Data Science & Society

Lecture 05: More *Spark Transformations, Neo Demos & Midterm Q/A*

INFOMDSS 2018 :: Dr. Marco Spruit

Agenda

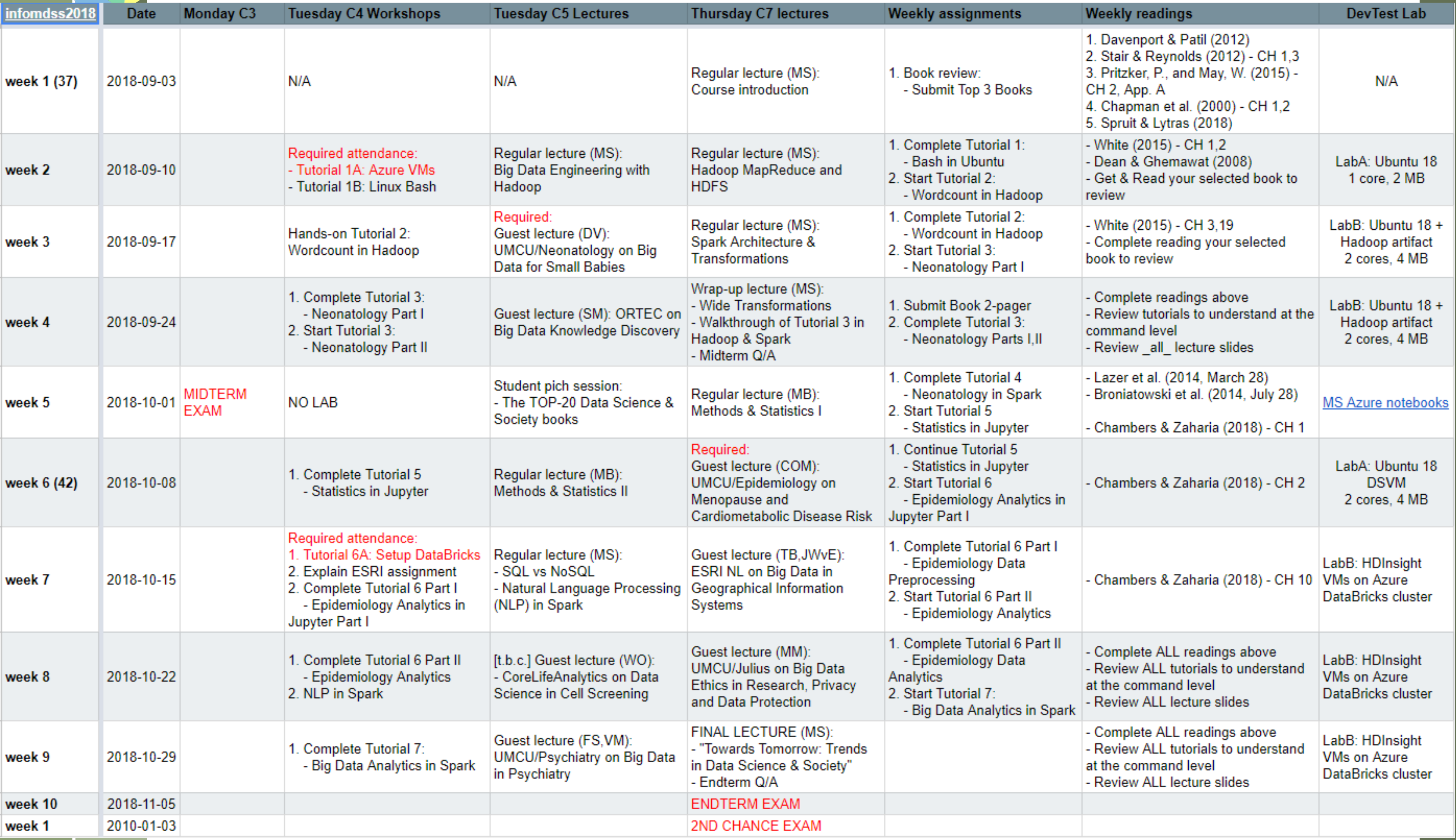
- Status Quo...
- Finalised Course Schedule
  - http://bit.ly/infomdss-schedule
- Spark
  - Wide Transformations in Spark
- *"Putting it together"*
  - RDDs + Transformations = DAGs
- Neonatology Demos
  - Tutorial 3 in Hadoop
  - Tutorial 3 in Spark
- Book review Q/A

Status Quo

- Severity 1
- Switch to **LabA** with DSVMs
  - Get your laptops NOW
  - goto portal.azure.com, log in, goto LabA
  - Claim a VM
  - Log in with the course pw (wait 2 minutes if first fails)
- Install Hadoop
  - cd ~/
  - wget https://raw.githubusercontent.com/rebremer/AzureDevTestLabHadoopSparkOnUbuntuScript/master/Artifacts/HadoopHive/HadoopHiveUbuntuArtifact.sh
  - chmod +x HadoopHiveUbuntuArtifact.sh
  - sudo ./HadoopHiveUbuntuArtifact.sh

Course Schedule 2018

http://bit.ly/infomdss-schedule



infomdss2018	Date	Monday C3	Tuesday C4 Workshops	Tuesday C5 Lectures	Thursday C7 lectures	Weekly assignments	Weekly readings	DevTest Lab
week 1 (37)	2018-09-03		N/A	N/A	Regular lecture (MS): Course introduction	1. Book review: - Submit Top 3 Books	1. Davenport & Patil (2012) 2. Stair & Reynolds (2012) - CH 1,3 3. Pritzker, P., and May, W. (2015) - CH 2, App. A 4. Chapman et al. (2000) - CH 1,2 5. Spruit & Lytras (2018)	N/A
week 2	2018-09-10		Required attendance: - Tutorial 1A: Azure VMs - Tutorial 1B: Linux Bash	Regular lecture (MS): Big Data Engineering with Hadoop	Regular lecture (MS): Hadoop MapReduce and HDFS	1. Complete Tutorial 1: - Bash in Ubuntu 2. Start Tutorial 2: - Wordcount in Hadoop	- White (2015) - CH 1,2 - Dean & Ghemawat (2008) - Get & Read your selected book to review	LabA: Ubuntu 18 1 core, 2 MB
week 3	2018-09-17		Hands-on Tutorial 2: Wordcount in Hadoop	Required: Guest lecture (DV): UMCU/Neonatology on Big Data for Small Babies	Regular lecture (MS): Spark Architecture & Transformations	1. Complete Tutorial 2: - Wordcount in Hadoop 2. Start Tutorial 3: - Neonatology Part I	- White (2015) - CH 3,19 - Complete reading your selected book to review	LabB: Ubuntu 18 + Hadoop artifact 2 cores, 4 MB
week 4	2018-09-24		1. Complete Tutorial 3: - Neonatology Part I 2. Start Tutorial 3: - Neonatology Part II	Guest lecture (SM): ORTEC on Big Data Knowledge Discovery	Wrap-up lecture (MS): - Wide Transformations - Walkthrough of Tutorial 3 in Hadoop & Spark - Midterm Q/A	1. Submit Book 2-pager 2. Complete Tutorial 3: - Neonatology Parts I,II	- Complete readings above - Review tutorials to understand at the command level - Review _all_ lecture slides	LabB: Ubuntu 18 + Hadoop artifact 2 cores, 4 MB
week 5	2018-10-01	MIDTERM EXAM	NO LAB	Student pich session: - The TOP-20 Data Science & Society books	Regular lecture (MB): Methods & Statistics I	1. Complete Tutorial 4 - Neonatology in Spark 2. Start Tutorial 5 - Statistics in Jupyter	- Lazer et al. (2014, March 28) - Broniatowski et al. (2014, July 28) - Chambers & Zaharia (2018) - CH 1	MS Azure notebooks
week 6 (42)	2018-10-08		1. Complete Tutorial 5 - Statistics in Jupyter	Regular lecture (MB): Methods & Statistics II	Required: Guest lecture (COM): UMCU/Epidemiology on Menopause and Cardiometabolic Disease Risk	1. Continue Tutorial 5 - Statistics in Jupyter 2. Start Tutorial 6 - Epidemiology Analytics in Jupyter Part I	- Chambers & Zaharia (2018) - CH 2	LabA: Ubuntu 18 DSVM 2 cores, 4 MB
week 7	2018-10-15		Required attendance: 1. Tutorial 6A: Setup DataBricks 2. Explain ESRI assignment 2. Complete Tutorial 6 Part I - Epidemiology Analytics in Jupyter Part I	Regular lecture (MS): - SQL vs NoSQL - Natural Language Processing (NLP) in Spark	Guest lecture (TB,JWvE): ESRI NL on Big Data in Geographical Information Systems	1. Complete Tutorial 6 Part I - Epidemiology Data Preprocessing 2. Start Tutorial 6 Part II - Epidemiology Analytics	- Chambers & Zaharia (2018) - CH 10	LabB: HDInsight VMs on Azure DataBricks cluster
week 8	2018-10-22		1. Complete Tutorial 6 Part II - Epidemiology Analytics 2. NLP in Spark	[t.b.c.] Guest lecture (WO): - CoreLifeAnalytics on Data Science in Cell Screening	Guest lecture (MM): UMCU/Julius on Big Data Ethics in Research, Privacy and Data Protection	1. Complete Tutorial 6 Part II - Epidemiology Data Analytics 2. Start Tutorial 7: - Big Data Analytics in Spark	- Complete ALL readings above - Review ALL tutorials to understand at the command level - Review ALL lecture slides	LabB: HDInsight VMs on Azure DataBricks cluster
week 9	2018-10-29		1. Complete Tutorial 7: - Big Data Analytics in Spark	Guest lecture (FS,VM): UMCU/Psychiatry on Big Data in Psychiatry	FINAL LECTURE (MS): - "Towards Tomorrow: Trends in Data Science & Society" - Endterm Q/A		- Complete ALL readings above - Review ALL tutorials to understand at the command level - Review ALL lecture slides	LabB: HDInsight VMs on Azure DataBricks cluster
week 10	2018-11-05				ENDTERM EXAM			
week 1	2010-01-03				2ND CHANCE EXAM			

Recap: Narrow Transformations

in Apache Spark

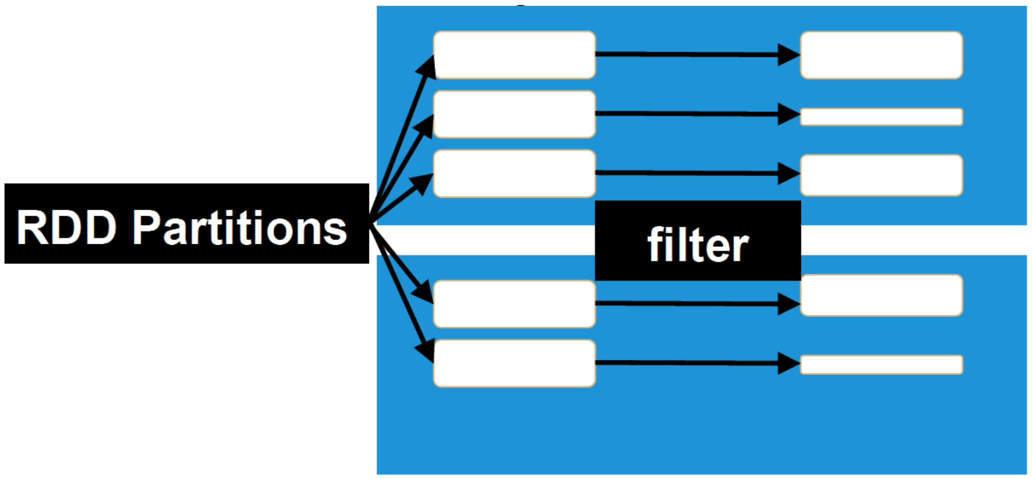
Some *Narrow* Transformations

- `map(func)`
  - apply function to each element of RDD
- `flatMap(func)`
  - map then flatten output
- `filter(func)`
  - keep only elements where func is `true`
- `sample(withReplacement,fraction,seed)`
  - get a random data fraction
- `coalesce(numPartitions)`
  - merge partitions to reduce them to numPartitions

Example transformation: `filter(func)`

1. Define a function to filter out all words which start with an "a"

```
def starts_with_a(word):
    return word.lower().startswith("a")

words_RDD.filter(starts_with_a).collect()

  Out[]:[u'A', u'ago', u'a', u'away']
```



filter() can result in unevenly distributed partitions...

