Summarizing Data Along Dimensions

Swetha Kolalapudi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Representing records as Pair RDDs

Summarizing Pair RDDs using reduceByKey and combineByKey

Merging data from separate RDDs

Traffic at the Dodgers Stadium

Summarizing Along Dimensions

Date	Game Day?	Opponent	Win/Loss	Traffic
5/04	Yes	San Antonio	W	1000
5/20	No	-		500
6/30	Yes	Colorado	L	1200
7/31	Yes	Ohio	W	1300

Metrics

Summarizing Along Dimensions

Date	Game Day?	Opponent	Win/Loss	Traffic
5/04	Yes	San Antonio	W	1000
5/20	No	-		500
6/30	Yes	Colorado	L	1200
7/31	Yes	Ohio	W	1300

Dimensions

Modeling Traffic Patterns

How many cars travel this road on a given day?

Which were the days that saw the highest traffic?

Were there Dodgers games on the days with high traffic?

What's the average traffic like on a game day vs a non-game day?

Two Types of RDDs

Each element is a single object

Each element is a Key/Value pair

Pair

Each record is a tuple with 2 objects

(Airline, Delay)

(City, Sales)

(Word, Count)

Pair

To create a Pair RDD, just make sure each record is a tuple

Pair

Summarize by keys

- reduceByKey
- combineByKey

Merge by keys

- join
- leftOuterJoin
- rightOuterJoin

reduceByKey

Combine records with the same key in a specified way

Sum

Maximum

Minimum

reduceByKey

Like reduce

A function with 2 arguments

reduceByKey

Unlike reduce

Only combines values with the same key

The operation is a transformation

```
reduceByKey(lambda x,y:x+y)
```

P1

JFK	2
JFK	14
PPG	5
PPG	10
JFK	0
PPG	4

P2

JFK	3
JFK	0
LAX	6
LAX	11
JFK	0
PPG	7

A Pair RDD with 2 partitions

```
reduceByKey(lambda x,y:x+y)
```

P1

JFK	2
JFK	14
PPG	5
PPG	10
JFK	0
PPG	4

P2

JFK	3
JFK	0
LAX	6
LAX	11
JFK	0
PPG	7

Reduce operation is performed on each partition

```
reduceByKey(lambda x,y:x+y)
```

