BITS Pilani presentation

BITS Pilani
Pilani Campus

Dr. Vivek V. Jog
Dept. Of Computer Engineering

innovate achieve lead

BITS Pilani
Pilani Campus

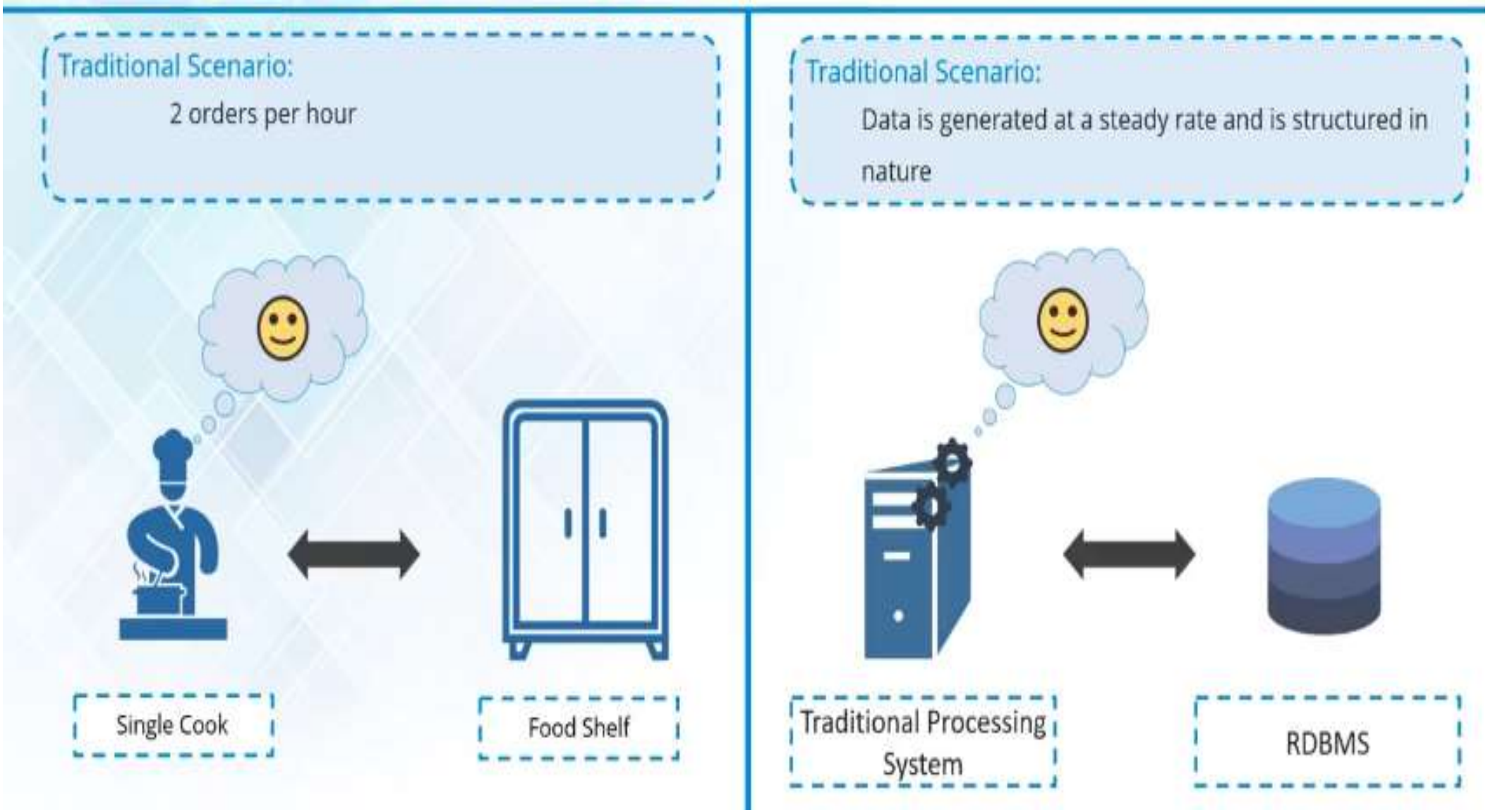
Big Data Systems (S1-24_CCZG522)
Lecture No.3

INDEX

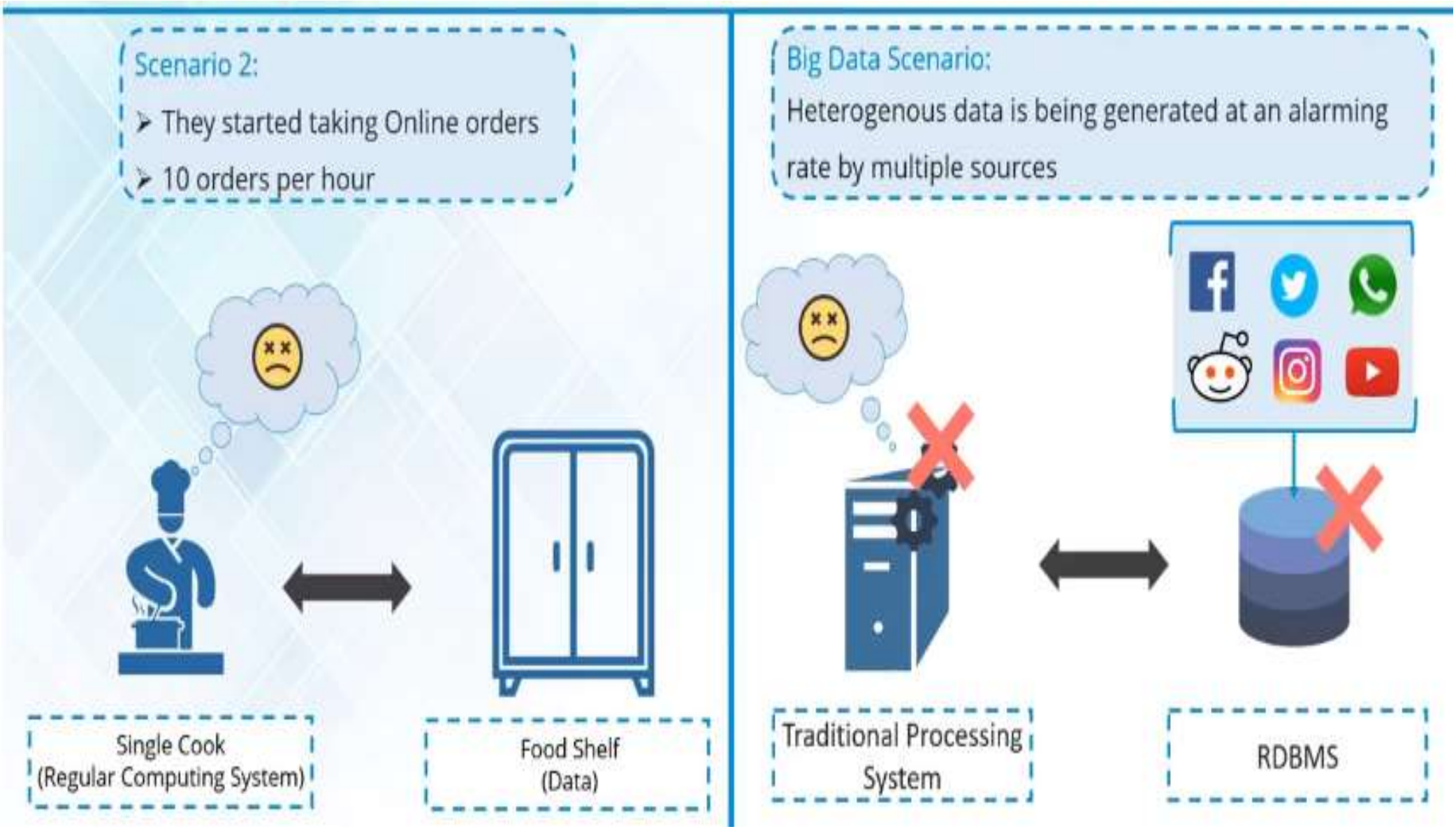
Five V's

"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"

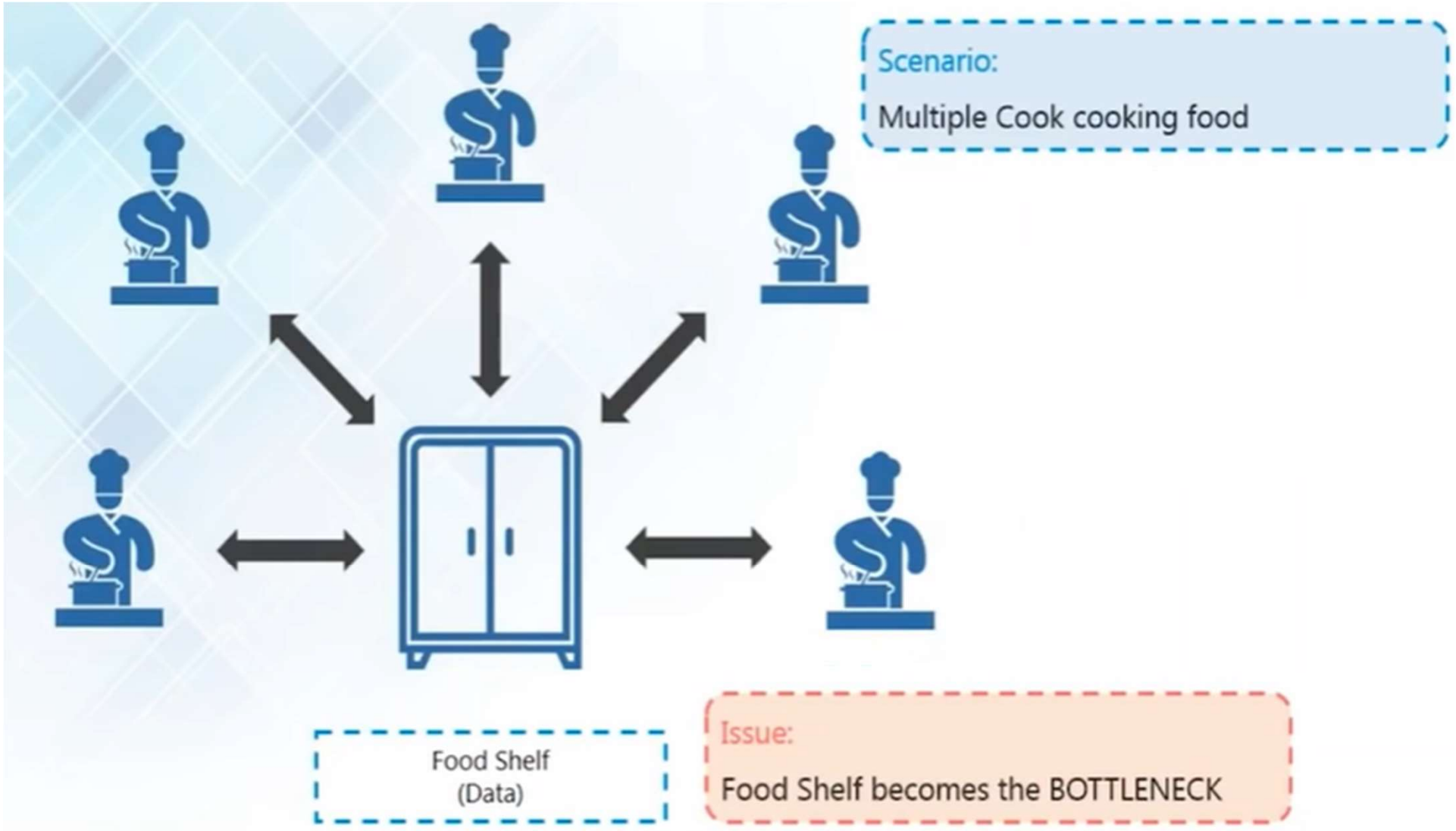
Traditional System V/s Big Data



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


Scenario:
Multiple Cook cooking food
Food Shelf
(Data)
Issue:
Food Shelf becomes the 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

Scenario:

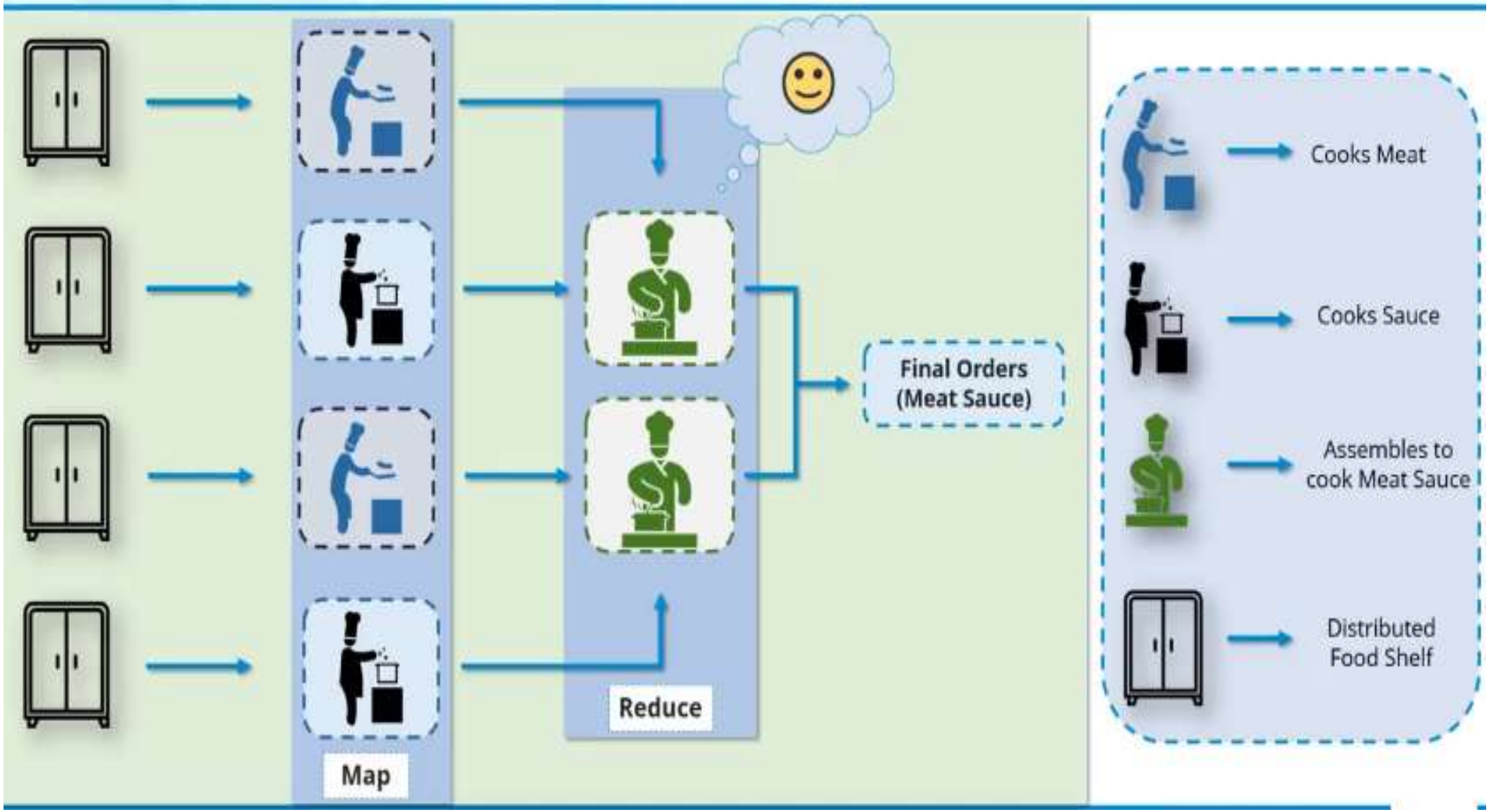
Multiple Processing Unit for data processing

