



# BITS Pilani presentation

**BITS Pilani**  
Pilani Campus

Dr. Vivek V. Jog  
Dept. Of Computer Engineering



**BITS Pilani**  
Pilani Campus



# Big Data Systems (S1-24\_CCZG522)

## Lecture No.3

# INDEX



- Big Data Growth Drivers
- What is Big Data?
- Hadoop Introduction
- Hadoop Master/Slave Architecture
- Hadoop Core Components
- HDFS Data Blocks
- HDFS Read/Write Mechanism
- What is MapReduce
- MapReduce Program
- MapReduce Job Workflow
- Hadoop Ecosystem



# Five V's

innovate

achieve

lead

"Big data is the term for a collection of data sets so **large and complex** that it becomes **difficult** to process using on-hand database management tools or traditional data processing applications"

## Volume



Processing increasing huge data sets

## Variety



Processing different types of data

## Velocity



Data is being generated at an **alarming rate**

## Value



Finding correct **meaning** out of the data

## Veracity

Age	Sex	Height	Wt
47	?	1.68	0.87
50	44	1.55	0.85
55	75	1.58	0.40
61	23	?	0.75

**Uncertainty and inconsistencies** in the data

# Traditional System V/s Big Data

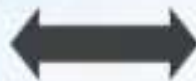


Traditional Scenario:

2 orders per hour



Single Cook



Food Shelf

Traditional Scenario:

Data is generated at a steady rate and is structured in nature



Traditional Processing System



RDBMS



# Contd..



## Scenario 2:

- They started taking Online orders
- 10 orders per hour



Single Cook  
(Regular Computing System)



Food Shelf  
(Data)

## Big Data Scenario:

Heterogenous data is being generated at an alarming rate by multiple sources

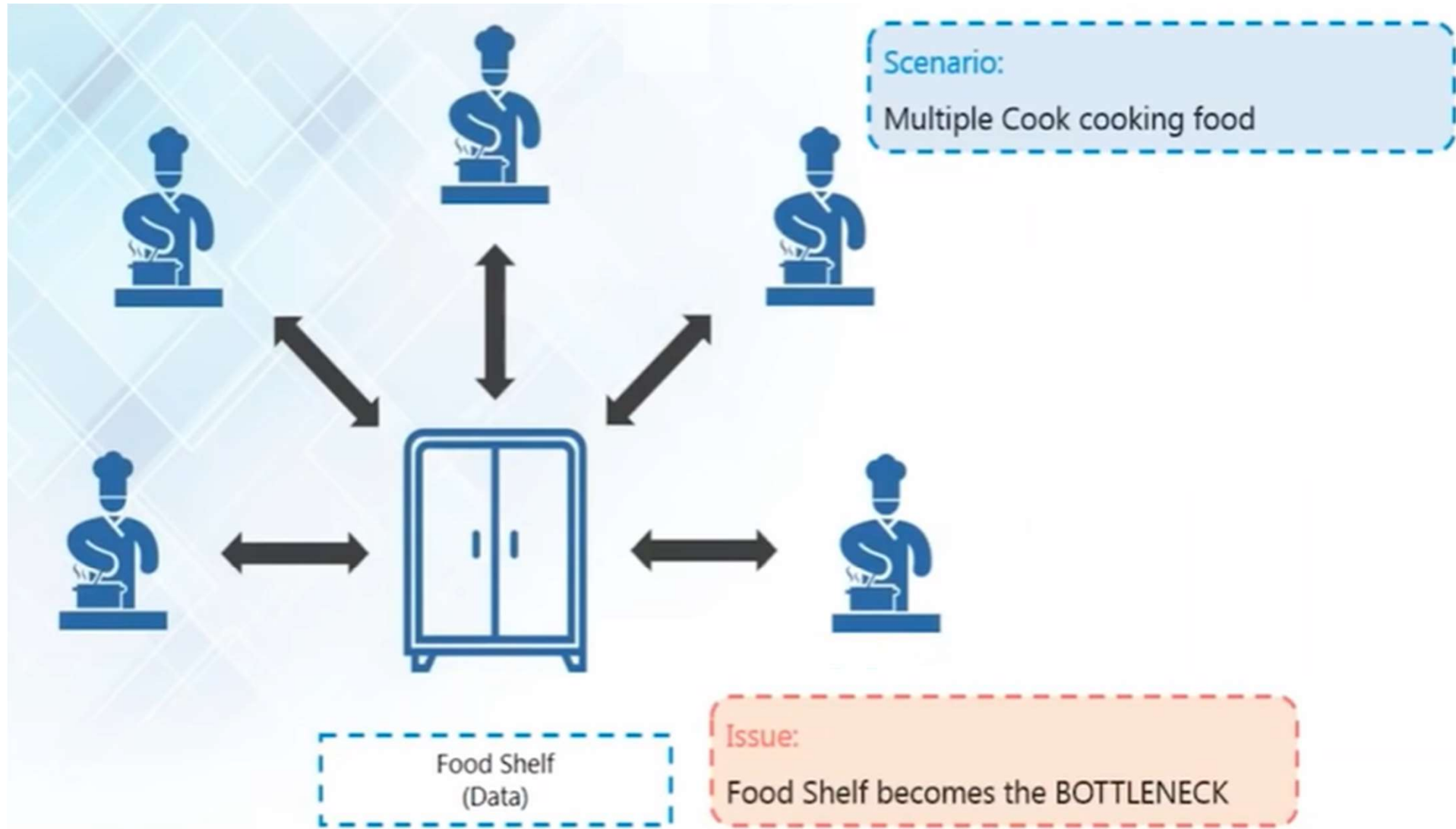


Traditional Processing  
System



RDBMS

# Bottleneck



# Bottleneck



## Scenario:

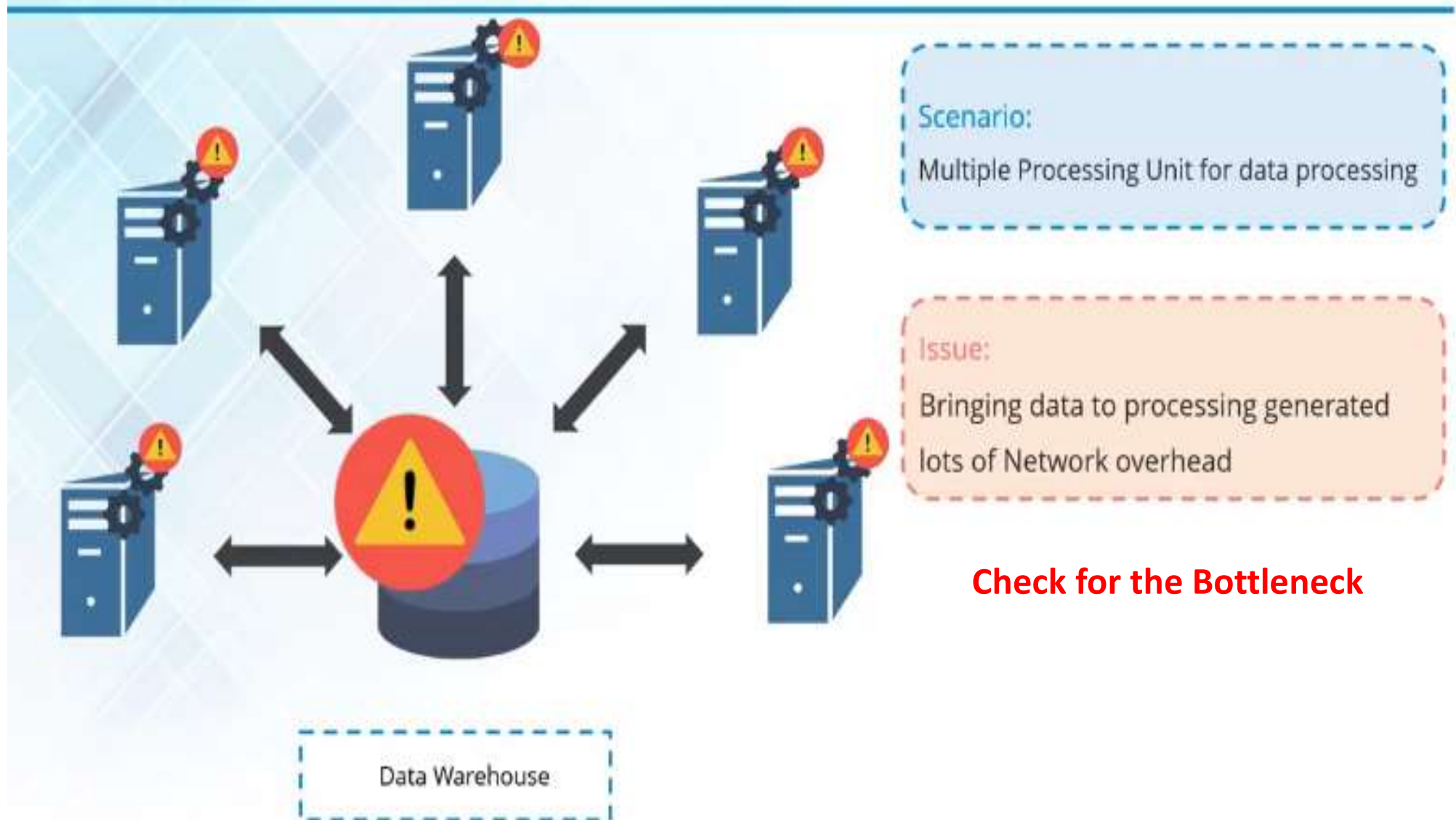
Multiple Processing Unit for data processing

## Issue:

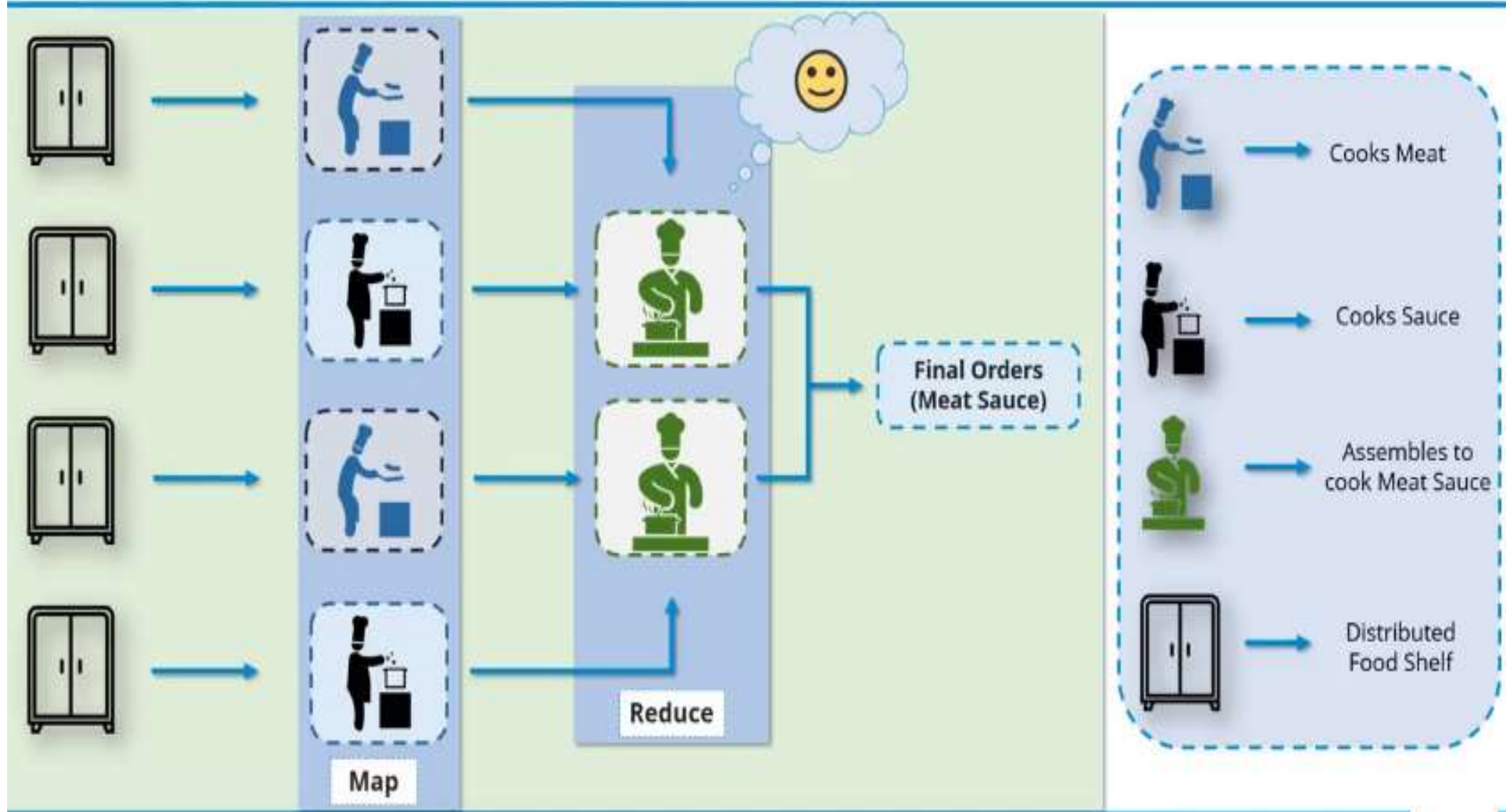
Bringing data to processing generated lots of Network overhead



