Yandex

## Caching & persistence

## Quick reminder

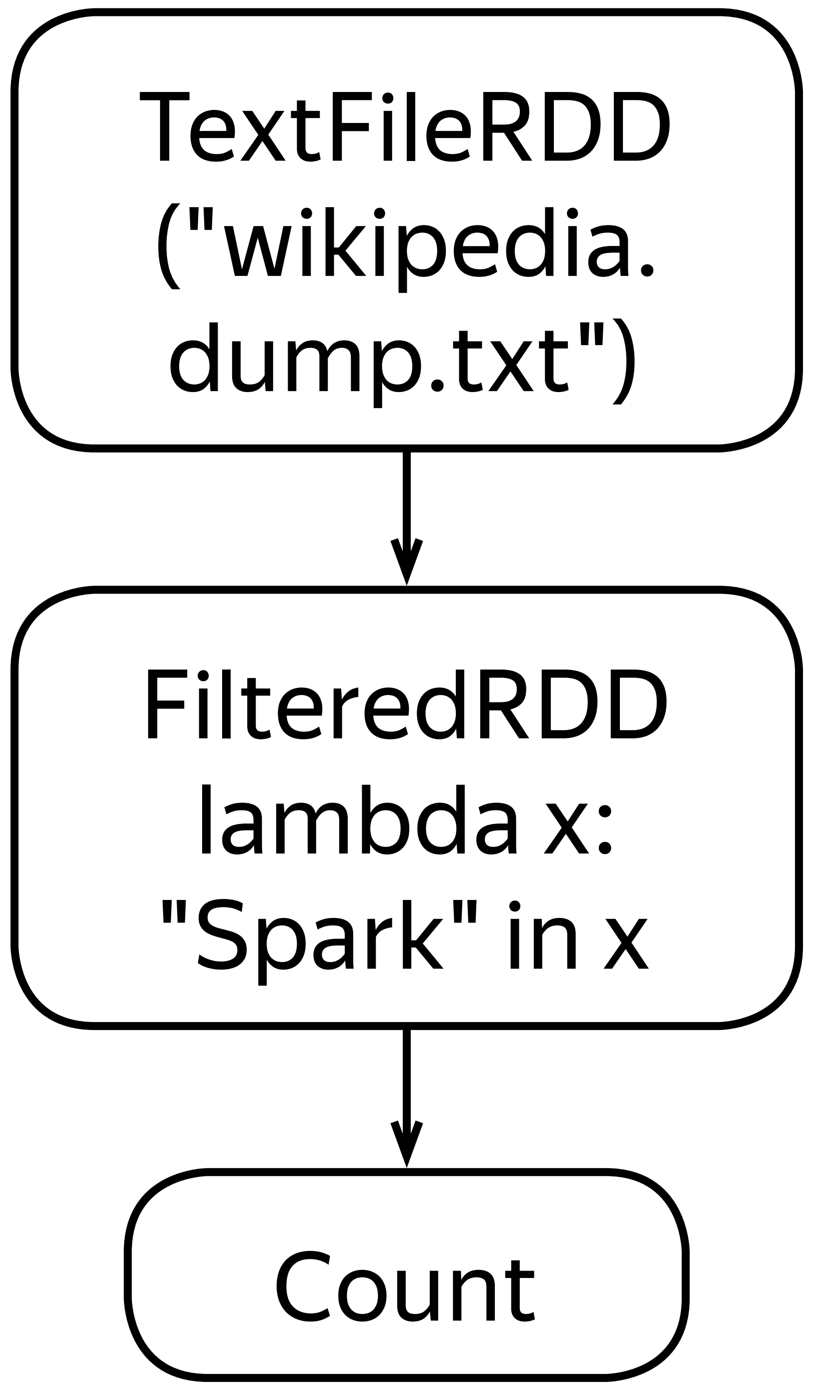
- RDDs are partitioned
- Execution is build around the partitions
- Block is a unit of input and output in Spark