Issue:

Bringing data to processing generated lots of Network overhead

Check for the Bottleneck

Effective Solution

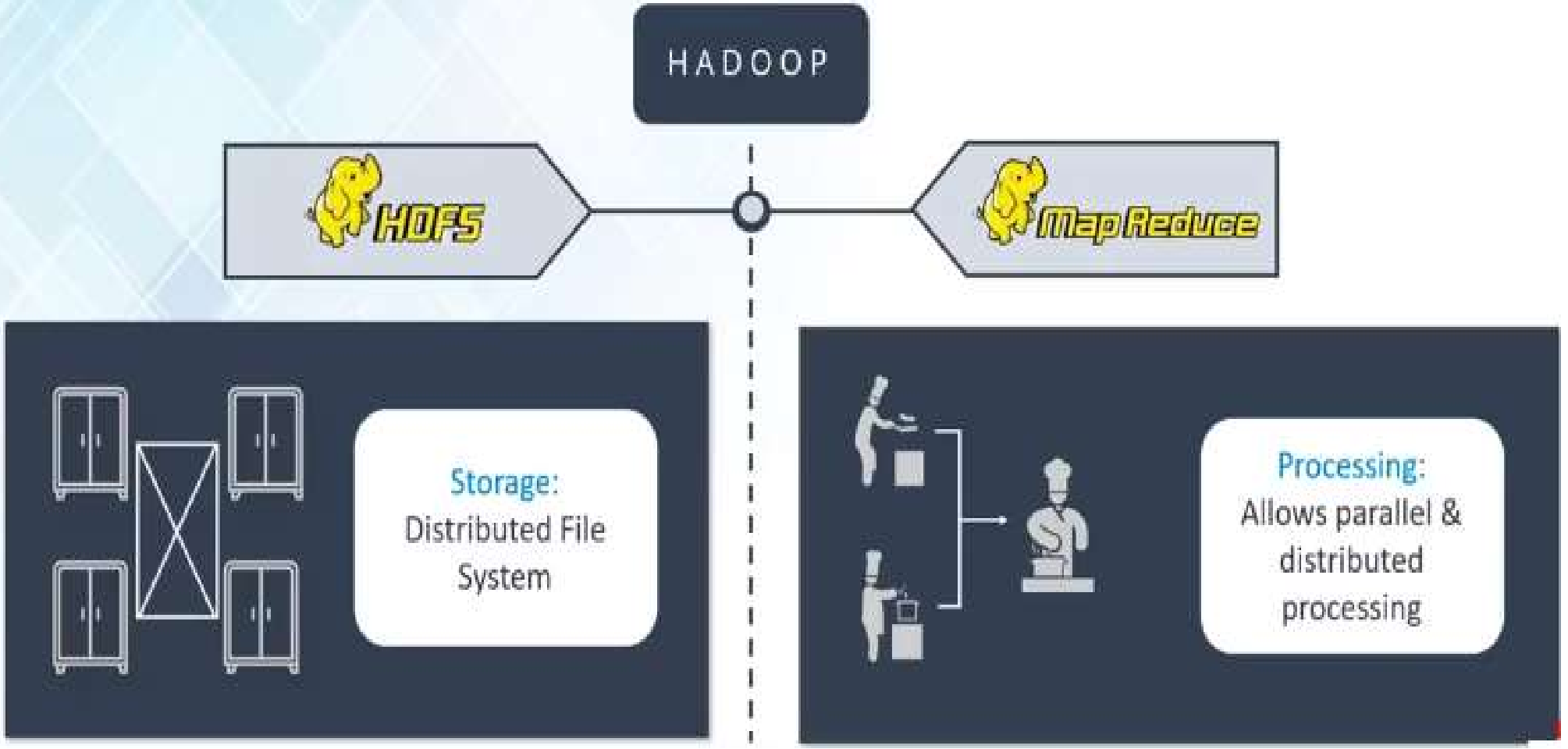


Solution Framework

WE NEED SOME FRAMEWORK THAT CAN PROVIDES SOLUTION

Apache Hadoop

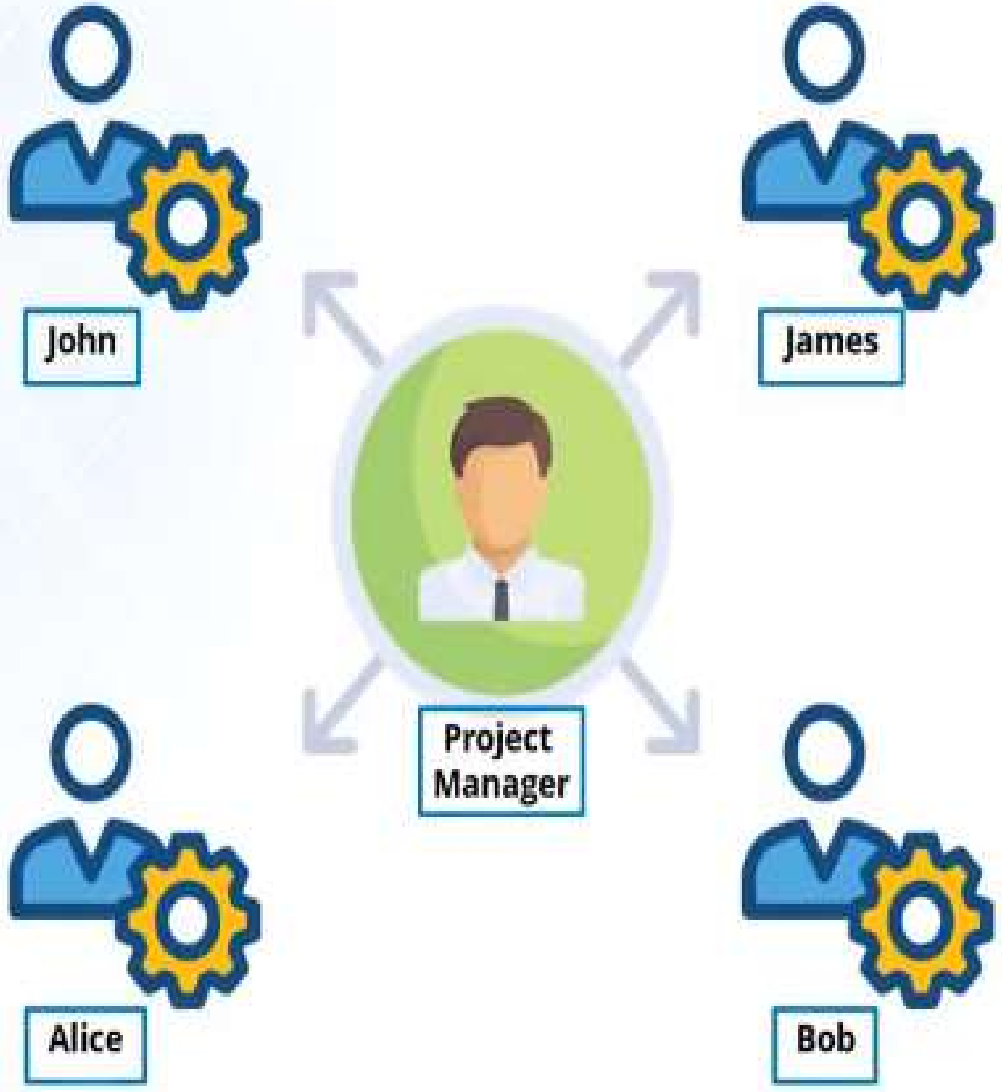
Hadoop is a framework that allows us to store and process large data sets in parallel and distributed fashion



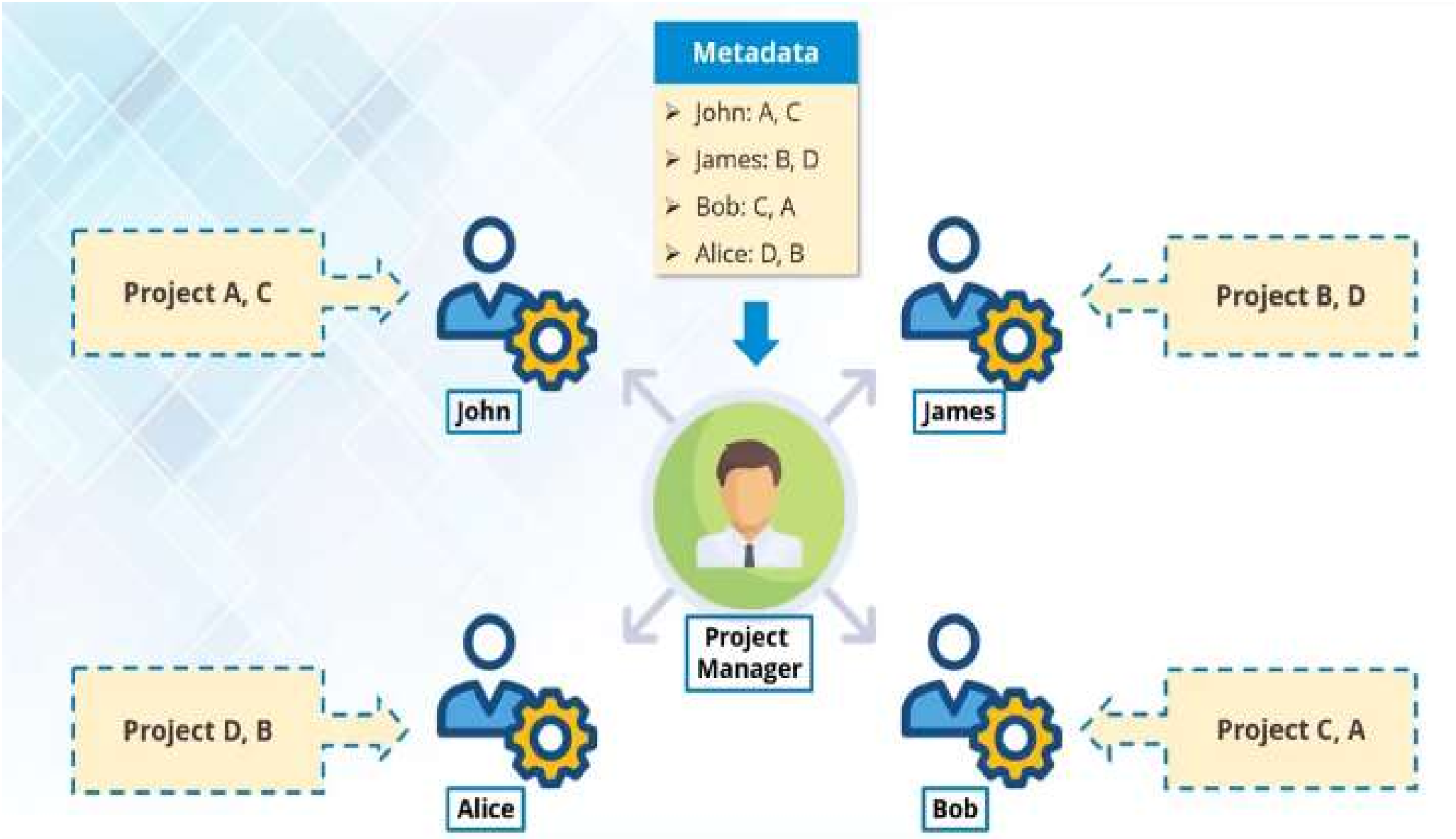
Master/Slave Approach

Scenario:

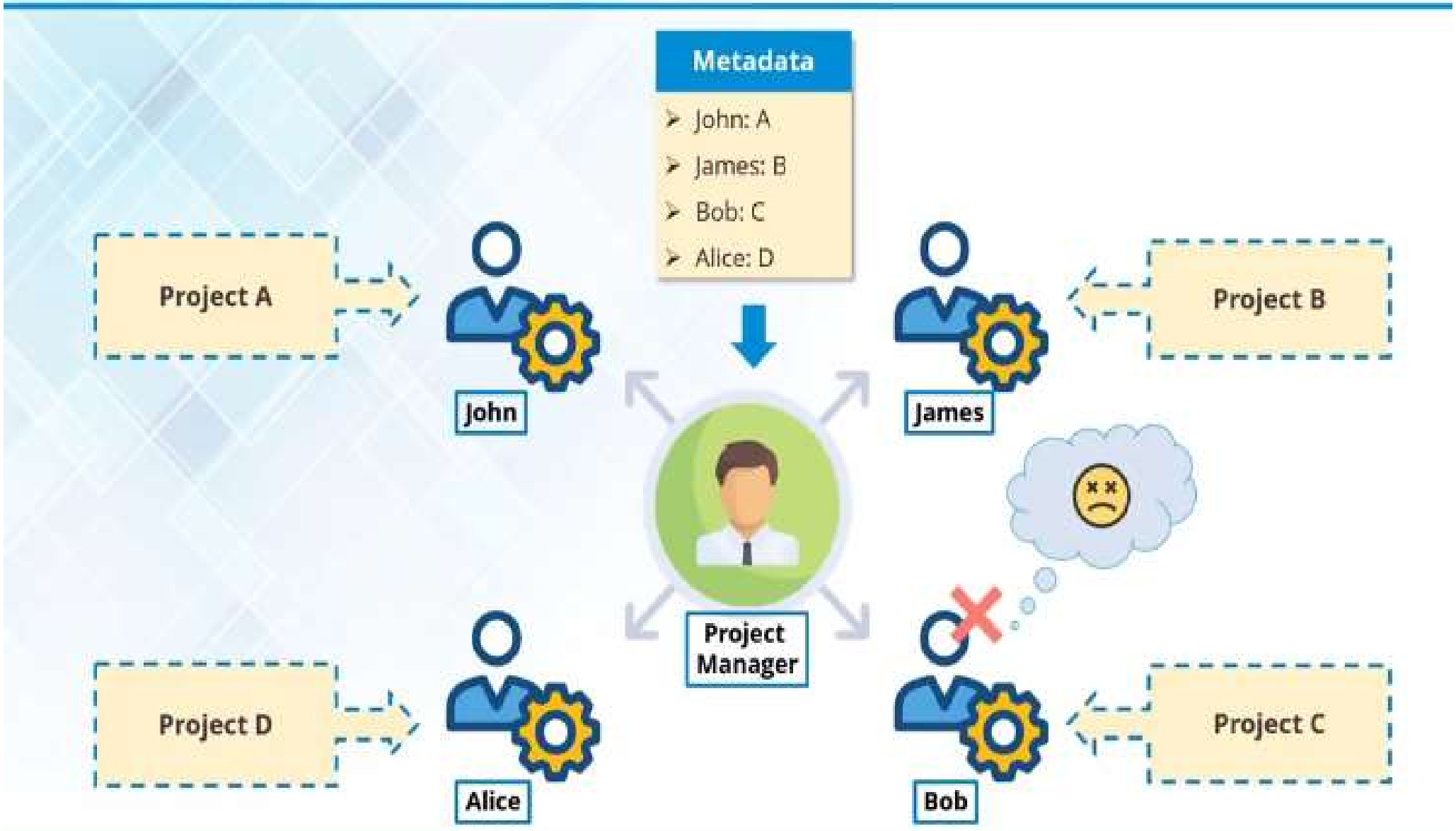
A project Manager managing a team of four employees. He assigns project to each of them and tracks the progress



