



WIZELINE ACADEMY

Grow your career:
Free courses in Artificial Intelligence,
Software Development, User Experience and
More

WIFI: WizelineAcademy
Password: academyGDL
Slack Channel: #



@WizelineAcademy



academy.wizeline.com



/WizelineAcademy



Get notified about courses:
tinyurl.com/WL-academy

Week 3 -
Session 2/2

Spark Optimization Scenarios

Spark Details

Agenda

- **Shuffling**
- **Common Problems**
 - Imbalanced Partitioning
 - Too Much Shuffling
- **Solution Approaches**
- **What NOT to Do in Spark**



Shuffling



Shuffling

[...] **mechanism** for **redistributing data** so that it's grouped differently **across partitions**. This typically involves **copying data** across **executors and machines**.

[Spark Documentation](#)



Shuffling in a previous Implementation

Example Using Code

```
package com.wizeline.wordcount

import org.apache.spark.sql._

object WordCount {
  def wordCount(
    documents: Dataset[String],
    separatorsRegex: String = """"\s+""") : Dataset[(String, Long)] =
    {
      val words = documents.flatMap(doc => doc.split(separatorsRegex))
      val lcWords = words.map(word => word.toLowerCase)
      val counts = lcWords.groupByKey(identity).count()
      counts
    }
}
```



Shuffling Visualized

Example Using a Diagram

["Mouse Cat", "Cat Dog",
"Mouse Bear", "Dog"]

["Bear Dog", "Cat",
"Cat", "Bear Bear"]

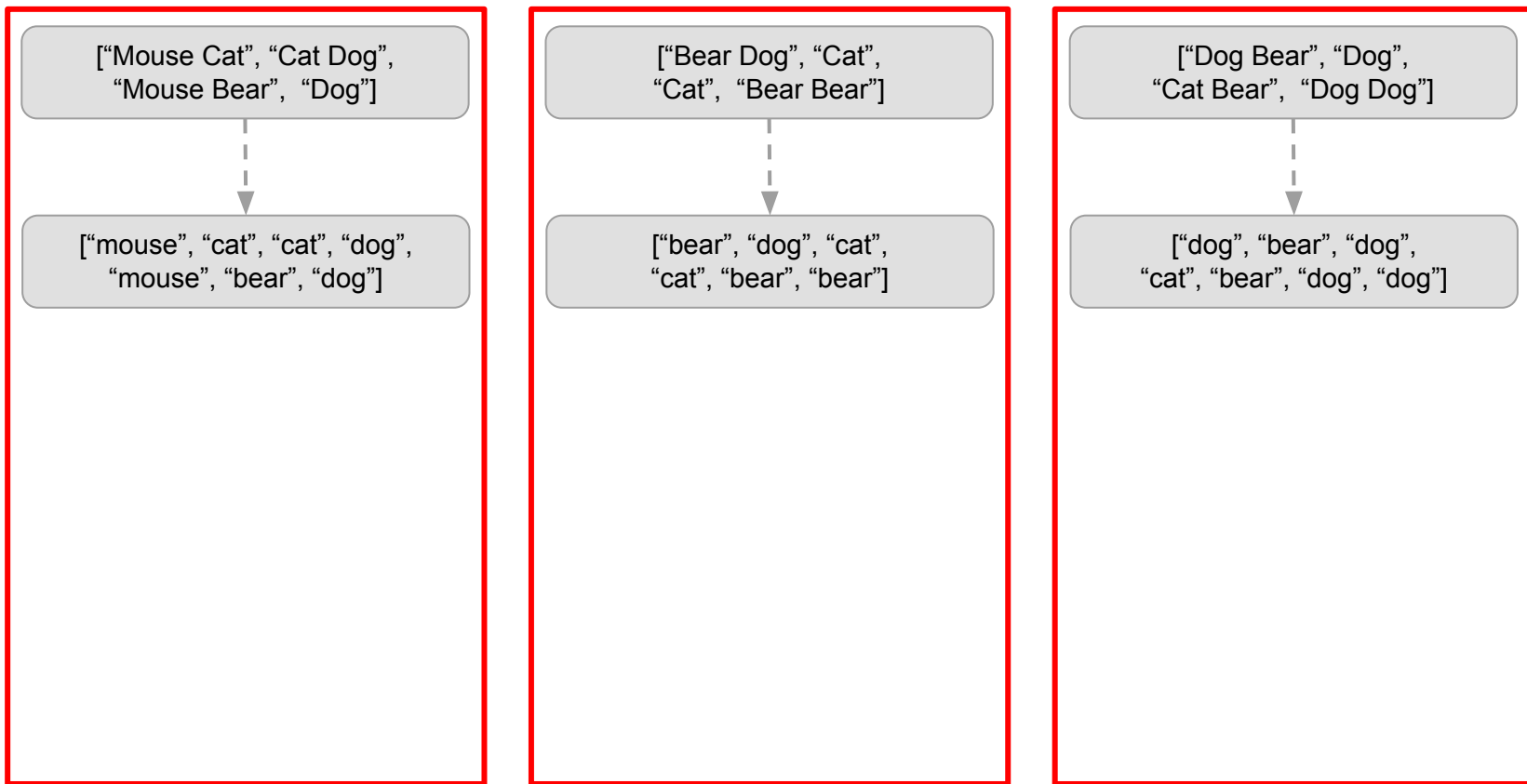
["Dog Bear", "Dog",
"Cat Bear", "Dog Dog"]



