Week 1 - Session 2/2

MapReduce and Fault Tolerance

Spark Details

WIZELINE

Agenda

- **Fault Tolerance at the Storage Level**
- **MapReduce**
  - Parallelization
  - The Framework
  - Iterations and MapReduce
- **Fault Tolerance at the Application Level (Spark)**

Fault Tolerance

Distributed Storage

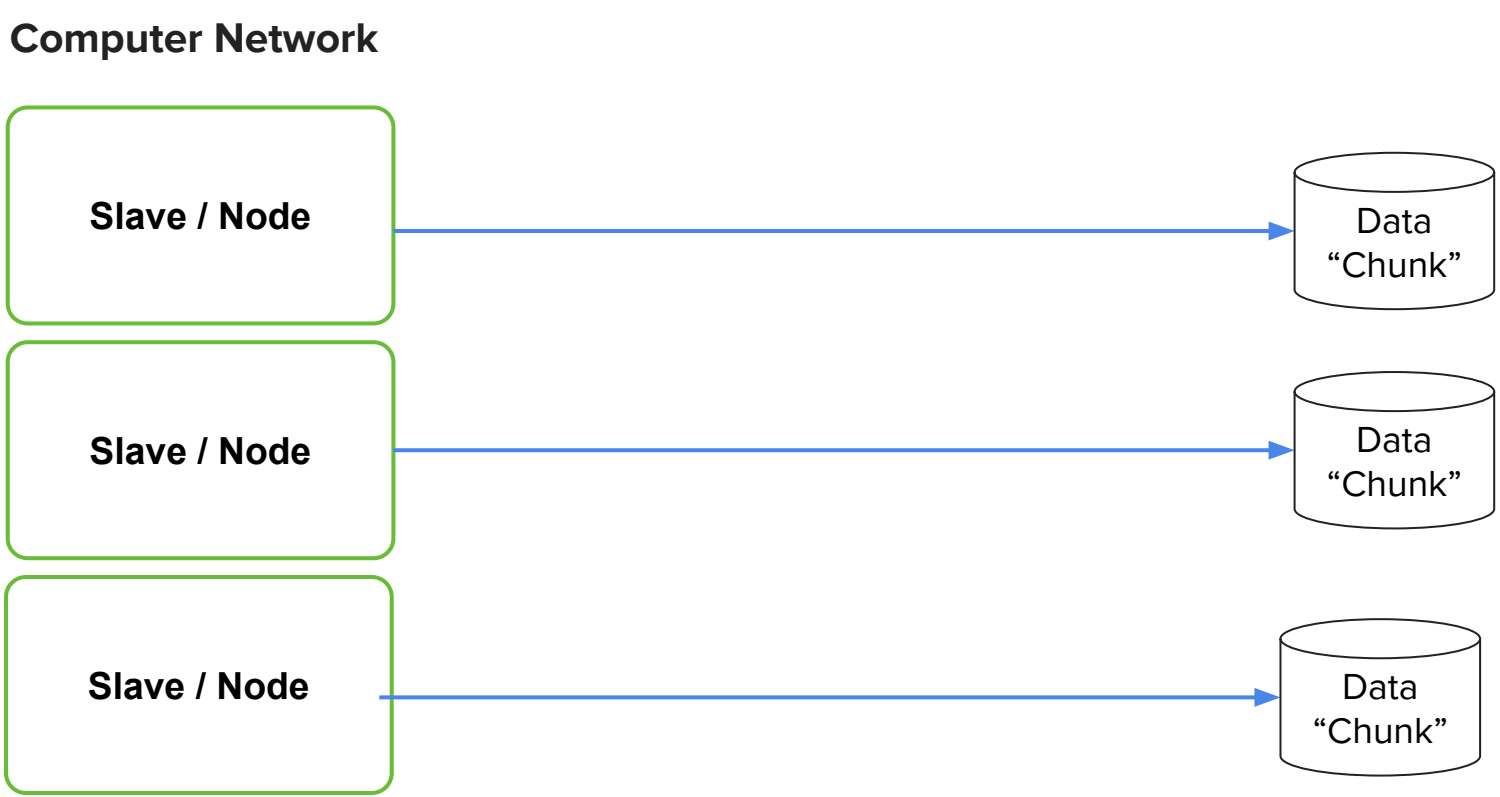
A **distributed data store** is a **computer network** where information is **stored** on **more** than one **node**.

Wikipedia

.

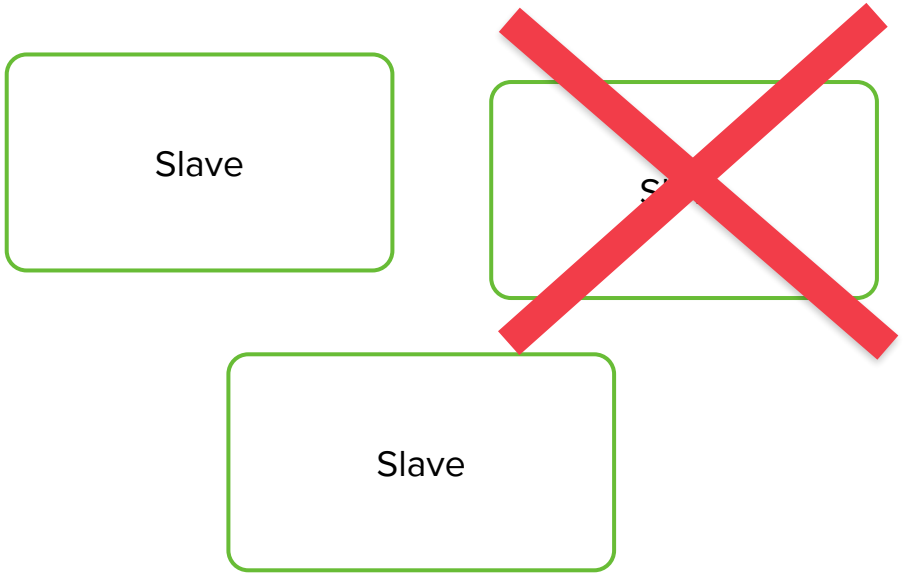
Distributed Storage

With Spark



What Happens When a Node Dies?

With Spark

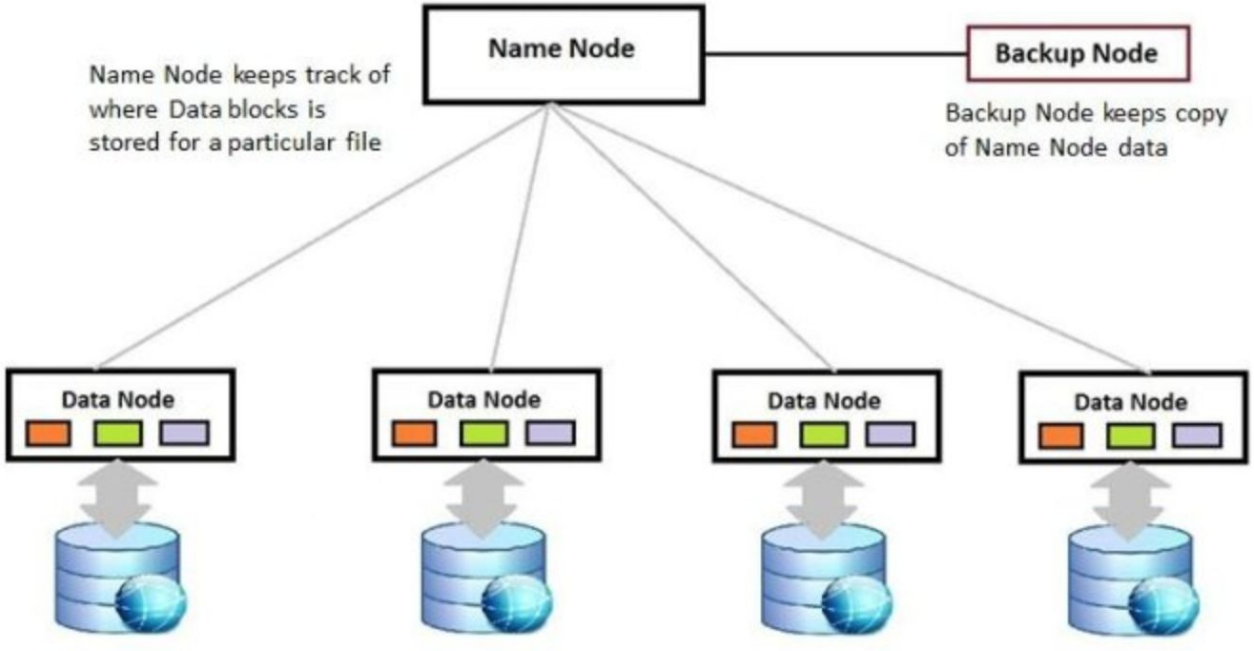


What happens if:

a) A node is no longer active?

b) The data chunks are corrupted?

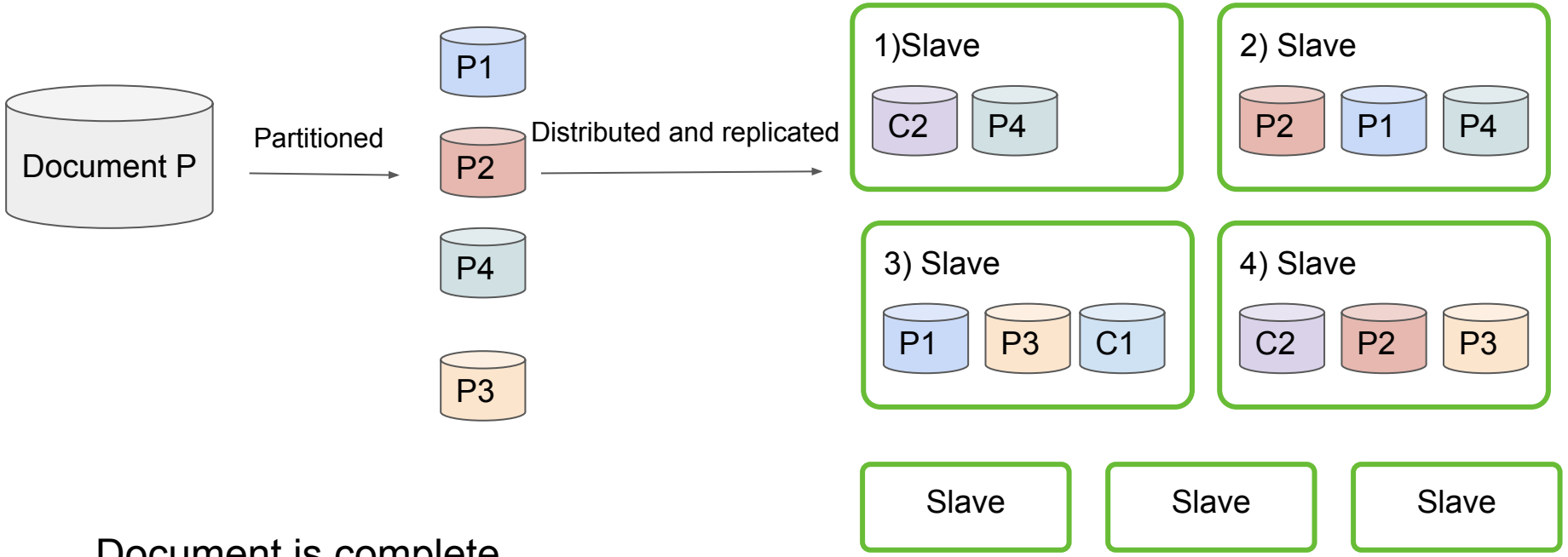
Fault Tolerance: Storage Handling



HDFS // Google GCS // Amazon S3: All storage clusters accomplish fault tolerance

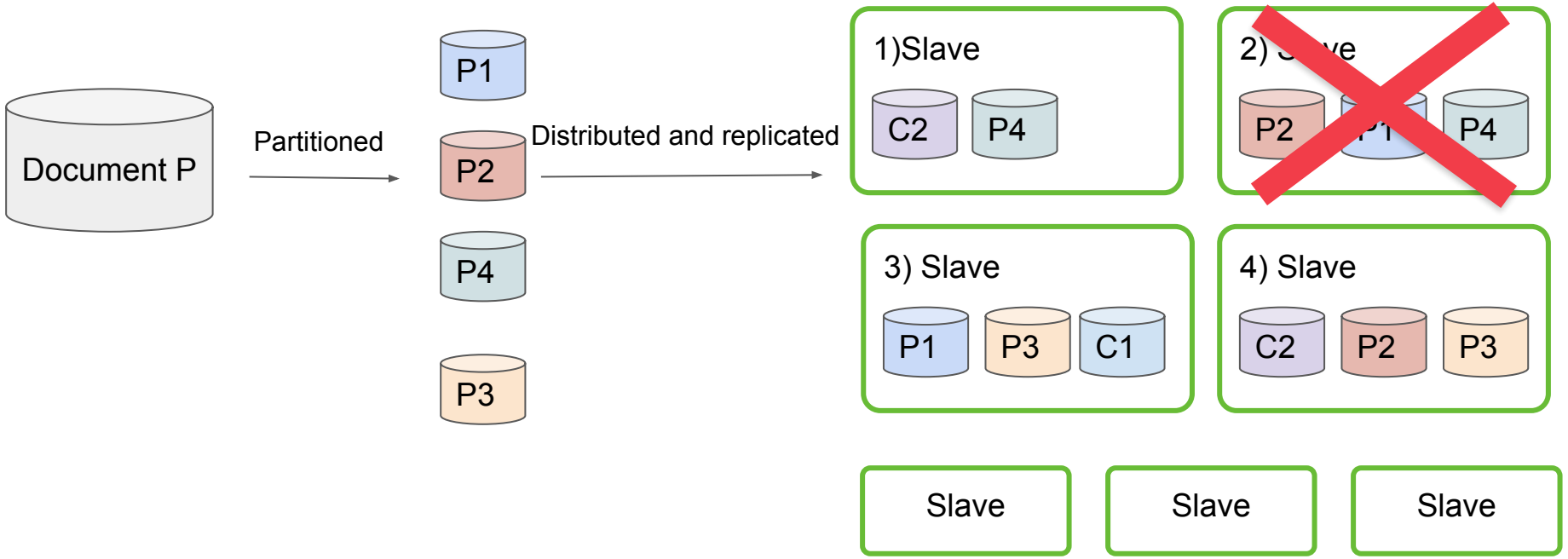
- Partitioning the dataset amongst different nodes (physical/virtual machines)
- Automated handling and balancing

Fault Tolerance: Storage Replication

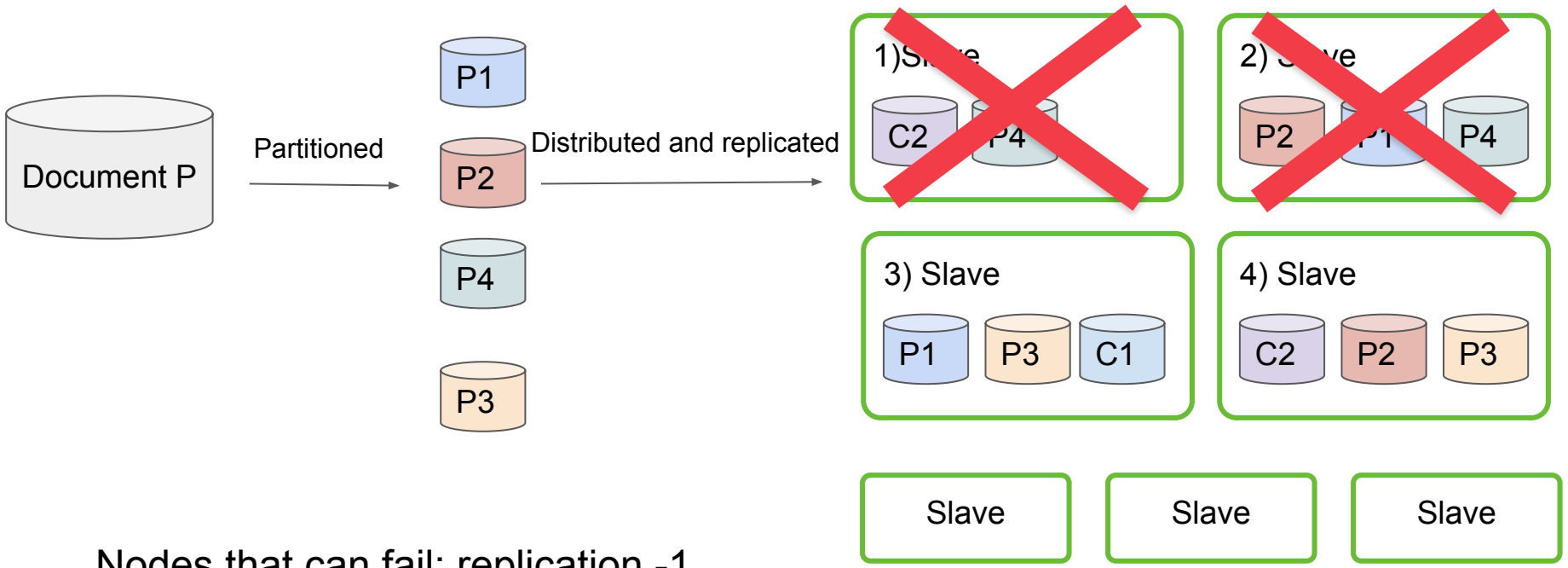


Document is complete

Fault Tolerance: Storage Replication

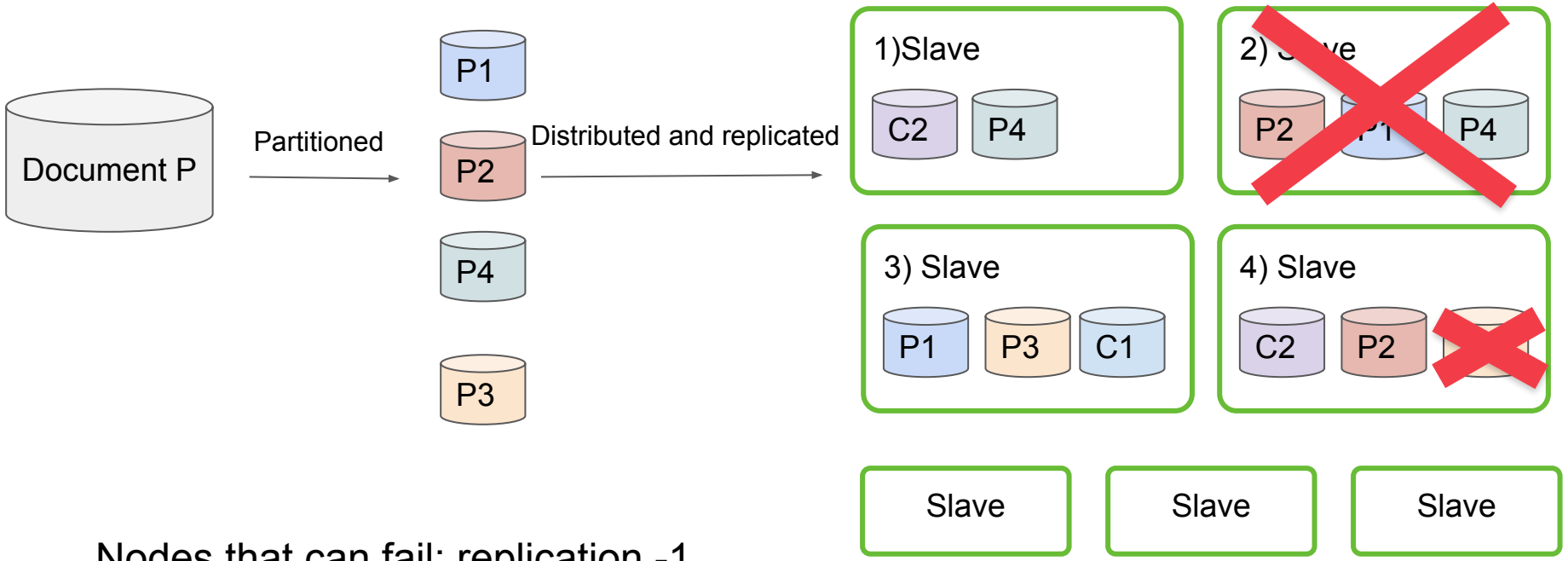


Fault Tolerance: Storage Replication



Nodes that can fail: replication -1

Fault Tolerance: Storage Replication



Nodes that can fail: replication -1

Activity

1. Connect to your zeppelin.
2. Open a new ssh tunnel to port 9870 and view the page.
3. Once on the HDFS, click around to answer the following questions:
    a. How much memory has been assigned?
    b. How many nodes are active and how many are death?
    c. How much capacity is remaining?
    d. What are the cluster metrics?
    e. Can you examine each node by itself?

