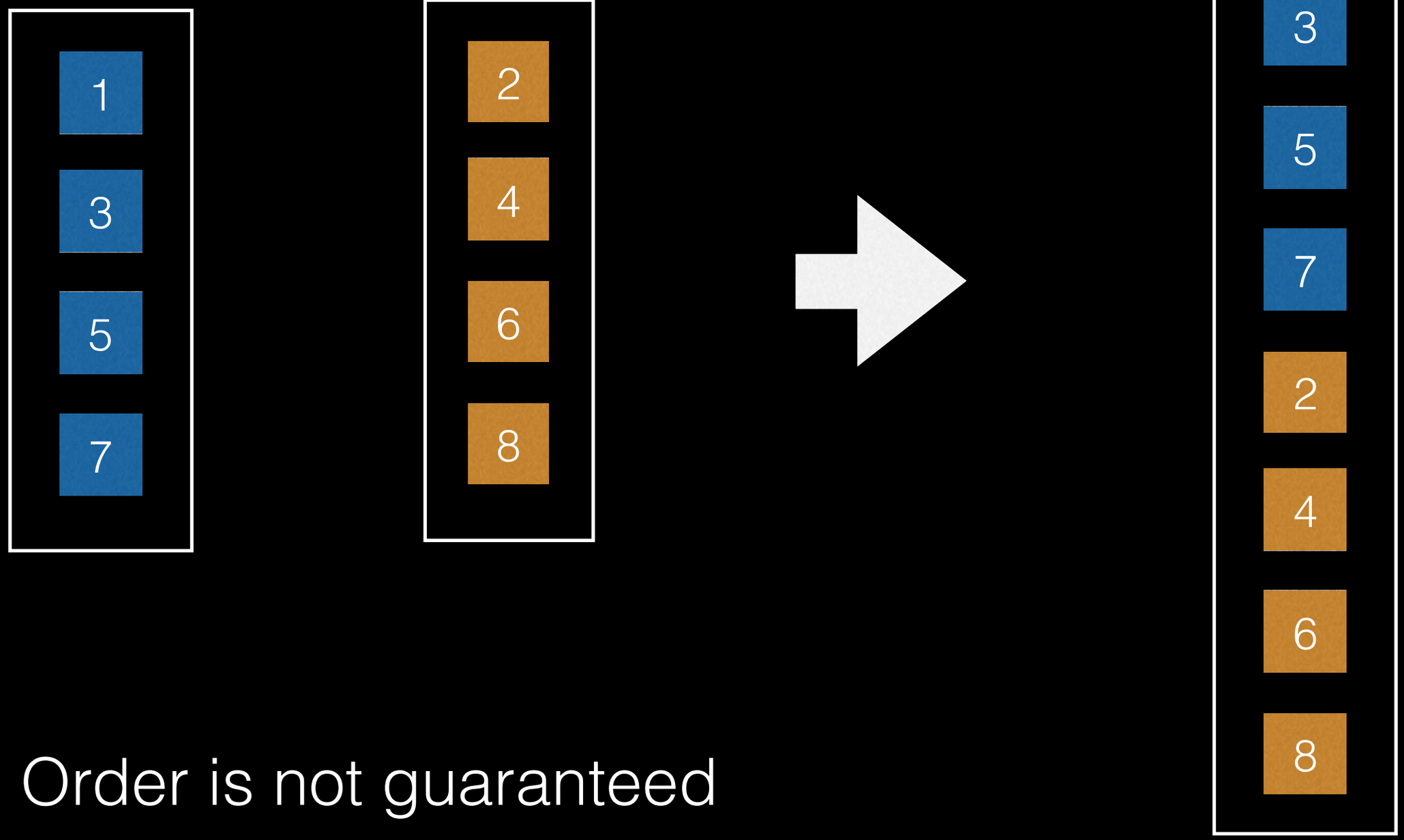
Transformations

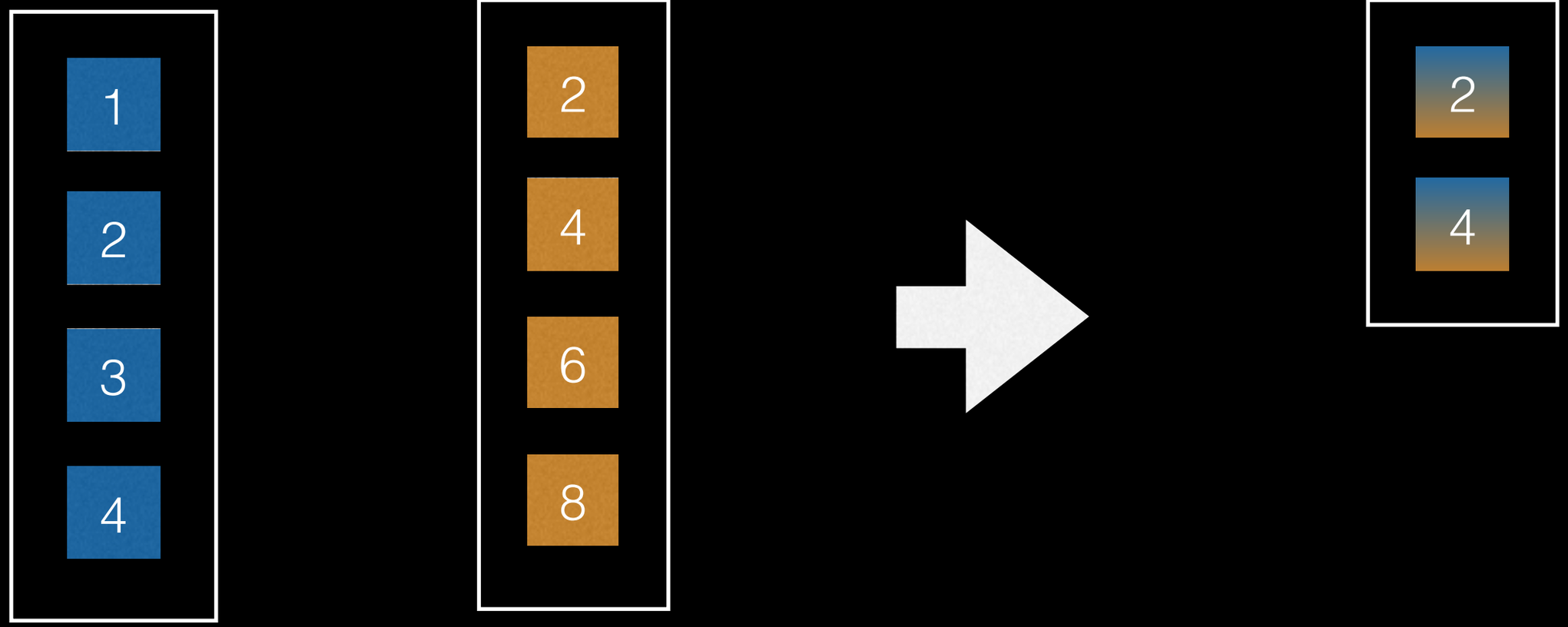
- union
- intersection
- subtract
- cartesian

