Concept & Examples of pyspark

Apache Spark?

Roughly speaking,

Spark = RDD + Interface

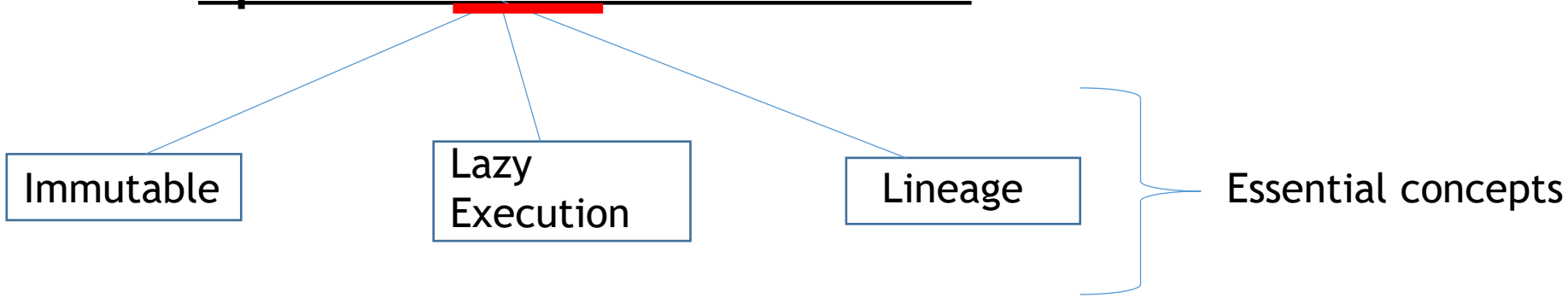
→ Support RDD-based processing interface

This ppt is based on pyspark

Apache Spark?

Roughly speaking,

Spark = RDD + Interface



What is RDD?

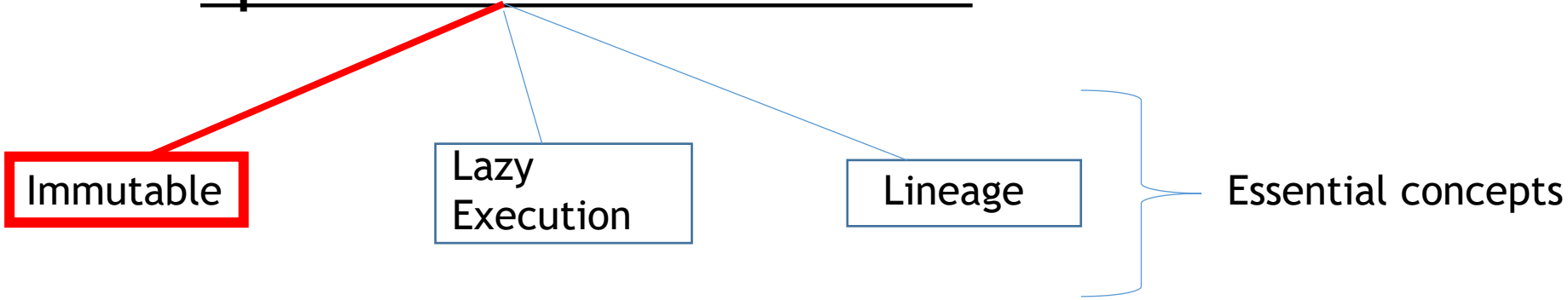
- RDD(Resilient Distributed Dataset)
  - Resilient : If data in memory is lost, it can be recreated
  - Distributed : Stored in memory across the cluster
  - Dataset : Initial data can come from a file or be created programmatically

Fundamental data unit of data in Spark

Apache Spark?