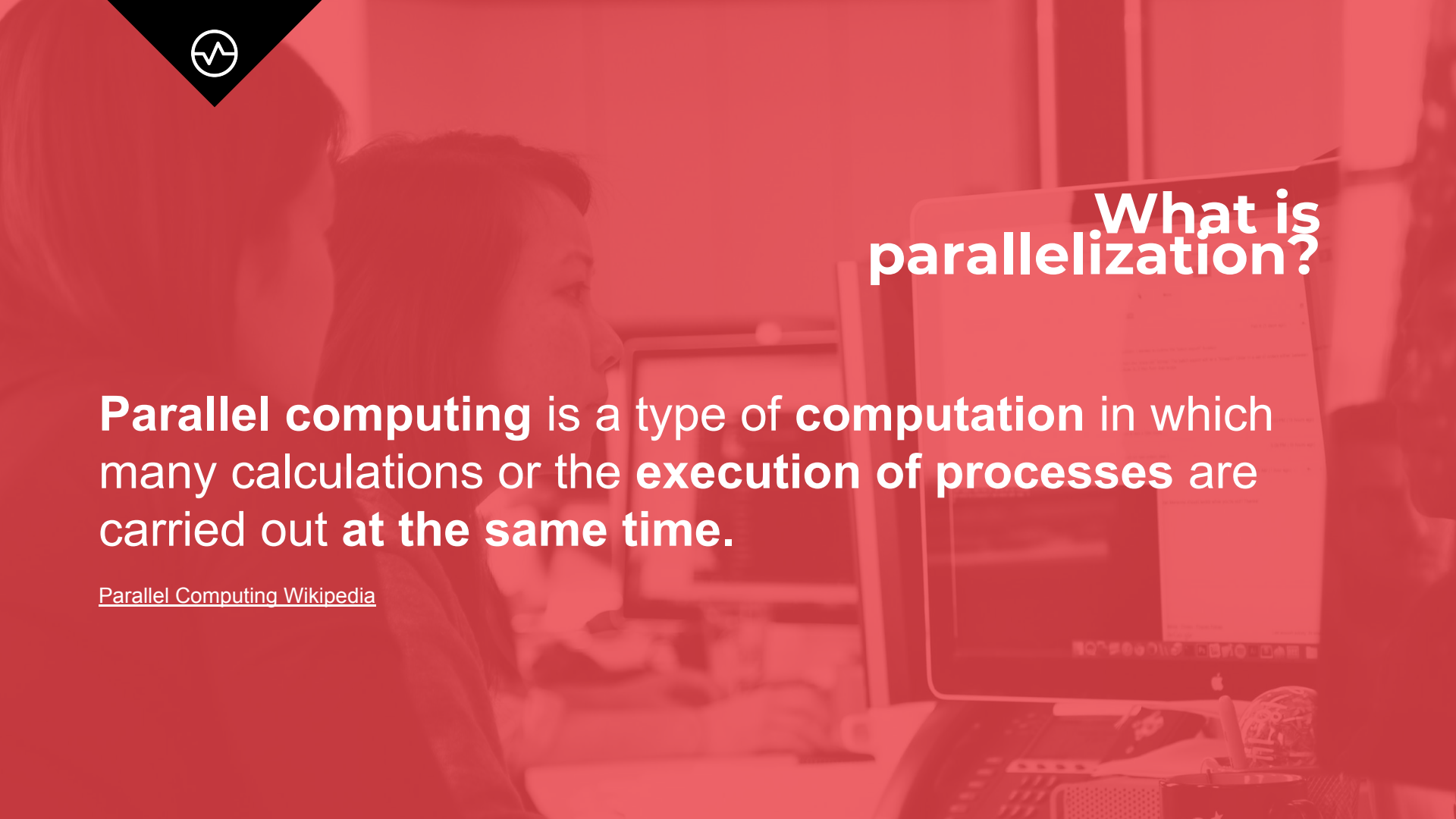
Fault Tolerance: Recap

- What is distributed storage?
- How does distributed storage address fault tolerance?
- What is a NameNode?
- How many nodes may a cluster afford to lose considering the replications it has?

MapReduce

Agenda

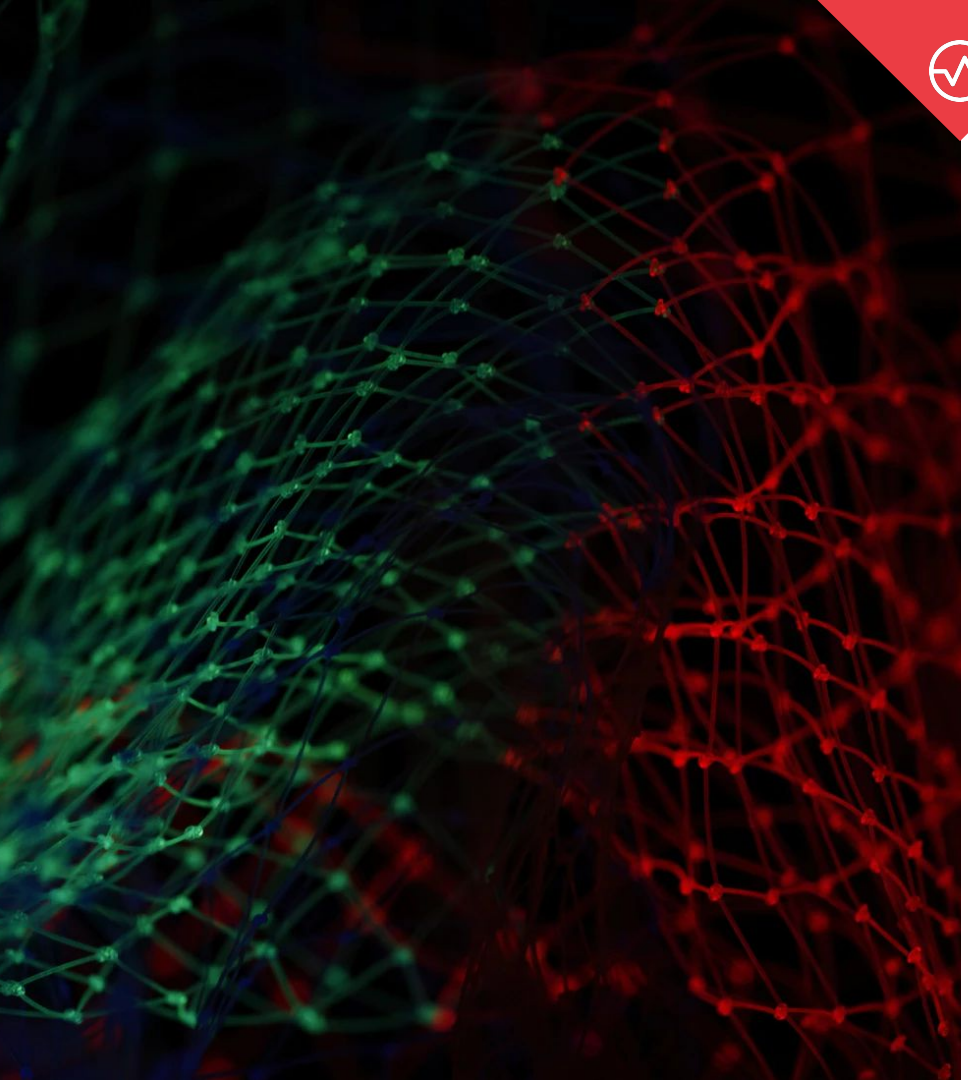
- Fault Tolerance at the Storage Level
- MapReduce
  - **Parallelization**
  - The Framework
  - Iterations and MapReduce
- Fault Tolerance at the Application Level (Spark)



What is parallelization?

Parallel computing is a type of **computation** in which many calculations or the **execution of processes** are carried out **at the same time.**

Parallel Computing Wikipedia

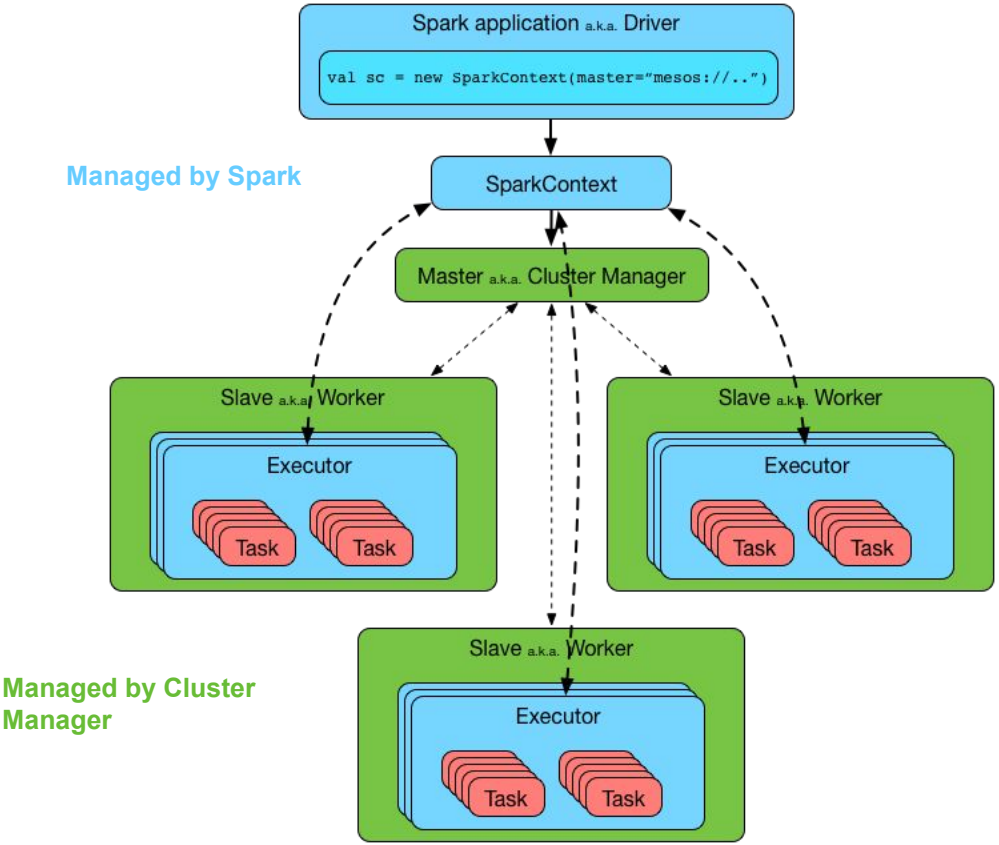


Distributed Computing

A **distributed system** is a system whose components are located on **different networked computers**, which communicate and coordinate their actions by **passing messages** to each other.

Distributed Storage Wikipedia

Remember…


Spark application a.k.a. Driver
val sc = new SparkContext(master="mesos://..")
Managed by Spark
SparkContext
Master a.k.a. Cluster Manager
Slave a.k.a. Worker
Executor
Task
Task
Slave a.k.a. Worker
Executor
Task
Task
Slave a.k.a. Worker
Executor
Task
Task
Managed by Cluster Manager

How to Parallelize?

General Parallelization Steps

Say you have to sum the values in an array, but it **does not fit in memory**.

1+2+3+4+5+6 = ?

Since you can do the same simple operation (+) in separate computers, this is an **embarrassingly parallel task.**

How to Parallelize?

General Parallelization Steps

- **Split** the activity into simple parts.

1+2+3+4+5+6 => (1+2) + (3+4) + (5+6)

How to Parallelize?

General Parallelization Steps

- **Split** the activity into simple parts.

  1+2+3+4+5+6 => (1+2) + (3+4) + (5+6)

- **Compute** each part in multiple devices.

  (1+2) + (3+4) + (5+6) => (3) + (7) + (11)

How to Parallelize?

General Parallelization Steps

- **Split** the activity into simple parts.

  1+2+3+4+5+6 => (1+2) + (3+4) + (5+6)

- **Compute** each part in multiple devices.

  (1+2) + (3+4) + (5+6) => (3) + (7) + (11)

- **Join** the work and coordinate the communication of those devices.

  (3) + (7) + (11) => 3+7+11

Example: How to Parallelize Word Count?

Text Processing

1) **Split** the text into words.
2) **Compute** the frequency of each word.
3) **Join** the frequencies from all devices.

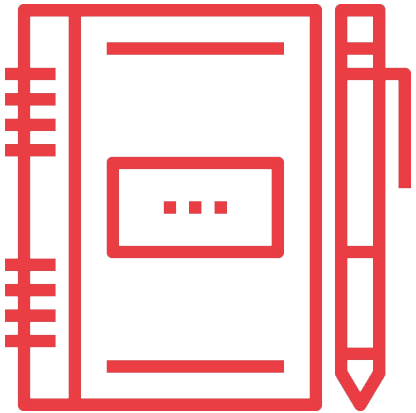
How to Parallelize?

General Parallelization Steps

- **Split** the activity into simple parts.
- **Compute** each part in multiple devices.
- **Join** the work and coordinate the communication of those devices.

Agenda

- Fault Tolerance at the Storage Level
- MapReduce
  - Parallelization
  - **The Framework**
  - Iterations and MapReduce
- Fault Tolerance at the Application Level (Spark)

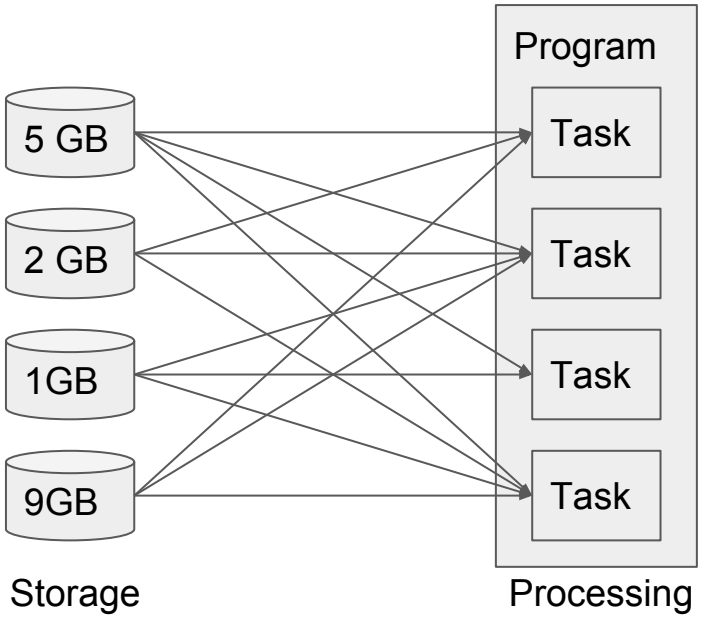


MapReduce Framework

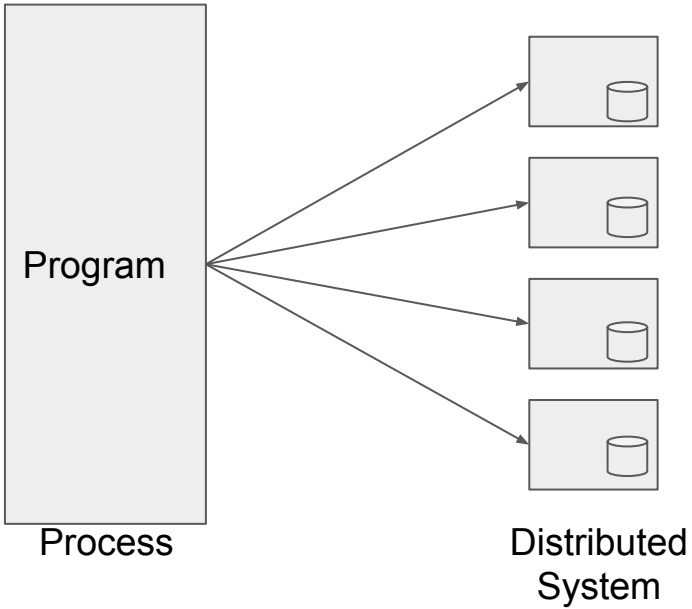
*A simple and powerful interface that enables automatic **parallelization** and **distribution** of **large-scale** computations, combined with an implementation of this interface that achieves high performance on large clusters of **commodity PCs**."*
Dean and Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Google Inc.