Shuffling Visualized

Example Using a Diagram

Narrow:
flatMap & map



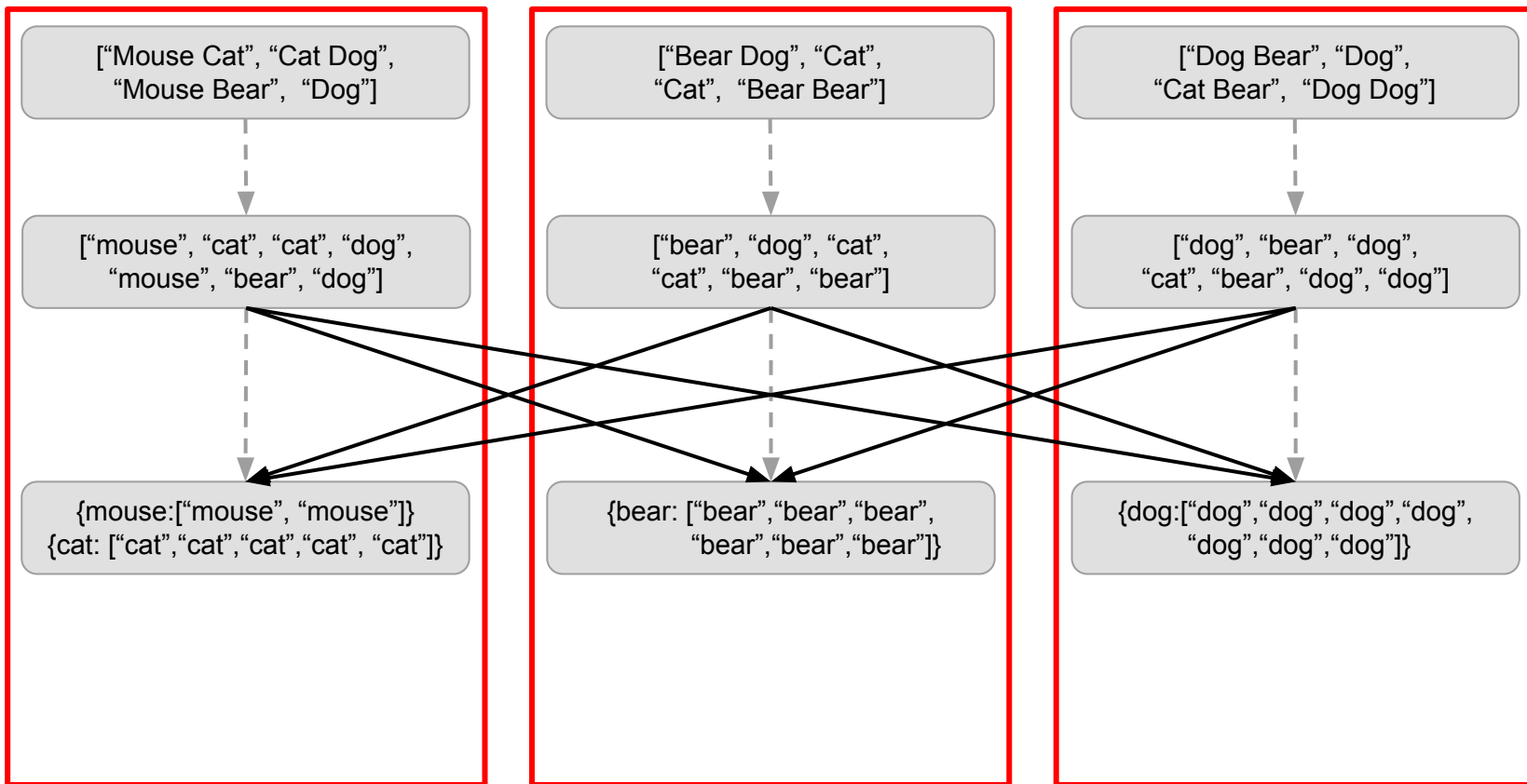


Shuffling Visualized

Example Using a Diagram

Narrow:
flatMap & map

Wide:
groupByKey



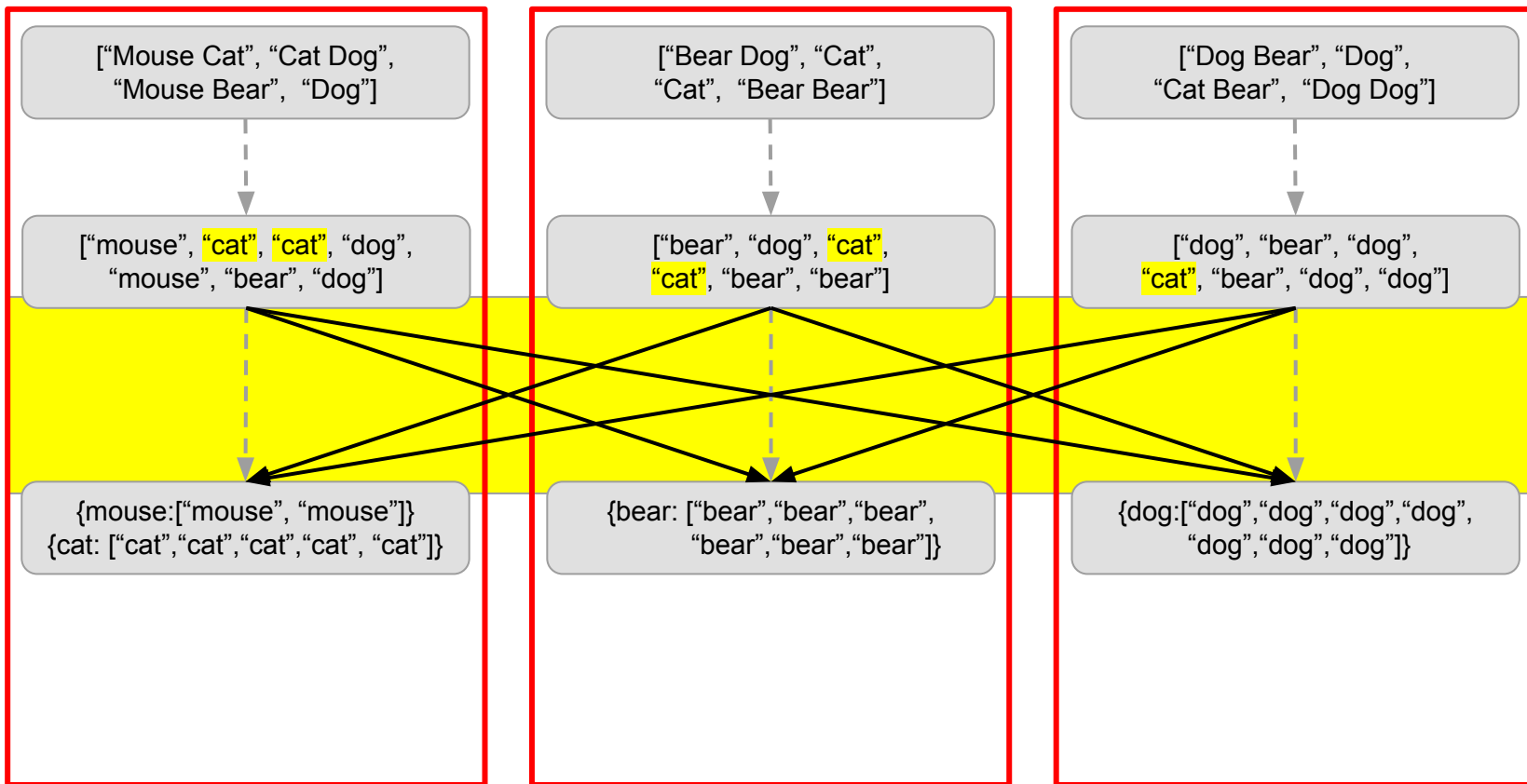


Shuffling Visualized

Example Using a Diagram

Narrow:
flatMap & map

Wide:
groupByKey



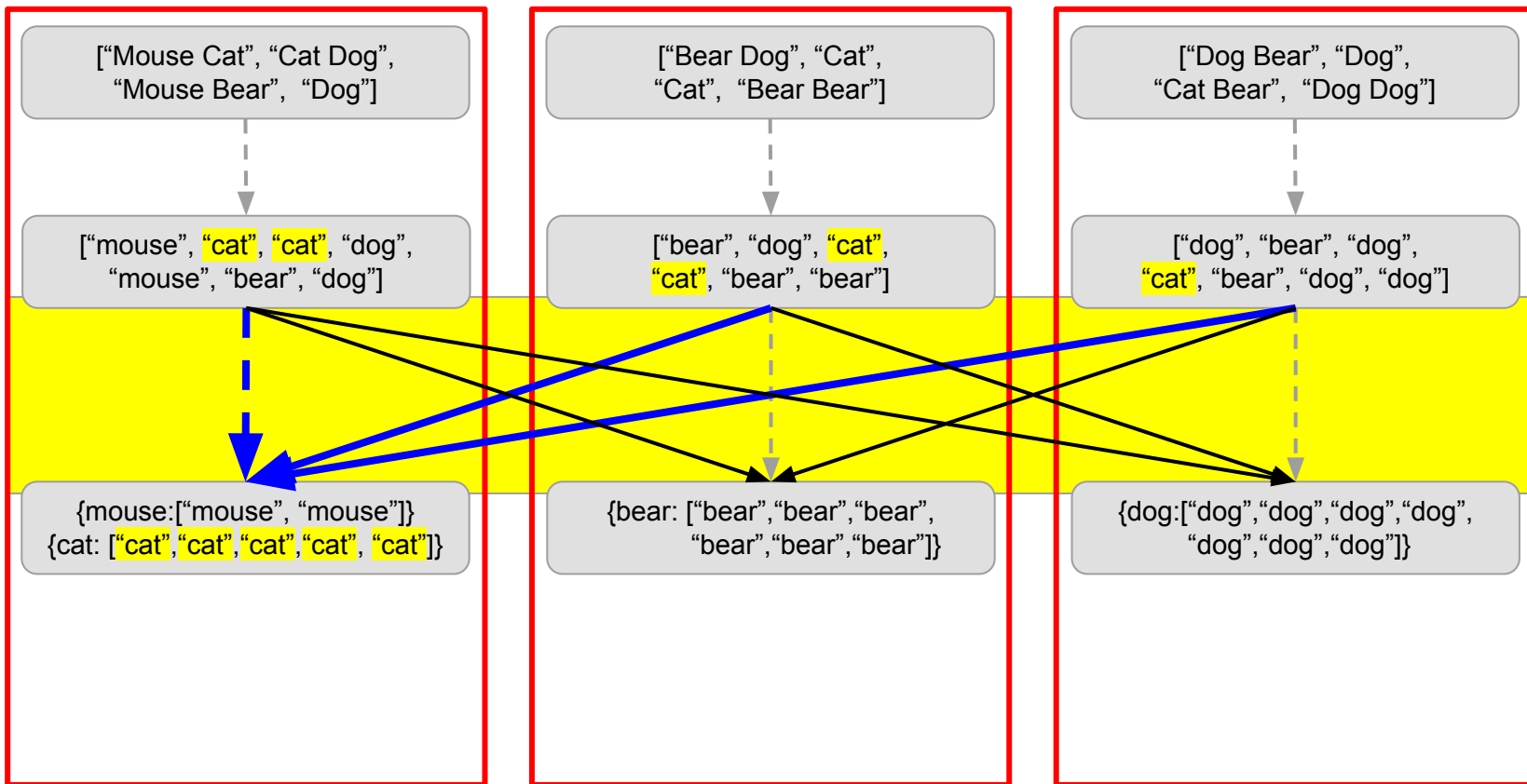


Shuffling Visualized

Example Using a Diagram

Narrow:
flatMap & map

Wide:
groupByKey





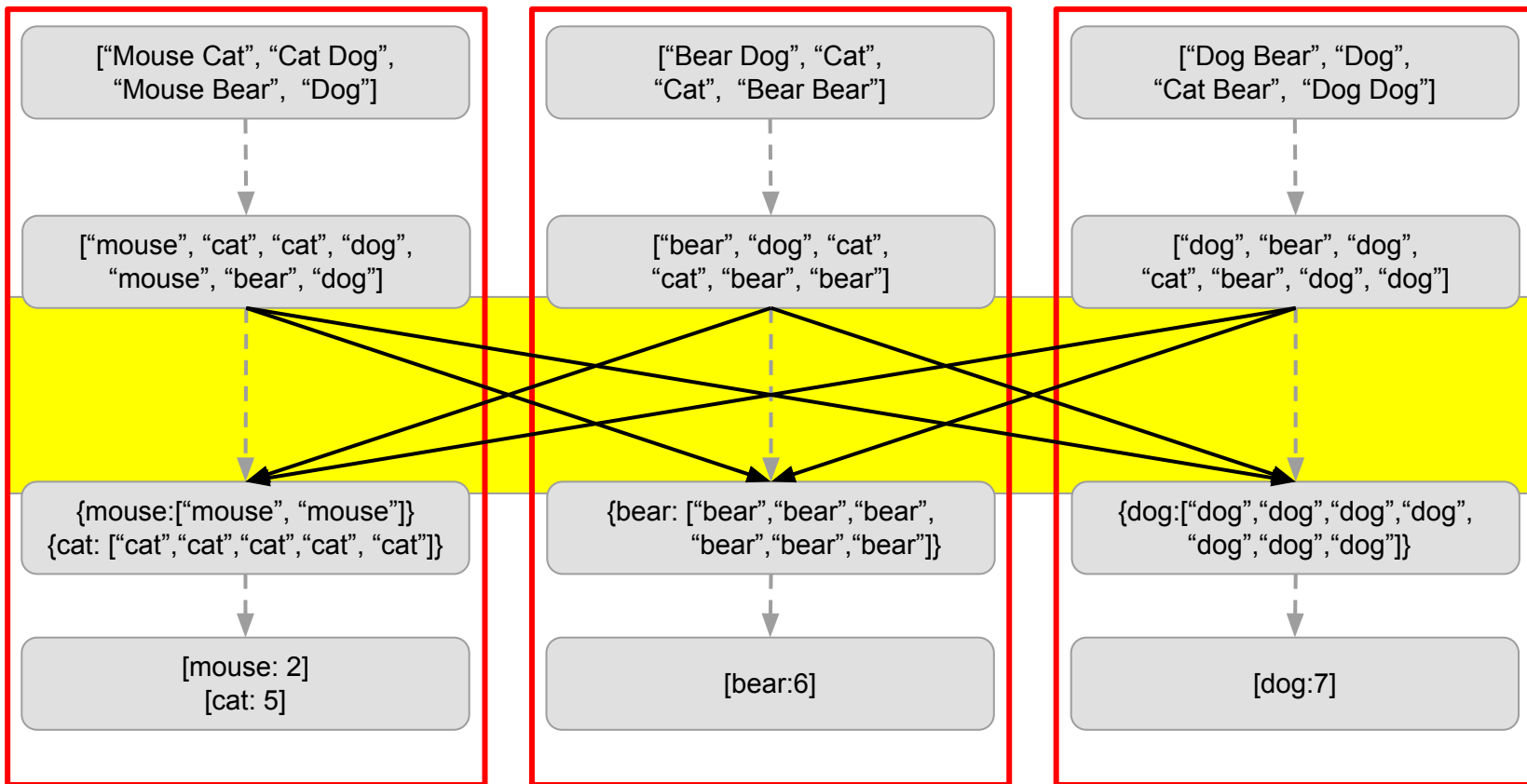
Shuffling Visualized

Example Using a Diagram

Narrow:
flatMap & map

Wide:
groupByKey

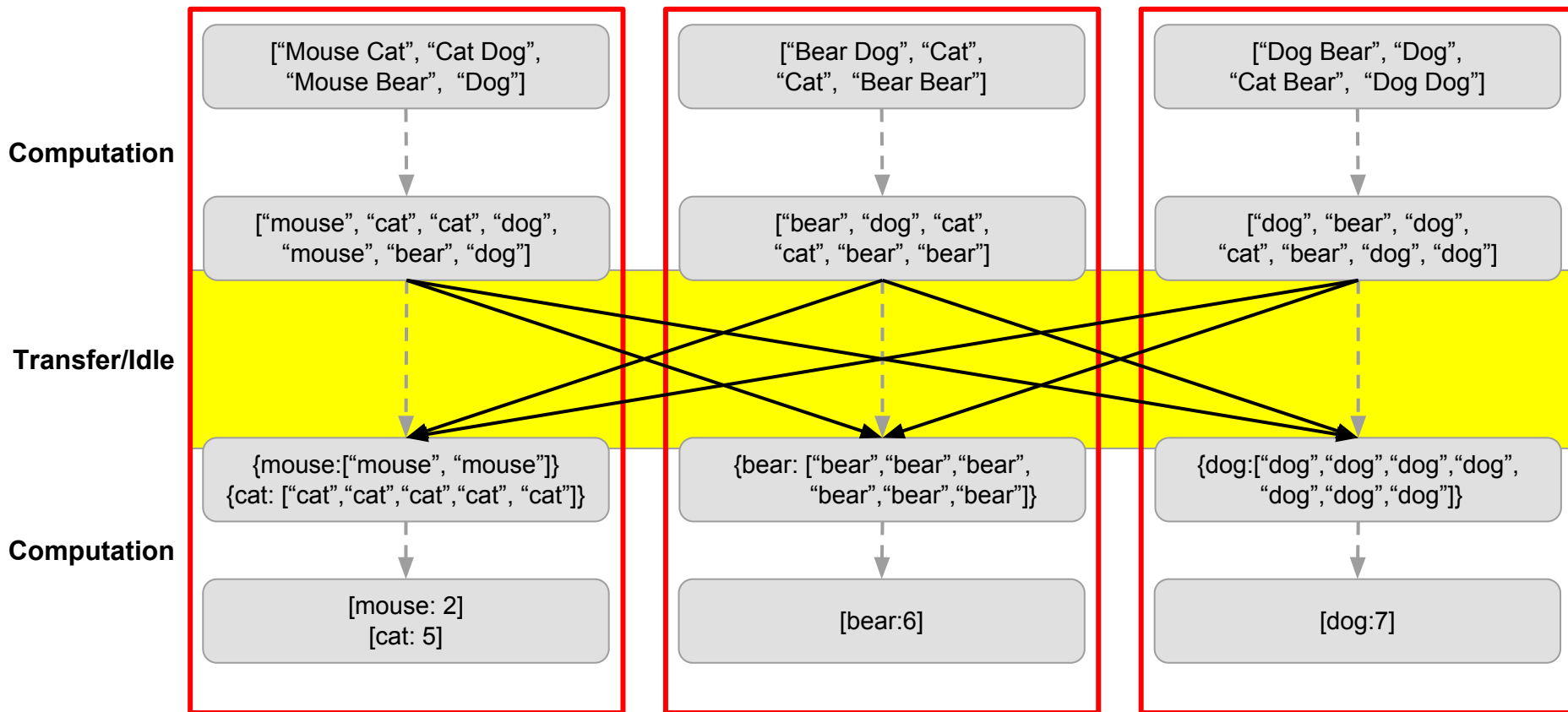
Narrow:
count





Shuffling Visualized

Example Using a Diagram





Shuffling Costs

It Takes a Toll

- **Transfer** of partitions between nodes and executors (**network movement**)
- Coordination **between nodes and executors**
 - **Idle time** of workers **waiting for others** to finish transfer

In essence... Latency!



Repartition During Shuffling

How Does Spark Do It?

- Shuffling the datasets involves **data sharding** between the nodes.
 - Spark handles it with its **default rules**
 - (HashPartitioner and RangePartitioner).
- Some partitions **may end up empty**.



Q&A





Shuffling Recap

You Should Be Able to Answer

- What is Shuffling in Spark?
- What are the costs of shuffling data?
- What is its relationship to narrow and wide dependencies?
- How does Spark handle the repartitioning?