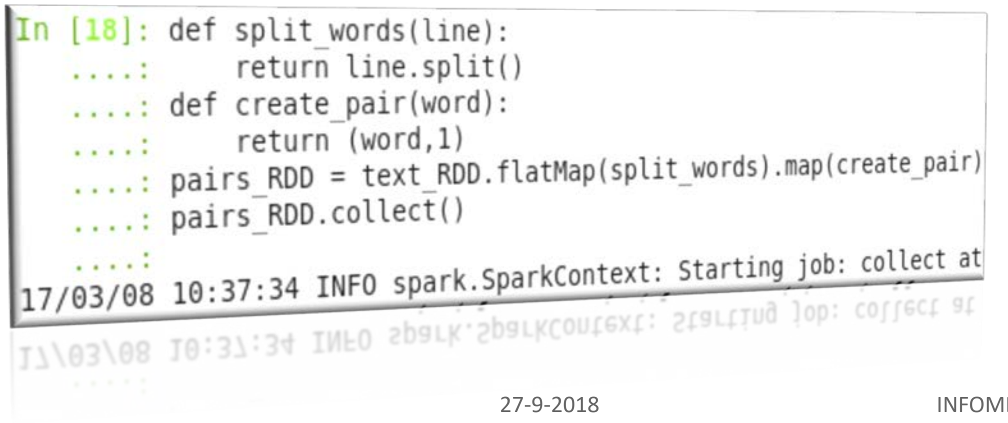
Wide Transformations

in Apache Spark

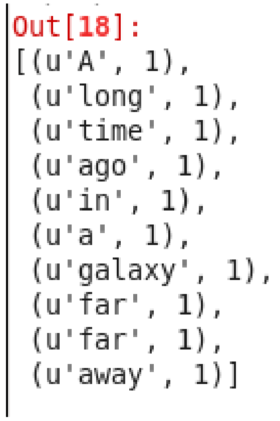
Wordcount in Spark: map

Revisiting Wordcount

```
def split_words(line):
    return line.split()

def create_pair(word):
    return (word,1)

pairs_RDD =
    text_RDD.flatMap(split_words).map(create_pair)
    pairs_RDD.collect()
```



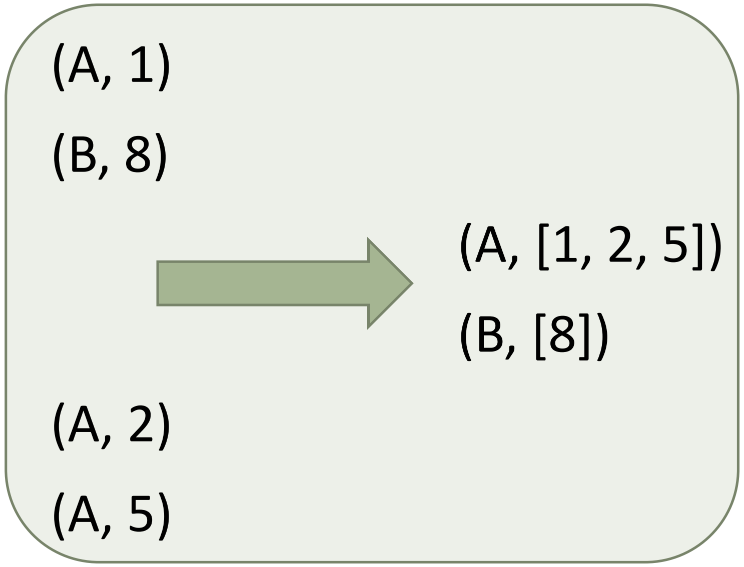
```
In [18]: def split_words(line):
   ....:     return line.split()
   ....: def create_pair(word):
   ....:     return (word,1)
   ....: pairs_RDD = text_RDD.flatMap(split_words).map(create_pair)
   ....: pairs_RDD.collect()
   ....:
17/03/08 10:37:34 INFO spark.SparkContext: Starting job: collect at
```



```
Out[18]:
[(u'A', 1),
 (u'long', 1),
 (u'time', 1),
 (u'ago', 1),
 (u'in', 1),
 (u'a', 1),
 (u'galaxy', 1),
 (u'far', 1),
 (u'far', 1),
 (u'away', 1)]
```

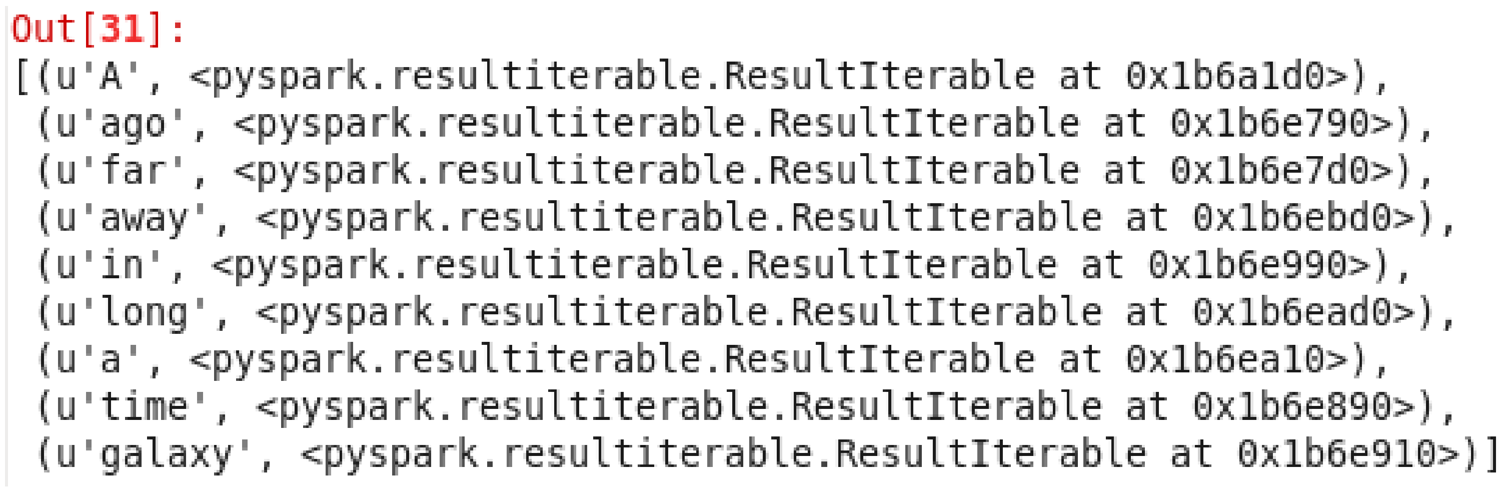
Example transformation: `groupByKey(func)`

› `groupByKey` : (K, V) pairs => (K, iterable of all V)



› `pairs_RDD.`**`groupByKey`**`().collect()   ... ...`

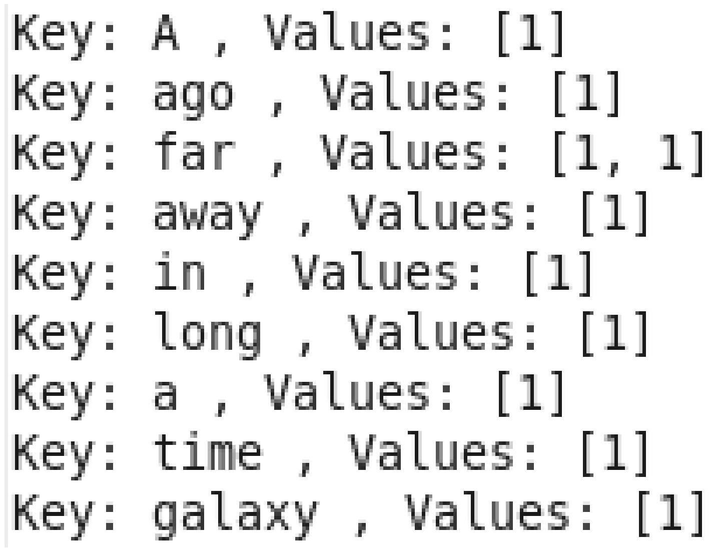
pairs_RDD.**groupByKey**().collect()



```
Out[31]:
[(u'A', <pyspark.resultiterable.ResultIterable at 0x1b6a1d0>),
 (u'ago', <pyspark.resultiterable.ResultIterable at 0x1b6e790>),
 (u'far', <pyspark.resultiterable.ResultIterable at 0x1b6e7d0>),
 (u'away', <pyspark.resultiterable.ResultIterable at 0x1b6ebd0>),
 (u'in', <pyspark.resultiterable.ResultIterable at 0x1b6e990>),
 (u'long', <pyspark.resultiterable.ResultIterable at 0x1b6ead0>),
 (u'a', <pyspark.resultiterable.ResultIterable at 0x1b6ea10>),
 (u'time', <pyspark.resultiterable.ResultIterable at 0x1b6e890>),
 (u'galaxy', <pyspark.resultiterable.ResultIterable at 0x1b6e910>)]
```

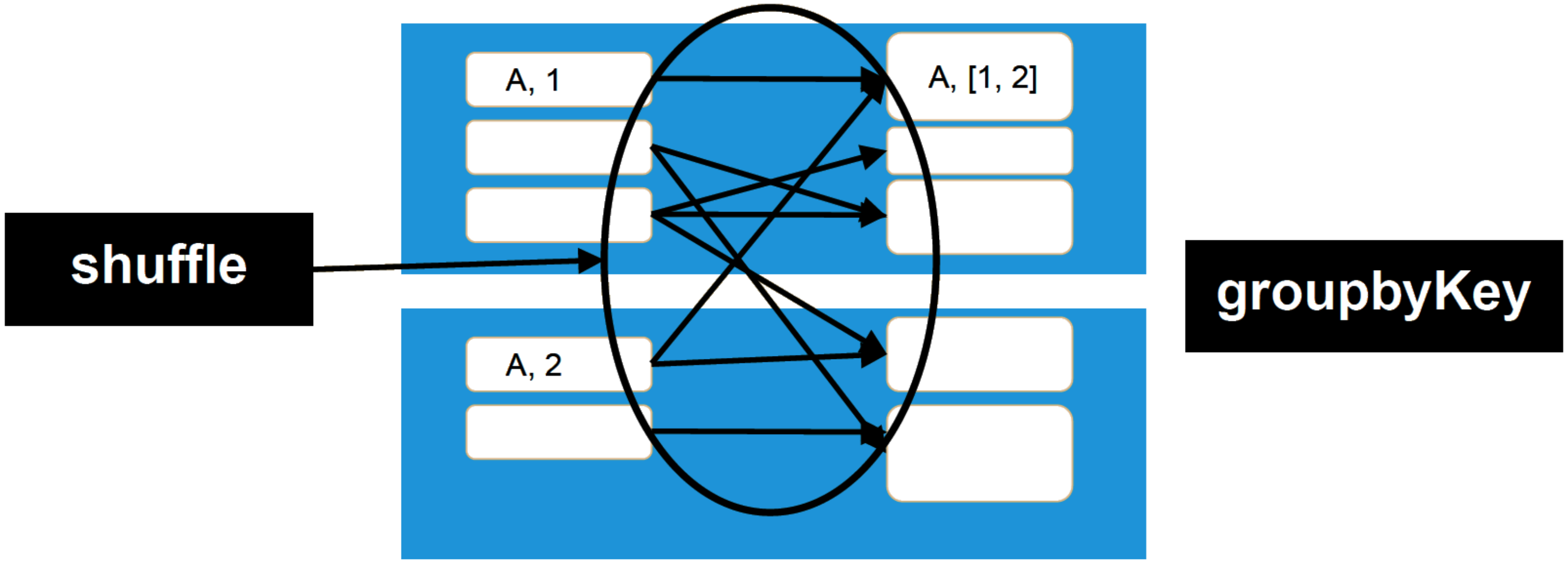
Example transformation: `groupByKey(func)`

```
for k,v in pairs_RDD.groupByKey().collect():
  print "Key:", k, ", Values:", list(v)
```



