Spark Programming

View of Sierra Nevada Range from White Mountain

Mahidhar Tatineni
User Services, SDSC

Costa Rica Big Data School
December 6, 2017

Hands On

- Clone the repository:

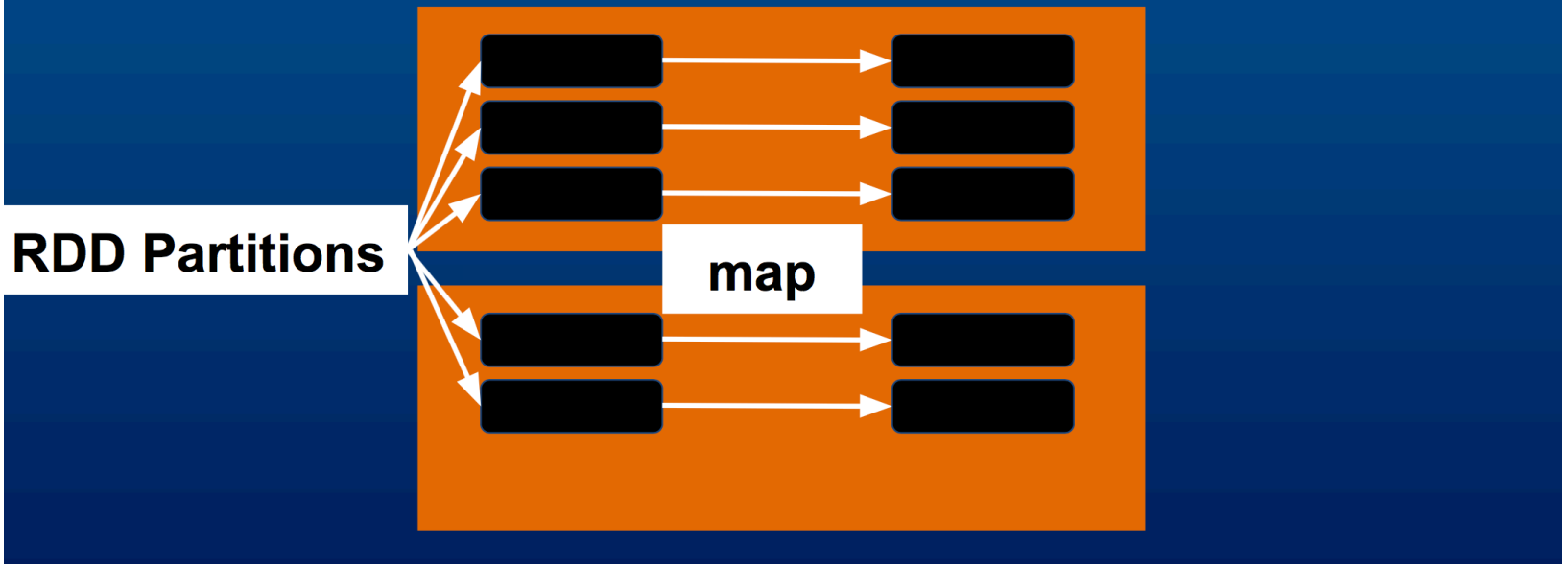
https://github.com/mahidhar/Spark_handson

- Command:

