Let’s draw quick Venn Diagrams


RDD1
{coffee, coffee, panda, monkey, tea}
RDD2
{coffee, money, kitty}
RDD1.distinct()
{coffee, panda, monkey, tea}
RDD1.union(RDD2)
{coffee, coffee, coffee, panda, monkey, monkey, tea, kitty}
RDD1.intersection(RDD2)
{coffee, monkey}
RDD1.subtract(RDD2)
{panda, tea}

filter()

Filter out unwanted elements

Can be performed on all RDDs

- Returns an RDD that only has elements that pass the filter() function (the argument)
- Are input RDD type and output RDD type will always be the same?

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}`
`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}`
`filter()`	Return an RDD consisting of only elements that pass the condition passed to `filter()`.	`rdd.filter(x => x != 1)`	`{2, 3, 3}`
`distinct()`	Remove duplicates.	`rdd.distinct()`	`{1, 2, 3}`
`sample(withReplacement, fraction, [seed])`	Sample an RDD, with or without replacement.	`rdd.sample(false, 0.5)`	Nondeterministic

Function name	Purpose	Example	Result
`union()`	Produce an RDD containing elements from both RDDs.	`rdd.union(other)`	`{1, 2, 3, 3, 4, 5}`
`intersection()`	RDD containing only elements found in both RDDs.	`rdd.intersection(other)`	`{3}`
`subtract()`	Remove the contents of one RDD (e.g., remove training data).	`rdd.subtract(other)`	`{1, 2}`
`cartesian()`	Cartesian product with the other RDD.	`rdd.cartesian(other)`	`{(1, 3), (1, 4), … (3,5)}`

Actions

List of Common Actions

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

reduce()

takes a function that operates on two elements of the type in input RDD and returns a new element of the same type

Example

- Add, Multiply etc
- Paste (strings)
- Cumulative Sum

collect()

- The dataset must fit into a single machine

first()

- Returns the first element

take()

- Order may not be preserved

More Operation

Numeric RDD Operations

- stats()
- max()
- mean()
- min()
- stdev()

Spark_Activities_02_Transformations_and_Actions.ipynb

Activity 5

Exercises

Exercises

Spark_Activities_02_Transformations_and_Actions.ipynb

Activity 6