P1

JFK	2
JFK	14
PPG	5
PPG	10
JFK	0
PPG	4

```
reduceByKey(lambda x,y:x+y)
```

P1

JFK	2
JFK	14
JFK	0

PPG	5
PPG	10
PPG	4

```
reduceByKey(lambda x,y:x+y)
```

P1

JFK	2
JFK	14
JFK	0

x

y

```
reduceByKey(lambda x,y:x+y)
```

P1

JFK	16
JFK	0

x + y

```
reduceByKey(lambda x,y:x+y)
```

P1

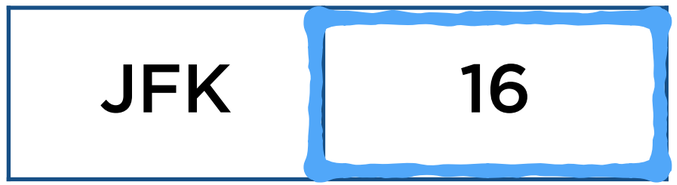
JFK	16
JFK	0

x

y

```
reduceByKey(lambda x,y:x+y)
```

P1



x + y

```
reduceByKey(lambda x,y:x+y)
```

P1

JFK	16

```
reduceByKey(lambda x,y:x+y)
```

P1

JFK	16
PPG	5

```
reduceByKey(lambda x,y:x+y)
```

P1

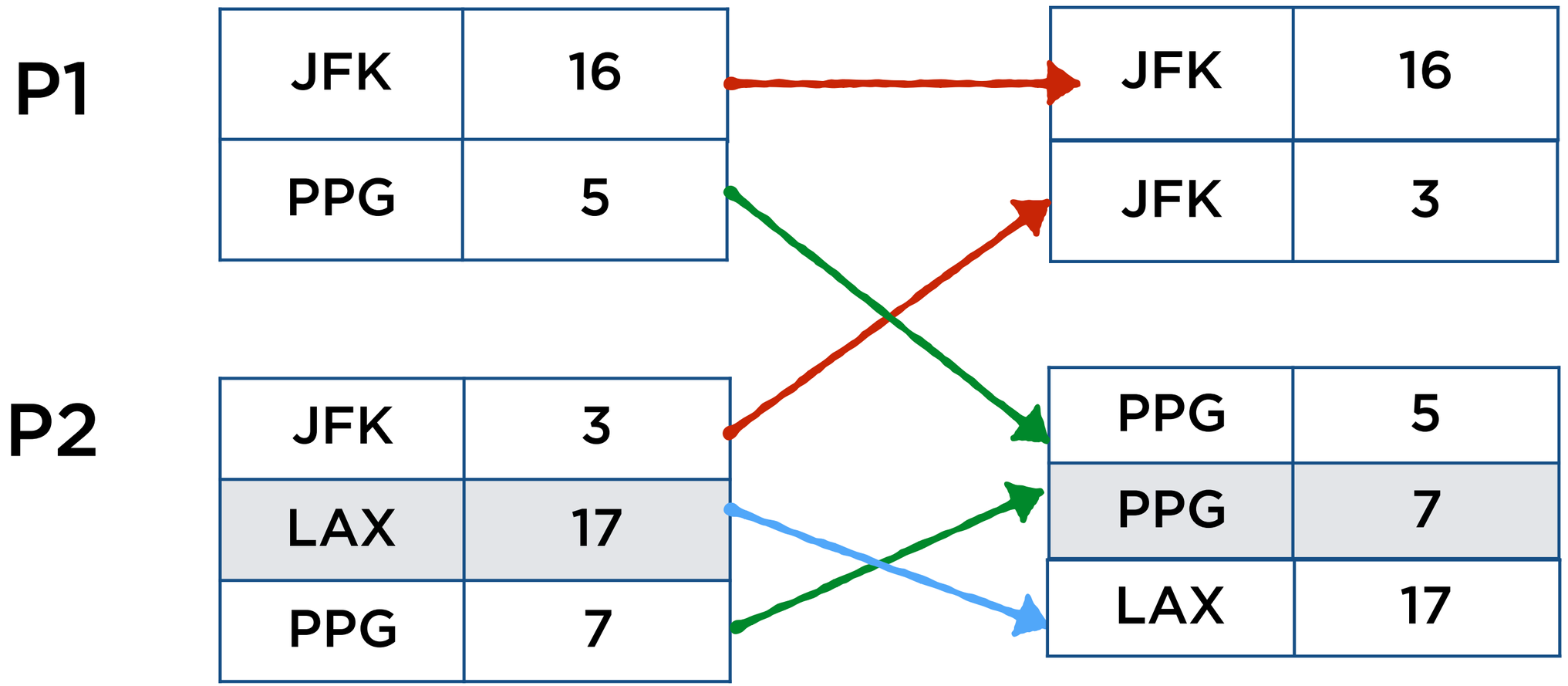
JFK	16
PPG	5

P2

JFK	3
LAX	17
PPG	7

These are shuffled so that all the values with same key are on a single partition

```
reduceByKey(lambda x,y:x+y)
```



```
reduceByKey(lambda x,y:x+y)
```