git clone https://github.com/mahidhar/Spark_handson

map

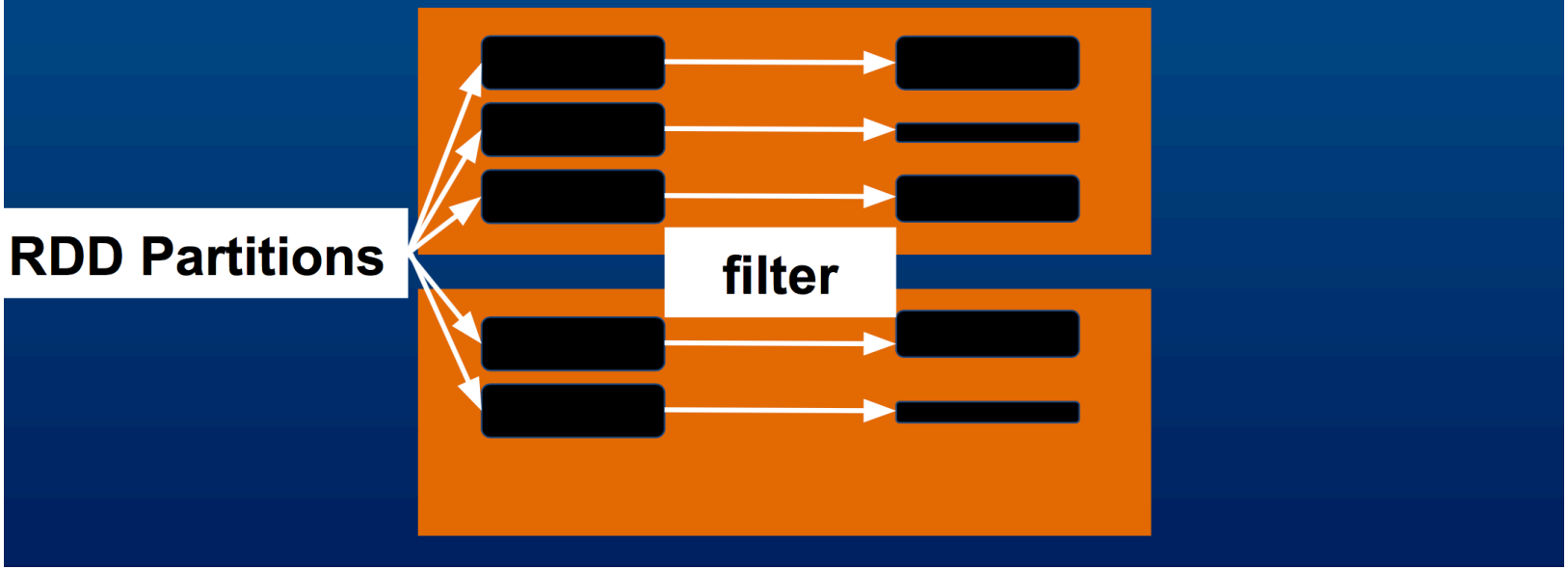
map : apply function to each element of RDD



Thanks to Andrea Zonca, SDSC for Housing data Spark Examples and following Slides

