# Big Data

Data Boot Camp

Lesson 22.1

# Class Objectives:

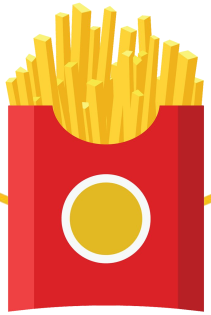
By the end of today’s class you will:

- Articulate the different components in the Hadoop ecosystem and their primary use cases.
- Run MapReduce jobs.
- Use Spark DataFrames to process large datasets.

# Big Data, Big Problems

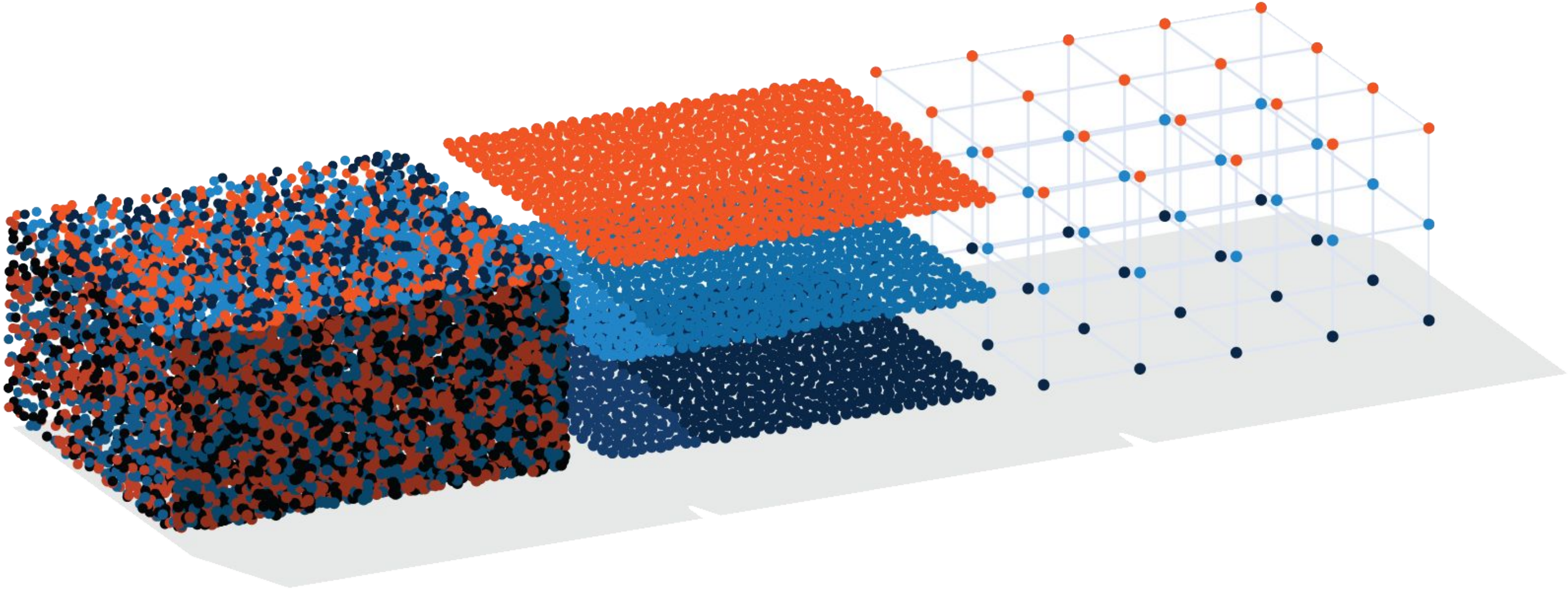
**Financial Industry Regulatory Authority (FINRA):**
Needs to store 37 records daily in the financial market and analyze trends over a period of days, weeks, and months