Roughly speaking,

Spark = RDD + Interface

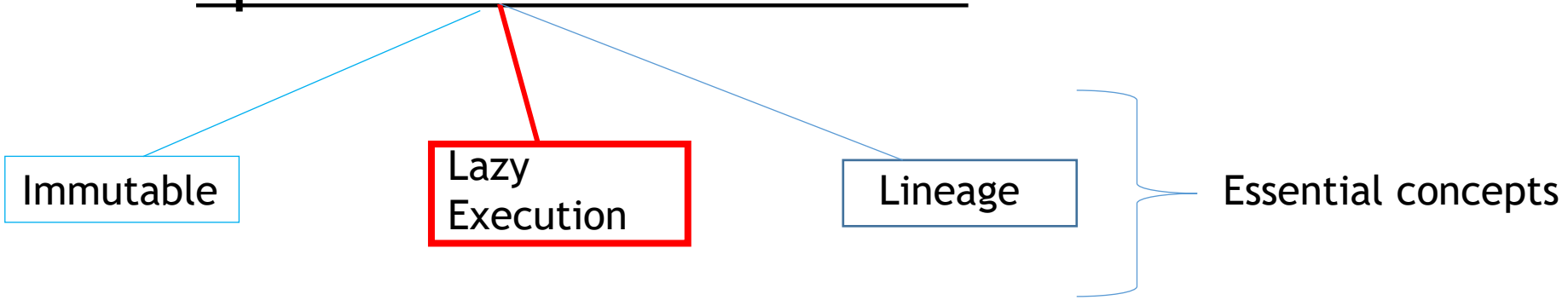


RDD cannot be modified but newly generated

Apache Spark?

Roughly speaking,

Spark = RDD + Interface

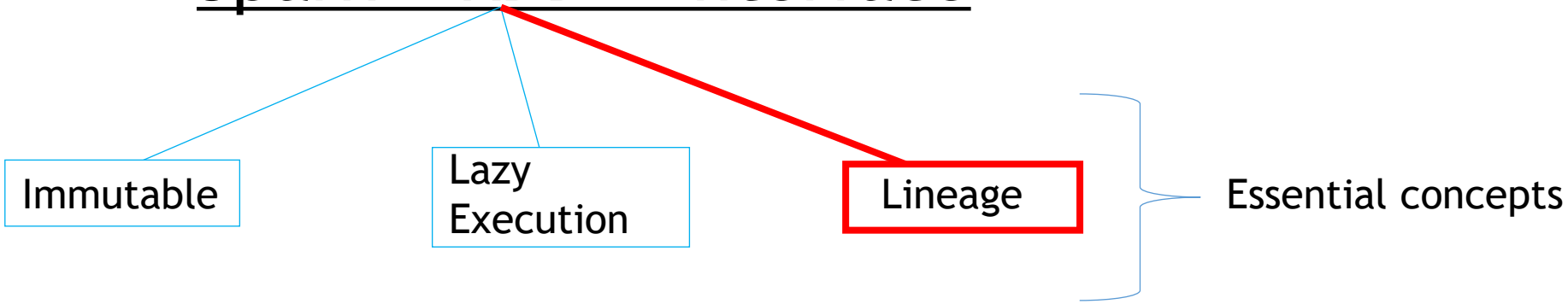


RDD processing doesn't occur until action operation comes

Apache Spark?

Roughly speaking,

Spark = RDD + Interface



Fault tolerant policy of Spark system(logging on lineage)

Creating RDD

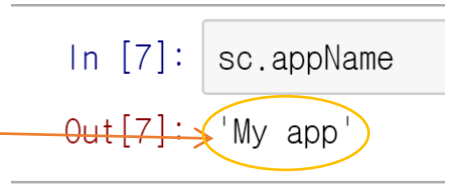
- RDD can be created from

  - A file or set of files(.txt, .json, .csv, object file etc.)
  - Data in memory
  - From another RDD

Creating RDD with pyspark

- Initializing pyspark

```
from pyspark import SparkConf, SparkContext
conf = (SparkConf()
         .setMaster("local")
         .setAppName("My app")
         .set("spark.executor.memory", "1g"))
sc = SparkContext(conf = conf)
```



Configured appName

➡ Define configuration option and create a SparkContext

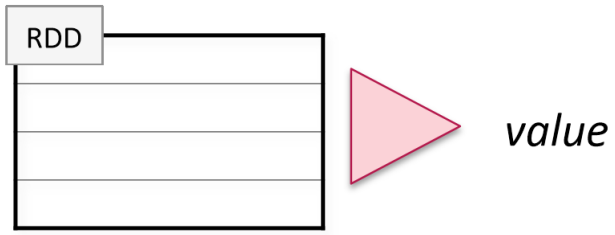
Creating RDD with pyspark

- Create RDD with the text file “purplecow.txt” from local
- Print out the lines of the number of lines of the “purplecow.txt”

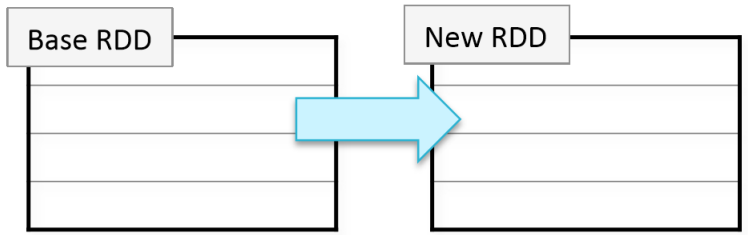
```
In [8]: mydata = sc.textFile("purplecow.txt")

In [11]: mydata.count()
Out[11]: 4
```