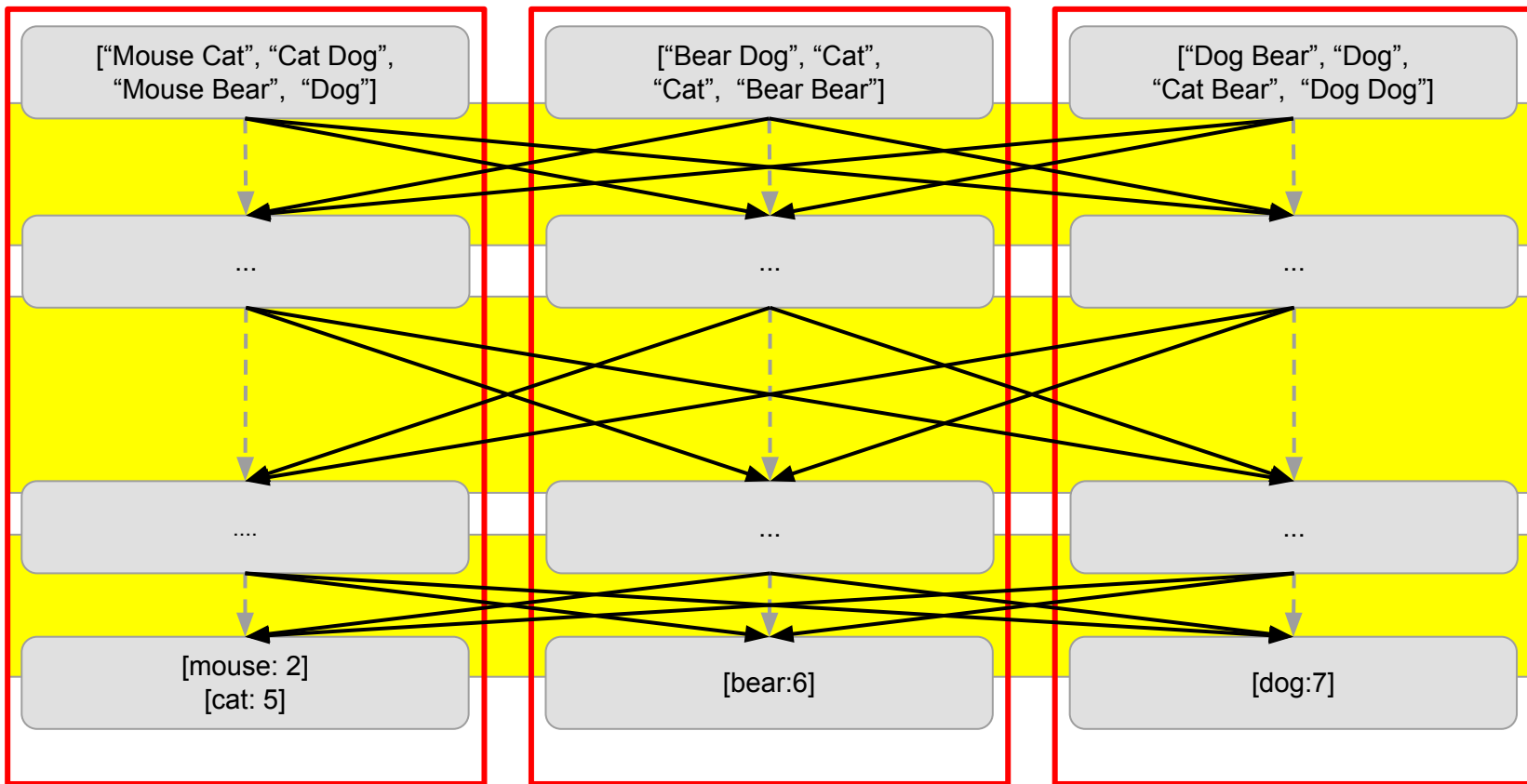


Common Problems



Too Much Shuffling

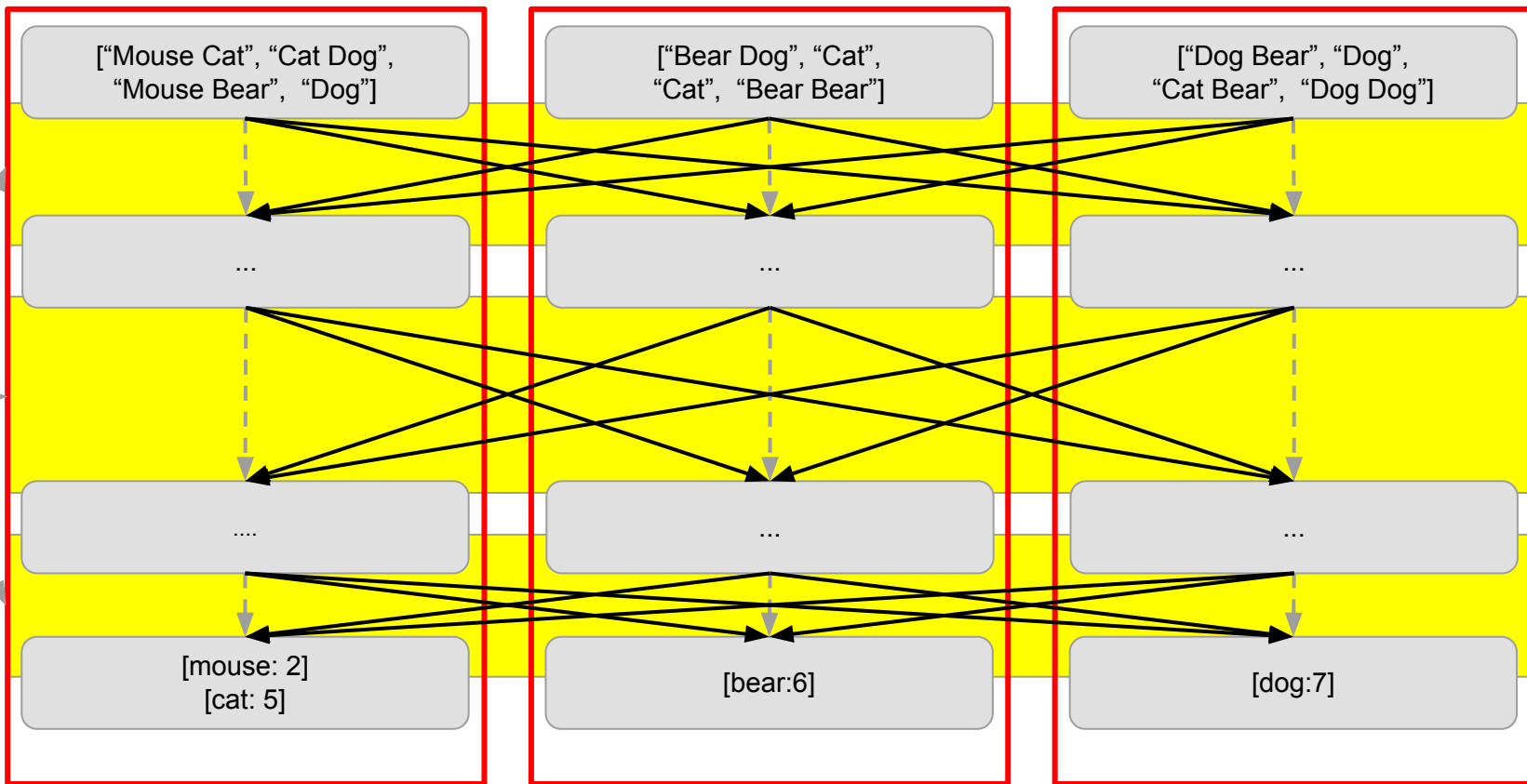
Example Using a Diagram





Too Much Shuffling

Example Using a Diagram

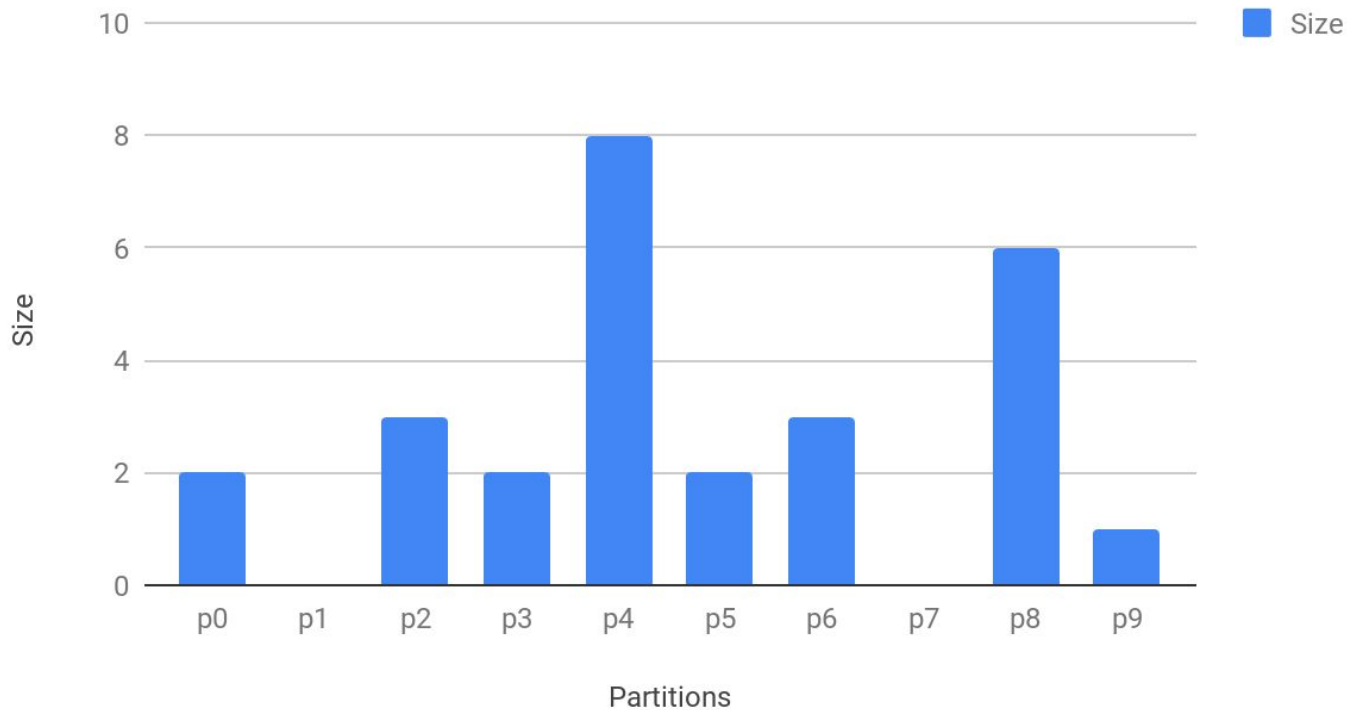




Imbalanced Partition

How Bad It Could Get

Imbalanced Partitions

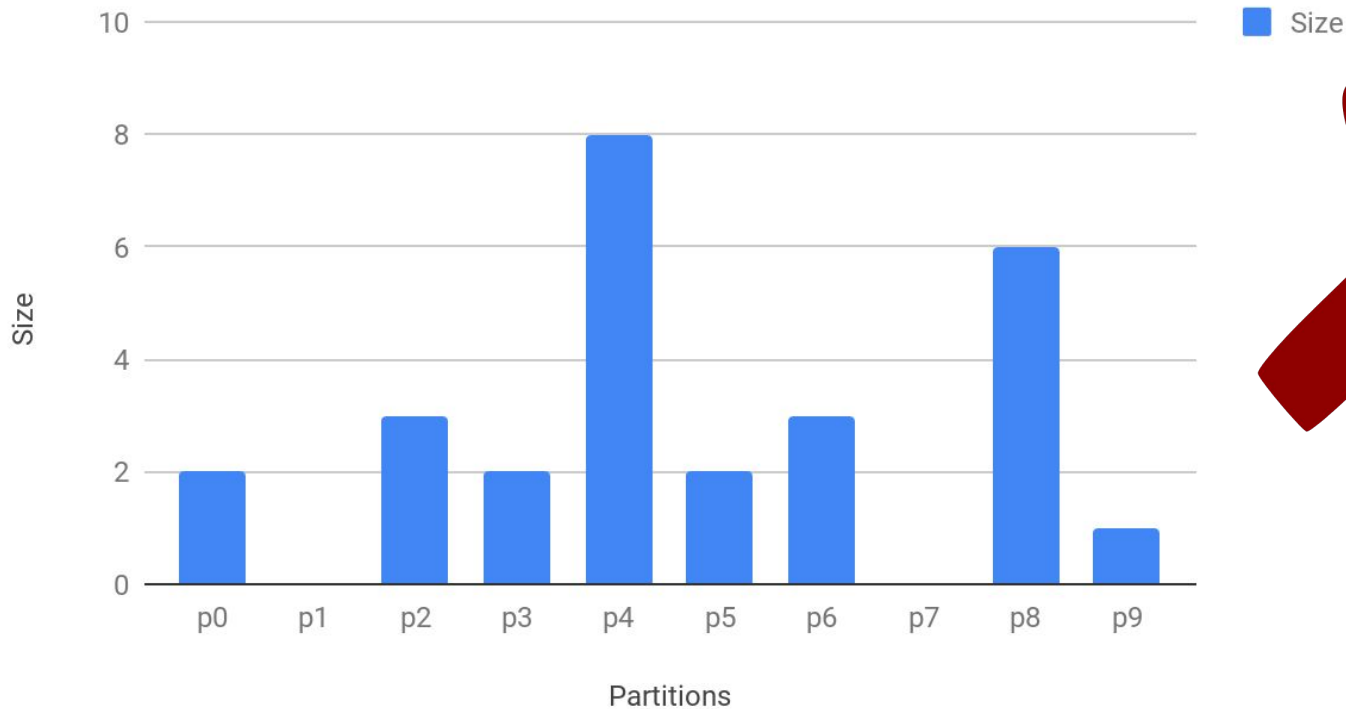