**McDonald's:**
Collects transactional data as it serves 69 million people each day in 188 countries

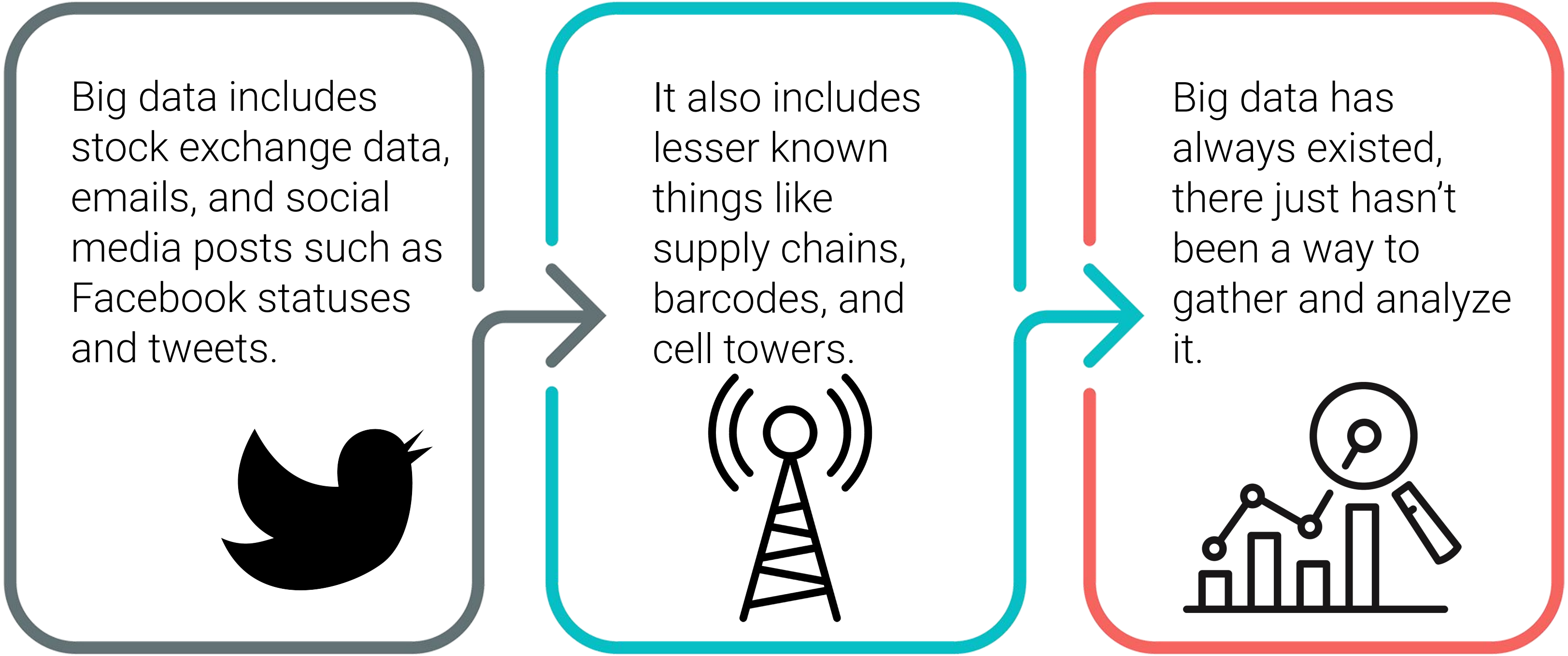


**Netflix:**
Collects ratings, searches, watch dates, device information, pause and skip data, repeat views, and more for 100+ million subscribers

# What Is Big Data?



# What Is Big Data?



Big data includes stock exchange data, emails, and social media posts such as Facebook statuses and tweets.

It also includes lesser known things like supply chains, barcodes, and cell towers.

Big data has always existed, there just hasn’t been a way to gather and analyze it.

# Four Vs of Big Data

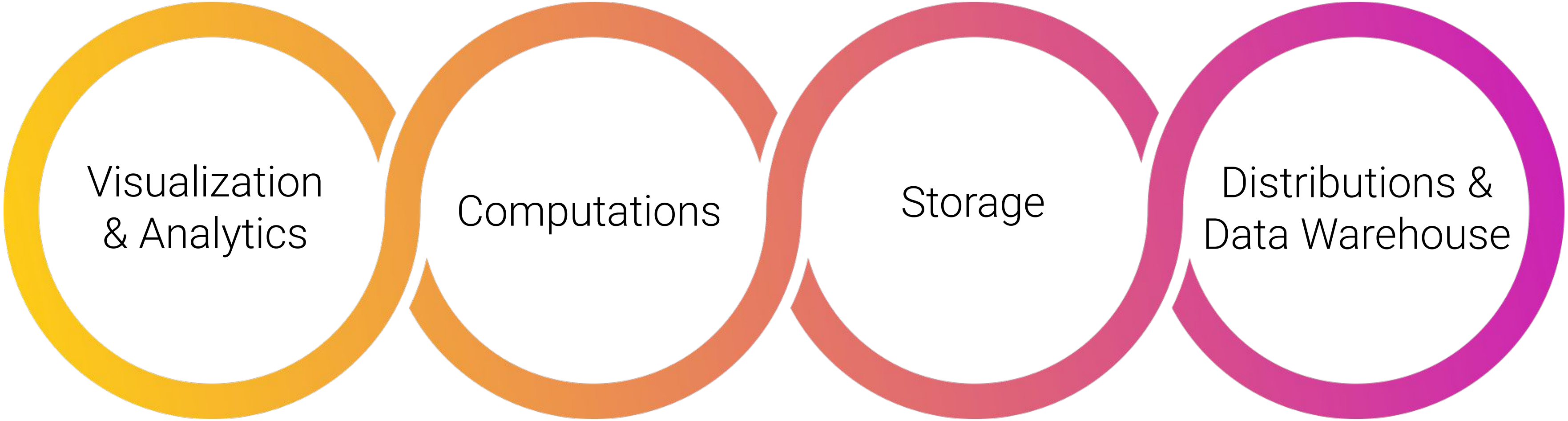
01 **Volume: Size of the data** (terabytes of stock data)

02 **Velocity: How quickly data is coming in** (car sensors sending information every second)

03 **Variety: Different forms of data** (social media posts, comments, photos, etc.)

04 **Veracity: Uncertainty of data** (social media data may not be precise, come from bots, etc.)

# Big Data Overview



Visualization & Analytics

IBM Cognos
SAP Business Objects
MicroStrategy
ZoomData
Qlik
PowerBI
Tableau

Computations

Hadoop MapReduce
HIVE
TEZ
Apache Drill
Apache Storm
Google BigQuery
Flink
Spark Streaming
Kafka

Storage

IBM GPFS
Hadoop
Apache HBASE
Apache Arrow
KUDU
MEMSQL

Distributions & Data Warehouse

Pivotal
Oracle EXADATA
TERADATA
GreenPlum
NETEZZA
Cloudera
Azure HDInsight
Amazon EMR

# What are some issues that you might encounter when dealing with extremely large datasets?

# Extremely Large Datasets

Issues you might encounter when dealing with extremely large datasets:

- Need a place to store massive amounts of data
- Need a way to access data quickly
- Need backups for hardware failure
- Need ways to analyze data quickly

# Hadoop Overview

# What is Hadoop?

> The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.
>
> —*Hadoop website*