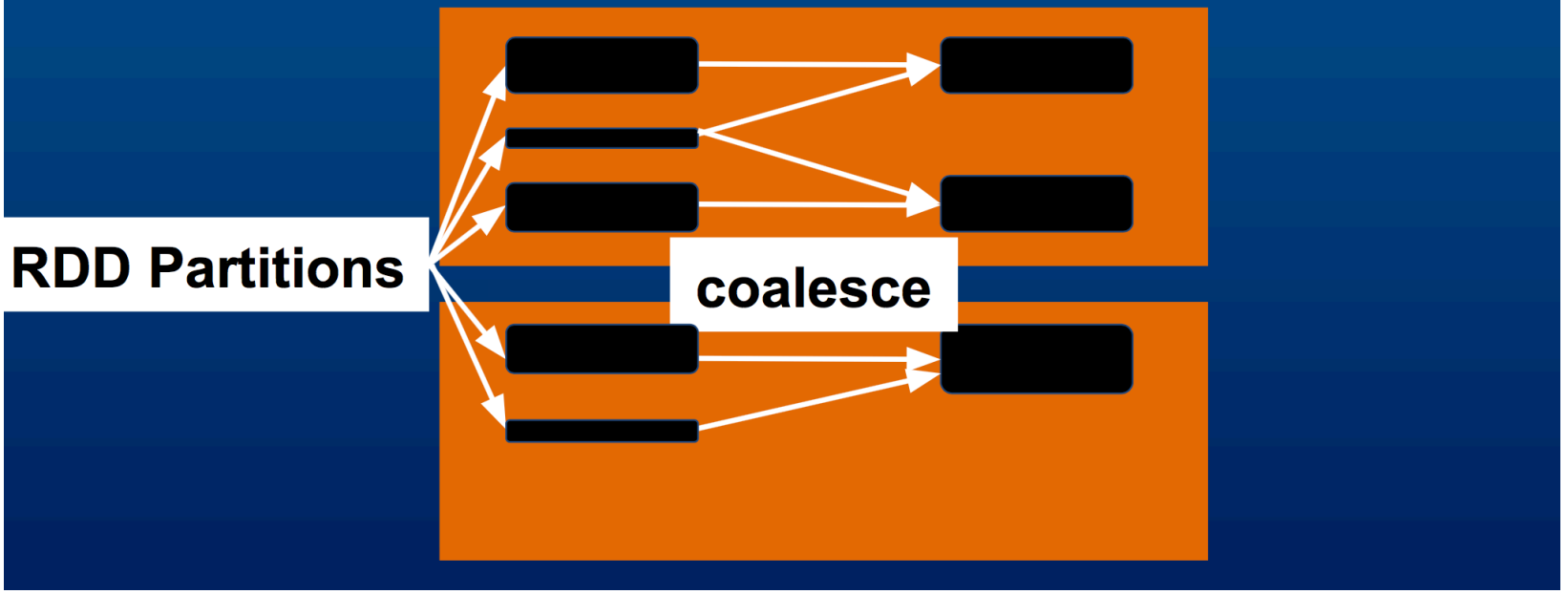
filter

filter : keep only elements where func is true



coalesce

coalesce : reduce the number of partitions



Coalesce

- sc.parallelize(range(10), 4).glom().collect()

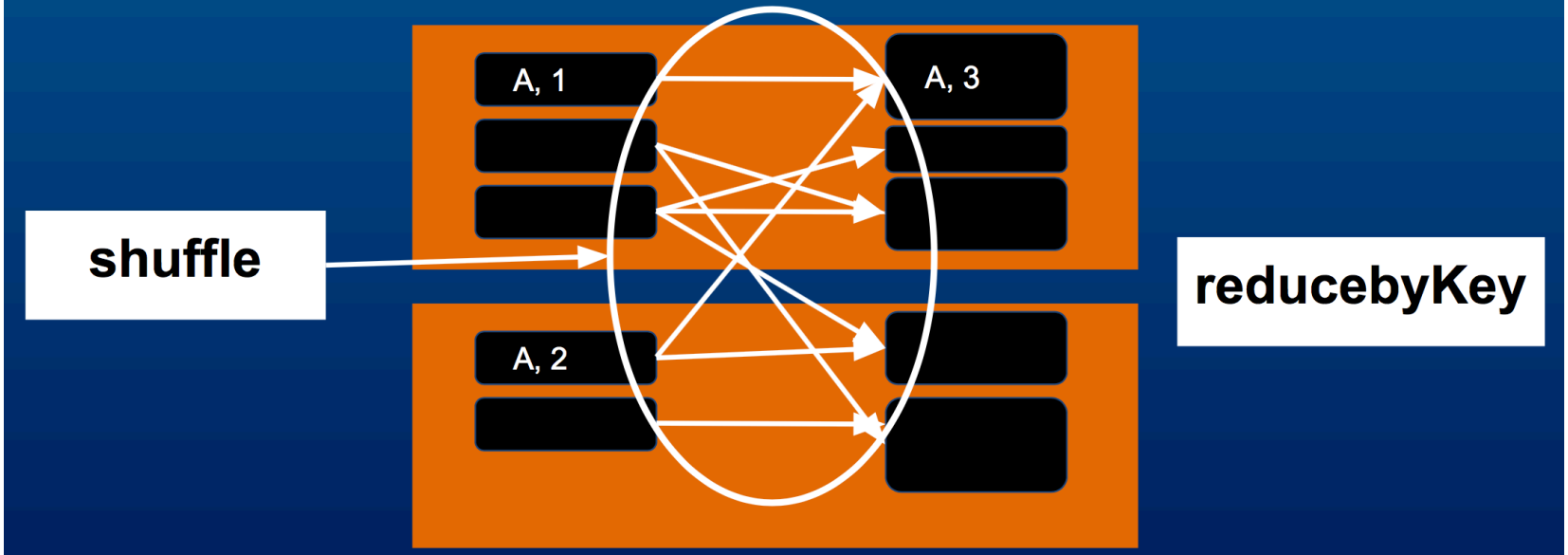
Out[]: [[0, 1], [2, 3], [4, 5], [6, 7, 8, 9]]