Imbalanced Partition

How Bad It Could Get

Imbalanced Partitions

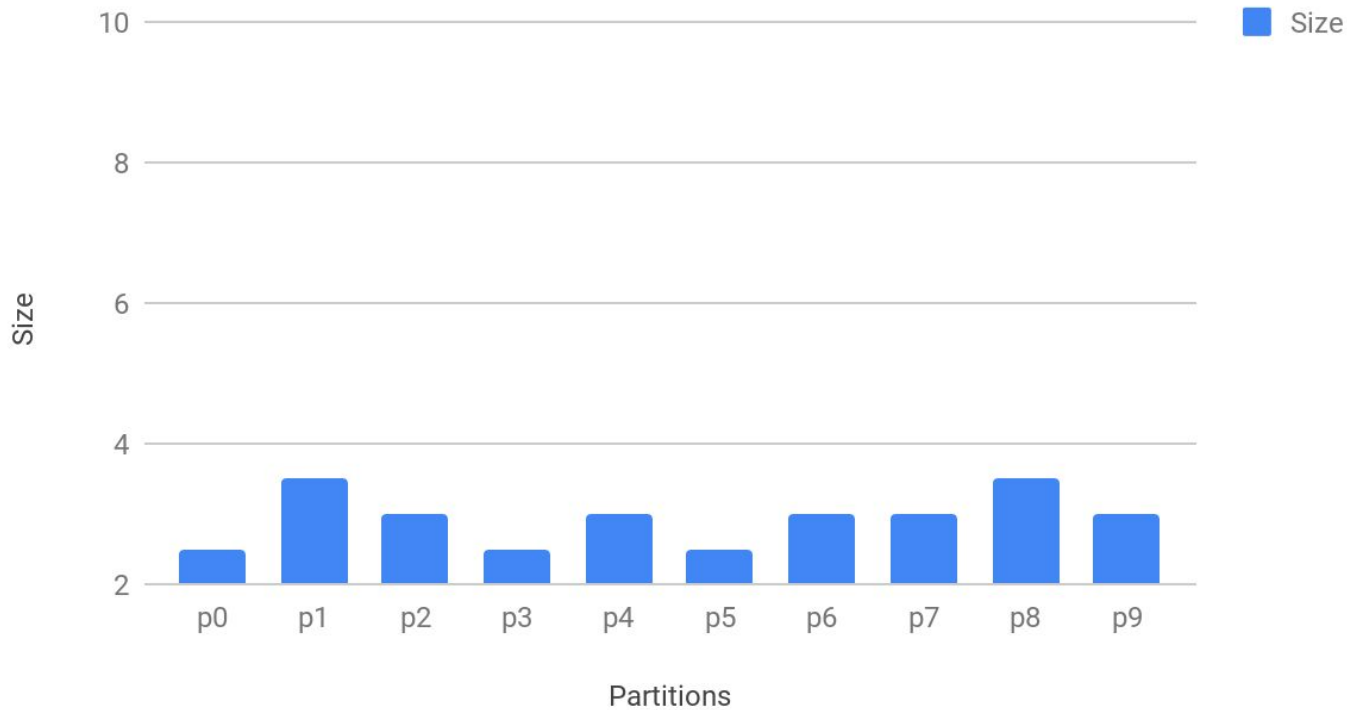




Balanced Partition

What You Expect

Balanced Partitions

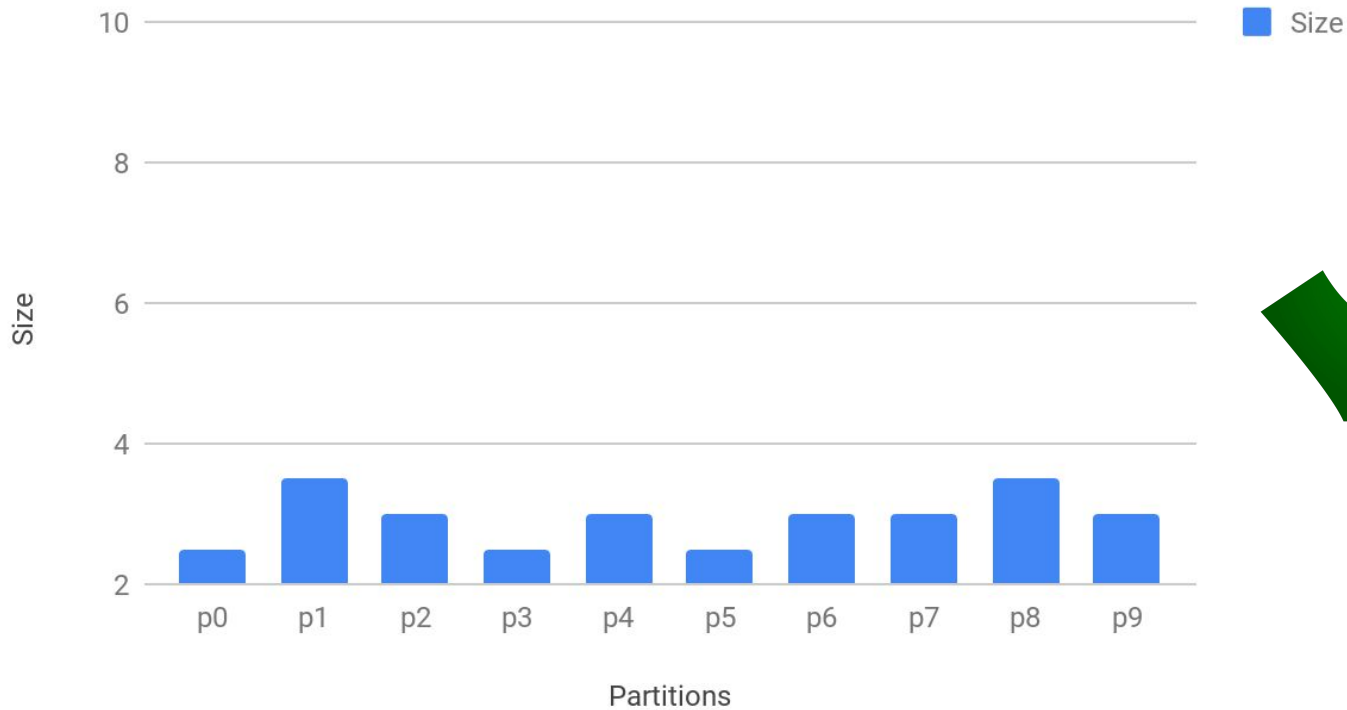




Balanced Partition

What You Expect

Balanced Partitions






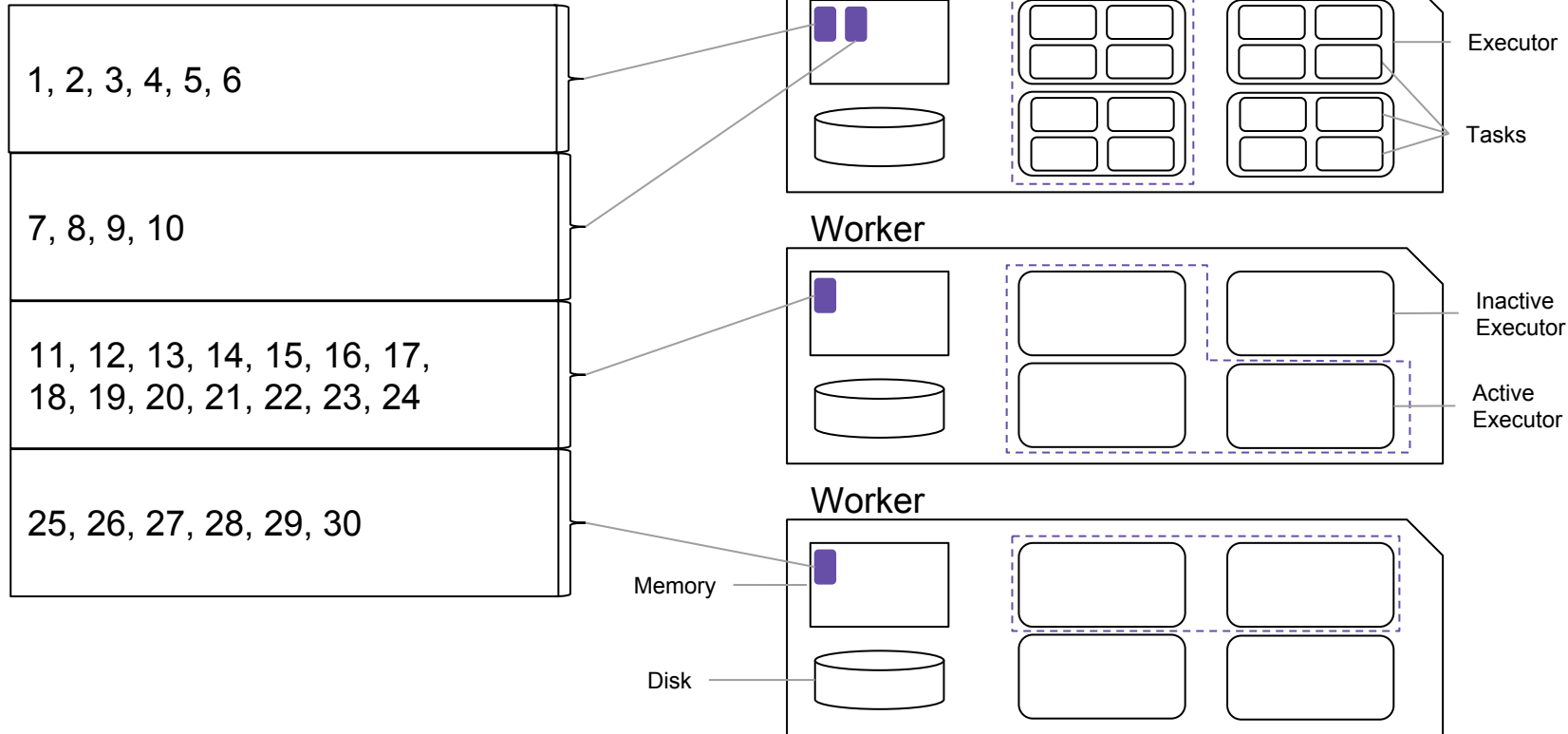
Imbalanced Partitioning

Why is Imbalanced Partitioning a
problem for Spark?



Partitioning Example

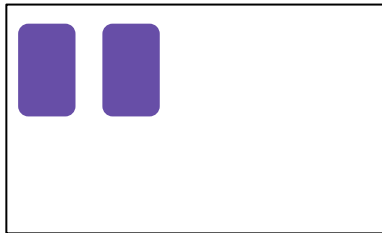
DataSet broken into 4 partitions 





Imbalanced Partitioning Problems

Worker
Node
#1



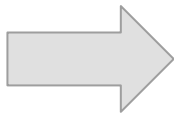
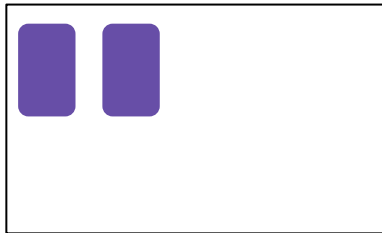
Worker
Node
#2