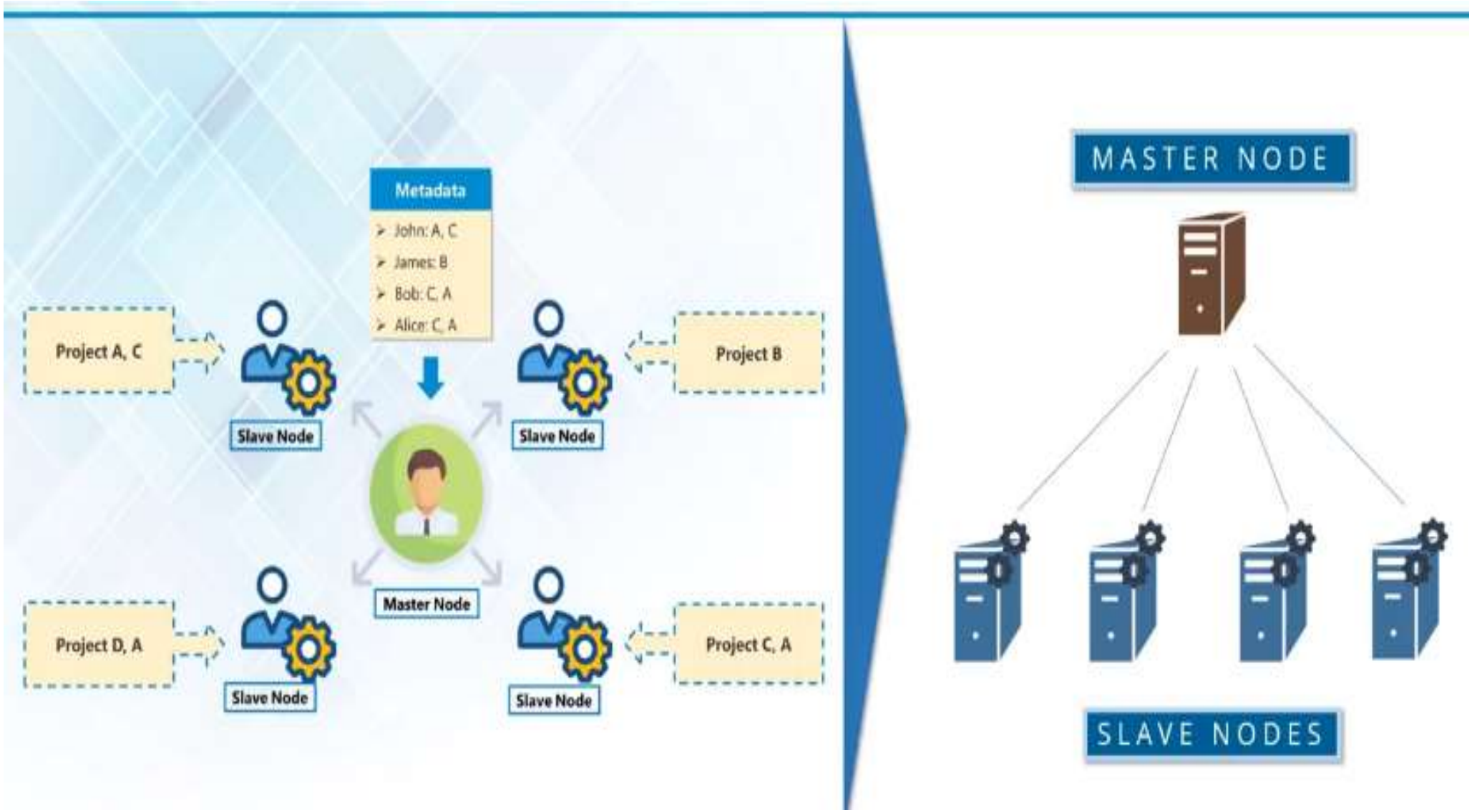
Maintain Metadata



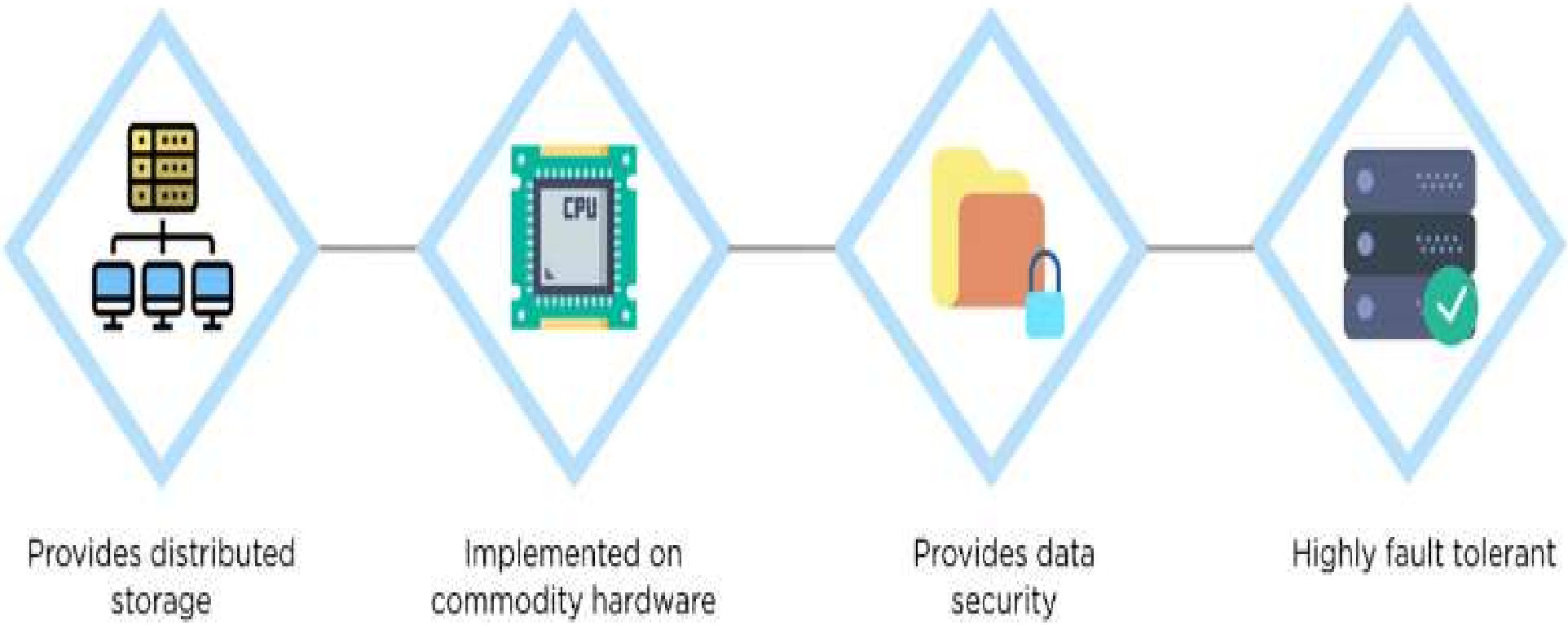
Backup Failover Solution



Evolution of Architecture

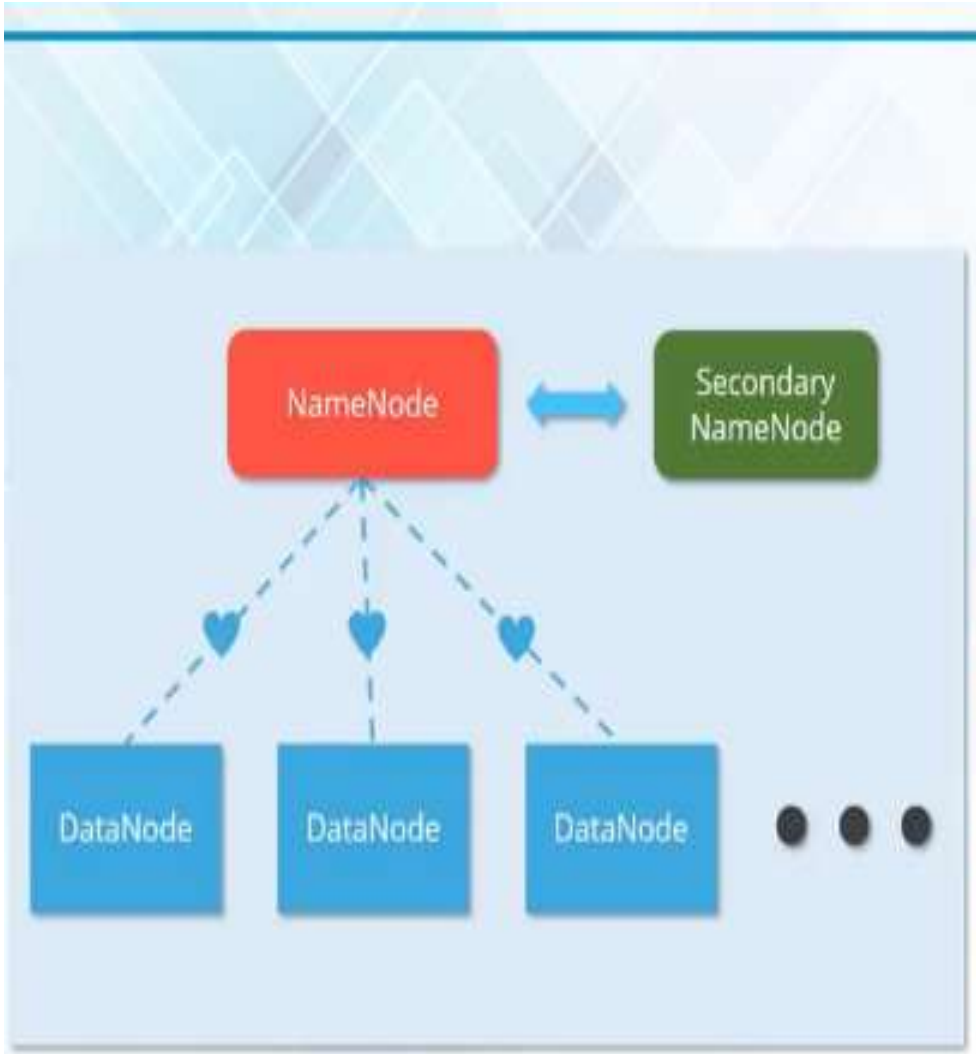


Evolution of HDFS



3 Core Components

Namenode & Datanode



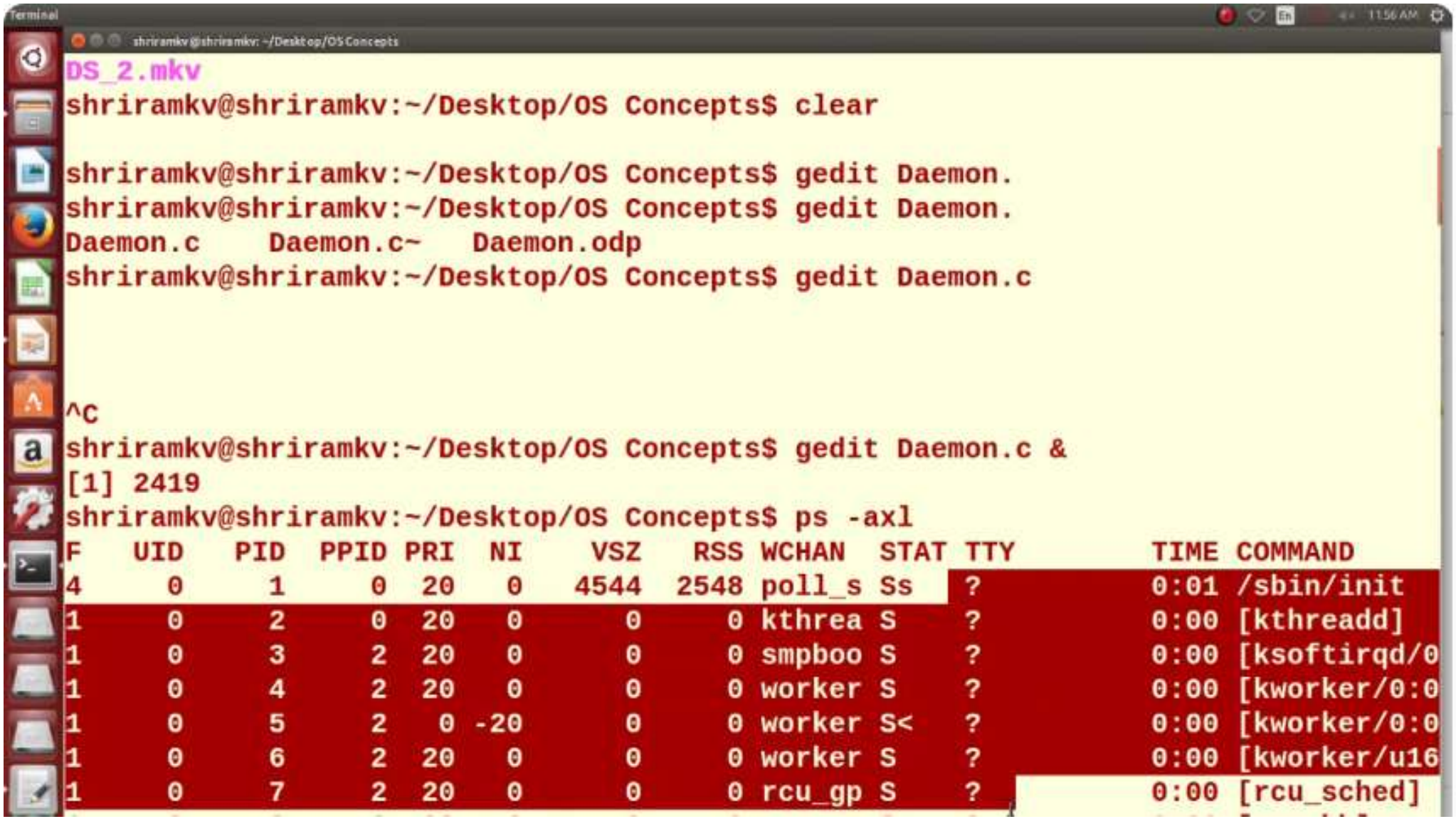
NameNode:

- 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

Daemon Services

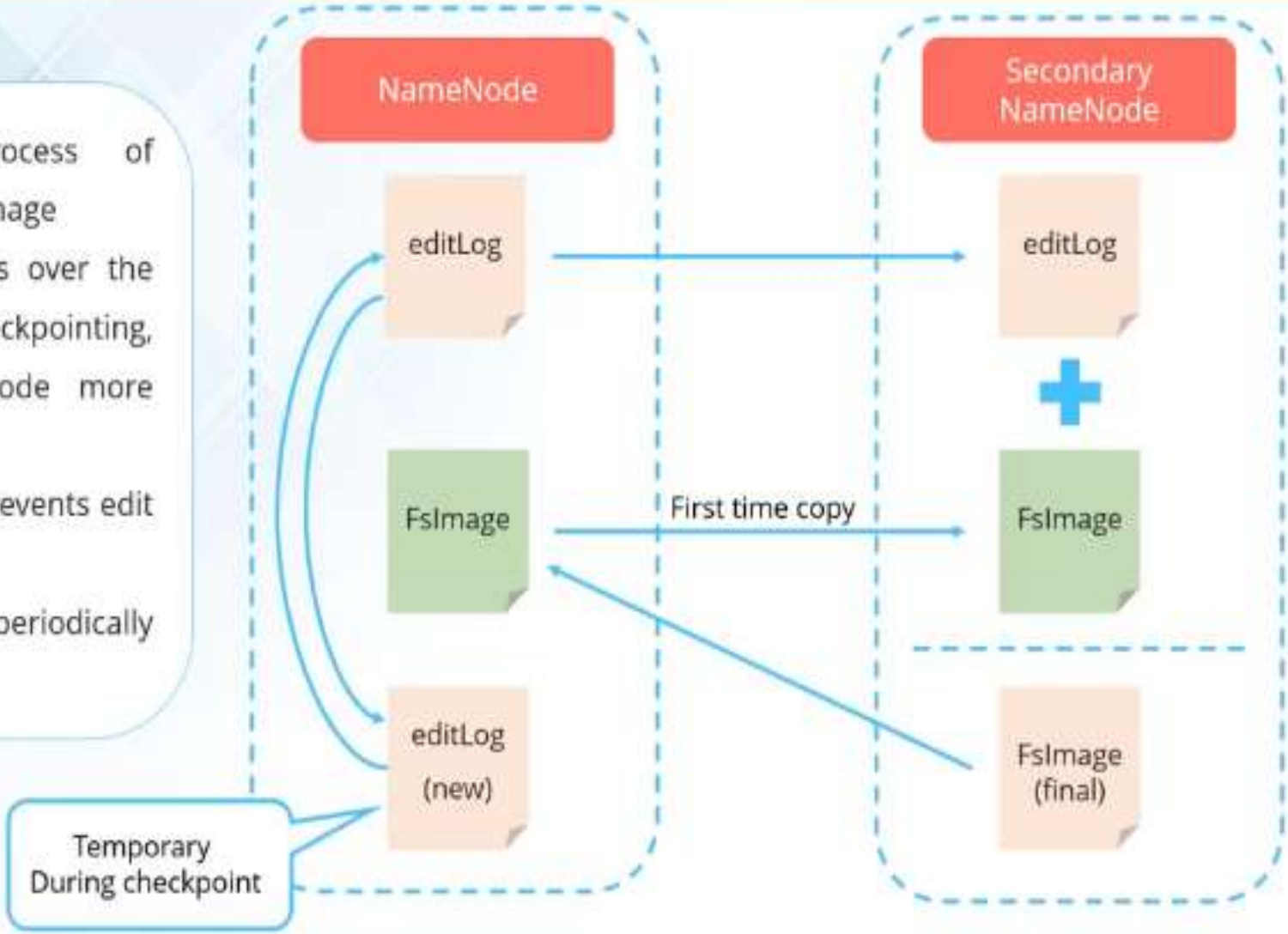


About Daemon Process

- It is a program.
- Runs in Unix/Linux without interruption or user intiation 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 exisitence)

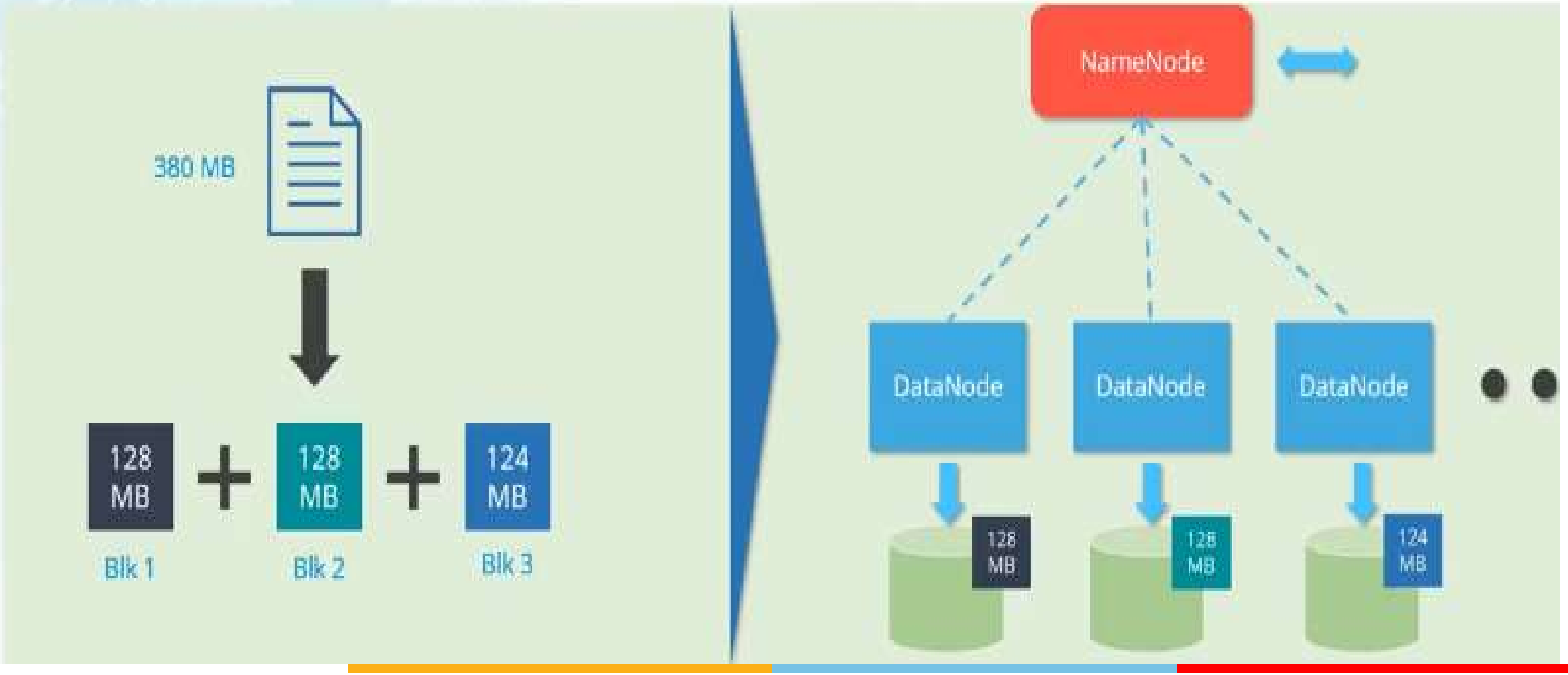
Secondary NameNode & Checkpoint

- 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)

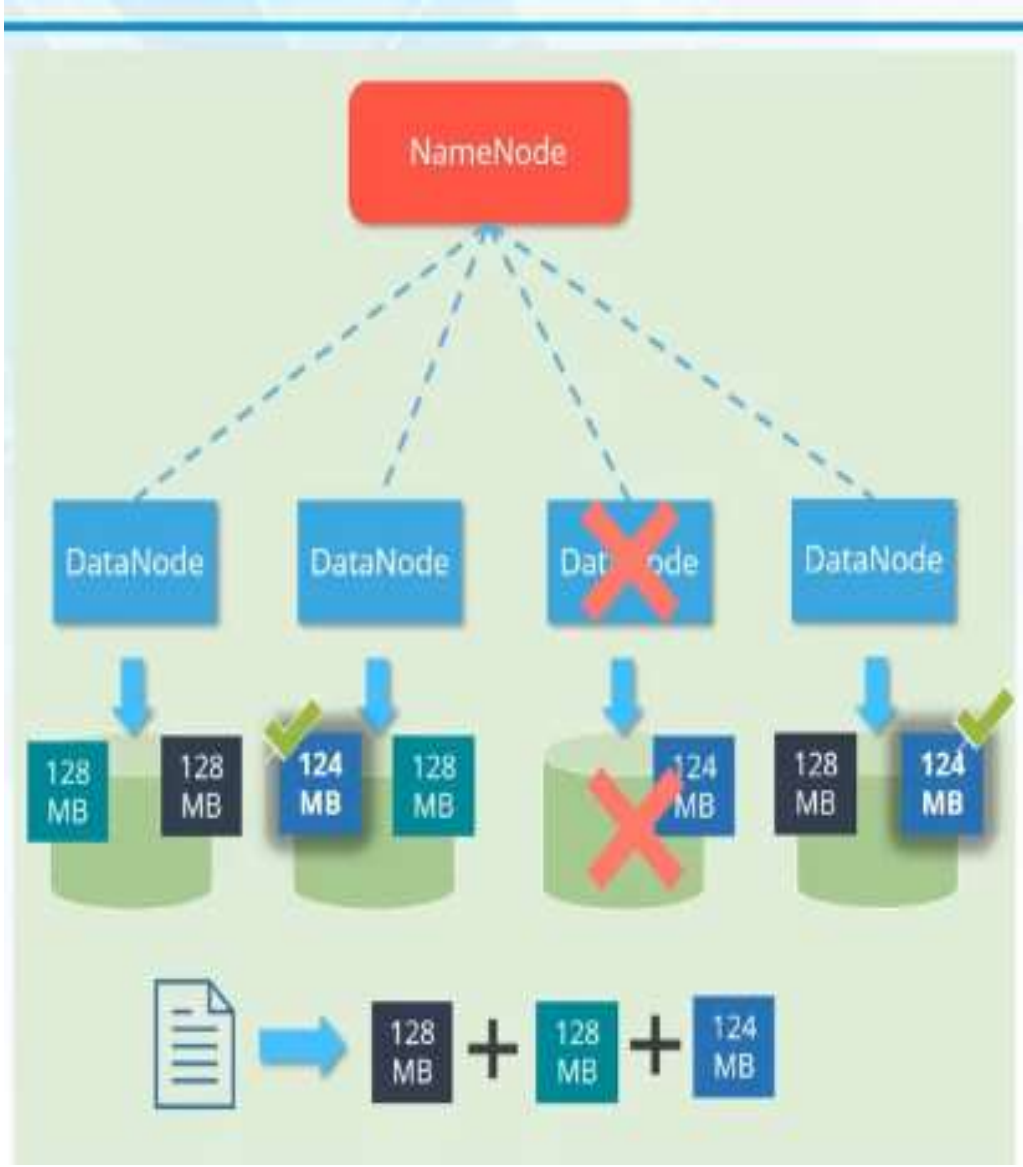


HDFS DataNode Blocks

- 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)



