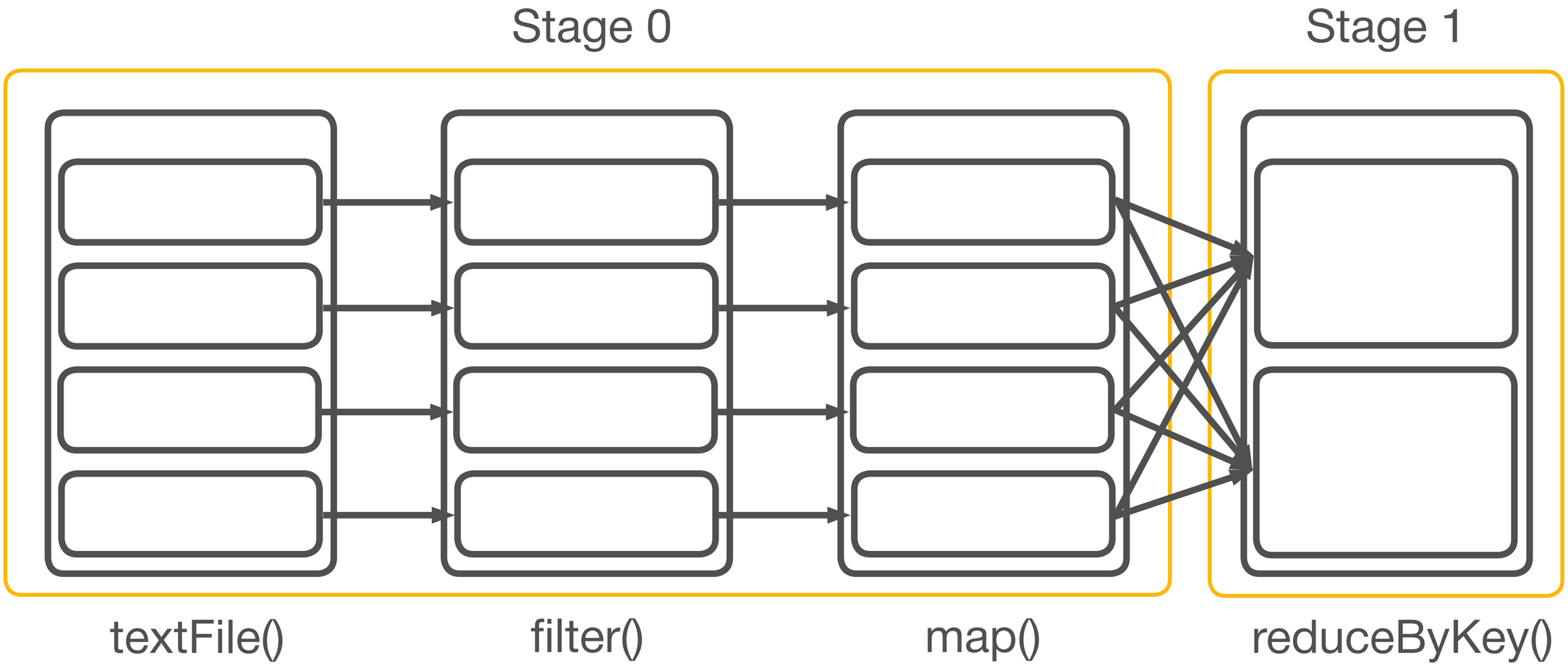
Shuffle. Partitioning

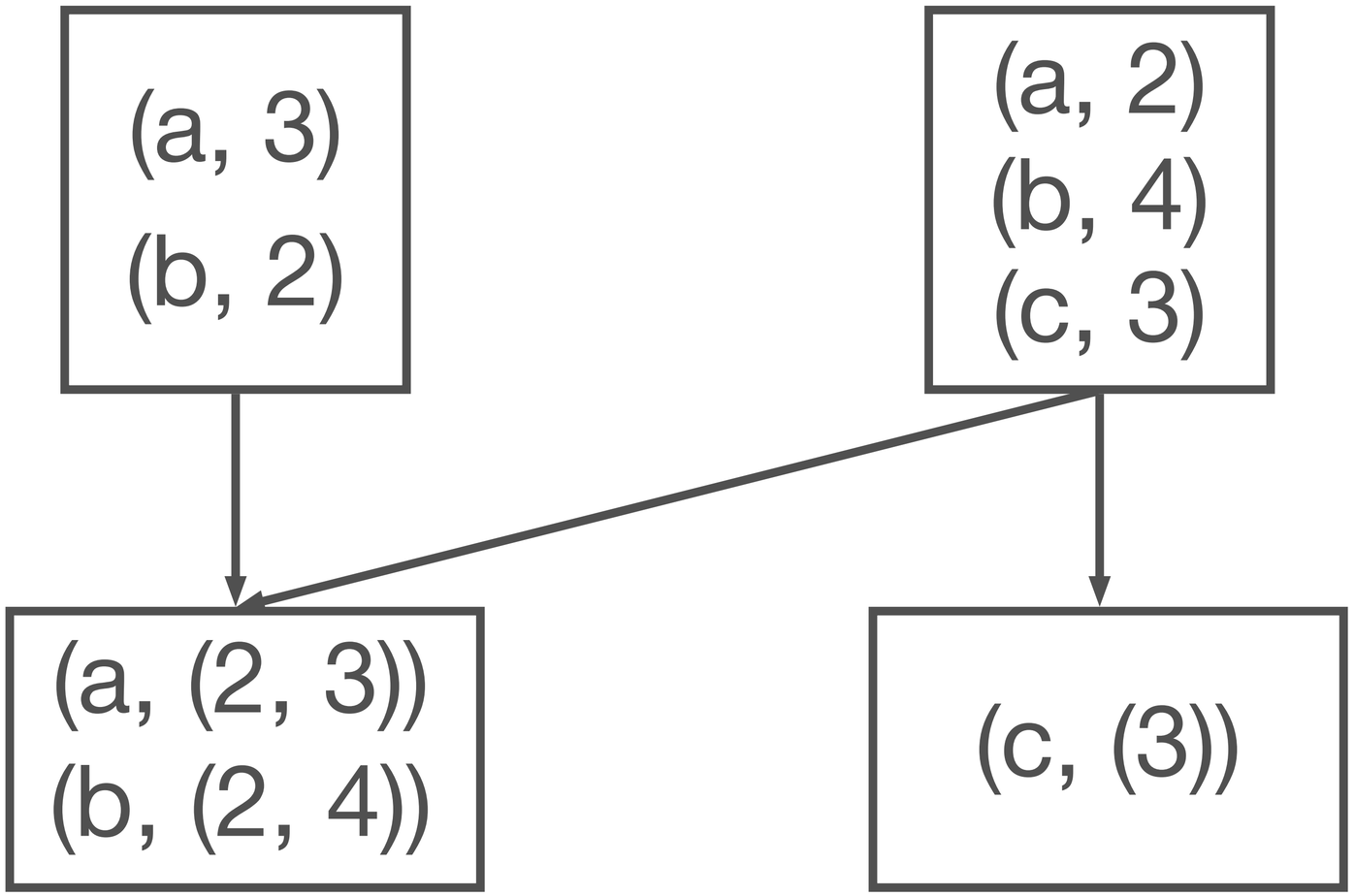

Stage 0
Stage 1
textFile()
filter()
map()
reduceByKey()

Duration	Task Deserialization Time	GC Time	Result Serialization Time	Write Time	Shuffle Write Size / Records
2 s	37 ms	0.1 s	0 ms	3 ms	217.0 B / 3
2 s	61 ms	87 ms	0 ms	6 ms	217.0 B / 3
2 s	44 ms	0.1 s	0 ms	3 ms	217.0 B / 3
3 s	40 ms	84 ms	0 ms	2 ms	217.0 B / 3

Spark has to answer two questions

- what executors to send data to?
- how to do it?

groupByKey()



The **partitioner** defines how records will be distributed and thus which records will be completed by each task