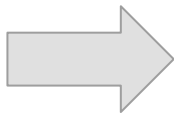


Imbalanced Partitioning Problems

Worker
Node
#1

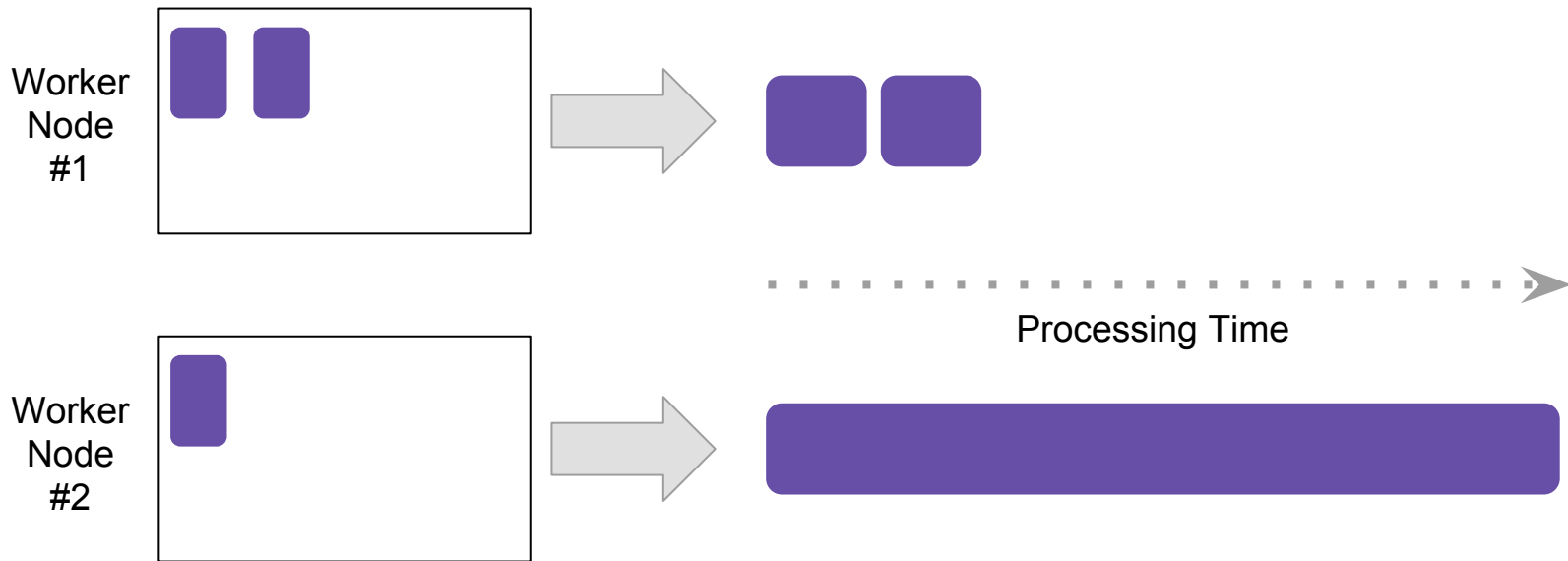


Worker
Node
#2



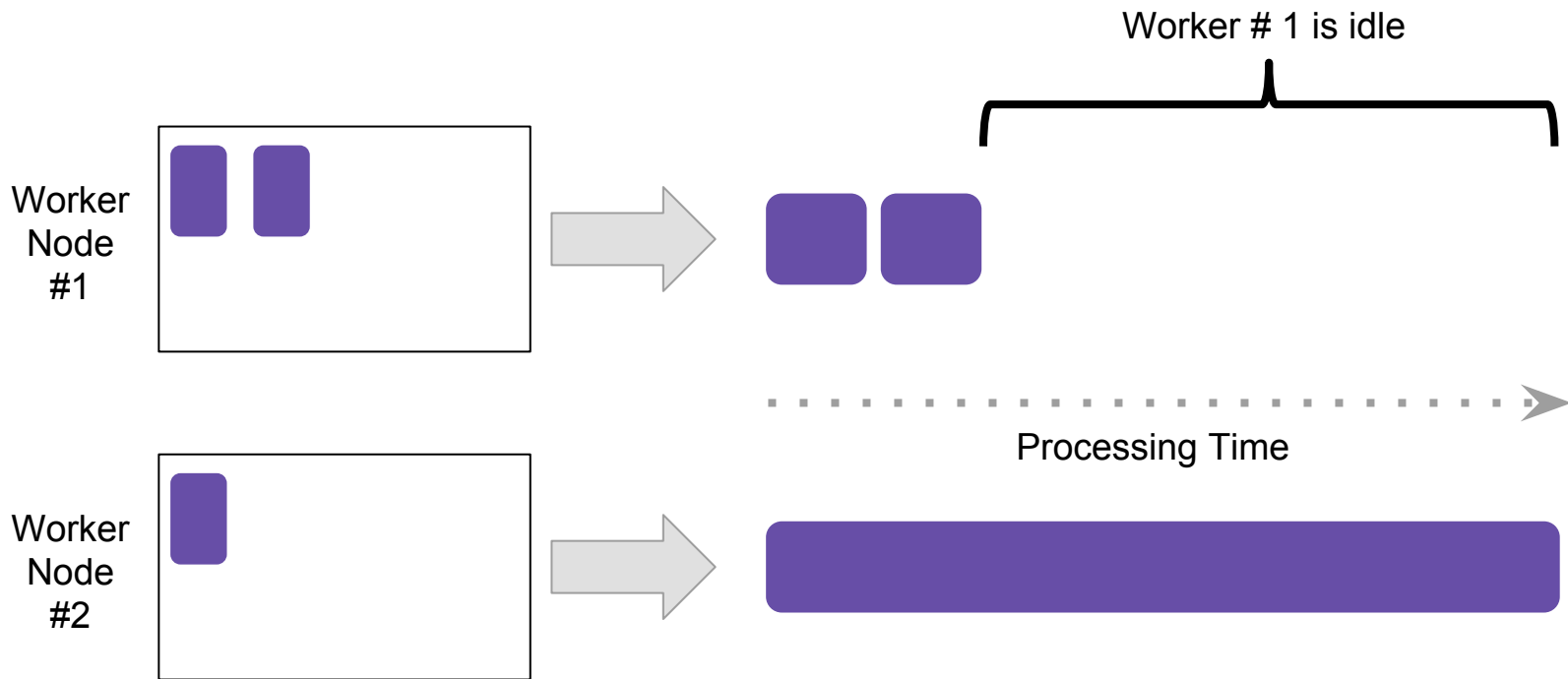


Imbalanced Partitioning Problems



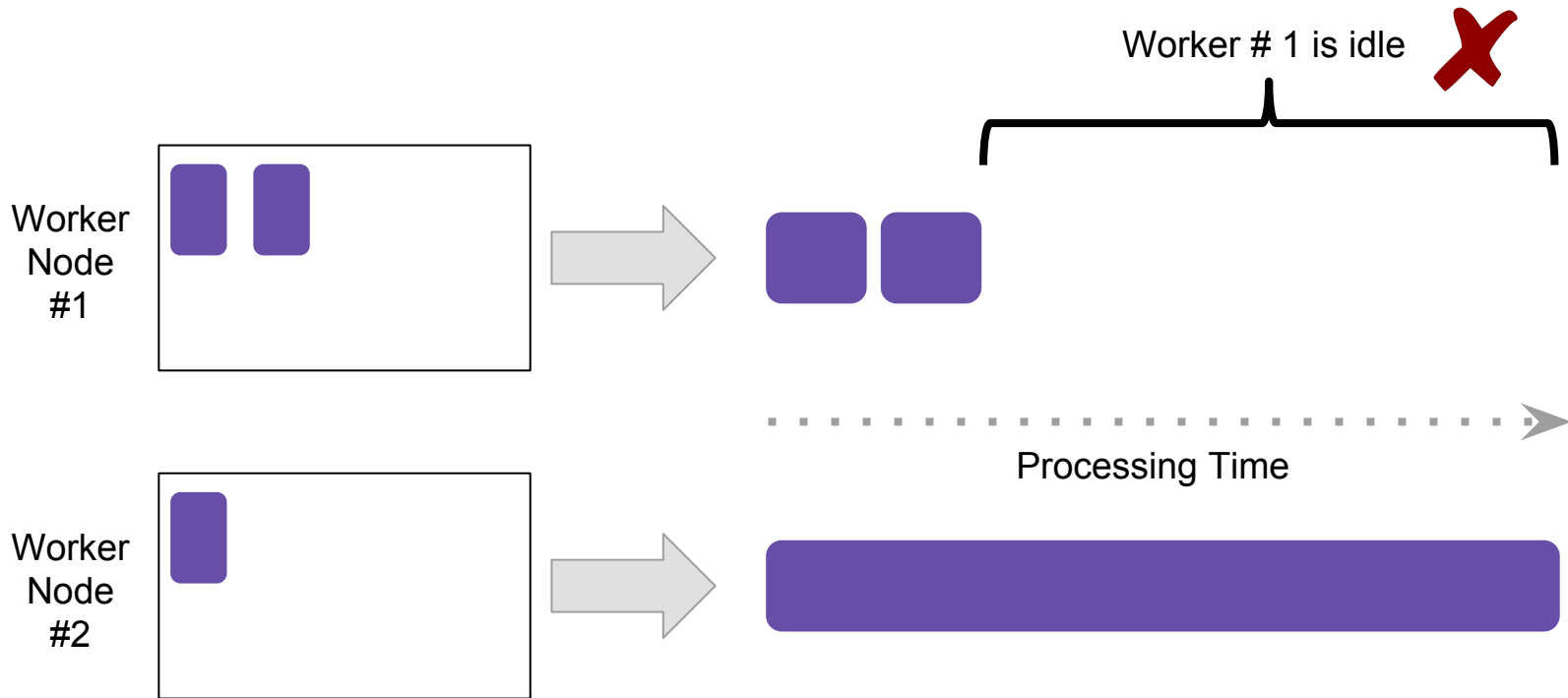


Imbalanced Partitioning Problems





Imbalanced Partitioning Problems





Q&A





Common Problems Recap

You Should Be Able to Answer

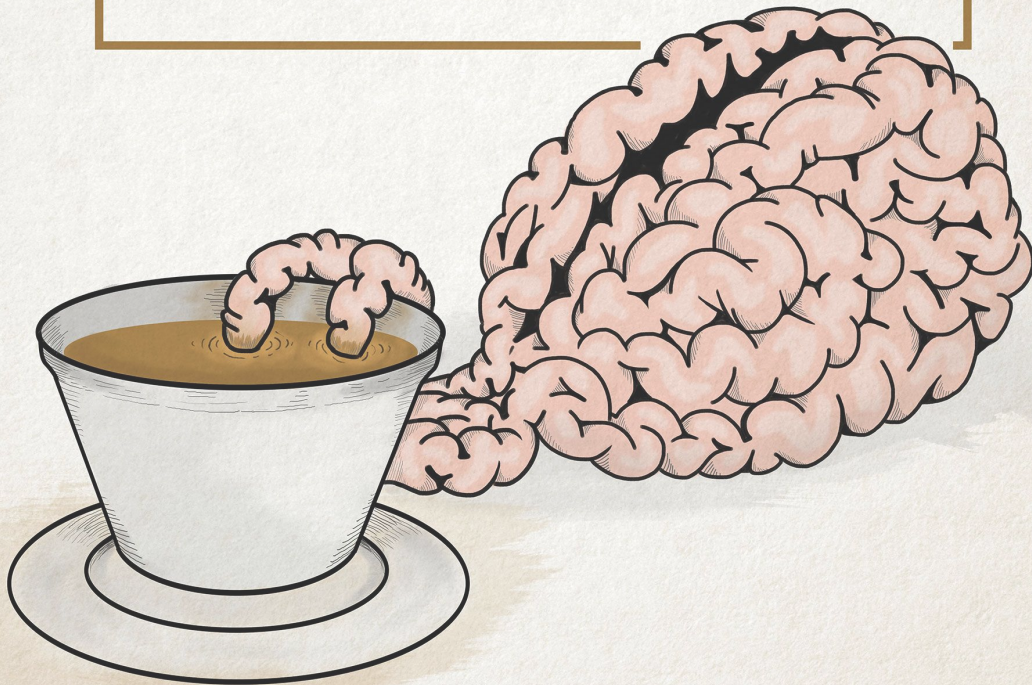
- What is an Imbalanced Partitioning (IP)?
- Why is IP a problem?
- What is “Too Much Shuffling” (TMS)?
- Why is TMS a problem?



Wait!

It's time for...

COFFEEBREAK





Solution Approaches



THE ROOT OF ALL EVIL



**PREMATURE
OPTIMIZATION IS**



TMS - DataFrames and Catalyst

Reorganize Your Partitions

Consider the Alamazon orders. Compute the total SUM of all orders per CLIENT and PRODUCT. Finally, sort the results by client and total amount spent.

The code of the first approach will incur in a **shuffle** at some point. Optimize it.

```
// Base Scenario - Using RDDs
val alamazon = "gs://de-training-input/alamazon/200000/client-orders/"
val orders = spark.read.json(alamazon)
                        .sample(false, 0.25)
                        .cache
val ordersRDD = orders.rdd
ordersRDD.map(row => ((row.getString(0), row.getString(2)), row.getDouble(5)))
        .groupByKey
        .map { case (key, iter) => (key, iter.sum) }
        .sortBy(pair => (pair._1._1, pair._2), ascending = false)
        .take(20)
        .foreach(row => println(row))
```



IP and TMS - Using Caching and Repartition

Reorganize Your Partitions

Consider the Alimazon orders and obtain several statistical metrics for the money spent per `client_id`. The metrics should be on separate reports for SUM, AVERAGE, MIN and MAX.

This code will incur in a shuffle at some point. Optimize it for performance.