P1

JFK	16
JFK	3

P2

PPG	5
PPG	7
LAX	17

```
reduceByKey(lambda x,y:x+y)
```

P1

JFK	19

P2

PPG	12
LAX	17

combineByKey

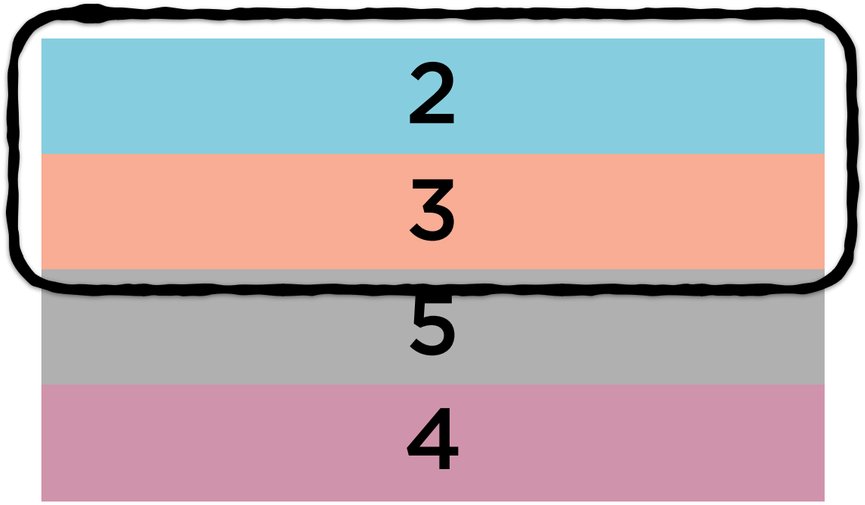
Combine records with the same key in a specified way

Very granular control over how the computation should happen

Computing Averages

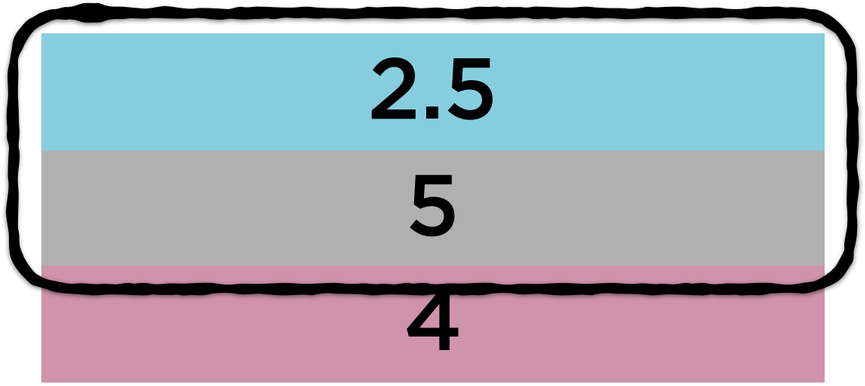
Cannot use a single reduceByKey operation

```
reduceByKey(lambda x,y: (x+y)/2)
```



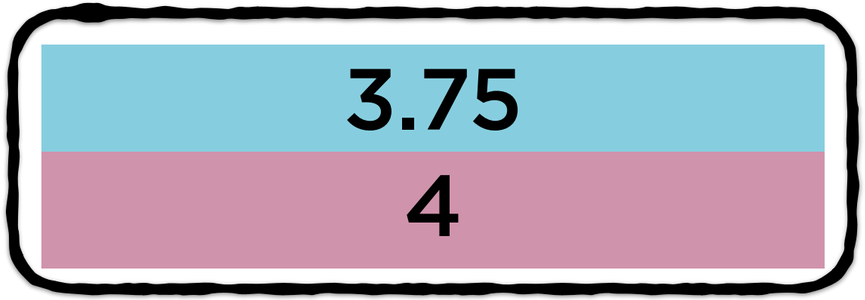
Average = 3.67

```
reduceByKey(lambda x,y: (x+y)/2)
```



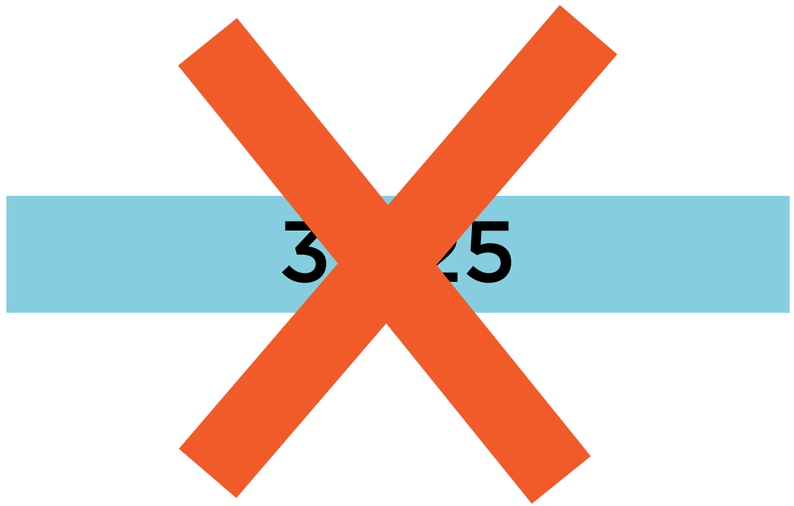
Average = 3.67

```
reduceByKey(lambda x,y: (x+y)/2)
```



