BDM1213 Data Encoding Principles

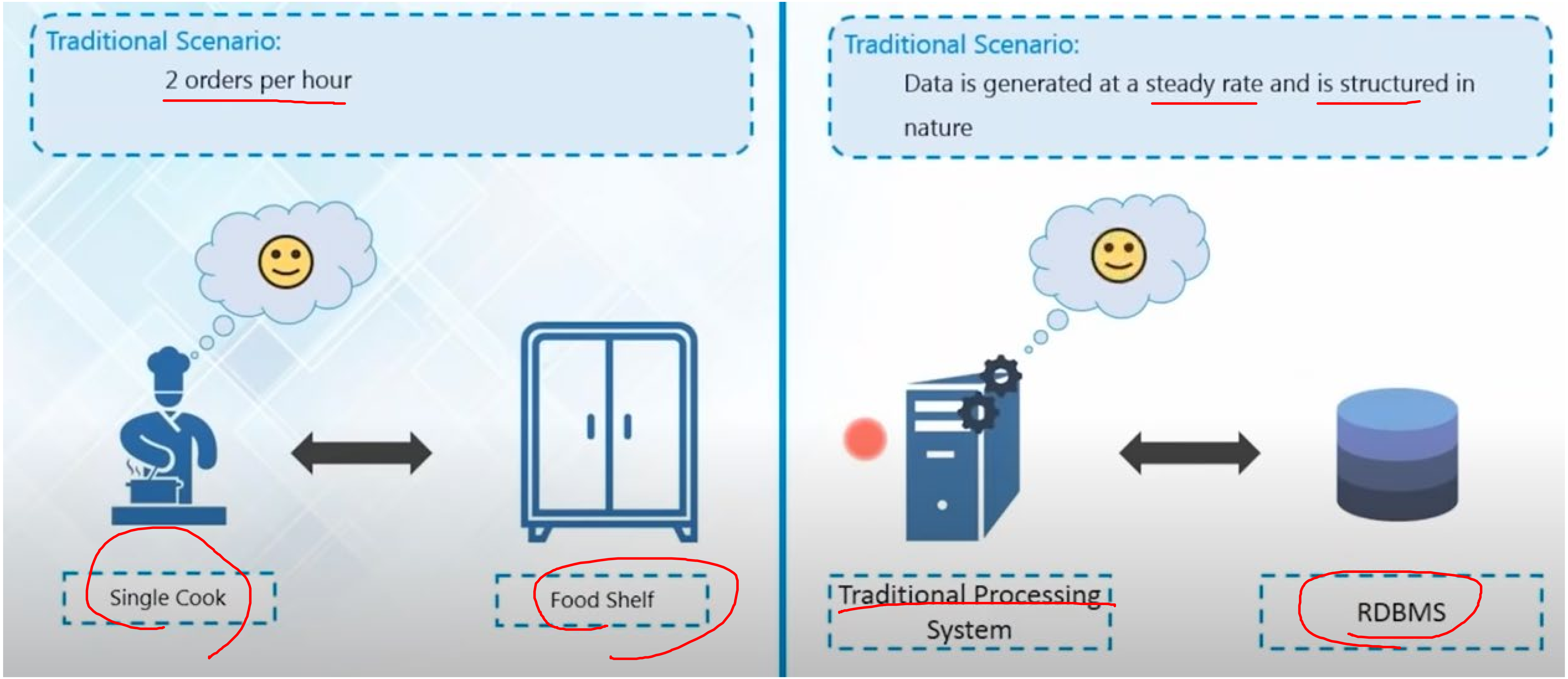
Week 03: Big data and Apache Hadoop

Dr. James Hong

Scenario:

Bob has opened a small restaurant in his city

• RESTAURANT •
MENU



Traditional Scenario:

2 orders per hour

Single Cook

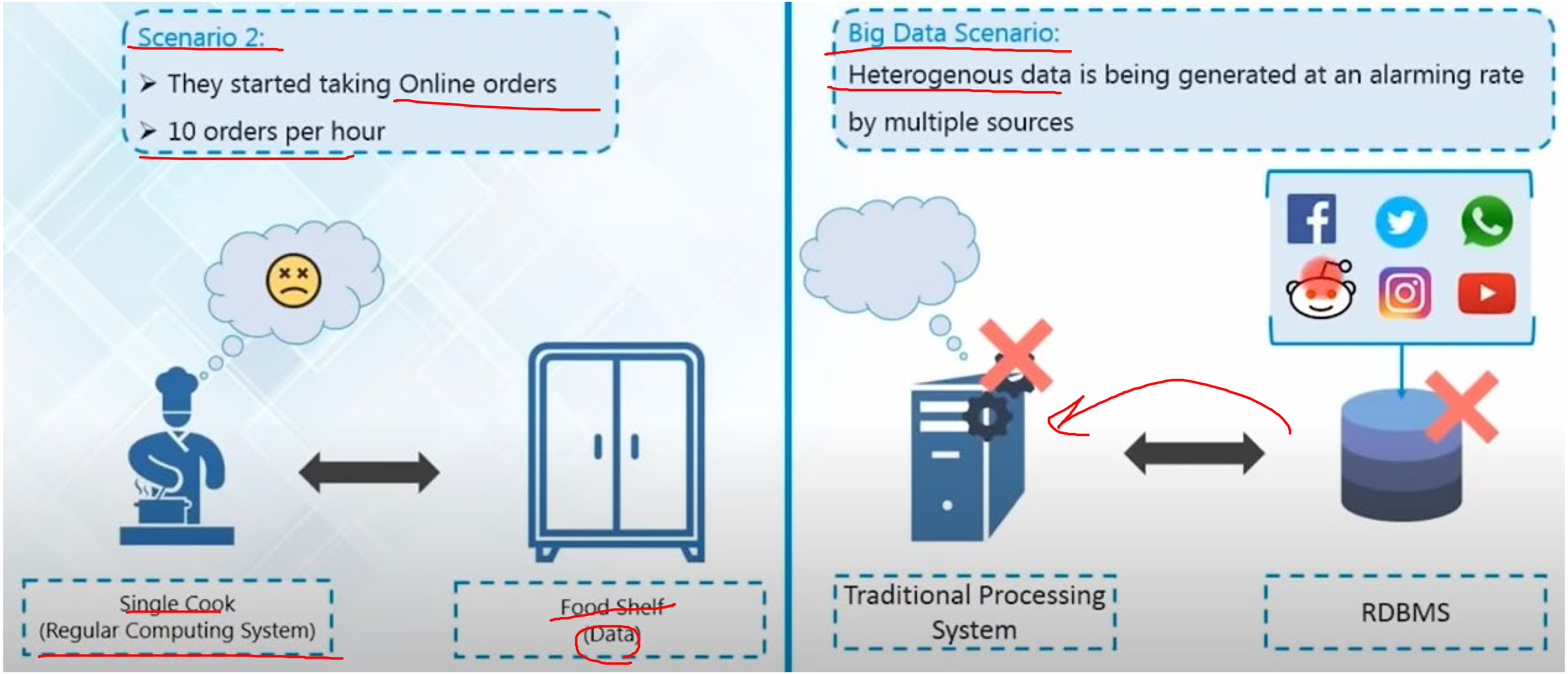
Food Shelf

Traditional Scenario:

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

Traditional Processing System

RDBMS



Scenario 2:

- ➢ They started taking Online orders
- ➢ 10 orders per hour

Single Cook
(Regular Computing System)

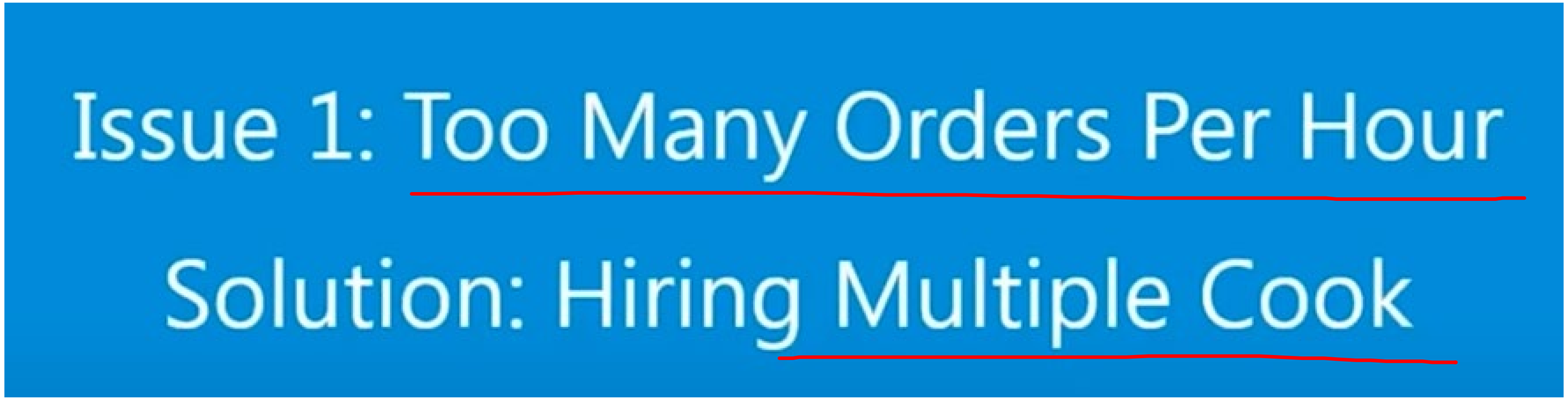
Food Shelf
(Data)

Big Data Scenario:

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

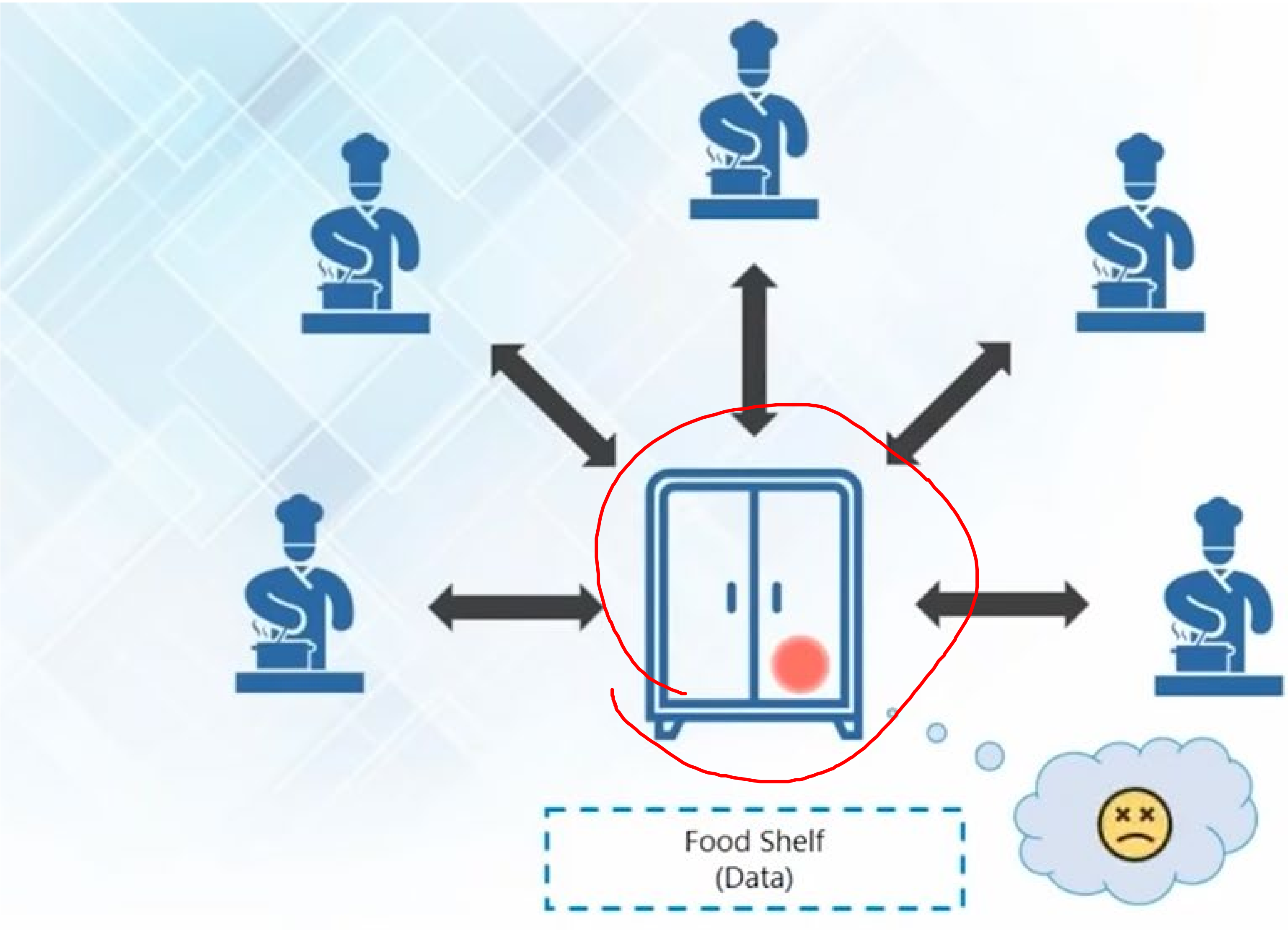
Traditional Processing System

RDBMS



Issue 1: Too Many Orders Per Hour

Solution: Hiring Multiple Cook

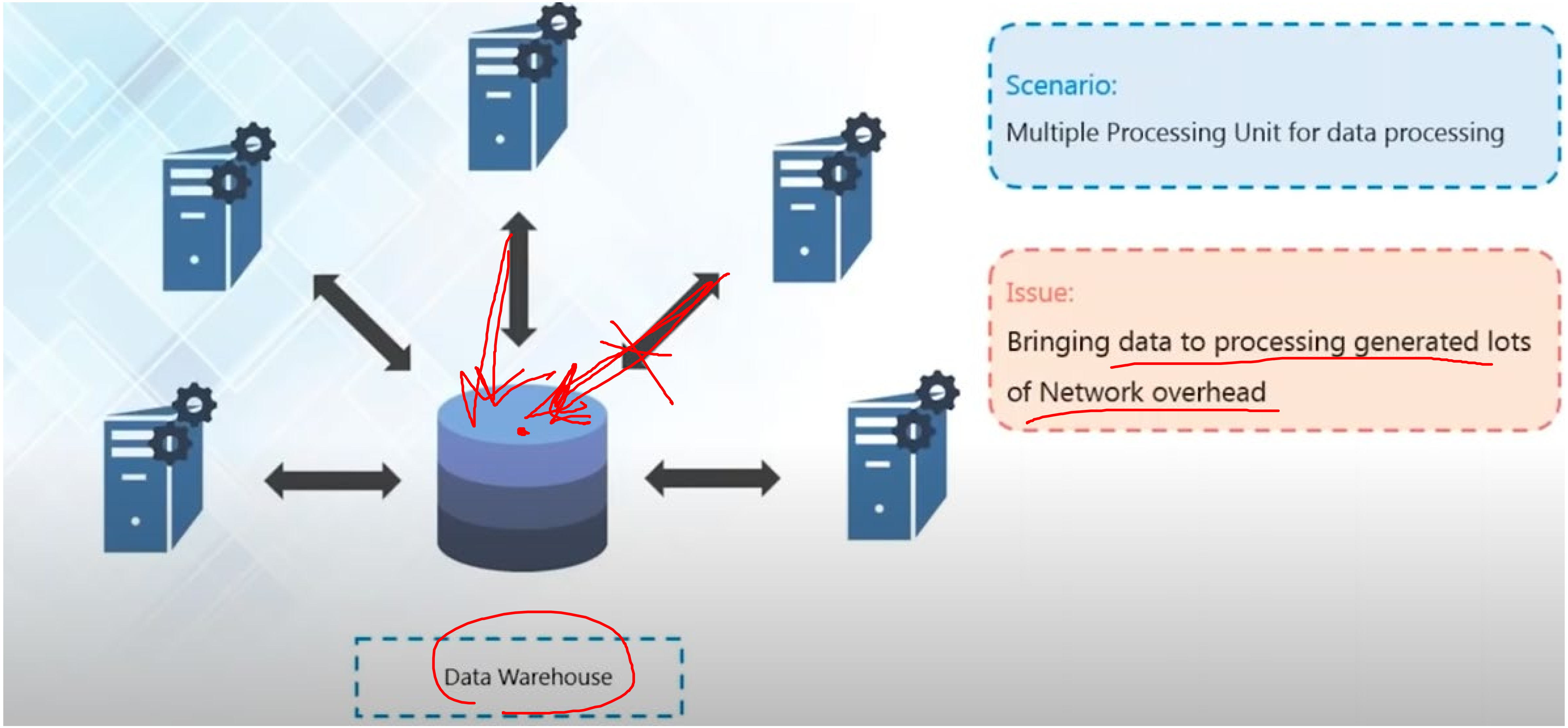


Scenario:

Multiple Cook cooking food

Issue:

Food Shelf becomes the BOTTLENECK

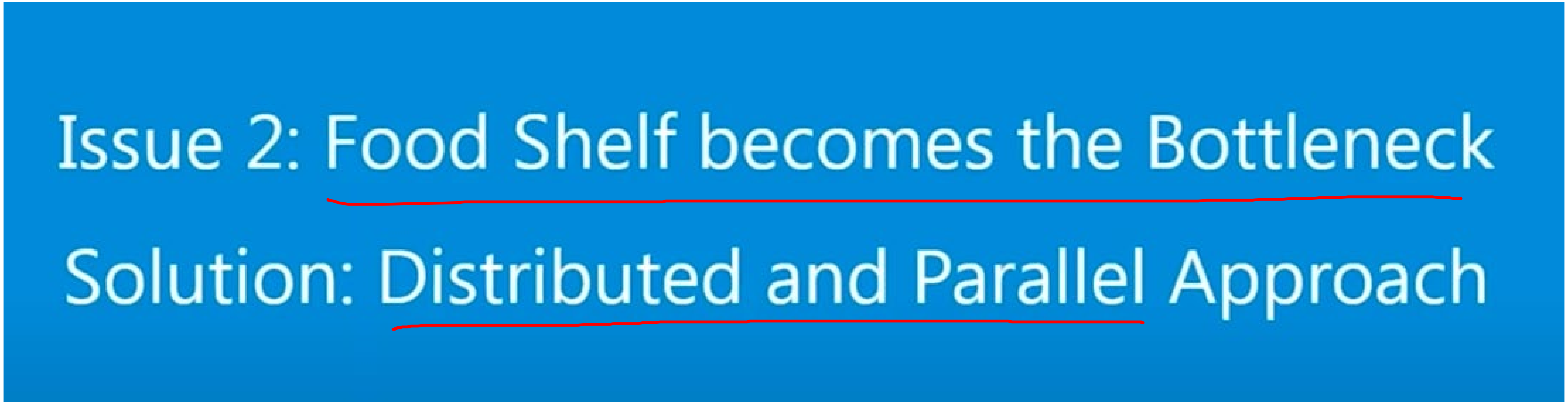


Scenario:

Multiple Processing Unit for data processing

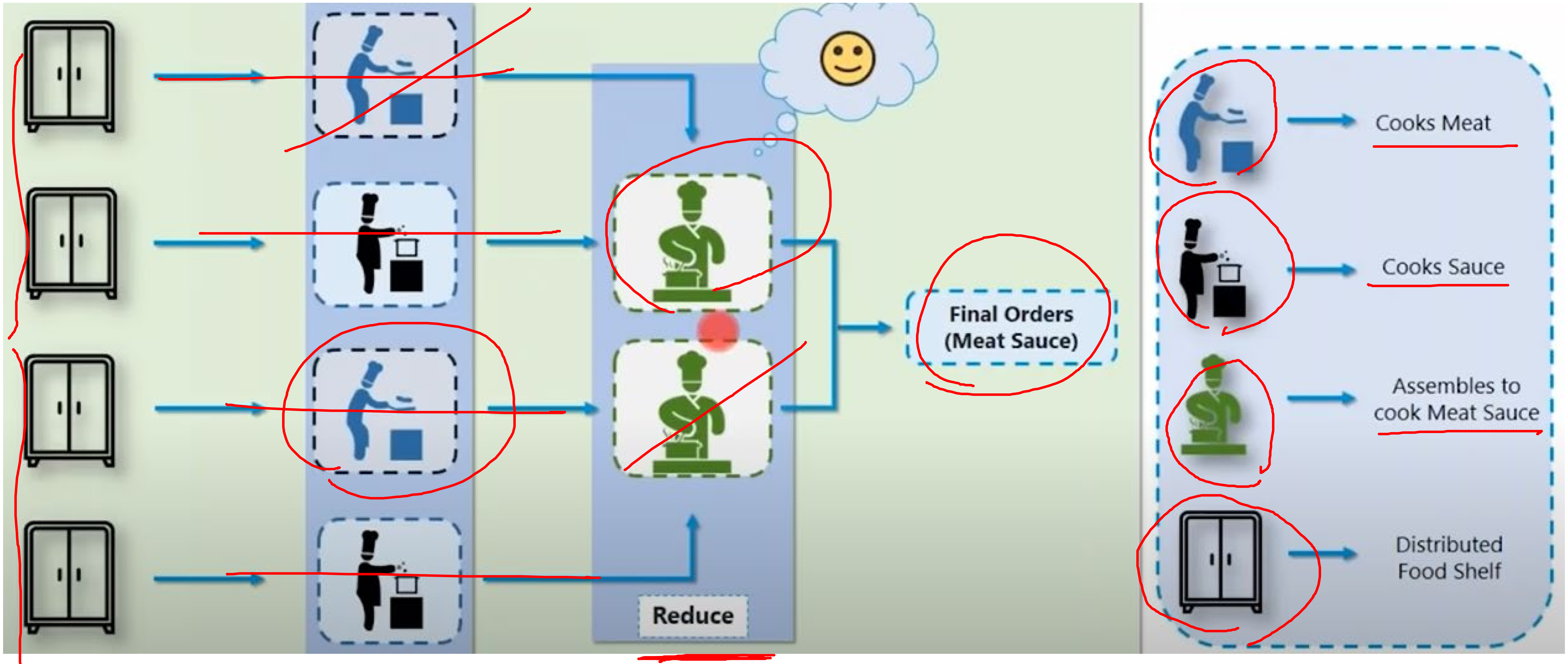
Issue:

Bringing data to processing generated lots of Network overhead

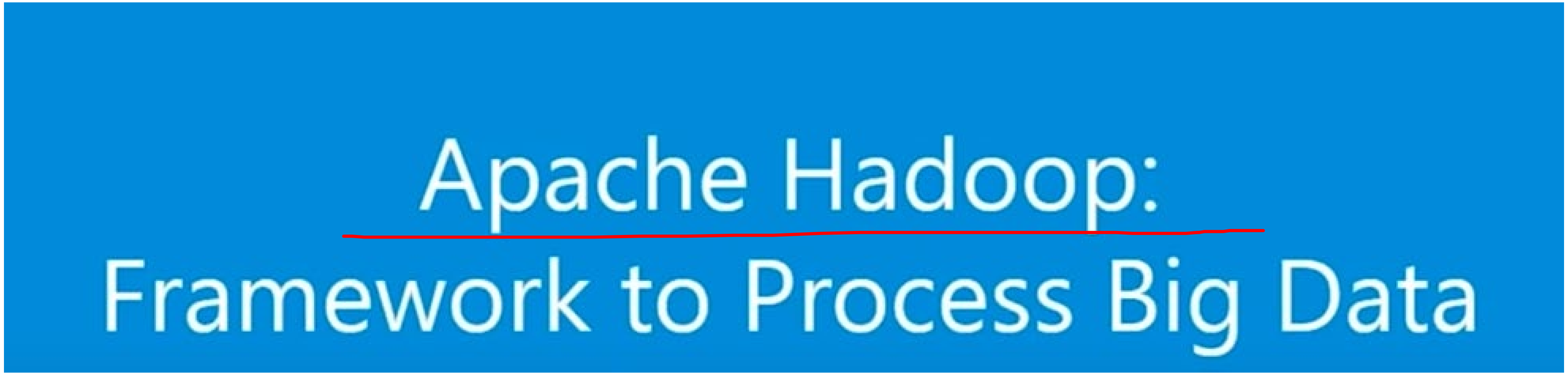


Issue 2: Food Shelf becomes the Bottleneck

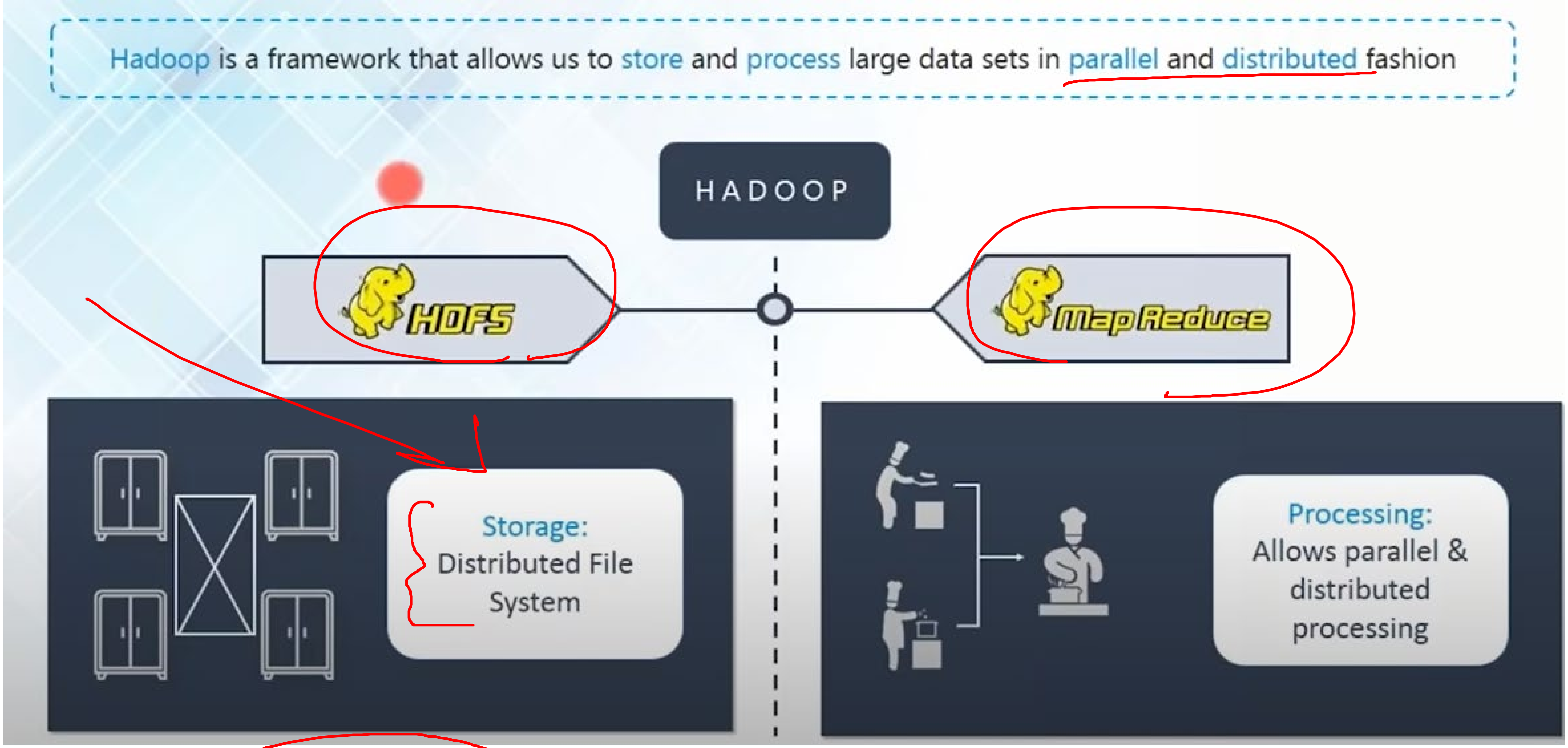
Solution: Distributed and Parallel Approach


Reduce
Final Orders
(Meat Sauce)
Cooks Meat
Cooks Sauce
Assembles to
cook Meat Sauce
Distributed
Food Shelf