union

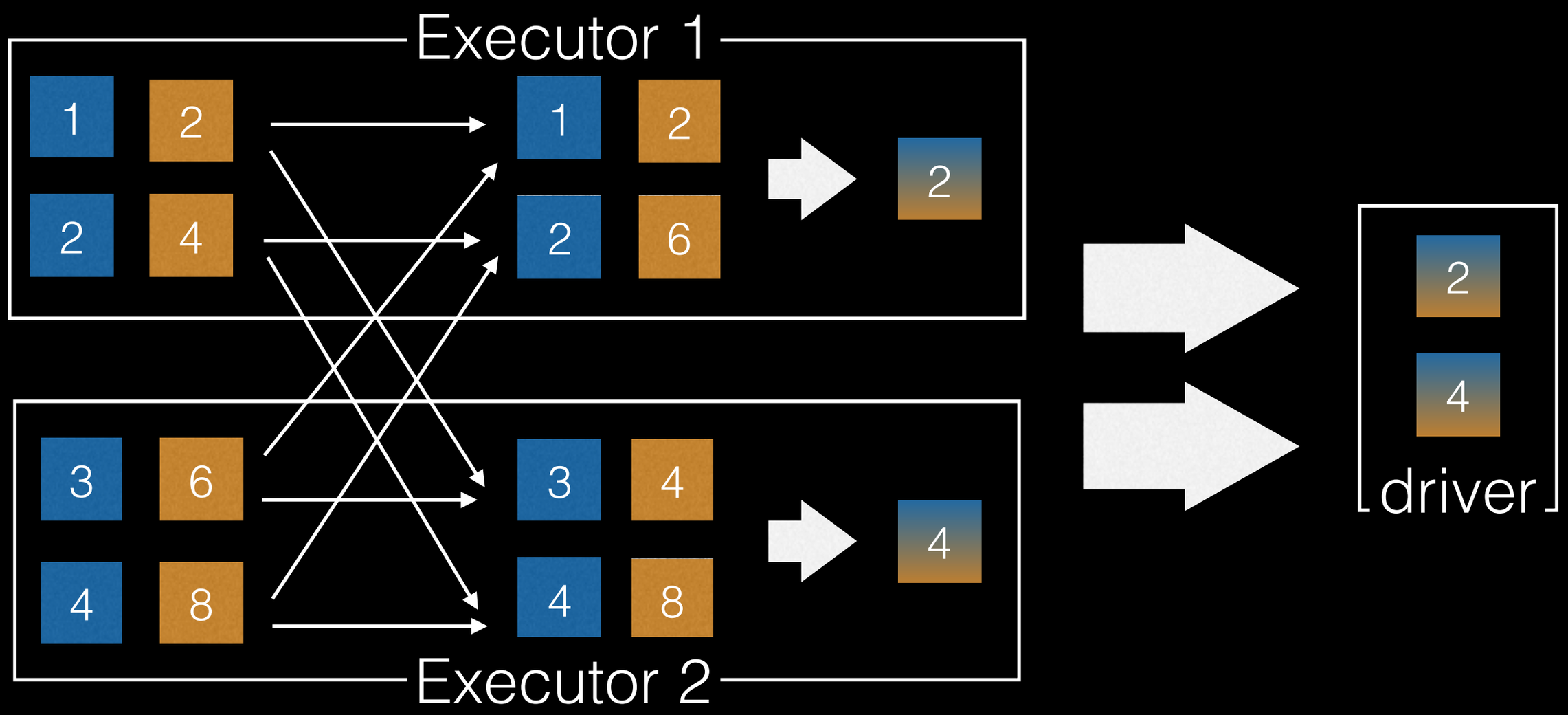


- Order is not guaranteed

intersection

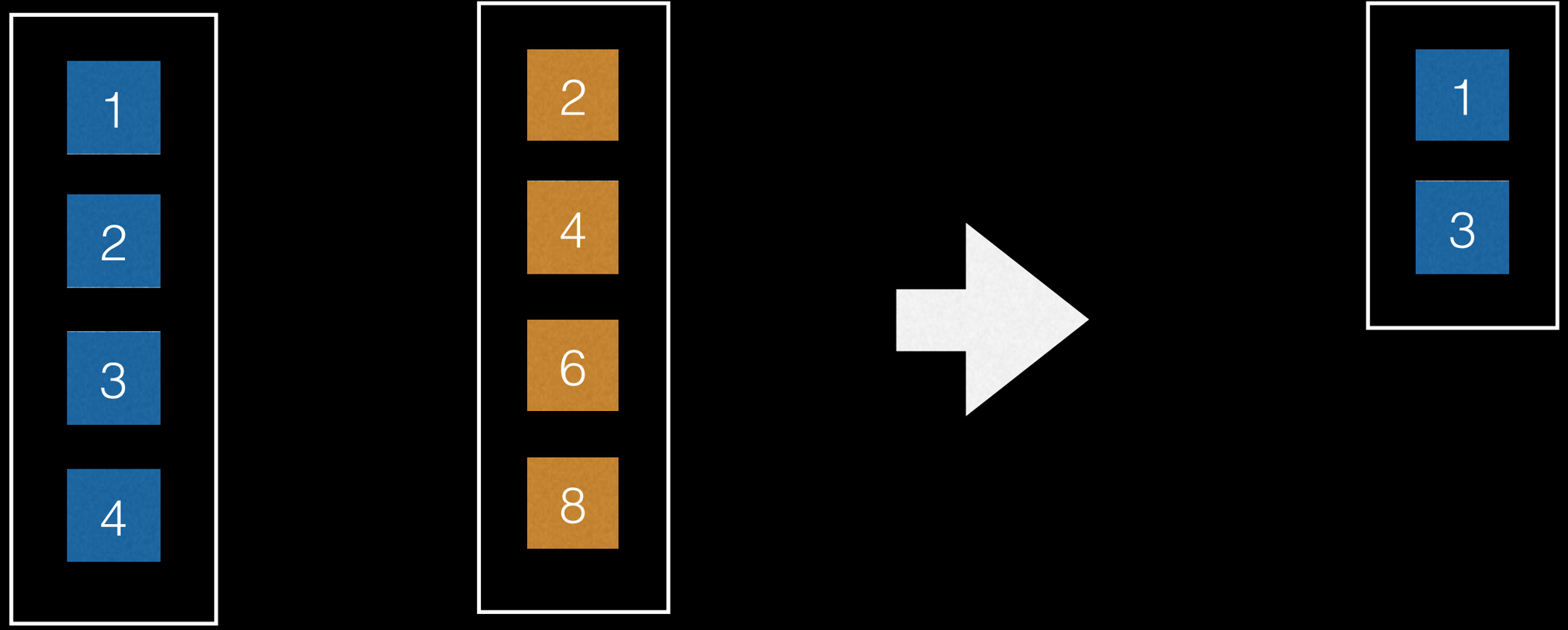


intersection

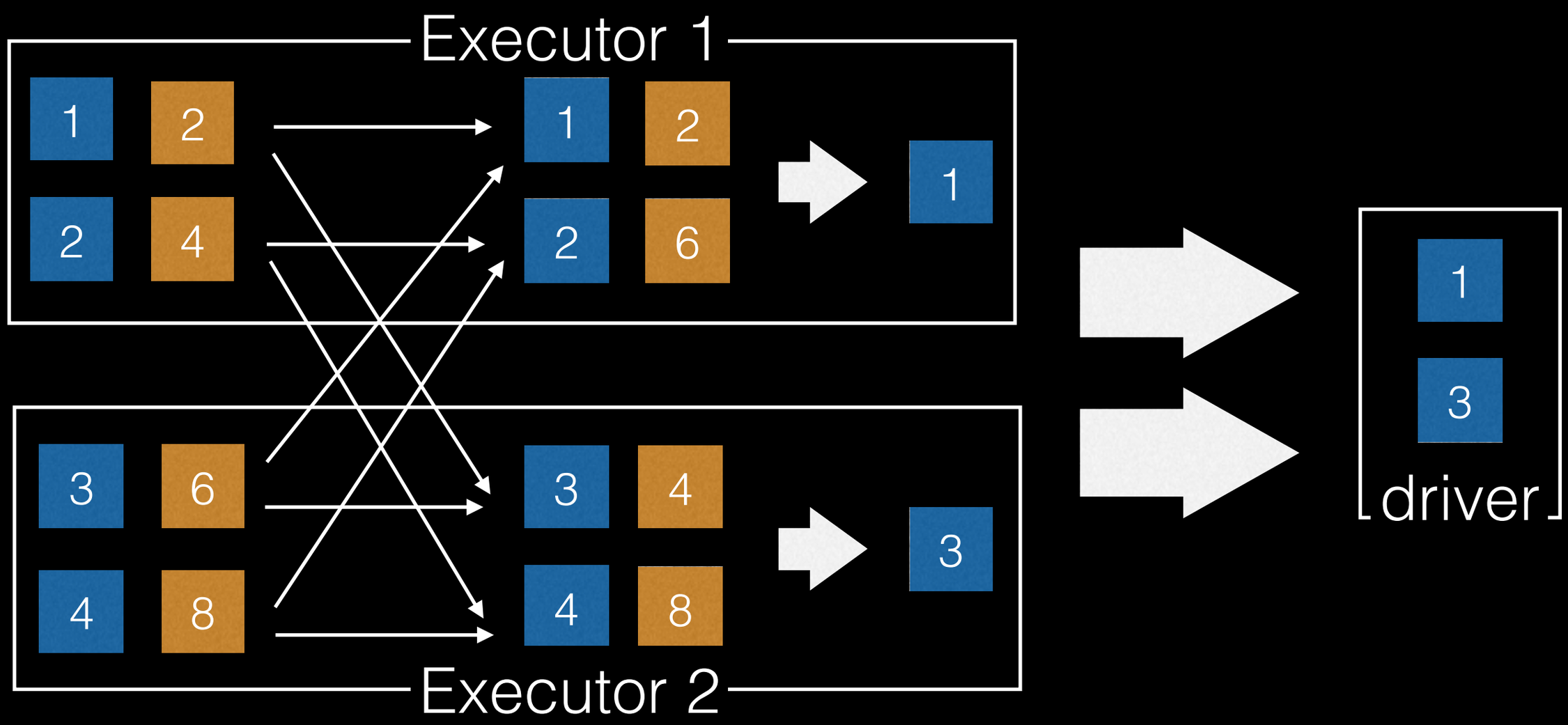


- Requires shuffle operation

subtract

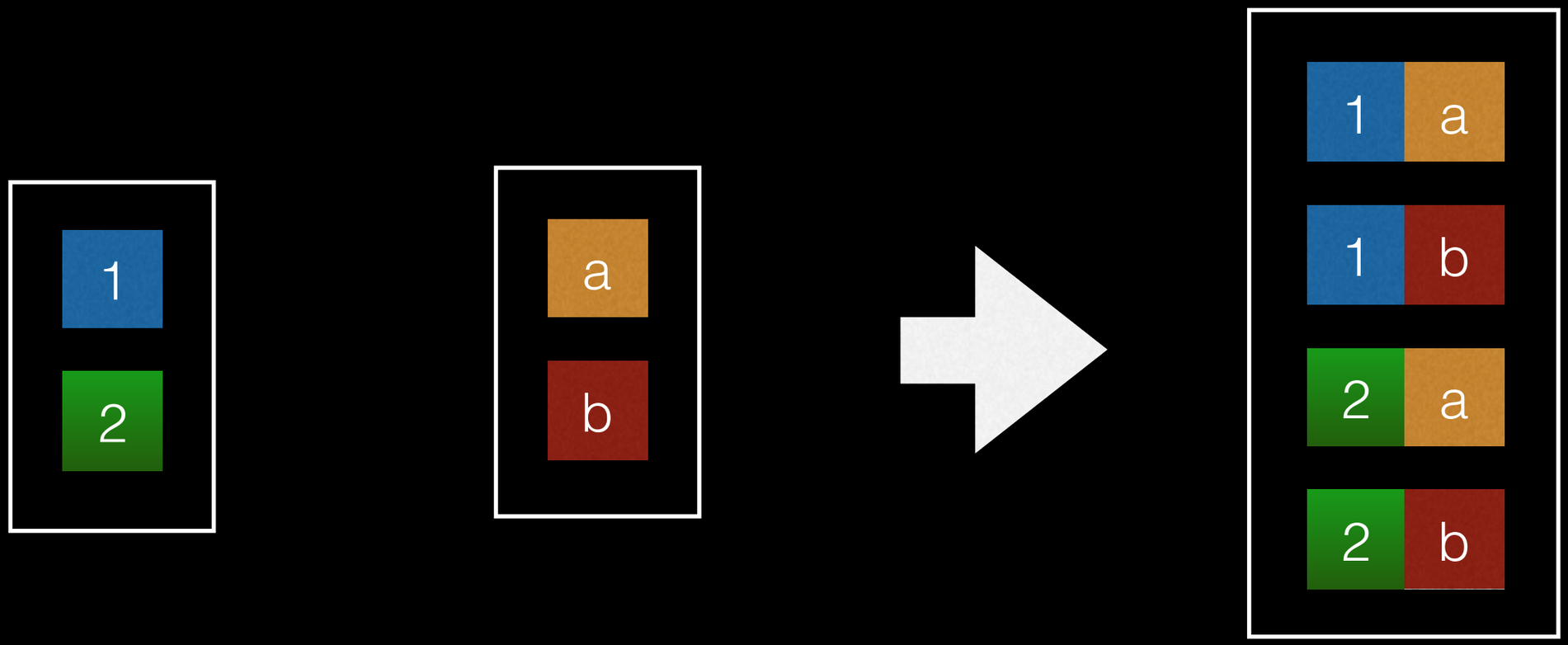


subtract



- Requires shuffle operation

cartesian



Lazy Evaluation

- transformations on RDDs are lazily evaluated
- calling `map` has no effect on the executors
- only calling `collect` (or any action) forces evaluation and returns the value to driver

Lazy Evaluation

```
val words =
  sc.parallelize(Seq("hello", "hi", "merhaba", "selam"))

val capitalWords = words.map(_.toUpperCase)
```

Now open SparkUI, you won't find new jobs

```
capitalWords.collect
```

Now open SparkUI, you should find the job now

Actions

- Operations that returns value to driver or write it to external storage.
- collect
- collectByValue
- take
- first
- top
- count
- reduce
- fold
- aggregate
- saveAsTextFile