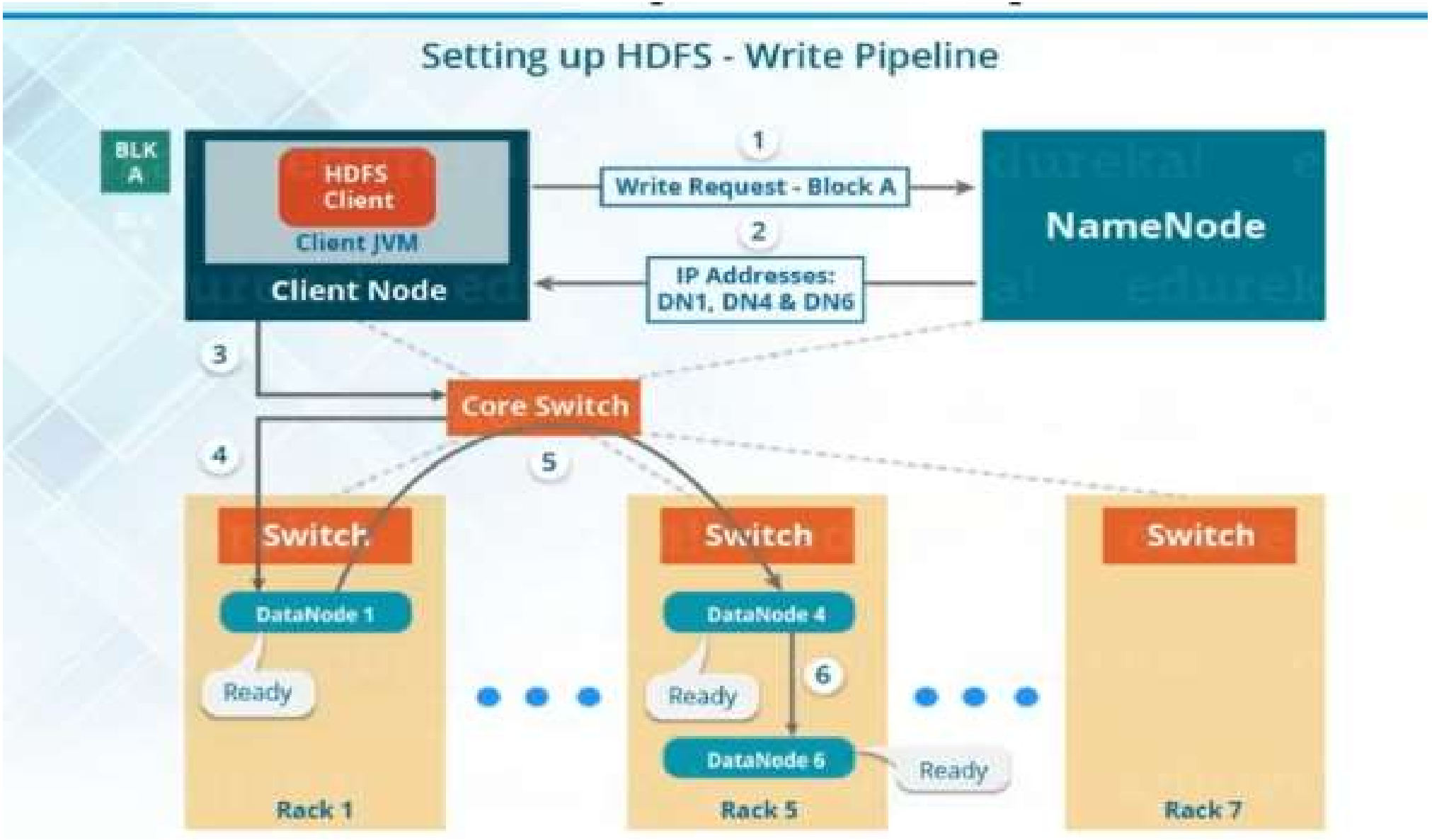
Replication



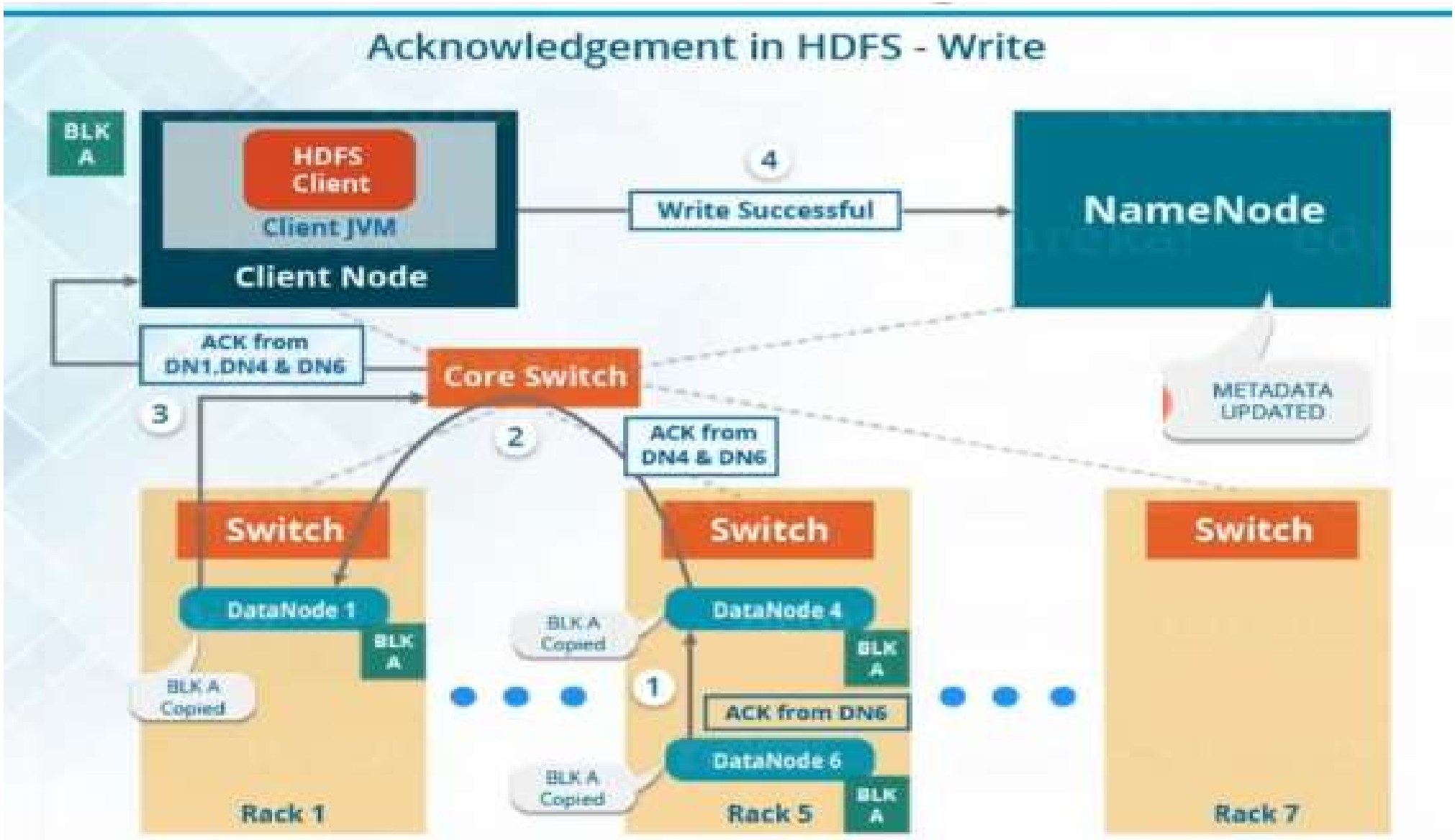
Solution:

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

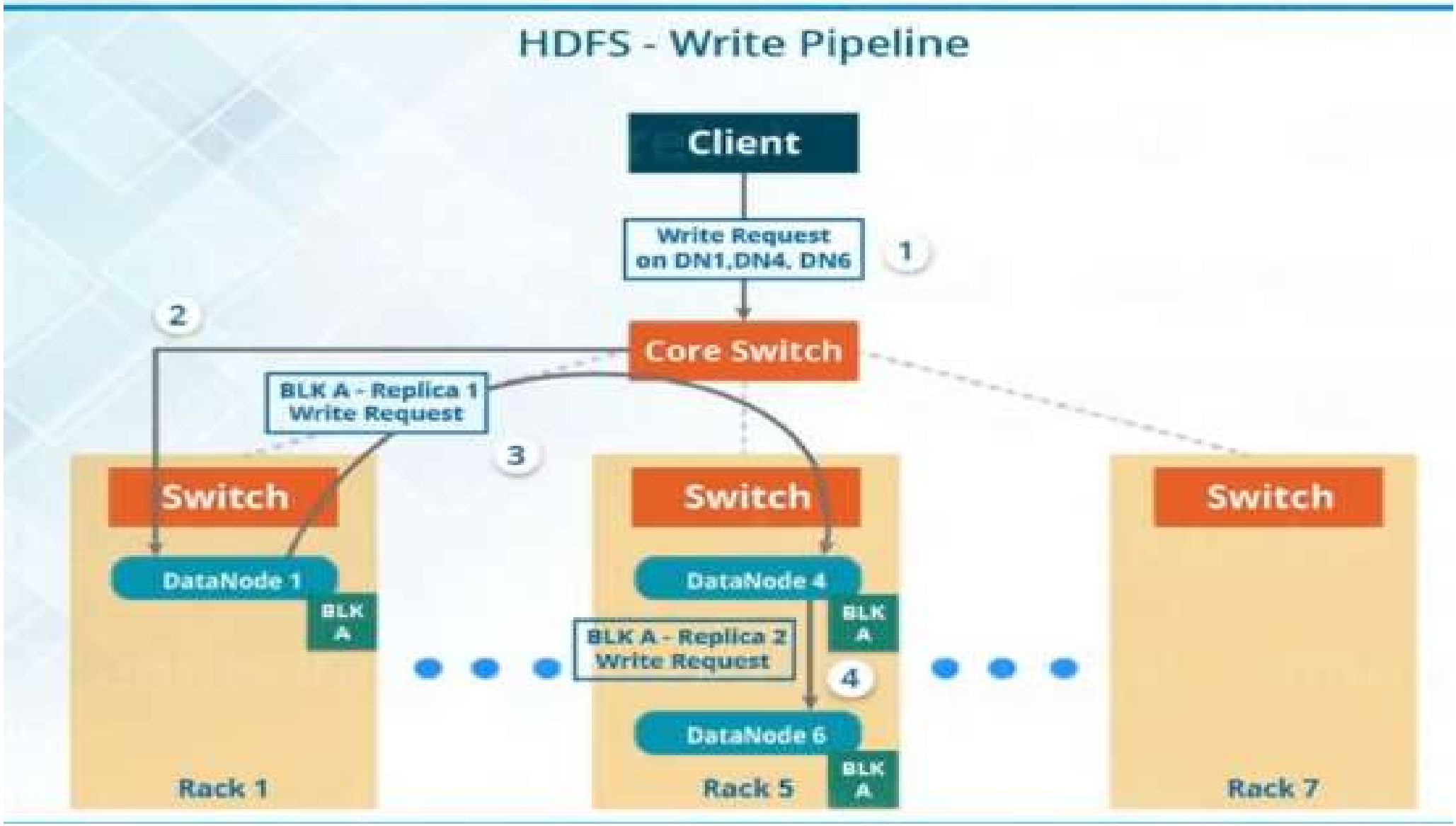
HDFS Write


Setting up HDFS - Write Pipeline
BLK A
HDFS Client
Client JVM
Client Node
1
Write Request - Block A
2
IP Addresses: DN1, DN4 & DN6
NameNode
3
Core Switch
4
5
Switch
DataNode 1
Ready
Rack 1
Switch
DataNode 4
Ready
6
DataNode 6
Ready
Rack 5
Switch
Rack 7