"Moving Computation is Cheaper than Moving Data"
Hadoop Team

Move Tasks Instead of Data



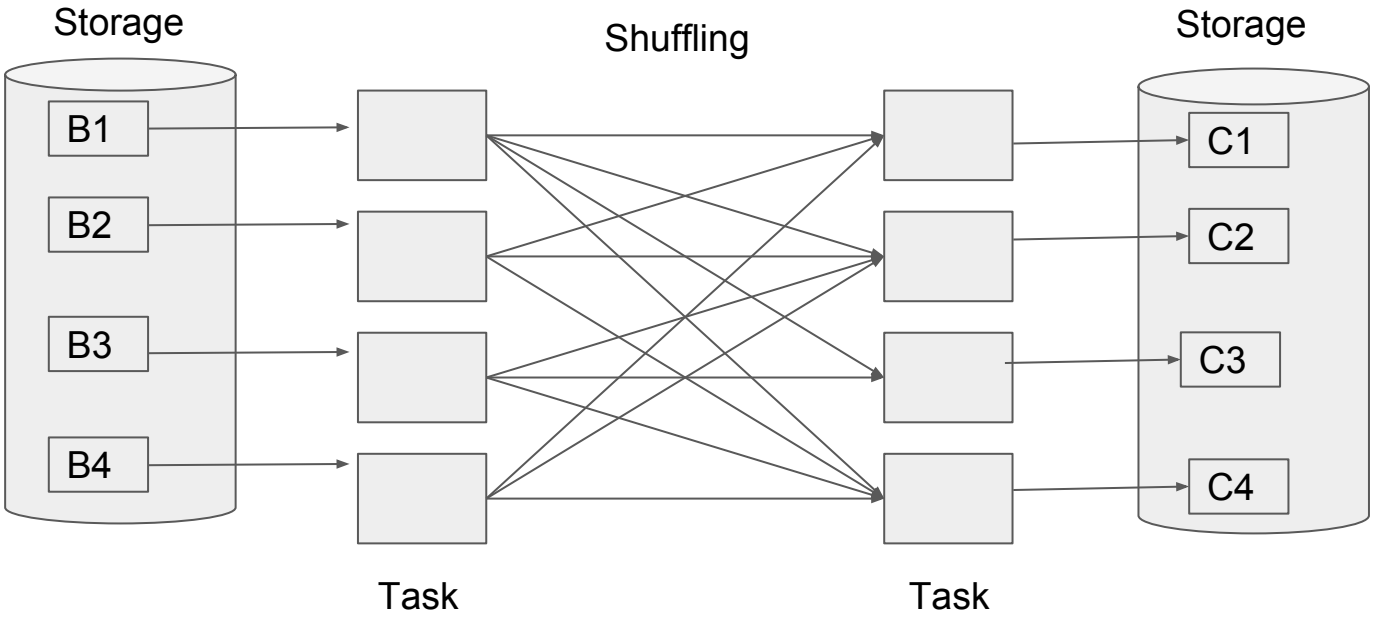
17 to 56 GB being moved



MB or Bytes being moved
(messages and programs only)

MapReduce, What Does It Consist Of?

What Is Map and What Is Reduce?

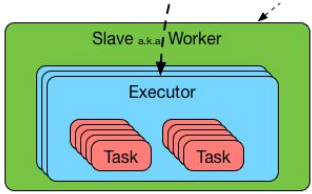


Primitive Map:

- Perform the same action on every item of the Dataset.
- Embarrassingly parallel.

Primitive Reduce:

- Combine the dataset.
- Gather the results.
- Recieves Ordered data.



Parallelization Framework with Word Count

A Parallelization Without Coordinated Efforts

Text processing, use word frequency to describe a text:

1) Ingest text into distributed storage. ← Distribute (data)

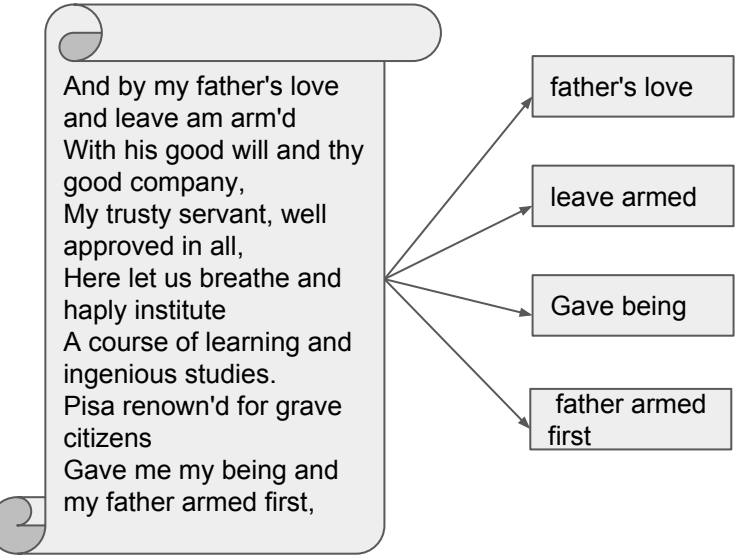
2) Split text into words. ← (computing)

3) Sort the words globally. ← Join (communication and networking)

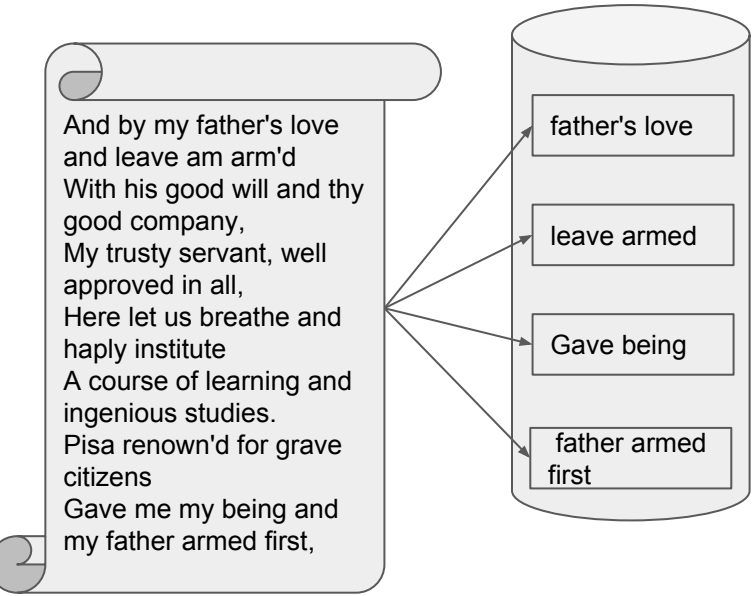
4) Compute each word. ← (computing)

5) Store the vector of repetition for future comparisons. ← Distribute (data)

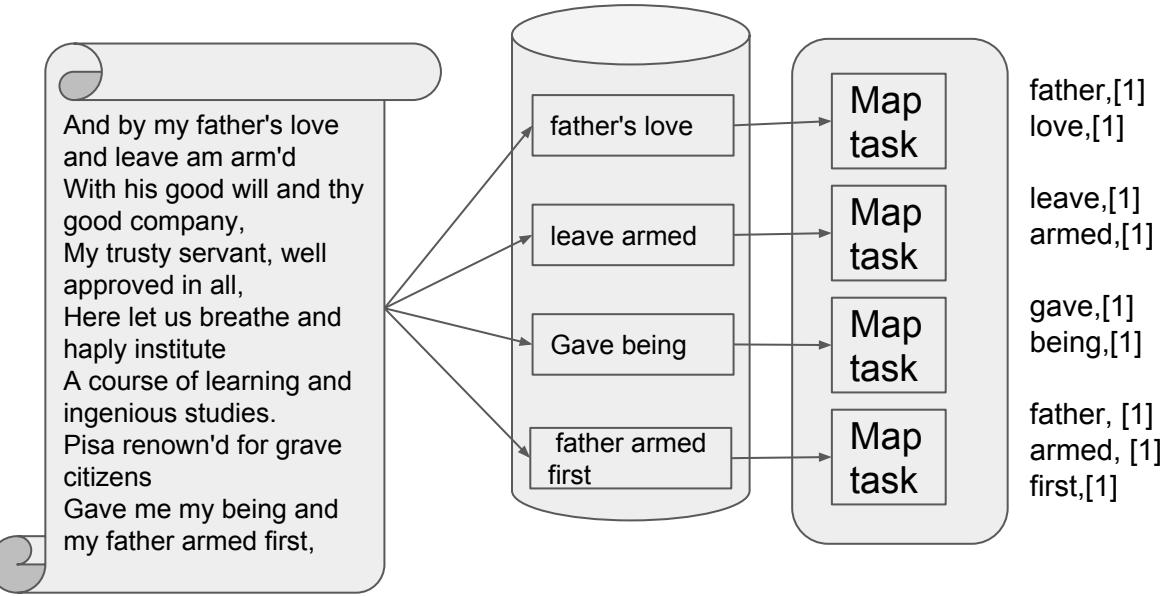
Step by Step Example



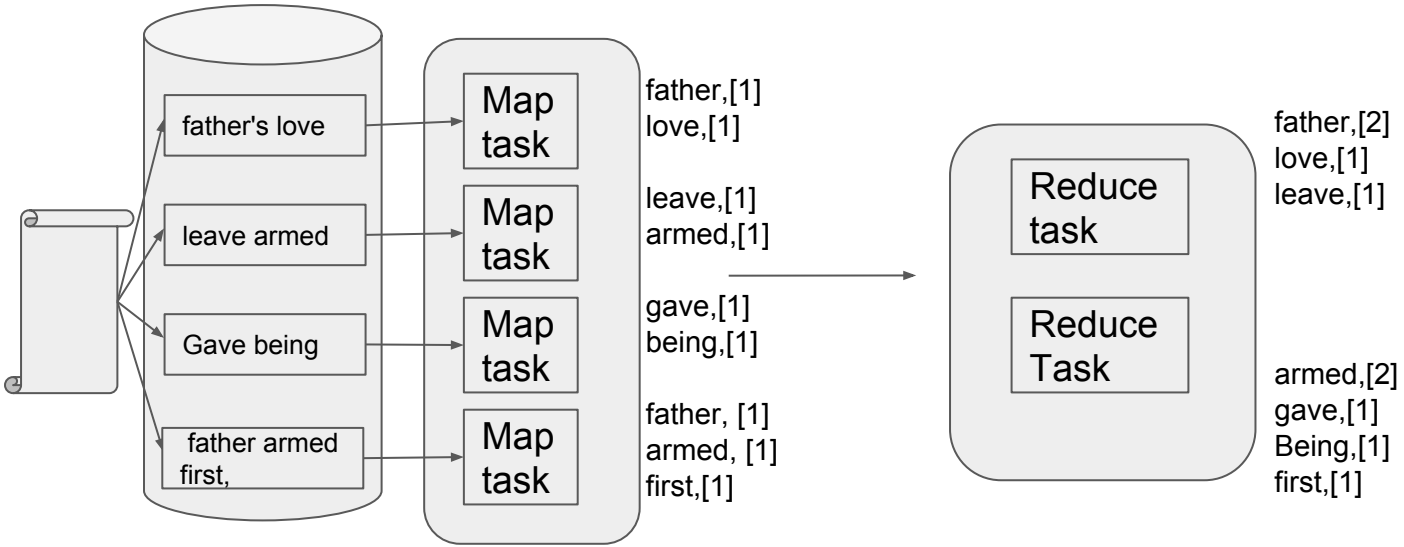
Step by Step Example



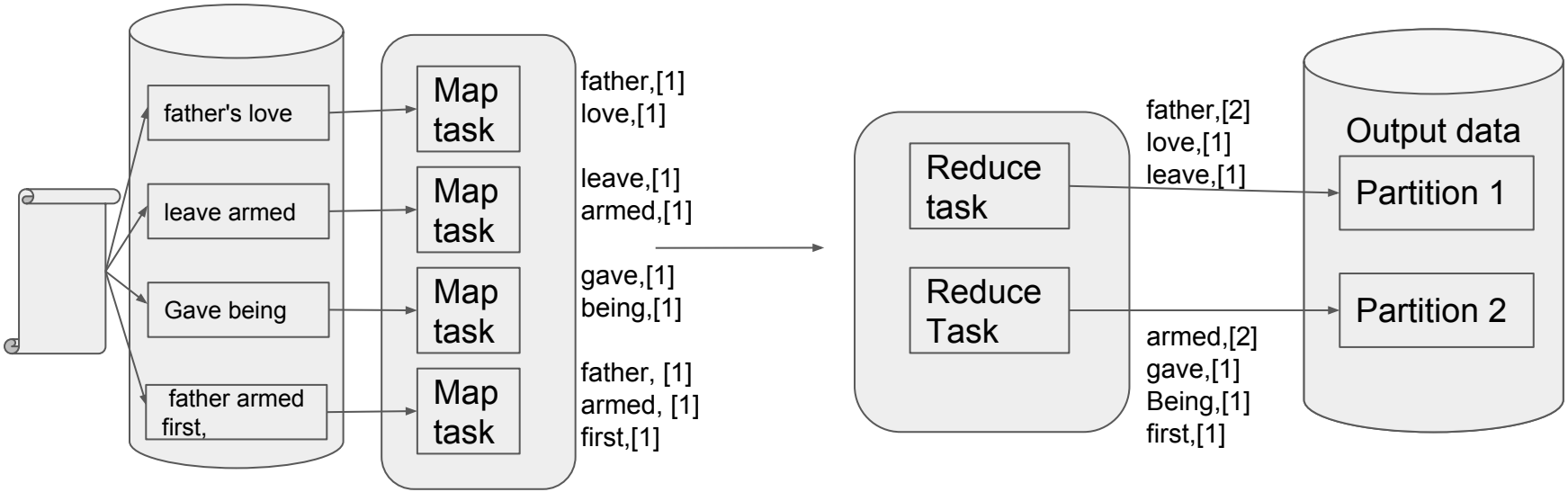
Step by Step Example



Step by Step Example



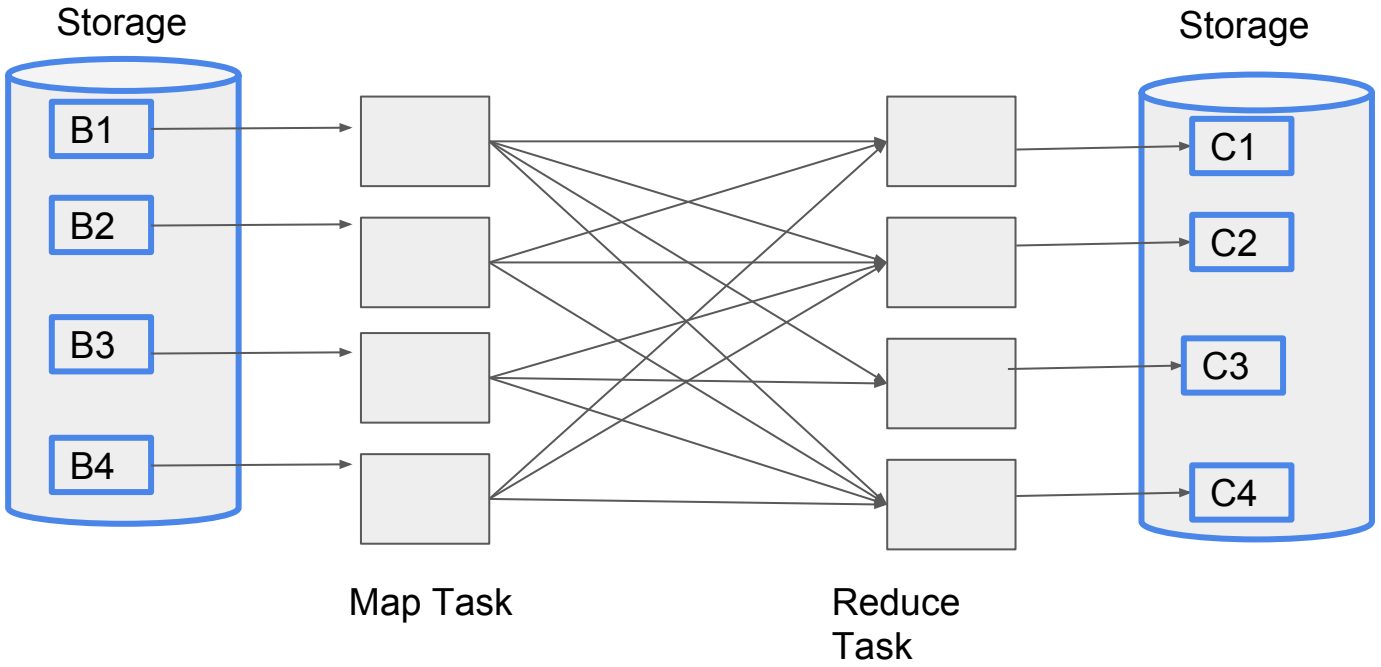
Step by Step Example



The Genius of Map-Reduce

Hadoop Provides

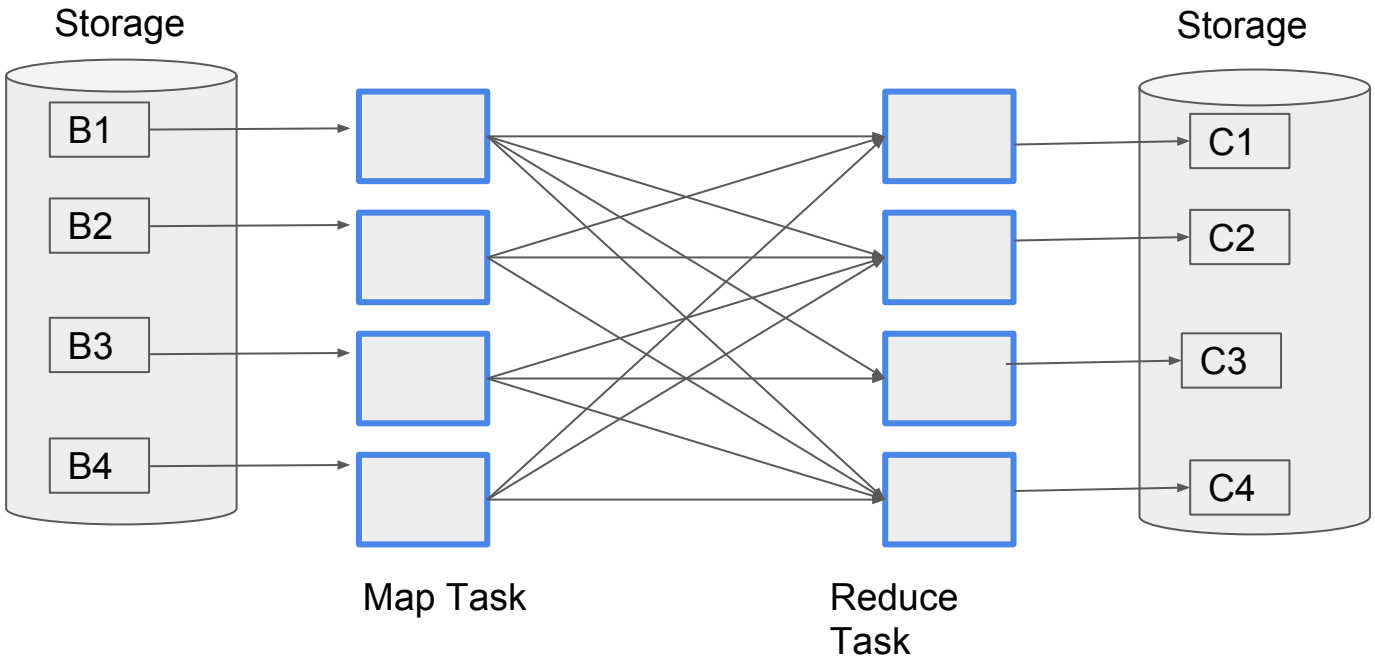
- Integrates distributed storage to the computing framework.