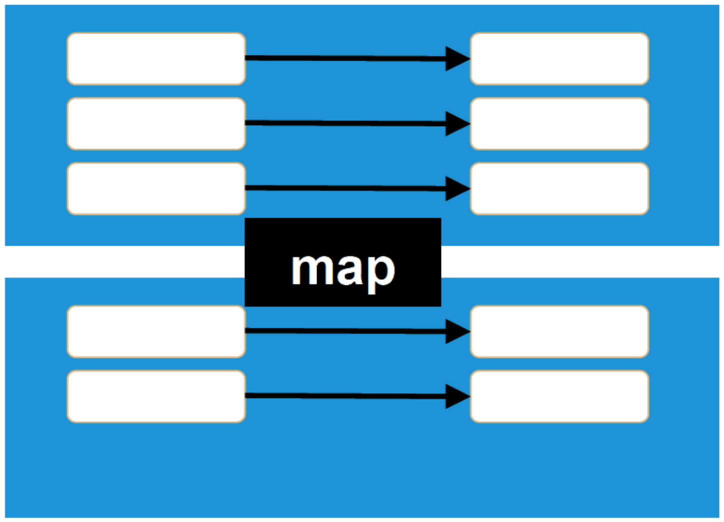
```
Key: A , Values: [1]
Key: ago , Values: [1]
Key: far , Values: [1, 1]
Key: away , Values: [1]
Key: in , Values: [1]
Key: long , Values: [1]
Key: a , Values: [1]
Key: time , Values: [1]
Key: galaxy , Values: [1]
```

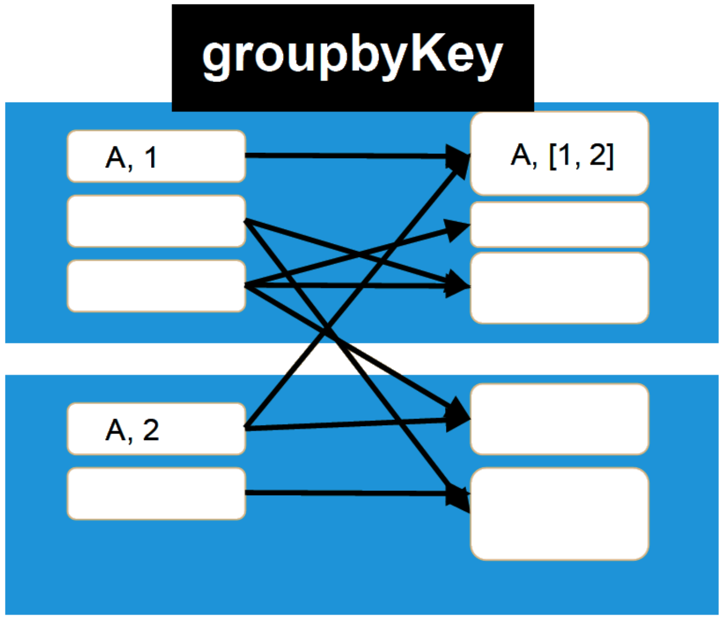
Example transformation: `groupByKey(func)`



Example transformation: `groupByKey(func)`

- Narrow
- Wide





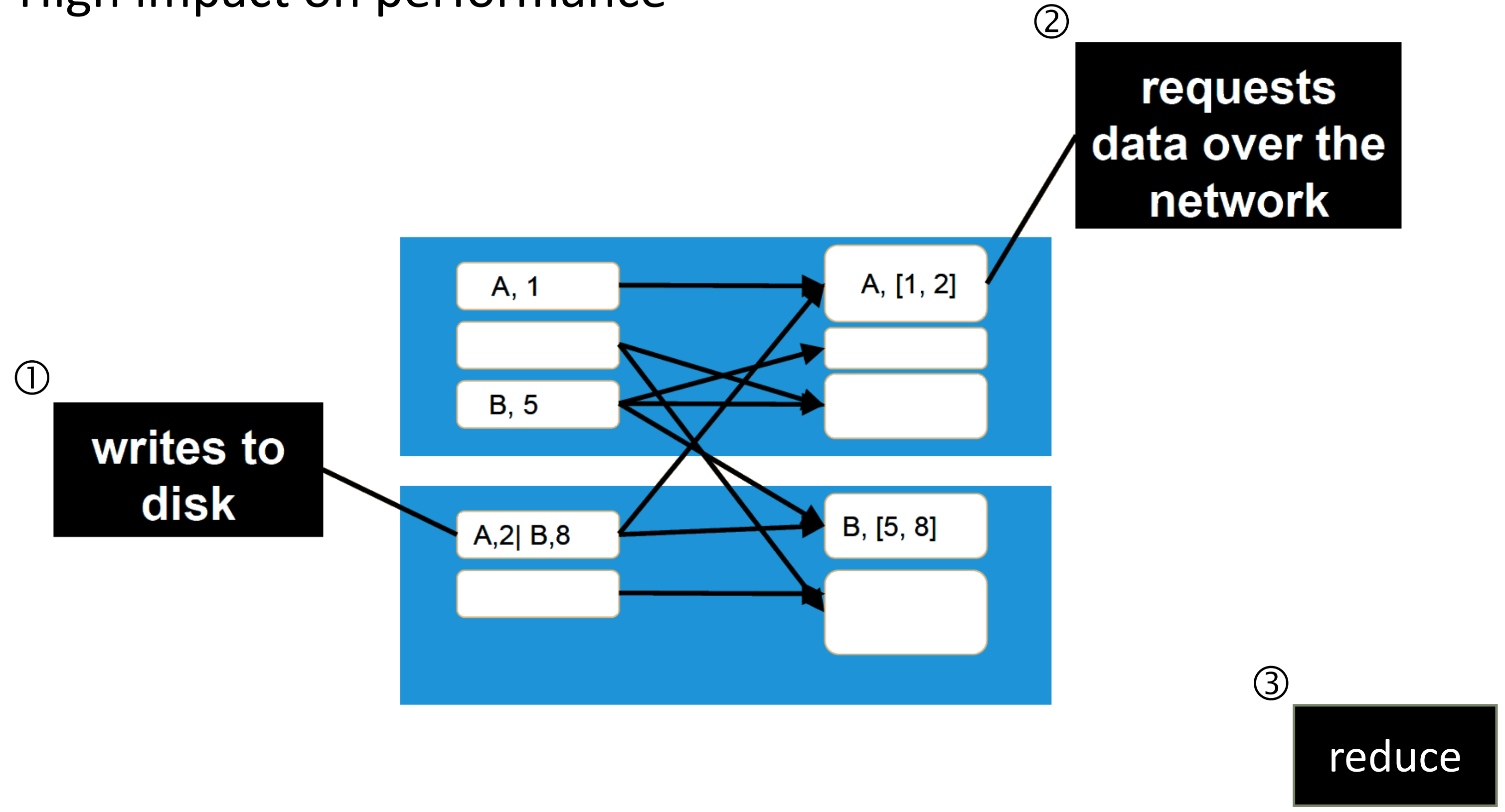
Wide transformations

- `groupByKey()`
  - (K, V) pairs => (K, iterable of all V)
- `reduceByKey(func)`
  - (K, V) pairs => (K, result of reduction by func on all V)
- `repartition(numPartitions)`
  - similar to `coalesce`, *shuffles* all data to increase or decrease number of partitions to numPartitions

Shuffle

› Global redistribution of data

› High impact on performance



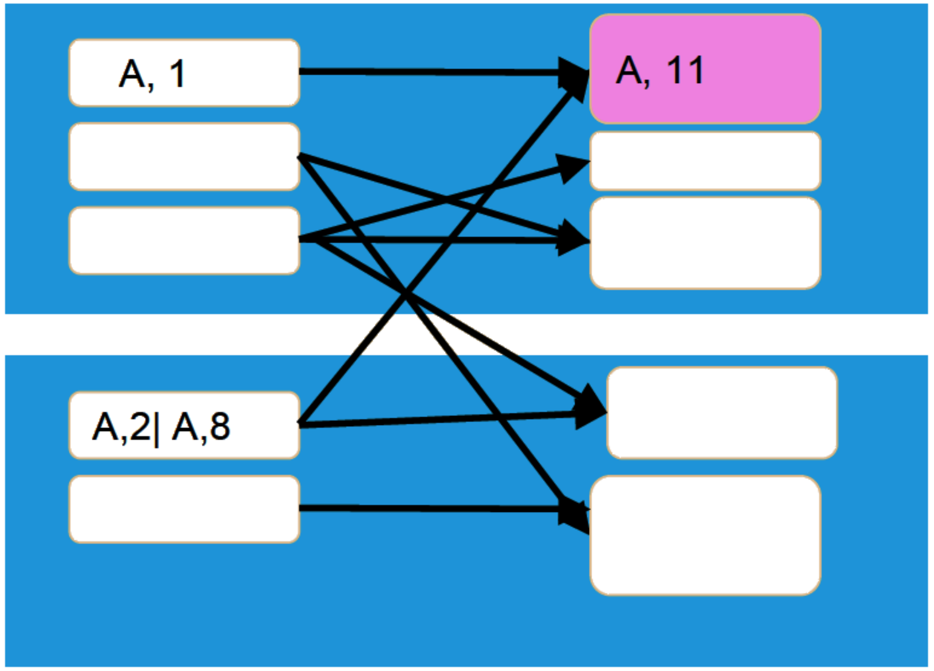
Know shuffle, avoid it

- Which operations cause it?
- Is it necessary?

- e.g. `groupByKey()`
  - (K, V) pairs => (K, iterable of all V)

  - if you plan to call `reduce` later in the pipeline,
  - use `reduceByKey` instead.