RDD Operations

- Two types of RDD Operations
  - Actions : return values
  - Transformations : Define a new RDD based on the current ones



[Actions]



[Transformations]

Action Operations with pyspark

- Action operations : operations that return values.
- count : return the number of elements
- take(n) : return an array of the first n-elements

```
In [13]: mydata.count()
Out[13]: 4

In [20]: mydata.take(2)
Out[20]: ["I've never seen a purple cow.", 'I never hope to see one;']

In [21]: for line in mydata.take(2):
             print(line)

I've never seen a purple cow.
I never hope to see one;
```

Transformations Operations with pyspark

- Transformation opeations : create a new RDD from an existing one
- map(function) : creates a new RDD by performing a function on each record in the base RDD
- filter(function) : creates a new RDD by including or excluding each record in the base RDD according to a boolean function

```
In [23]: mydata2 = mydata.map(lambda line: line.upper())

In [26]: mydata3 = mydata2.filter(lambda line: line.startswith('I'))

In [34]: for line in mydata3.take(mydata3.count()):
             print(line)

I'VE NEVER SEEN A PURPLE COW.
I NEVER HOPE TO SEE ONE;
I'D RATHER SEE THAN BE ONE.
```

Lazy Execution

- Data in RDDs is not processed until an action is performed

```
In [1]: from pyspark import SparkConf, SparkContext
        conf = (SparkConf()
                .setMaster("local")
                .setAppName("My app")
                .set("spark.executor.memory", "1g"))
        sc = SparkContext(conf = conf)
        mydata = sc.textFile("purplecow.txt")

In [2]: mydata_uc = mydata.map(lambda line: line.upper())

In [3]: mydata_filt = mydata_uc.filter(lambda line: line.startswith('I'))

In [4]: mydata_filt.count()
Out[4]: 3
```

RDD process is not performed

RDD process is performed

Lazy Execution

- When using transformation funtions, there are no RDD process.
- It creates the lineage log.
- When using action functions, RDD process are done based on the lineage log.

Functional Programming in spark

- Key Concepts
  - Passing functions as input to other functions
  - Anonymous functions

- Passing functions example with pyspark

```
In [5]: def toUpper(s):
            return s.upper()

In [6]: mydata.map(toUpper).take(2)
Out[6]: ["I'VE NEVER SEEN A PURPLE COW.", 'I NEVER HOPE TO SEE ONE;']
```

Define a function 'toUpper'

Passing function as input to map function

Functional Programming in spark

- Anonymous Functions
  - Functions defined in-line without an identifier
  - Supported in python and scala(ex. lambda x: …)

```
In [7]: mydata.map(lambda line: line.upper()).take(2)
Out[7]: ["I'VE NEVER SEEN A PURPLE COW.", 'I NEVER HOPE TO SEE ONE;']
```

Working with RDDs

- RDDs can hold any type of element
  - Primitive types : integers, characters, booleans, etc.
  - Sequence types : strings, lists, arrays, tuples, dicts, etc.
  - Scala/Java objects(if serializable)
  - Mixed types
- Some types of RDDs have additional functionality
  - Pair RDDs : RDDs consisting of key-value pairs
  - Double RDDs : RDDs consisting of numeric data

Some other general RDD operations

- Transformations
  - flatMap : maps one element in the base RDD to multiple elements
  - distinct : filter out duplicates
  - union : add all elements of two RDDs into a single new RDD
- Other RDD operations
  - first – return the first element of the RDD
  - foreach – apply a function to each element in an RDD
  - top(n) – return the largest n elements using natural ordering

flatMap and distinct

- Example of flatMap and distinct

```
In [41]: test = sc.textFile("purplecow.txt")

In [52]: test_flm = test.flatMap(lambda line: line.split())

In [53]: test_flm.take(3)
Out[53]: ["I've", 'never', 'seen']
```

[without distinct]

```
In [41]: test = sc.textFile("purplecow.txt")

In [54]: test_flm = test.flatMap(lambda line: line.split()).distinct()

In [55]: test_flm.take(3)
Out[55]: ['see', 'I', 'can']
```

[with distinct]

Pair RDD

- Pair RDDs are a special form of RDD
  - Each element must be a key -value pair(a two element tuple)
  - Keys and values can be any type.