Average = 3.67

```
reduceByKey(lambda x,y: (x+y)/2)
```



Average = 3.67

Computing Averages with reduceByKey

1. Compute sum per group
2. Compute counts per group
3. Join
4. Divide sum and count

Sum and Count in One Step

```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```

combineByKey requires 3 functions

createCombiner Function

```
combineByKey( (lambda value:(value,1)),
              (lambda acc, value: (acc[0]+value,acc[1]+1)),
              (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```

Initializes a value when a key is first seen within a partition

merge Function

```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```

Specifies how values with the same key should be combined

within a partition

mergeCombiners Function

```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```

Specifies how the results from each partition should be combined

```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```

P1

JFK	2
JFK	14
PPG	5
PPG	10
JFK	0
PPG	4

P2

JFK	3
JFK	0
LAX	6
LAX	11
JFK	0
PPG	7

A Pair RDD with 2 partitions

```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```

P1

JFK	2
JFK	14
PPG	5
PPG	10
JFK	0
PPG	4

This is first time the key “JFK” is seen

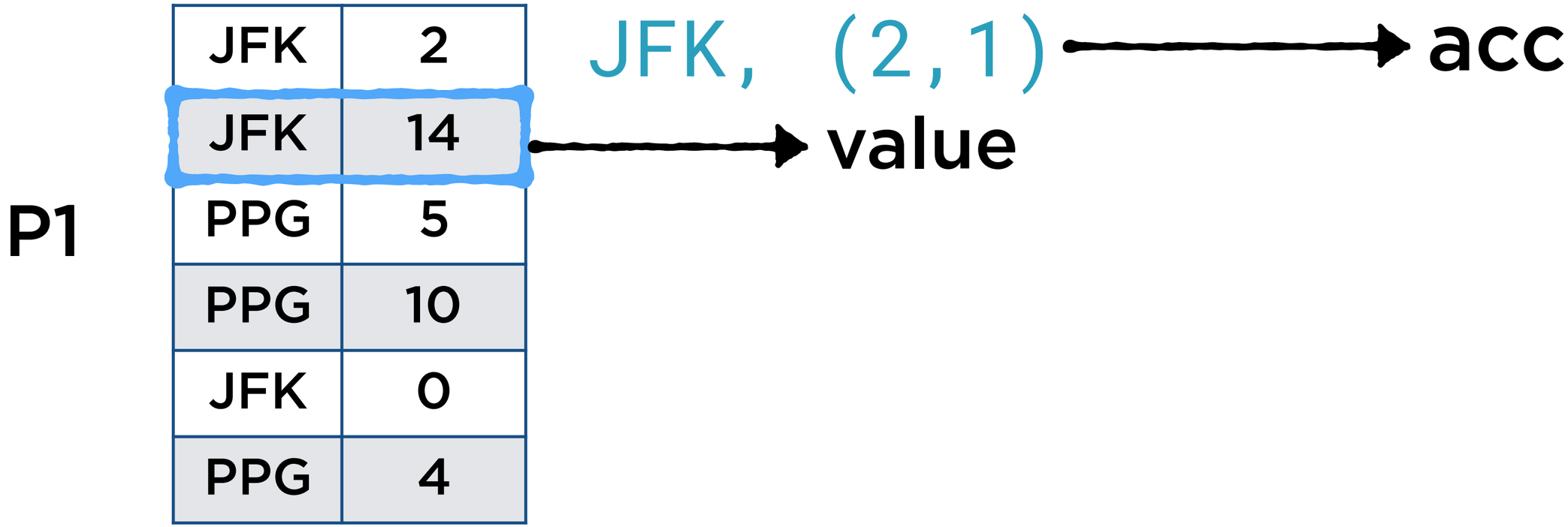
```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```

P1

JFK	2
JFK	14
PPG	5
PPG	10
JFK	0
PPG	4

JFK, (2,1)