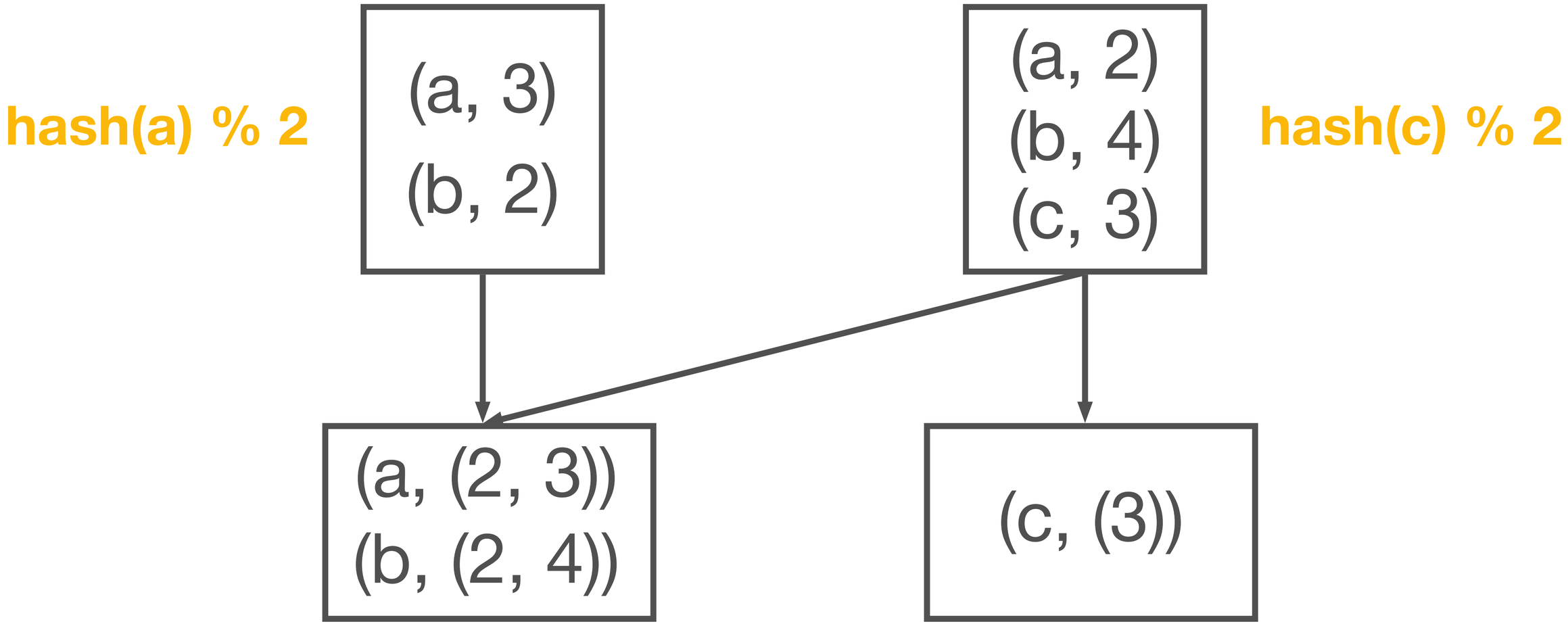
Interface with 2 methods:

- `numPartitions` – number of partitions in RDD after partitioning
- `getPartition` – mapping from a key to the index of partition

```
groupByKey(numPartitions=None,
           partitionFunc=<function portable_hash…>)
```

Group the values for each key in the RDD into a single sequence.
Hash-partitions the resulting RDD with *numPartitions* partitions.

groupByKey()



Spark partitioners

- hash partitioner
- range partitioner (e.g. sorting)