- Commonly used functions to create pair RDDs
  - map
  - flatMap/flatMapValues
  - keyBy

Pair RDD

(key1,value1)
(key2,value2)
(key3,value3)
…

Pair RDD Examples

- Examples creating example

[Sample.txt]

Split by '\t' and map the data into key-value pair RDD

```
mydata = sc.textFile("sample.txt")
```

```
mydata2 = mydata.map(lambda line: line.split('Wt')).map(lambda fields: (fields[0],fields[1]))
```

```
mydata2.collect()
```

```
[('user001', 'Fred Flintstone'),
 ('user090', 'Bugs Bunny'),
 ('user111', 'Harry Porter')]
```

Example. Keying web log by User ID

```
In [15]: originalLog = sc.textFile("2013-09-15.log")

In [16]: originalLog.take(2)

Out[16]: ['116.180.70.237 - 128 [15/Sep/2013:23:59:53 +0100] "GET /KBDOC-00031.html HTTP/1.0" 200 1388 "http://www.loudacre.com"  "Loudacre CSR Browser"',
          '116.180.70.237 - 128 [15/Sep/2013:23:59:53 +0100] "GET /theme.css HTTP/1.0" 200 5531 "http://www.loudacre.com"  "Loudacre CSR Browser"']

In [11]: myLog = sc.textFile("2013-09-15.log").keyBy(lambda line: line.split(' ')[2])

In [12]: myLog.take(2)

Out[12]: [('128',
           '116.180.70.237 - 128 [15/Sep/2013:23:59:53 +0100] "GET /KBDOC-00031.html HTTP/1.0" 200 1388 "http://www.loudacre.com"  "Loudacre CSR Browser"'),
          ('128',
           '116.180.70.237 - 128 [15/Sep/2013:23:59:53 +0100] "GET /theme.css HTTP/1.0" 200 5531 "http://www.loudacre.com"  "Loudacre CSR Browser"')]
```

Split by empty space and key by 3rd element

Mapping Single Rows to Multiple Pairs

```
In [51]: skus = sc.textFile("orderskus.txt")

In [56]: skus.collect()

Out[56]: ['00001₩tsku010:sku933:sku022',
          '00002₩tsku912:sku331',
          '00003₩tsku888:sku022:sku010:sku594',
          '00004₩tsku411',
          '00005₩tsku331:sku010']
```

[Original textfile]

Textfile format

1) Number and IDs are delimited by '\t'

2) Multiple IDs are delimited by ':'

Mapping Single Rows to Multiple Pairs

```
In [52]: skus2 = skus.map(lambda line : line.split('Wt'))

In [53]: skus2.collect()

Out[53]: [['00001', 'sku010:sku933:sku022'],
          ['00002', 'sku912:sku331'],
          ['00003', 'sku888:sku022:sku010:sku594'],
          ['00004', 'sku411'],
          ['00005', 'sku331:sku010']]

In [54]: skus3=skus2.map(lambda fields: (fields[0],fields[1])).flatMapValues(lambda skus: skus.split(':'))

In [55]: skus3.collect()

Out[55]: [('00001', 'sku010'),
          ('00001', 'sku933'),
          ('00001', 'sku022'),
          ('00002', 'sku912'),
          ('00002', 'sku331'),
          ('00003', 'sku888'),
          ('00003', 'sku022'),
          ('00003', 'sku010'),
          ('00003', 'sku594'),
          ('00004', 'sku411'),
          ('00005', 'sku331'),
          ('00005', 'sku010')]
```