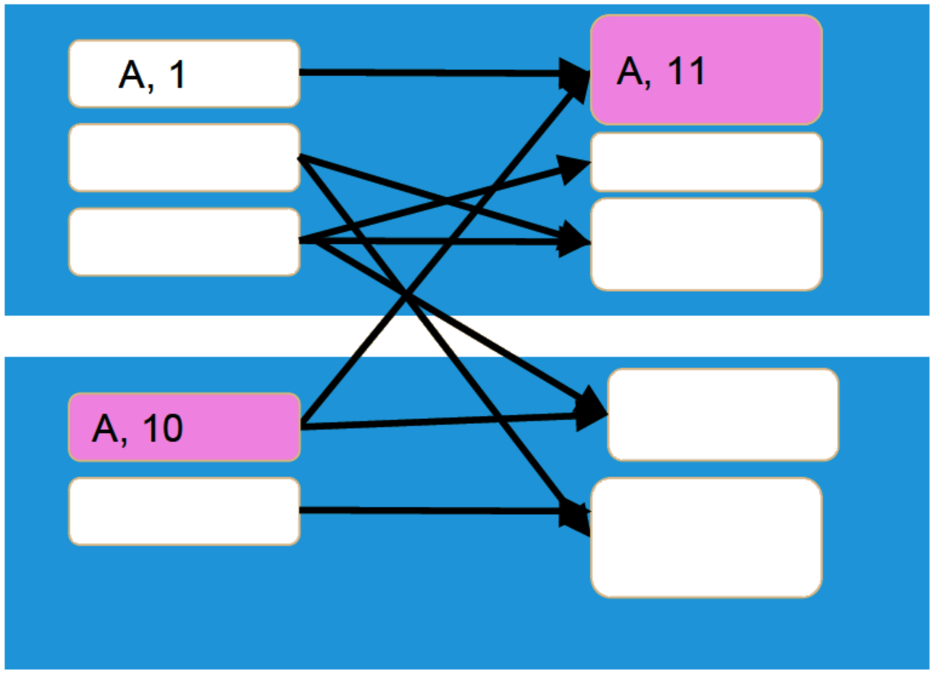
groupByKey + reduce



reduce
after transfer
of all v's

reduceByKey

reduce (k,v) *before* transfer

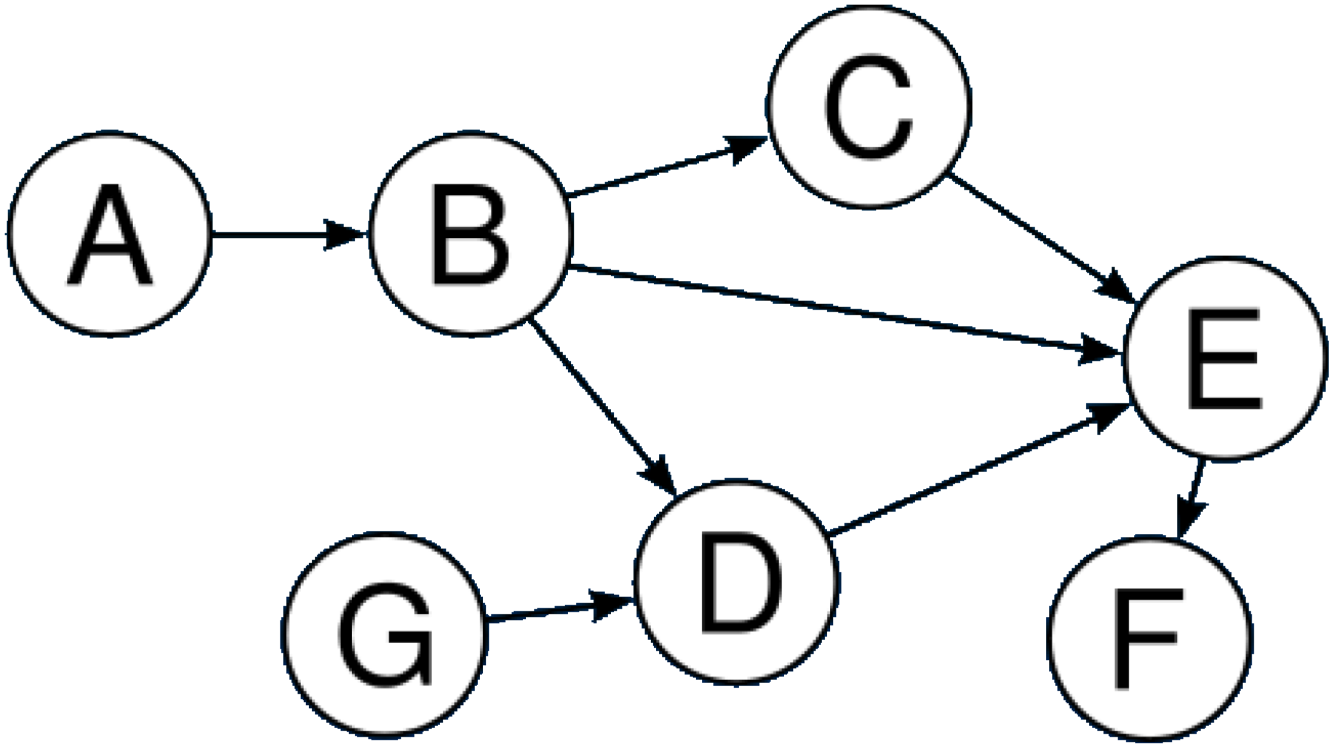


"Putting it together"

DAGs, Actions, Caching, Broadcast, Accumulator

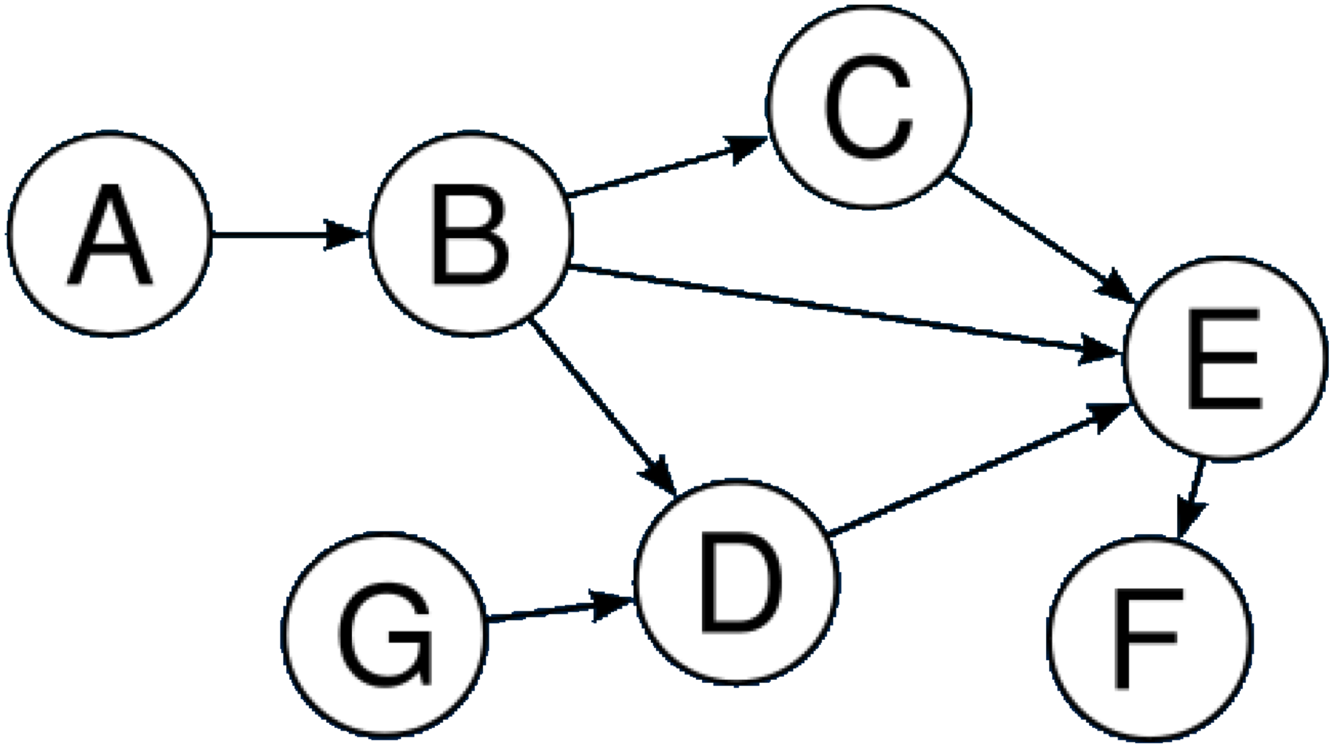
Directed Acyclic Graph (DAG) Scheduler

› Nodes, edges, in directed graph → arrows.



DAG in Spark

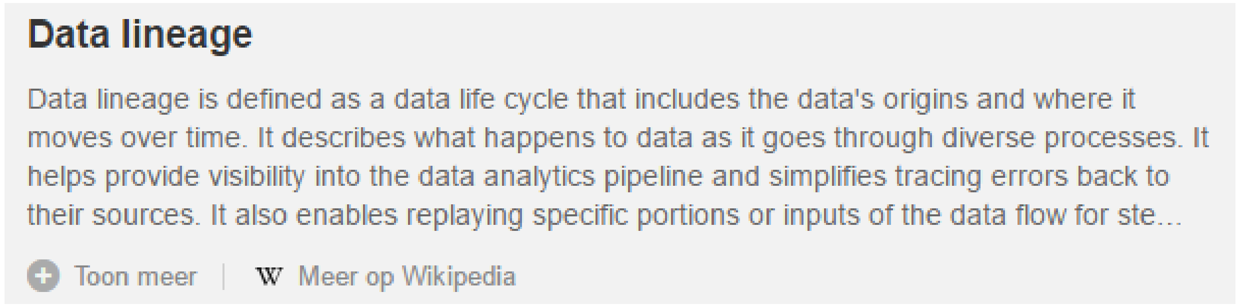
- Nodes are RDDs
- Arrows are Transformations


$$\begin{tikzcd}
 & & C \arrow[rd] & \\
A \arrow[r] & B \arrow[ru] \arrow[rr] \arrow[d] & & E \arrow[d] \\
G \arrow[r] & D \arrow[rru] & & F
\end{tikzcd}$$

ADS

Directed Acyclic Graph Scheduler

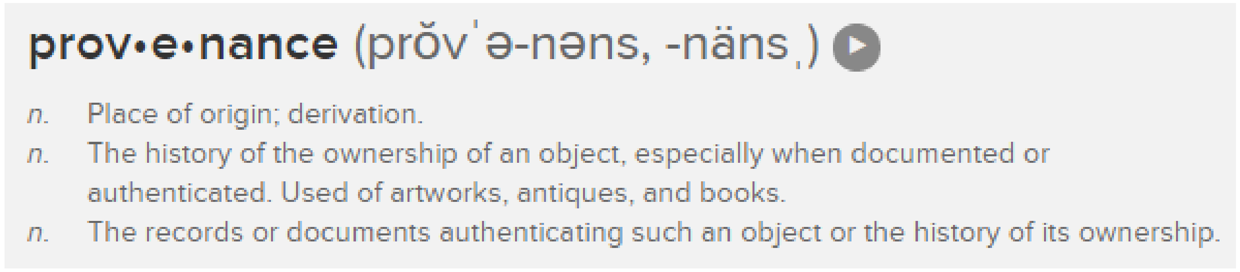
› To track dependencies
  – A.k.a. lineage or provenance



Data lineage

Data lineage is defined as a data life cycle that includes the data's origins and where it moves over time. It describes what happens to data as it goes through diverse processes. It helps provide visibility into the data analytics pipeline and simplifies tracing errors back to their sources. It also enables replaying specific portions or inputs of the data flow for ste...

⊕ Toon meer | W Meer op Wikipedia



prov•e•nance (prŏv′ə-nəns, -näns′) ▶

n. Place of origin; derivation.