# More number of orders = More number of cooks



# Effective Solution



# Solution Framework

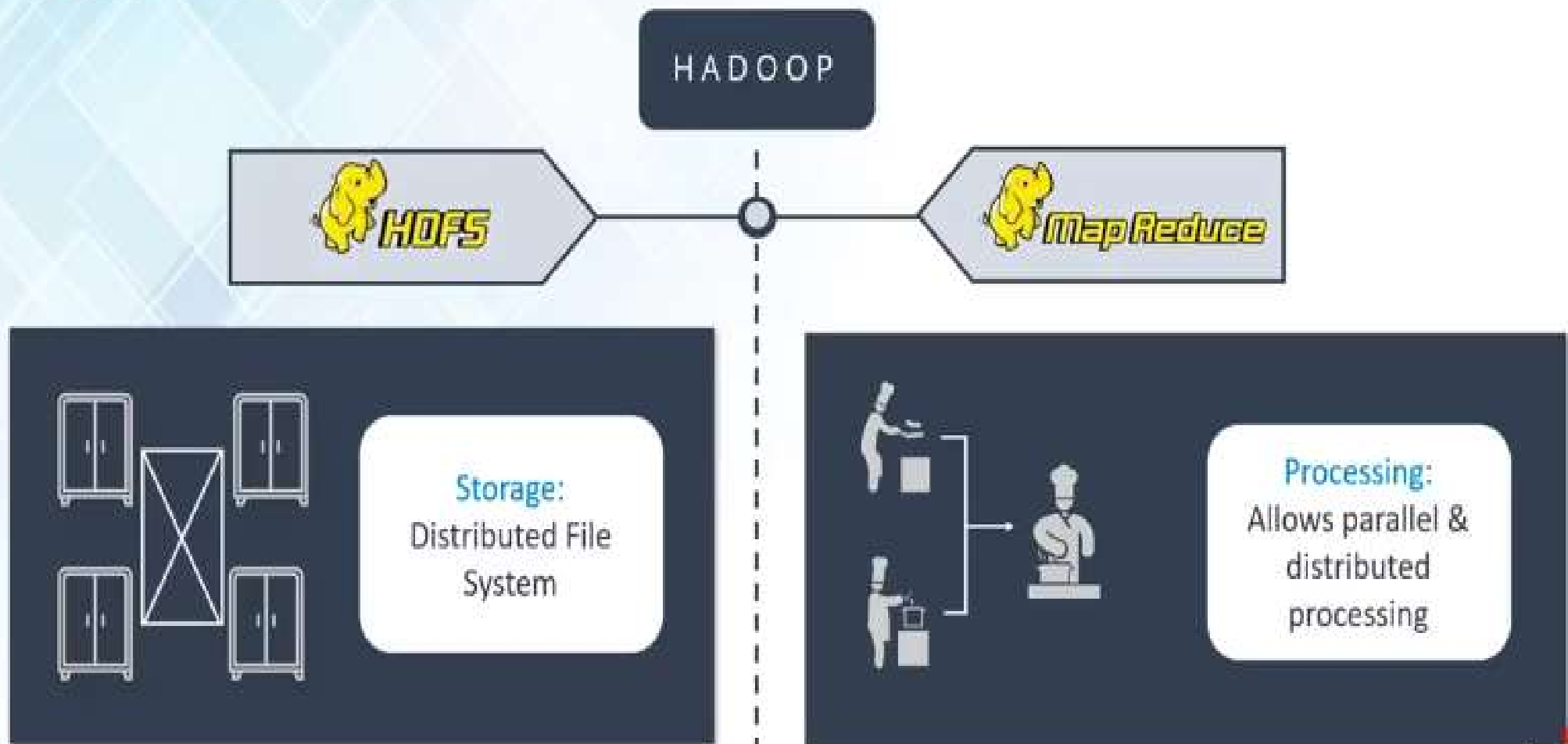


WE NEED SOME FRAMEWORK THAT CAN PROVIDES  
SOLUTION

# Apache Hadoop



Hadoop is a framework that allows us to store and process large data sets in parallel and distributed fashion

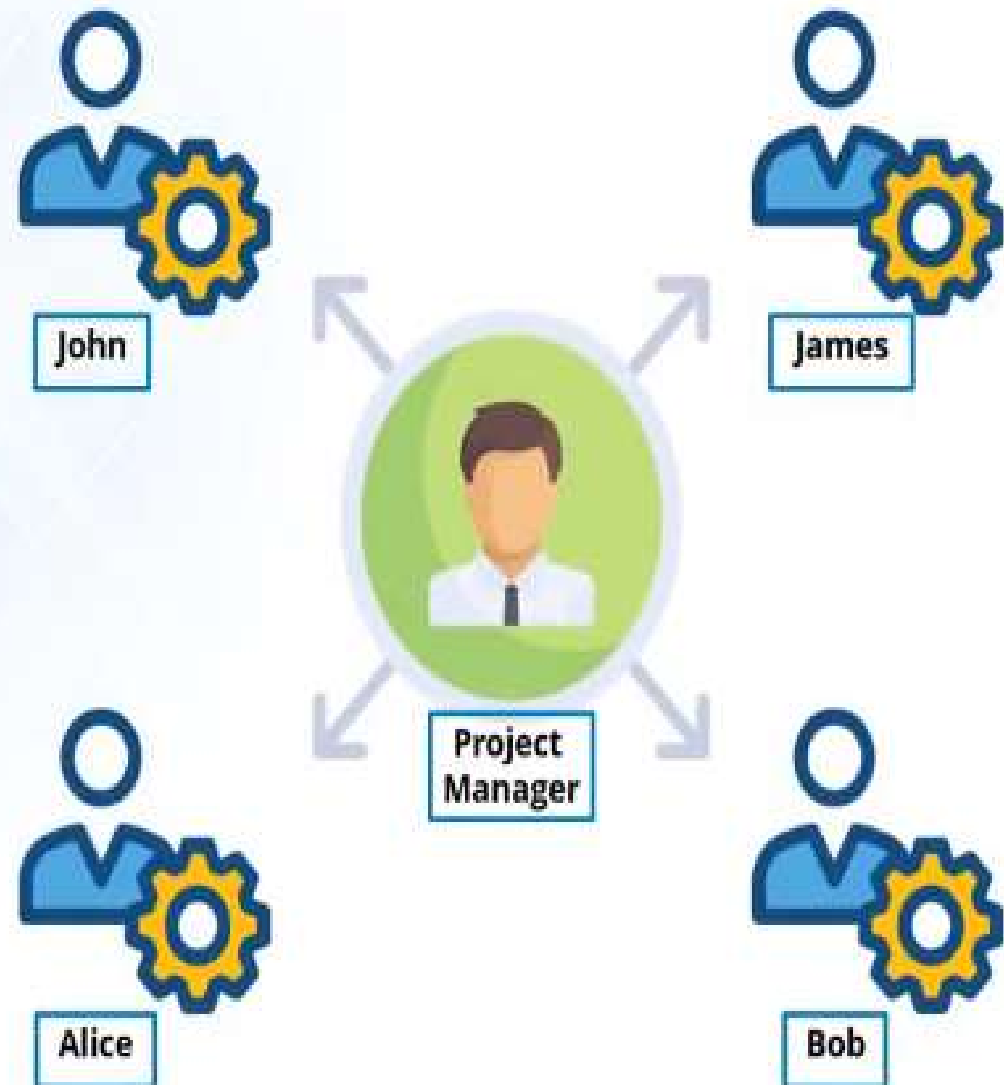


# Master/Slave Approach



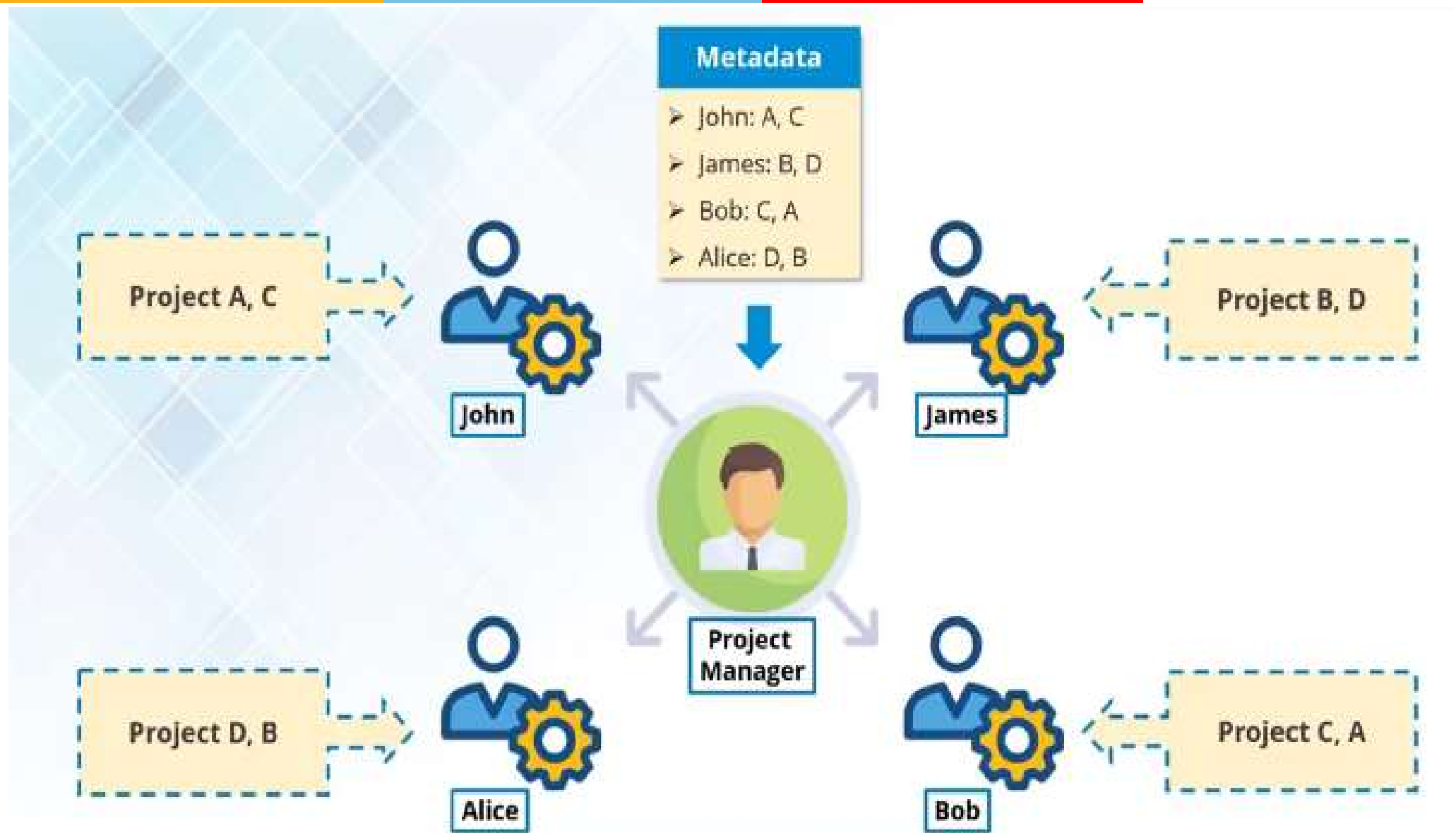
## Scenario:

A project Manager managing a team of four employees. He assigns project to each of them and tracks the progress