Split the text by '\t' line by line

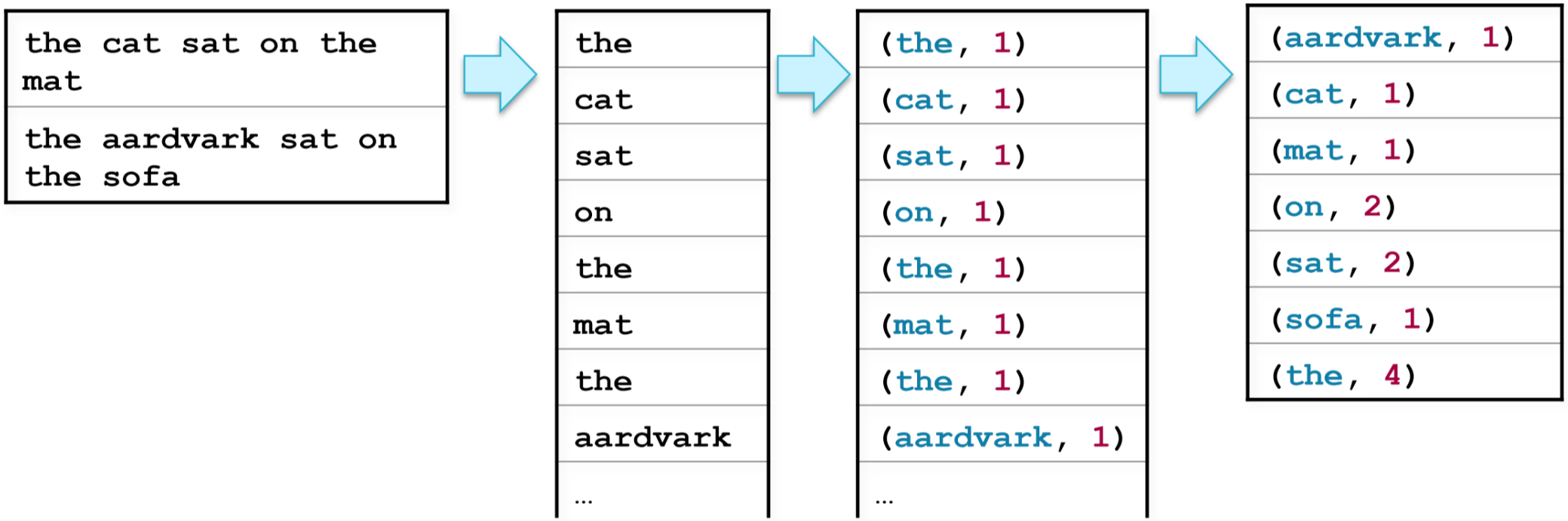
Create pair RDD

flatMapValues by delimiter ':'