```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
              (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```



```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```

P1

JFK	2
JFK	14
PPG	5
PPG	10
JFK	0
PPG	4

JFK, (16,2)

PPG, (5,1)

```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```

P1

JFK, (16,3)

PPG, (19,3)

```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```

P1

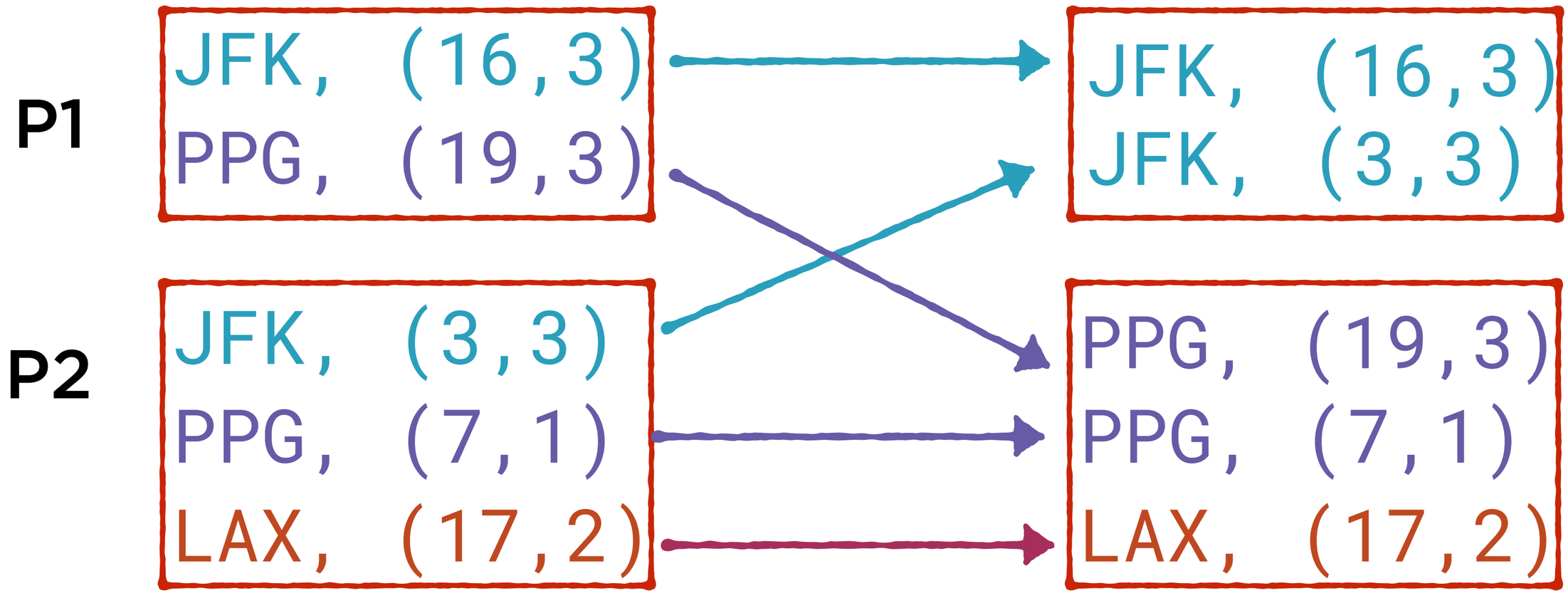
JFK, (16,3)
PPG, (19,3)

P2

JFK, (3,3)
PPG, (7,1)
LAX, (17,2)

The records are shuffled until all records with the same key are on the same partition