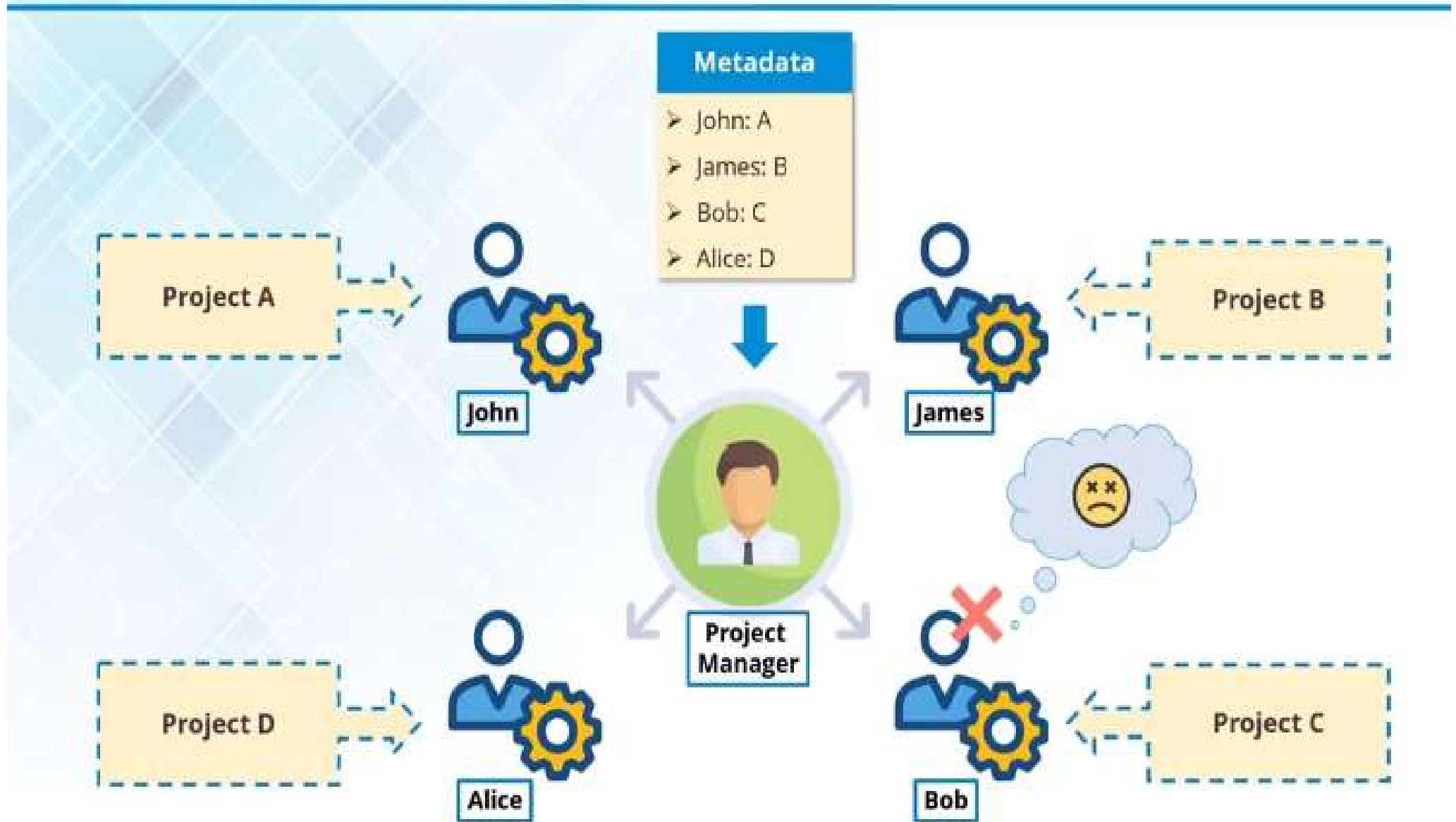




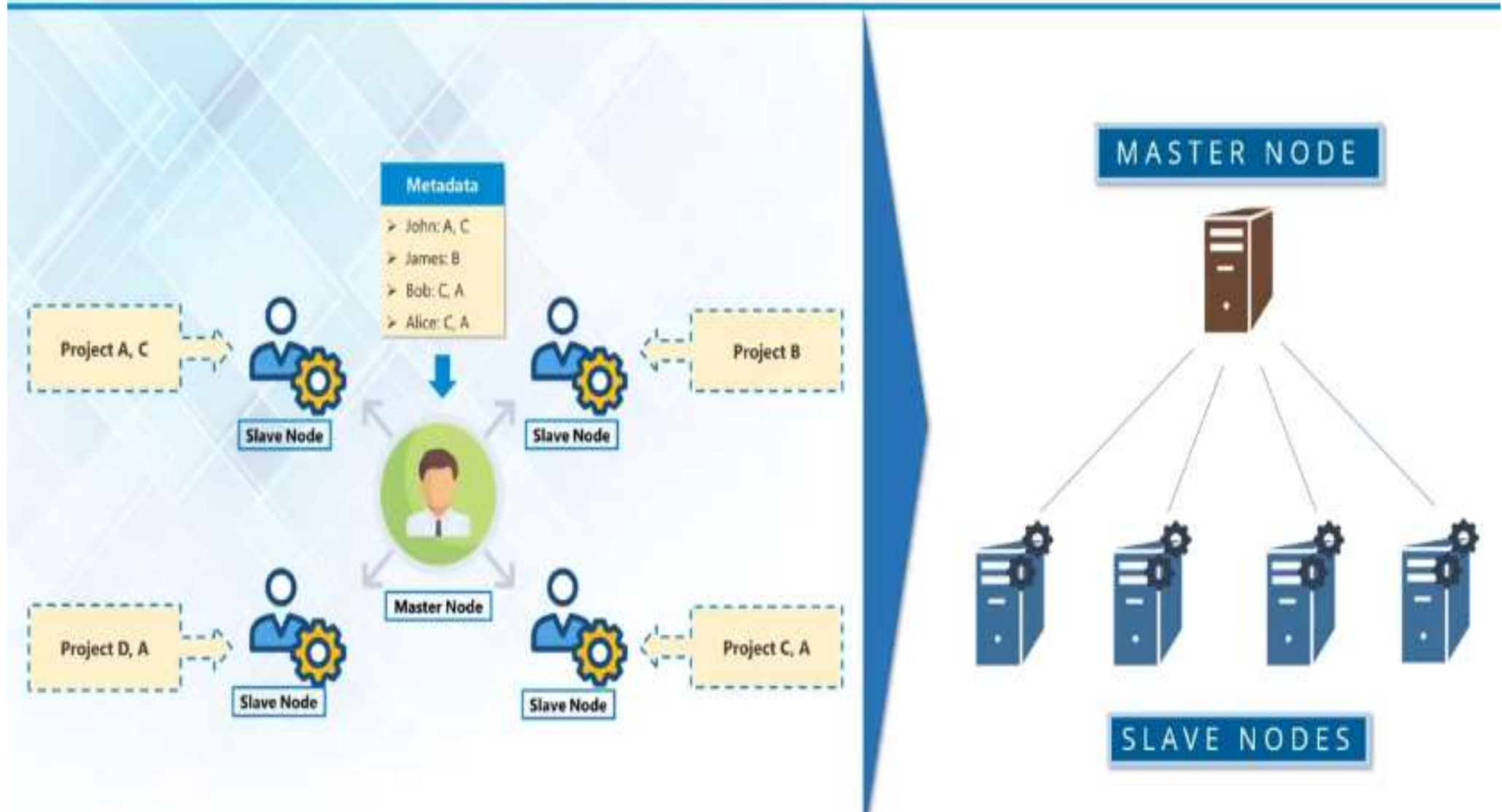
# Maintain Metadata



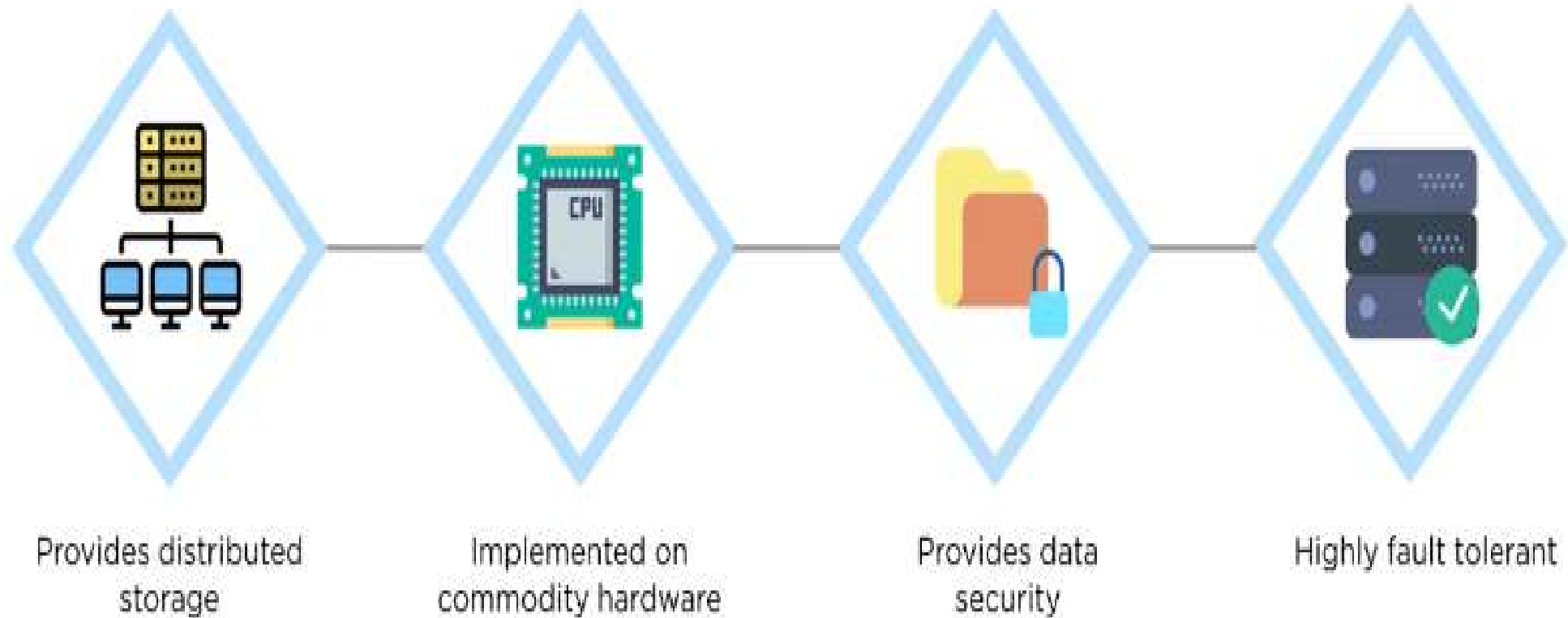
# Backup Failover Solution



# Evolution of Architecture



# Evolution of HDFS

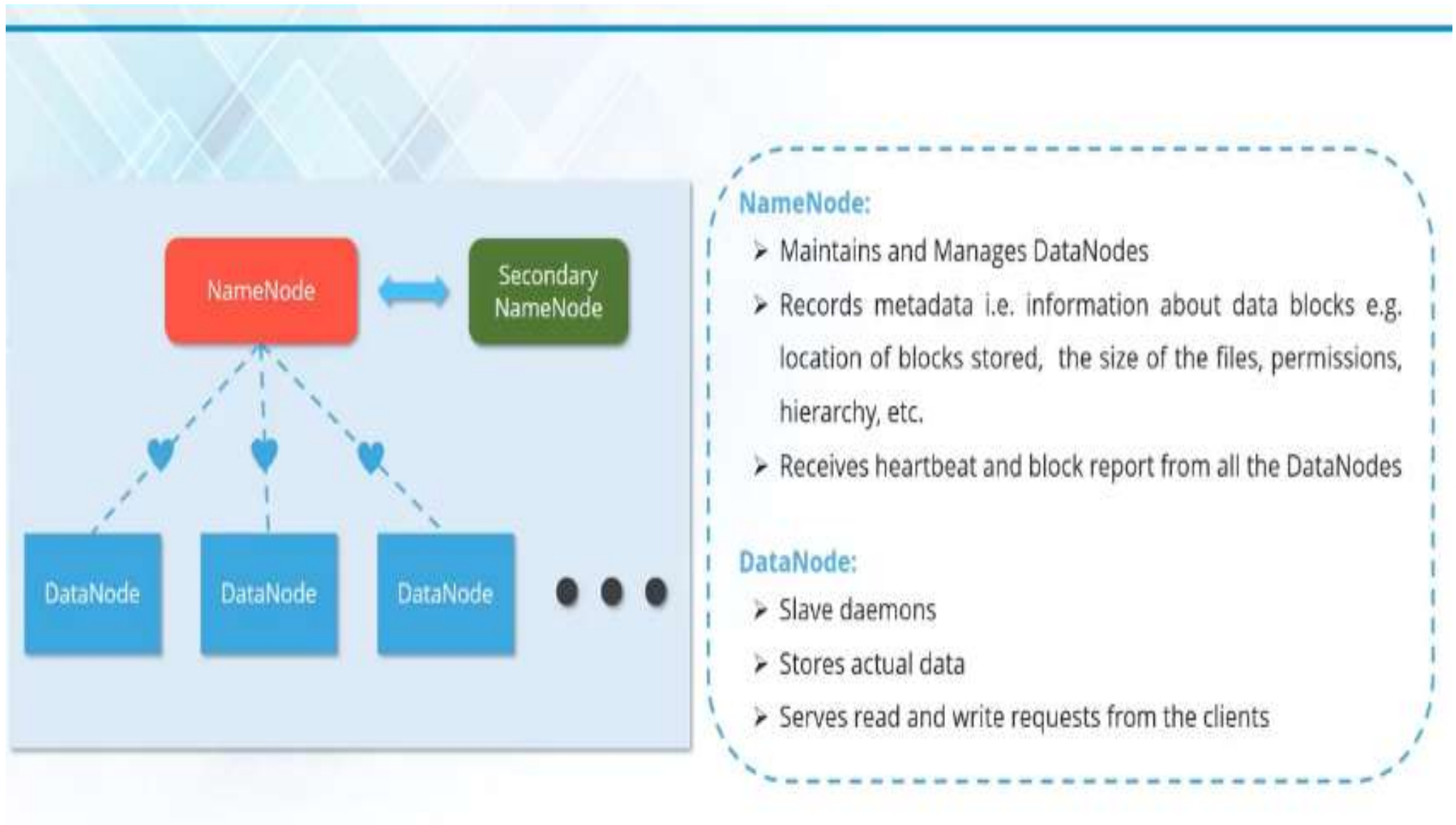


# 3 Core Components





# Namenode & Datanode



# Daemon Services



```
Terminal
shriramkv@shriramkv: ~/Desktop/OS Concepts
DS_2.mkv
shriramkv@shriramkv:~/Desktop/OS Concepts$ clear

shriramkv@shriramkv:~/Desktop/OS Concepts$ gedit Daemon.
shriramkv@shriramkv:~/Desktop/OS Concepts$ gedit Daemon.
Daemon.c      Daemon.c~      Daemon.odp
shriramkv@shriramkv:~/Desktop/OS Concepts$ gedit Daemon.c

^C
shriramkv@shriramkv:~/Desktop/OS Concepts$ gedit Daemon.c &
[1] 2419
shriramkv@shriramkv:~/Desktop/OS Concepts$ ps -axl
```

F	UID	PID	PPID	PRI	NI	VSZ	RSS	WCHAN	STAT	TTY	TIME	COMMAND
4	0	1	0	20	0	4544	2548	poll_s	Ss	?	0:01	/sbin/init
1	0	2	0	20	0	0	0	kthrea	S	?	0:00	[kthreadd]
1	0	3	2	20	0	0	0	smpboo	S	?	0:00	[ksoftirqd/0
1	0	4	2	20	0	0	0	worker	S	?	0:00	[kworker/0:0
1	0	5	2	0	-20	0	0	worker	S<	?	0:00	[kworker/0:0
1	0	6	2	20	0	0	0	worker	S	?	0:00	[kworker/u16
1	0	7	2	20	0	0	0	rcu_gp	S	?	0:00	[rcu_sched]