Multiple pairs

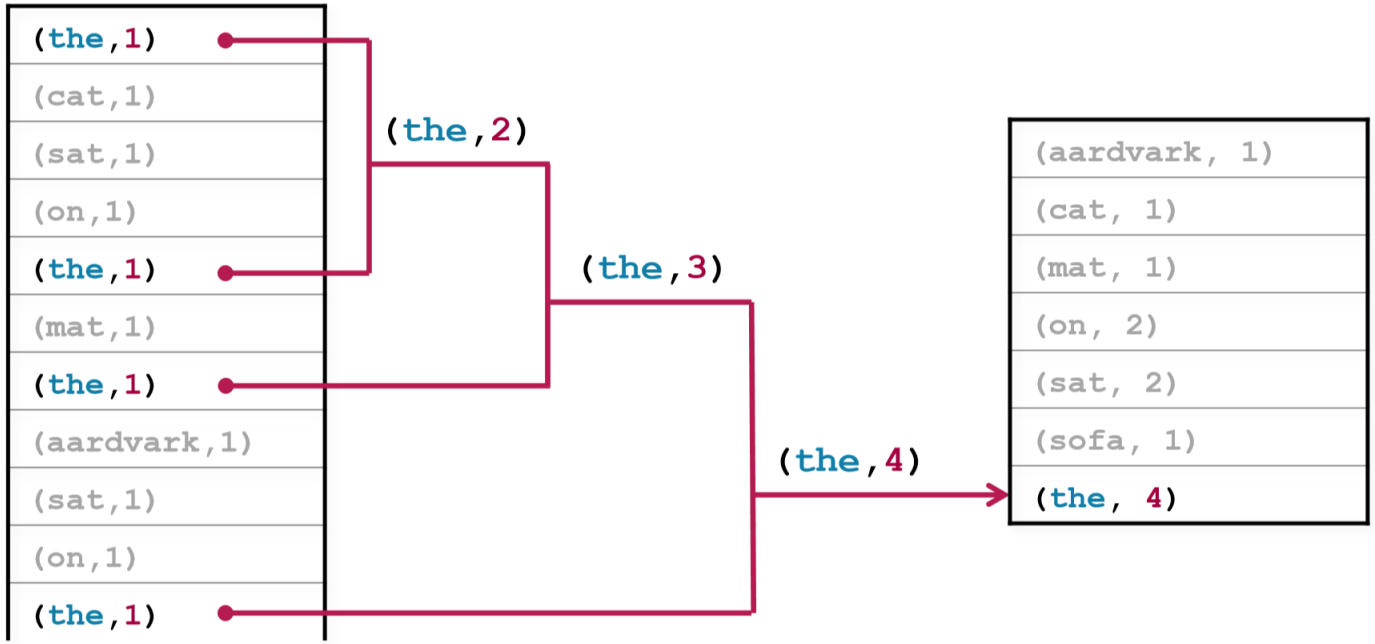
MapReduce

- Mapping the textfile data



MapReduce

- Reducing phase



MapReduce with pyspark

```
In [57]: input = sc.textFile("input.txt")
```

```
In [63]: input.collect()
```

```
Out[63]: ['the cat sat on the mat', 'the aardvark sat on the sofa']
```

```
In [61]: input2 = input.flatMap(lambda lines: lines.split()).map(lambda word: (word,1)).reduceByKey(lambda v1,v2 : v1+v2)
```

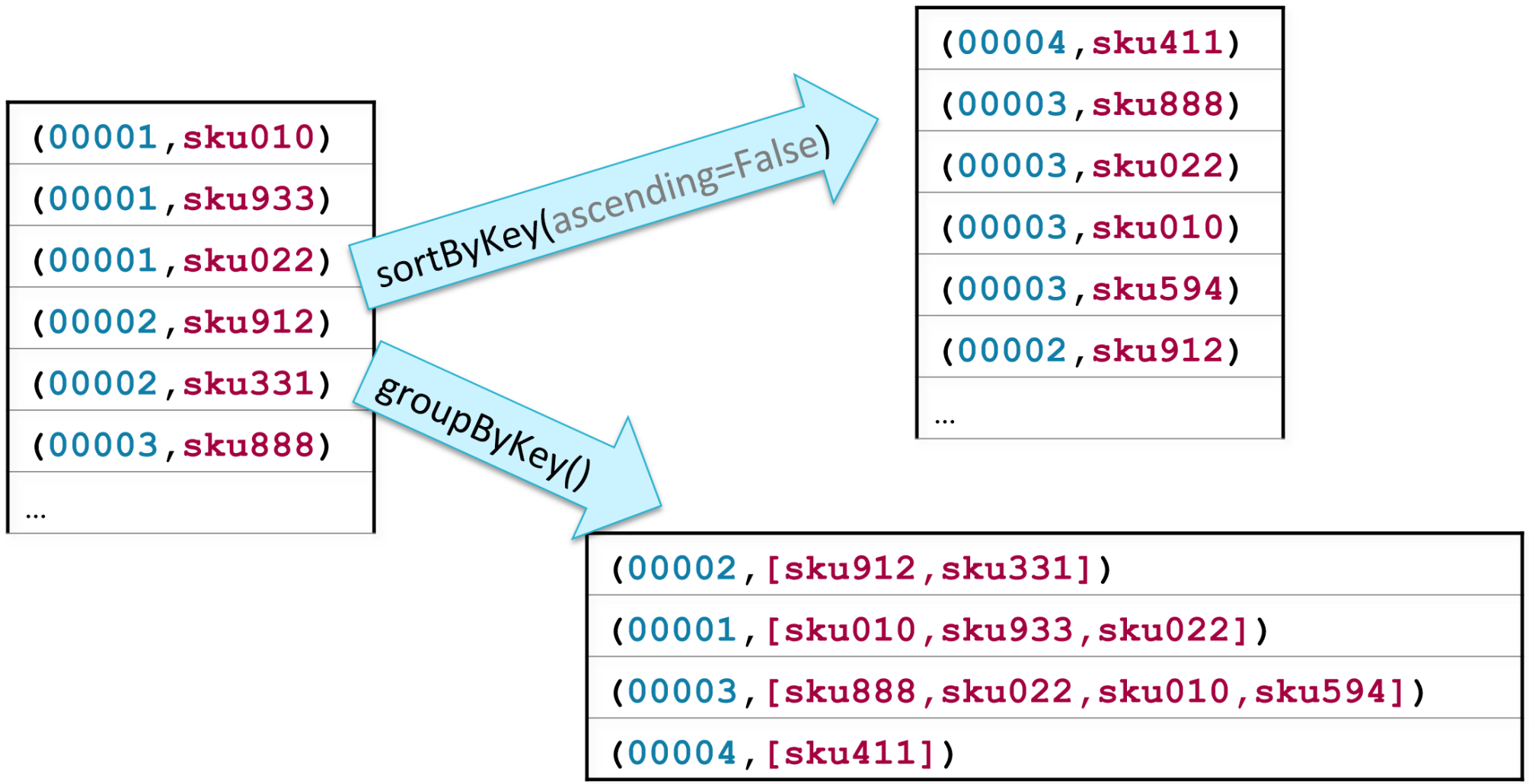
```
In [62]: input2.collect()
```

```
Out[62]: [('sofa', 1),
          ('cat', 1),
          ('the', 4),
          ('aardvark', 1),
          ('on', 2),
          ('sat', 2),
          ('mat', 1)]
```