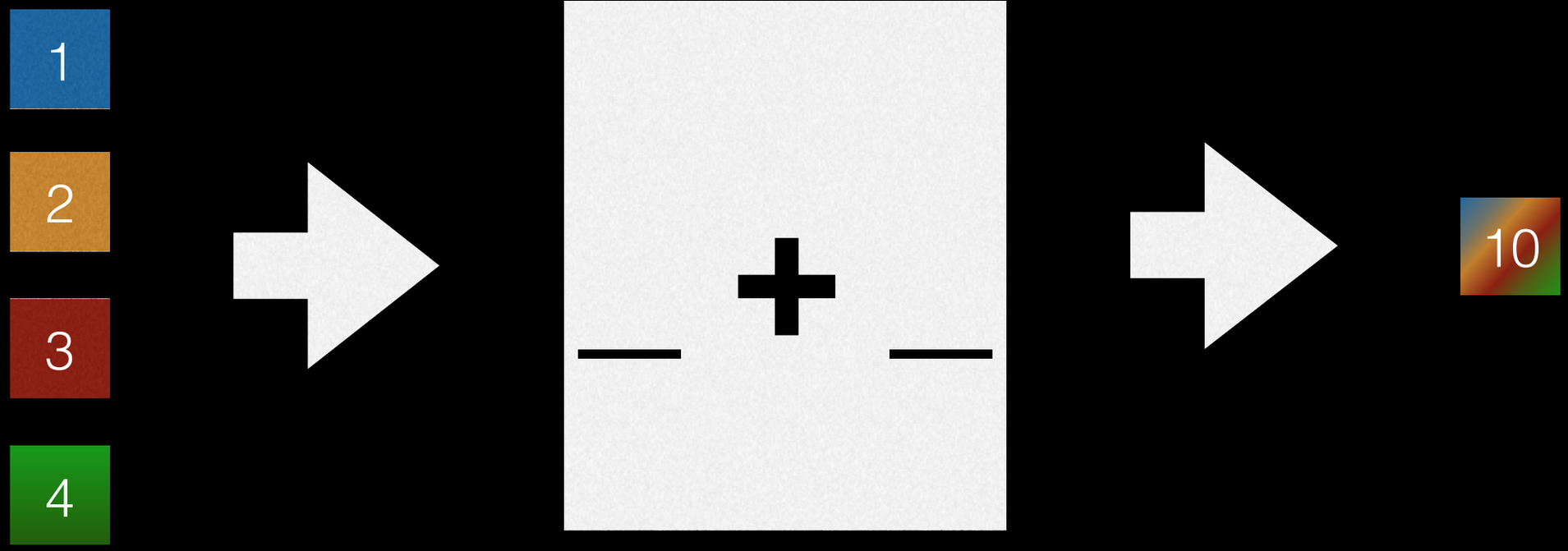
collect

- returns all elements in the RDD back to driver
- Note that RDD may be very large, it may not fit in driver's RAM

take

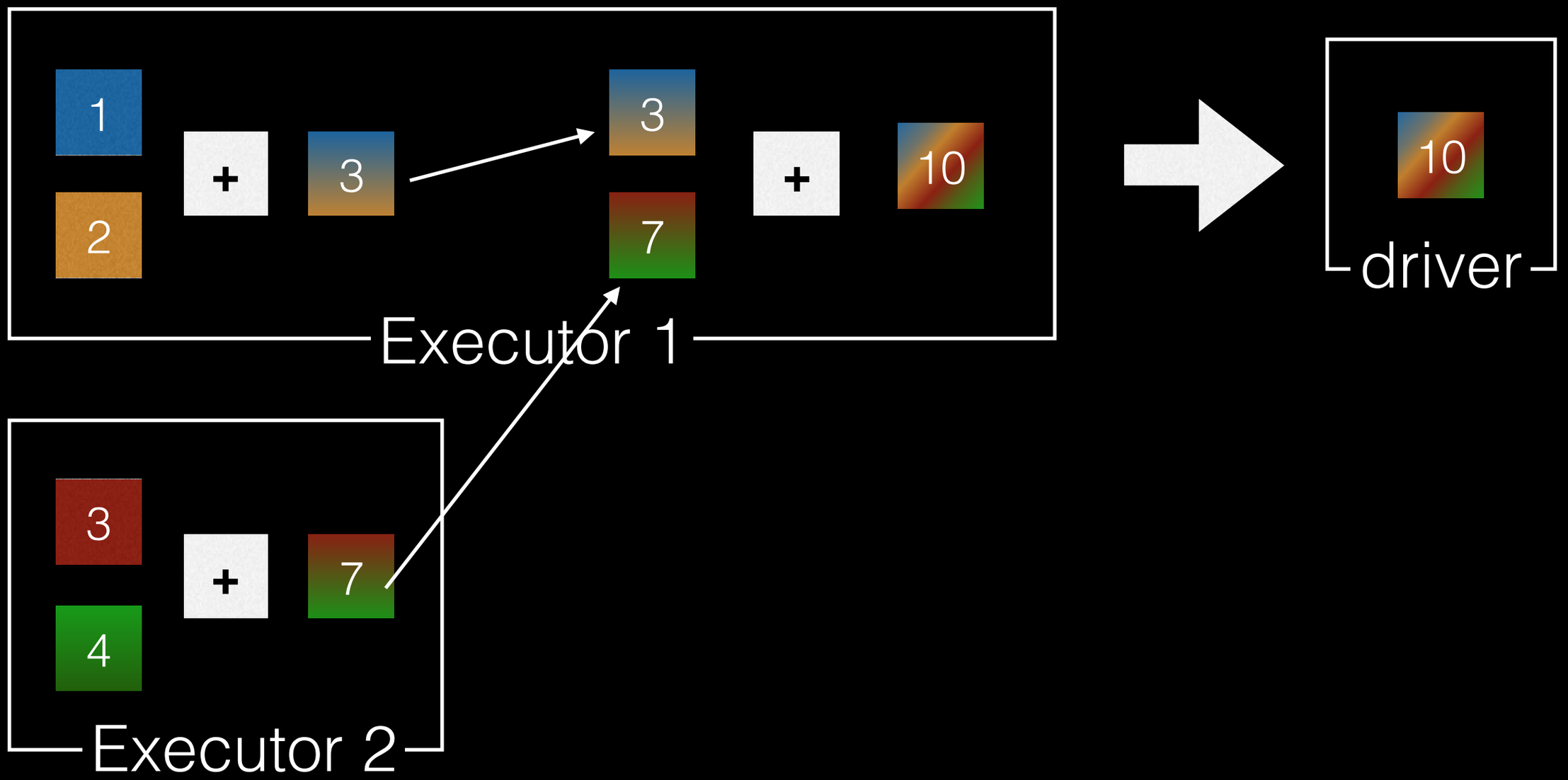
- like collect, but allows you to specify number of elements to return, and skip the rest
- return the first `n` elements to driver