**Hadoop was named after a stuffed yellow elephant belonging to the son of Doug Cutting, the Hadoop creator.**

# Fundamentals

01

MapReduce

02

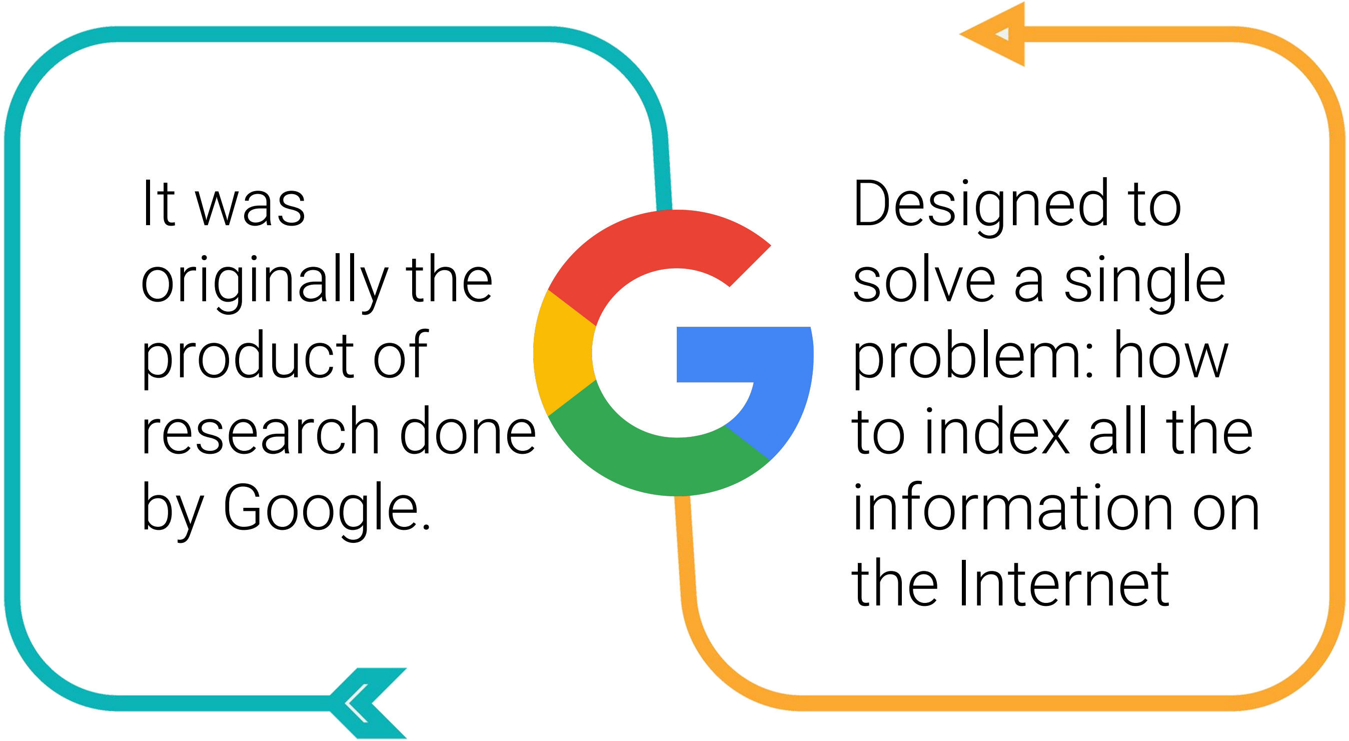
Hadoop Distributed File System (HDFS)

03

Spark

# MapReduce

# MapReduce History



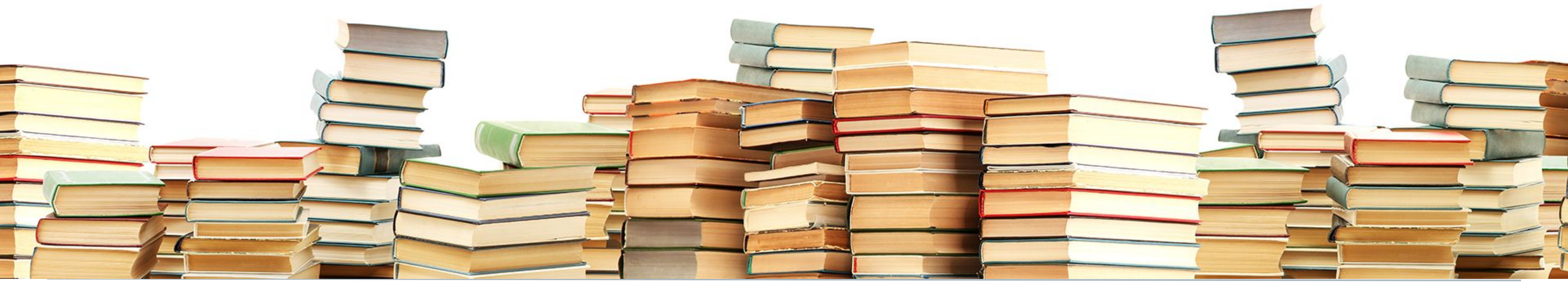
It was originally the product of research done by Google.

Designed to solve a single problem: how to index all the information on the Internet

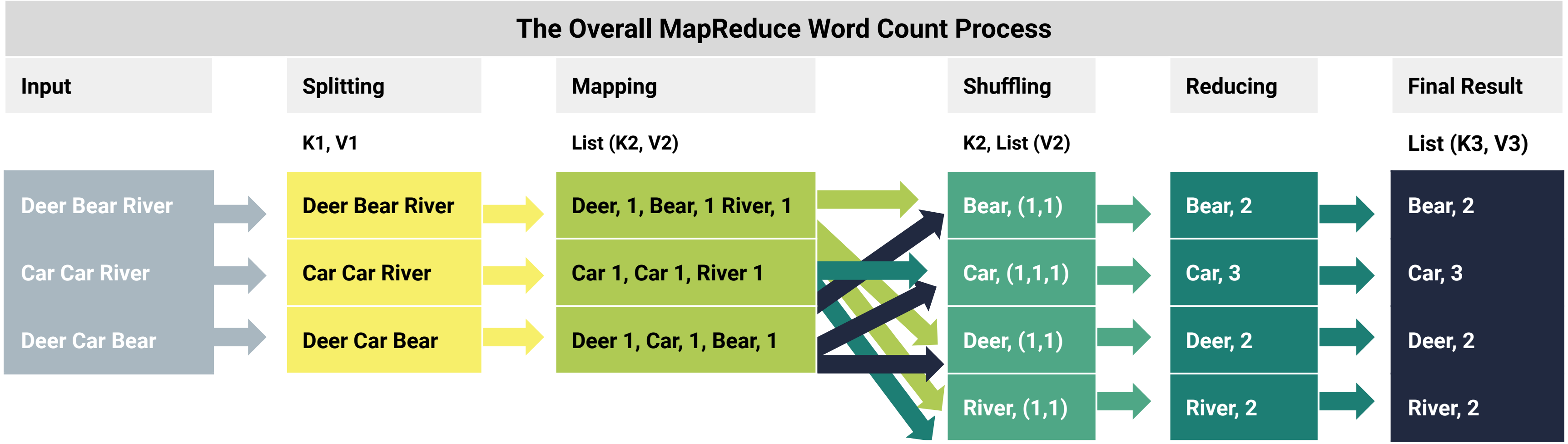
# **Example:** Counting the Number of Books in a Library