## Motivating example

```
sc = SparkContext(...)
```

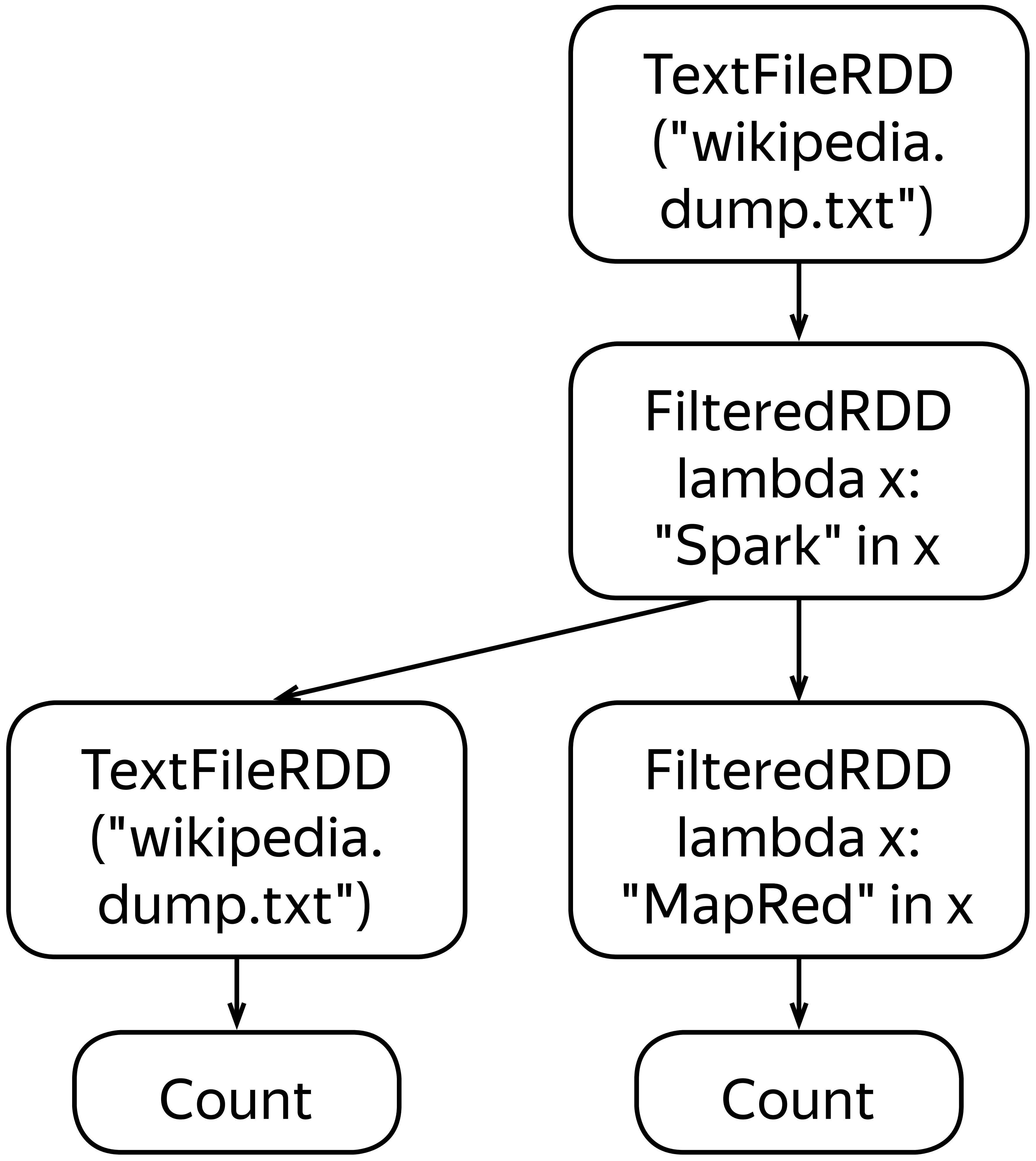
## Motivating example

```
sc = SparkContext(...)
wiki = sc.textFile("wikipedia.dump.txt")

spark_articles = wiki.filter(
                  lambda x: "Spark" in x)

print(spark_articles.count())
```