reduce

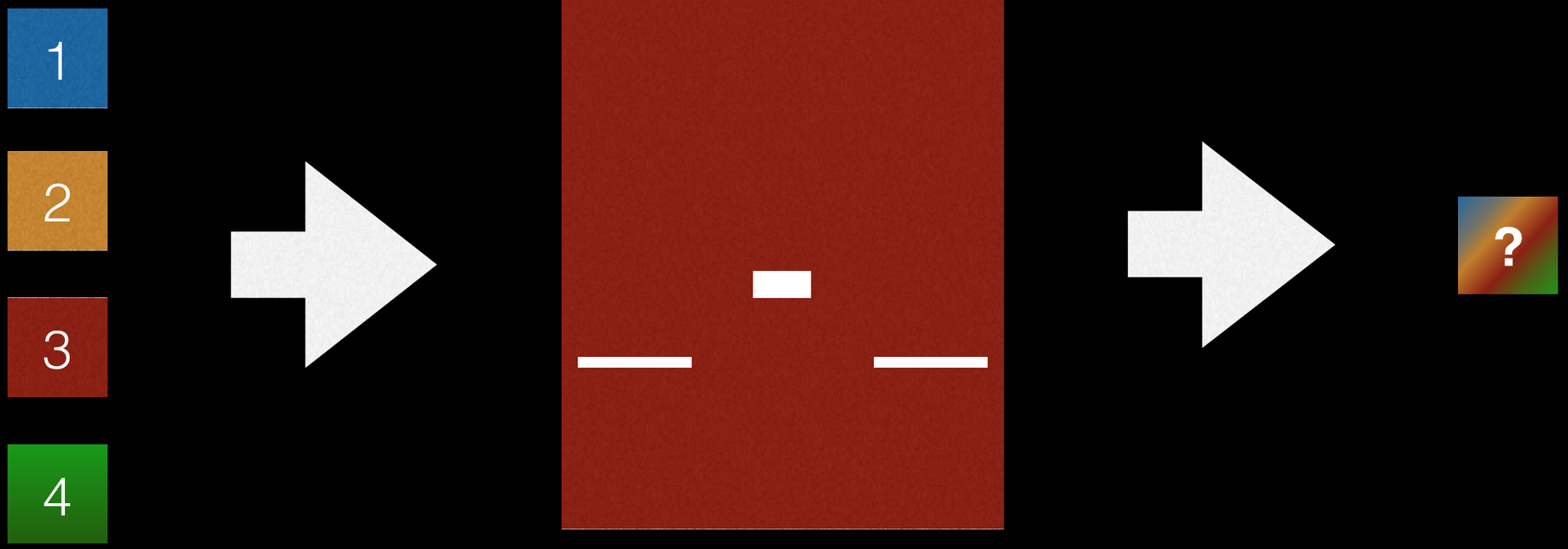


- Output is a single value
- Functions takes two elements and returns one
- Return must have same type as RDD’s elements