The Genius of Map-Reduce

Hadoop Provides

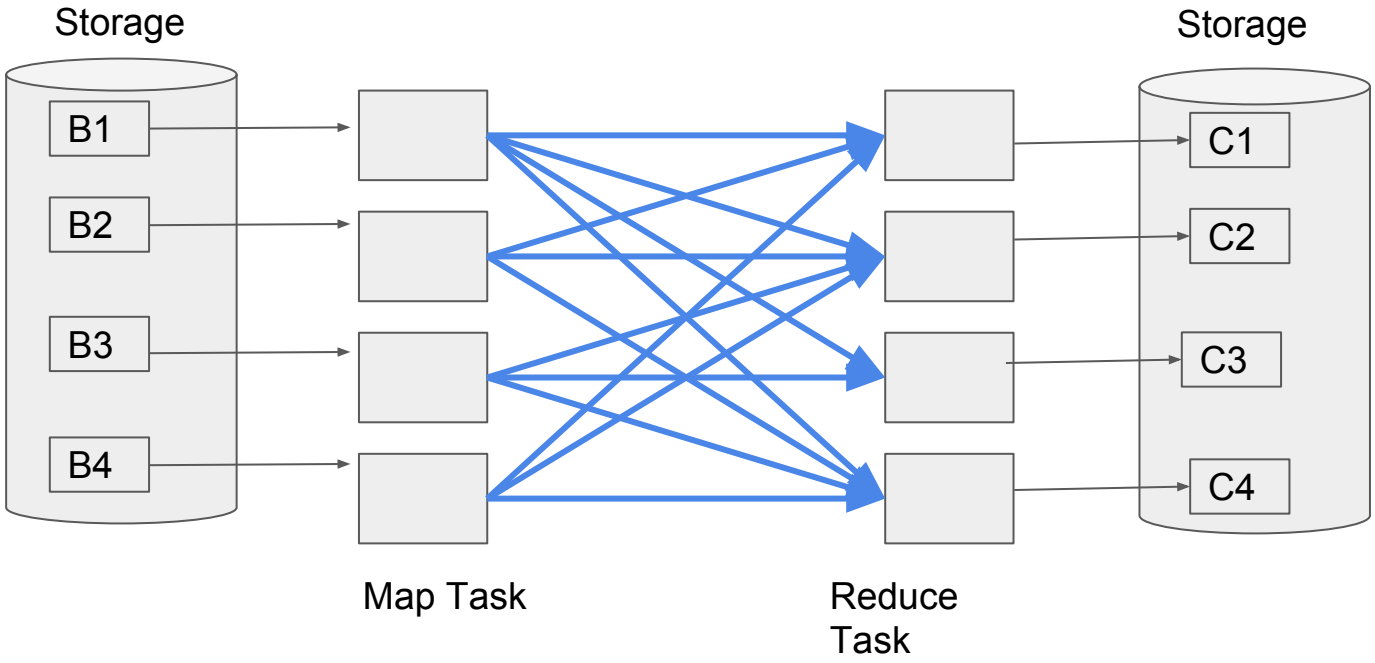
- Integrates distributed storage to the computing framework.
- Sends tasks to the data, not data to the tasks (reduce network overhead).



The Genius of Map-Reduce

Hadoop Provides:

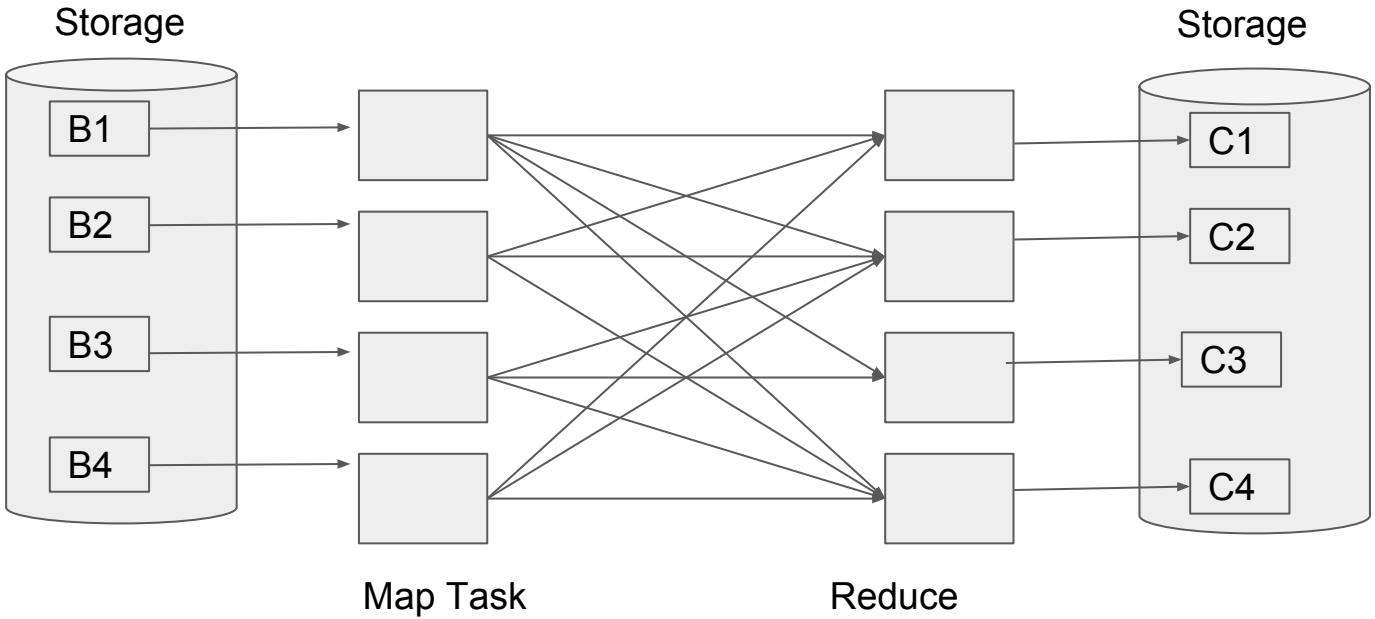
- Integrates distributed storage to the computing framework.
- Sends tasks to the data, not data to the tasks (reduce network overhead).
- Is a high-level parallel programming abstraction (you don't have to coordinate the tasks).



The Genius of Map-Reduce

A Parallelization Without Coordinated Efforts

- Integrates distributed storage to the computing framework.
- Sends tasks to the data, not data to the tasks (reduce network overhead).
- Is a high level parallel programming abstraction (you don't have to coordinate the tasks).
- Seems to have close to linear scalability*****.

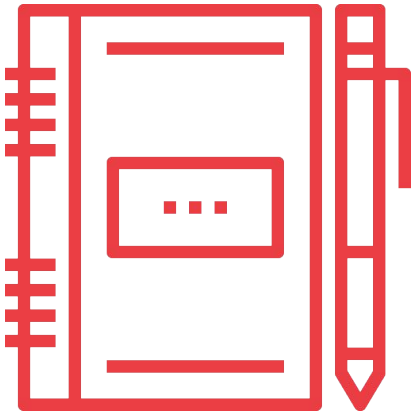


Map Reduce: Recap

- What is parallelization?
- What does the concept “embarrassingly parallel” mean?
- What is MapReduce?
- What is the benefit of the MapReduce -hadoop- framework?

Agenda

- Fault Tolerance at the Storage Level
- MapReduce
  - Parallelization
  - The Framework
  - **Iterations and MapReduce**
- Fault Tolerance at the Application Level (Spark)



Iterations and MR

The MapReduce paradigm involves **reading, processing and writing** if you want the outcome to be useful for further processing.

This is **ideal for batch processing,** which is the case for most of the initial business requirements where you make historical data available.

But what about iterative work?

Imagine you have a matrix and want to multiply it by other matrices.

$$\begin{bmatrix} 21, 22 \\ 23, 24 \end{bmatrix} \times \begin{bmatrix} 11, 12 \\ 13, 14 \end{bmatrix} \times \begin{bmatrix} 0.4, 0.3 \\ 0.2, 0.1 \end{bmatrix} \times \begin{bmatrix} 4, 3 \\ 2, 1 \end{bmatrix}$$

Sequential Analysis

Math

Imagine you have a matrix and want to multiply it by four other matrices

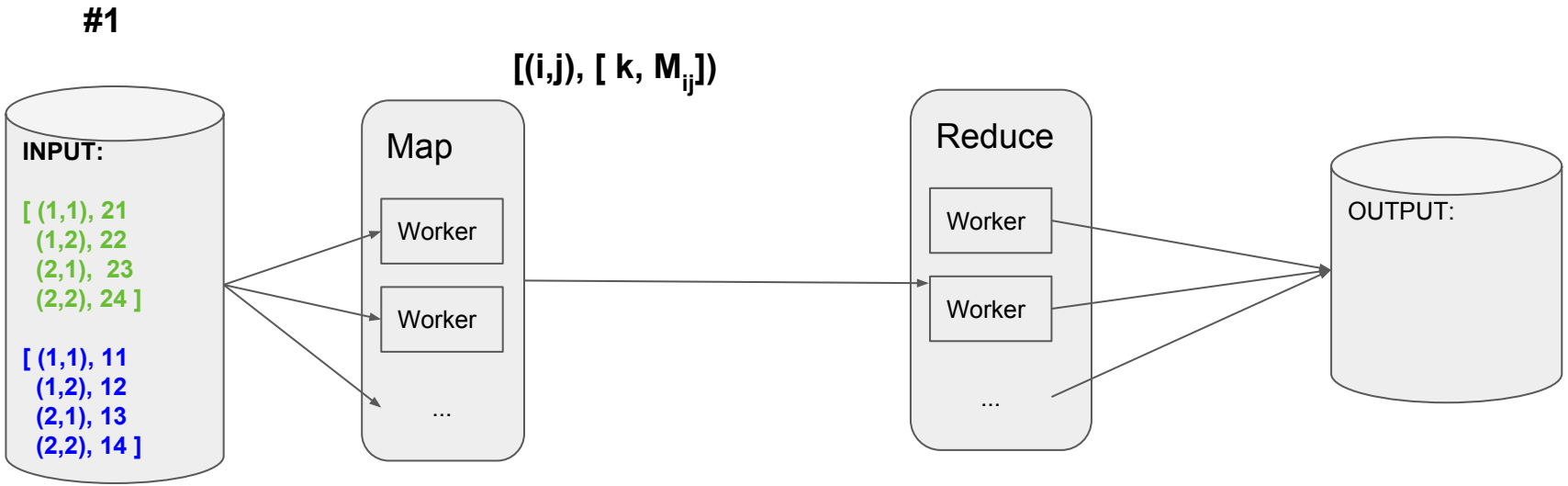
$$\begin{bmatrix} 21, 22 \\ 23, 24 \end{bmatrix} \times \begin{bmatrix} 11, 12 \\ 13, 14 \end{bmatrix} \times \begin{bmatrix} 0.4, 0.3 \\ 0.2, 0.1 \end{bmatrix} \times \begin{bmatrix} 4, 3 \\ 2, 1 \end{bmatrix}$$

Matrix multiplication refresh:

$$\begin{bmatrix} 21, 22 \\ 23, 24 \end{bmatrix} \times \begin{bmatrix} 11, 12 \\ 13, 14 \end{bmatrix} \Rightarrow (21*11) + (22*13) \Rightarrow \begin{bmatrix} 517, 560 \\ 565, 612 \end{bmatrix}$$