TextFileRDD ("wikipedia. dump.txt")
FilteredRDD lambda x: "Spark" in x
Count

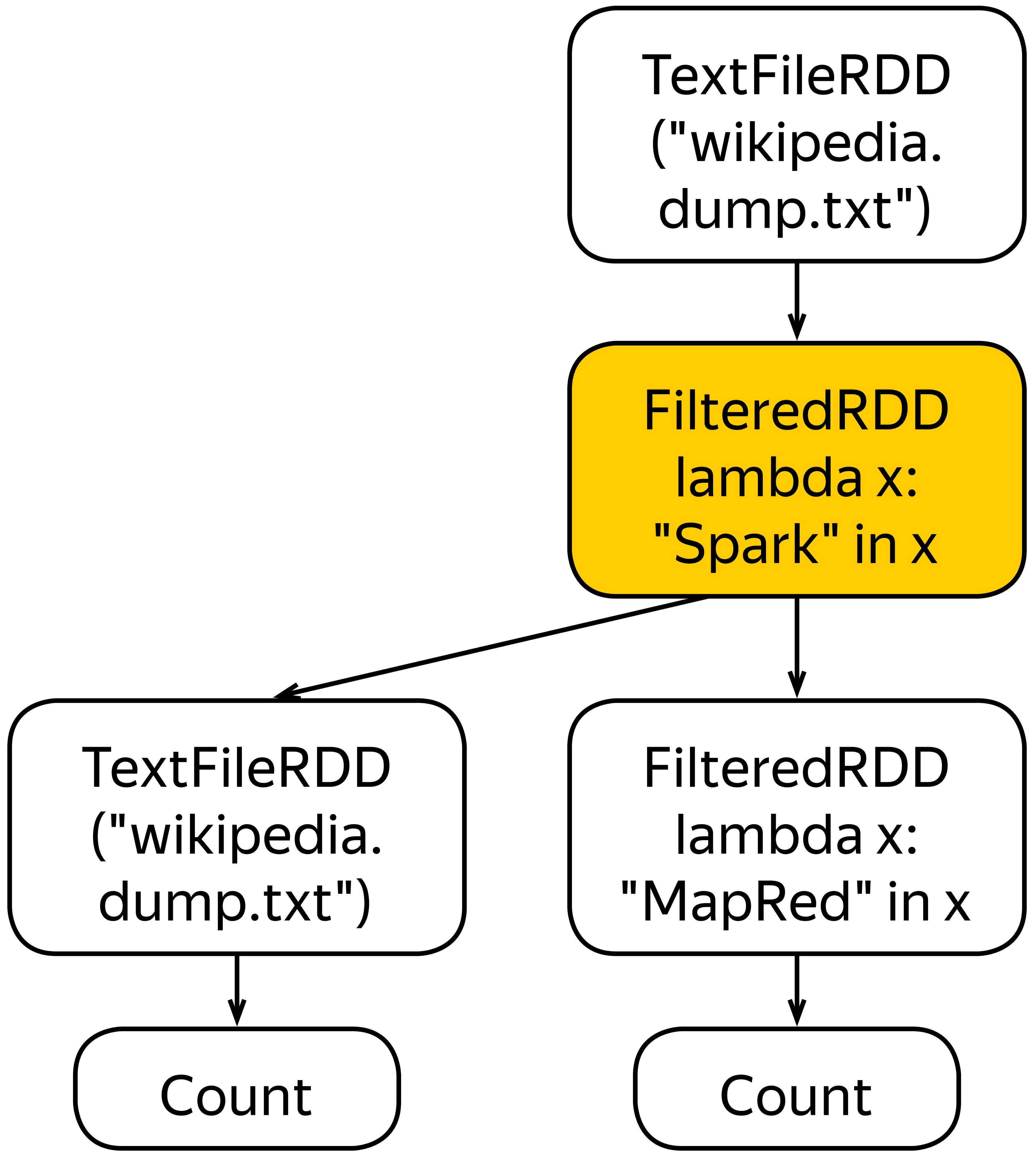
## Motivating example

```
sc = SparkContext(...)
wiki = sc.textFile("wikipedia.dump.txt")

spark_articles = wiki.filter(
                 lambda x: "Spark" in x)

hadoop_articles = spark_articles.filter(
                 lambda x: "Hadoop" in x)
mapreduce_articles = spark_articles.filter(
                 lambda x: "MapRed" in x)
print(hadoop_articles.count())
print(mapreduce_articles.count())
```