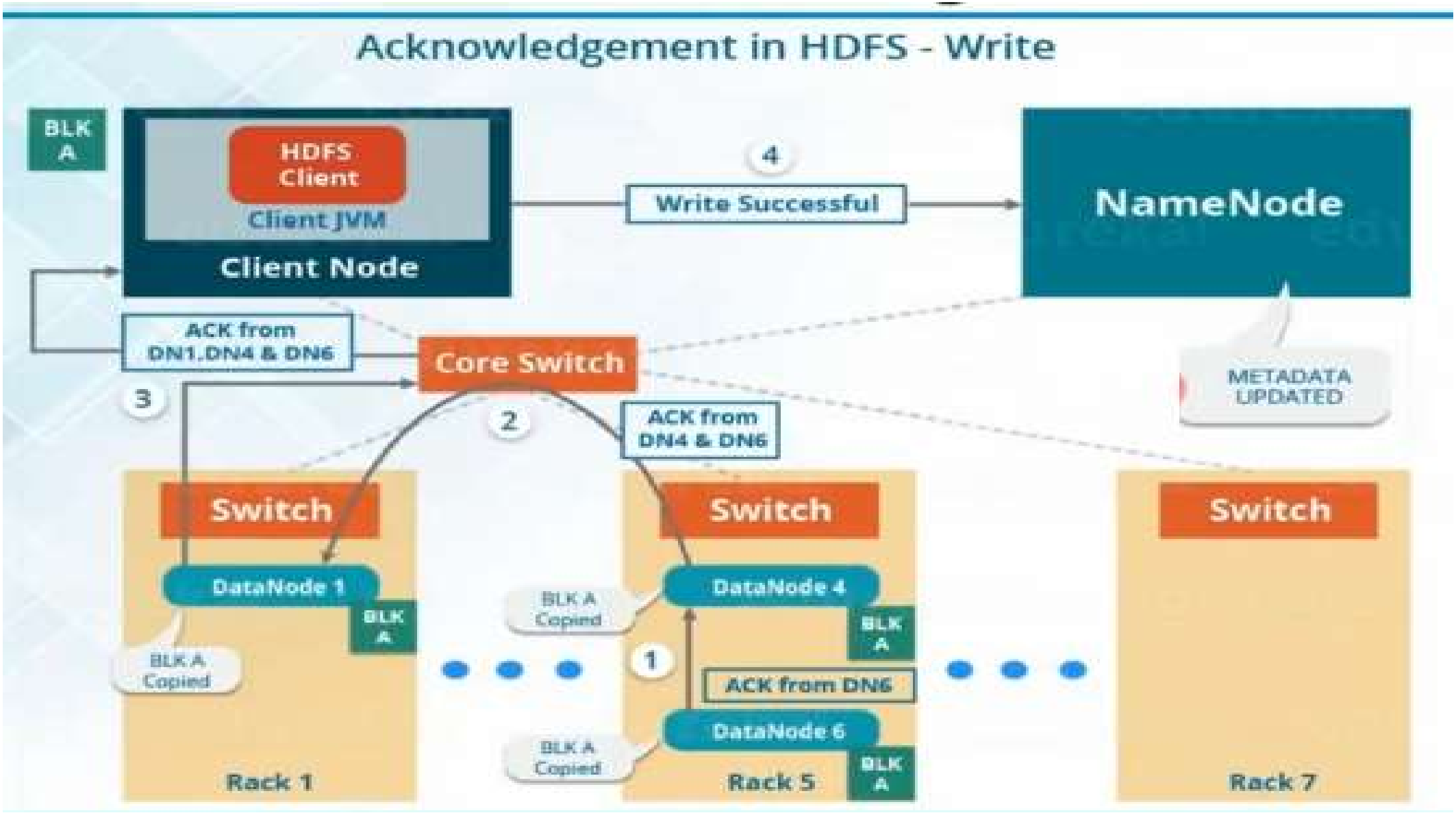
HDFS Write - Ack



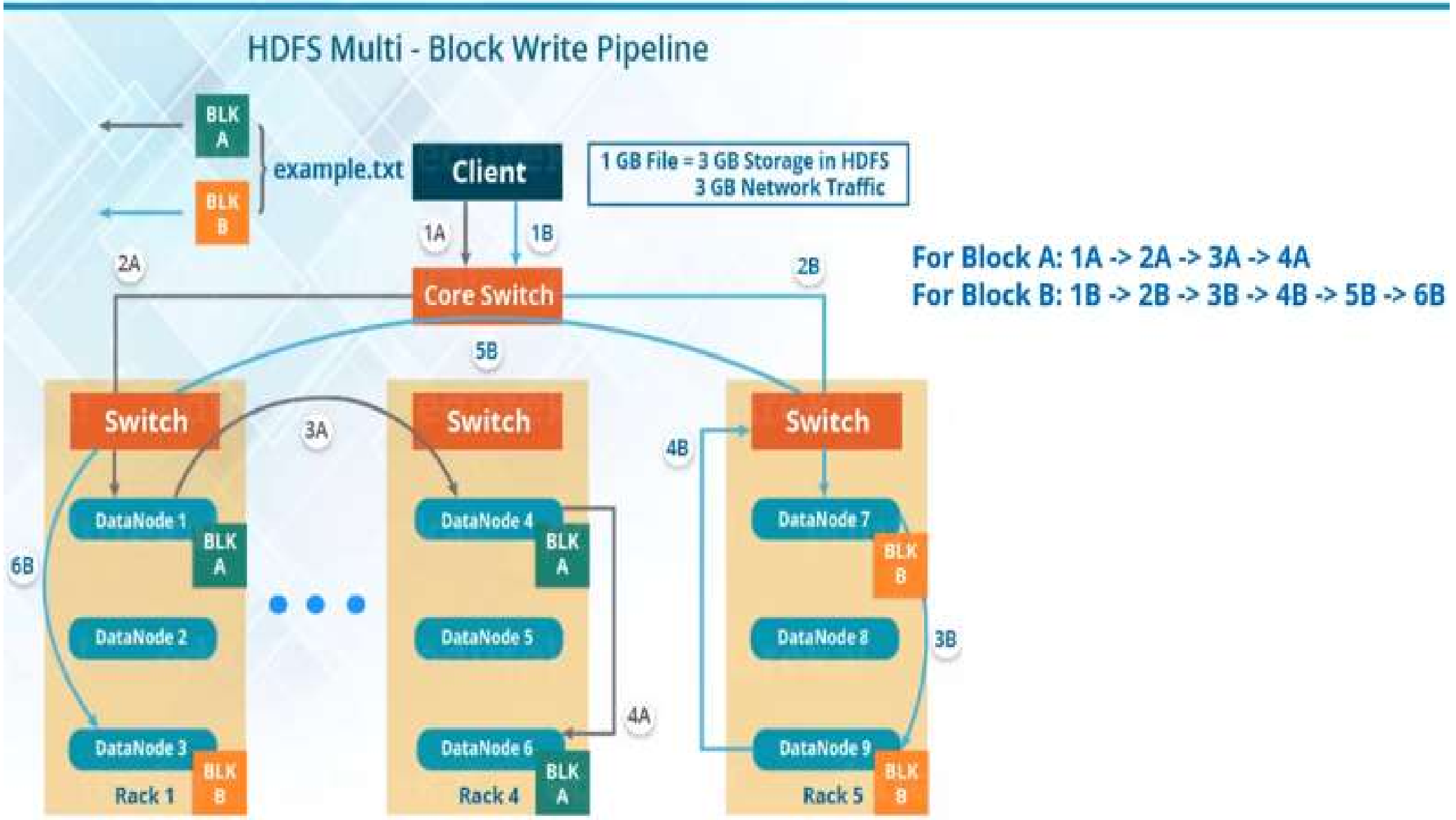
Write Request



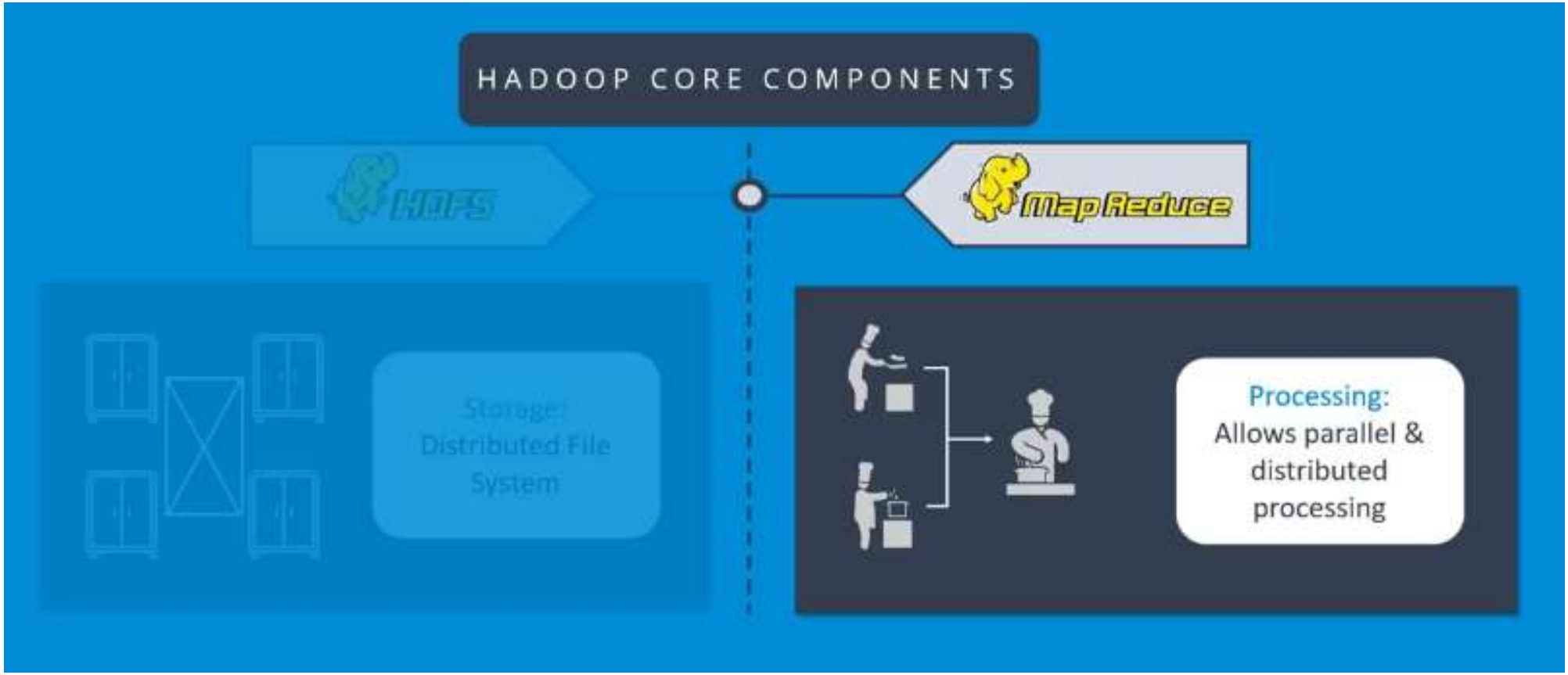
Ack


Acknowledgement in HDFS - Write
BLK A
HDFS Client
Client JVM
Client Node
4
Write Successful
NameNode
ACK from DN1,DN4 & DN6
Core Switch
3
2
ACK from DN4 & DN6
METADATA UPDATED
Switch
Switch
Switch
DataNode 1
BLK A
BLK A Copied
BLK A Copied
DataNode 4
BLK A
1
ACK from DN6
DataNode 6
BLK A Copied
BLK A
Rack 1
Rack 5
Rack 7

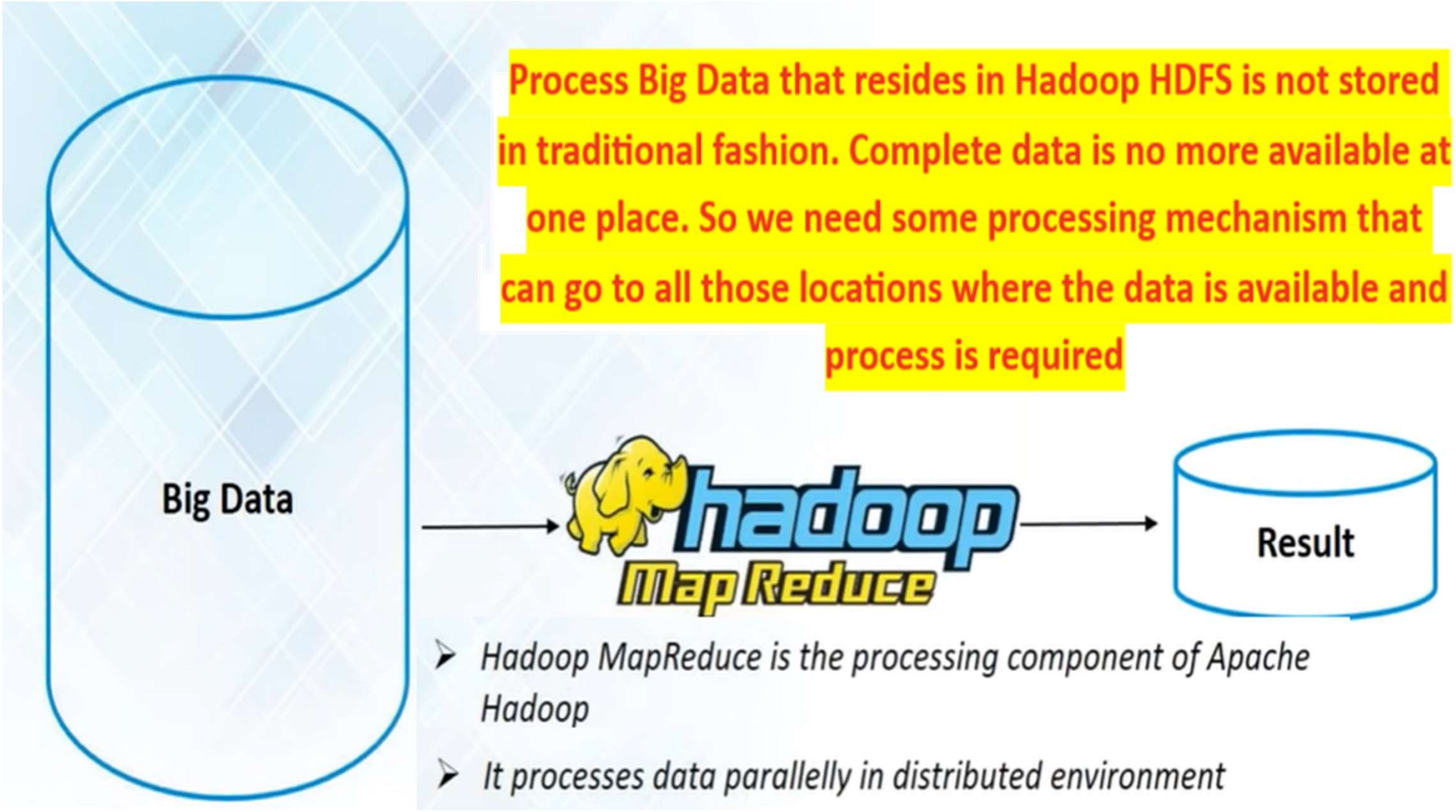
Multi Data Block write



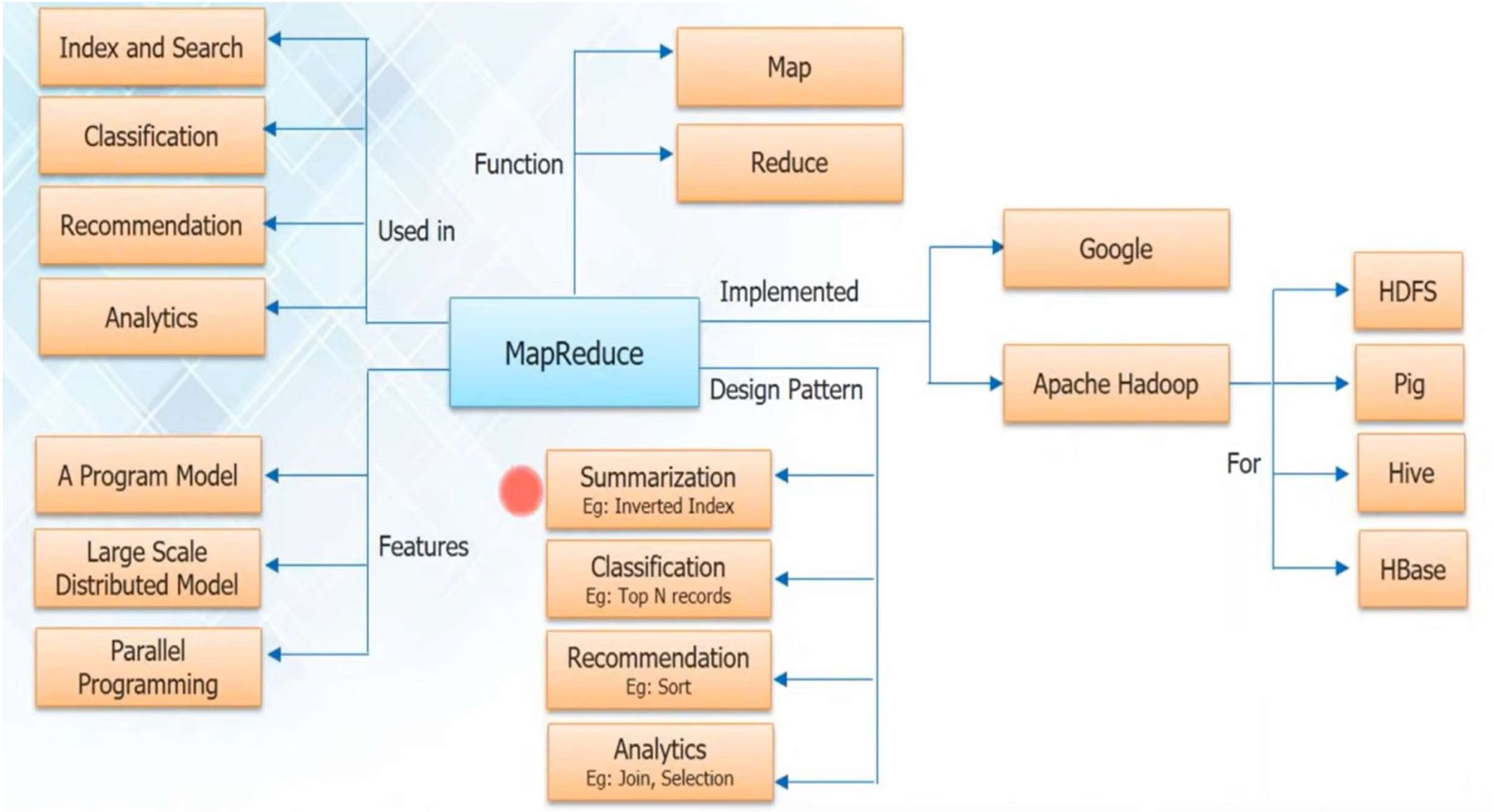
MapReduce Layer



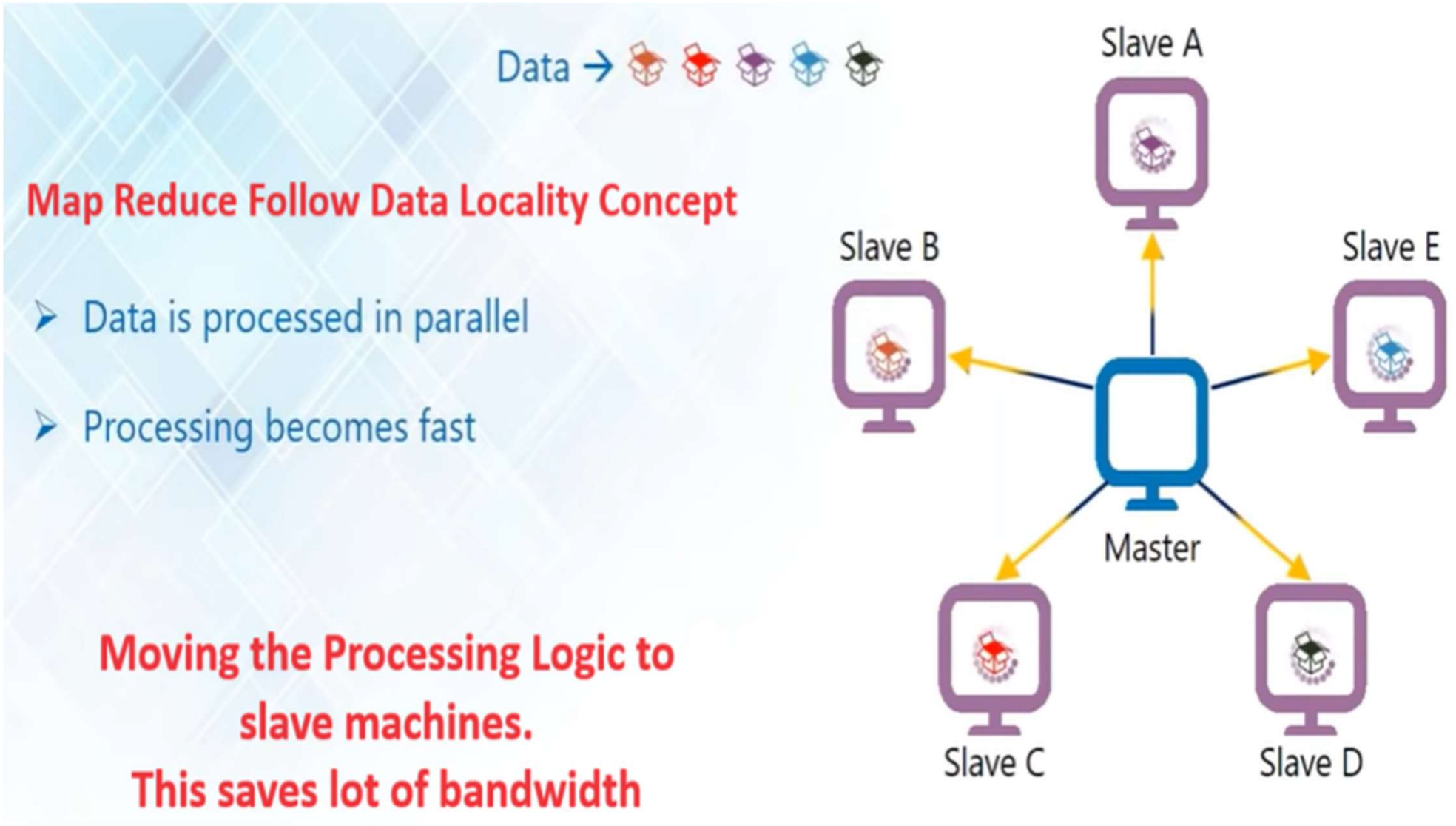
Why Map-Reduce?