**Map:** You count this half of the library, and I’ll count the other.

**Reduce:** We get together and add up our counts.



# Classic Word Count Example

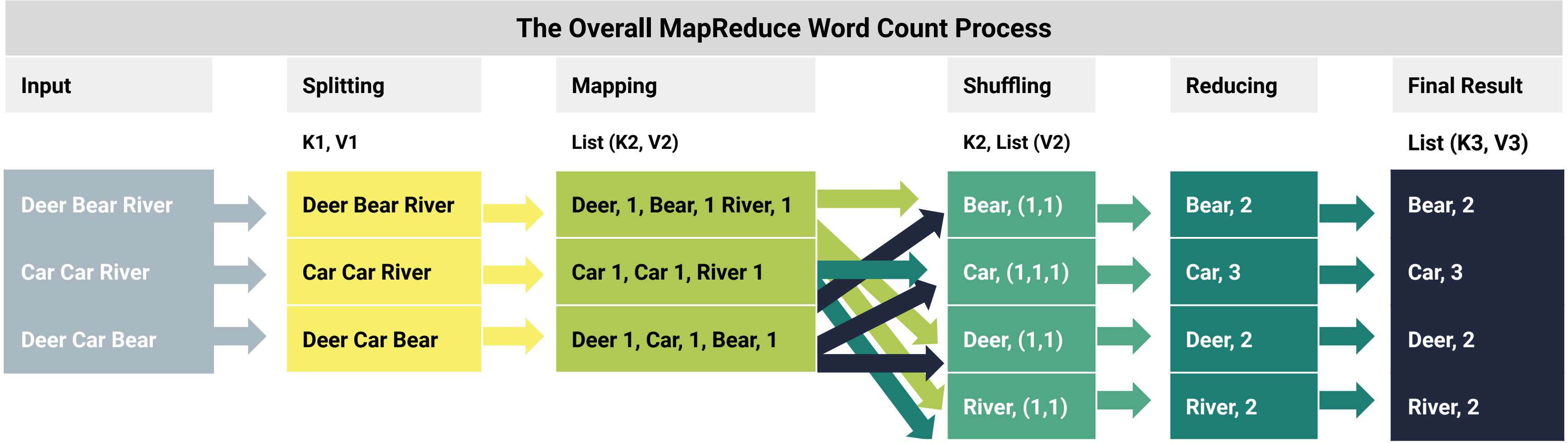


The Overall MapReduce Word Count Process

| Input | Splitting | Mapping | Shuffling | Reducing | Final Result |
|---|---|---|---|---|---|
| | K1, V1 | List (K2, V2) | K2, List (V2) | | List (K3, V3) |
| Deer Bear River | Deer Bear River | Deer, 1, Bear, 1 River, 1 | Bear, (1,1) | Bear, 2 | Bear, 2 |
| Car Car River | Car Car River | Car 1, Car 1, River 1 | Car, (1,1,1) | Car, 3 | Car, 3 |
| Deer Car Bear | Deer Car Bear | Deer 1, Car, 1, Bear, 1 | Deer, (1,1) | Deer, 2 | Deer, 2 |
| | | | River, (1,1) | River, 2 | River, 2 |

# The Map Part of MapReduce

- Execute the `Map()` function on data.
- Execute on each node.
- Output `<key, value>` pairs on each node.