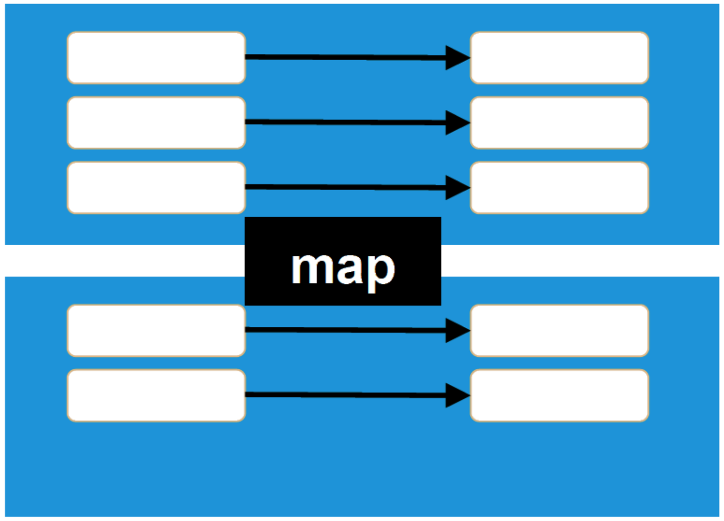
n. The history of the ownership of an object, especially when documented or authenticated. Used of artworks, antiques, and books.

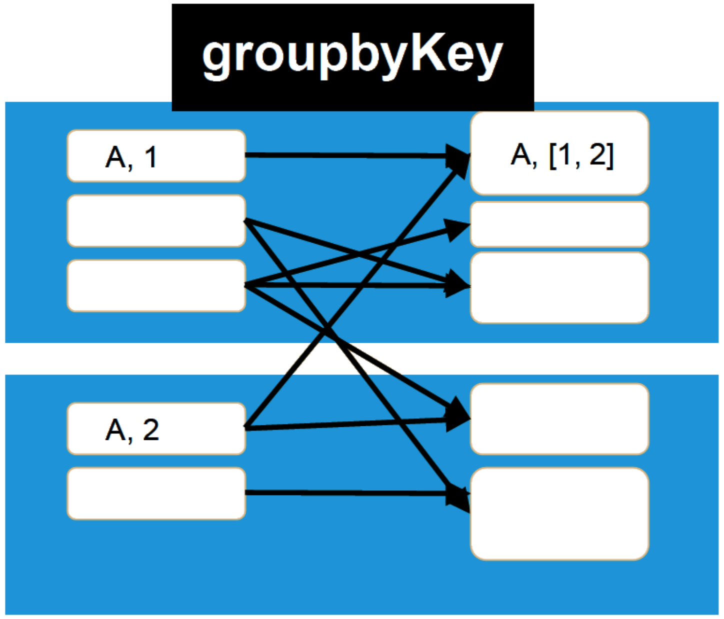
n. The records or documents authenticating such an object or the history of its ownership.

Revisited

Example transformation: `groupByKey(func)`

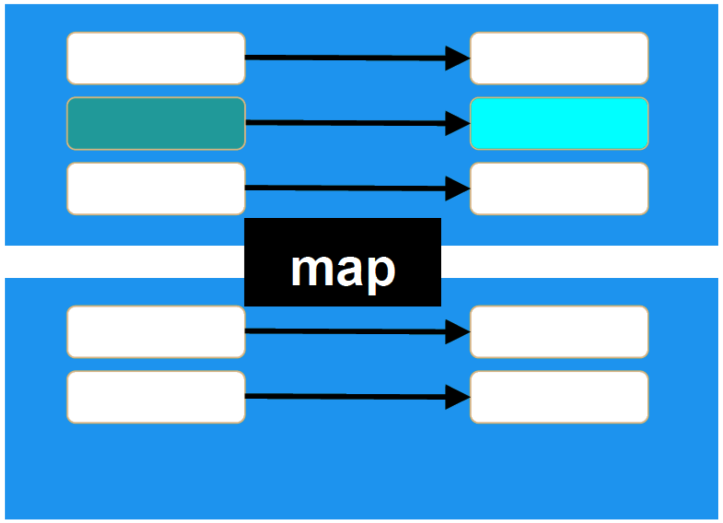
- Narrow
- Wide





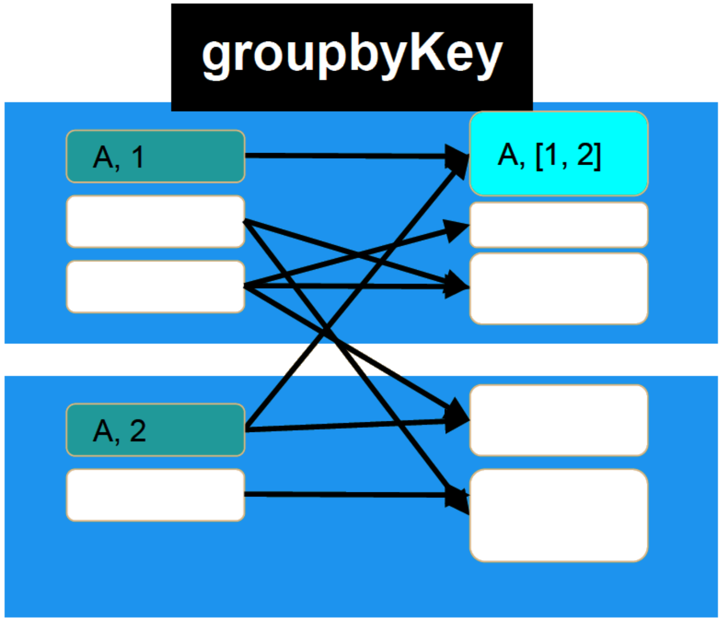
DAGs in example transformations

› Narrow



For a narrow transformation,
One partition depends on one
other partition...

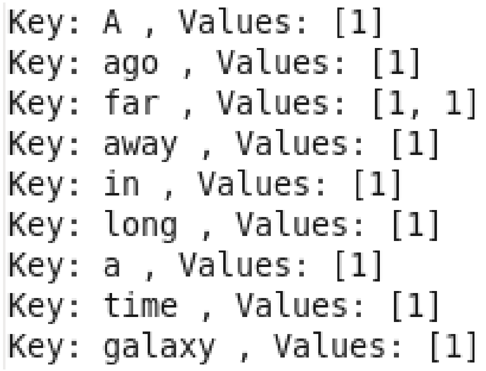
› Wide



Revisited

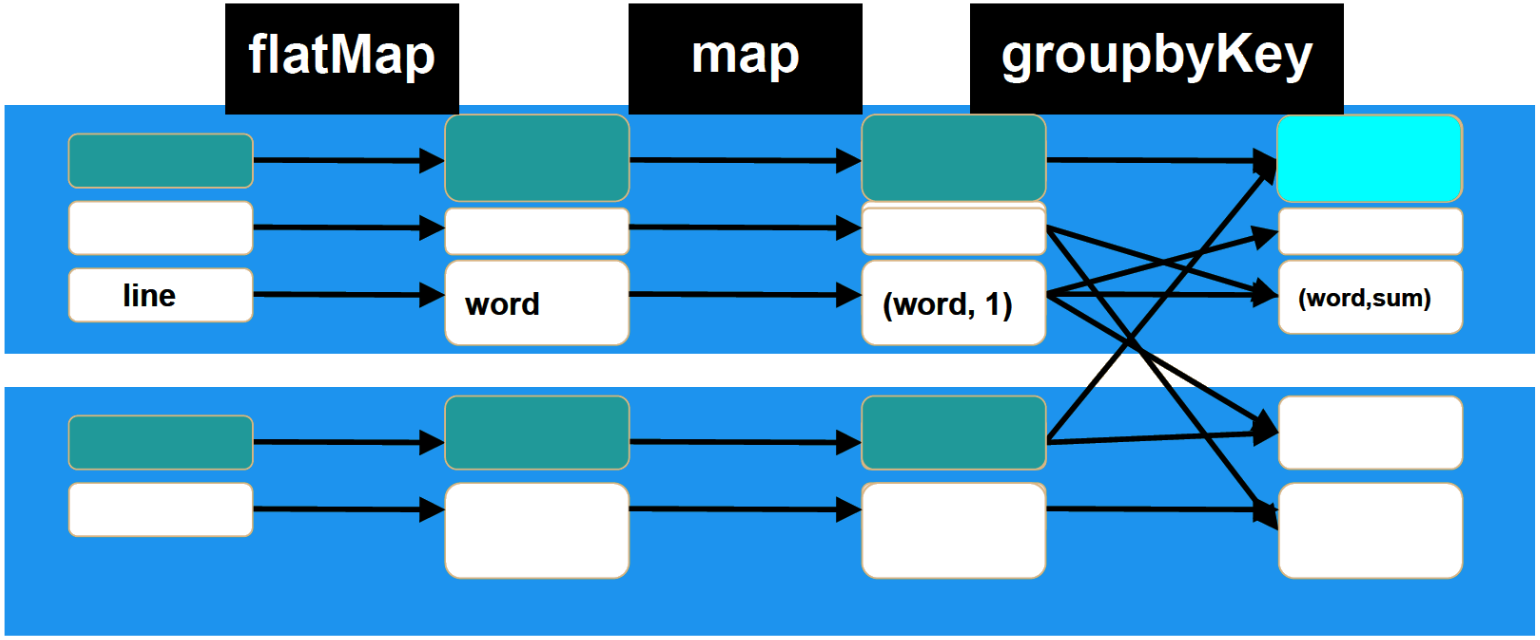
Wordcount: flatMap | map | groupByKey

```
def split_words(line):
   return line.split()

def create_pair(word):
   return (word,1)

pairs_RDD =
   text_RDD.flatMap(split_words).map(create_pair)
   pairs_RDD.collect()

pairs_RDD.groupByKey().collect()
```

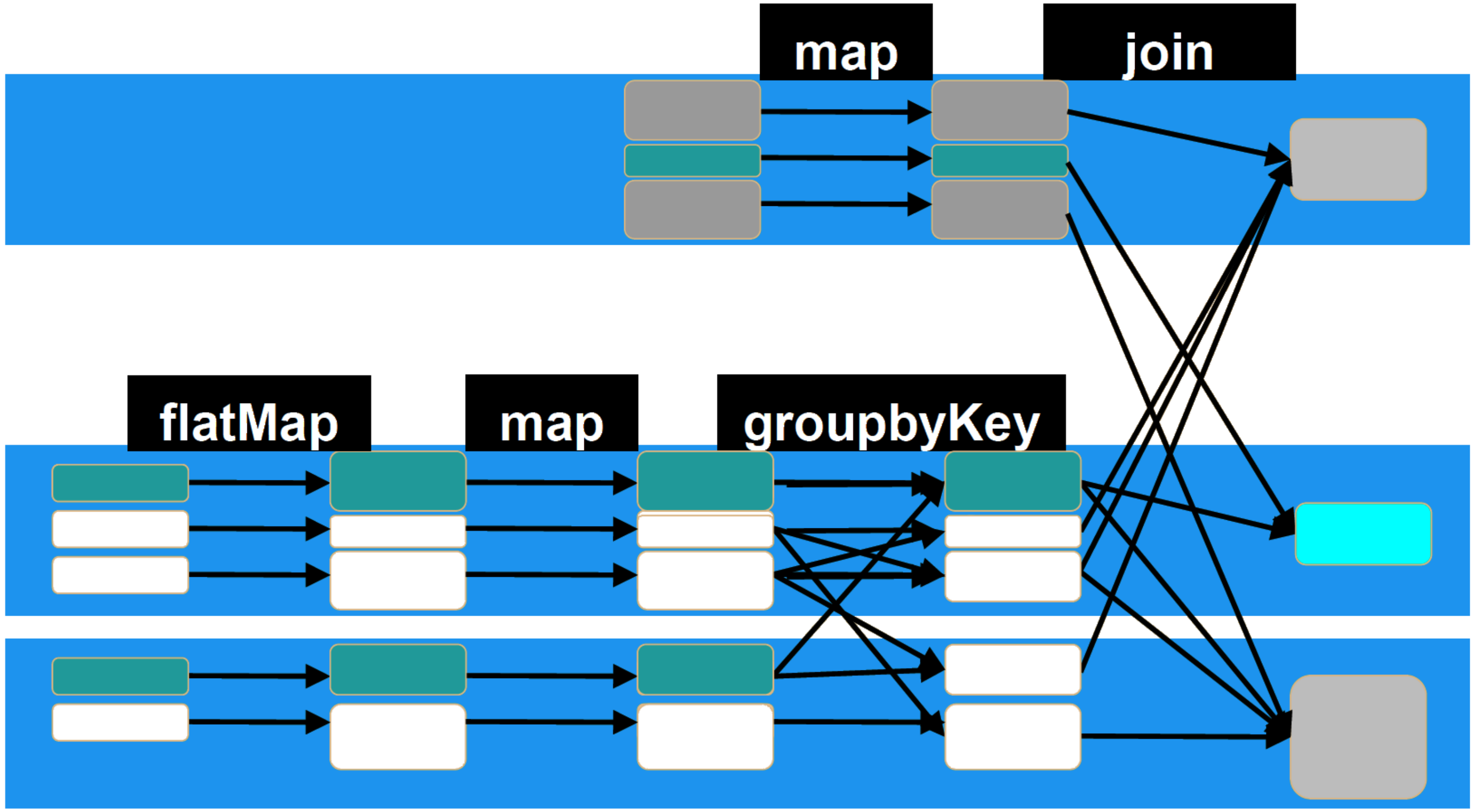


```
Key: A , Values: [1]
Key: ago , Values: [1]
Key: far , Values: [1, 1]
Key: away , Values: [1]
Key: in , Values: [1]
Key: long , Values: [1]
Key: a , Values: [1]
Key: time , Values: [1]
Key: galaxy , Values: [1]
```