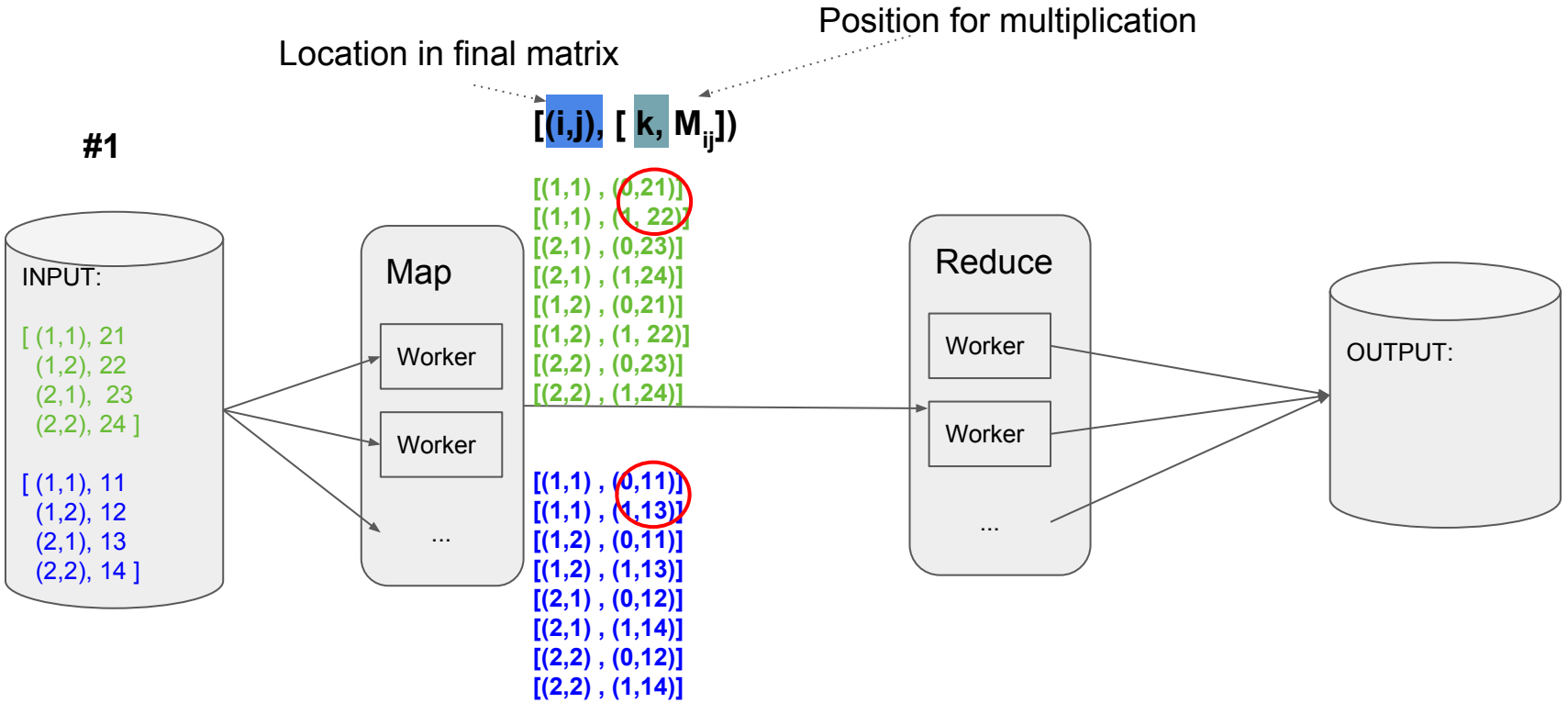
Matrix Multiplication

With MapReduce



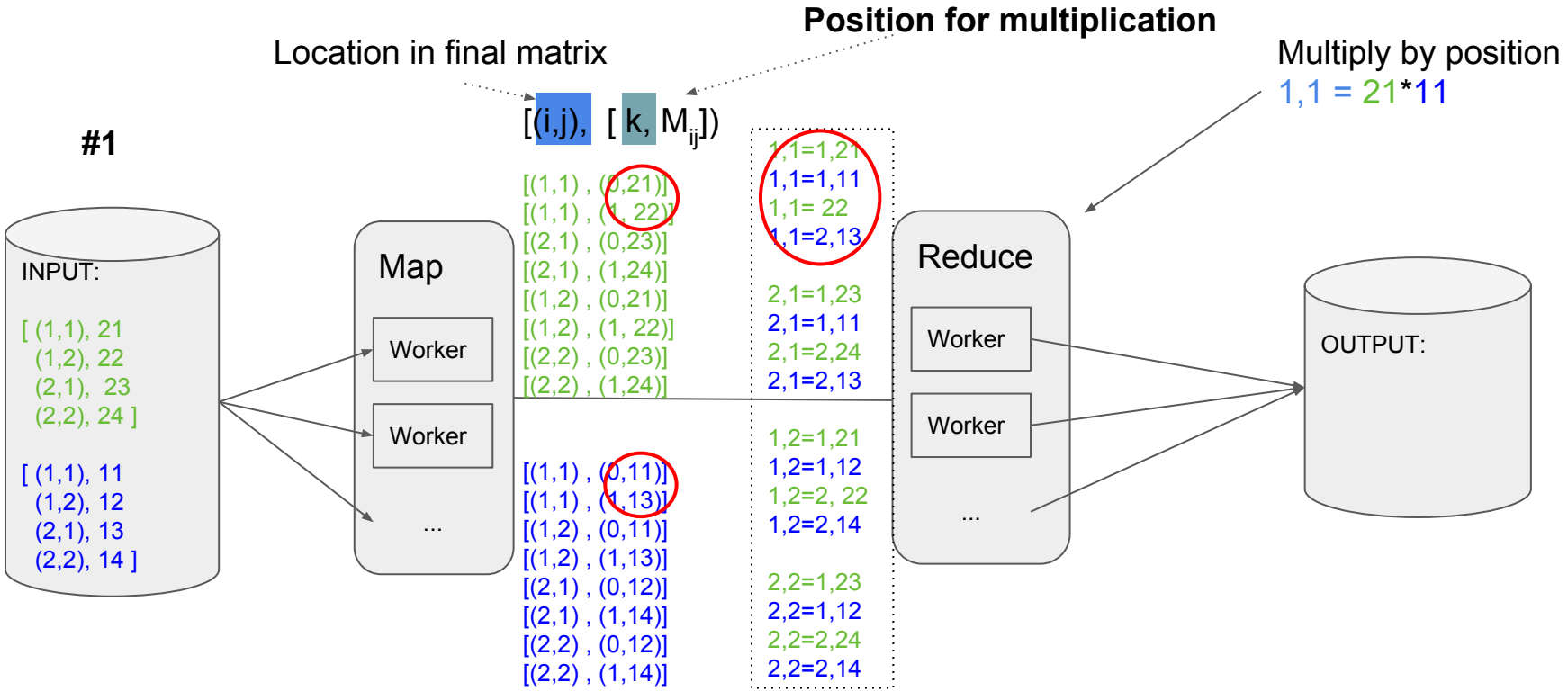
Matrix Multiplication

With MapReduce



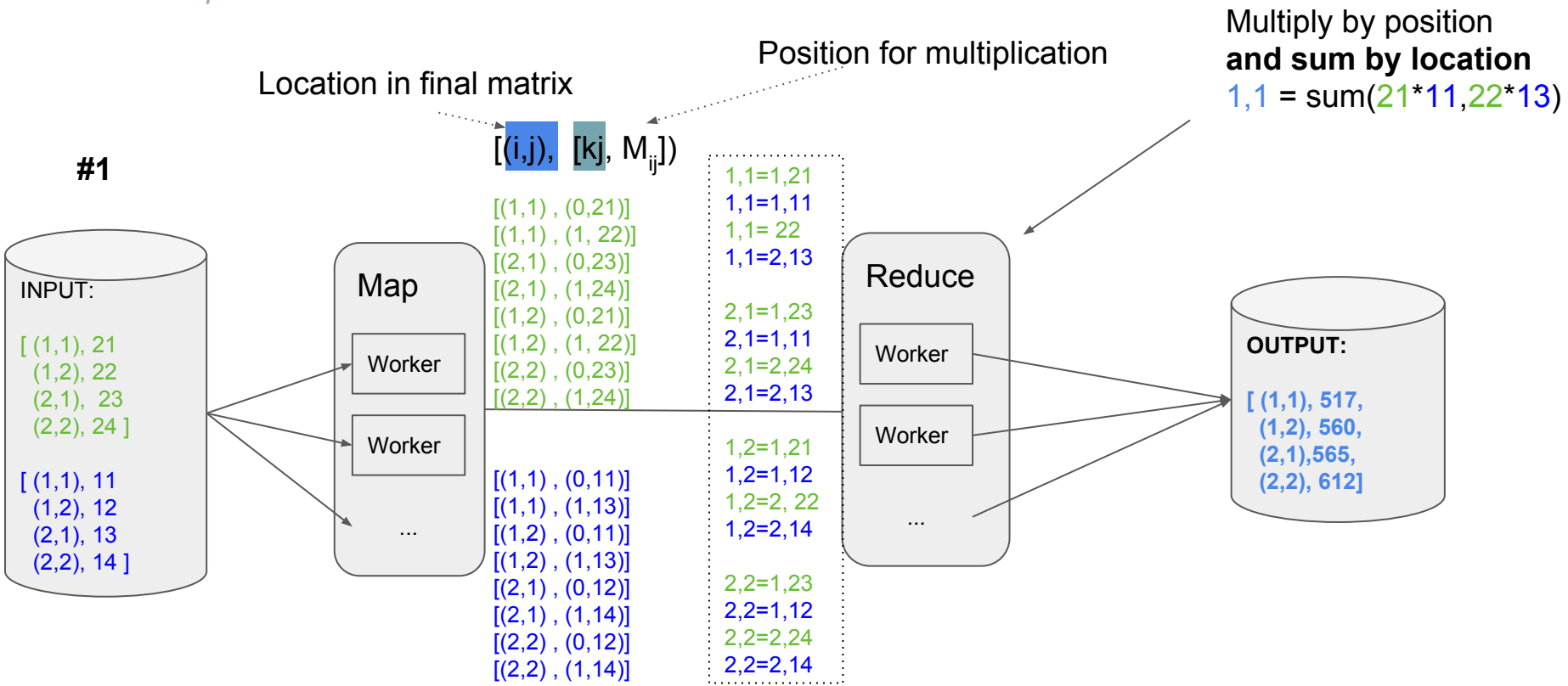
Matrix Multiplications: Reduce Detail

With MapReduce



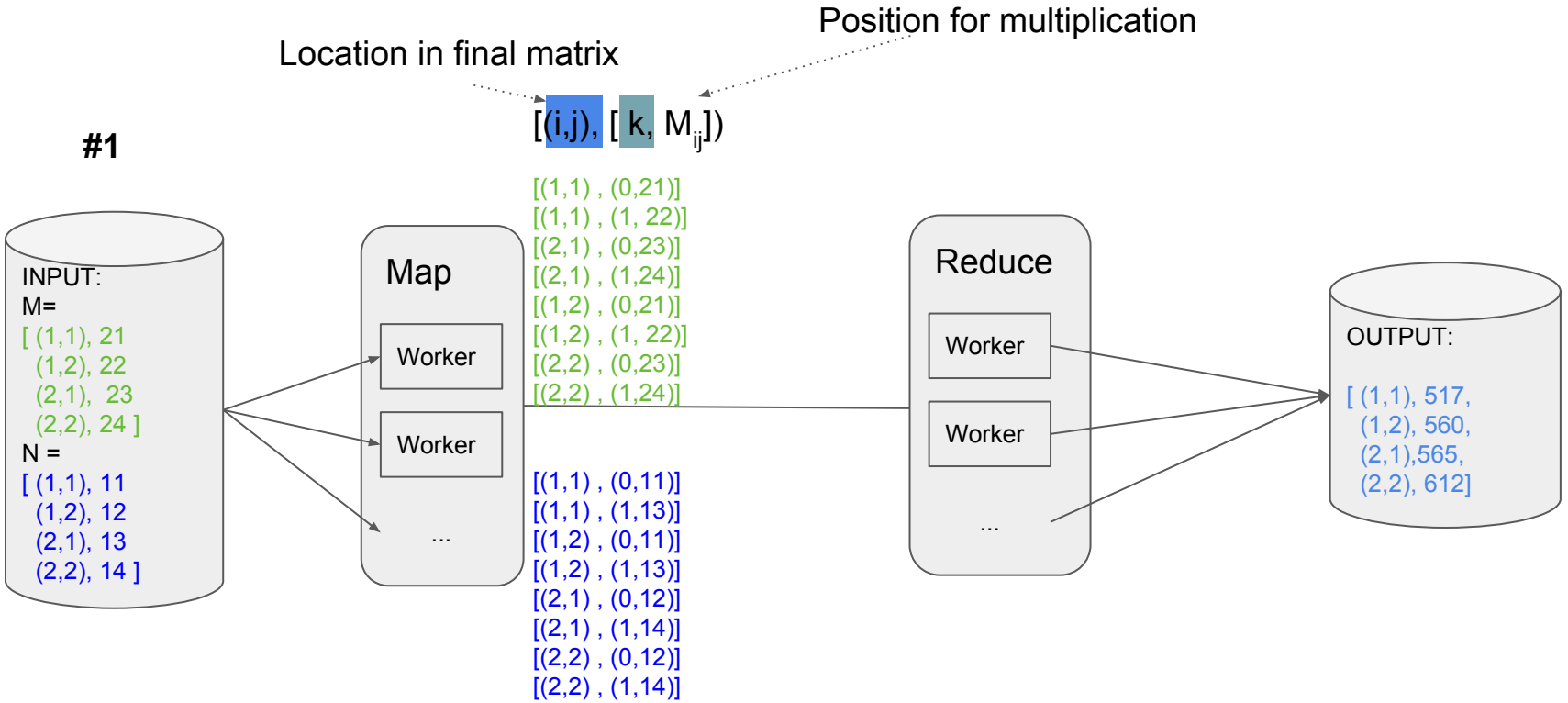
Matrix Multiplications: Reduce Detail

With MapReduce



Matrix Multiplications

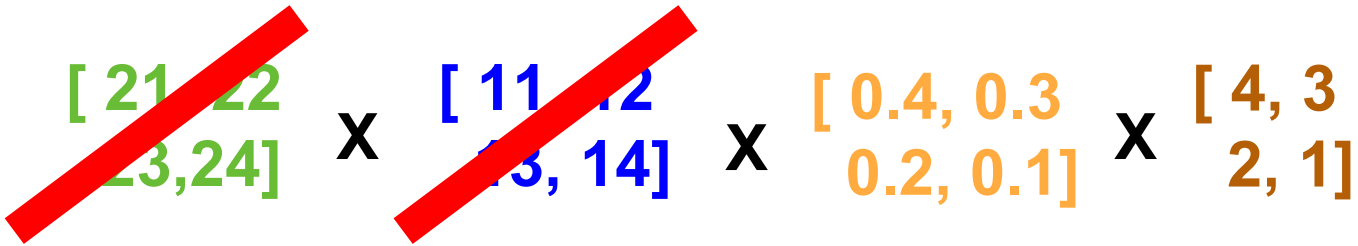
With MapReduce



Sequential Analysis

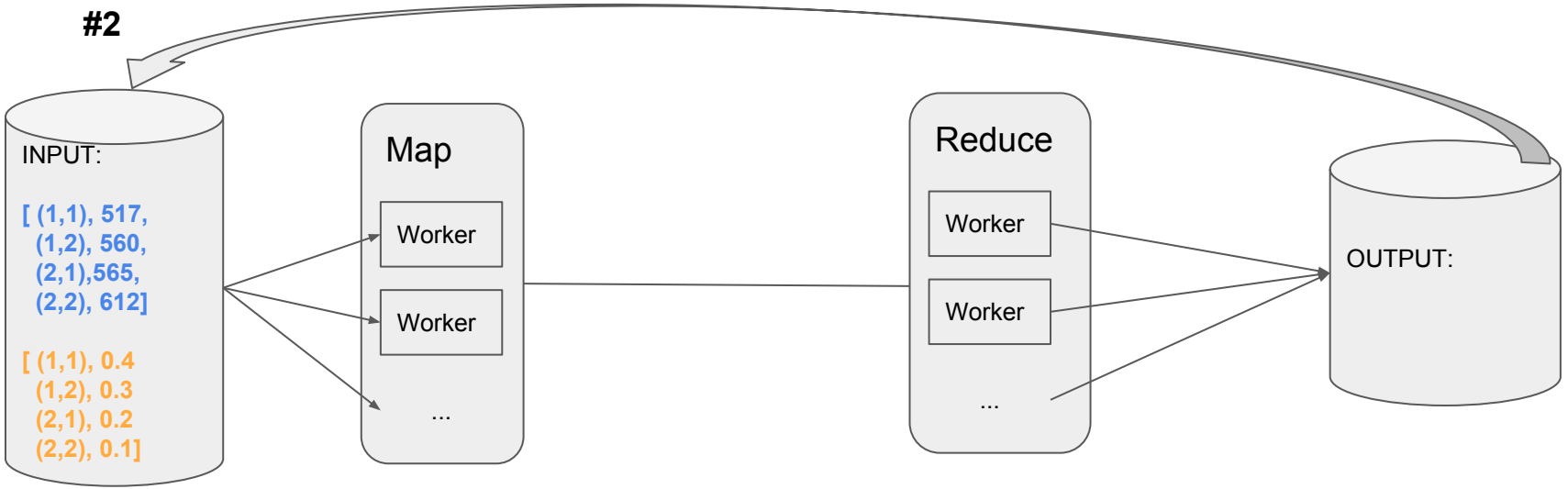
Math

Imagine you have a matrix and want to multiply it by other matrices.