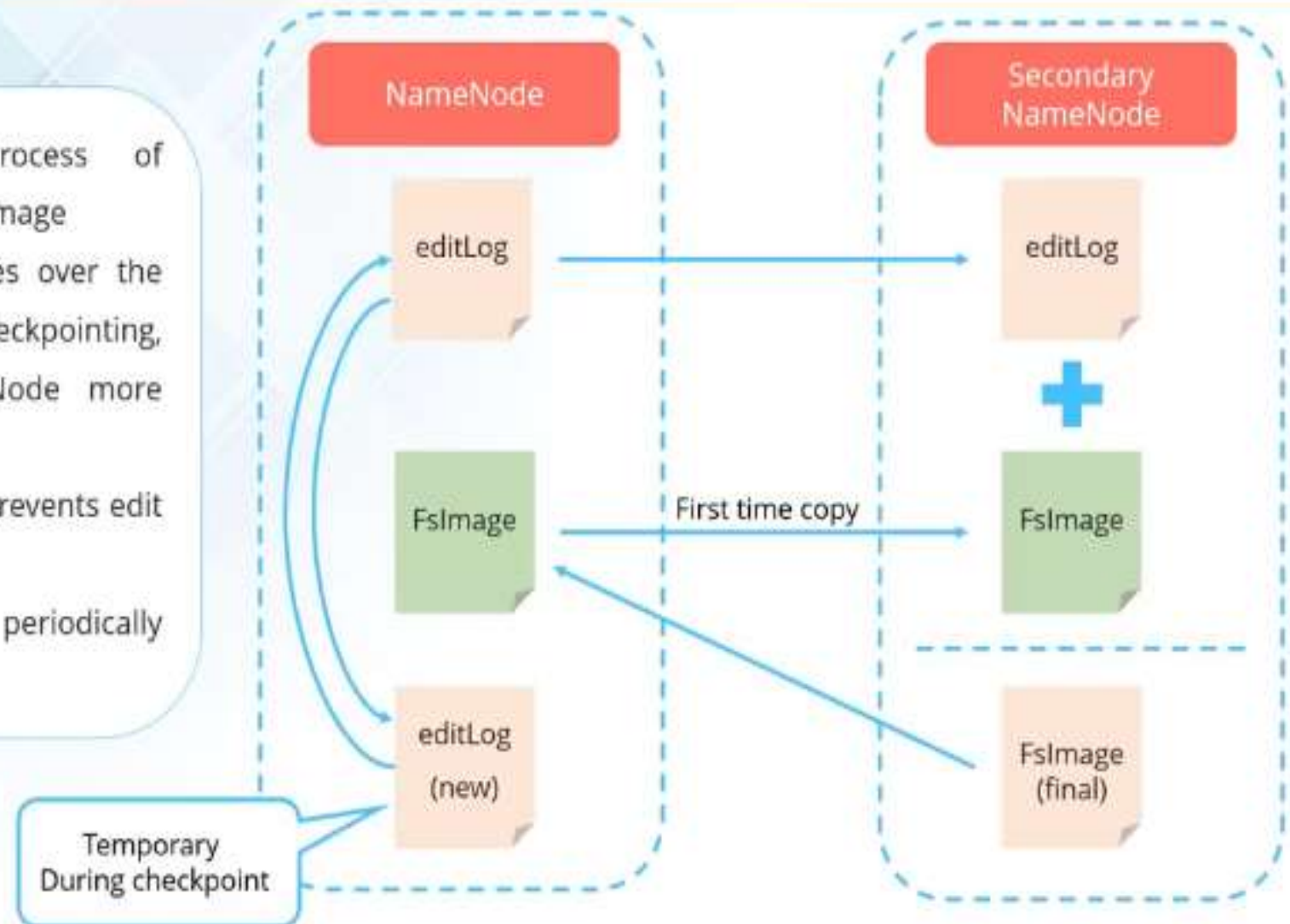
# About Daemon Process

---

- It is a program.
- Runs in Unix/Linux without interruption or user initiation to happen.
- Executed in the background.
- Recollect, orphans.
- No terminal usage
- Command to see the daemons
- `Ps -axl` (results will have ? Under TTY, it tells Daemon's existence)

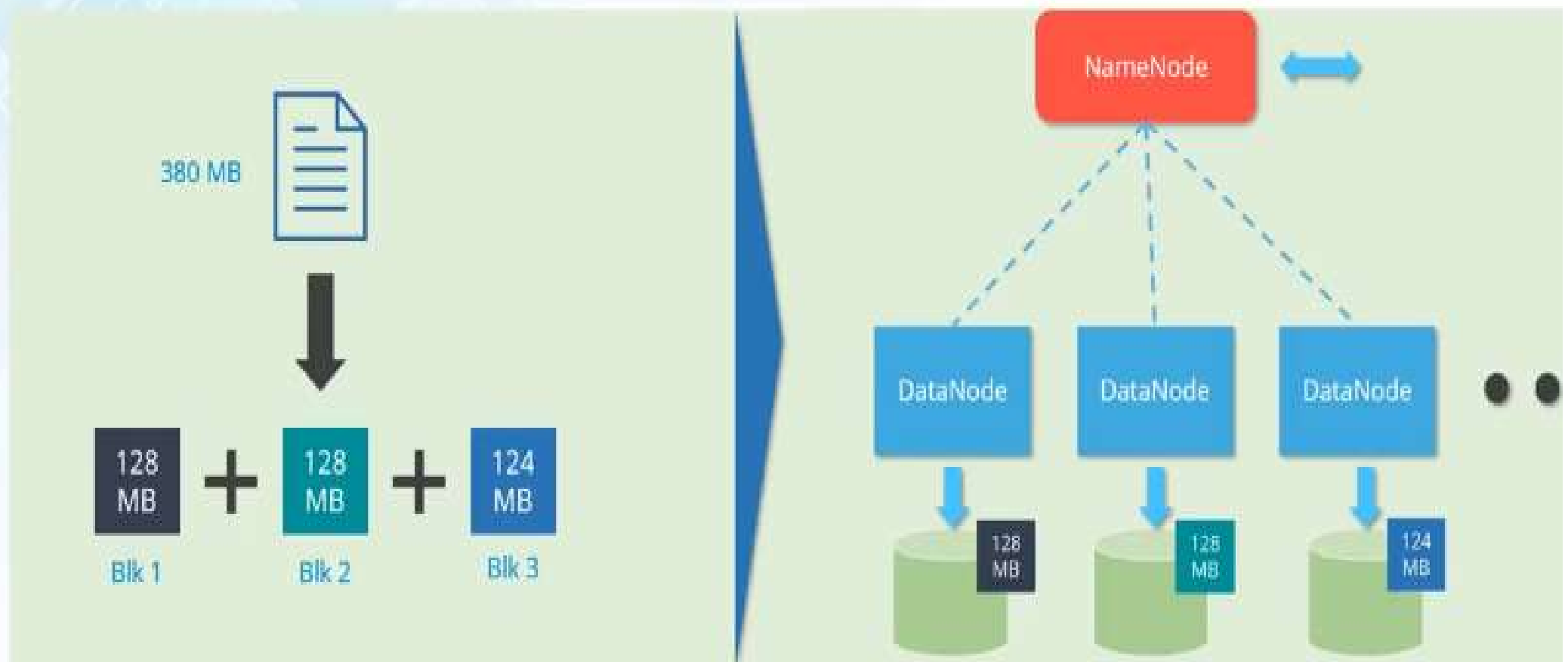
# Secondary NameNode & Checkpoint

- Checkpointing is a process of combining edit logs with FsImage
- Secondary NameNode takes over the responsibility of checkpointing, therefore, making NameNode more available
- Allows faster Failover as it prevents edit logs from getting too huge
- Checkpointing happens periodically (default: 1 hour)



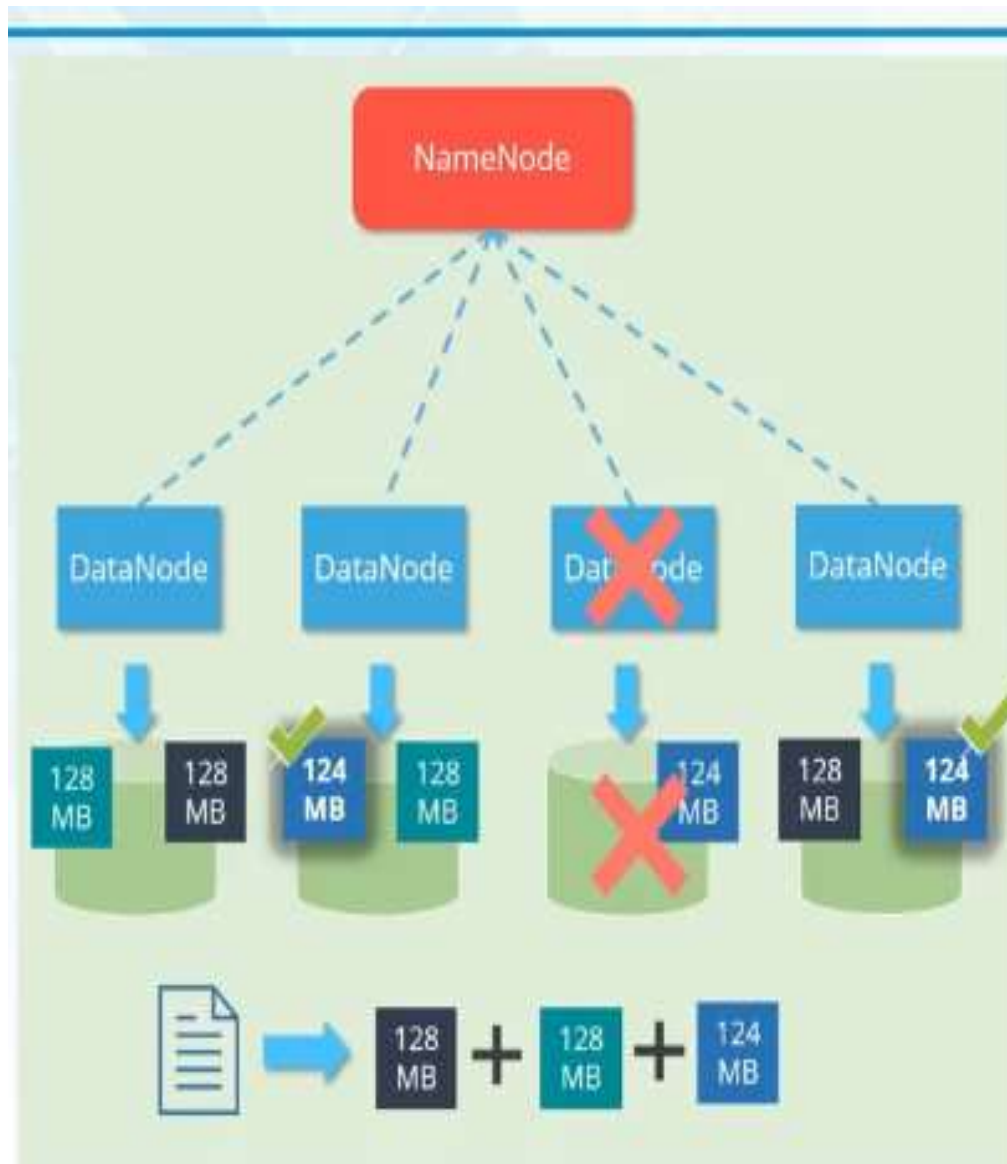
# HDFS DataNode Blocks

- Each file is stored on HDFS as blocks
- The default size of each block is 128 MB in Apache Hadoop 2.x (64 MB in Apache Hadoop 1.x)





# Replication



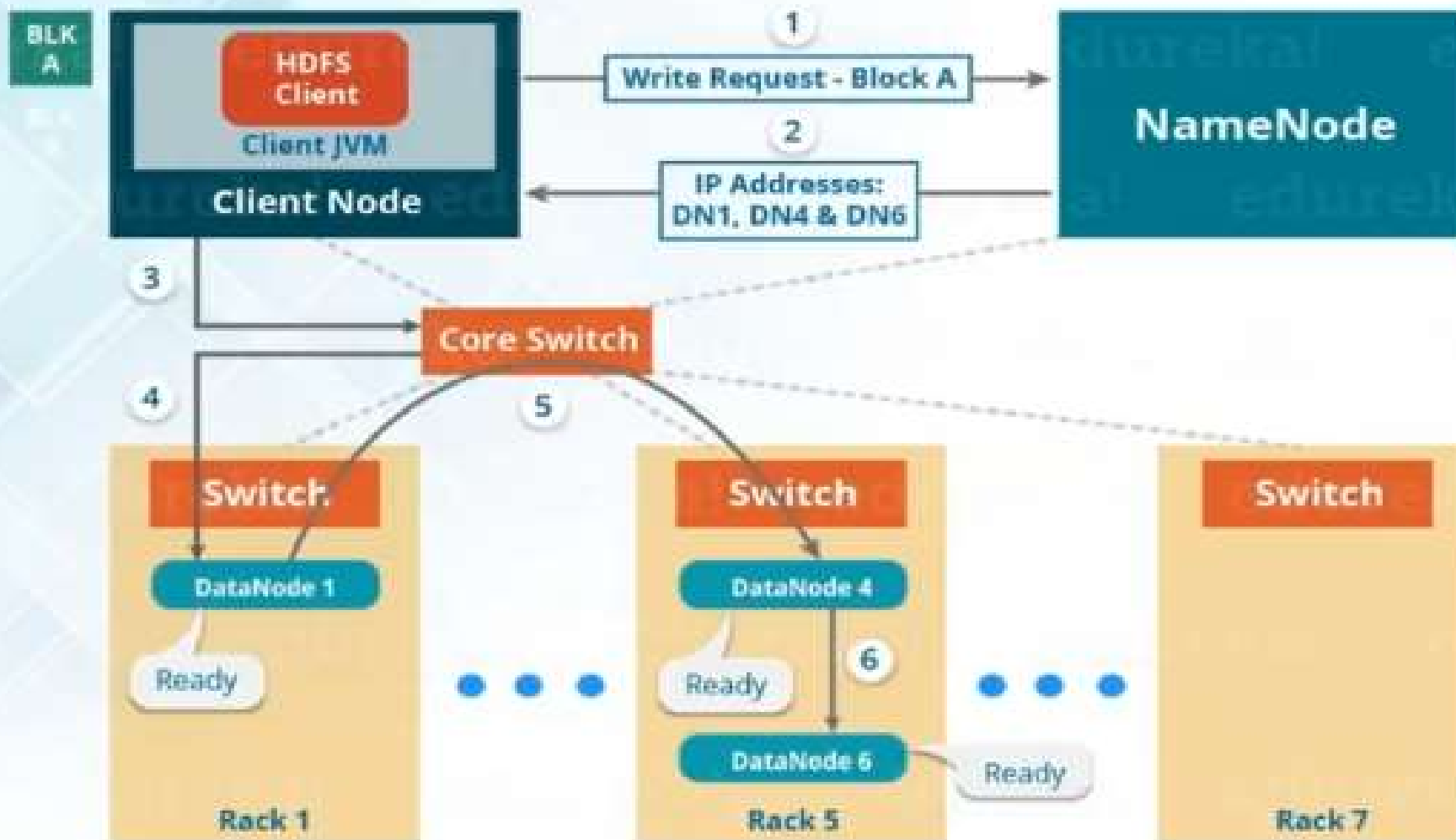
## Solution:

Each data blocks are replicated (thrice by default) and are distributed across different DataNodes

# HDFS Write



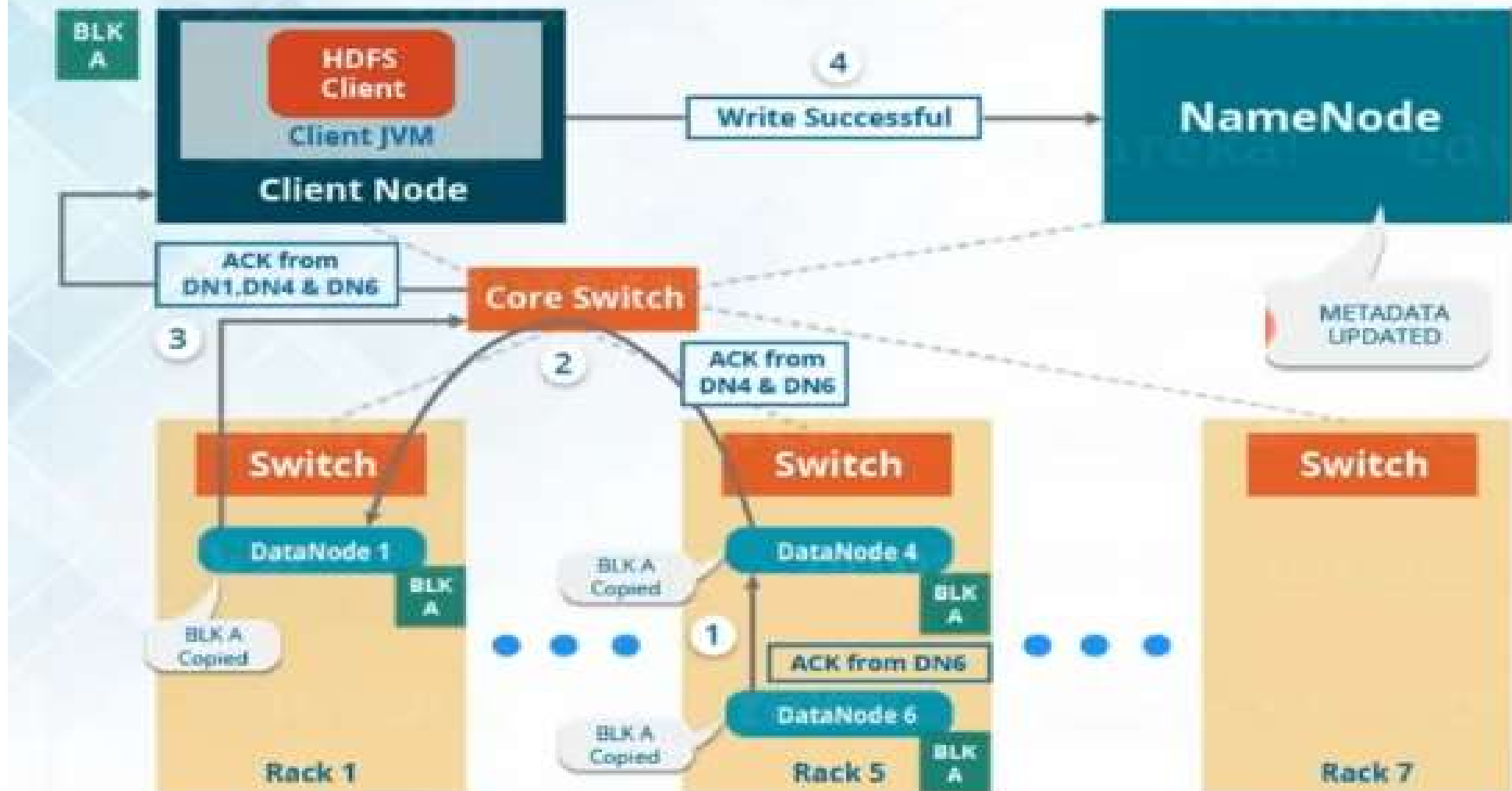
## Setting up HDFS - Write Pipeline



# HDFS Write - Ack



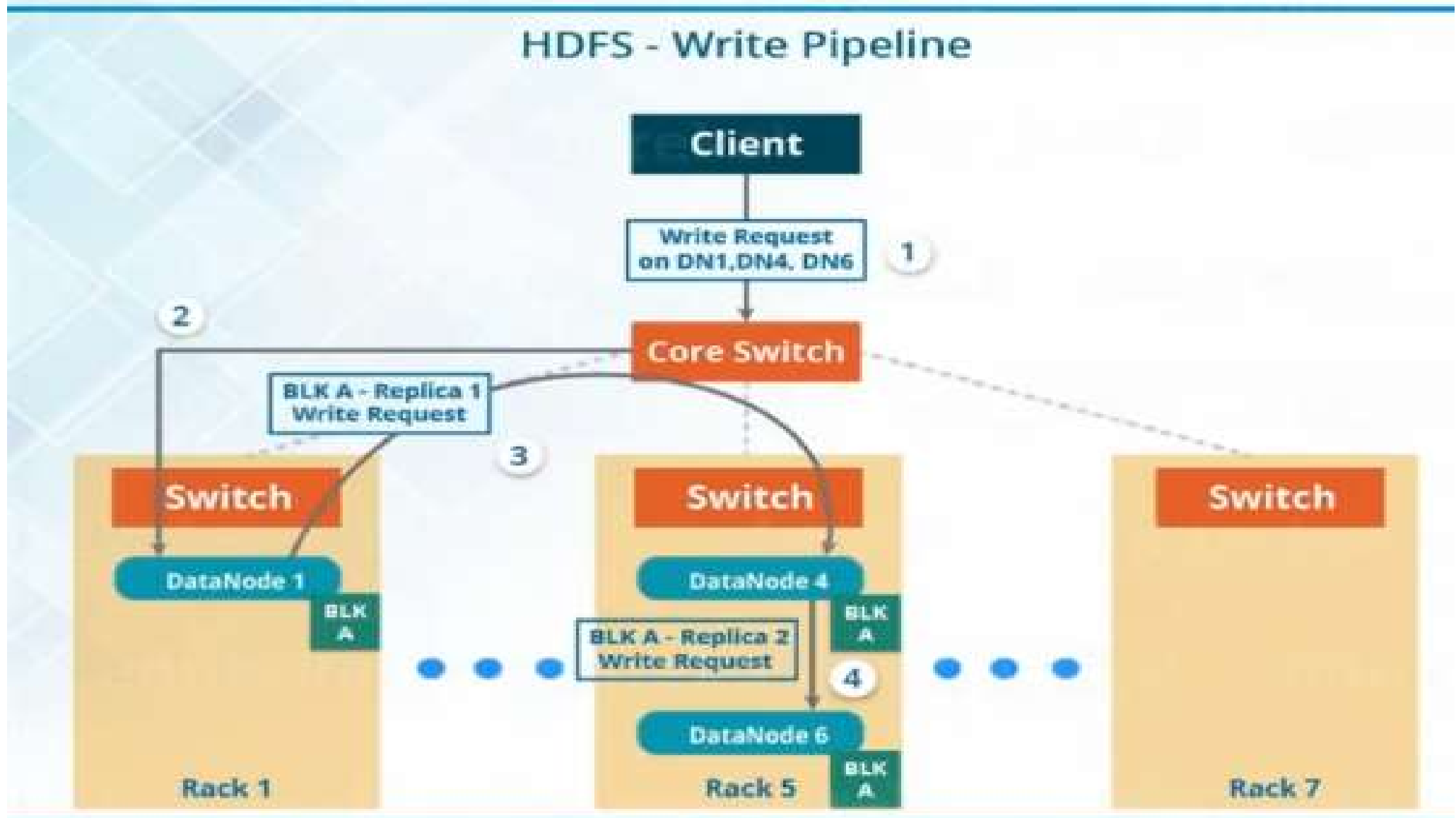
## Acknowledgement in HDFS - Write



# Write Request



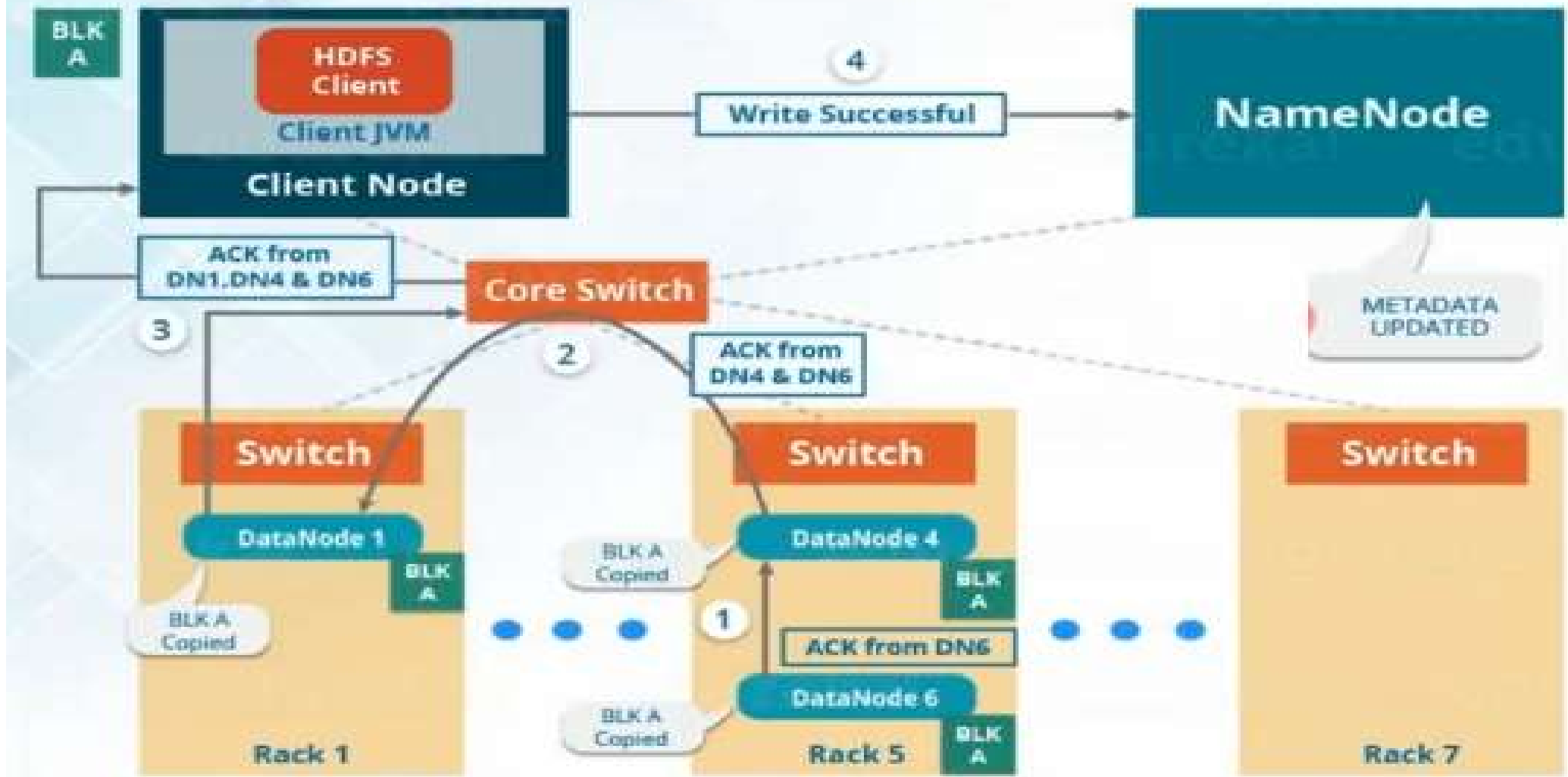
## HDFS - Write Pipeline



# Ack

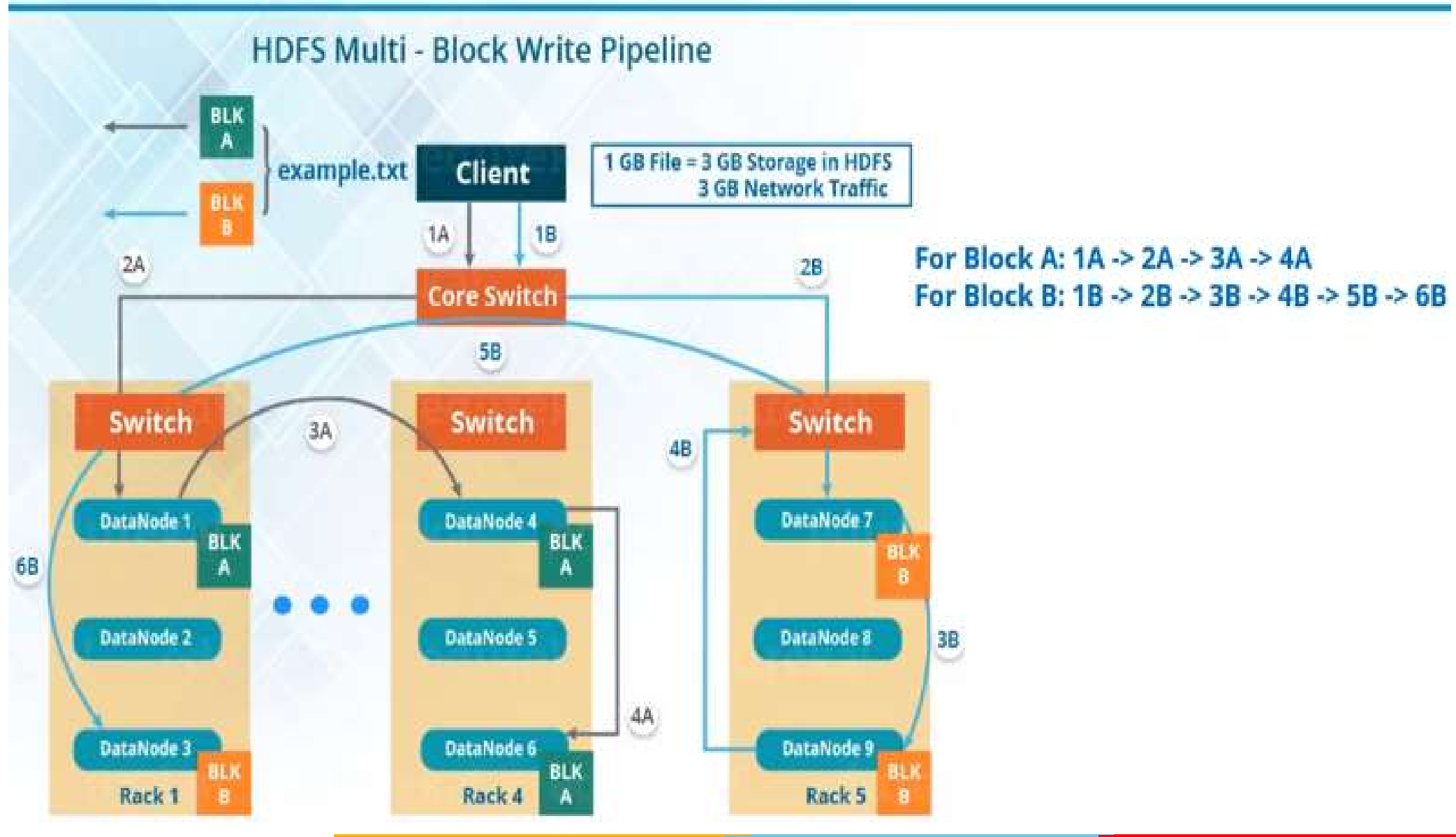


## Acknowledgement in HDFS - Write

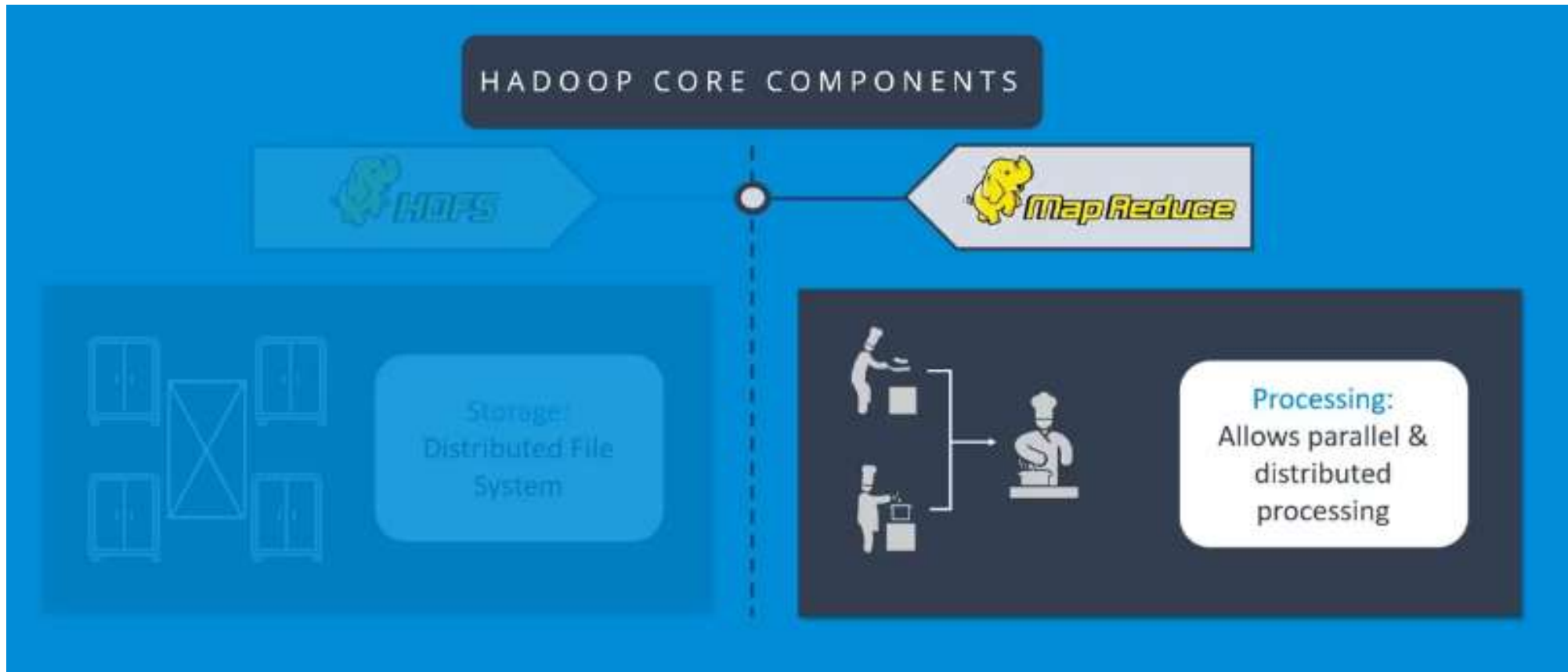




# Multi Data Block write



# MapReduce Layer



# Why Map-Reduce?



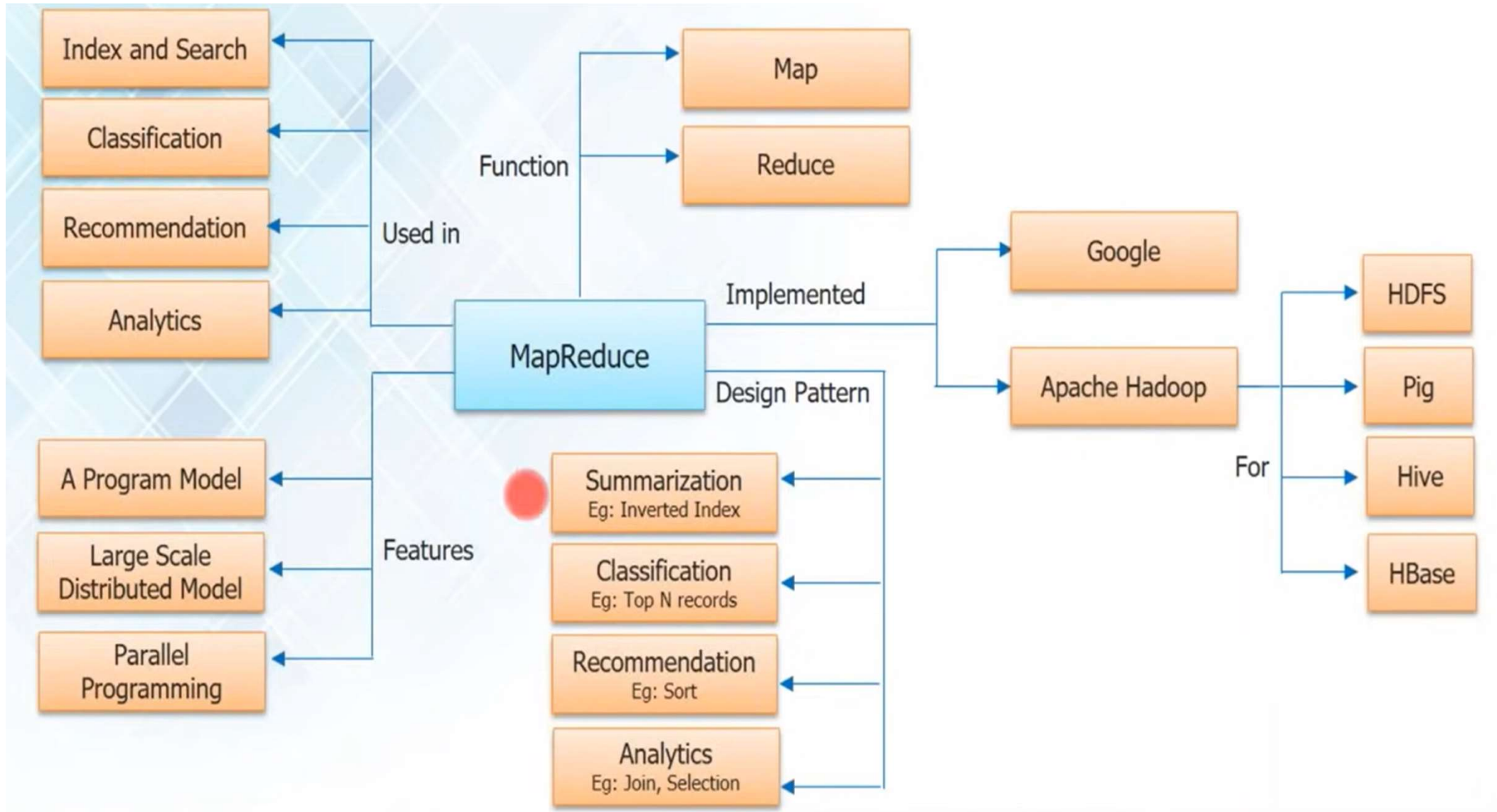
Process Big Data that resides in Hadoop HDFS is not stored in traditional fashion. Complete data is no more available at one place. So we need some processing mechanism that can go to all those locations where the data is available and process is required

Big Data



- Hadoop MapReduce is the processing component of Apache Hadoop
- It processes data parallelly in distributed environment