Map



Apache Hadoop:
Framework to Process Big Data



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

HADOOP

HDFS

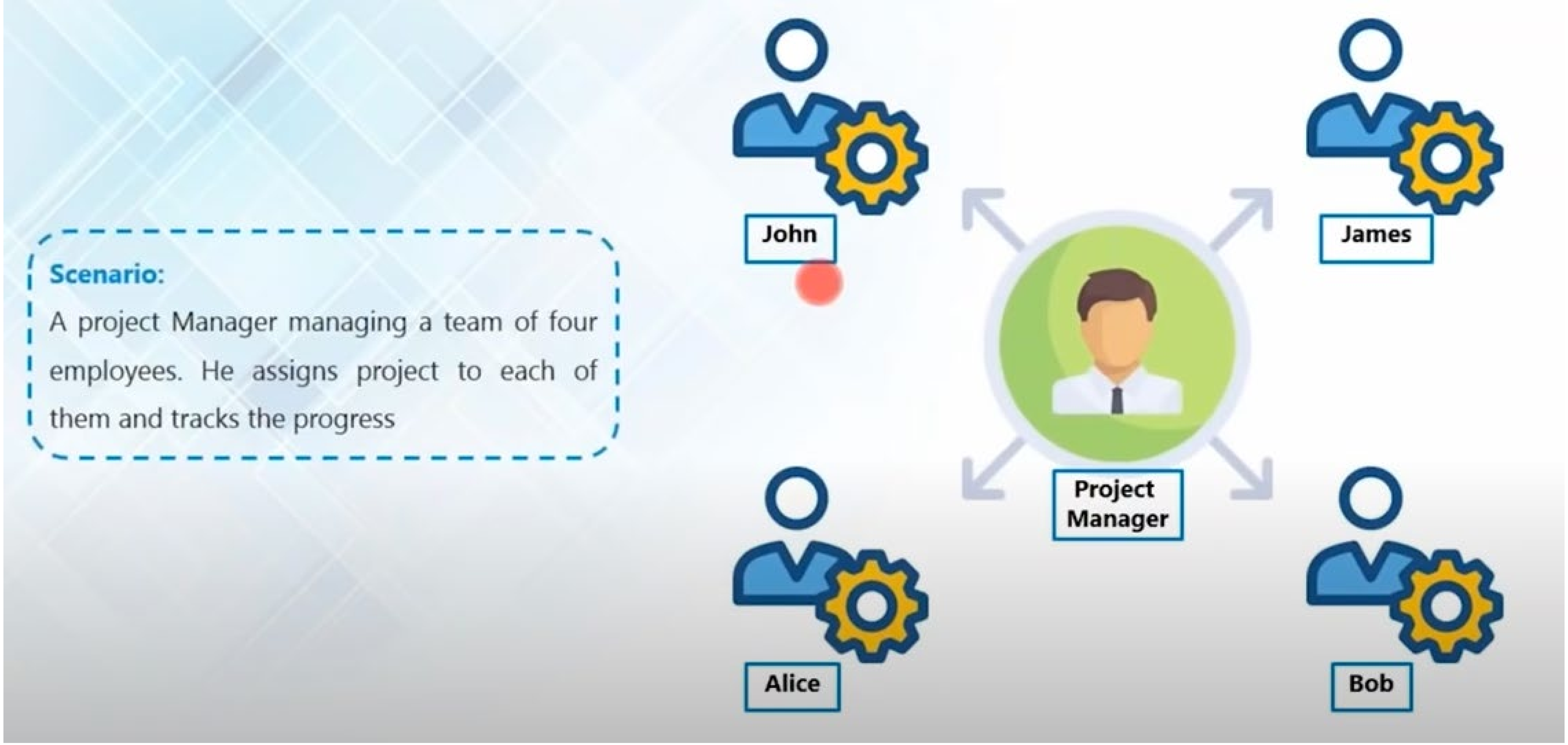
MapReduce

Storage:
Distributed File System

Processing:
Allows parallel & distributed processing

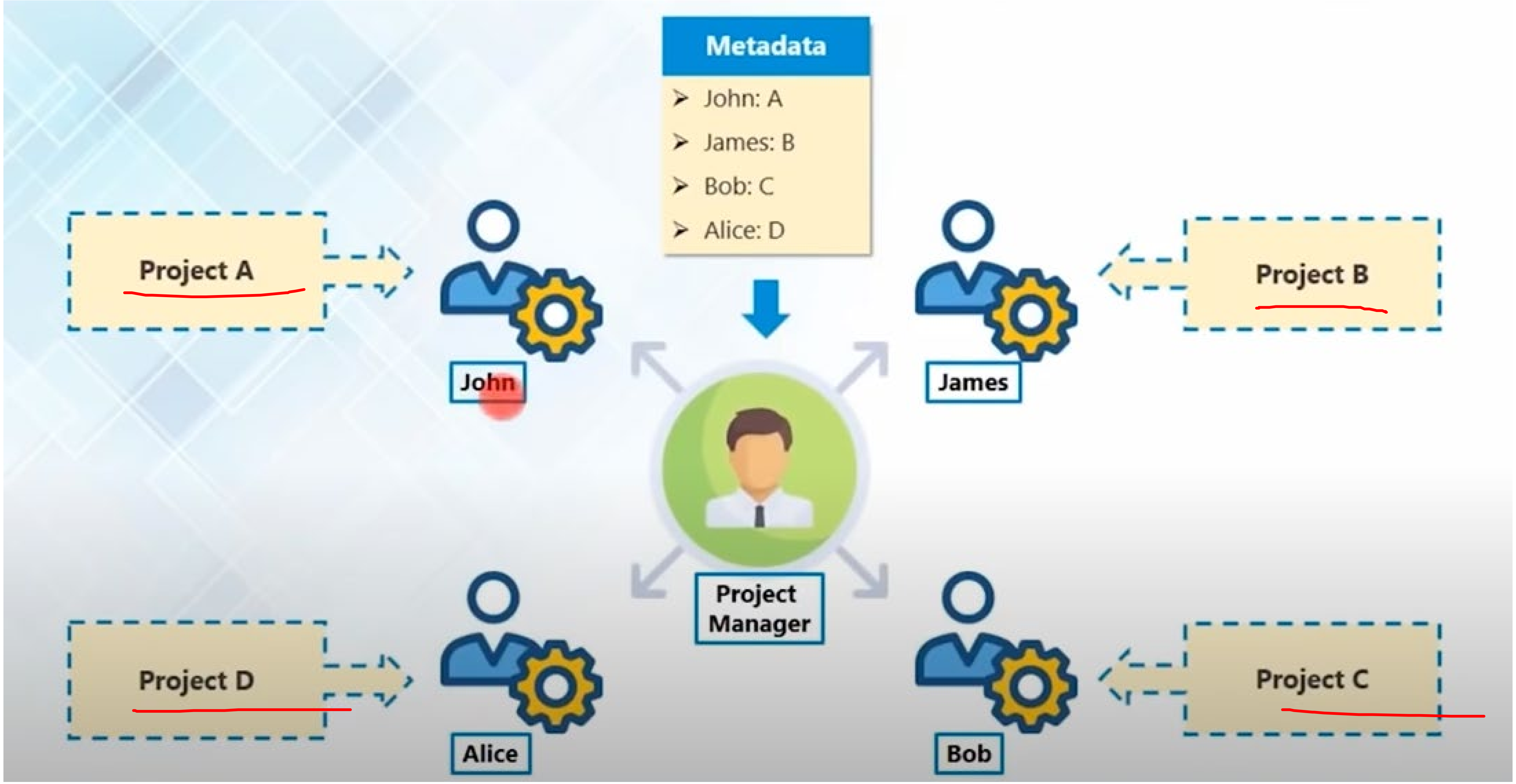
Hadoop Cluster

Hadoop: Master/Slave Architecture



Scenario:

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



Metadata

- John: A
- James: B
- Bob: C
- Alice: D

Project A

John

James

Project B

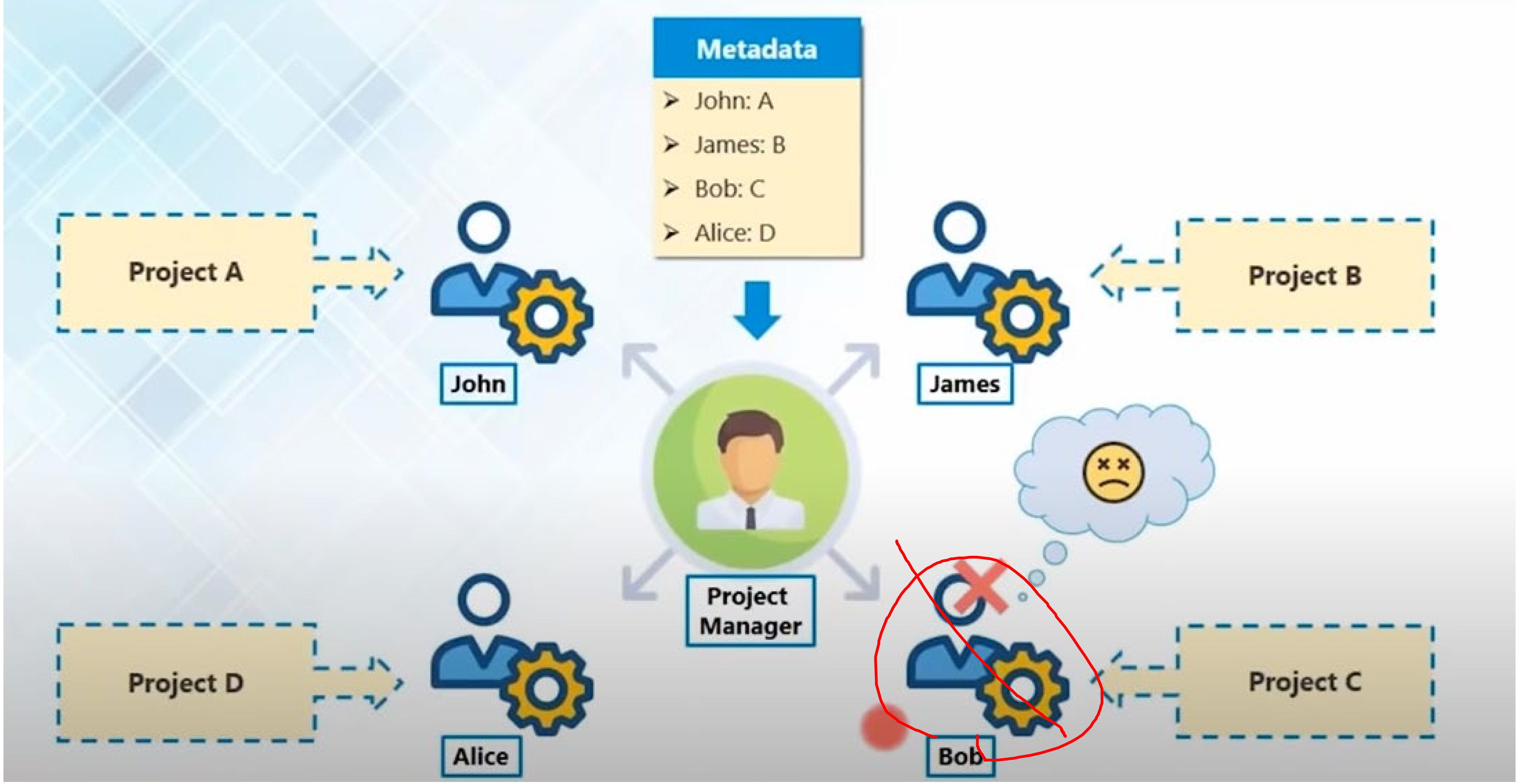
Project Manager

Project D

Alice

Bob

Project C



Metadata

- John: A
- James: B
- Bob: C
- Alice: D

Project A

John

Project B

James

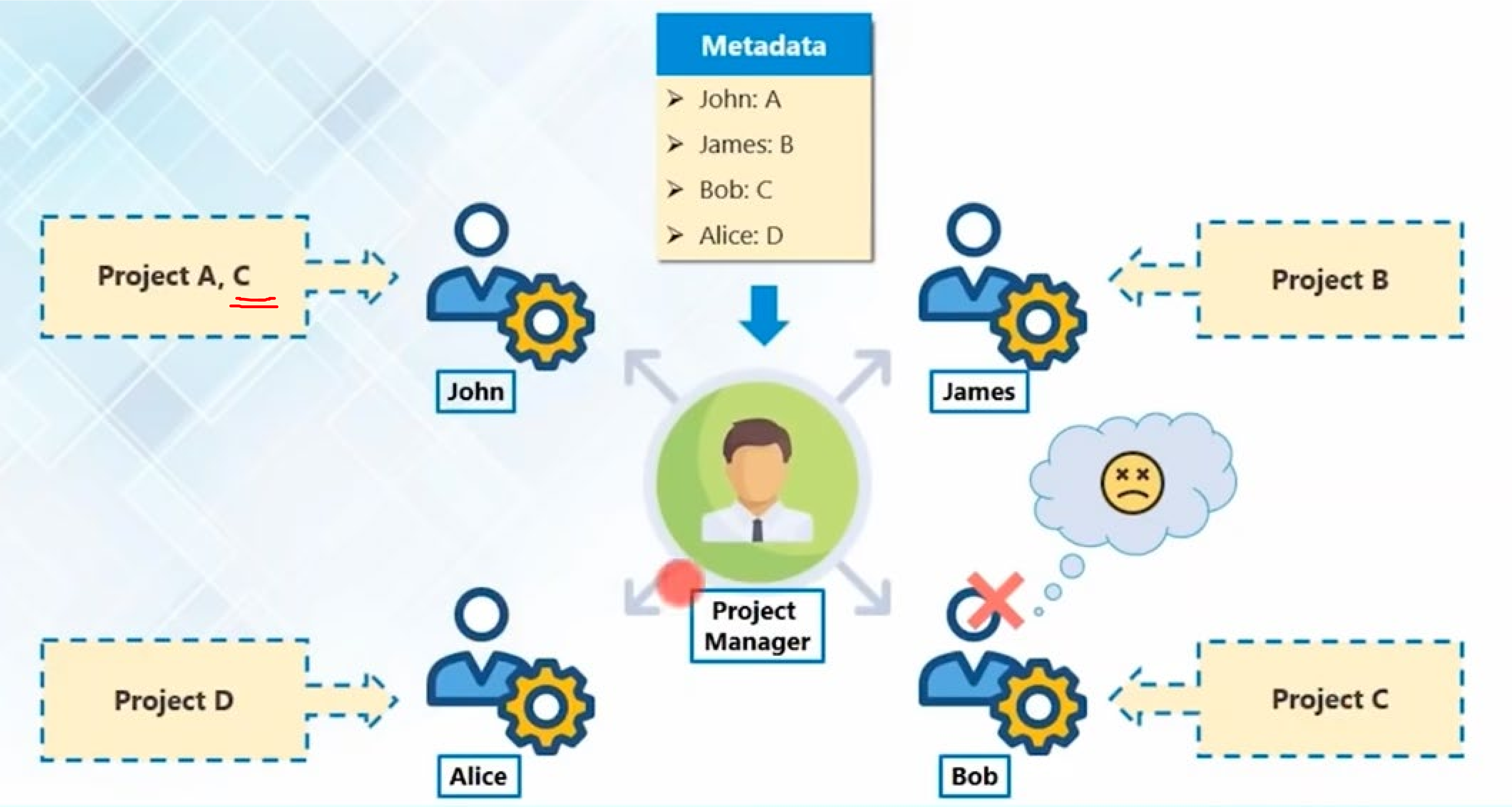
Project Manager

Project D

Alice

Project C

Bob



Metadata

- John: A
- James: B
- Bob: C
- Alice: D

Project A, C

John

Project B

James

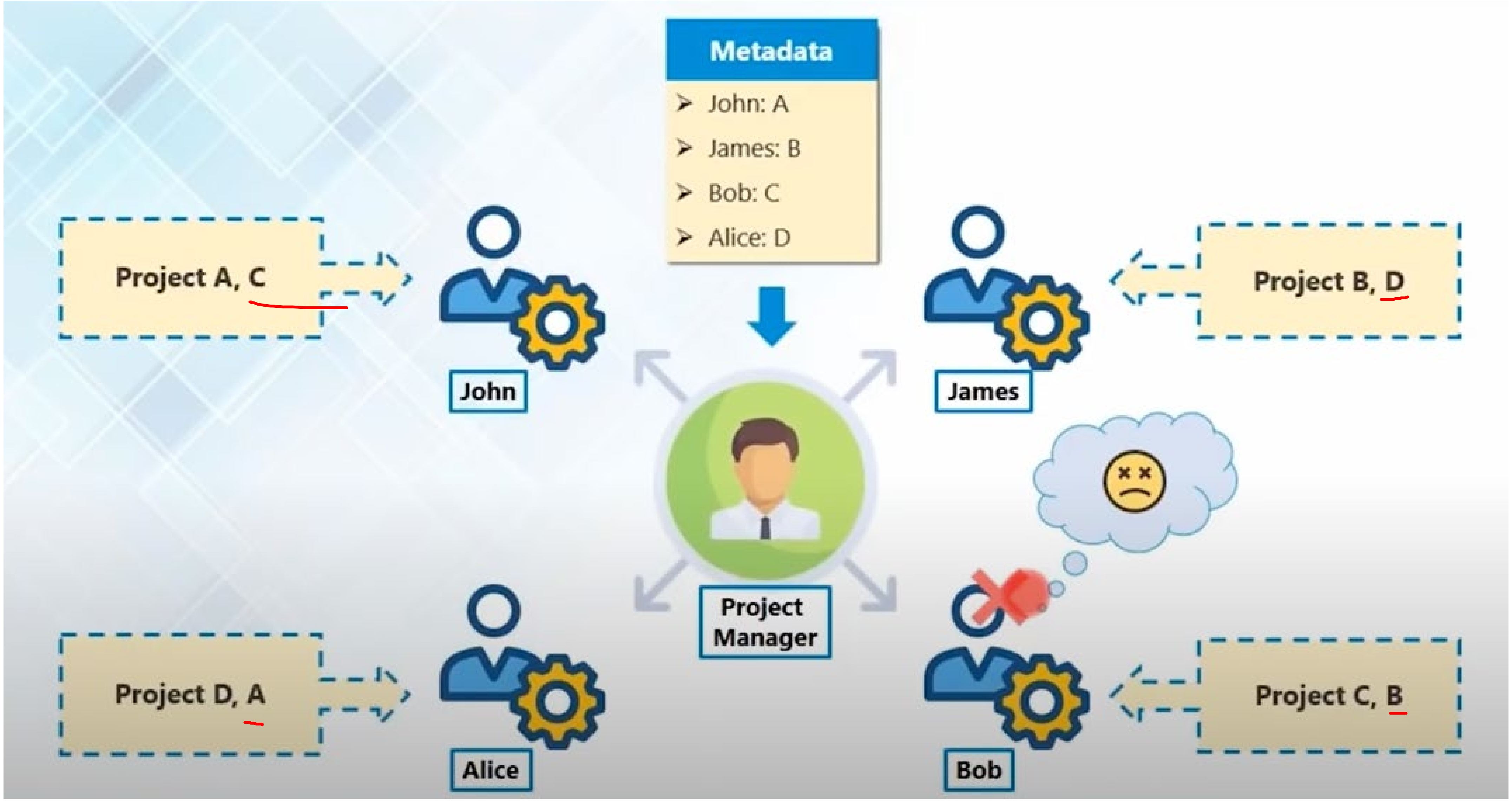
Project Manager

Project D

Alice

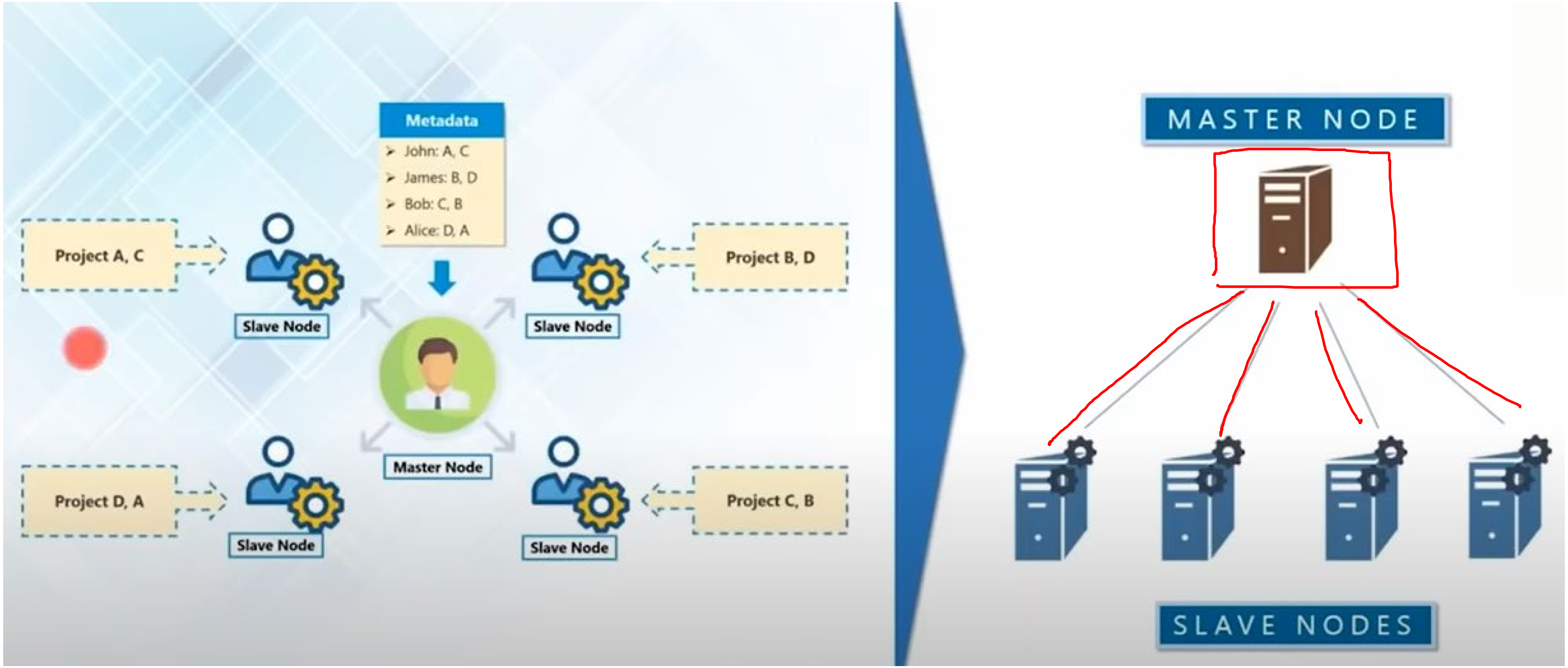
Project C

Bob



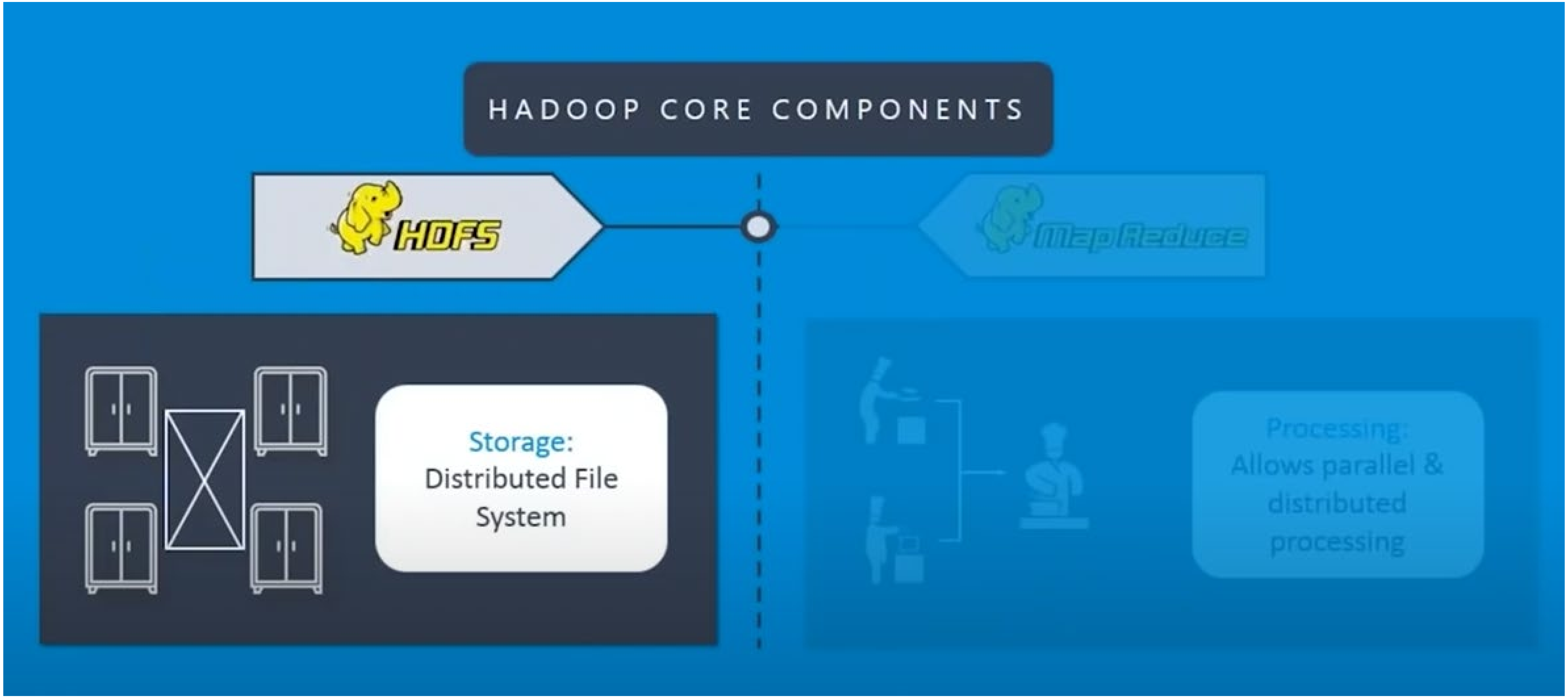
Metadata

- John: A
- James: B
- Bob: C
- Alice: D



Metadata

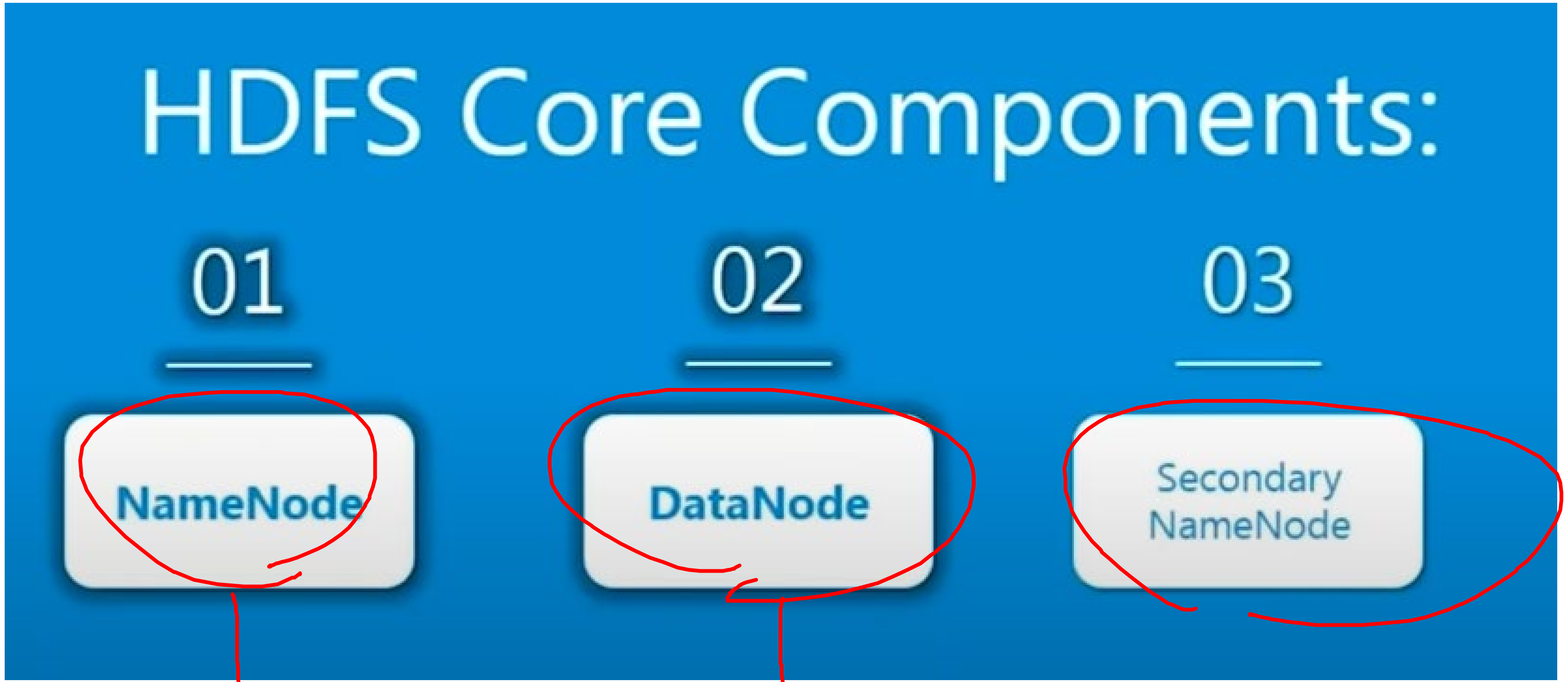
- John: A, C
- James: B, D
- Bob: C, B
- Alice: D, A



HADOOP CORE COMPONENTS

Storage: Distributed File System

Processing: Allows parallel & distributed processing



HDFS Core Components:

01

NameNode

02

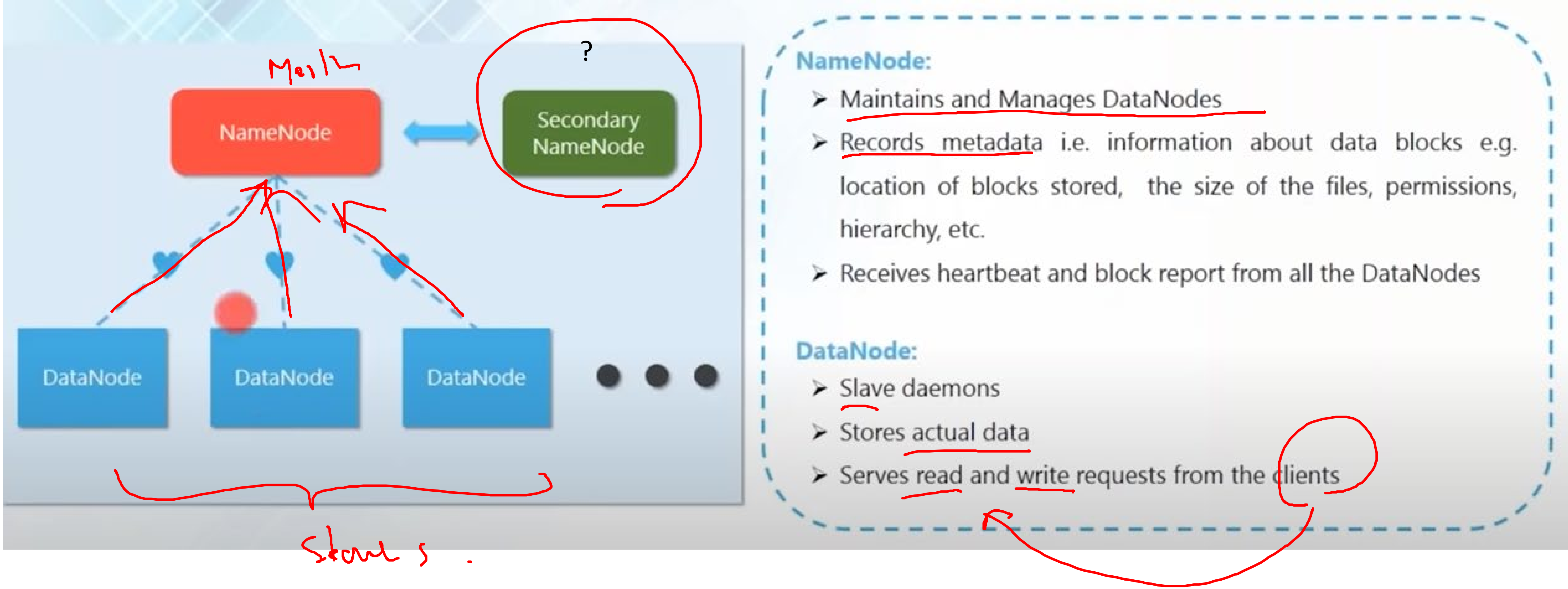
DataNode

03

Secondary NameNode

Master

Slave



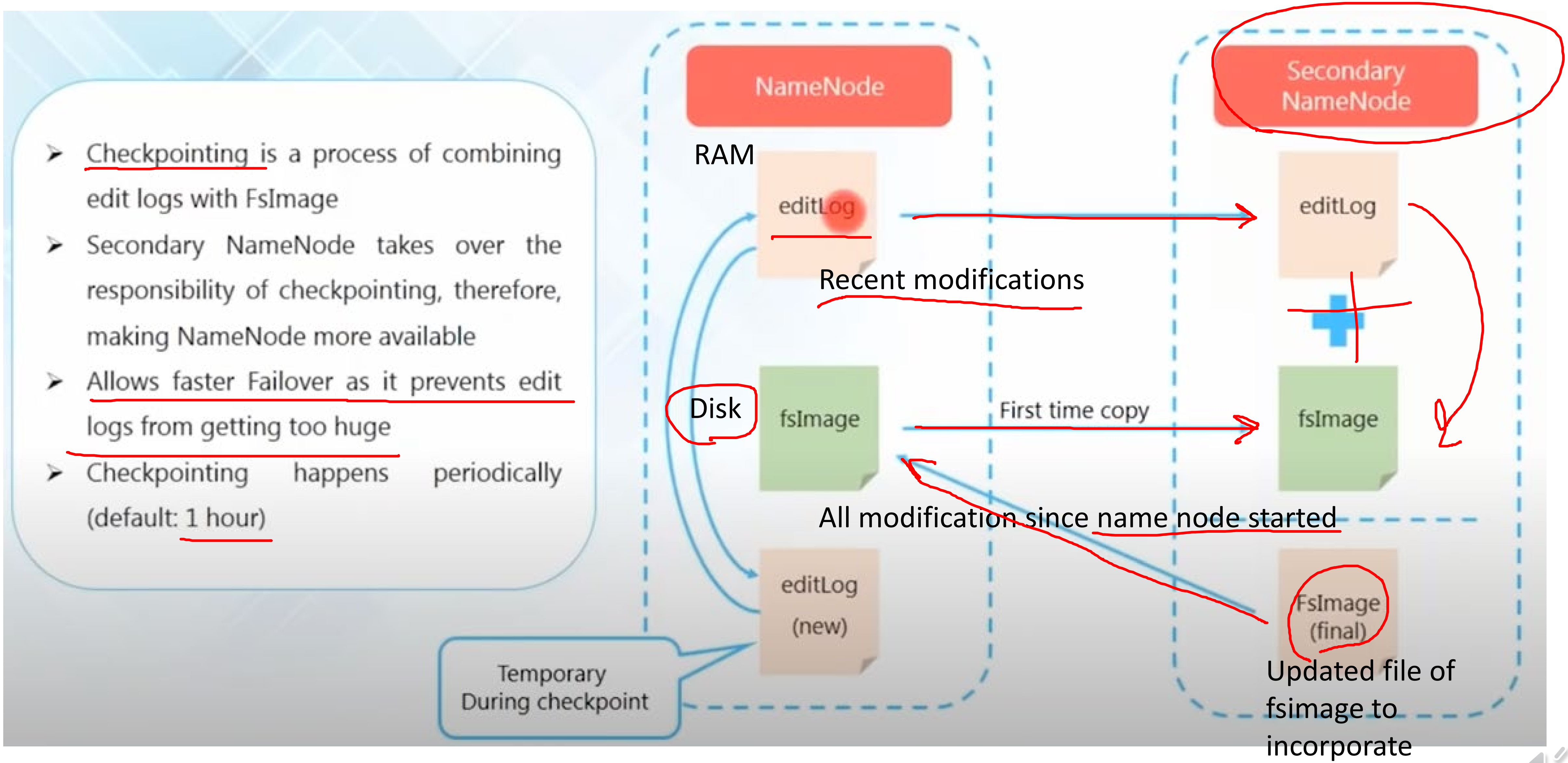
NameNode:

- Maintains and Manages DataNodes
- Records metadata i.e. information about data blocks e.g. location of blocks stored, the size of the files, permissions, hierarchy, etc.
- Receives heartbeat and block report from all the DataNodes

DataNode:

- Slave daemons
- Stores actual data
- Serves read and write requests from the clients

Meta data = changes made to a file



- 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)

RAM

Recent modifications

Disk

All modification since name node started

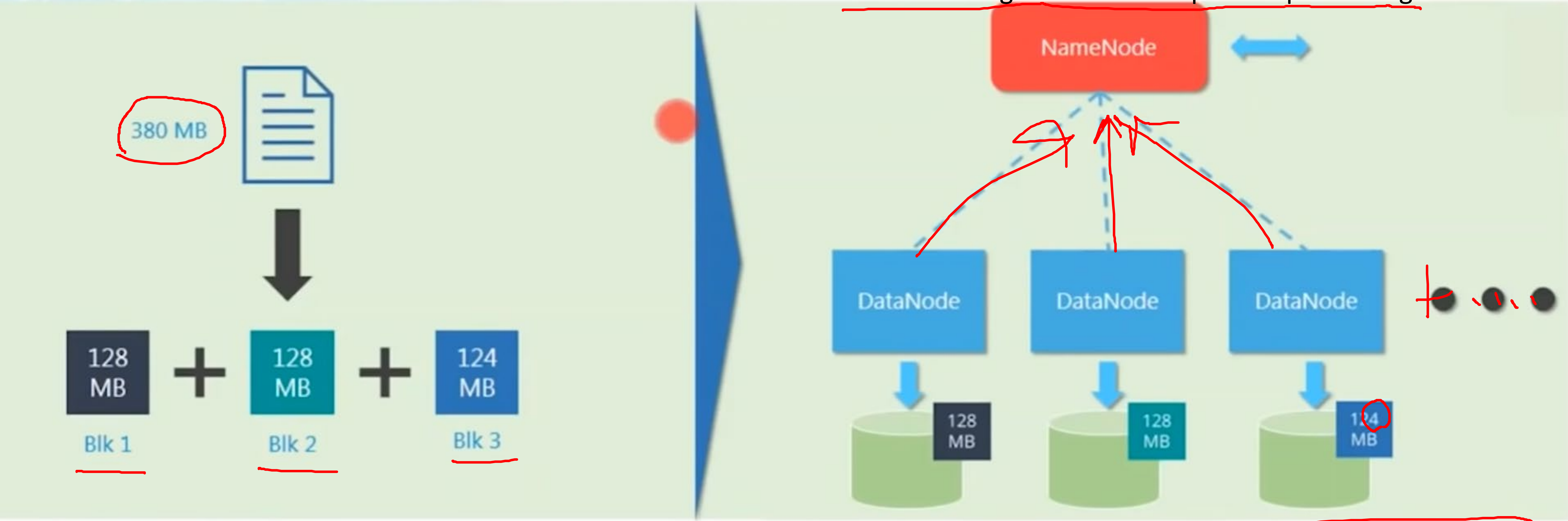
Updated file of fsimage to incorporate recent changes

~~regular~~

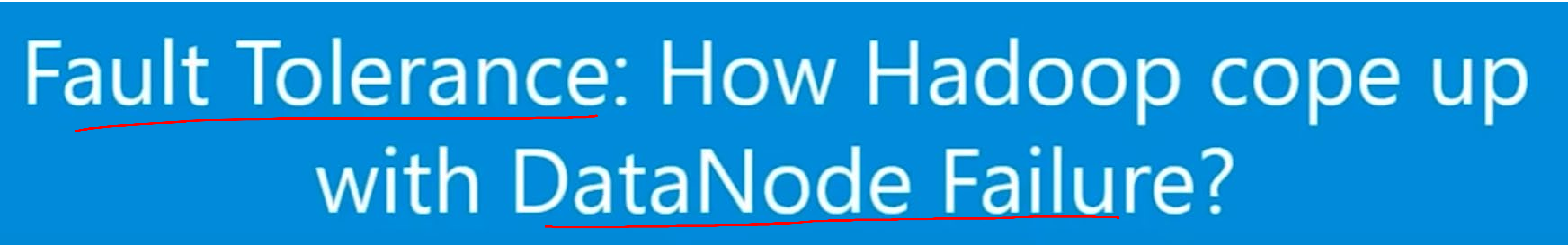
How the data is actually stored in DataNodes?
HDFS Data Blocks

- 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)

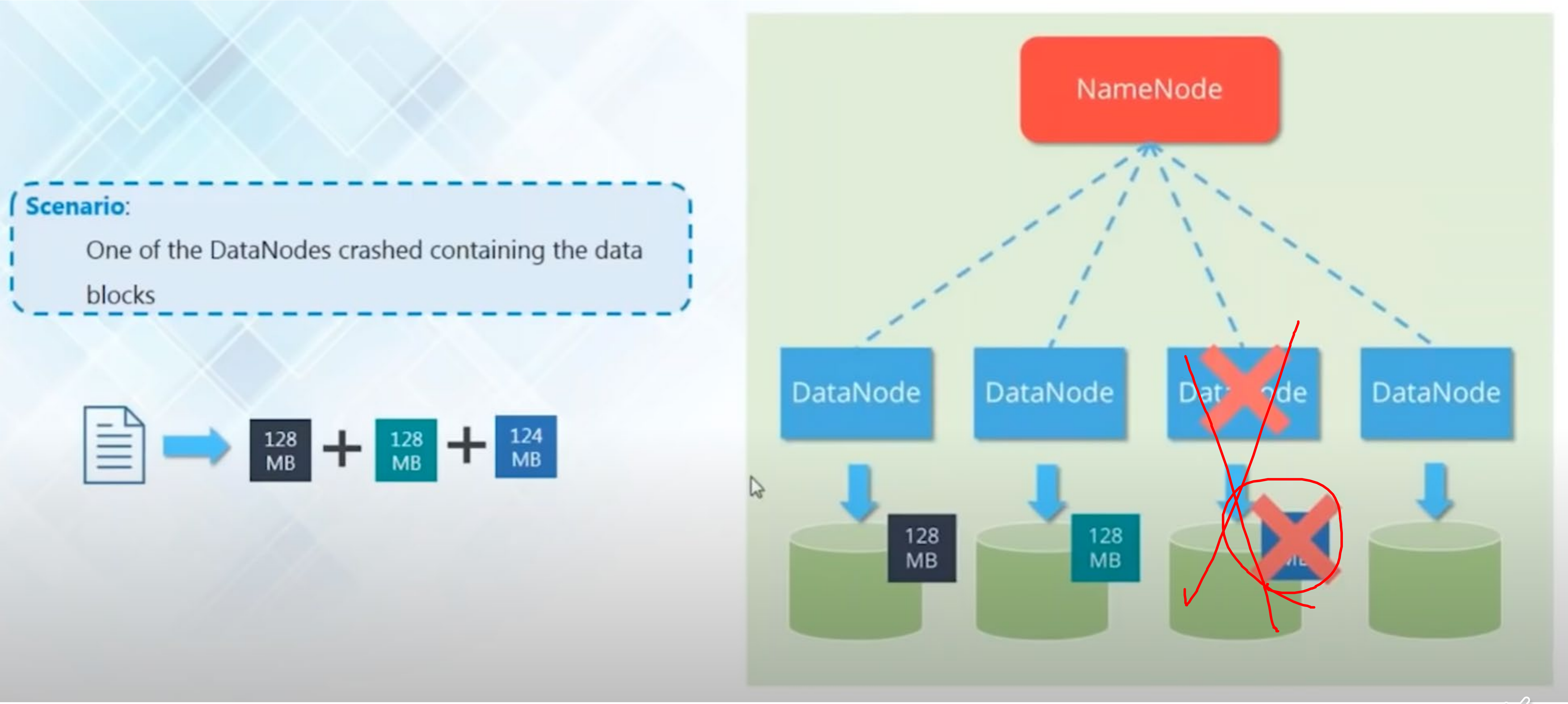
Scalable storage and efficient parallel processing



4 MB saved using 128 mb blocks

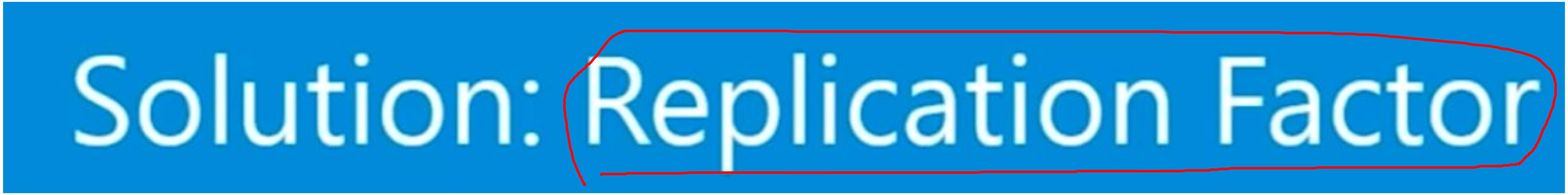


Fault Tolerance: How Hadoop cope up with DataNode Failure?

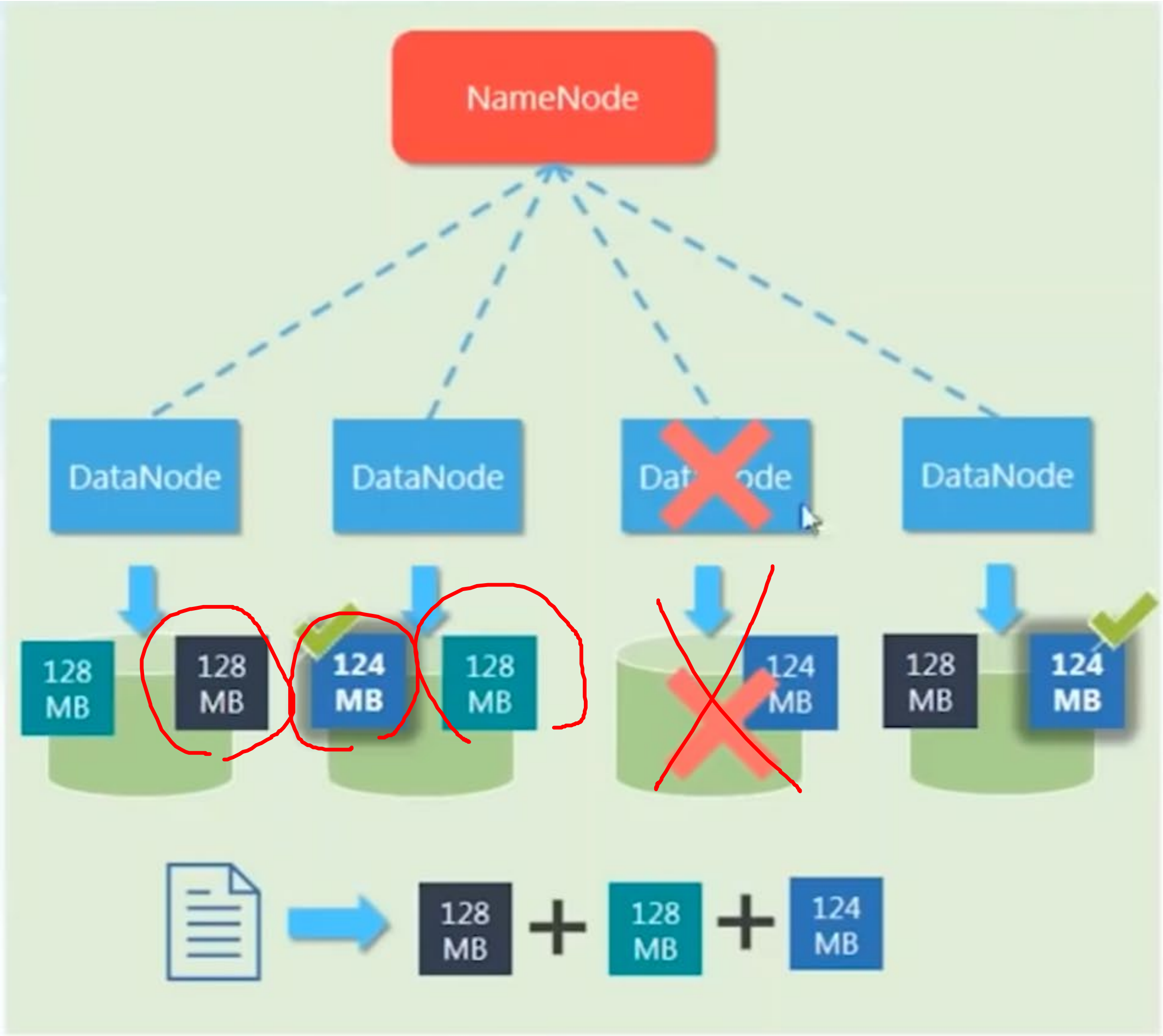


Scenario:

One of the DataNodes crashed containing the data blocks



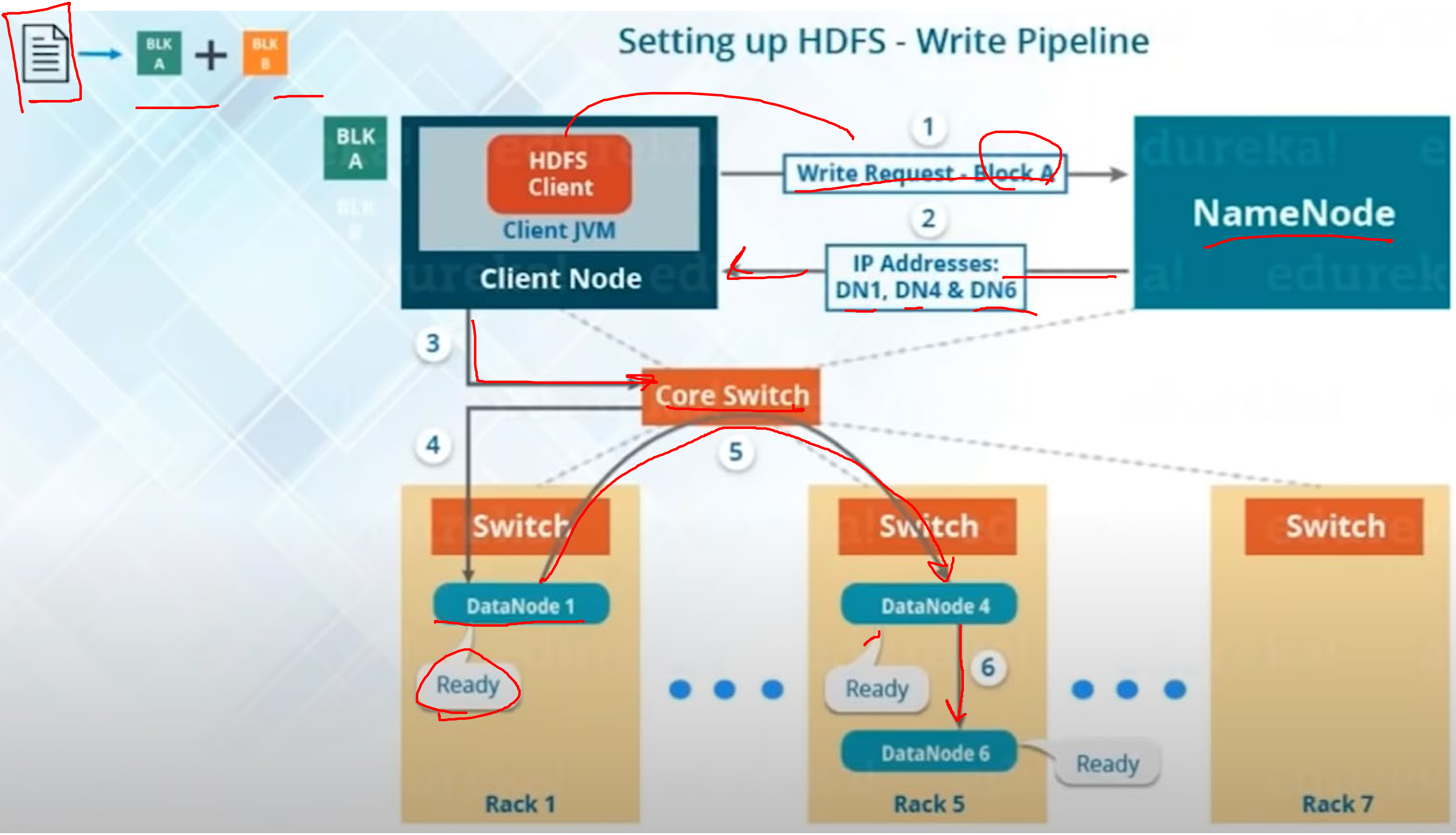
Solution: Replication Factor



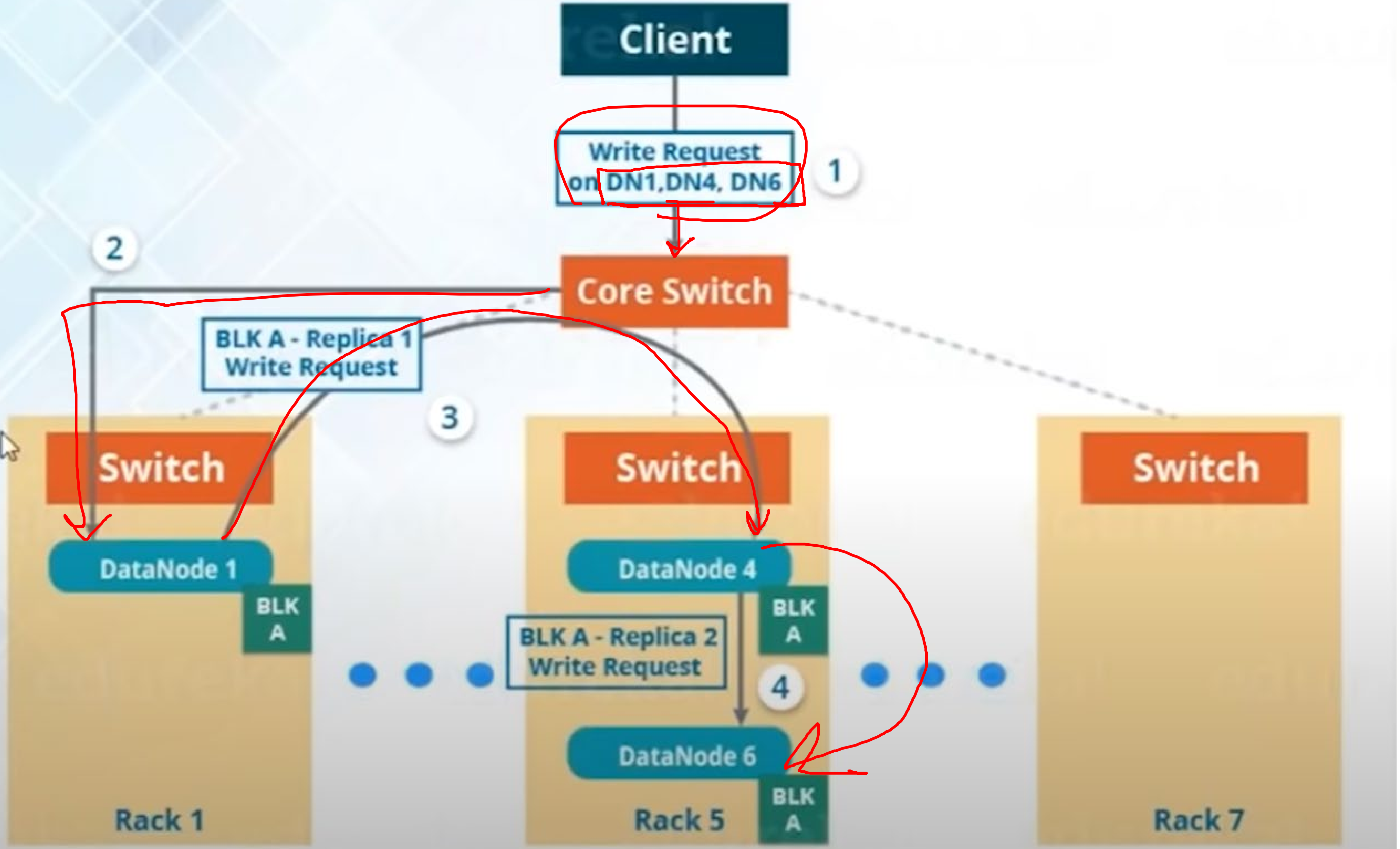
Solution:

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

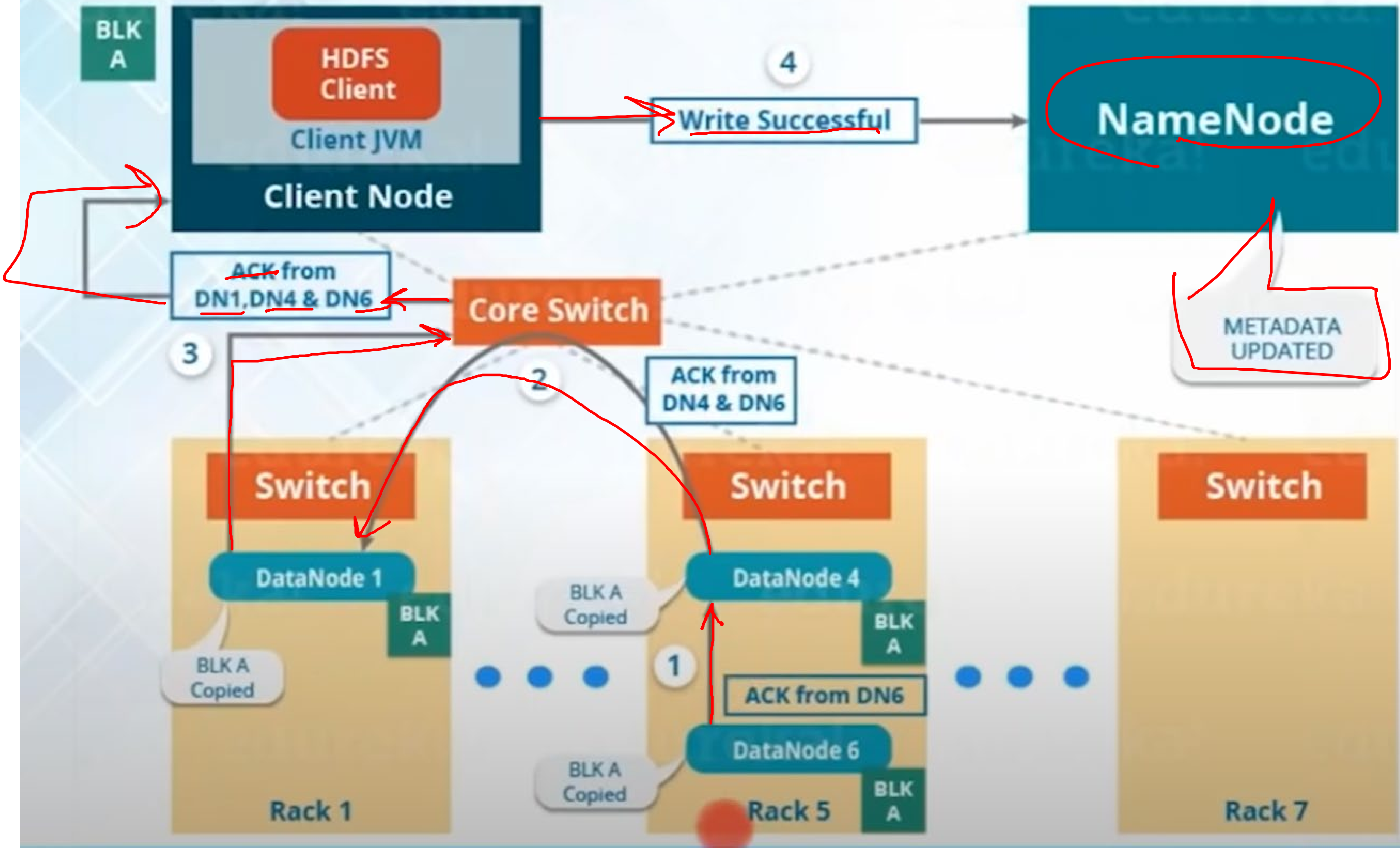
Setting up HDFS - Write Pipeline

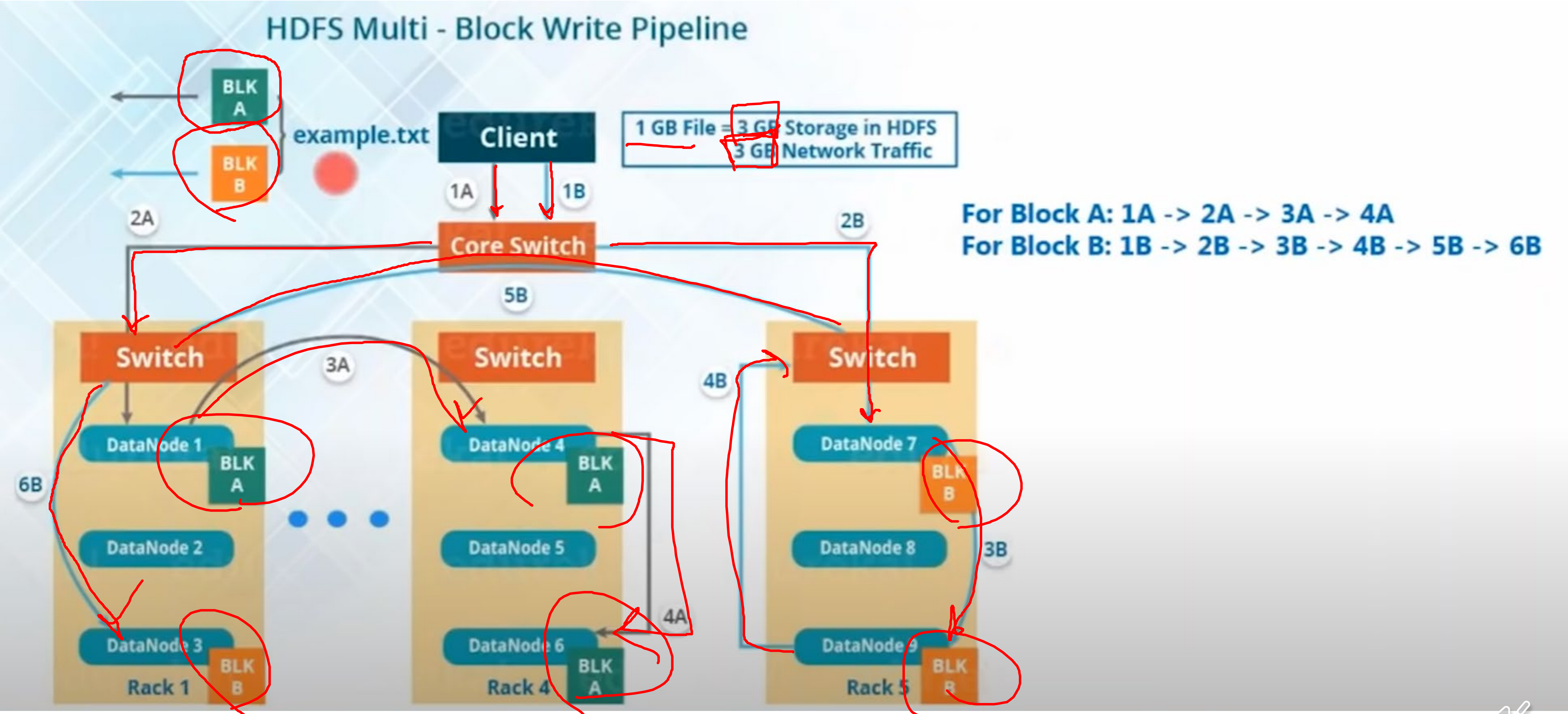


HDFS - Write Pipeline



Acknowledgement in HDFS - Write



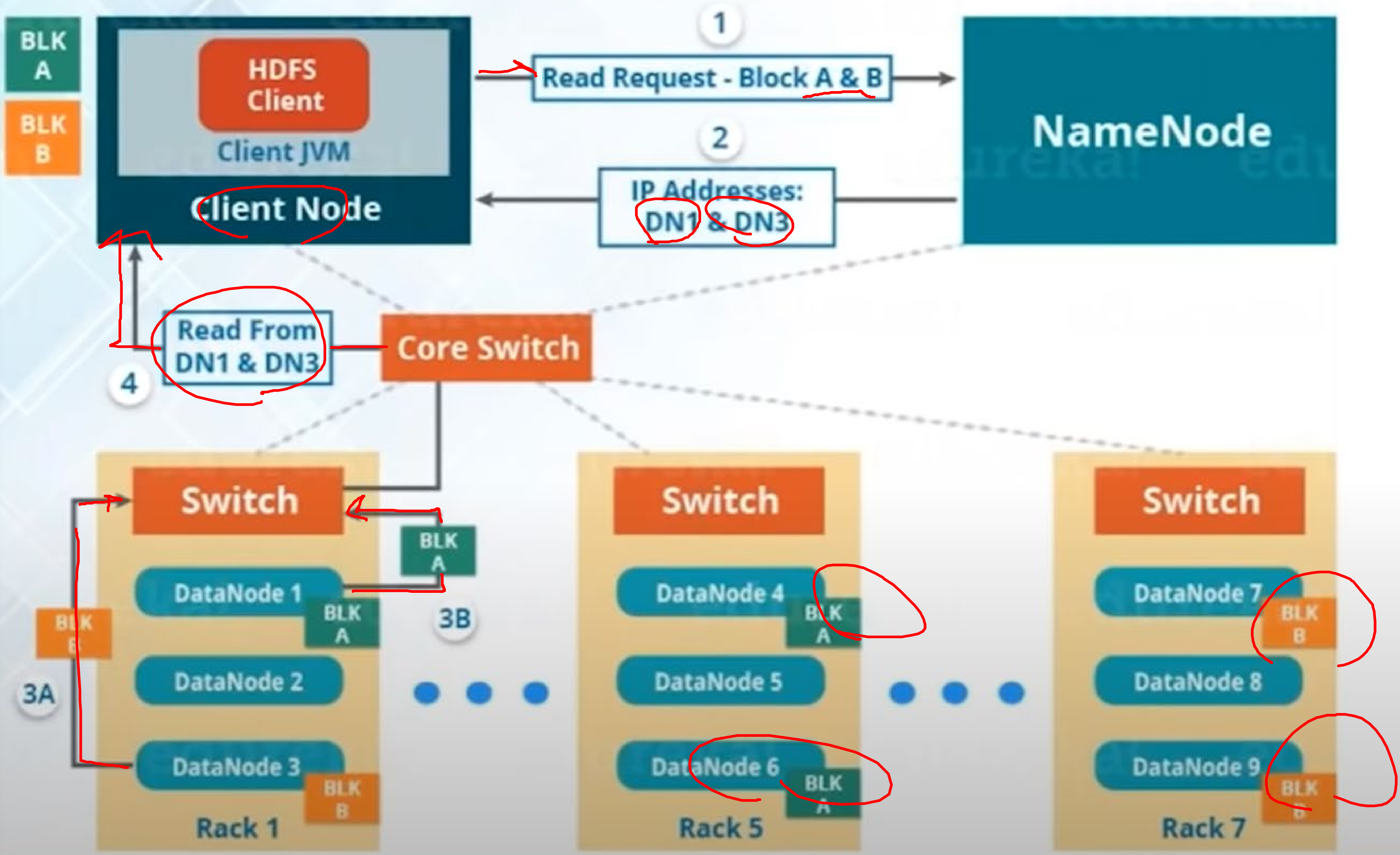


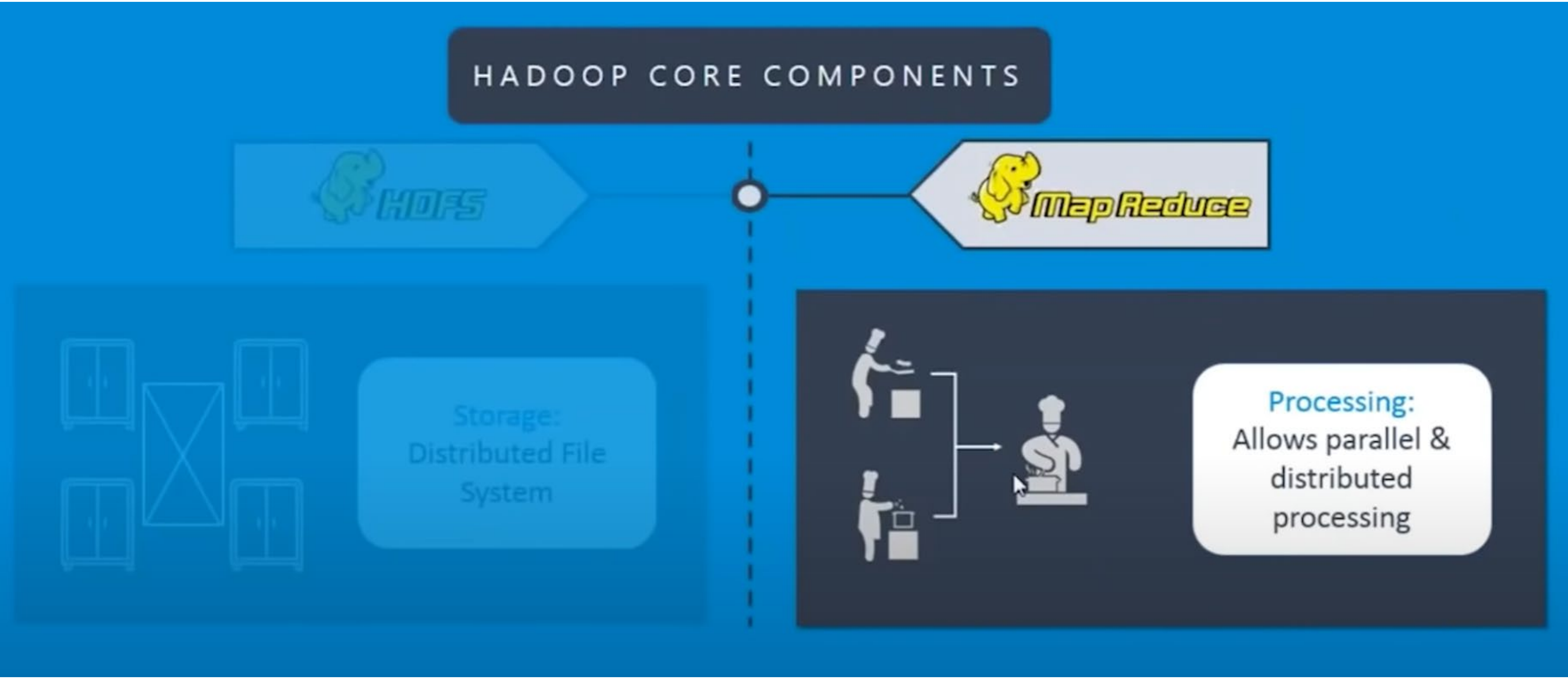
HDFS Multi - Block Write Pipeline

1 GB File = 3 GB Storage in HDFS
3 GB Network Traffic

For Block A: 1A -> 2A -> 3A -> 4A
For Block B: 1B -> 2B -> 3B -> 4B -> 5B -> 6B

HDFS - Read Architecture





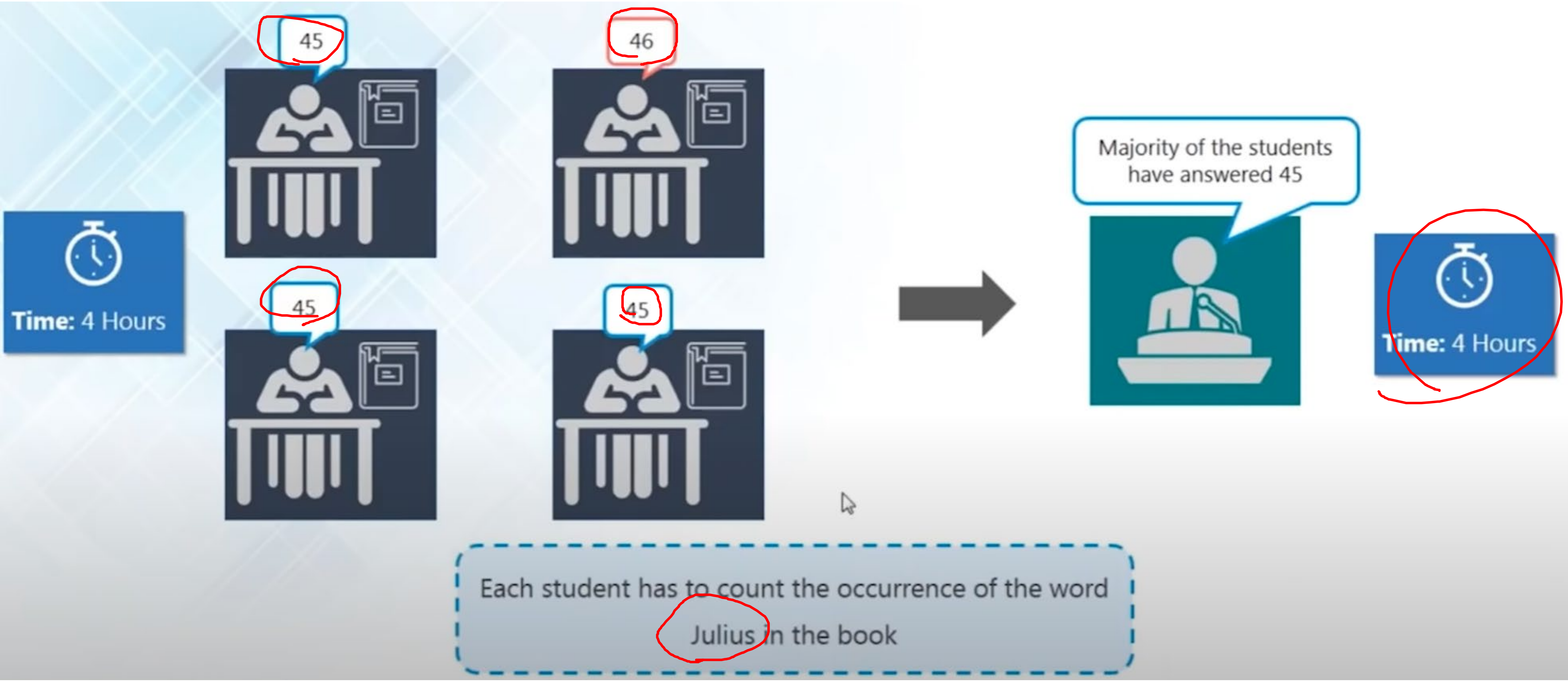
HADOOP CORE COMPONENTS

HDFS

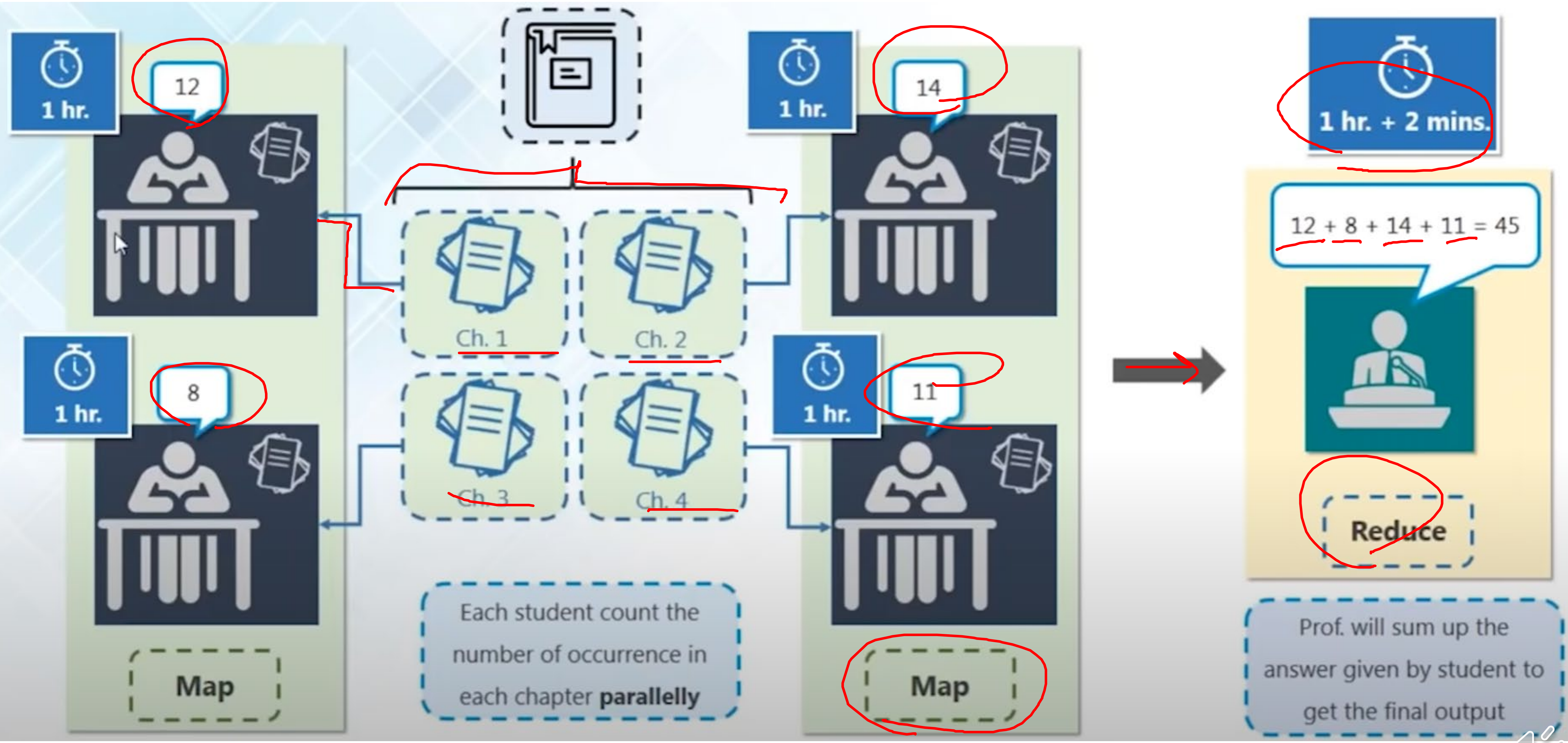
Storage: Distributed File System

MapReduce

Processing: Allows parallel & distributed processing

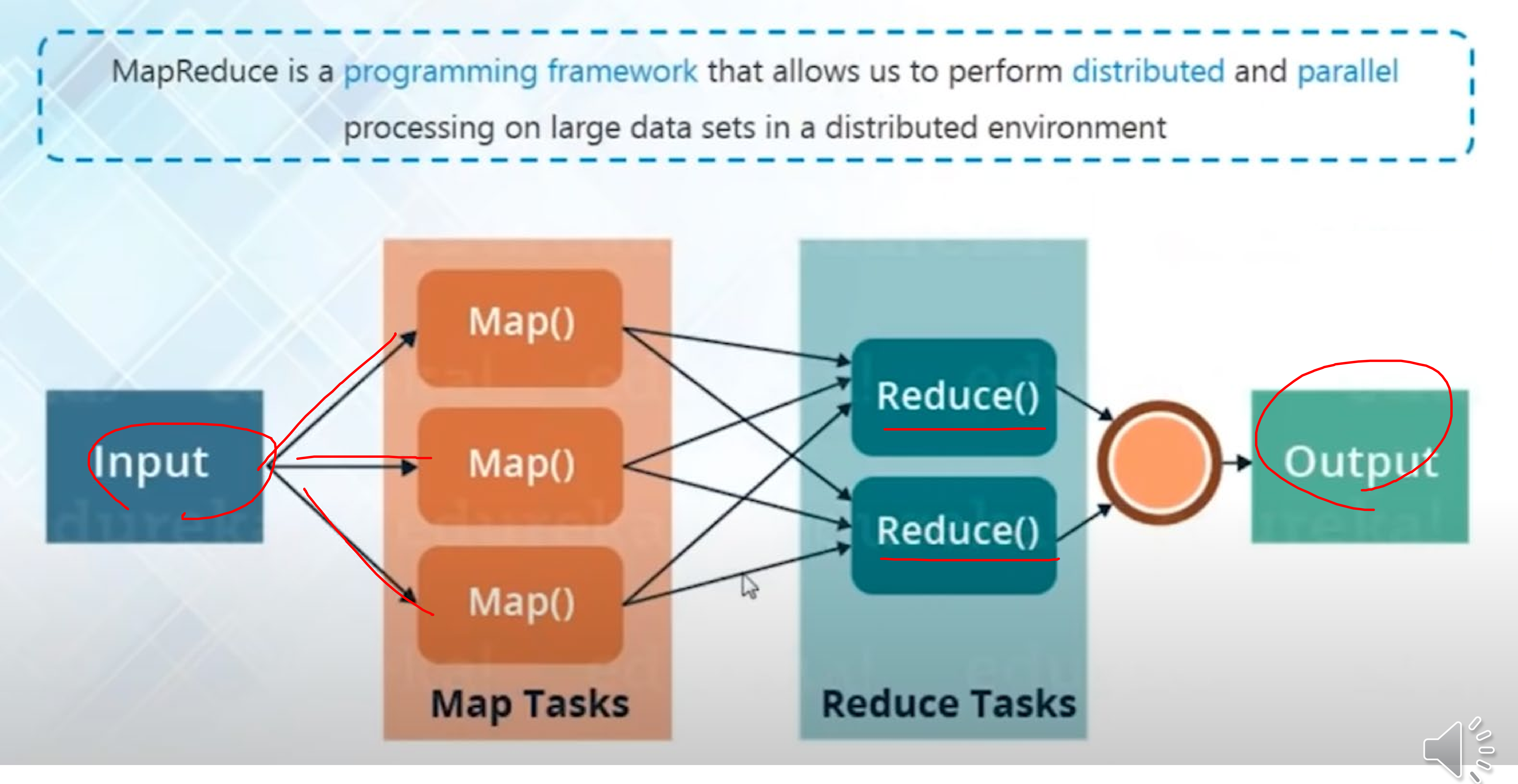


Each student has to count the occurrence of the word Julius in the book

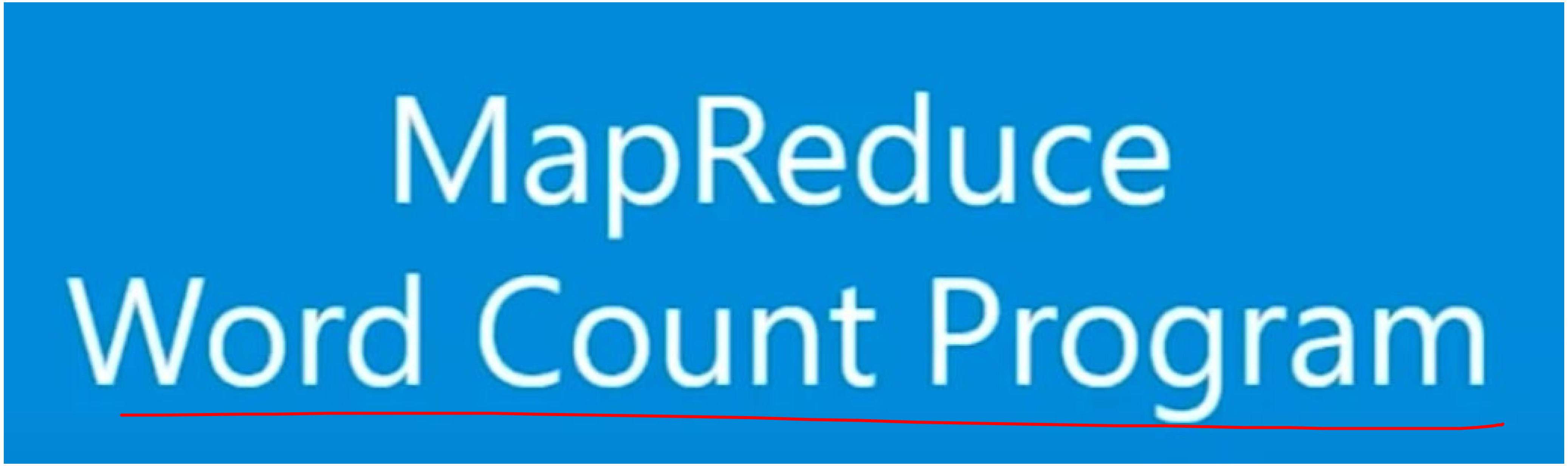


Each student count the number of occurrence in each chapter **parallelly**

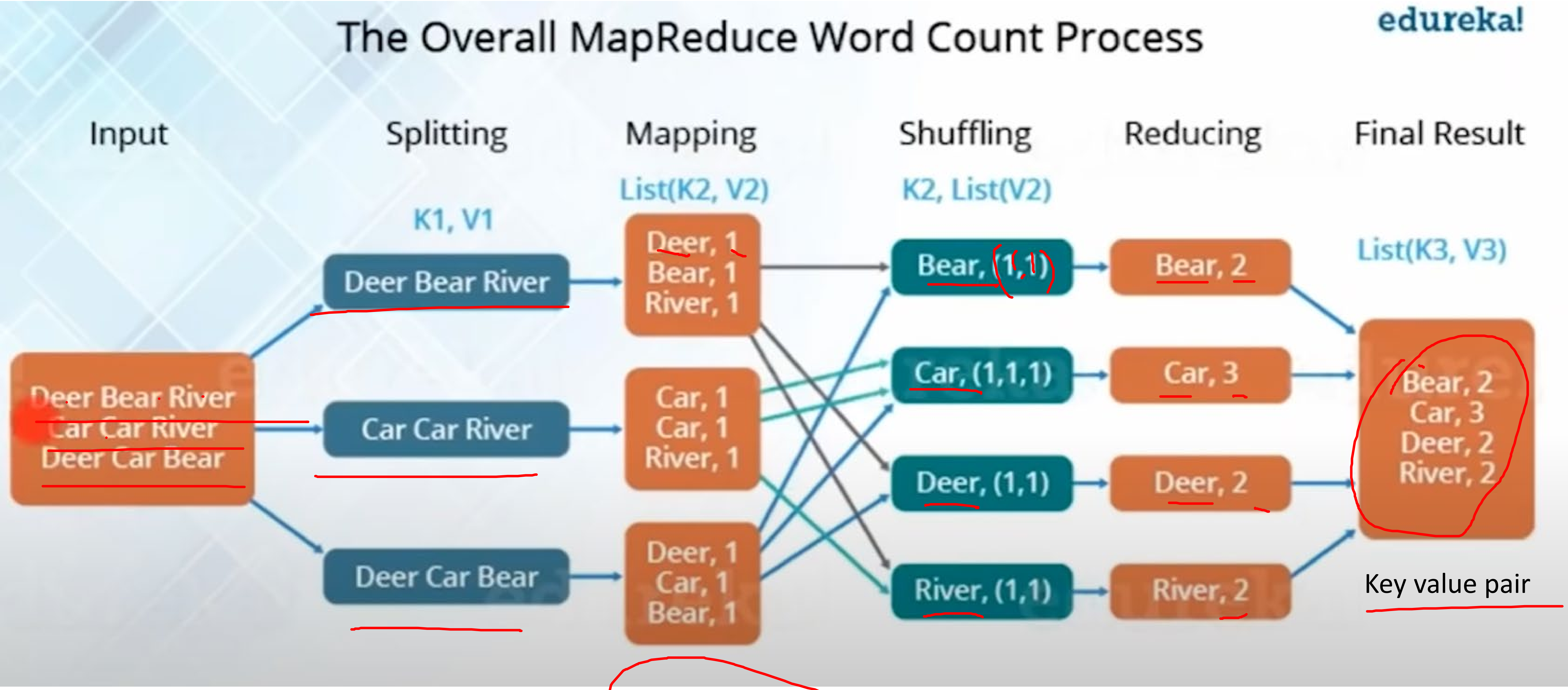
Prof. will sum up the answer given by student to get the final output