- sc.parallelize(range(10),4).coalesce(2).glom().collect()

Out[]: [[0, 1, 2, 3], [4, 5, 6, 7, 8, 9]]

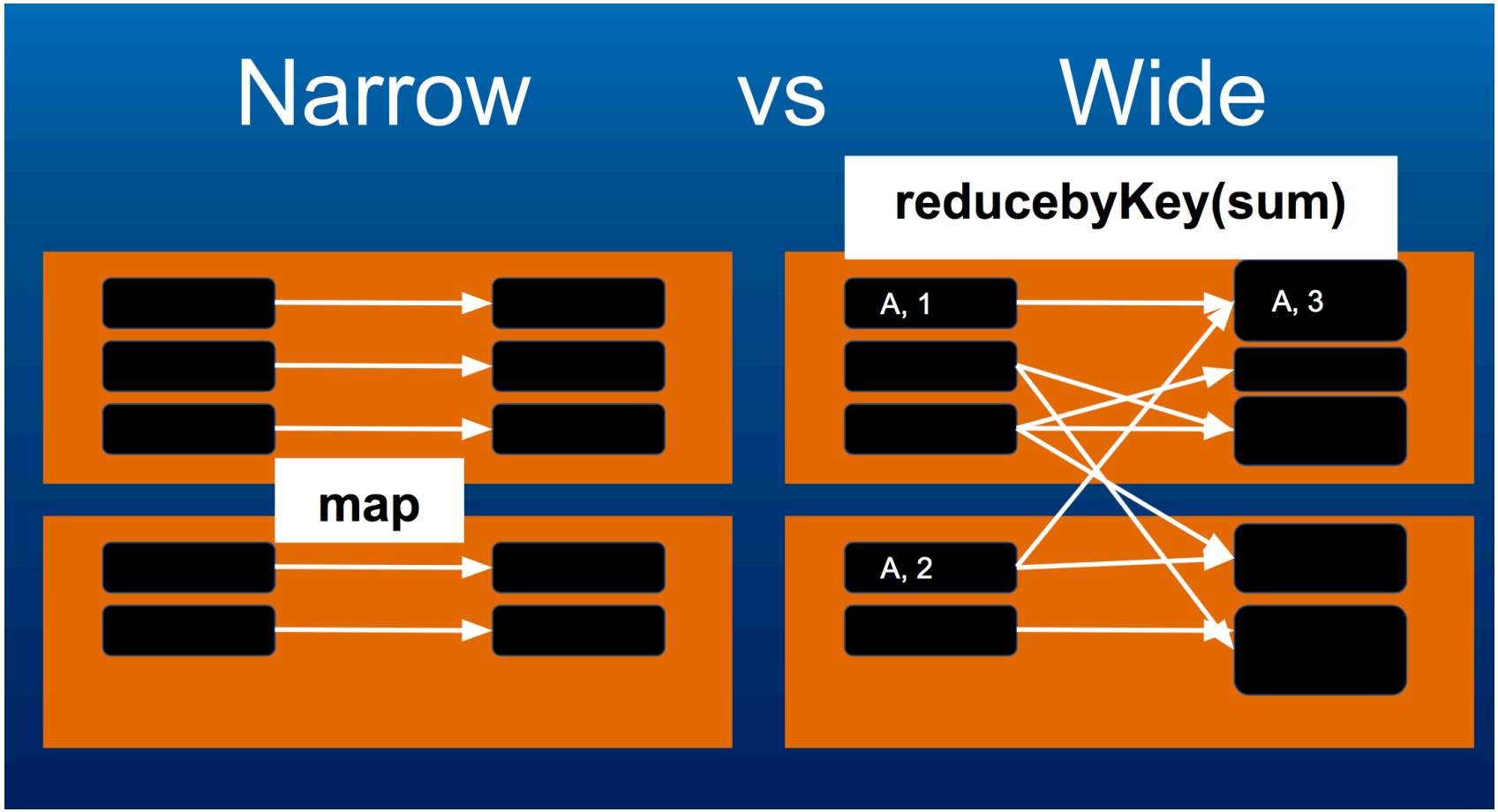
reduceByKey(func)