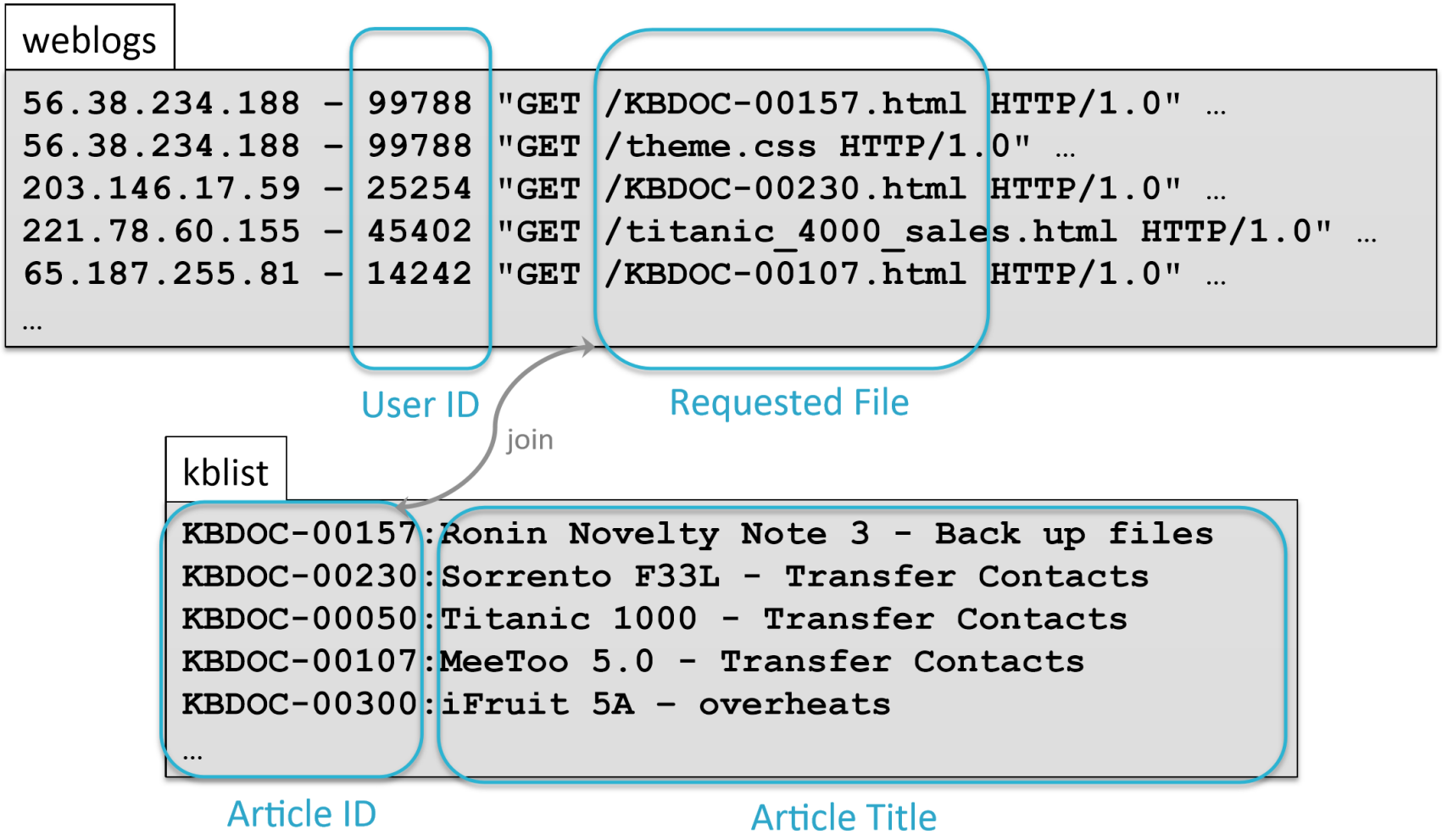
Other Pair RDD Operations

- **`countByKey`** – return a map with the count of occurrences of each key
- **`groupByKey`** – group all the values for each key in an RDD
- **`sortByKey`** – sort in ascending or descending order
- **`join`** – return an RDD containing all pairs with matching keys from two RDDs