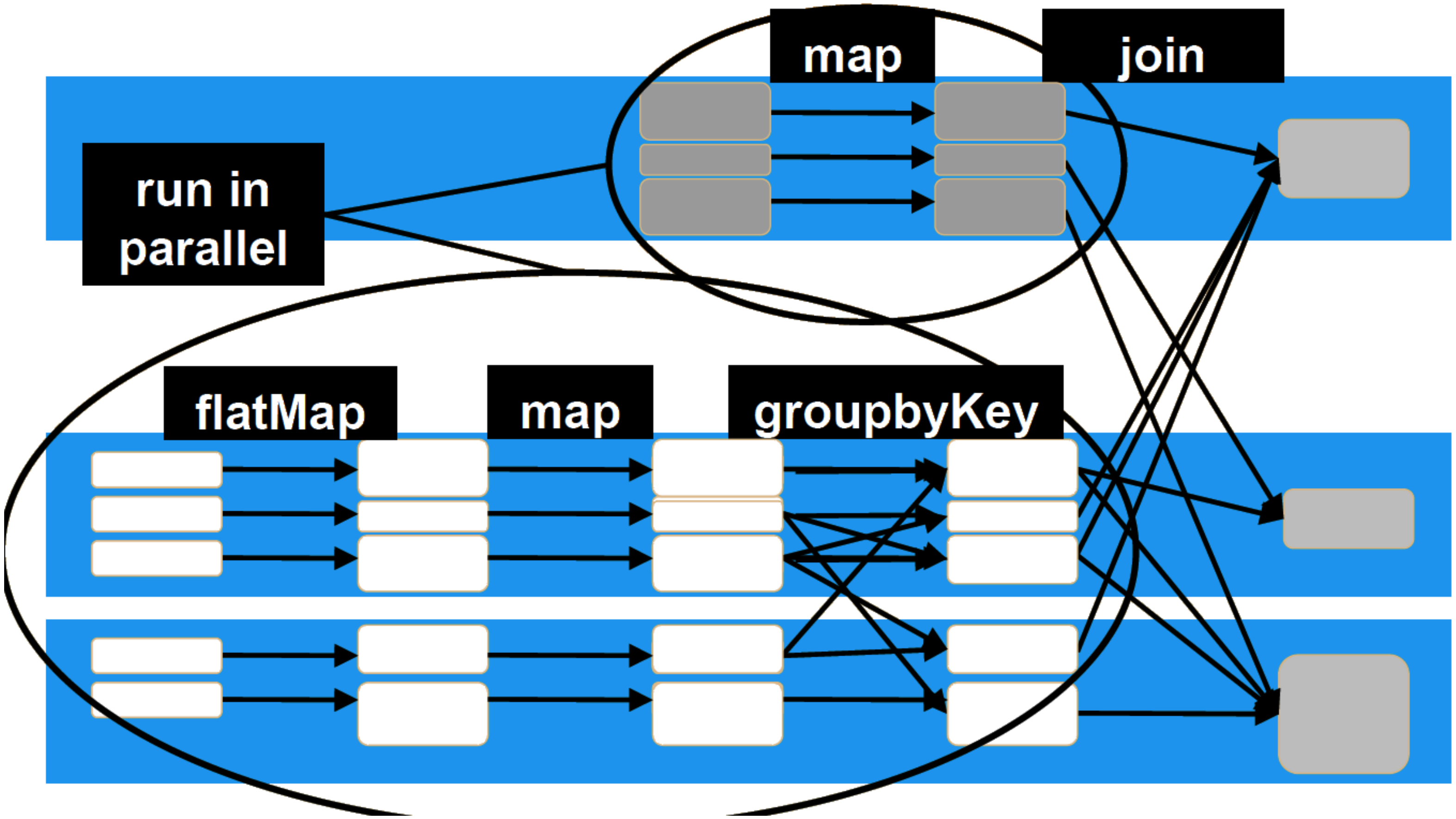
Spark DAG of transformations



Spark DAG of transformations with join



Spark DAG of transformations with join



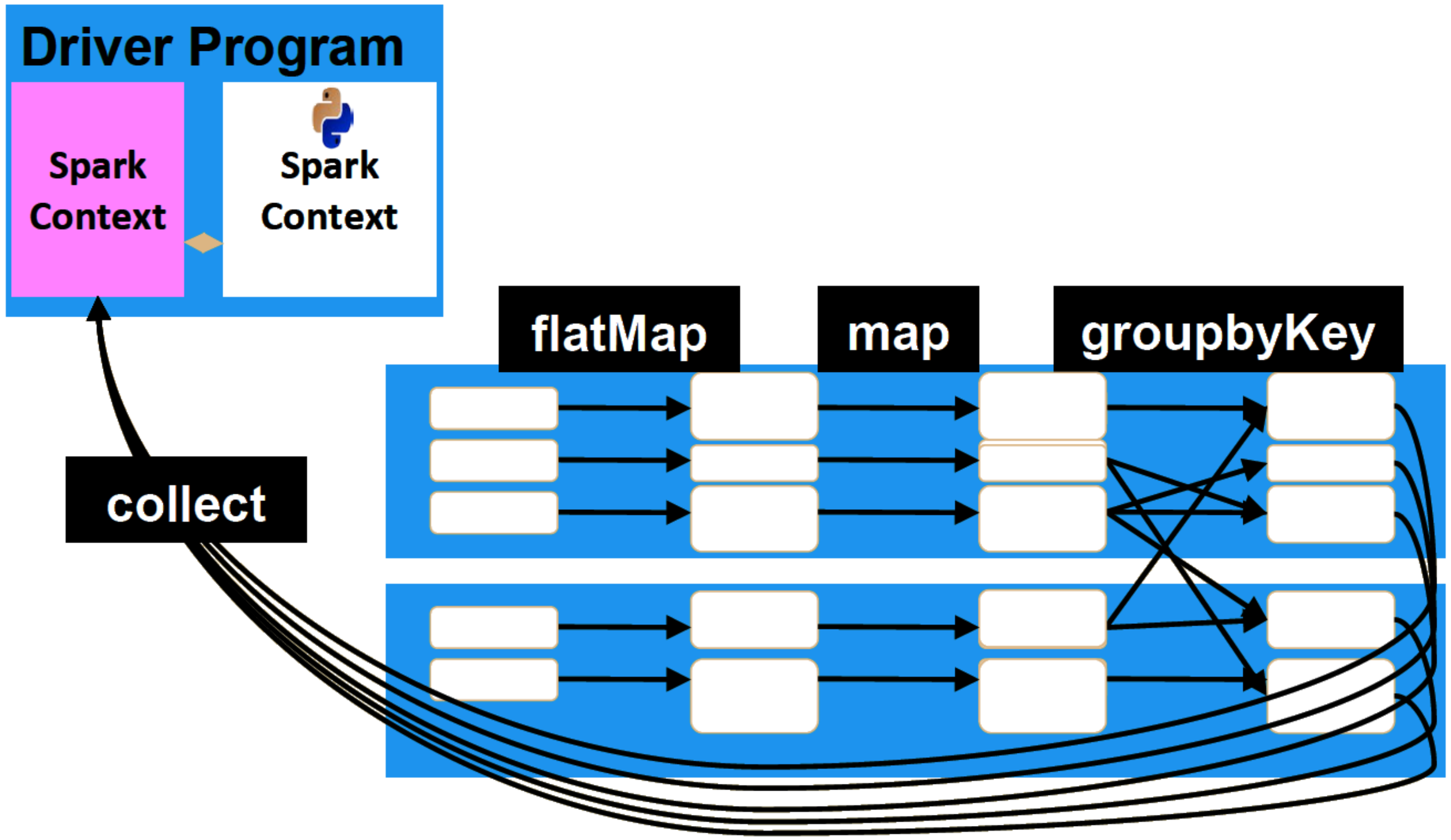
Action! What is an action?

› Final stage of workflow

› Triggers execution of the DAG

› Returns results to the Driver or writes to HDFS

Action! What is an action?