(K, V) pairs => (K, reduce V with func)



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


Narrow
vs
Wide
reducebyKey(sum)
map
A, 1
A, 2
A, 3

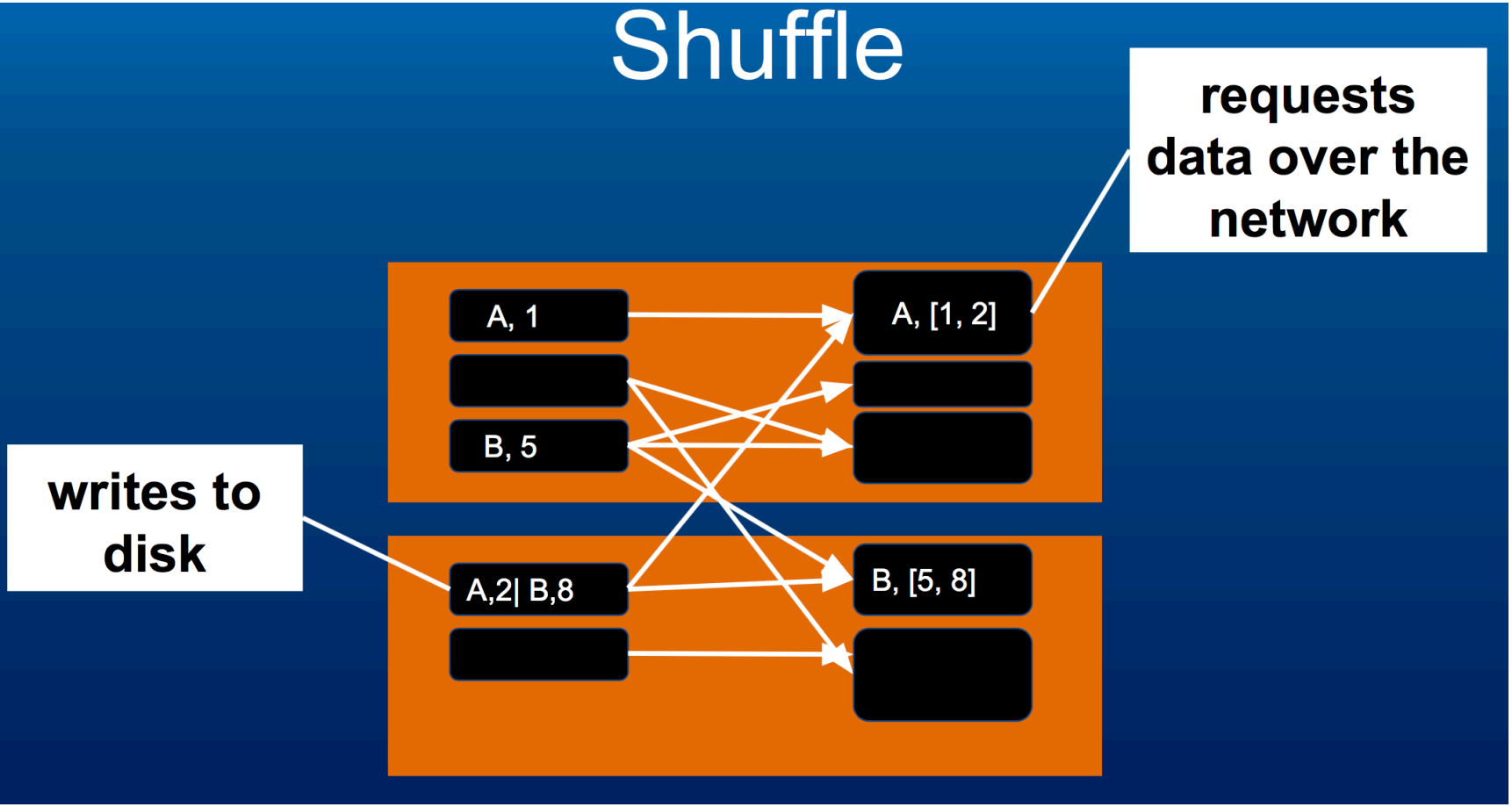
Shuffle

- Global redistribution of data
- High impact on performance

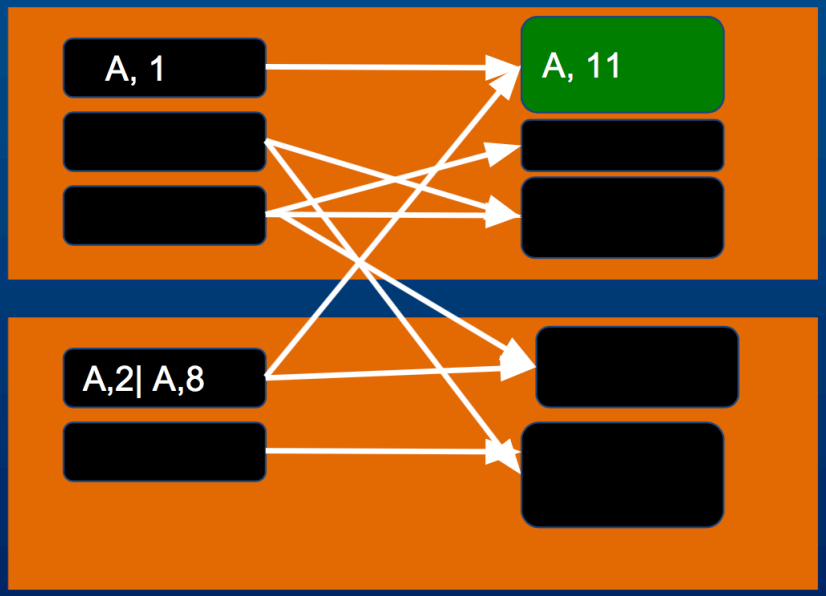
- Know shuffle, avoid it