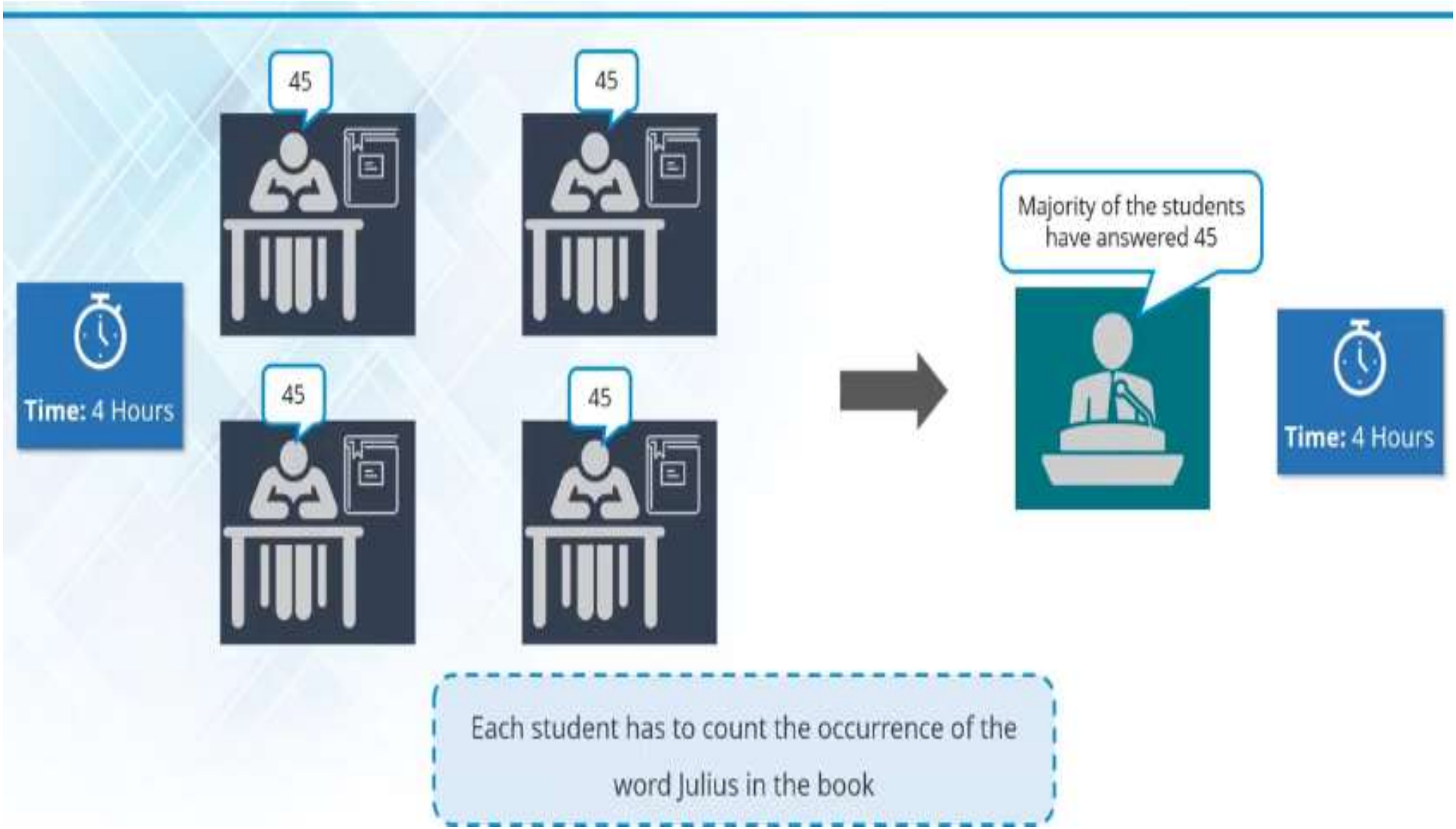
Bird Eye View



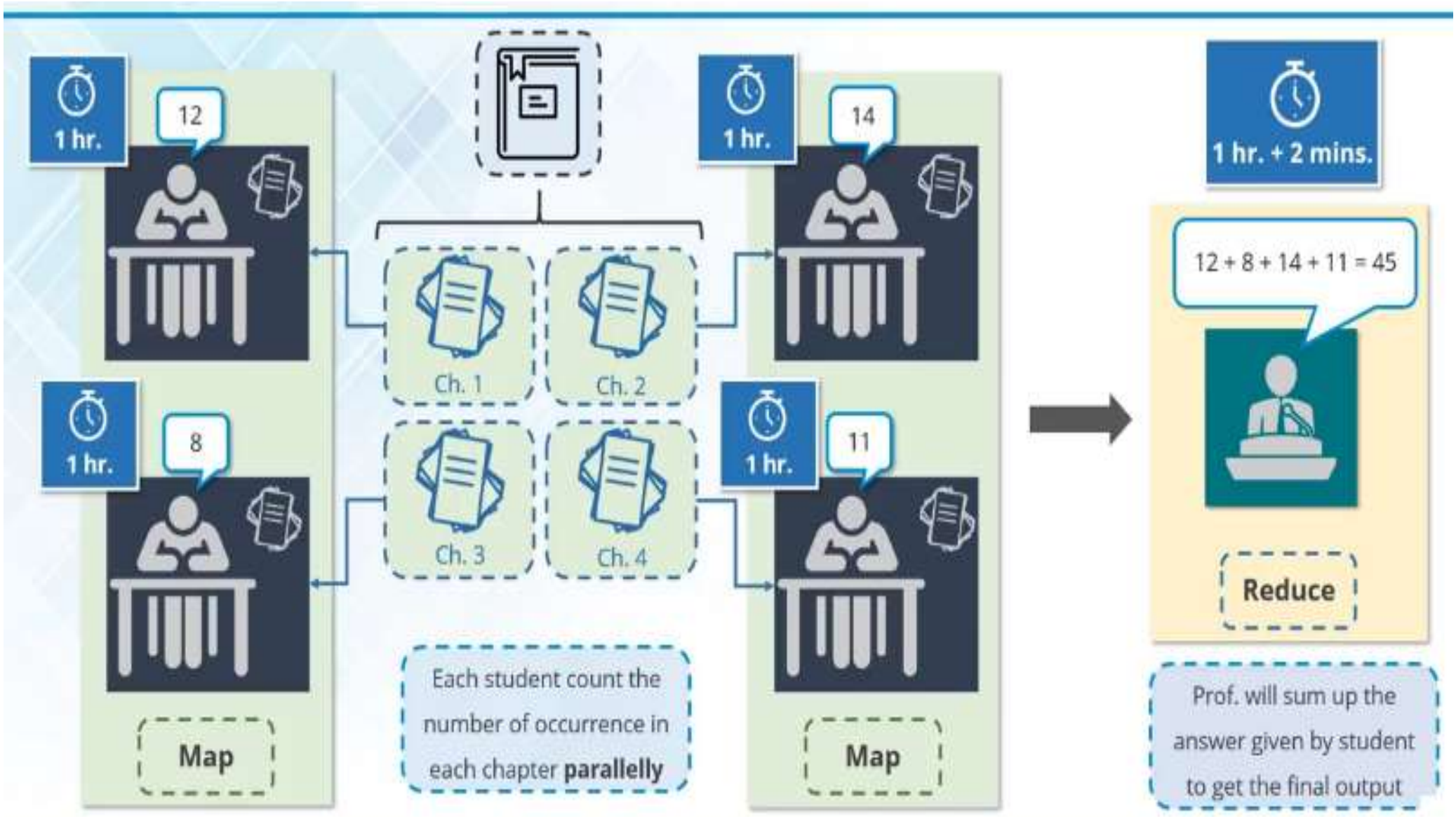
Parallel Processing scenario



Word Count in a Book – Normal mode

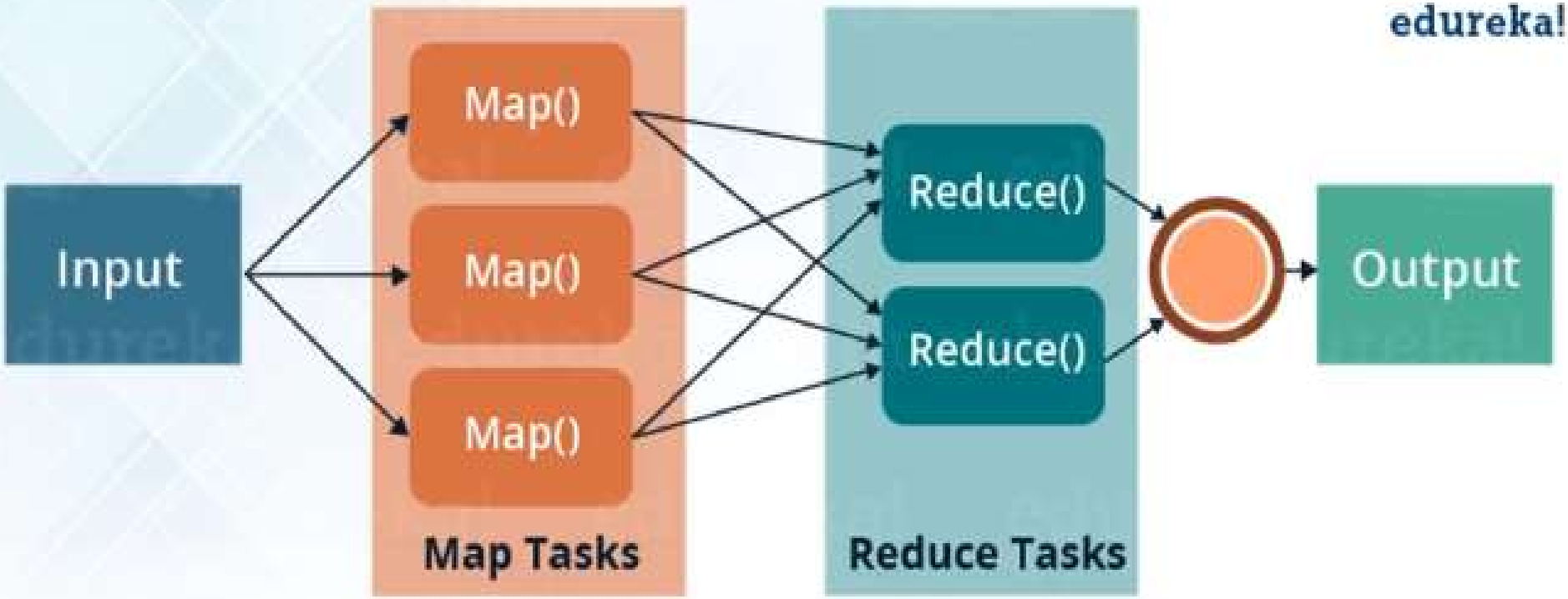


MapReduce Operation - Example

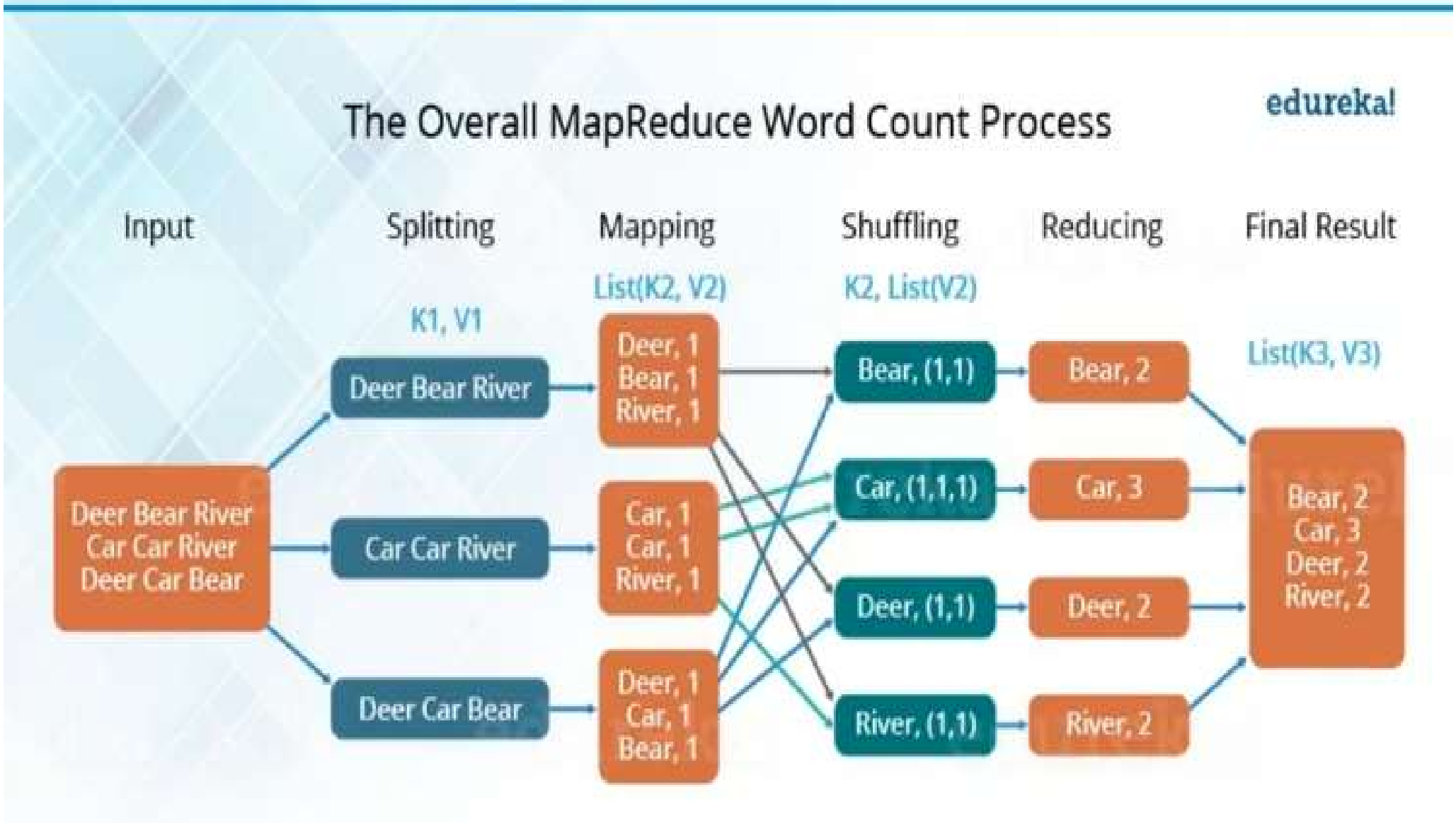


Exact Flow - MapReduce

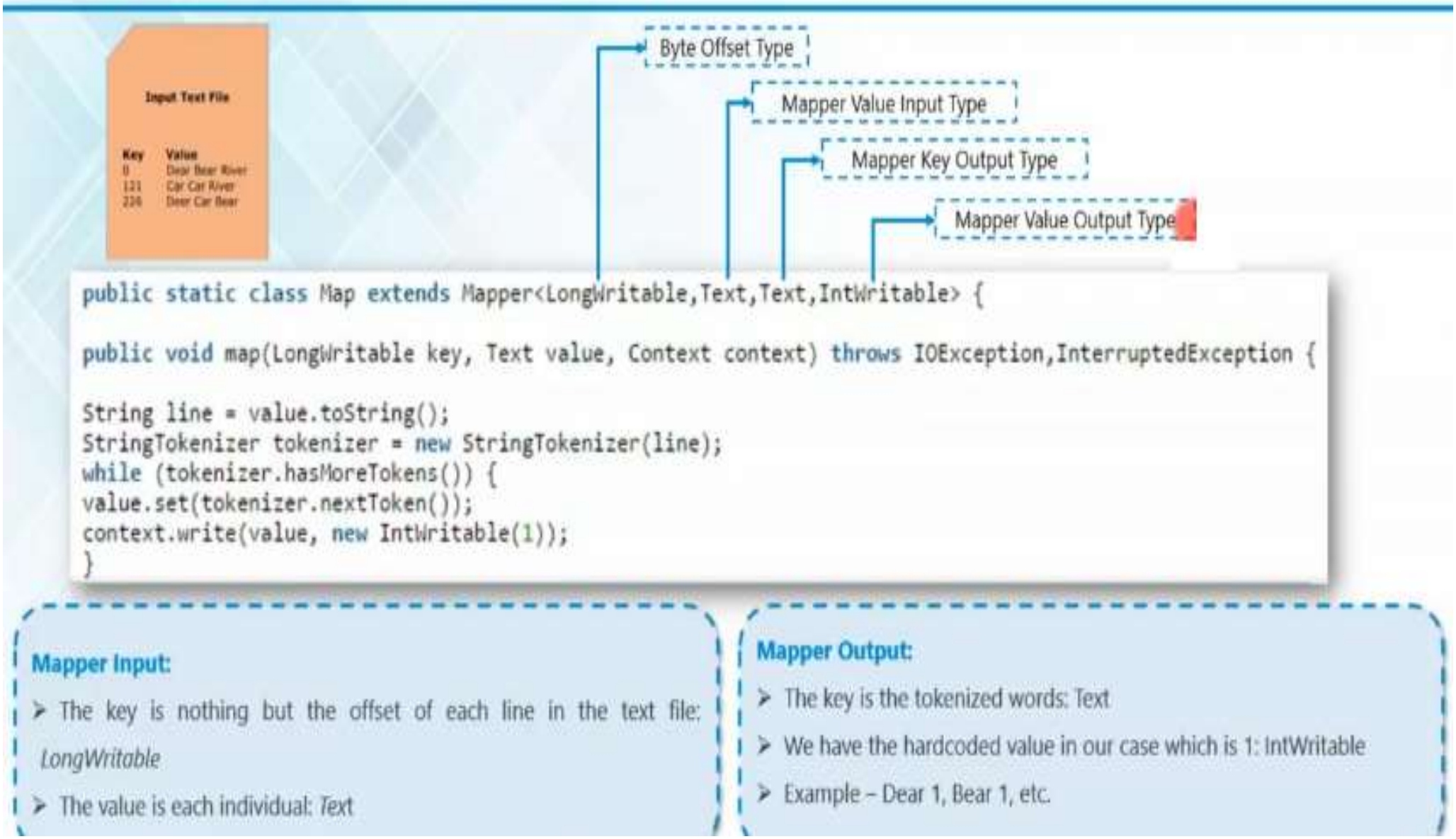
MapReduce is a programming framework that allows us to perform distributed and parallel processing on large data sets in a distributed environment



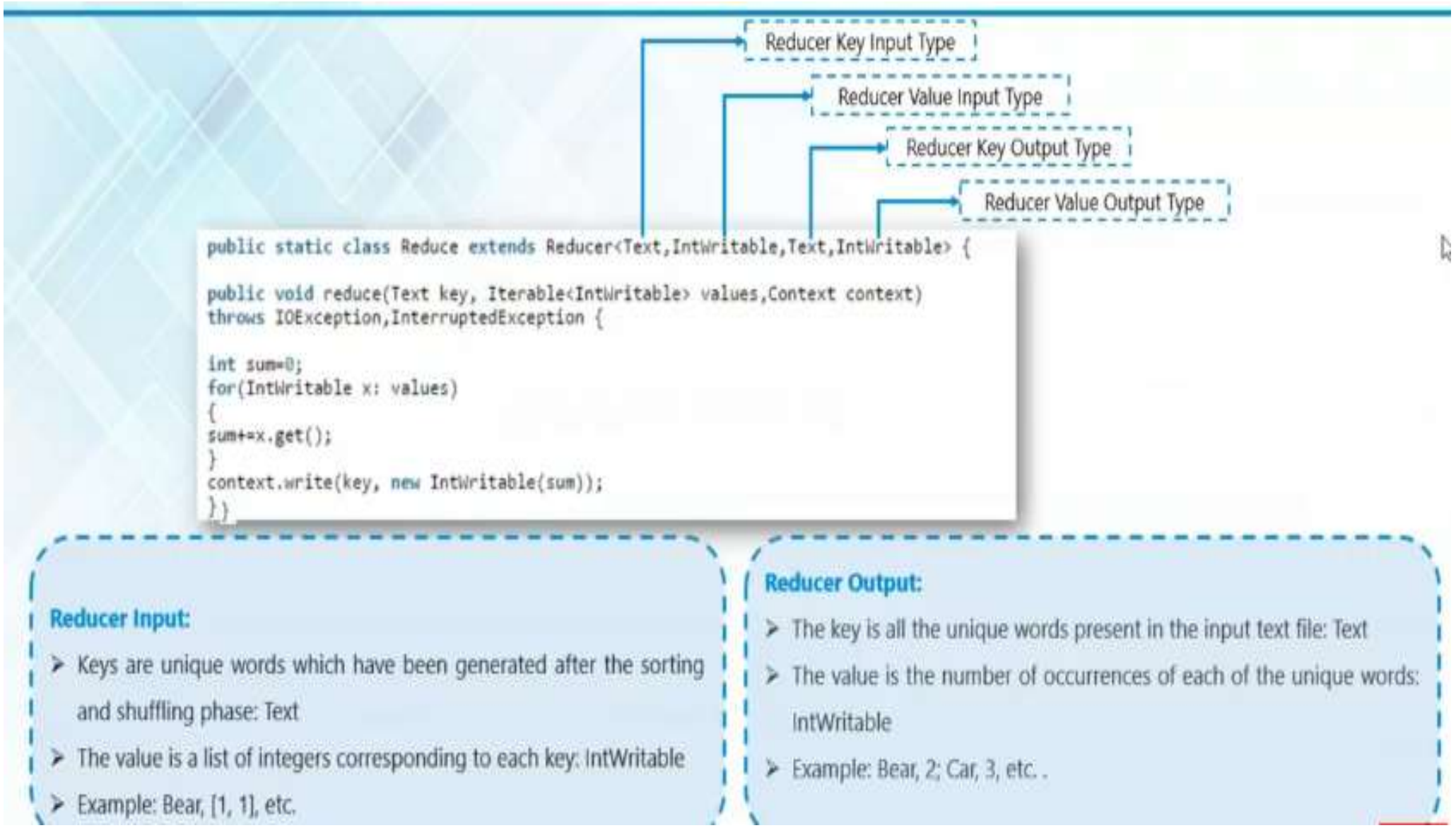
Steps with example



Mapper Code –Word Count Example



Reducer Code –Word Count Example



Driver Code for Configuration