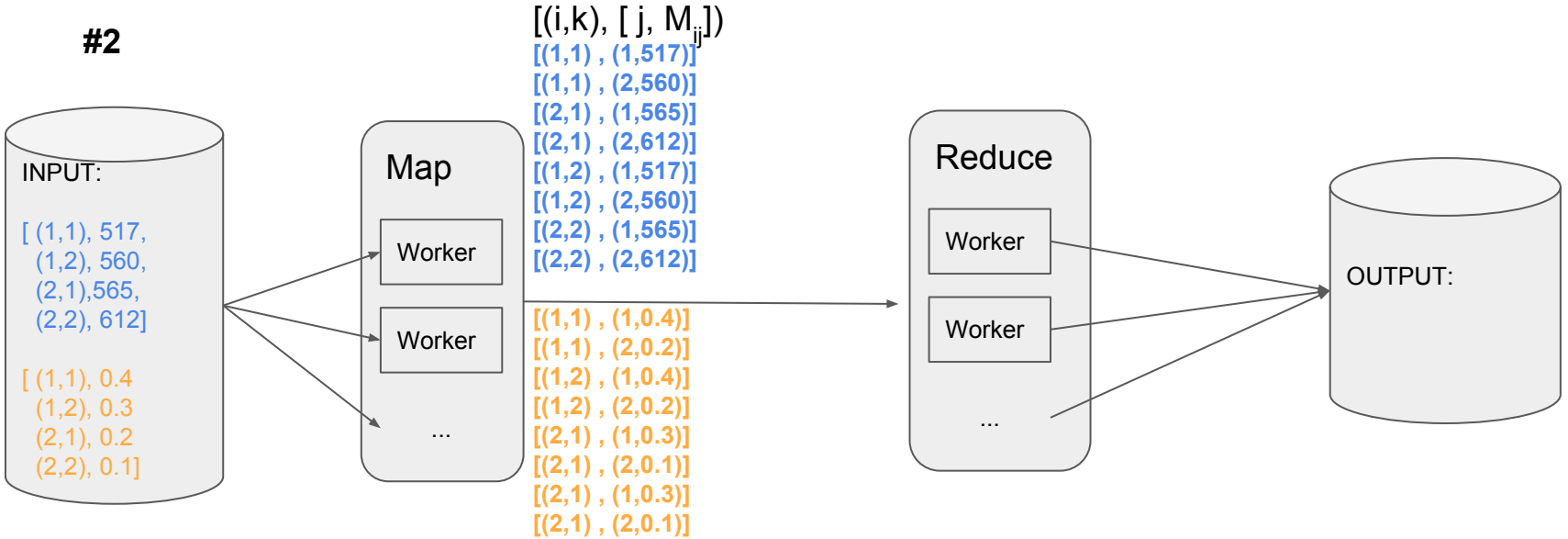
Matrix Multiplications

With MapReduce



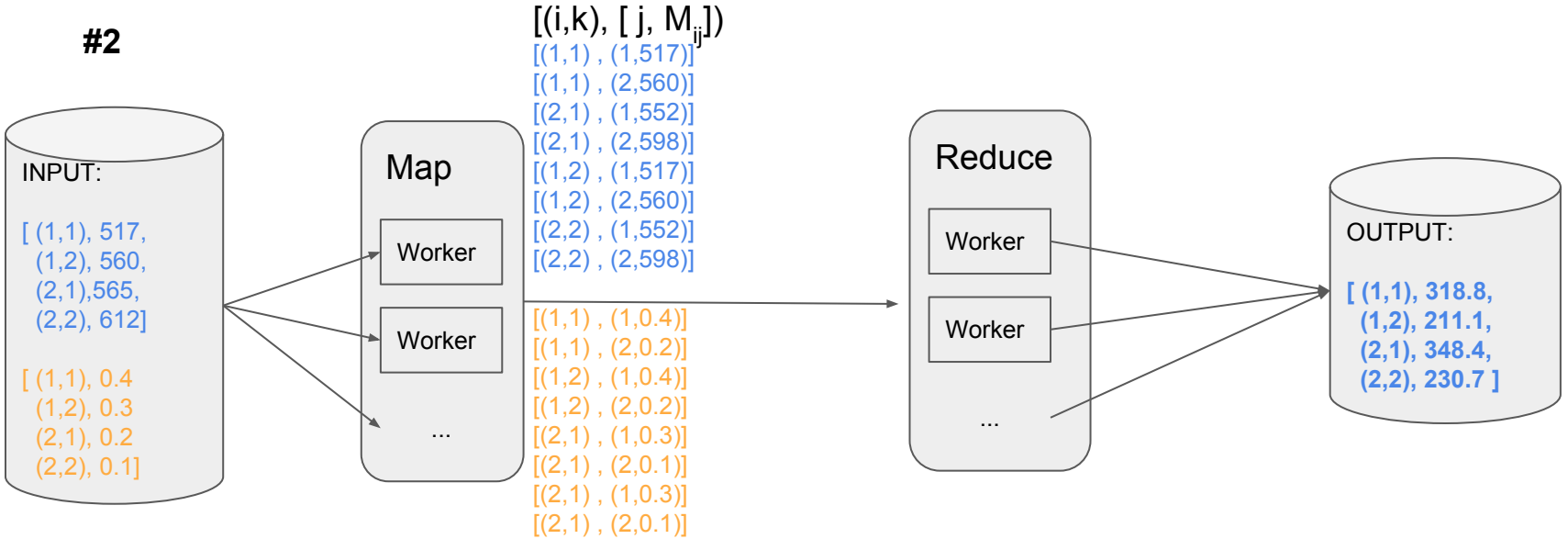
Matrix Multiplications

With MapReduce



Matrix Multiplications

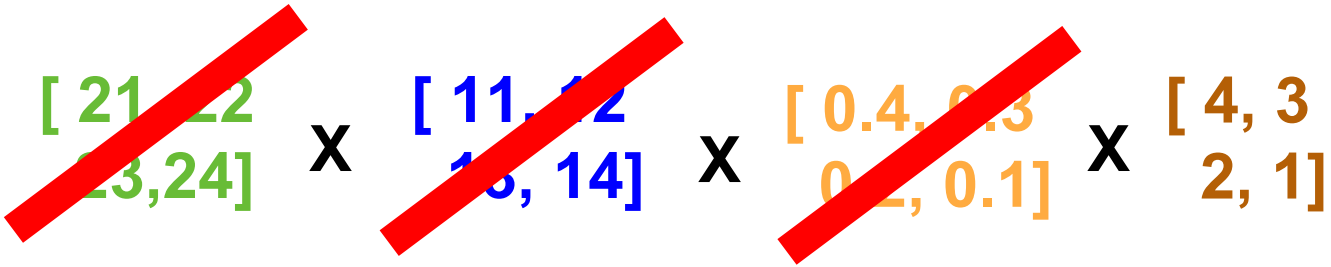
With MapReduce



Sequential Analysis

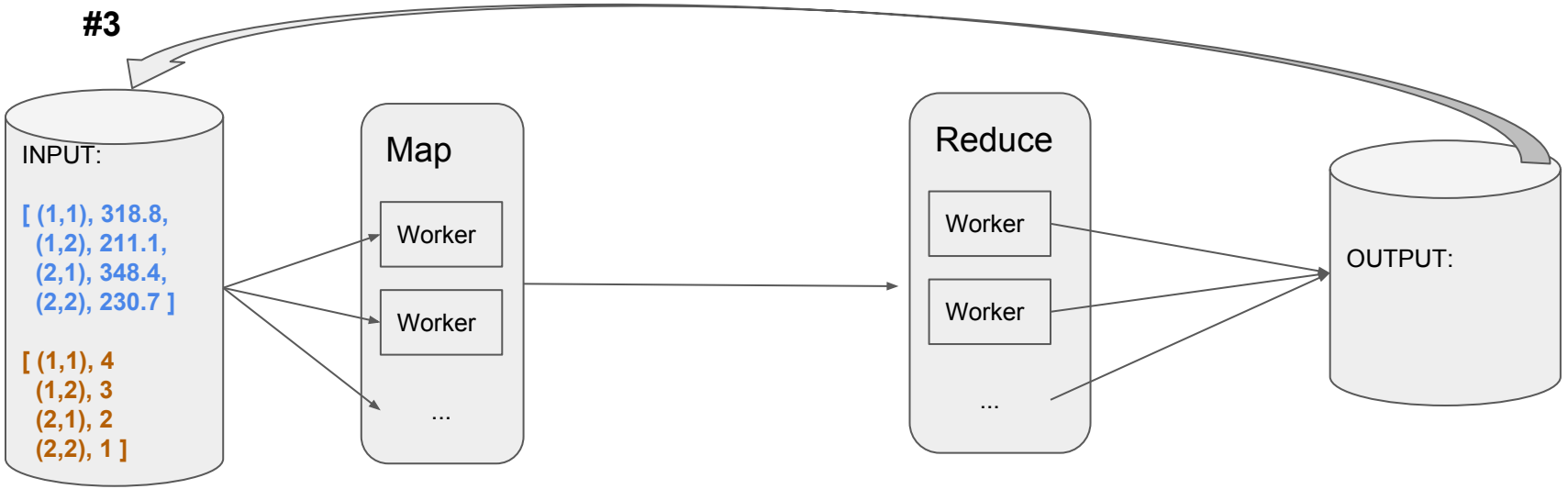
Math

Imagine you have a matrix and want to multiply it by other matrices.



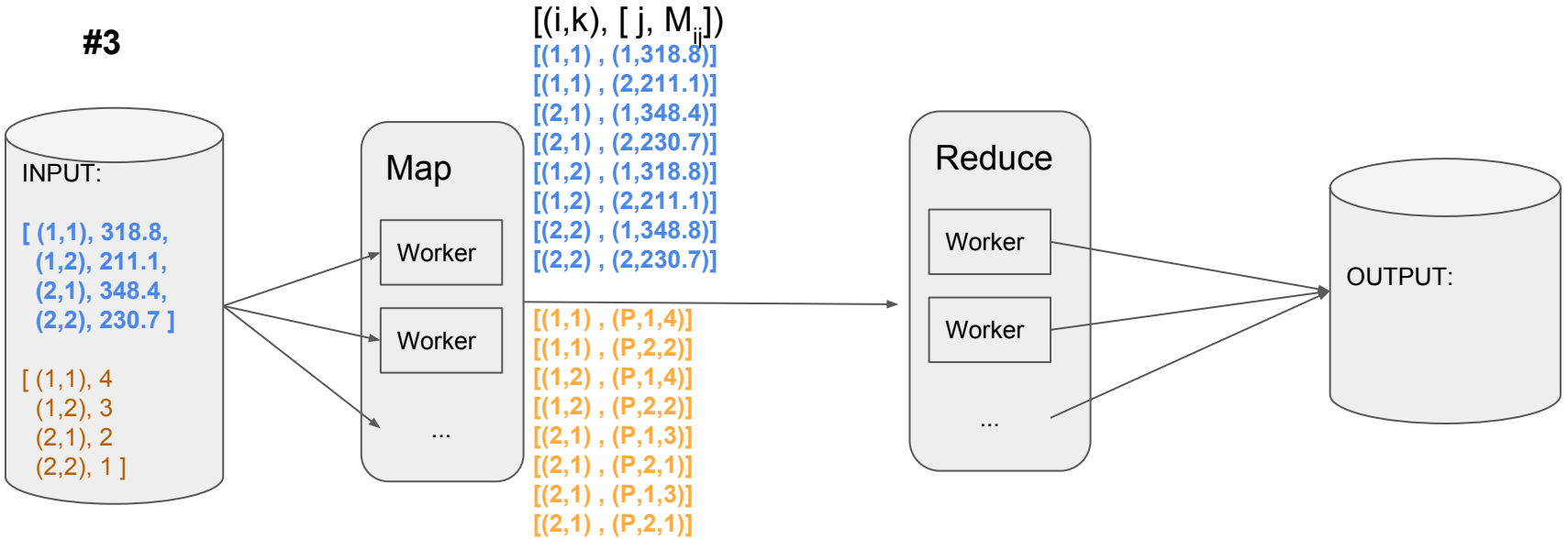
Matrix Multiplications

With MapReduce



Matrix Multiplications

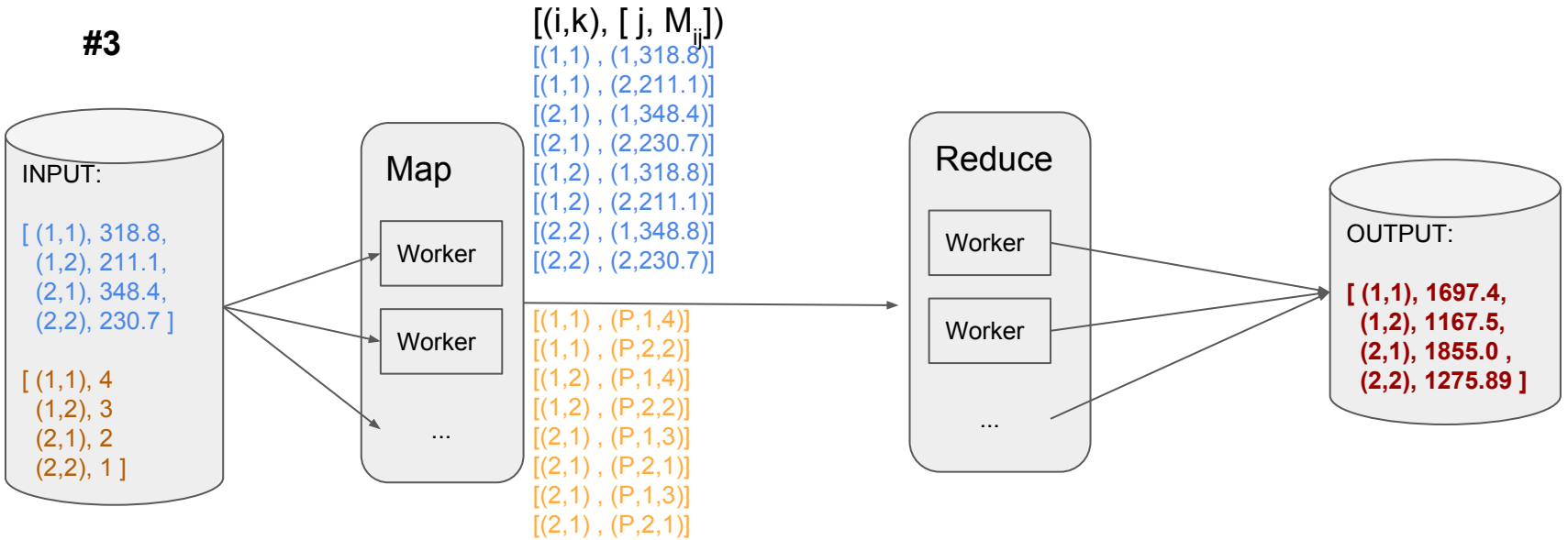
With MapReduce



Matrix Multiplications

With MapReduce

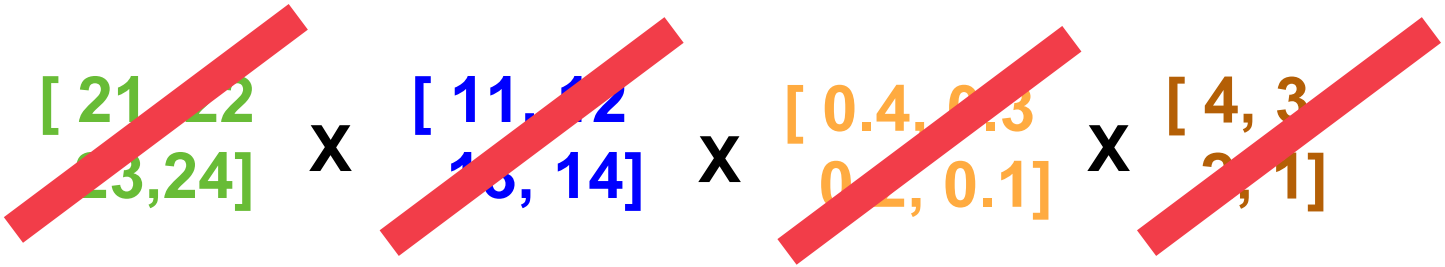
#3



Sequential Analysis

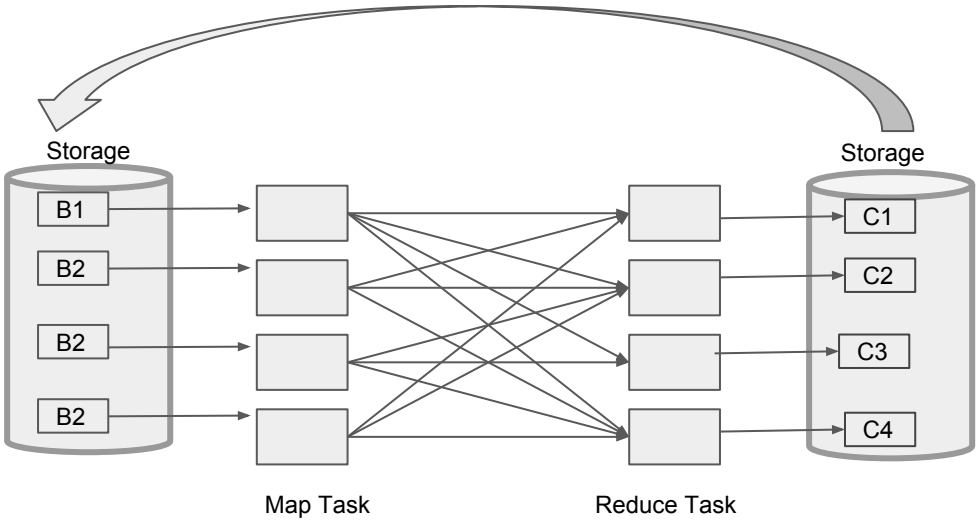
Math

Imagine you have a matrix and want to multiply it by other matrices.



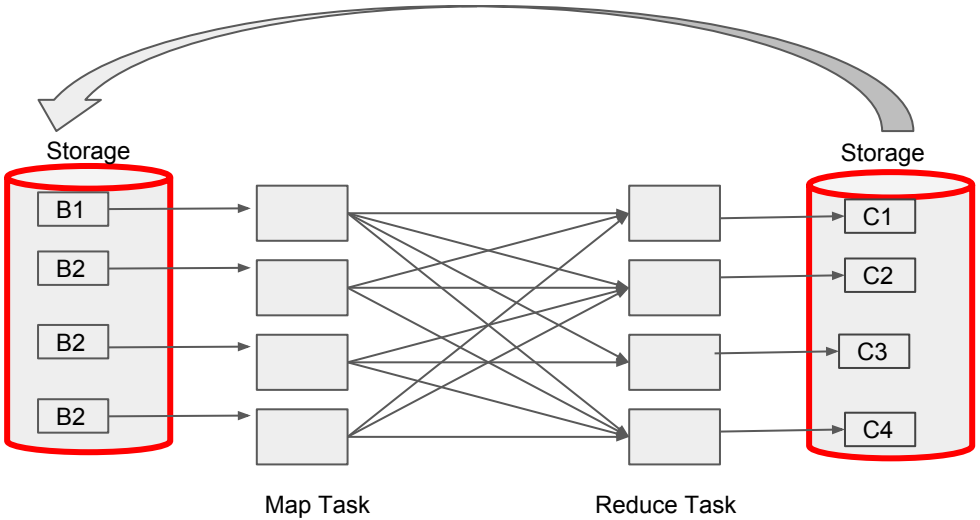
Iterations

With MapReduce



- The information passed is incremental: in every mapreduce you send the original + calculations/annotations.
- We don’t have any memory other than storage.

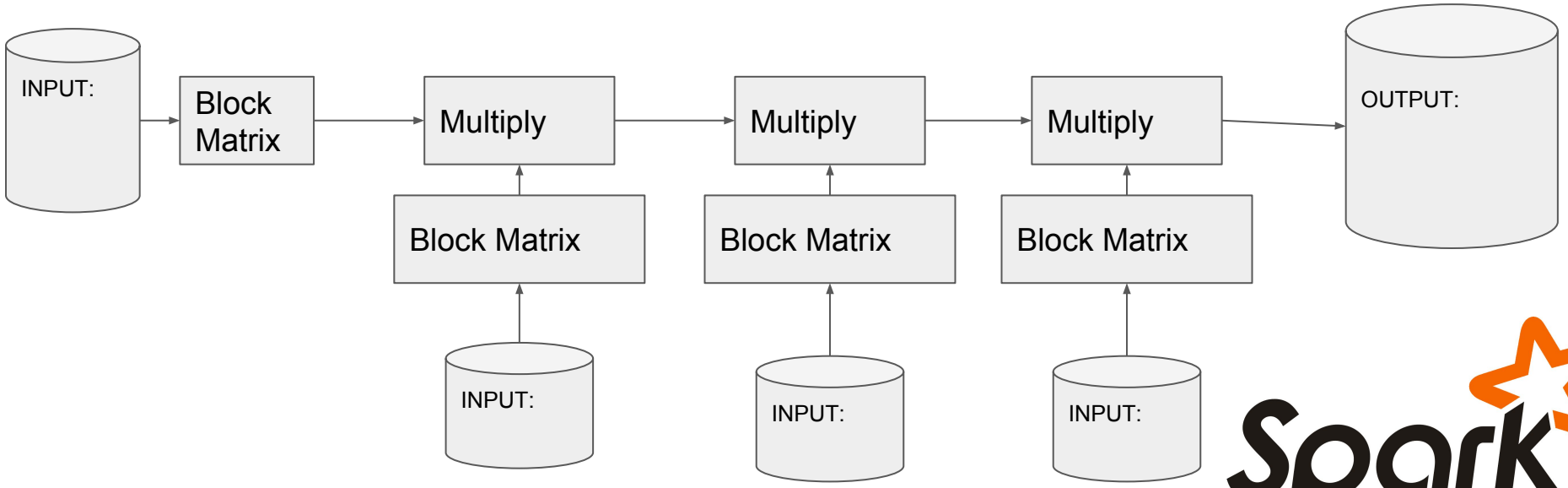
Sparks' approach to iterations



RDD

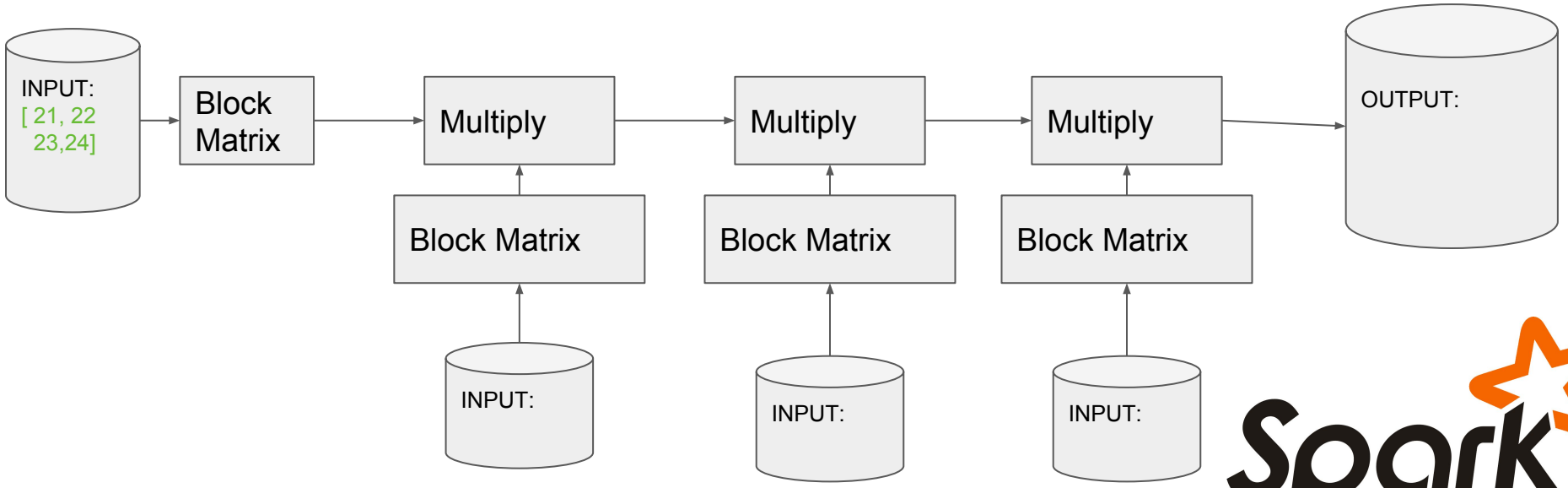
Matrix Multiplication in Spark

Spark is more than MR.