- Which operations cause it?
- Is it necessary?

- **Really need groupByKey?**
- **groupByKey: (K, V) pairs => (K, iterable of all V)**
- **if you plan to call reduce later in the pipeline, use reduceByKey instead.**

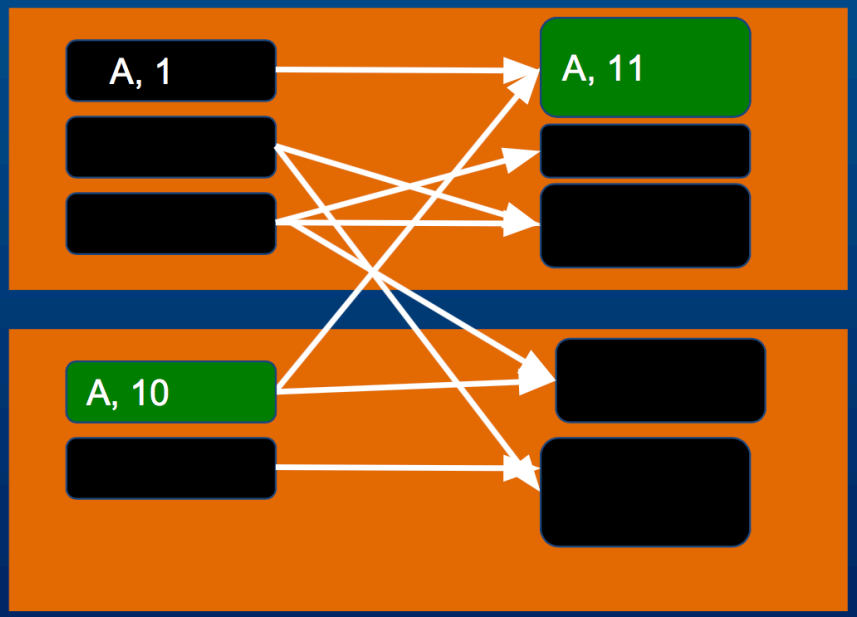
Shuffle


requests data over the network
writes to disk
A, 1
B, 5
A,2| B,8
A, [1, 2]
B, [5, 8]