Actions

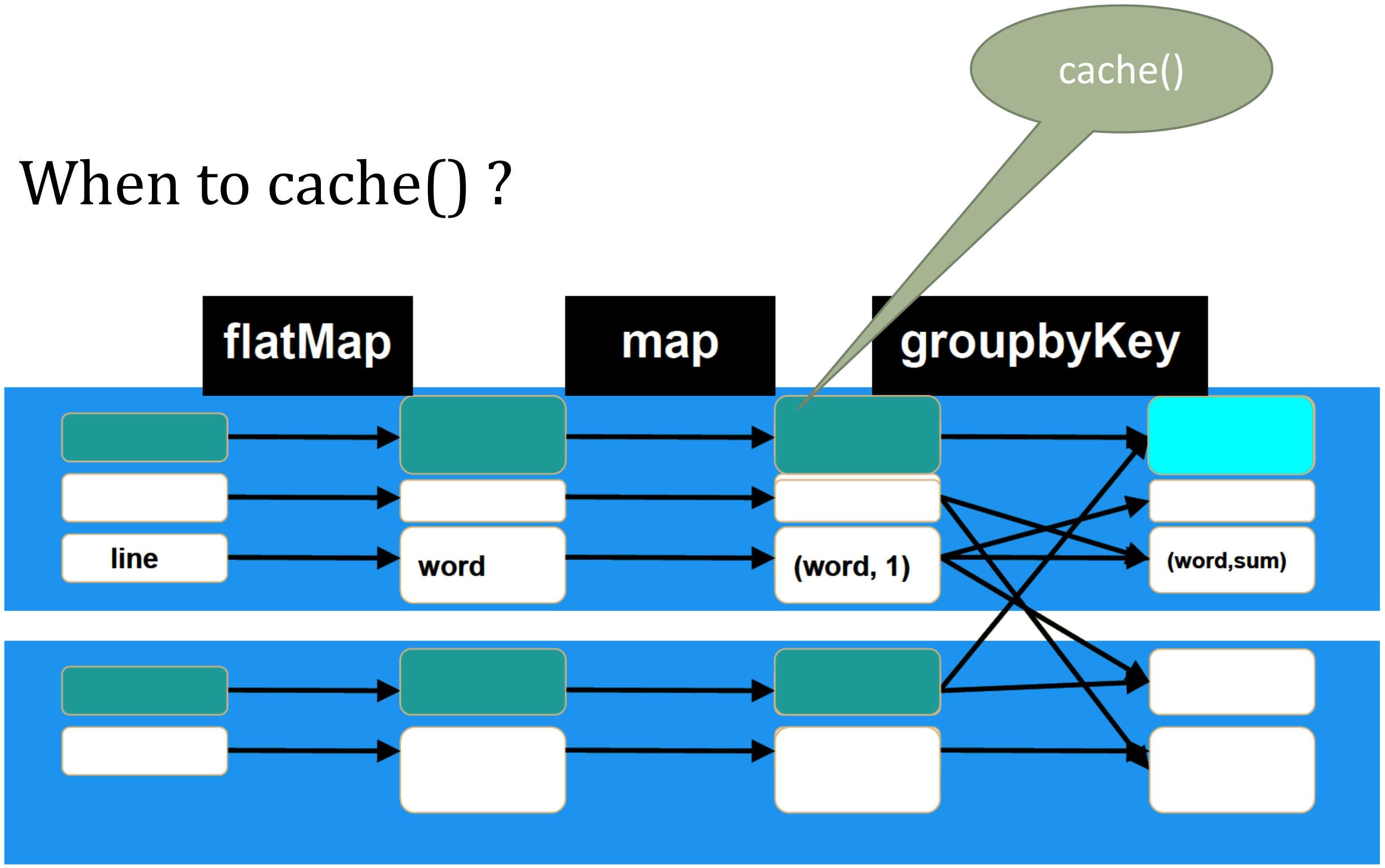
- `collect()`
  - copy all elements to the driver
- `take(n)`
  - copy first n elements
- `reduce(func)`
  - aggregate elements with func (takes 2 elements, returns 1)
- `saveAsTextFile(filename)`
  - save to local file or HDFS

Caching

- By default each job re-processes from HDFS
- Mark RDD with `.cache()`
  - Raw storage (fast and simple)
  - Serialized (for big data sets, with `persist()`)
- Lazy

When?
  - Generally not the input data
  - Do validation and cleaning
  - Cache for iterative algorithm
- How?
  - Memory (most common)
  - Disk (rare)
  - Both (for heavy calculations)

When to cache() ?



Broadcast variables

› Large variable used in all nodes
  – *E.g.* a large configuration dictionary or lookup table
› Transfer just once per Executor
› Efficient peer-to-peer transfer

```
config=sc.broadcast({"order":3,"filter":True})
config.value
```

Accumulator

- Common pattern of accumulating to a variable across the cluster
- Write-only on nodes

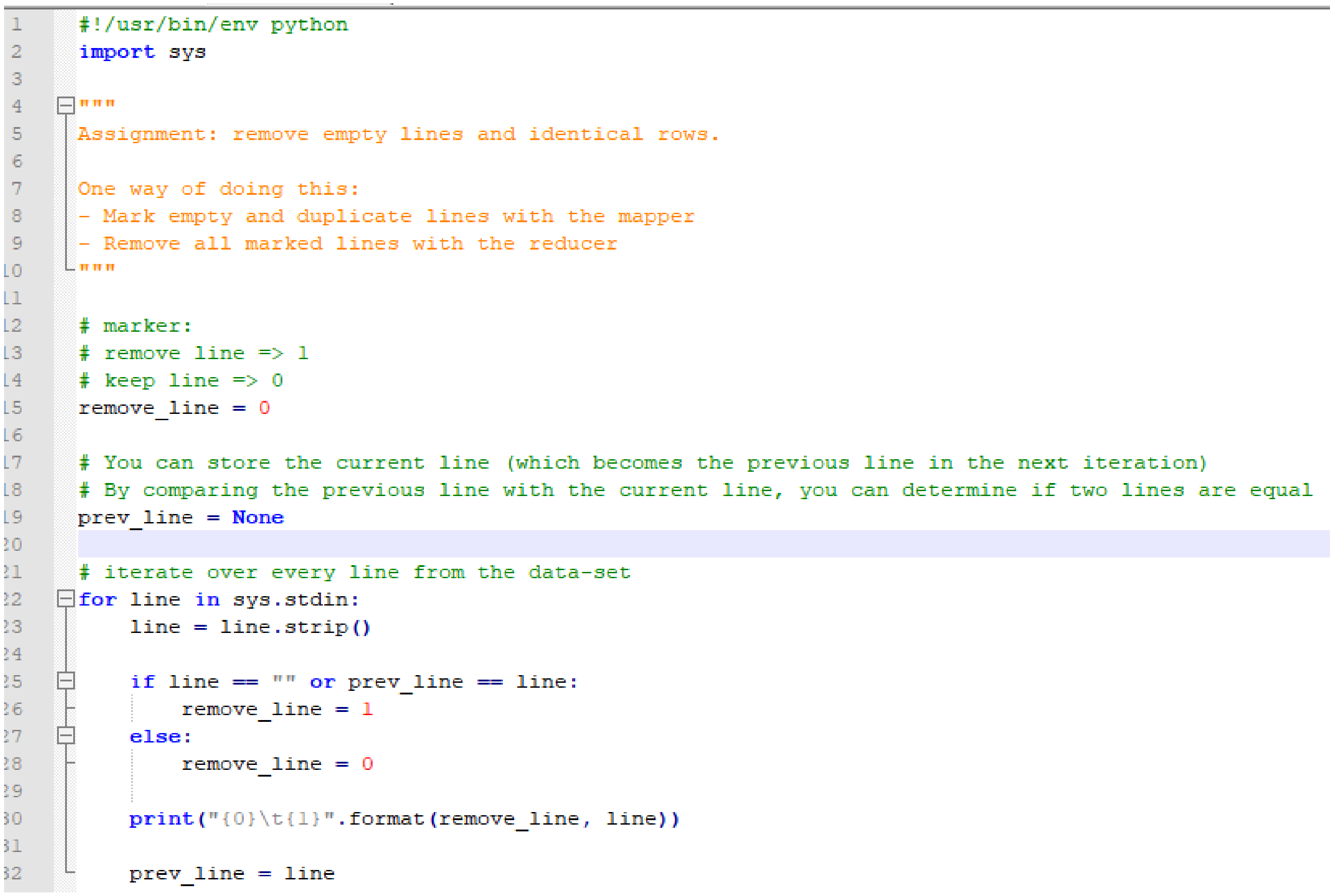
```
accum = sc.accumulator(0)

def test_accum(x):
   accum.add(x)

sc.parallelize([1,2,3,4]).foreach(test_accum)
accum.value

Out[]: ???
```

10

On Tutorial 3

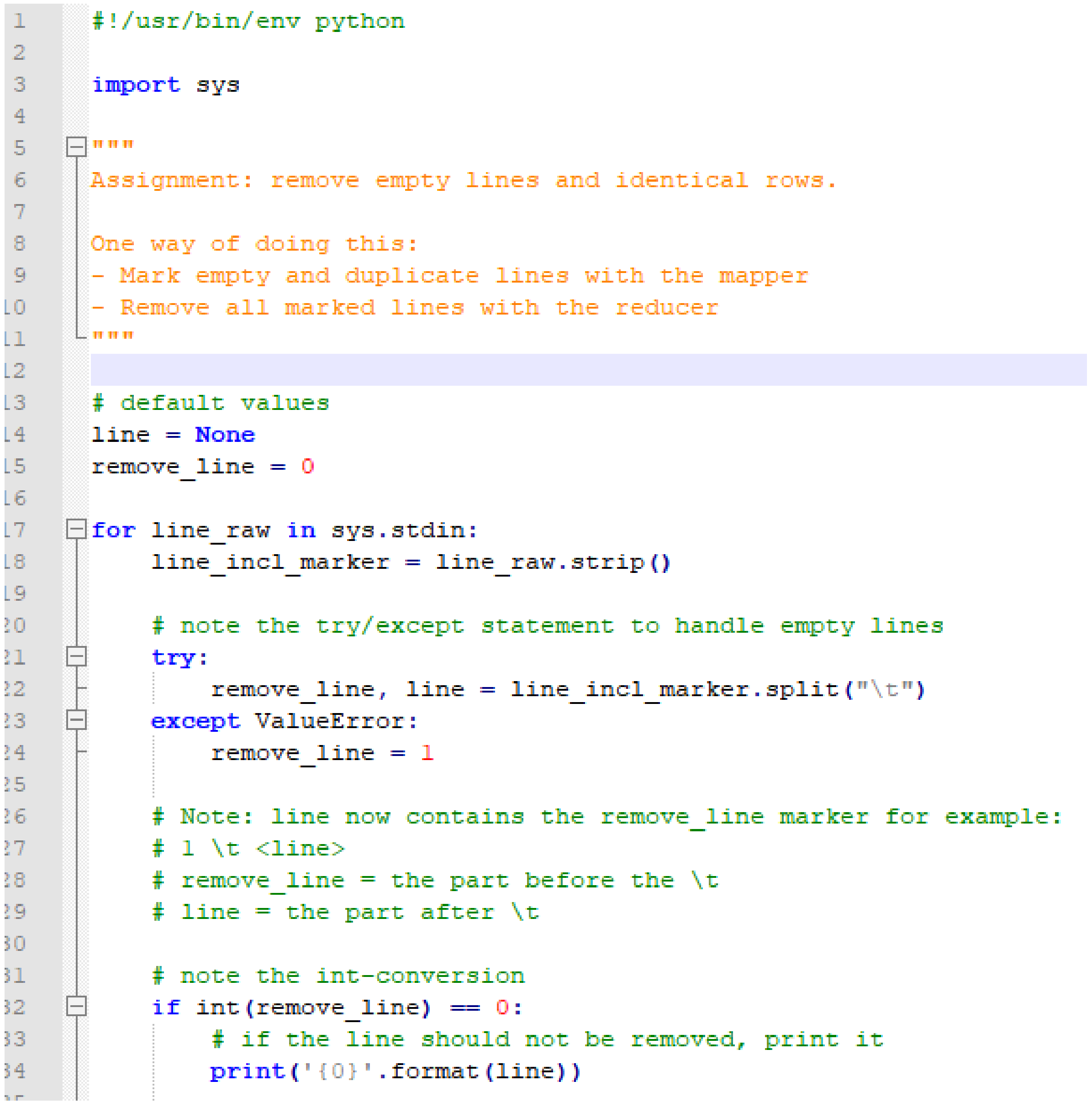
Neo in Hadoop, in LabA DSVM

Tutorial 3: Step 2: Mapper



```python
#!/usr/bin/env python
import sys

"""
Assignment: remove empty lines and identical rows.

One way of doing this:
- Mark empty and duplicate lines with the mapper
- Remove all marked lines with the reducer
"""

# marker:
# remove line => 1
# keep line => 0
remove_line = 0

# You can store the current line (which becomes the previous line in the next iteration)
# By comparing the previous line with the current line, you can determine if two lines are equal
prev_line = None

# iterate over every line from the data-set
for line in sys.stdin:
    line = line.strip()

    if line == "" or prev_line == line:
        remove_line = 1
    else:
        remove_line = 0

    print("{0}\t{1}".format(remove_line, line))

    prev_line = line
```