# Bird Eye View



# Parallel Processing scenario

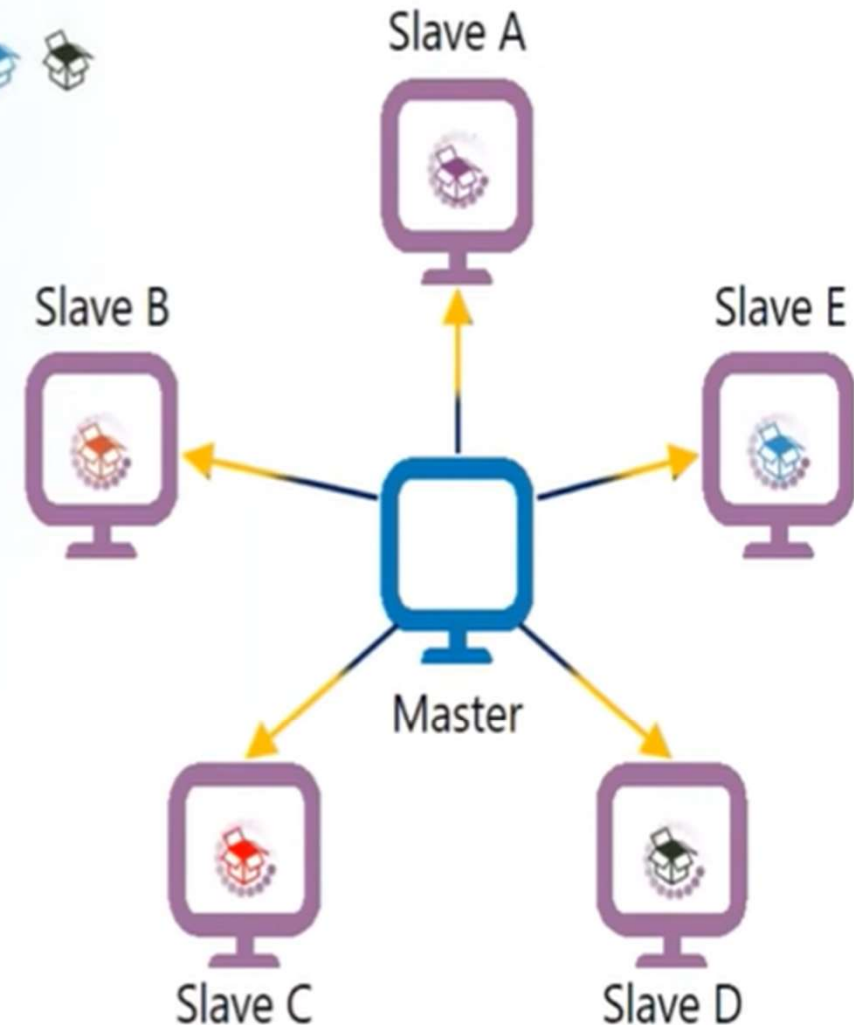


Data →

## Map Reduce Follow Data Locality Concept

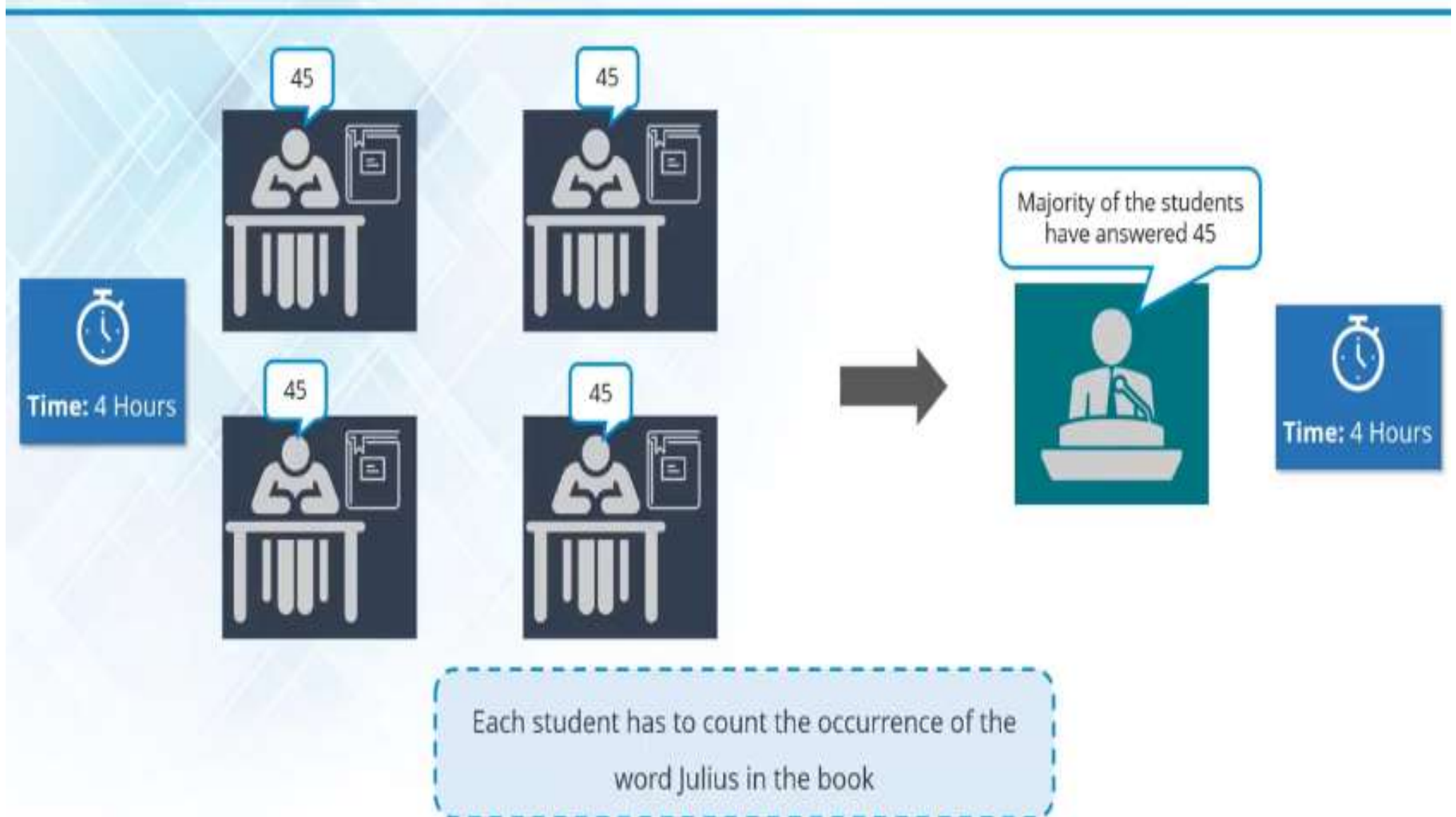
- Data is processed in parallel
- Processing becomes fast