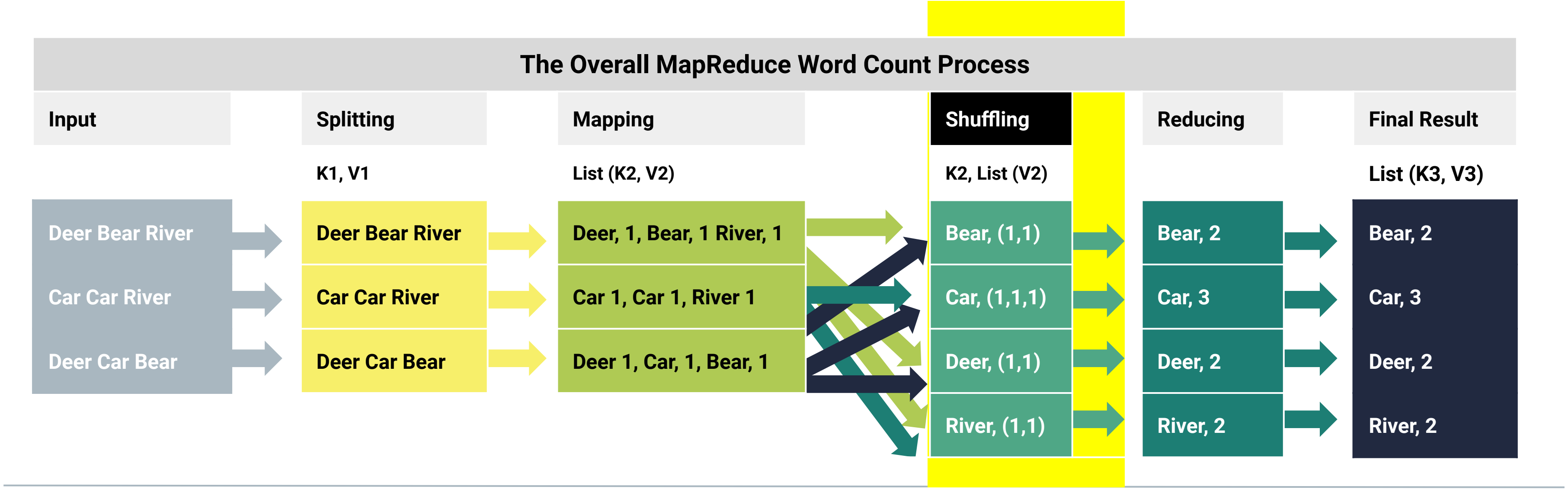


**The Overall MapReduce Word Count Process**

| Input | Splitting | Mapping | Shuffling | Reducing | Final Result |
|---|---|---|---|---|---|
| | K1, V1 | List (K2, V2) | K2, List (V2) | | List (K3, V3) |
| Deer Bear River | Deer Bear River | Deer, 1, Bear, 1 River, 1 | Bear, (1,1) | Bear, 2 | Bear, 2 |
| Car Car River | Car Car River | Car 1, Car 1, River 1 | Car, (1,1,1) | Car, 3 | Car, 3 |
| Deer Car Bear | Deer Car Bear | Deer 1, Car, 1, Bear, 1 | Deer, (1,1) | Deer, 2 | Deer, 2 |
| | | | River, (1,1) | River, 2 | River, 2 |

# The Shuffle in MapReduce

The shuffle step groups the keys together. Each value found for a key is appended to the list of values.

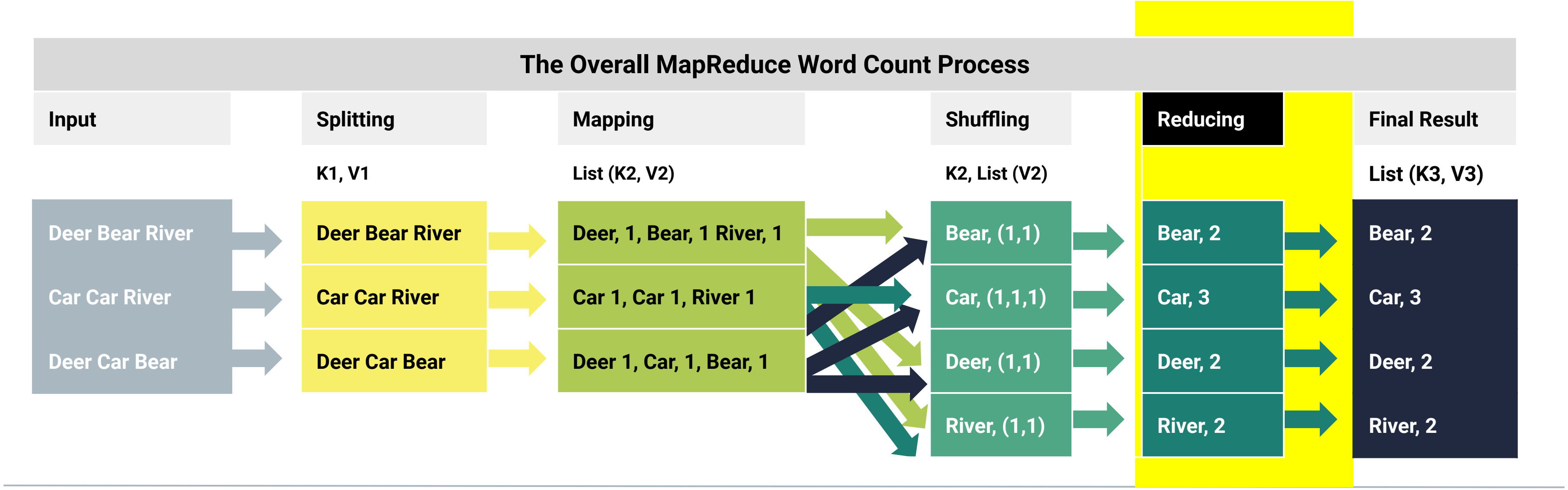


The Overall MapReduce Word Count Process

| Input | Splitting | Mapping | Shuffling | Reducing | Final Result |
|---|---|---|---|---|---|
| | K1, V1 | List (K2, V2) | K2, List (V2) | | List (K3, V3) |
| Deer Bear River | Deer Bear River | Deer, 1, Bear, 1 River, 1 | Bear, (1,1) | Bear, 2 | Bear, 2 |
| Car Car River | Car Car River | Car 1, Car 1, River 1 | Car, (1,1,1) | Car, 3 | Car, 3 |
| Deer Car Bear | Deer Car Bear | Deer 1, Car, 1, Bear, 1 | Deer, (1,1) | Deer, 2 | Deer, 2 |
| | | | River, (1,1) | River, 2 | River, 2 |

# The Reduce in MapReduce

The reducing step reduces the list of the values for a key to a single value. Tin this example, the values are added to get the count of each word.



The Overall MapReduce Word Count Process

| Input | Splitting | Mapping | Shuffling | Reducing | Final Result |
|---|---|---|---|---|---|
| | K1, V1 | List (K2, V2) | K2, List (V2) | | List (K3, V3) |
| Deer Bear River | Deer Bear River | Deer, 1, Bear, 1 River, 1 | Bear, (1,1) | Bear, 2 | Bear, 2 |
| Car Car River | Car Car River | Car 1, Car 1, River 1 | Car, (1,1,1) | Car, 3 | Car, 3 |
| Deer Car Bear | Deer Car Bear | Deer 1, Car, 1, Bear, 1 | Deer, (1,1) | Deer, 2 | Deer, 2 |
| | | | River, (1,1) | River, 2 | River, 2 |

# mrjob

# What Is MRjob?

- mrjob is a Python library built by Yelp.