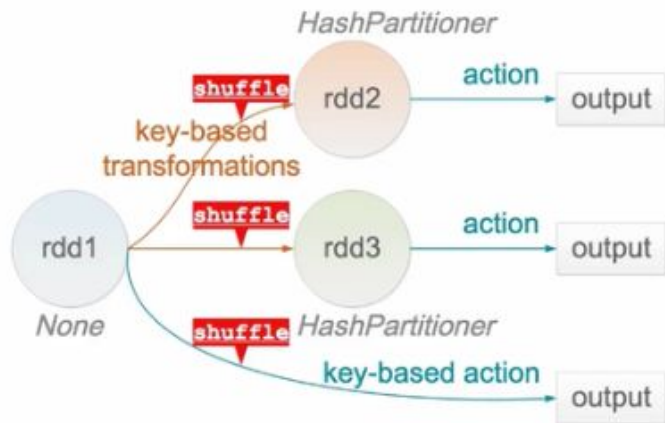
```
val alimazon = "gs://de-training-input/alimazon/200000/client-orders/"
val orders = spark.read.json(alimazon).cache
// Base Scenario - Data as read by spark
orders.groupBy("client_id").sum("total").show
orders.groupBy("client_id").avg("total").show
orders.groupBy("client_id").min("total").show
orders.groupBy("client_id").max("total").show
```



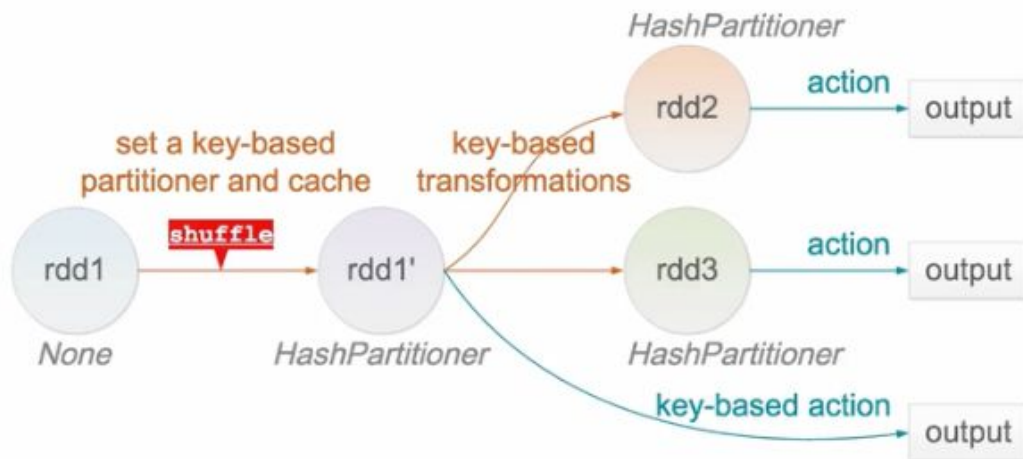
IP and TMS - Using Caching and Repartition

Basically Reorganize Your Partitions

Suboptimal case



Optimal case





IP and TMS - Use Repartition

Basically Reorganize Your Partitions

Alamazon partitionates its data using month as the base. You intend to change that, but you need evidence of a better partition. Your test consists on computing several metrics for the month of February.

Several repartitions will be tested. Partition analysis and execution time will be considered as our comparison criteria.

```
// Base scenario:  
val ordersByMonth = orders.repartition(12, month($"timestamp").alias("month"))  
                                .cache  
val ordersInFebruary = ordersByMonth.filter(month($"timestamp") === "2")  
ordersInFebruary.groupBy(month($"timestamp")).sum("total").show  
ordersInFebruary.groupBy(month($"timestamp")).avg("total").show  
ordersInFebruary.groupBy(month($"timestamp")).min("total").show  
ordersInFebruary.groupBy(month($"timestamp")).max("total").show
```




Spark's DO NOTs



Spark Code Can Also Be Bad

...but





Over-Caching

Caching More than Your Cluster Can Handle

- **AVOID** caching at every step.
- **AVOID** using a suboptimal storage level.
- **AVOID** caching unfrequently used objects.
- **AVOID** unnecessary disk use or CPU overhead when selecting a storage level.



One Program Per Partition

It's Distributed for a Reason

- **AVOID** writing programs that will execute differently per partition.

```
rdd.mapPartitions(iterator => {  
    val row = iterator.next  
    if isInPartitionOne(row){  
        ...  
    } if else isInPartitionTwo(row){  
        ...  
    } if else isInPartitionThree(row){  
        ...  
    } else {  
        ...  
    }  
    ...  
})
```



Language Native vs Spark Native

Spark's framework works only when applied

- **AVOID** using language functions over Spark functions.
- **AVOID** using for loops over Spark data structures.

FURTHER READING



Tips to Write Better Spark Jobs

<https://www.slideshare.net/databricks/s-trata-sj-everyday-im-shuffling-tips-for-writing-better-spark-programs>

Shuffling Explained in Depth

<http://hydronitrogen.com/apache-spark-shuffles-explained-in-depth.html>

For the Knowledge Hungry...

Shuffling

<https://jaceklaskowski.gitbooks.io/mastering-apache-spark/spark-rdd-shuffle.html>



Assignment

To Work on Your Own at Home

A photograph of a silver laptop and a white cup of coffee on a light-colored wooden desk. The laptop is open, and the coffee is in a simple white mug. The background is slightly blurred, focusing on the desk items.

**Dataset and Dataframe
Operations Using Alimazon**



Assignment

A full-fledged report for Alimazon

In this assignment you will be working with the stock-order dataset on this URI

`gs://de-training-input/alimazon/200000/stock-orders`

Mock stock-orders

id	product_id	quantity	timestamp	total
dw329	d5A	2	2018-08-30 19:00	\$5,712.00
sm199	d5C	3	2018-08-30 19:00	\$712.00
nb271	zp2	2	2018-08-30 19:00	\$313.00
xq109	zp4	6	2018-08-30 19:00	\$1,823.00

Stock-orders schema

name	type	notes
id	string	Stock order ID
product_id	string	UUID for a product
quantity	long	Amount of items in order
timestamp	string	Date and time in formatted string
total	double	Monetary value in order



Assignment

A full-fledged report for Alimazon

The *Alimazon Strategy Department* wants to analyze the trend over time of the product categories they have to stock. They want to do it as a **time series** considering **year and week number**.

The product categories are defined by the **first two characters** of the `product_id`

d571c165-e64b-4ac3-b09d-7725f49b7dcd ⇒ Category **d5**

Each category will be modeled using the Top 5 of most purchased product as well as the Top 5 of least purchased products. This is to analyze the effects at each tail end. The output tables should be separate, but following the same schema.

The output should include the total amount of items purchased each week as well as the total amount spent in dollars.



Assignment

A full-fledged report for Alimazon

Output table example

year	week_num	prod_cat	total_qty_top5	total_spent_top5
2018	35	d5	17,403	\$235,768.21
2018	34	h5	12	\$5,127.00
2018	33	zp	729	\$5,689,141.00
2018	32	77	6,123	\$781,902.00

Output table schema

name	type	notes
year	int	Year of data point
week_num	int	Week number
prod_cat	string	Product category
total_qty_top5	long	Total items purchased from top 5 items
total_spent_top5	double	Total money spent in top 5 items



Assignment

A full-fledged report for Alimazon

BUT... wait a minute...

Besides the two tables with the time series, the *Alimazon Strategy Department* has also requested a third table that describes the categories discovered. This will allow them to do percent comparisons in their final analysis.

This final table should include the relation between each category to

- Their top 5 most purchased products
- Their top 5 least purchased products
- Total amount of items purchased in the full category
- Total amount of money spent in the full category



Assignment

A full-fledged report for Alimazon

Output table example

category	top5_most	top5_least	total_qty_cat	total_spent
d5	d5A,d5B,d5C,d5D,d5E	d5V,d5W,d5X,d5Y,d5Z	10,000,000	\$421,347.09
zp	zp1,zp2,zp3,zp4,zp5	zp6,zp7,zp8,zp9,zp0	6,381	\$11,379,129.00

Output table schema

name	type	notes
category	string	Category of the products
top5_most	Array[string]	Array of Top 5 Most Purchased product_ids
top5_least	Array[string]	Array of Top 5 Least Purchased product_ids
total_qty_cat	long	Total items purchased in the category
total_spent_cat	double	Total money spent in the category



Assignment

A full-fledged report for Alimazon

You are tasked to deliver a script that will deliver all three tables

- Report with Top 5 Most Purchased Products
- Report with Top 5 Least Purchased Products
- Report with Categories

Your code should be **EFFICIENT**.

Your code should use optimization techniques **CORRECTLY**.

Your code should include everything you have learned **SO FAR**.



C6 - Data Engineering Academy

Where Can I Get this Presentation?

PDF is on the Slack channel

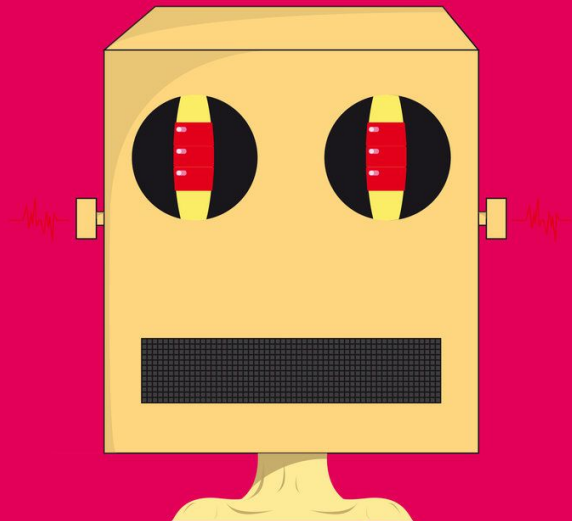


C6 - Data Engineering Academy

Your feedback is very valuable!

<https://goo.gl/forms/kRKCihEMsRIQZjeK2>

Survey Answers == Class Attendance



everyday
i'm
SHUFFLING



THANK YOU

wizeline.com | confidential - do not distribute

WIZELINE®