reduce

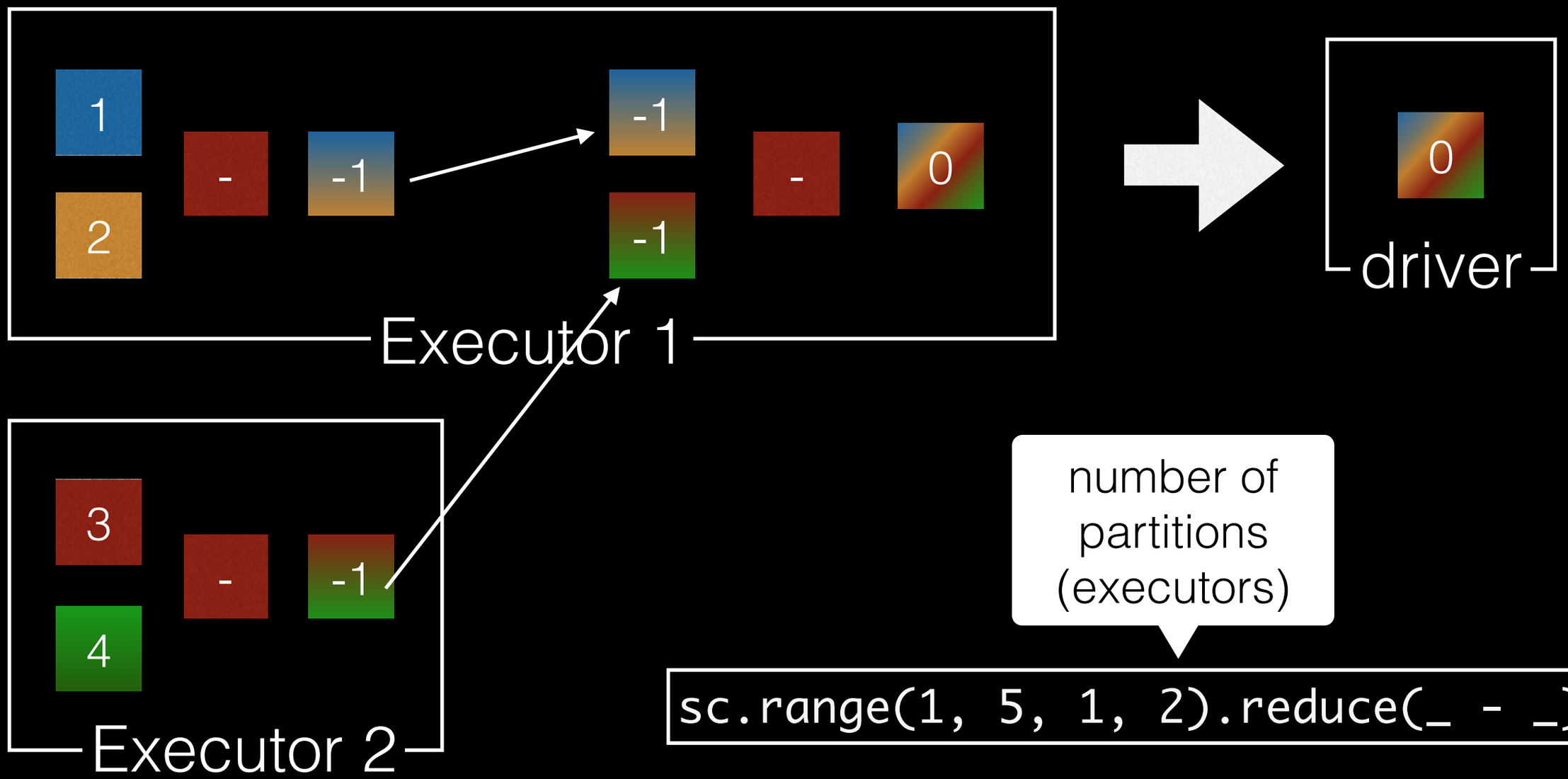


reduce

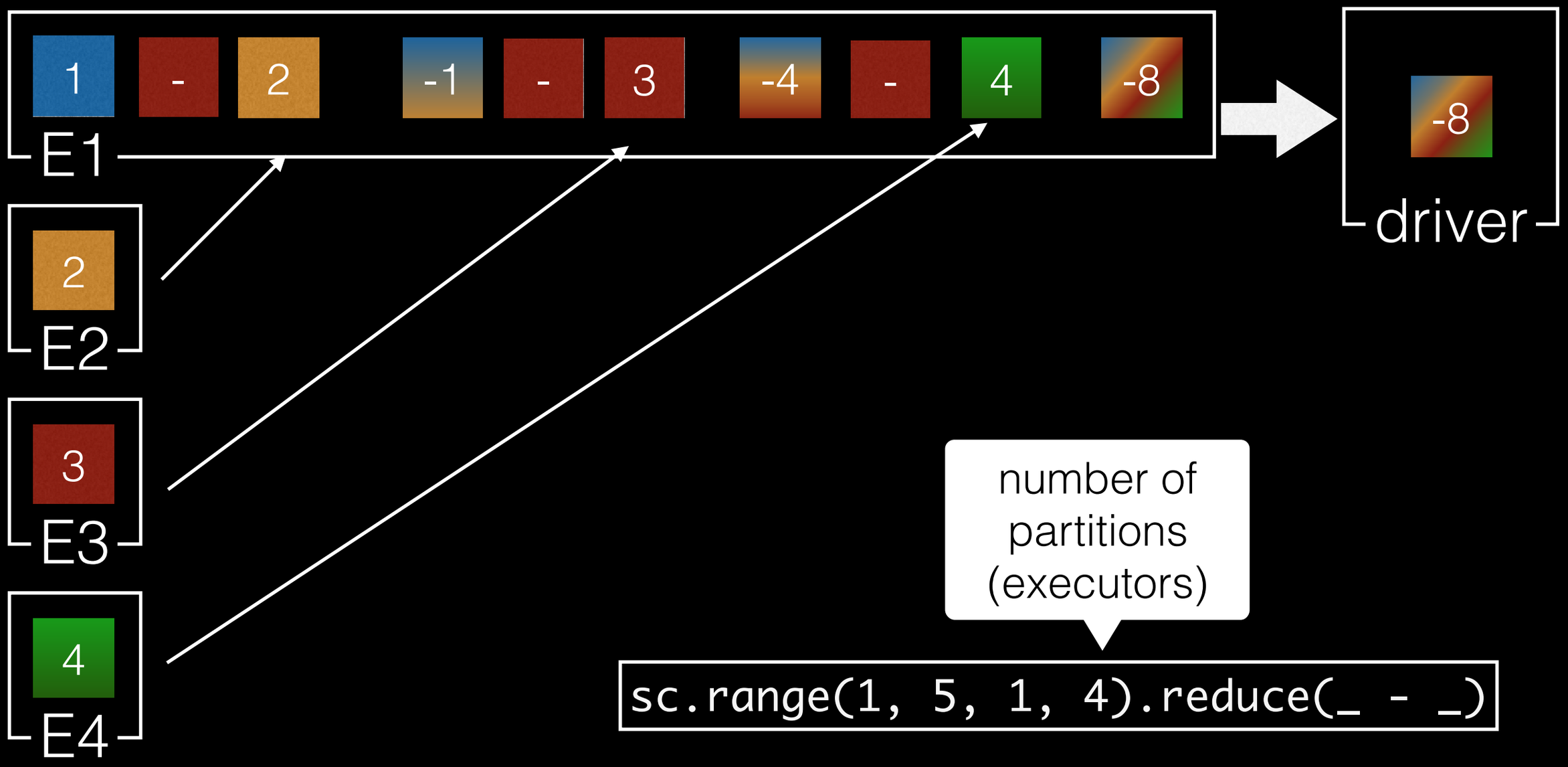


- Function must be **associative:** 1 + 2 == 2 + 1
- 1 - 2 != 2 -1
  (Not associative, problematic with reduce)

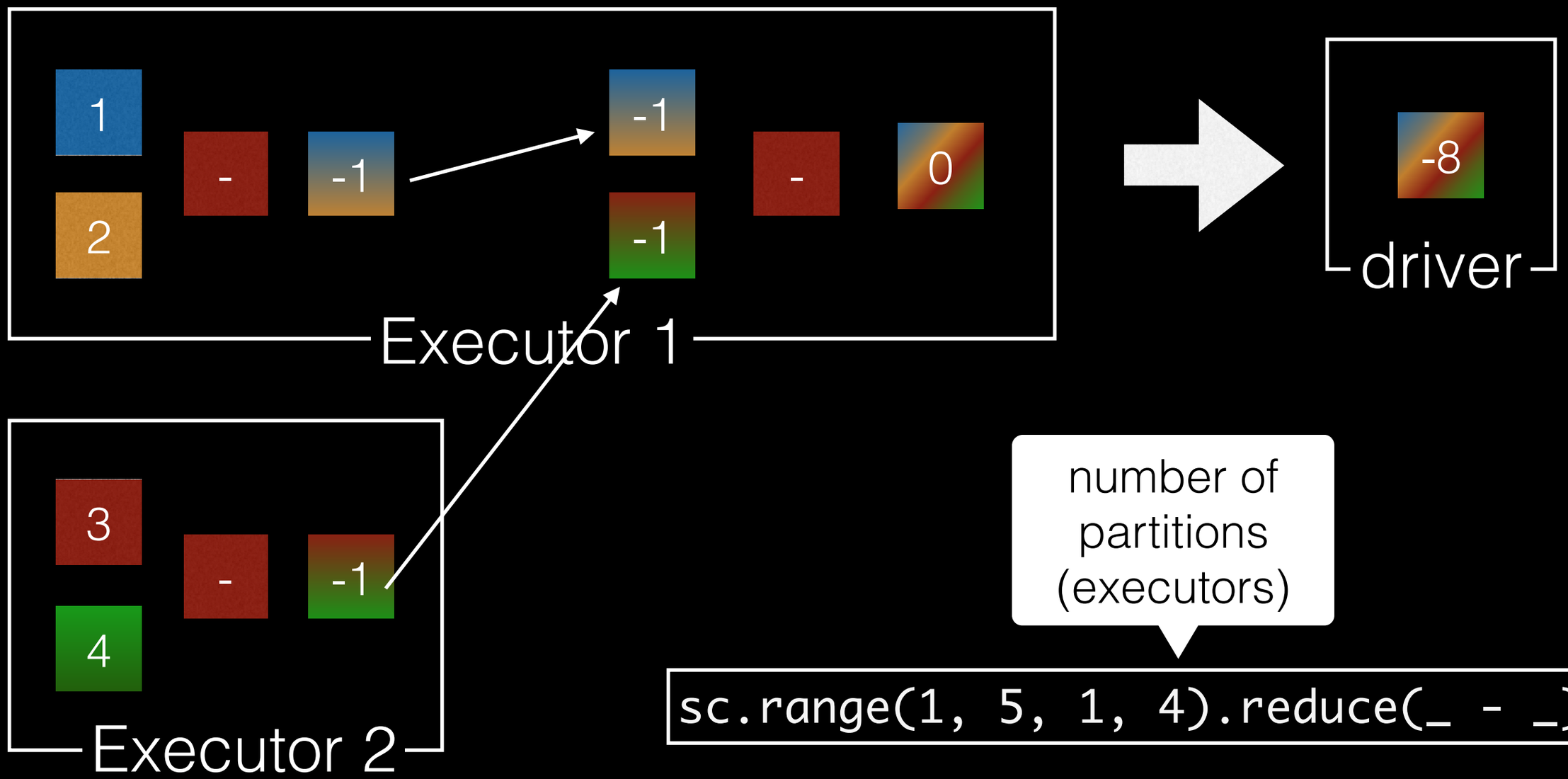
reduce


1
2
-
-1
-1
-1
-
0
Executor 1
3
4
-
-1
Executor 2
0
driver
number of partitions (executors)
sc.range(1, 5, 1, 2).reduce(_ - _)

reduce


1
-
2
-1
-
3
-4
-
4
-8
E1
2
E2
3
E3
4
E4
-8
driver
number of partitions (executors)
sc.range(1, 5, 1, 4).reduce(_ - _)