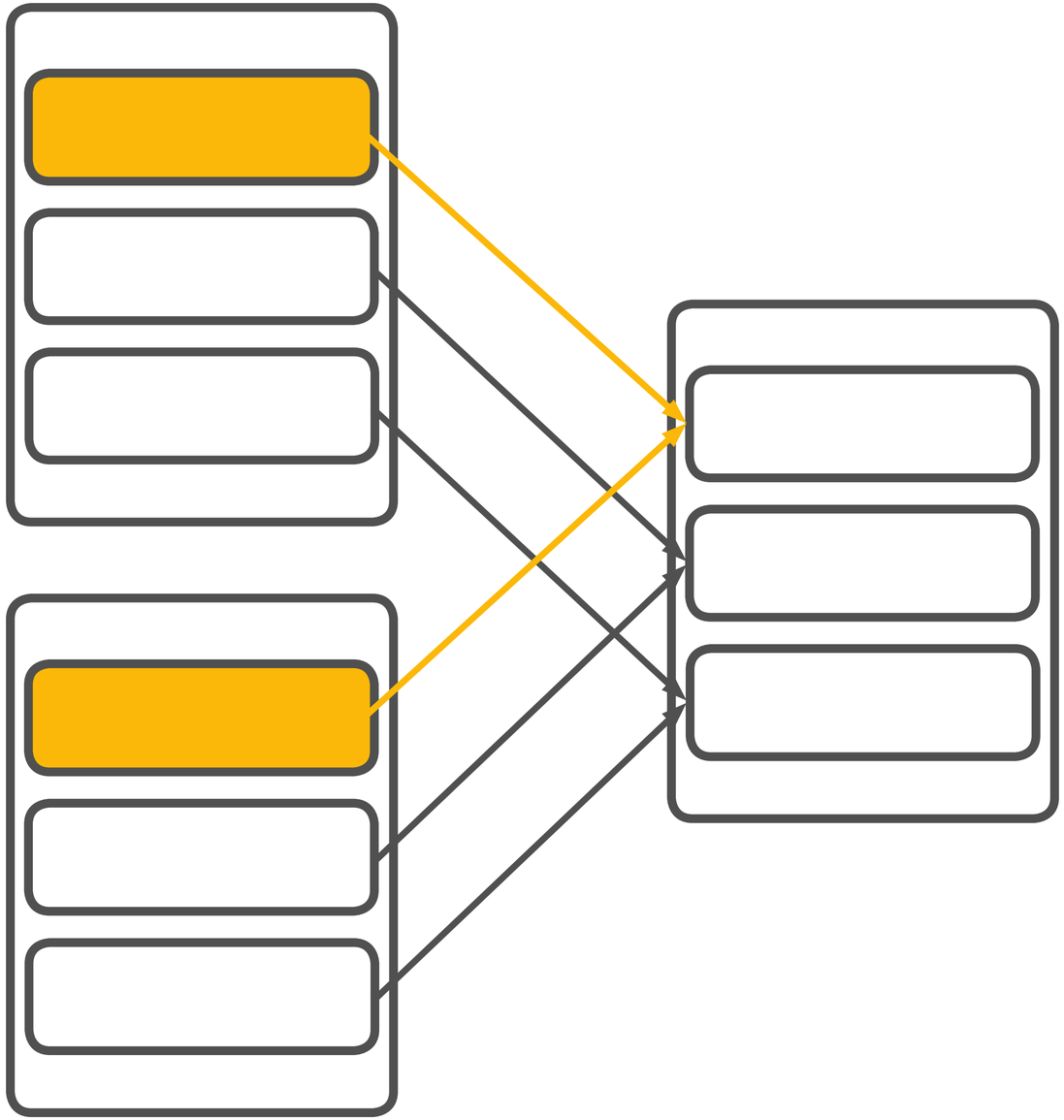
Known partitioner

```
>>> rdd = sc.textFile("log.txt")
>>> print rdd.paritioner
None

>>> rdd = sc.textFile("log.txt").partitionBy(10)
>>> print rdd.partitioner
<pyspark.rdd.Partitioner object at 0x7f4589395290>
```

RDDs are ***co-partitioned*** if they are partitioned by the same known partitioner

```
>>> rdd1 = sc.parallelize([1, 2, 3]).partitionBy(10)
>>> rdd2 = sc.parallelize([4, 5, 6]).partitionBy(10)
>>> rdd1.partitioner == rdd2.partitioner
True
```



Partitions are ***co-located*** if they are both loaded into memory on the same machine

```
>>> rdd1 = sc.parallelize([1, 2, 3]).partitionBy(10)
>>> rdd1.cache()
>>> rdd2 = sc.parallelize([4, 5, 6]).partitionBy(10)
>>> rdd2.cache()
>>> rdd1.partitioner == rdd2.partitioner
True
```

Transformations preserving partitioning

```
map(f, preservesPartitioning=False)
flatMap(f, preservesPartitioning=False)
mapPartitions(f, preservesPartitioning=False)
mapPartitionsWithIndex(f, preservesPartitioning=False)
mapValues(f)
flatMapValues(f)
```

Summary

- The partitioner defines how records will be distributed
  - numPartitions
  - getPartition
- Default partitioner uses key hashing
- Co-partitioned RDDs reduce the volume of the shuffle
- Co-located RDDs don’t shuffle
- Preserve the partitioner to avoid shuffles