Other Pair RDD Operations



Example. Join Web log with KB articles



Join the requested file which has a format 'KBDOC-[0-9]* of weblogs with Article ID of kblist data

Example. Join Web log with KB articles

- Mapping kblist.txt

```
In [69]: kblist = sc.textFile("kblist.txt").map(lambda line : line.split(':')).map(lambda fields: (fields[0],fields[1]))

In [73]: kblist.collect()

Out[73]: [('KBDOC-00087', 'Ronin Novelty Note 1 - Back up files'),
          ('KBDOC-00293', 'Ronin S2 - Battery Life'),
          ('KBDOC-00199',
           'Titanic 2000 - Change the phone ringtone and notification sound'),
          ('KBDOC-00211',
           'MeeToo 5.1 - Change the phone ringtone and notification sound'),
          ('KBDOC-00037',
           'iFruit 2 - Change the phone ringtone and notification sound'),
          ('KBDOC-00245', 'Sorrento F31L - Battery Life'),
          ('KBDOC-00058', 'MeeToo 1.0 - reboot'),
          ('KBDOC-00067',
           'iFruit 4 - Change the phone ringtone and notification sound'),
          ('KBDOC-00116', 'iFruit 3A - Transfer Contacts'),
          ('KBDOC-00164', 'Titanic 4000 - Transfer Contacts'),
          ('KBDOC-00039', 'iFruit 2 - Back up files'),
```

Example. Join Web log with KB articles

- Mapping ‘2013-09-17.log’

```
In [74]: kbreq = sc.textFile("2013-09-17.log")

In [78]: import re
         def getRequestDoc(s):
             return re.search(r'KBDOC-[0-9]*', s).group()

In [79]: kbreqs = kbreq.filter(lambda line: 'KBDOC-' in line).map(lambda line: (getRequestDoc(line), line.split(' ')[2]))

In [80]: kbreqs.collect()

Out[80]: [('KBDOC-00158', '30967'),
          ('KBDOC-00140', '95'),
          ('KBDOC-00011', '36386'),
          ('KBDOC-00257', '28268'),
          ('KBDOC-00031', '90'),
          ('KBDOC-00060', '3025'),
          ('KBDOC-00249', '10'),
          ('KBDOC-00235', '72'),
          ('KBDOC-00235', '63'),
          ('KBDOC-00053', '1245'),
          ('KBDOC-00066', '57'),
          ('KBDOC-00214', '9664'),
```

Example. Join Web log with KB articles

- Joining two RDD by key

```
In [82]: titleReqs = kbreqs.join(kblist)

In [83]: titleReqs.collect()

Out[83]: [('KBD0C-00066', ('57', 'Sorrento F21L - overheating')),
          ('KBD0C-00066', ('125', 'Sorrento F21L - overheating')),
          ('KBD0C-00066', ('1207', 'Sorrento F21L - overheating')),
          ('KBD0C-00143', ('25818', 'Sorrento F32L - Battery Life')),
          ('KBD0C-00143', ('56', 'Sorrento F32L - Battery Life')),
          ('KBD0C-00143', ('160', 'Sorrento F32L - Battery Life')),
          ('KBD0C-00241',
           ('178', 'Sorrento F31L - Change the phone ringtone and notification sound')),
          ('KBD0C-00241',
           ('6157',
            'Sorrento F31L - Change the phone ringtone and notification sound')),
          ('KBD0C-00065', ('6', 'Sorrento F21L - Battery Life')),
          ('KBD0C-00065', ('64455', 'Sorrento F21L - Battery Life')),
          ('KBD0C-00065', ('25', 'Sorrento F21L - Battery Life')),
          ('KBD0C-00065', ('52', 'Sorrento F21L - Battery Life')),
          ('KBD0C-00065', ('50155', 'Sorrento F21L - Battery Life')),
          ('KBD0C-00065', ('91', 'Sorrento F21L - Battery Life')),
          ('KBD0C-00065', ('38741', 'Sorrento F21L - Battery Life')),
          ('KBD0C-00101', ('106', 'Ronin Novelty Note 3 - Battery Life')),
          ('KBD0C-00018', ('24264', 'MeeTee 3.0 - overheat'))
```