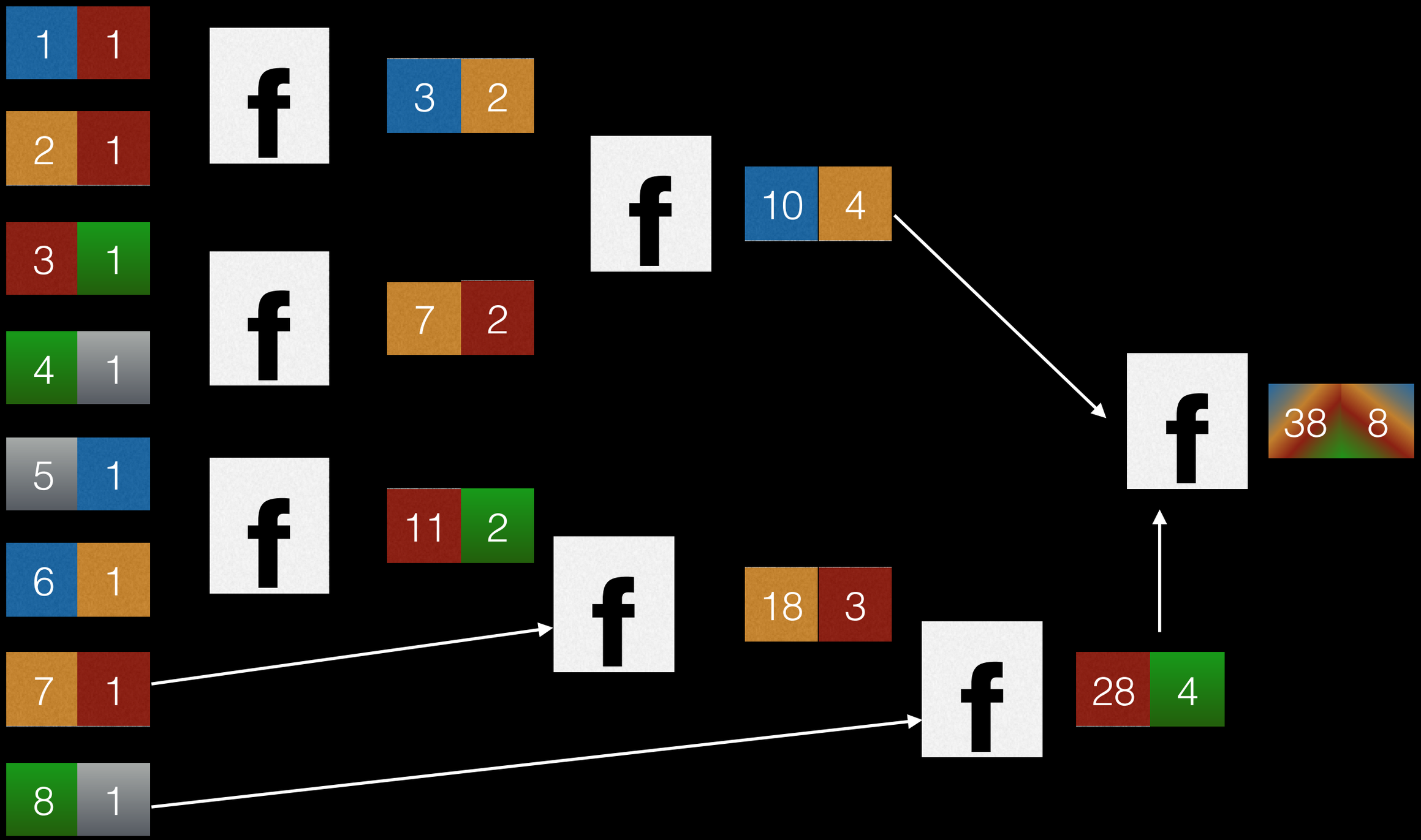
reduce


1
2
-
-1
-1
-1
-
0
Executor 1
3
4
-
-1
Executor 2
-8
driver
number of partitions (executors)
sc.range(1, 5, 1, 4).reduce(_ - _)

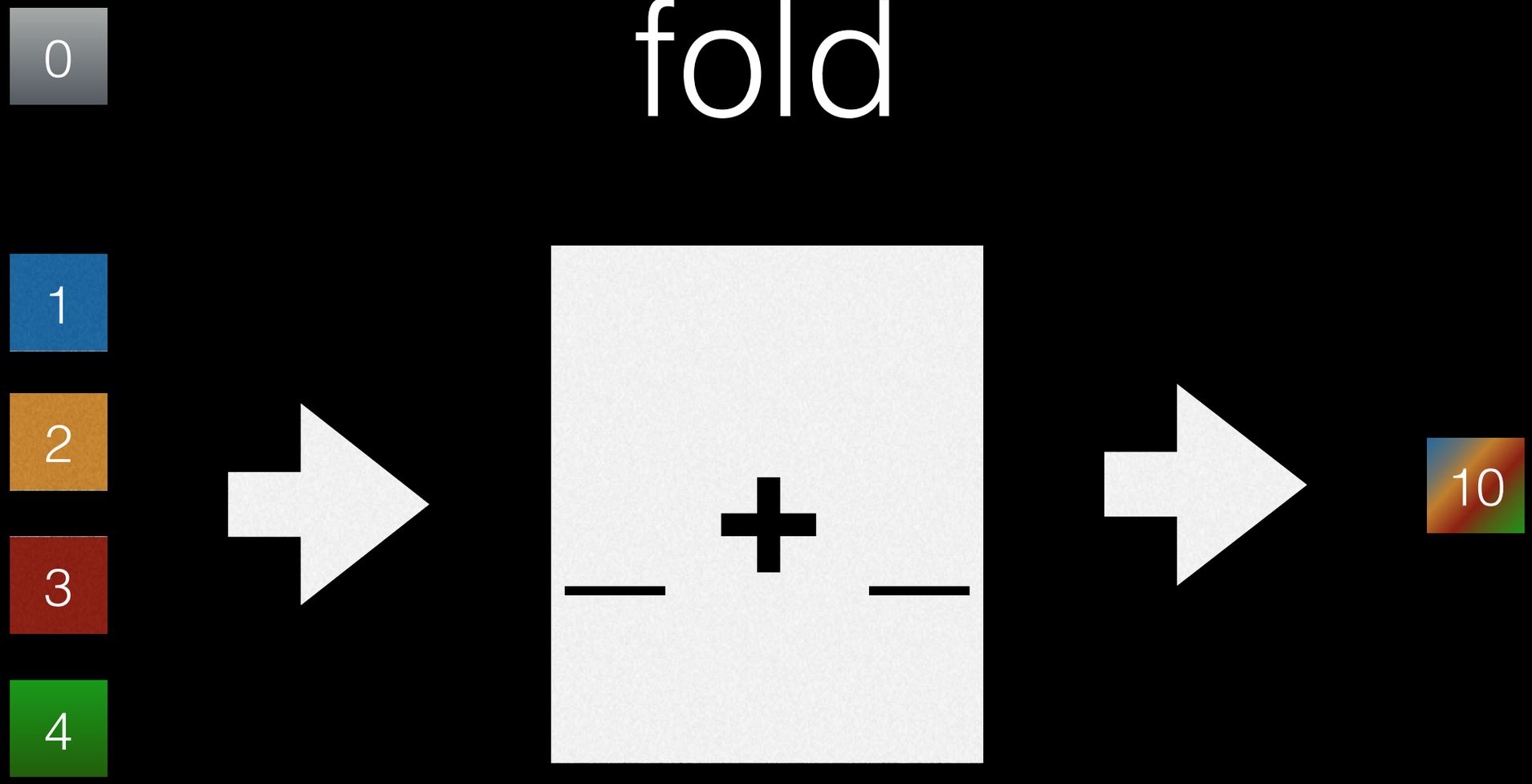
Calculate running average

- sum / count
- Can we calculate both on one go?