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



MapReduce
Word Count Program

# The Overall MapReduce Word Count Process

edureka!

Input

Deer Bear River
Car Car River
Deer Car Bear

Splitting

K1, V1

Deer Bear River

Car Car River

Deer Car Bear

Mapping

List(K2, V2)

Deer, 1
Bear, 1
River, 1

Car, 1
Car, 1
River, 1

Deer, 1
Car, 1
Bear, 1

Shuffling

K2, List(V2)

Bear, (1,1)

Car, (1,1,1)

Deer, (1,1)

River, (1,1)

Reducing

Bear, 2

Car, 3

Deer, 2

River, 2

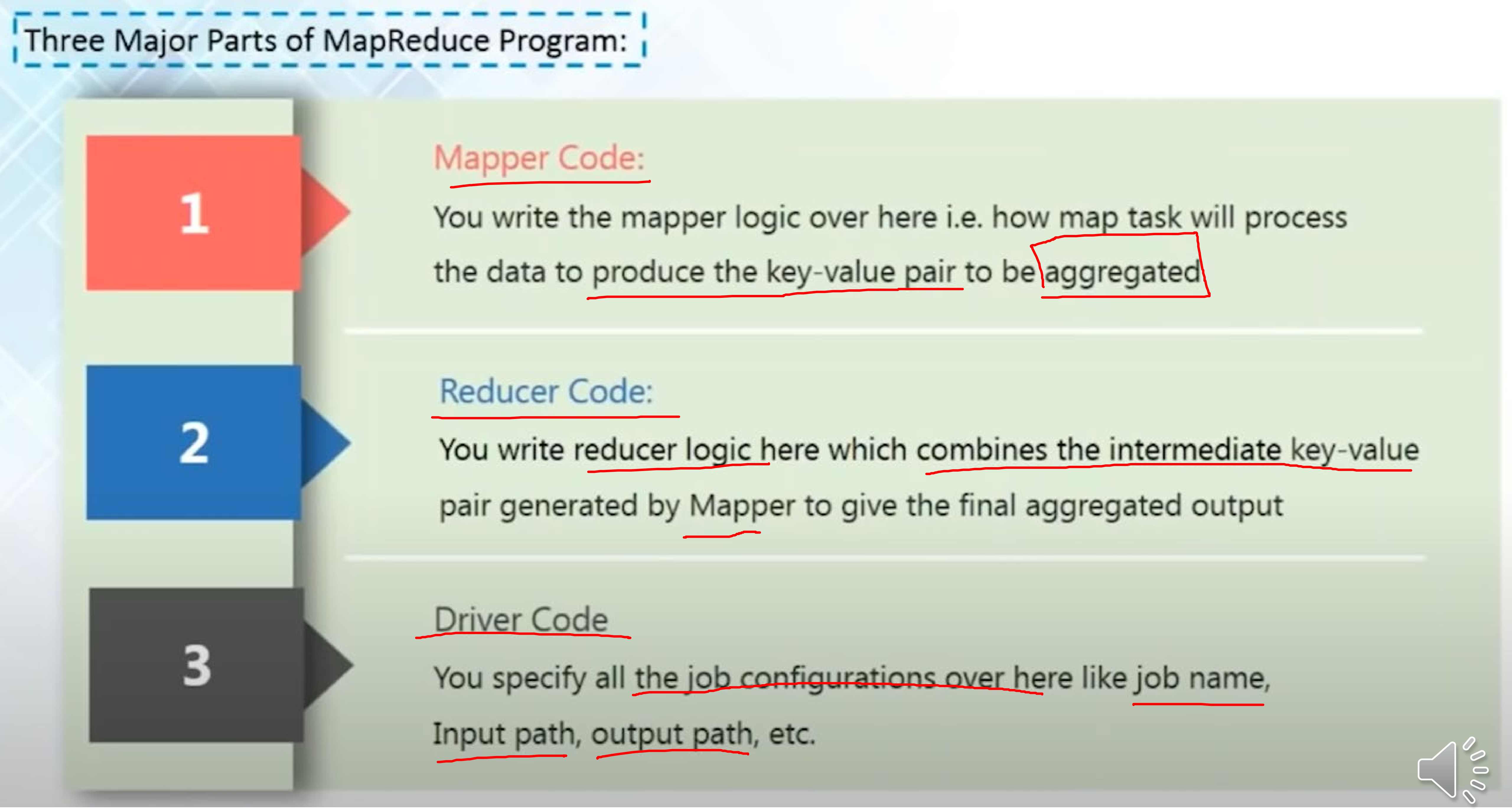
Final Result

List(K3, V3)

Bear, 2
Car, 3
Deer, 2
River, 2

Key value pair

Key value pair



Three Major Parts of MapReduce Program:

1

Mapper Code:

You write the mapper logic over here i.e. how map task will process the data to produce the key-value pair to be aggregated

2

Reducer Code:

You write reducer logic here which combines the intermediate key-value pair generated by Mapper to give the final aggregated output

3

Driver Code

You specify all the job configurations over here like job name, Input path, output path, etc.