- It lets you write MapReduce jobs in Python and run them on different platforms.

- It also lets you stay in Python while making it easier to process and analyze big data.

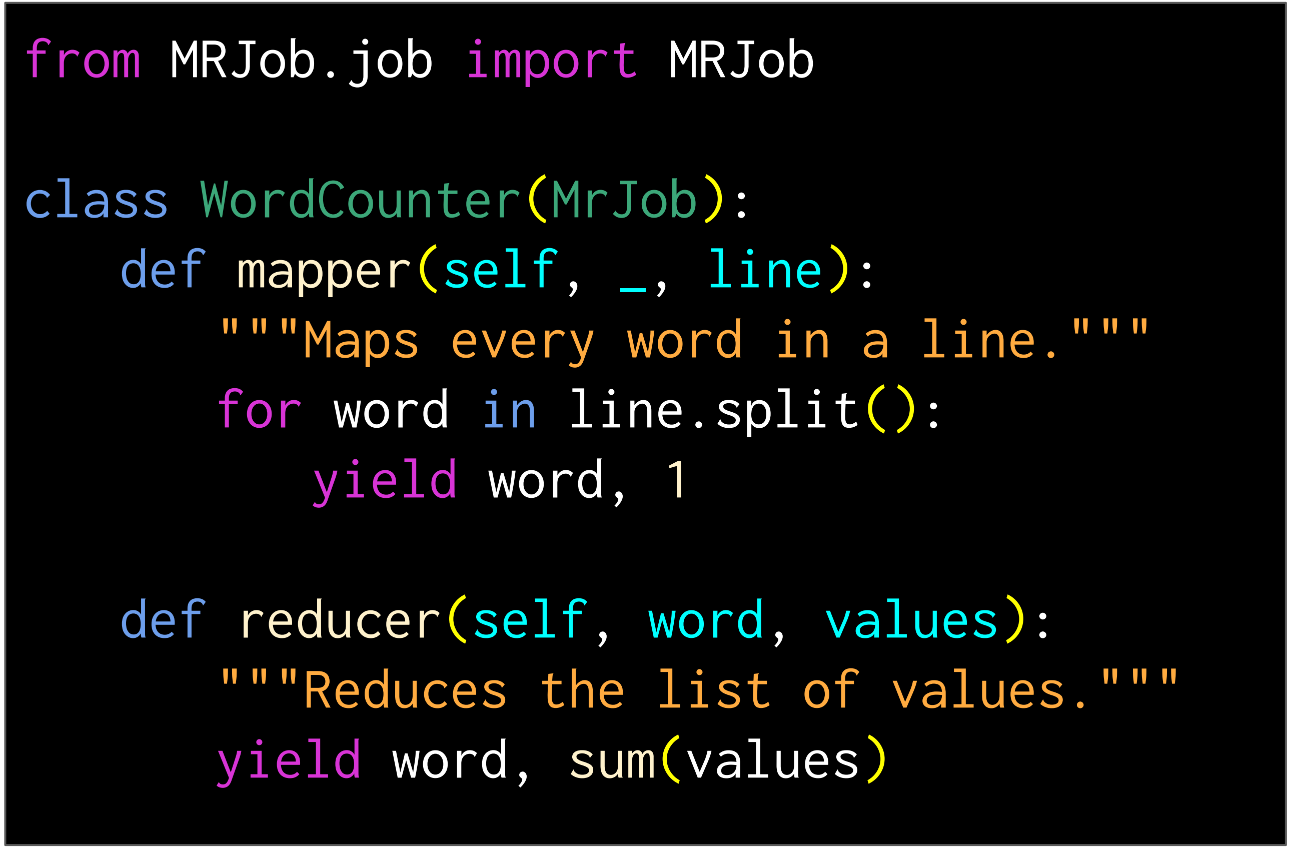
# What Is a Job?

- A job is defined by a class that inherits from `MRjob`.
- This class contains methods that define the steps of your job.
- Can translate the job and run it locally or on a Hadoop cluster.
- Note: the shuffle step is handled behind the scenes

```
from mrjob.job import MRJob

class WordCounter(MRJob):
```

# Basic Example



```
from MRJob.job import MRJob

class WordCounter(MrJob):
    def mapper(self, _, line):
        """Maps every word in a line."""
        for word in line.split():
            yield word, 1

    def reducer(self, word, values):
        """Reduces the list of values."""
        yield word, sum(values)
```

# Activity: Word Count

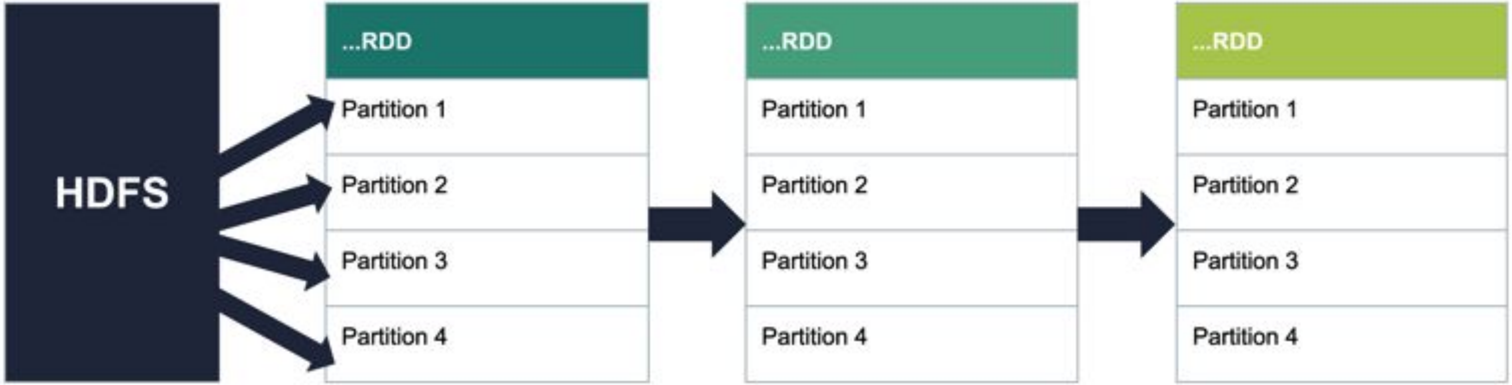
In this activity, we will create our first Job using MRJob. Our task is to count how many times the word “bacon” appears in a text file.