**Moving the Processing Logic to  
slave machines.  
This saves lot of bandwidth**



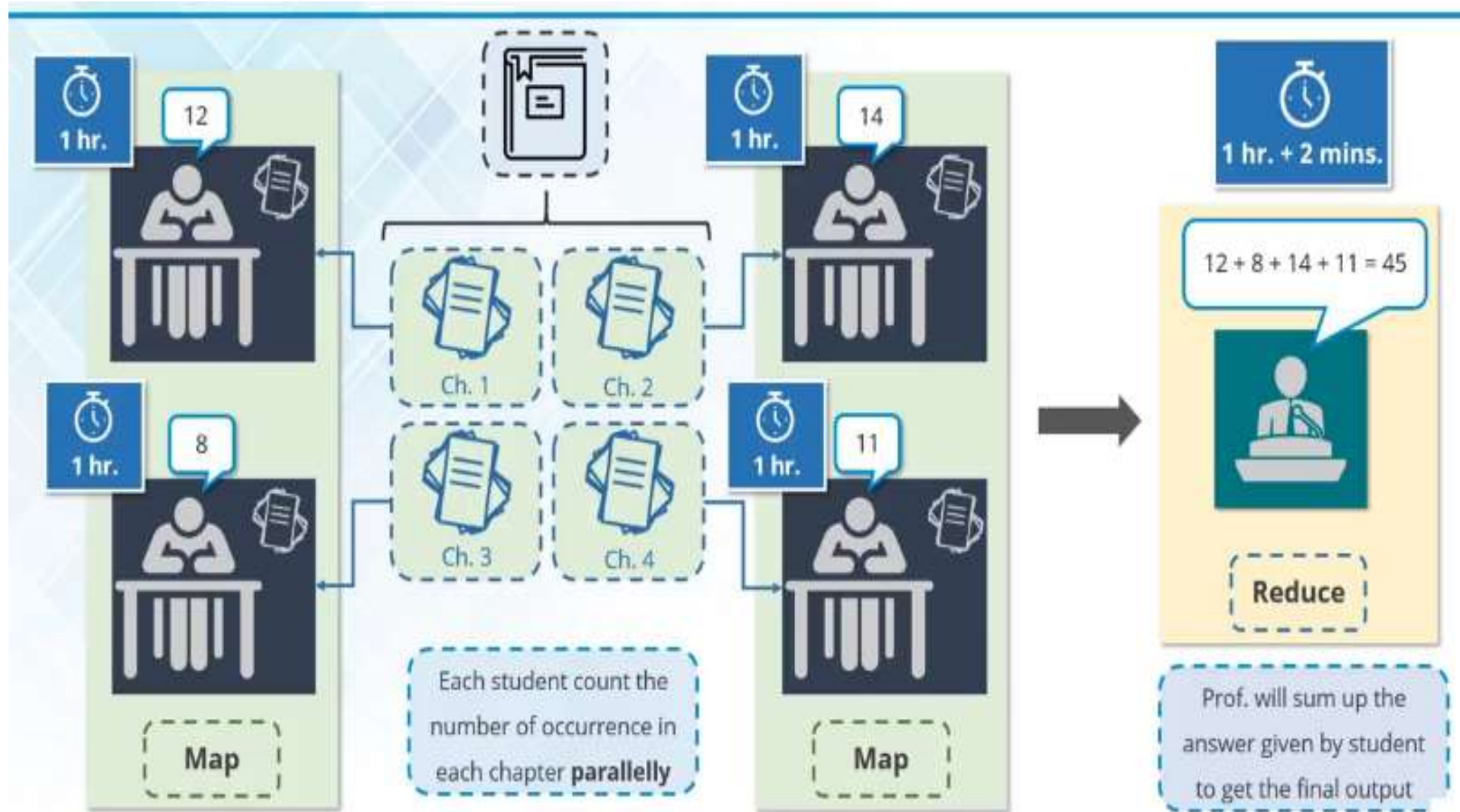


# Word Count in a Book – Normal mode



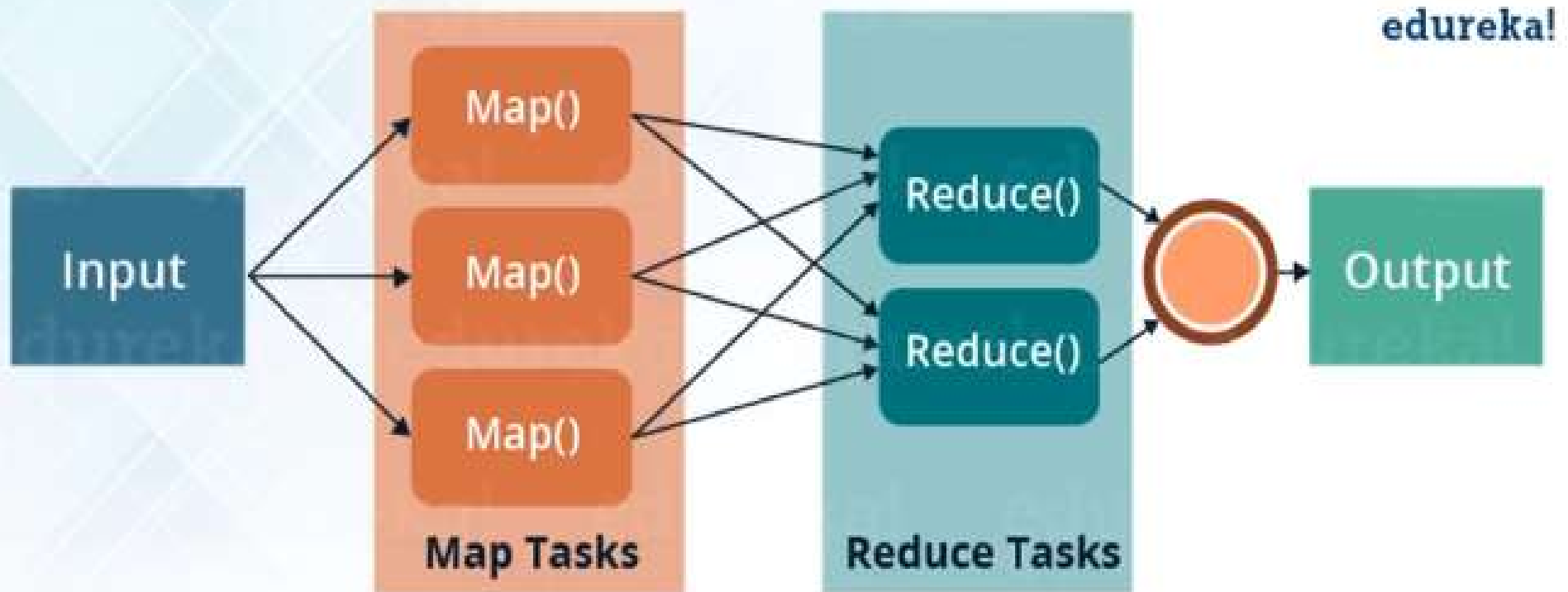


# MapReduce Operation - Example

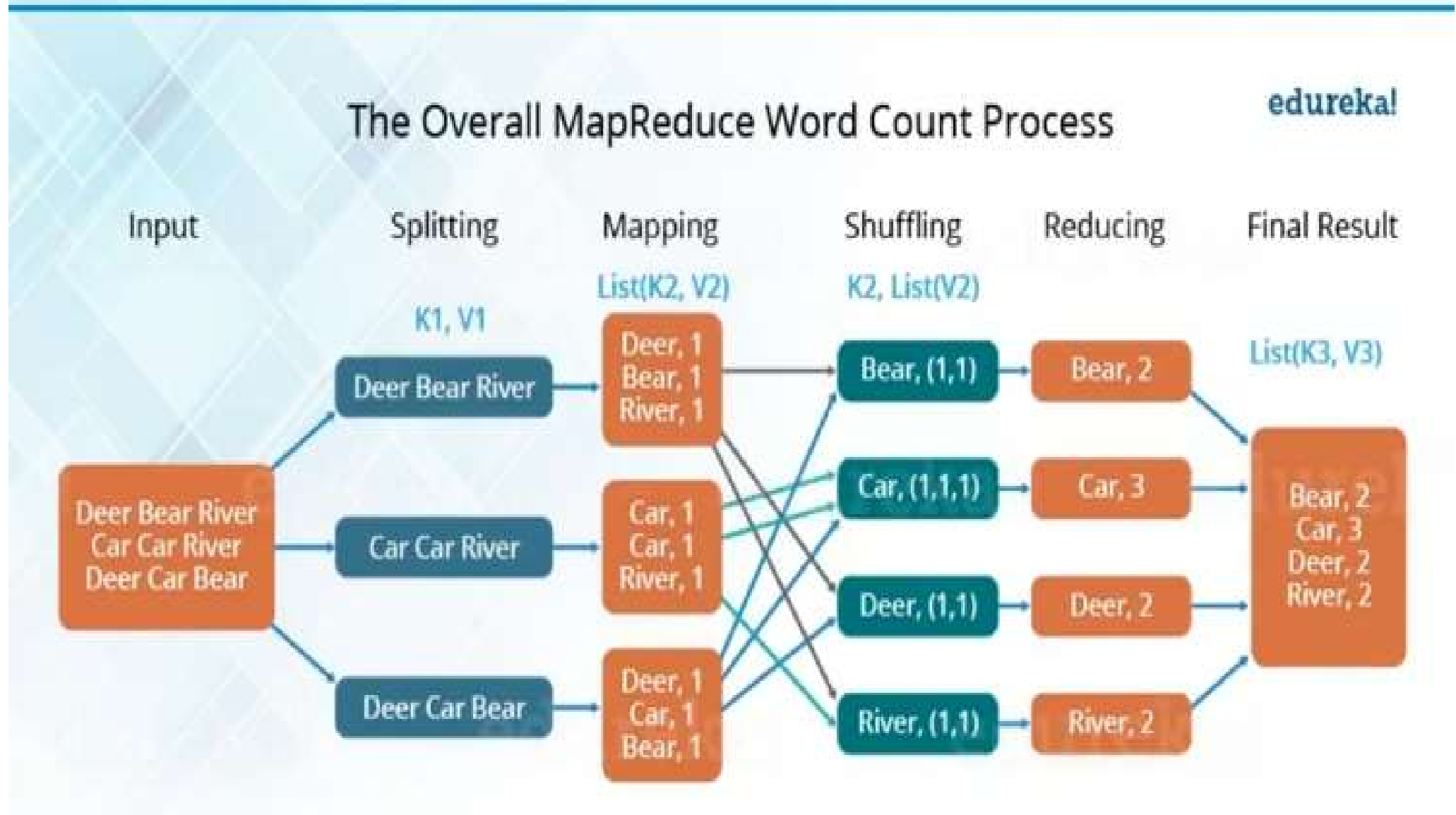


# Exact Flow - MapReduce

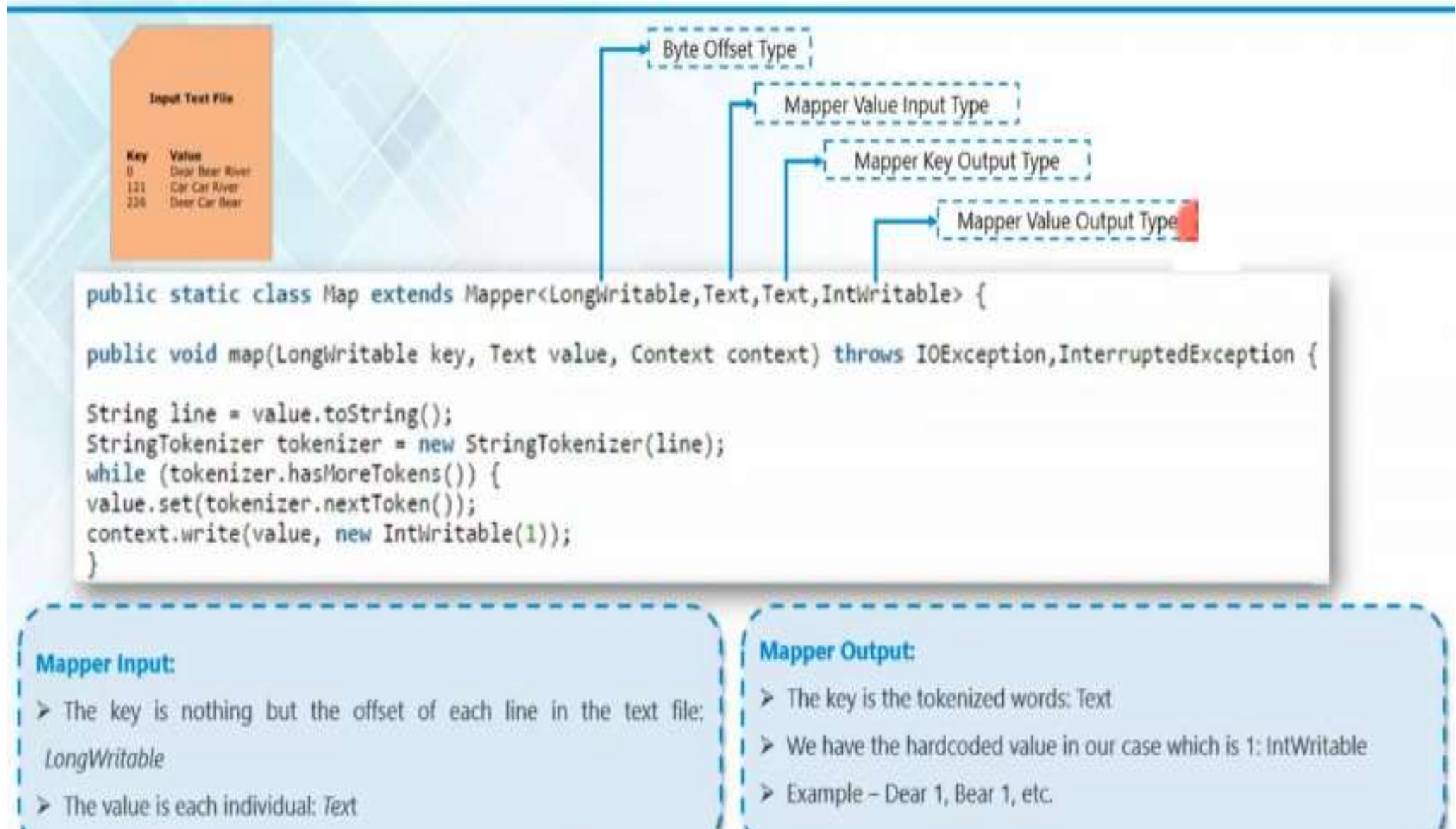
MapReduce is a **programming framework** that allows us to perform **distributed** and **parallel** processing on large data sets in a distributed environment



# Steps with example



# Mapper Code –Word Count Example





# Reducer Code –Word Count Example

Reducer Key Input Type

Reducer Value Input Type

Reducer Key Output Type

Reducer Value Output Type

```

public static class Reduce extends Reducer<Text,IntWritable,Text,IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values,Context context)
    throws IOException,InterruptedException {

        int sum=0;
        for(IntWritable x: values)
        {
            sum+=x.get();
        }
        context.write(key, new IntWritable(sum));
    }
}

```

**Reducer Input:**

- Keys are unique words which have been generated after the sorting and shuffling phase: Text
- The value is a list of integers corresponding to each key: IntWritable
- Example: Bear, [1, 1], etc.

**Reducer Output:**

- The key is all the unique words present in the input text file: Text
- The value is the number of occurrences of each of the unique words: IntWritable
- Example: Bear, 2; Car, 3, etc. .

# Driver Code for Configuration



In the driver class, we set the configuration of our MapReduce job to run in Hadoop

```
Configuration conf= new Configuration();
Job job = new Job(conf,"My Word Count Program");
job.setJarByClass(WordCount.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
Path outputPath = new Path(args[1]);

//Configuring the input/output path from the filesystem into the job
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

- Specify the name of the job, the data type of input/output of the mapper and reducer
- Specify the names of the mapper and reducer classes.
- Path of the input and output folder
- The method `setInputFormatClass()` is used for specifying the unit of work for mapper
- `Main()` method is the entry point for the driver



# Sample Text File



The screenshot displays a Linux desktop environment with three main windows:

- Text Editor:** A window titled 'text' showing the following content:

```
Dear Bear River  
Car Car River  
Dear Car River
```
- Terminal:** A terminal window showing the execution of Hadoop commands:

```
[edureka@localhost ~]$ cd Desktop/  
[edureka@localhost Desktop]$ hadoop fs -put text /
```
- Web Browser:** A Mozilla Firefox window titled 'Namenode information - Mozilla Firefox' showing the Hadoop Namenode overview page. The address bar indicates the URL `localhost:50070/dfshealth.html#tab=overview`. The page features a green navigation bar with the following links: **Hadoop**, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities.

Below the browser window, the text 'Hadoop, 2017.' is visible.

# NameNode Summary



Heap Memory used 36.64 MB of 119.94 MB Heap Memory. Max Heap Memory is 966.69 MB.	