reduce

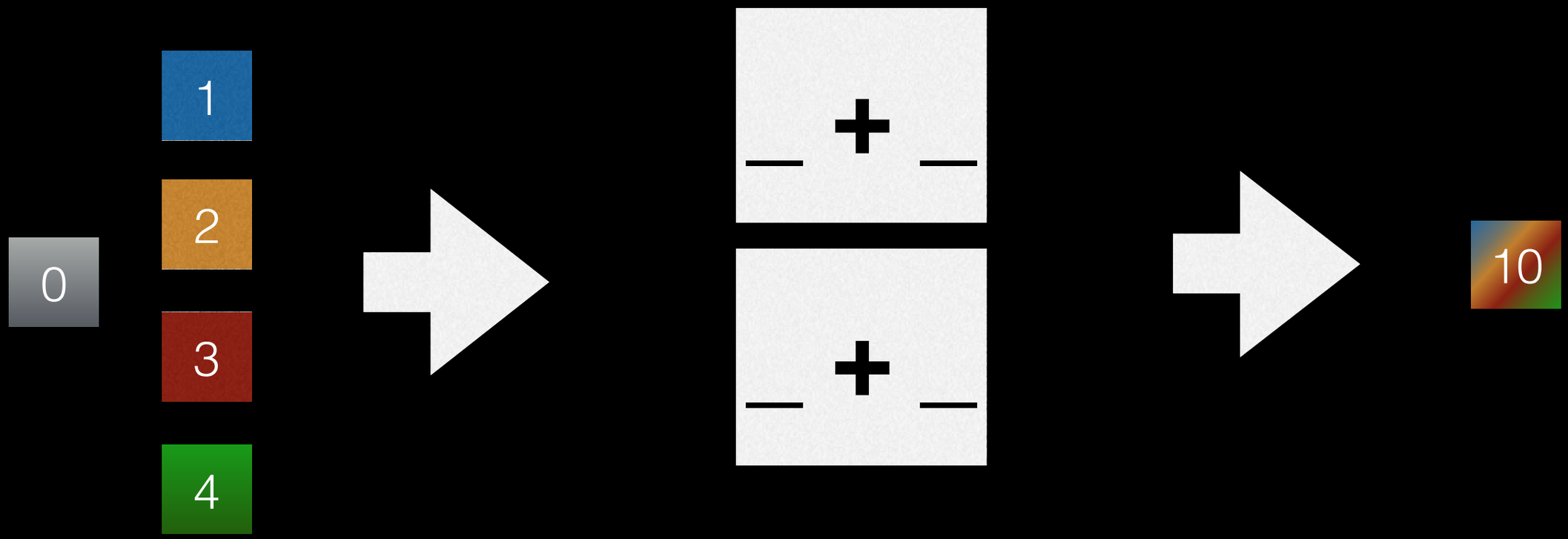


fold



- Same as reduce
- Allows specifying initial value
- useful if the RDD may be empty (or has 1 element)

aggregate

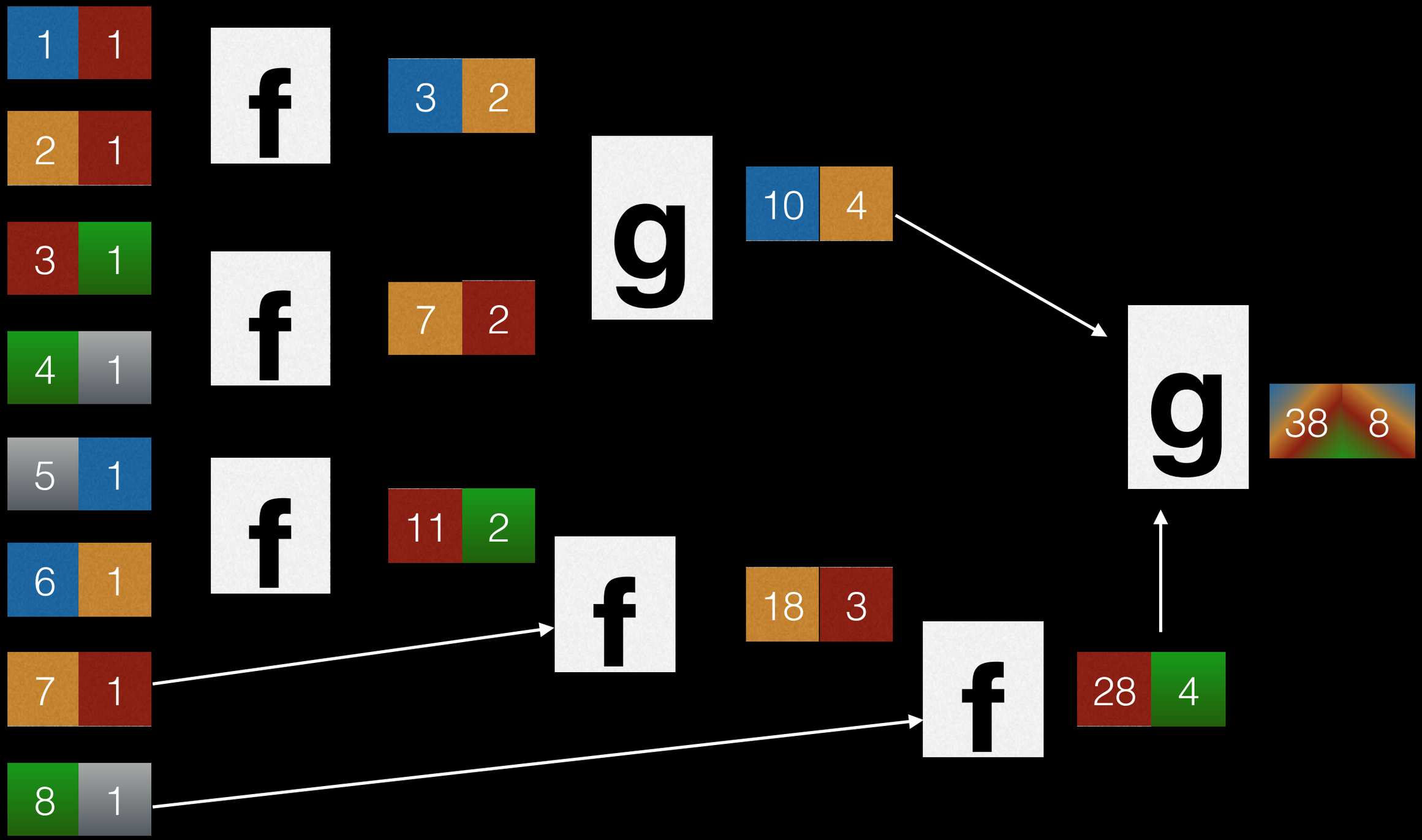


- Same as fold and reduce
- Allows specifying initial value
- Allow custom merging between executors

Calculate running average

- Can we improve reduce version?

aggregate



count

- returns all count of elements in the RDD back to driver

saveAsTextFile

- saves RDD into text file

Special RDDs

- Spark has special RDDs that provides more functions
- Numeric RDD has mean() and variance()
  - In Scala: compiler won't allow calling those functions unless RDD has numbers
  - In Python: calling these methods will fail at runtime
  - In Java: You can't call these function. Unless you change the RDD type

Change RDD type

```
JavaRDD<Integer> rdd =
  sc.parallelize(Arrays.asList(1, 2, 3, 4, 5));

JavaDoubleRDD doubles =
  rdd.mapToDouble(x -> 1.0 * x)

System.out.println(doubles.mean());
```