**Suggested Time:**
10 Minutes

# Hadoop Distributed File System

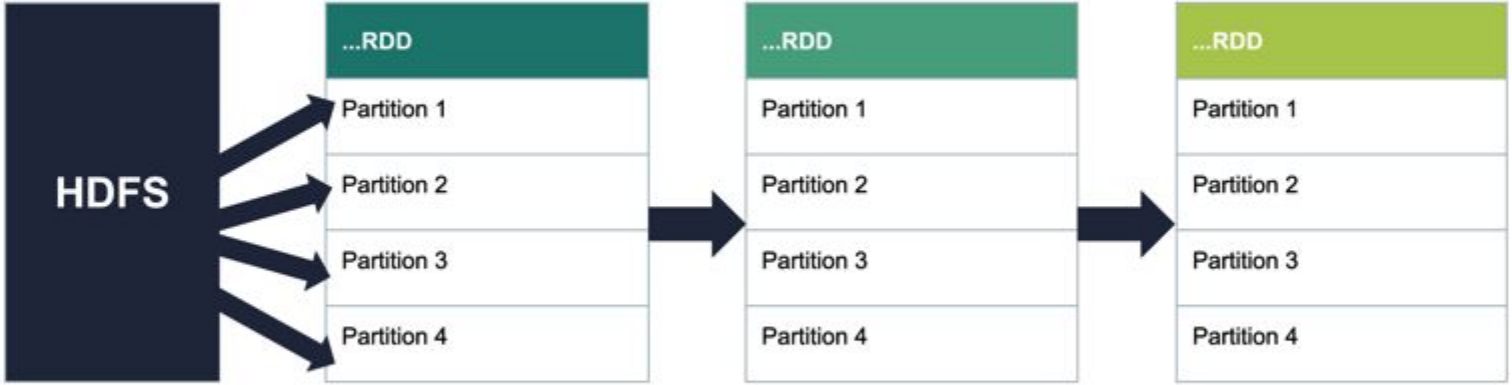
# Hadoop Distributed File System (HDFS)

HDFS is a file system that is used to store data across server clusters, and is **scalable, fault-tolerant,** and **distributed.**



# Hadoop Distributed File System (HDFS)

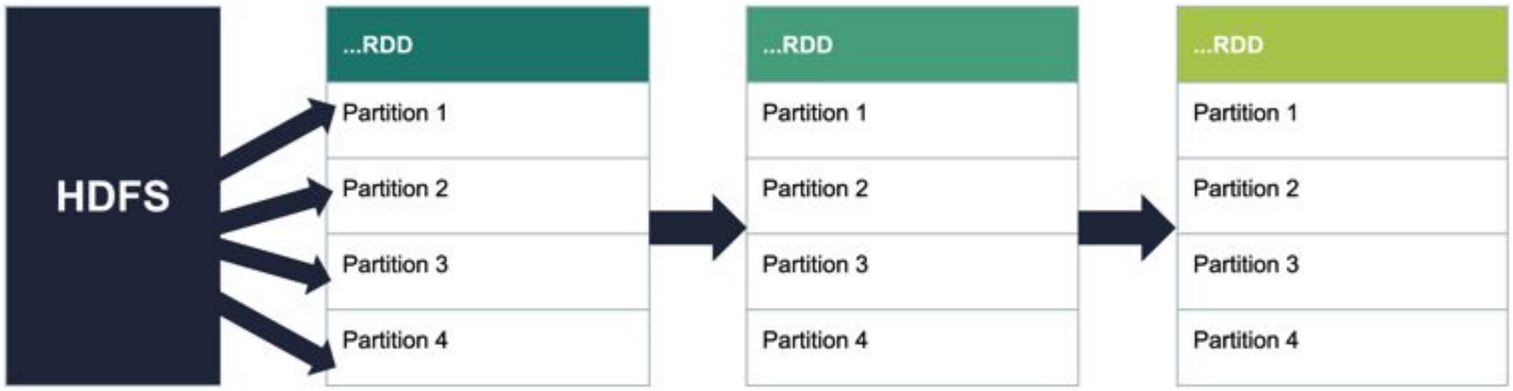
- Partitioned large data sets
- Store files across a network of machines
- Evolved from the GFS, or Google File System