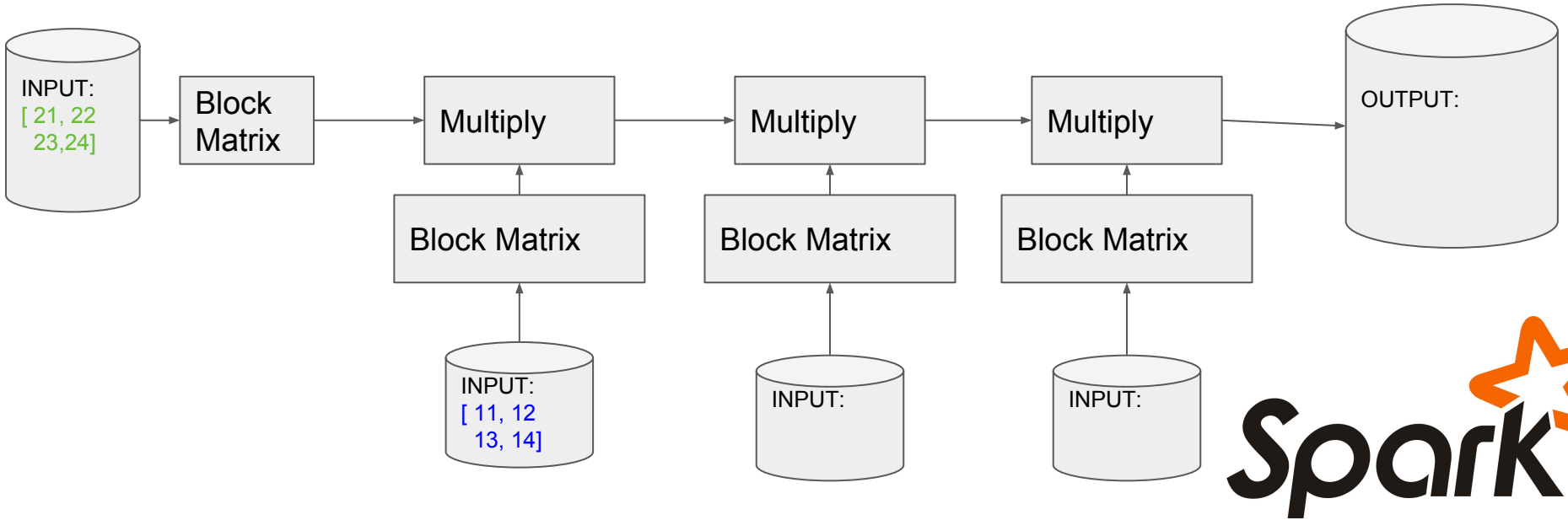
Matrix Multiplication in Spark

Spark is more than MR.



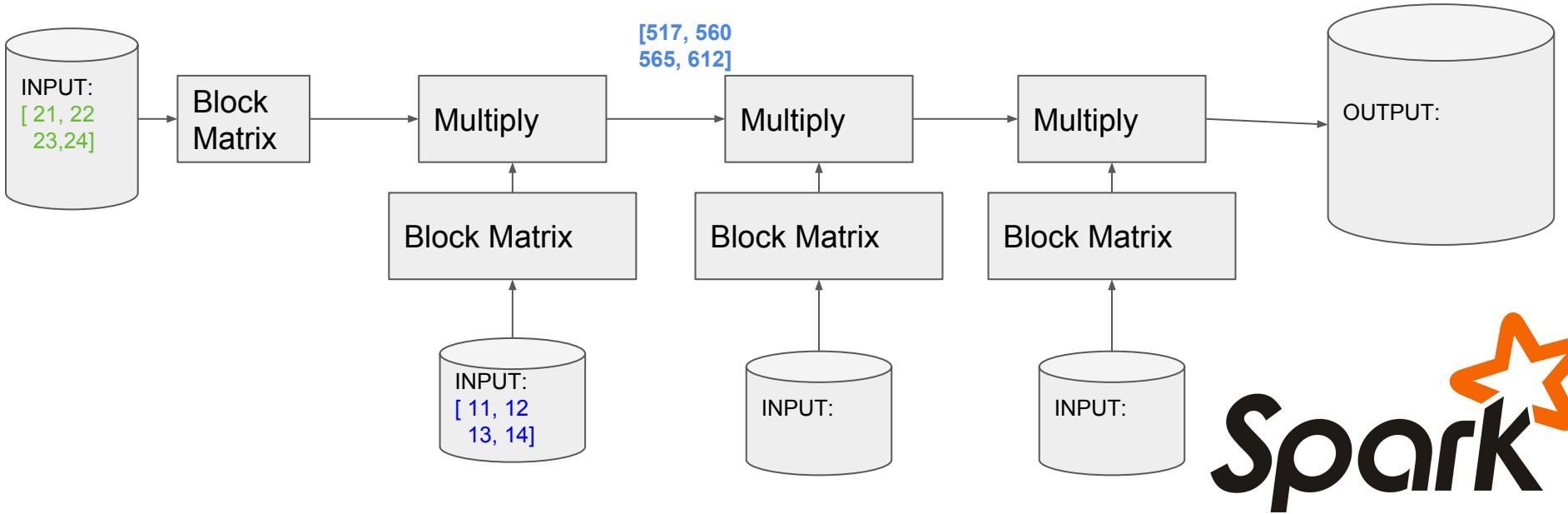
Matrix Multiplication in Spark

Spark is more than MR.



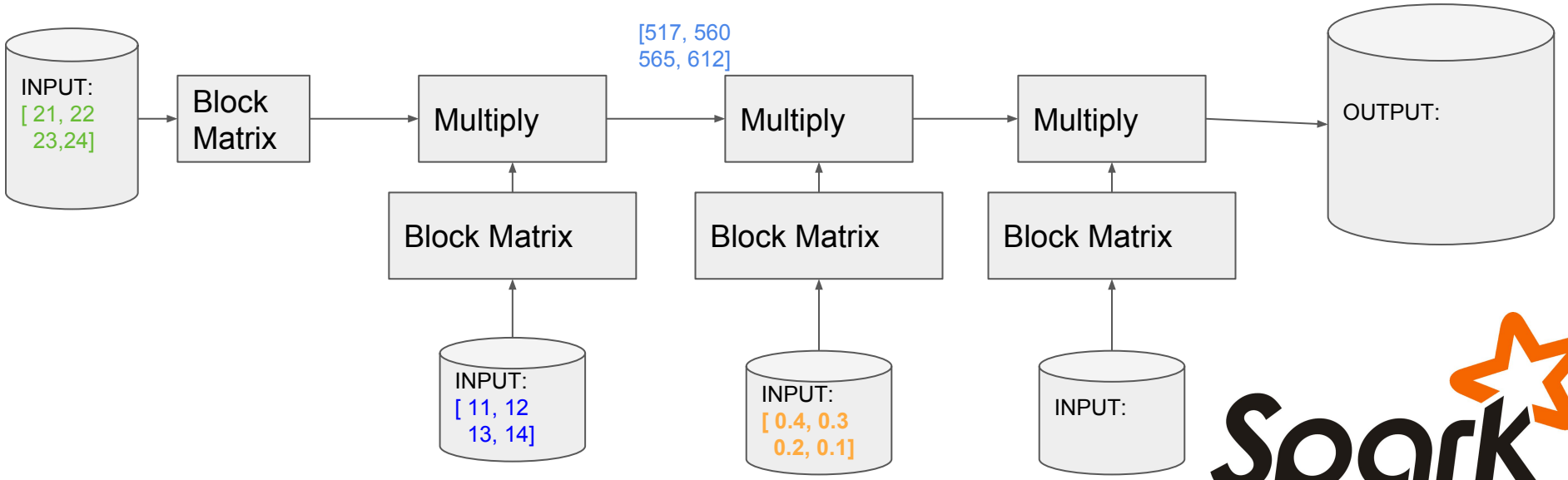
Matrix Multiplication in Spark

Spark is more than MR.



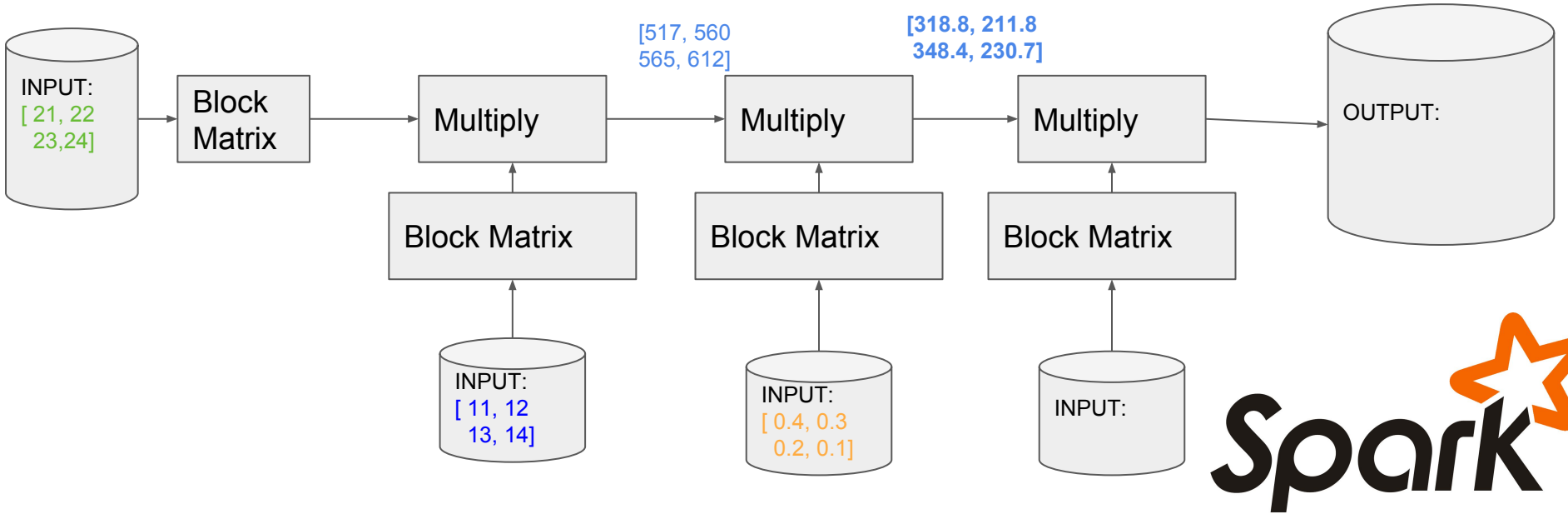
Matrix Multiplication in Spark

Spark is more than MR.



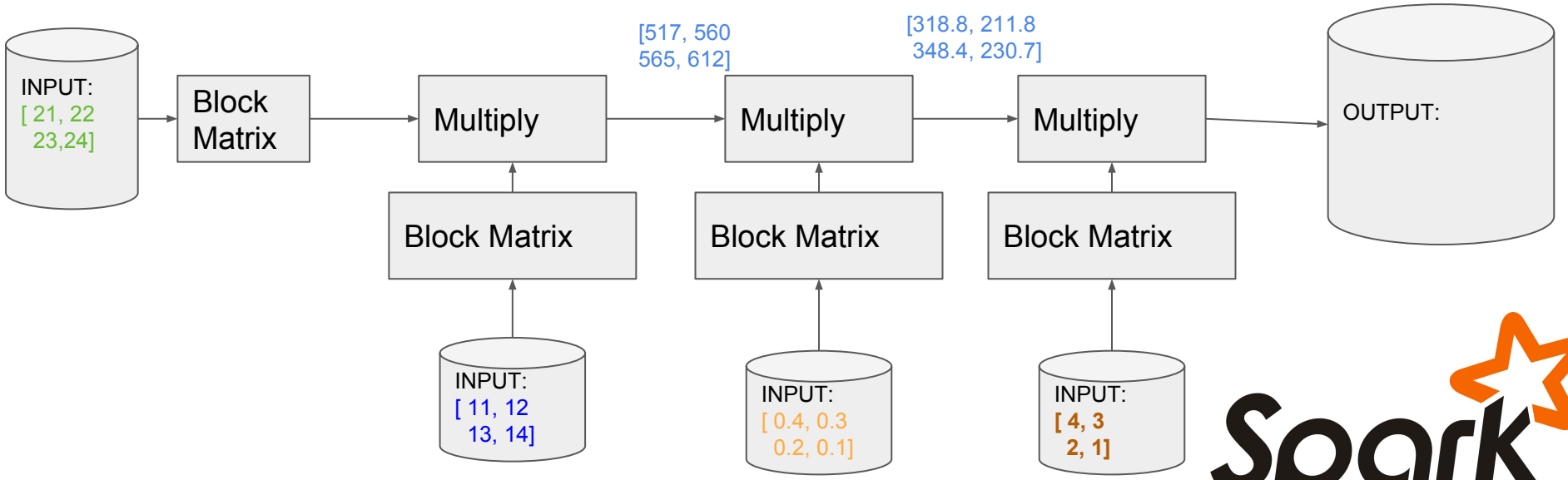
Matrix Multiplication in Spark

Spark is more than MR.



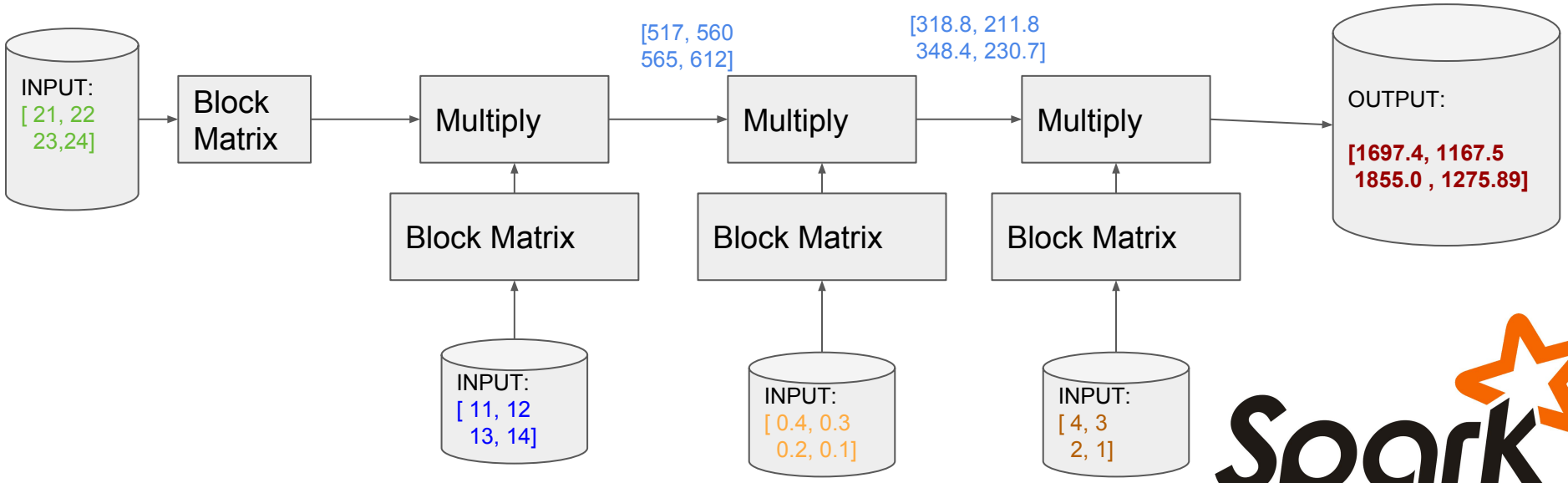
Matrix Multiplication in Spark

Spark is more than MR.