Tutorial 3: Step 2 - Reducer



```python
#!/usr/bin/env python

import sys

"""
Assignment: remove empty lines and identical rows.

One way of doing this:
- Mark empty and duplicate lines with the mapper
- Remove all marked lines with the reducer
"""

# default values
line = None
remove_line = 0

for line_raw in sys.stdin:
    line_incl_marker = line_raw.strip()

    # note the try/except statement to handle empty lines
    try:
        remove_line, line = line_incl_marker.split("\t")
    except ValueError:
        remove_line = 1

    # Note: line now contains the remove_line marker for example:
    # 1 \t <line>
    # remove_line = the part before the \t
    # line = the part after \t

    # note the int-conversion
    if int(remove_line) == 0:
        # if the line should not be removed, print it
        print('{0}'.format(line))
```

Problem modelling in Spark

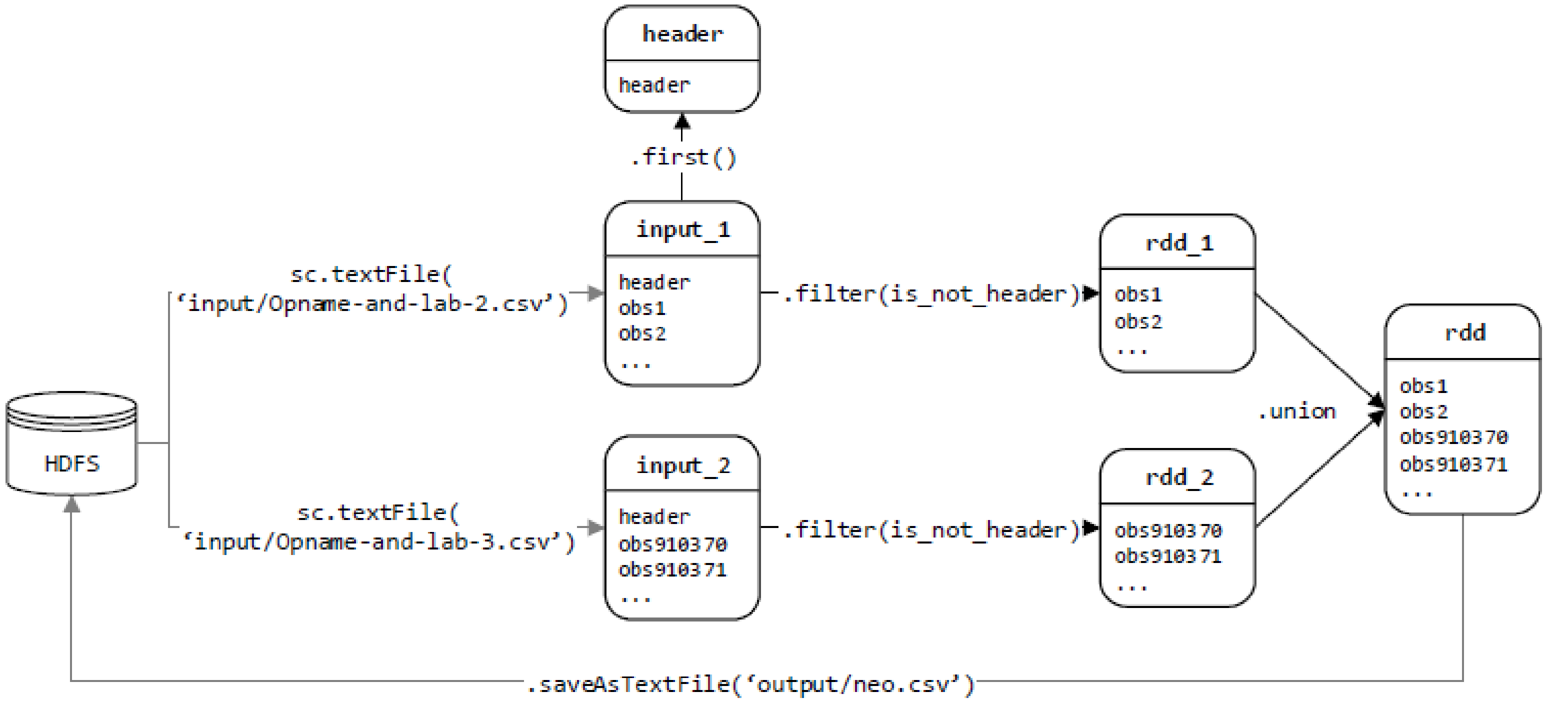
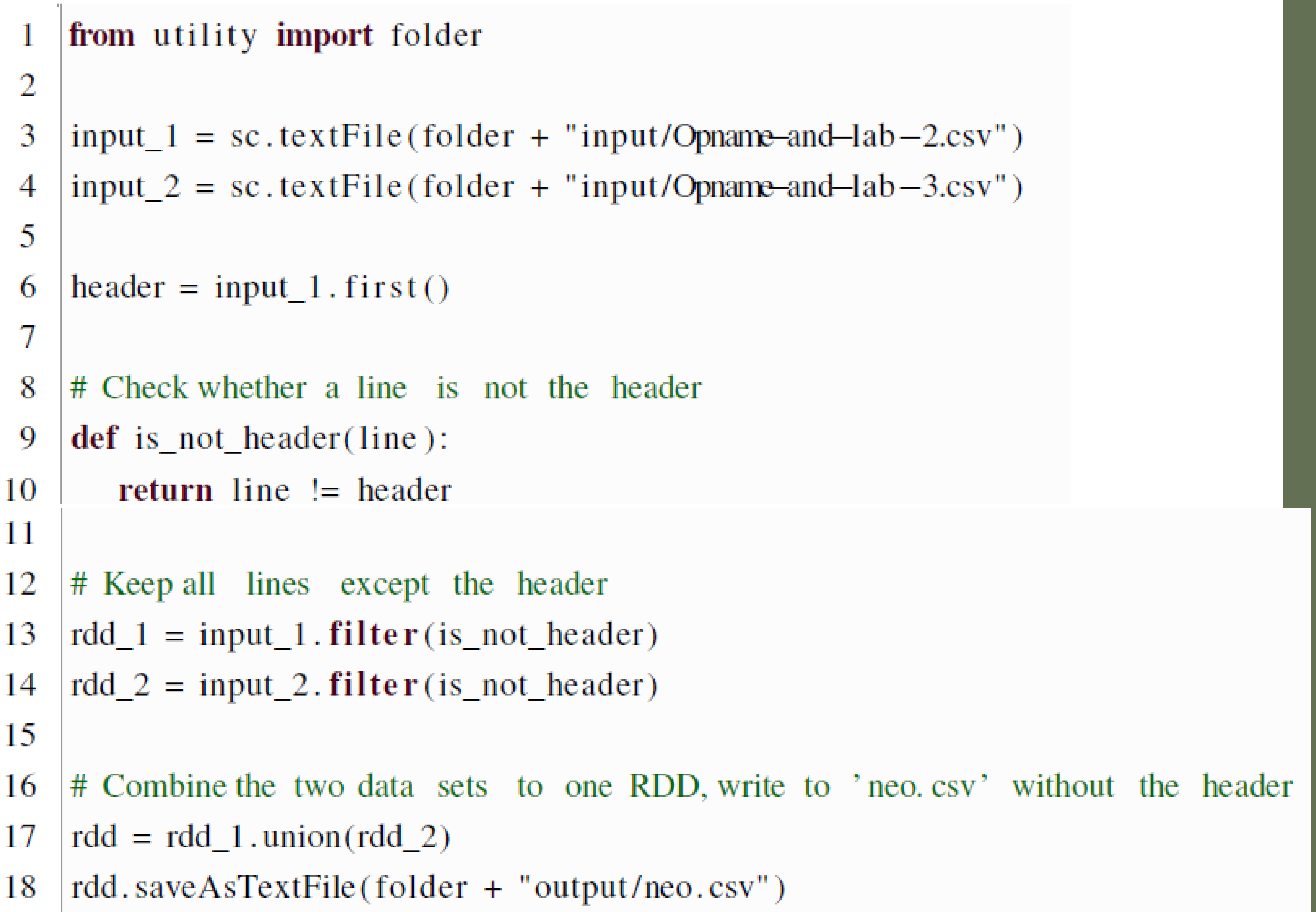


Figure 2.1 *DAG for question 1*

Preparing the RDD in Spark



```python
from utility import folder

input_1 = sc.textFile(folder + "input/Opname-and-lab-2.csv")
input_2 = sc.textFile(folder + "input/Opname-and-lab-3.csv")

header = input_1.first()

# Check whether a line is not the header
def is_not_header(line):
    return line != header

# Keep all lines except the header
rdd_1 = input_1.filter(is_not_header)
rdd_2 = input_2.filter(is_not_header)

# Combine the two data sets to one RDD, write to 'neo.csv' without the header
rdd = rdd_1.union(rdd_2)
rdd.saveAsTextFile(folder + "output/neo.csv")
```

Midterm Q/A

[Tentative!]
All multiple choice

Learning objectives in midterm exam

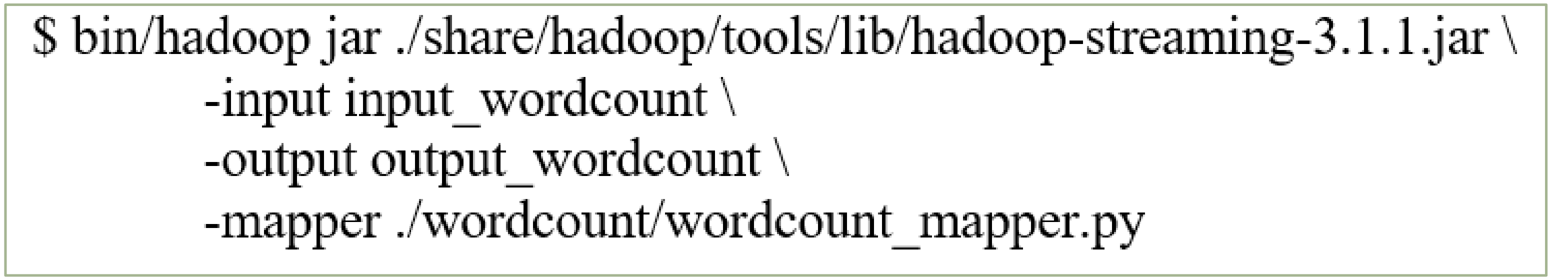
	Litera-ture	Work-shops	Guest talks	Regular lectures	TOTAL
Understand the role of data science and its societal impact	5		4	5	14
Recognise the knowledge discovery processes in applied data science	10	10	2	5	27
Identify trends and developments in big data engineering & analytics	10		2	5	17
Apply selected big data technologies to solve real-world problems		30	2	10	42
TOTAL	25	40	10	25	**100**

Example Q: tutorial 1

› What command can be used to display the content of file?
  a) Cat
  b) Li
  c) Echo
  d) Ls

Example Q: tutorial 2

› Analyse the following code fragment. What does this command do?



```
$ bin/hadoop jar ./share/hadoop/tools/lib/hadoop-streaming-3.1.1.jar \
        -input input_wordcount \
        -output output_wordcount \
        -mapper ./wordcount/wordcount_mapper.py
```

a) It processes an input file with hadoop

b) Nothing; it will report an error stating that there is no jar file

c) Nothing; it will report that there is no reducer

d) It will produce three output files in output_wordcount

Example Q: literature

› Why do Davenport & Patil (2012) refer to Data scientist as being the Sexiest Job of the 21st Century?

a) Data scientists want to build things, not just give advice in the role of consultant

b) The shortage of data scientists is becoming a serious constraint in some sectors

c) Data scientists today are akin to the Wall Street “quants” of the 1980s and 1990s

d) All of the above

Example Q: regular lectures, literature

› What is generally considered to be part of the Apache Basic Hadoop Modules?

a) YARN

b) Impala

c) MapReduce

d) HDFS