groupByKey + reduce



reduceByKey



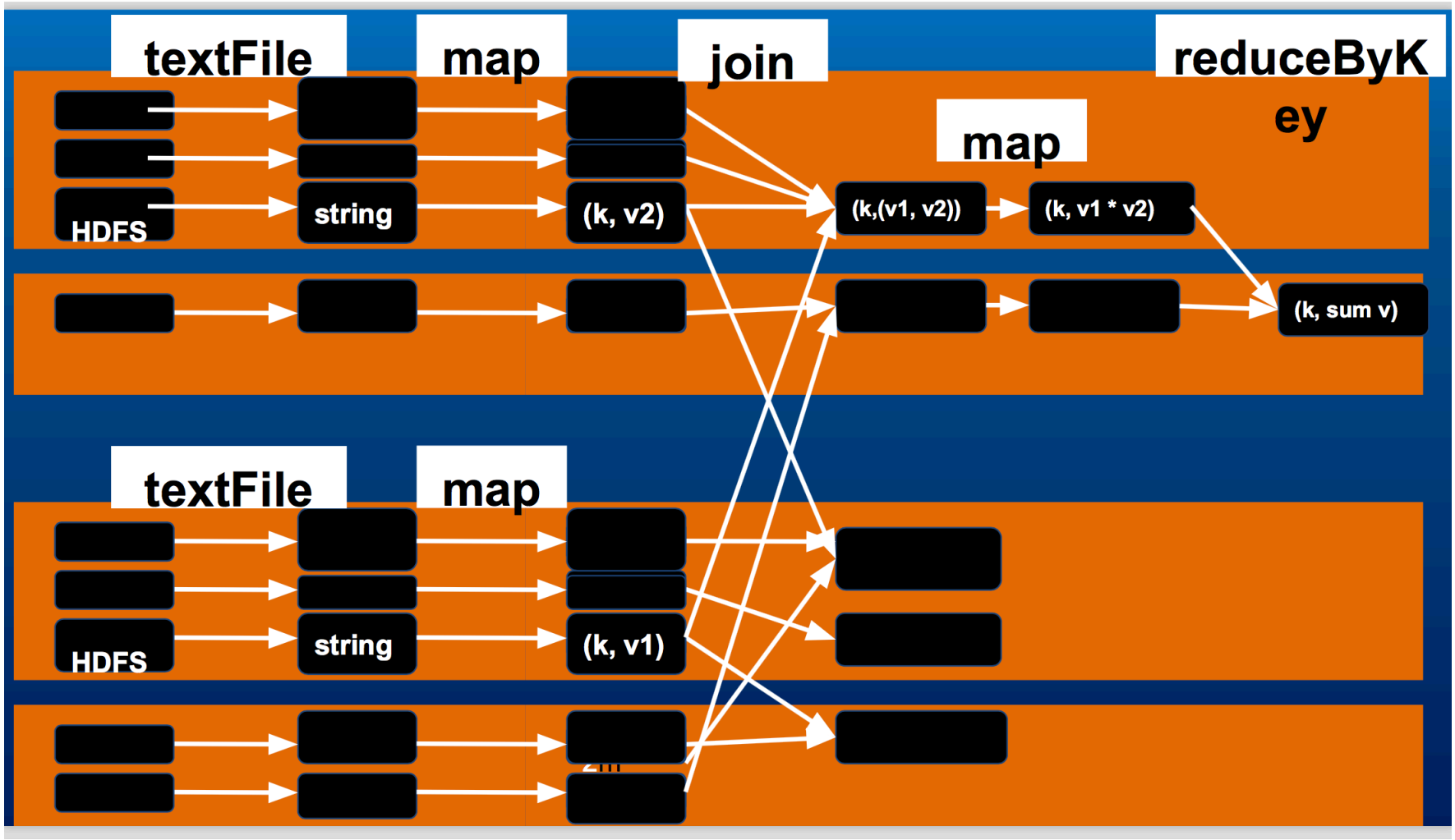
Cached RDDs

- Generally recommended after data cleaning
- Reusing cached data: 10x speedup
- Great for iterative algorithms
- If RDD too large, will only be partially cached in memory

DAG in Spark

- nodes are RDDs
- arrows are Transformations

House Price JOIN



Thank You!

Questions