TextFileRDD ("wikipedia. dump.txt")
FilteredRDD lambda x: "Spark" in x
TextFileRDD ("wikipedia. dump.txt")
FilteredRDD lambda x: "MapRed" in x
Count
Count

## Motivating example

```
sc = SparkContext(...)
wiki = sc.textFile("wikipedia.dump.txt")

spark_articles = wiki.filter(
                lambda x: "Spark" in x)
spark_articles.cache()
hadoop_articles = spark_articles.filter(
                lambda x: "Hadoop" in x)
mapreduce_articles = spark_articles.filter(
                lambda x: "MapRed" in x)
print(hadoop_articles.count())
print(mapreduce_articles.count())
```


TextFileRDD ("wikipedia. dump.txt")
FilteredRDD lambda x: "Spark" in x
TextFileRDD ("wikipedia. dump.txt")
FilteredRDD lambda x: "MapRed" in x
Count
Count

# Controlling persistence level

- rdd.persist(storageLevel)
  - sets RDD’s storage to persist across operations after it is computed for the first time
  - storageLevel is a set of flags controlling the persistence, typical values are
    DISK_ONLY
        – save the data to the disk,
    MEMORY_ONLY
        – keep the data in the memory
    MEMORY_AND_DISK
        – keep the data in the memory; when out of memory – save it to the disk
    DISK_ONLY_2, MEMORY_ONLY_2, MEMORY_AND_DISK_2
        – same as about, but make two replicas
- rdd.cache() = rdd.persist(MEMORY_ONLY)

## Controlling persistence level

- rdd.persist(storageLevel)
  - sets RDD's storage to persist across operations after it is computed for the first time
  - storageLevel is a set of flags controlling the persistence, typical values are

    DISK_ONLY

    – save the data to the disk,

    MEMORY_ONLY

    – keep the data in the memory

    MEMORY_AND_DISK

    – keep the data in the memory; when out of memory – save it to the disk

    DISK_ONLY_2, MEMORY_ONLY_2, MEMORY_AND_DISK_2

    – same as about, but make two replicas ← improves failure recovery times!
- rdd.cache() = rdd.persist(MEMORY_ONLY)

## Best practices

## Best practices

- For interactive sessions
  - cache preprocessed data

## Best practices

- For interactive sessions
  - cache preprocessed data

- For batch computations
  - cache dictionaries
  - cache other datasets that are accessed multiple times

## Best practices

- For interactive sessions
  - cache preprocessed data
- For batch computations
  - cache dictionaries
  - cache other datasets that are accessed multiple times
- For iterative computations
  - cache static data
- And do benchmarks!

## Summary

- › Performance may be improved by persisting data across operations
  - › in interactive sessions, iterative computations and hot datasets
- › You can control the persistence of a dataset
  - › whether to store in the memory or on the disk
  - › how many replicas to create

**Big**DATAteam