```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```



```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```

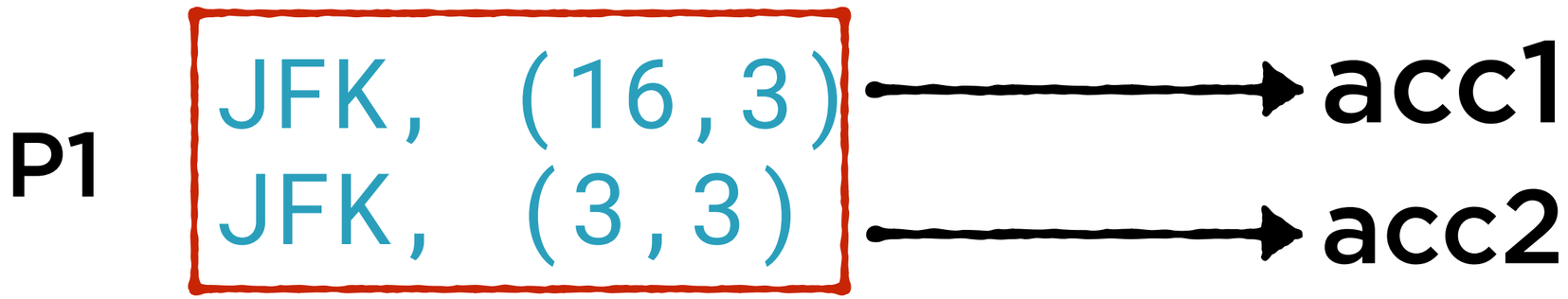
P1

JFK, (16,3)
JFK, (3,3)

P2

PPG, (19,3)
PPG, (7,1)
LAX, (17,2)

```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```



```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```

P1 JFK, (19,6)

```
combineByKey((lambda value:(value,1)),
             (lambda acc, value: (acc[0]+value,acc[1]+1)),
             (lambda acc1, acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))
```

P1 JFK, (19,6)

P2 PPG, (26,4)

LAX, (17,2)

Merging Pair RDDs

Merge 2 Pair RDDs

based on the keys

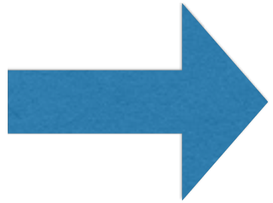
Merging Pair RDDs

Pair RDD1

```
BLR,3
MUM,1
DEL,2
```

Pair RDD2

```
BLR,"B"
MUM,"M"
DEL,"D"
```



```
BLR,[3,"B"]
MUM,[1,"M"]
DEL,[2,"D"]
```

Types of Joins

join	leftOuterJoin	rightOuterJoin

Similar to their counter parts in SQL

join

A join returns a new Pair RDD

Values whose keys match are grouped together

join

An inner join

Only keys that match from both RDDs are returned

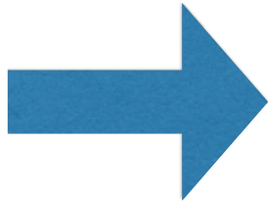
join = An Inner Join

Pair RDD1

```
BLR,3
MUM,1
DEL,2
```

Pair RDD2

```
BLR,"B"
MUM,"M"
```



```
BLR,[3,"B"]
MUM,[1,"M"]
```

Types of Joins

join	leftOuterJoin	rightOuterJoin

leftOuterJoin

All keys from the left RDD are returned

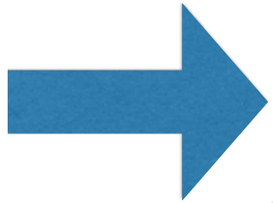
leftOuterJoin

Pair RDD1

```
BLR,3
MUM,1
DEL,2
```

Pair RDD2

```
BLR,"B"
MUM,"M"
KOL,"K"
```



```
BLR,[3,"B"]
MUM,[1,"M"]
DEL,[2,None]
```

Types of Joins

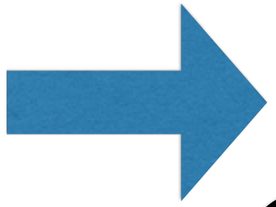
join	leftOuterJoin	rightOuterJoin

rightOuterJoin

All keys from the right RDD are returned

rightOuterJoin

Pair RDD1	Pair RDD2
BLR,3	BLR,"B"
MUM,1	MUM,"M"
DEL,2	KOL,"K"



```
BLR,[3,"B"]
MUM,[1,"M"]
KOL,[None,"K"]
```

Demo

Creating a Pair RDD

Demo

Computing a daily trend

Demo

Merging with the games data

Demo

Comparing average traffic on game days and non-game days

Summary

Representing records as Pair RDDs

Summarizing Pair RDDs using reduceByKey and combineByKey

Merging data from separate RDDs