Non Heap Memory used 55.88 MB of 56.96 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.	
Configured Capacity:	35.66 GB
DFS Used:	18.16 MB (0.05%)
Non DFS Used:	9.32 GB
DFS Remaining:	26.33 GB (73.83%)
Block Pool Used:	18.16 MB (0.05%)
DataNodes usages% (Min/Median/Max/stdDev):	0.05% / 0.05% / 0.05% / 0.00%
Live Nodes	1 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	12

# Live Node Details

Namenode information

localhost:50070/dfshealth.html#tab-datanode

Search

In operation

Show 25 entries

Search

Node	Http Address	Last contact	Capacity	Blocks	Block pool used	Version
✓localhost:50010 (127.0.0.1:50010)	localhost:50075	1s	35.66 GB <div> <div></div> </div>	19	18.17 MB (0.05%)	2.8.1

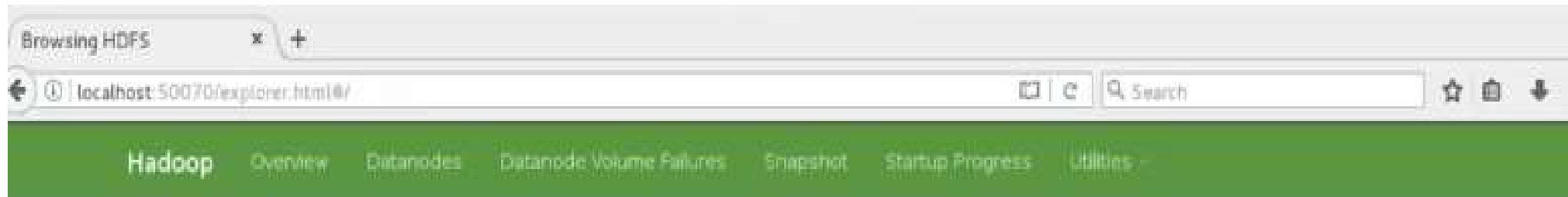
Showing 1 to 1 of 1 entries

Previous

1

Next

# NameNode Files



## Browse Directory

/

Go!

Show 25 entries

Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	498.2 KB	Dec 31 15:01	1	128 MB	olympix_data.csv
-rw-r--r--	edureka	supergroup	45 B	Jan 03 16:26	1	128 MB	text
drwxr-xr-x	edureka	supergroup	0 B	Oct 17 09:53	0	0 B	system
drwxrwxrwx	root	supergroup	0 B	Dec 31 14:58	0	0 B	tmp

# Execution



```
File Edit View Search Terminal Help
[edureka@localhost Desktop]$ hadoop jar abc.jar WordCount /text /output
```

I

```
File Edit View Search Terminal Help
Job Counters
  Launched map tasks=1
  Launched reduce tasks=1
  Data-local map tasks=1
  Total time spent by all maps in occupied slots (ms)=10030
  Total time spent by all reduces in occupied slots (ms)=11901
  Total time spent by all map tasks (ms)=10030
  Total time spent by all reduce tasks (ms)=11901
  Total vcore-milliseconds taken by all map tasks=10030
  Total vcore-milliseconds taken by all reduce tasks=11901
  Total megabyte-milliseconds taken by all map tasks=10270720
  Total megabyte-milliseconds taken by all reduce tasks=12186624
Map-Reduce Framework
  Map input records=3
  Map output records=9
  Map output bytes=81
  Map output materialized bytes=105
  Input split bytes=91
  Combine input records=0
  Combine output records=0
```

# Result



Browsing HDFS

localhost:50070/explorer.html#/

/

Show 25 entries

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	498.2 KB	Dec 31 15:01	1	128 MB	olympix_data.csv
-rw-r--r--	edureka	supergroup	45 B	Jan 03 16:26	1	128 MB	text
drwxr-xr-x	edureka	supergroup	0 B	Jan 03 16:34	0	0 B	output
drwxr-xr-x	edureka	supergroup	0 B	Oct 17 09:53	0	0 B	tmp
drwxrwxrwx	root	supergroup	0 B	Dec 31 14:58	0	0 B	tmp

Browsing HDFS

localhost:50070/explorer.html#/output

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

## Browse Directory

/output

Show 25 entries

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	edureka	supergroup	0 B	Jan 03 16:34	1	128 MB	_SUCCESS
-rw-r--r--	edureka	supergroup	28 B	Jan 03 16:34	1	128 MB	part-r-00000



# Result-Contd



File Edit View Search Terminal Help

```
[edureka@localhost Desktop]$ hadoop fs -cat /output/part-r-000000
```

```
Bear 1
```

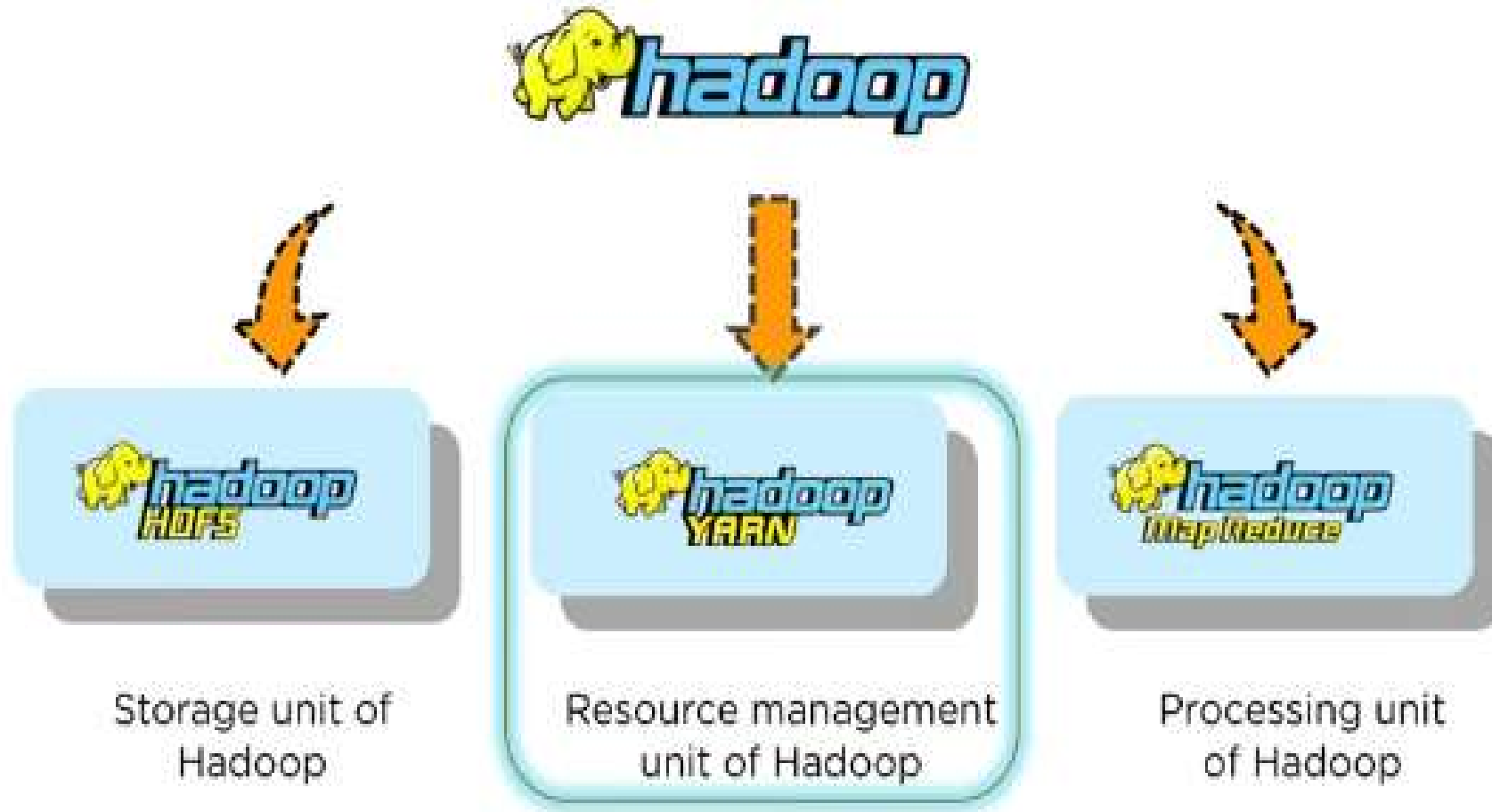
```
Car 3
```

```
Dear 2
```

```
River 3
```

```
[edureka@localhost Desktop]$
```

# Basic Hadoop-2 Architecture



# YARN



---

## YET ANOTHER RESOURCE NEGOTIATOR

Coming Up.....



---

That is all for the day

Thank you