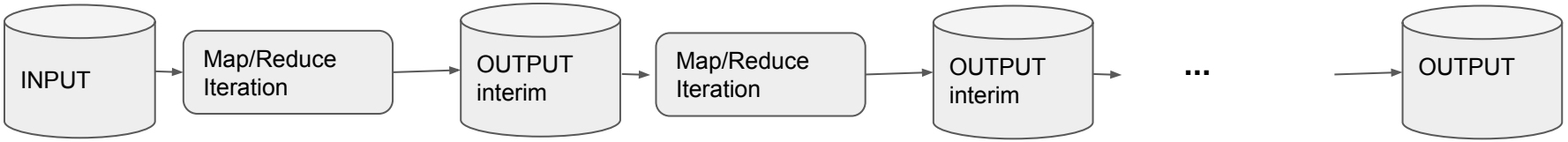


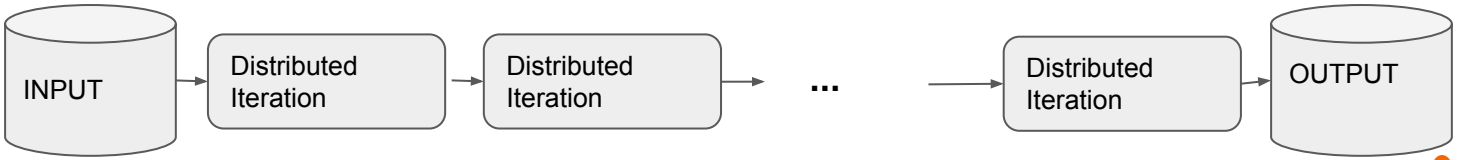
Matrix Multiplication in Spark

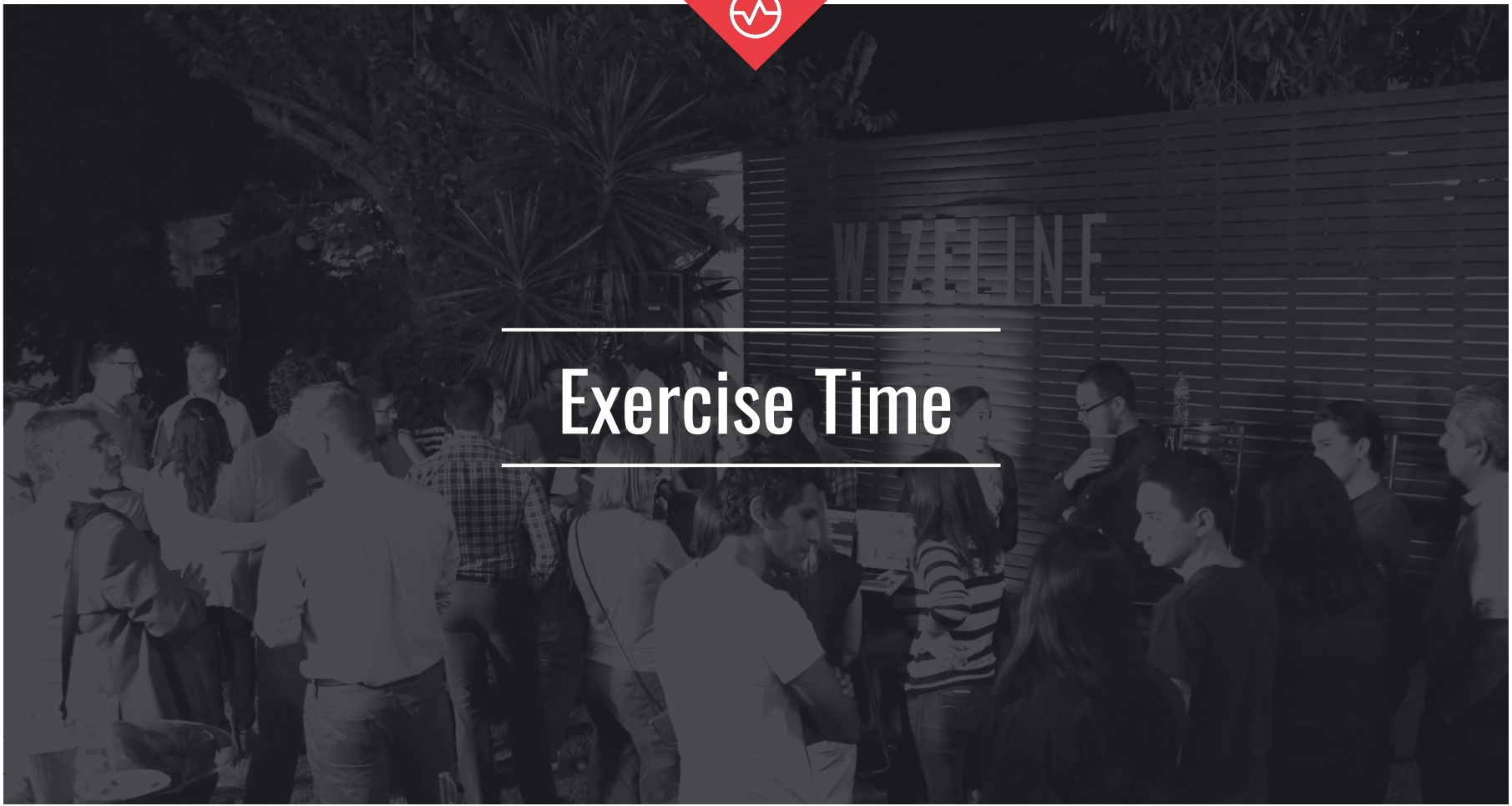
Spark is more than MR.



Iterations


INPUT
Map/Reduce Iteration
OUTPUT interim
Map/Reduce Iteration
OUTPUT interim
...
OUTPUT


INPUT
Distributed Iteration
Distributed Iteration
...
Distributed Iteration
OUTPUT



Exercise Time

Exercises

Exercise 2:

Implement the matrix multiplication using the same four original matrix.

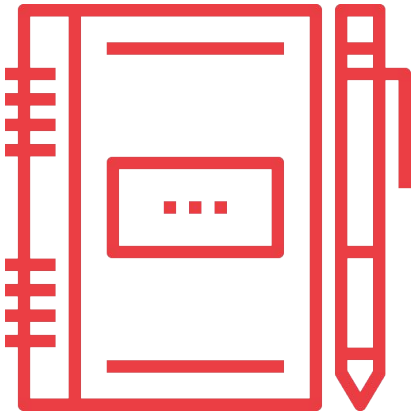
Exercise 3:

Using the same four original matrix, find the result of the multiplication of each squared matrix.

Hint 1: how can you extend the matrix multiplication to consider exponents?

Agenda

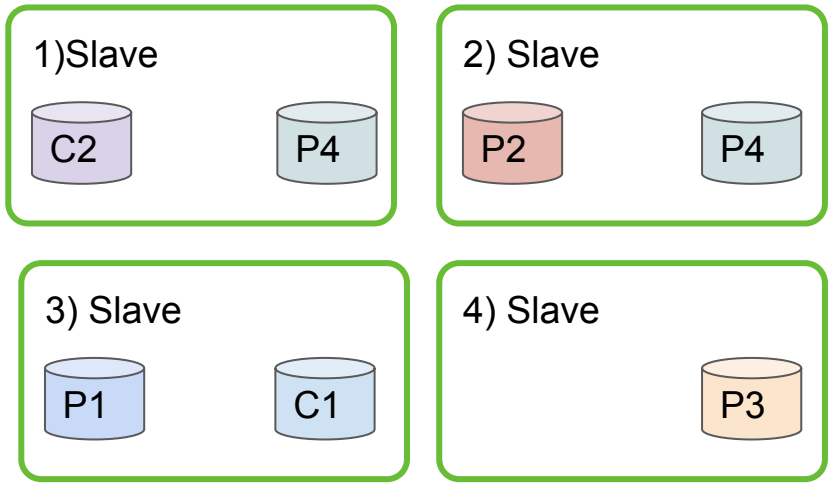
- Fault Tolerance at the Storage Level
- MapReduce
  - Parallelization
  - The Framework
  - Iterations and MapReduce
- **Fault Tolerance at the Application Level (Spark)**



What Happens If a Node Fails?

With Spark

It is also distributed:
RDD/DataFrame/DataSet



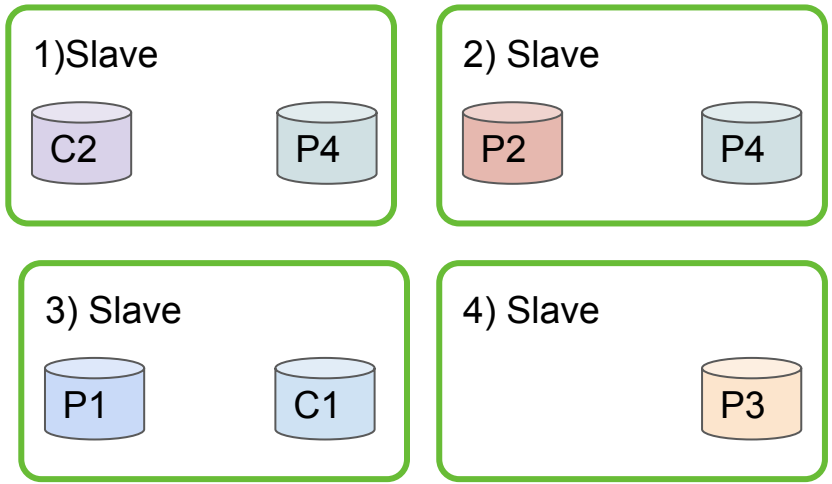
What happens if:

a) The connection to certain nodes gets bottlenecked?
b) The node that was completing the task fails?
c) One of the chunks of the partitioned dataset gets compromised at mid process?

Immutable datasets

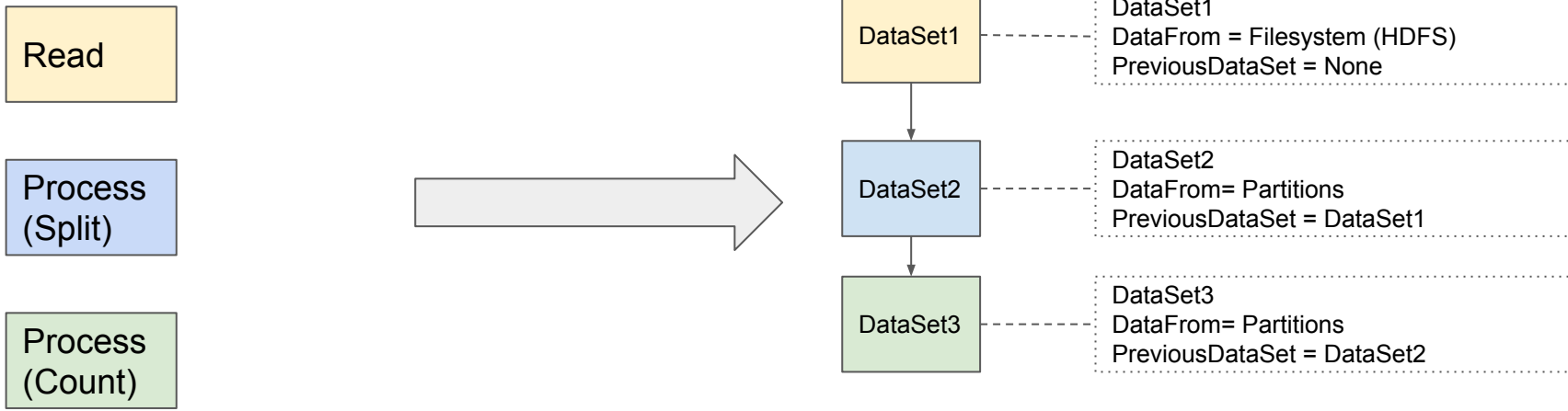
It is also distributed:
RDD/DataFrame/DataSet

Immutable distributed



Inmutable distributed: word count example

With Spark



Fault Tolerance: Example
Remember the Notebook

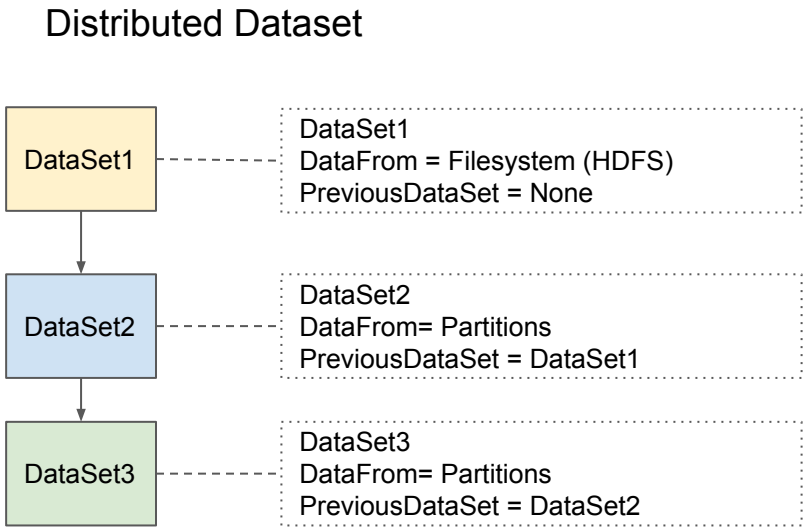

```
package com.wizeline.wordcount

import org.apache.spark.sql._

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

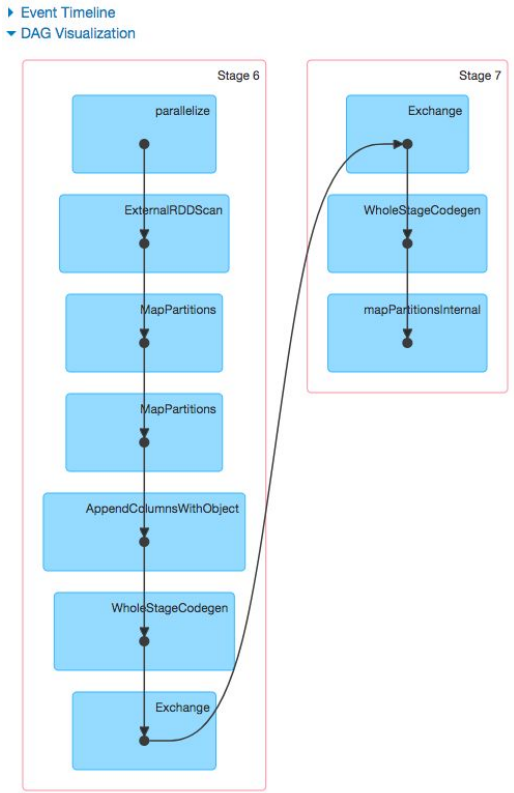
Fault Tolerance

OK, you can recover if a node fails, but who keeps track of it?



?

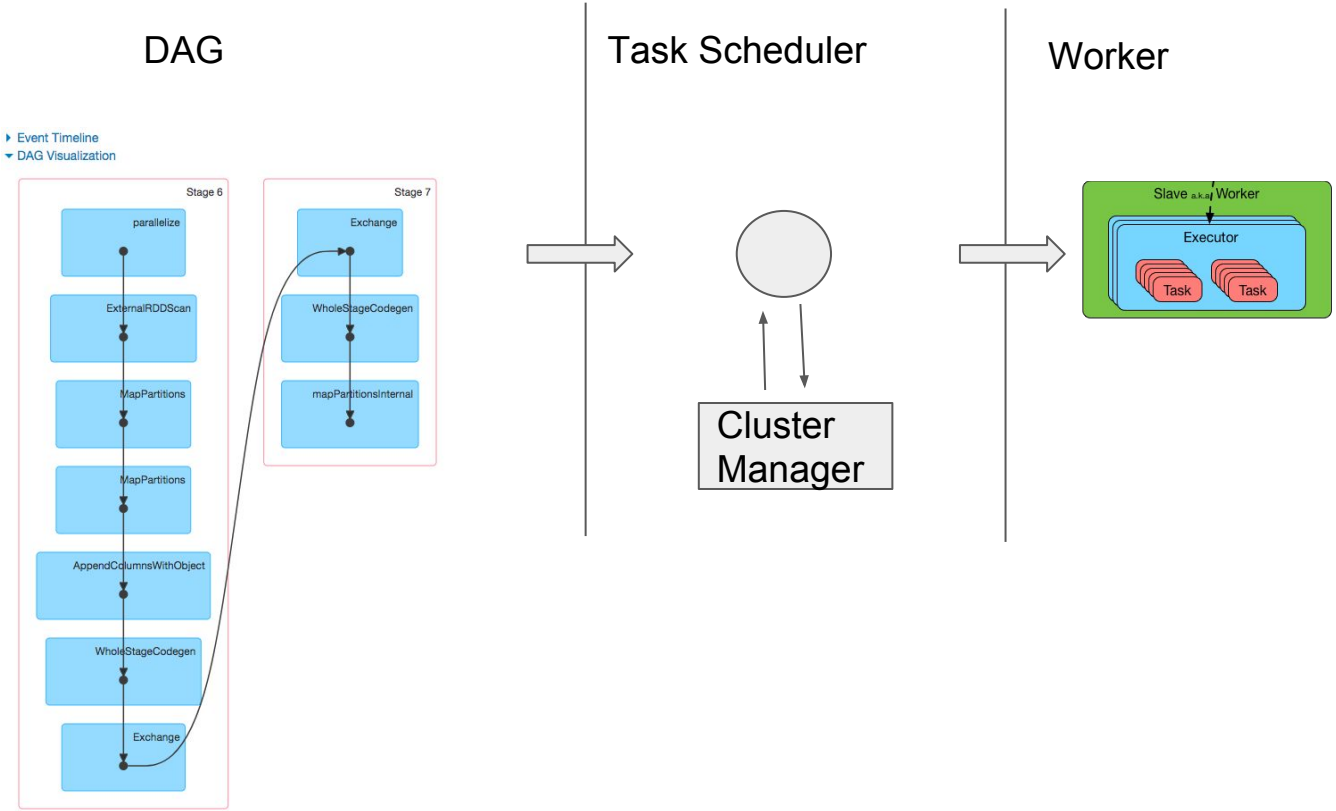
Fault Tolerance: DAG



Any distributed object is managed and represented by a Directional Acyclic Graph (DAG):

- It is separated on stages that comprise sets of tasks that can be accomplished in parallel (single node).
- It submits the stages to be performed.
- It is notified when a stage fails.

Fault Tolerance
Spark Application Overview



RECAP

- MapReduce is a parallelization paradigm that delegates all coordination and maintenance tasks to the framework.
- Spark extends this paradigm to exploit parallelization and repetition, and considers iterative activities.
- Fault tolerance is present on the file system for the cluster.
- Fault tolerance is also present with the distributed graph representation that Spark uses.

FURTHER READING

Spark

Big Data Analysis with Scala and Spark
(Coursera MOOC – Week 1)

Spark UI visualizations

HDFS

S3

YARN

Graphs implementation

Djikstra algorithm

GraphX programming

https://stanford.edu/~rezab/classes/cme323/S15/notes/lec8.pdf

For the Knowledge Hungry...

Spark Explanation
https://www.youtube.com/watch?v=49Hr5xZyTEA&feature=youtu.be

Map-Reduce costs
https://www.youtube.com/watch?v=sCaKwQEF8Ao

Assignment

To Work on Your Own at Home

Reading and Writing larger Dataset

Assignment

Bucket Location and Files Structure

The dataset you'll need to read is located in the Google bucket:

`gs://de-training-input/matrices/matrix*.txt`

Within that bucket, you'll find two files, one for each matrix:

`Matrix1.txt and matrix2.txt`

Each row in the matrix documents contains:

row number, Value1, Value2, Value3, etc

Assignment

Reformat the input so a CoordinateMatrix can be created:

1. **Perform the matrix multiplication on a bigger matrix?**

For this assignment you'll need to do several things:

- Transform the information so you can have [i,j,value] for each value in the matrix.
- Use the functions created during the exercises.
- Save the output of the multiplied matrix in your bucket under `matrix_mult-#.txt` a.

NOTE: To aid you with this particular assignment, we have made available a notebook to guide you. Bear in mind that you are still expected to submit your job with a JAR or .py

Where Can I Get this Presentation?

https://tinyurl.com/y9wybmbl

C2 - Data Engineering Academy

Your feedback is very valuable!

https://goo.gl/forms/F3hlFk1yldyXZdy03


THANK
YOU
WIZELINE®