# Resilient Distributed Dataset

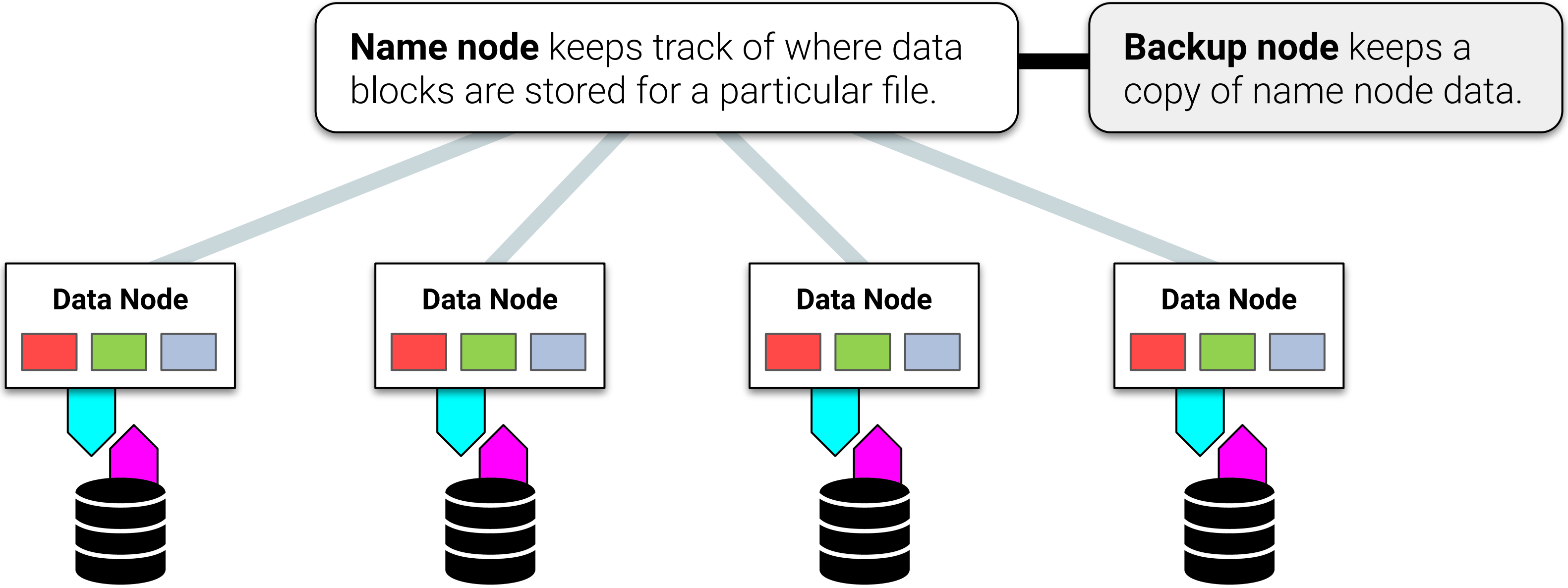
A resilient distributed dataset (RDD) is the basic abstraction in Spark.

It represents an immutable, partitioned collection of elements that can be operated on in parallel.



# Node

The client will request data through the name node. The name node will locate that data across its data node, and then return data back to the client.



**Name node** keeps track of where data blocks are stored for a particular file.

**Backup node** keeps a copy of name node data.

Data nodes store the data and return it on request.

# Hadoop Distributed File System (HDFS)

## Pros

1. Handles terabytes of data
2. Write once, read many times
3. Uses commodity hardware

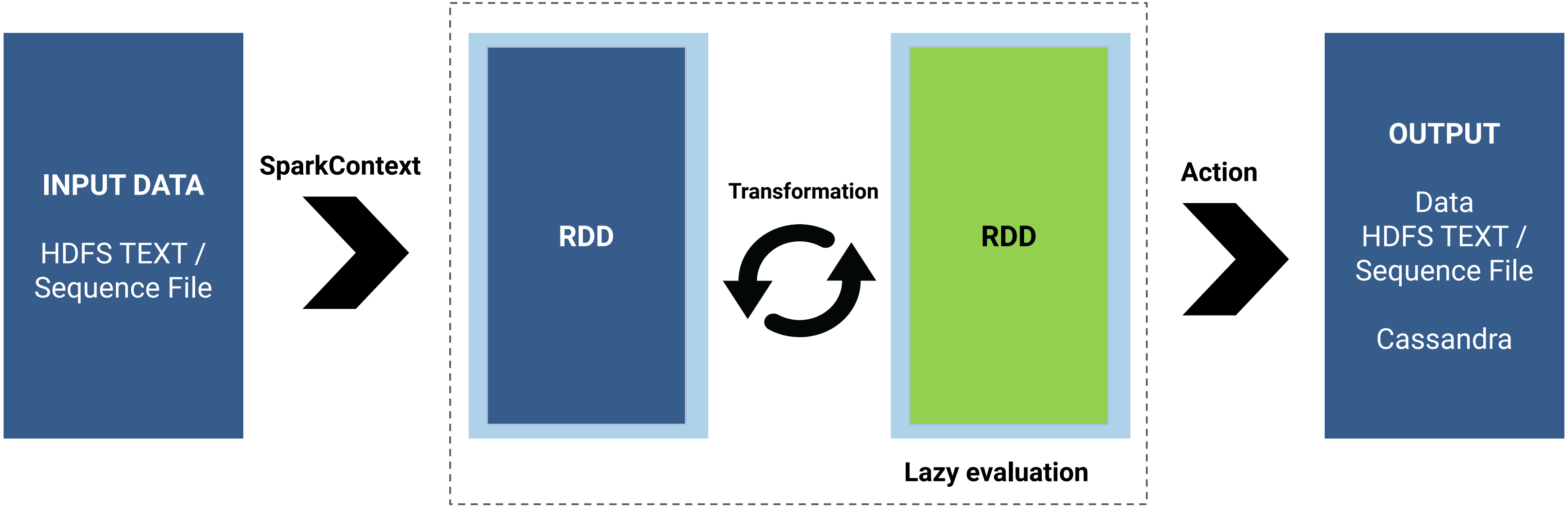
## Cons

1. Not good to low latency access
2. Bad for lots of small files
3. Not for multiple writers

# Spark (PySpark)

# What Is Spark?

Apache Spark is a unified analytics engine for large-scale data processing.
It lets you write applications in Java, Scala, Python, R, and SQL and runs on Hadoop, stand-alone, or in the cloud (and many other platforms).
Spark can be 100 times faster than Hadoop.



# Spark DataFrames

- Spark DataFrames are similar to Pandas.
- They hold data in a column and row format.
- Each column represents a variable or feature.
- Each row represents one data point.

Unlike Pandas, Spark DataFrames can scale to handle petabytes of data on clusters of servers or in the cloud.