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

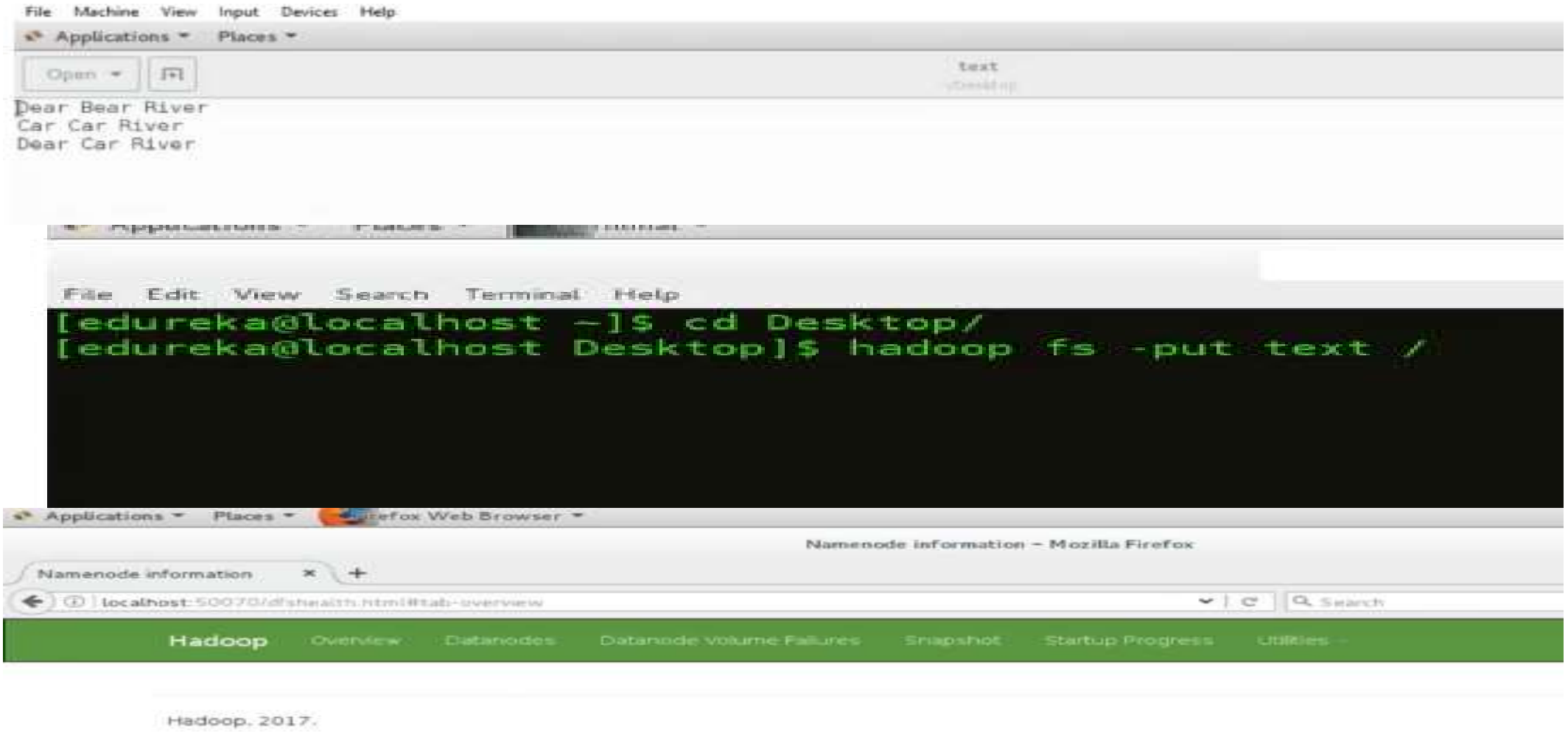
```
Configuration conf= new Configuration();
Job job = new Job(conf,"My Word Count Program");
job.setJarByClass(WordCount.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setOutputKeyClass(Text.class);

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]));
```

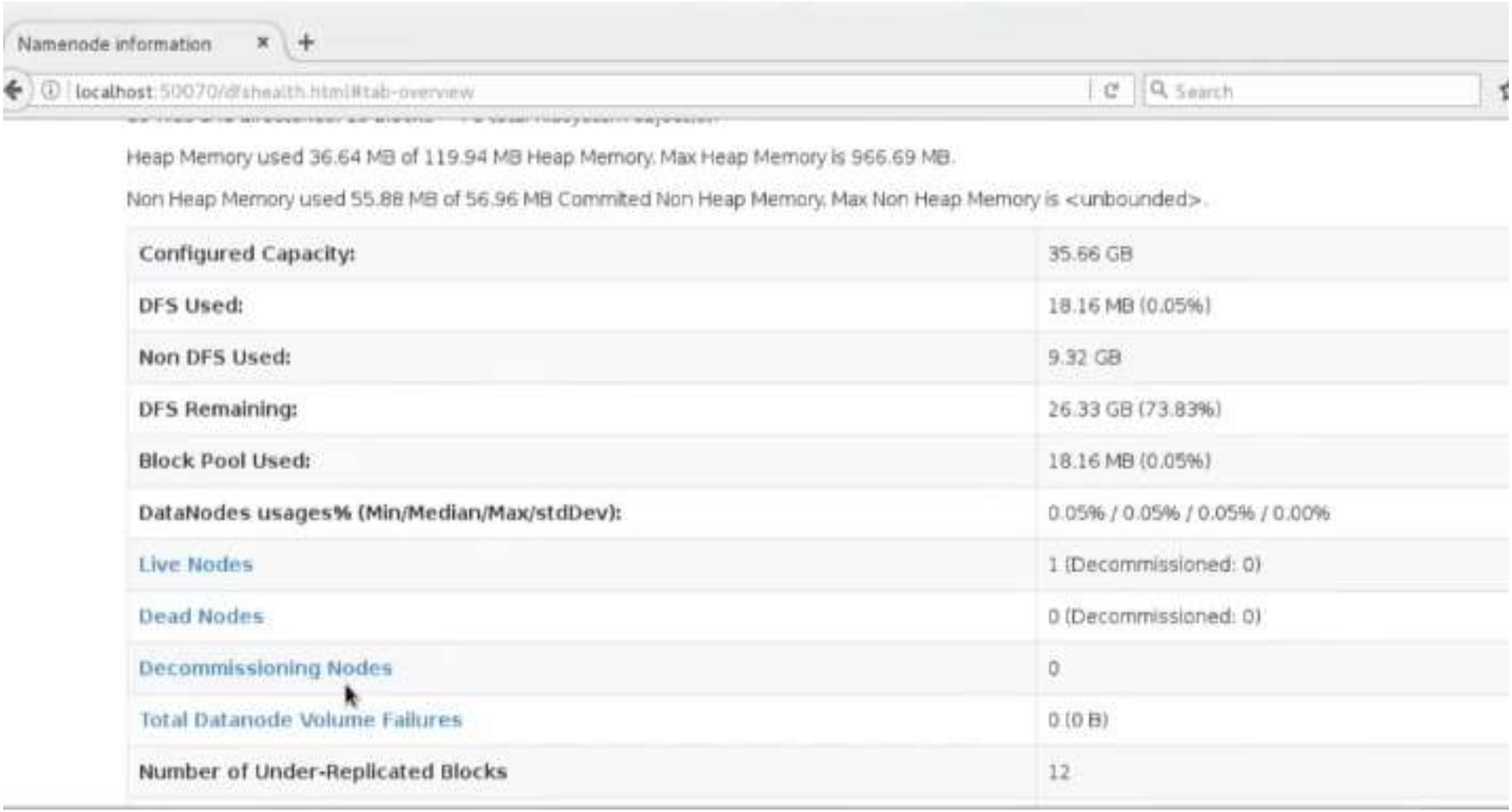
- Specify the name of the job , the data type of input/output of the mapper and reducer
- Specify the names of the mapper and reducer classes.
- Path of the input and output folder
- The method setInputFormatClass () is used for specifying the unit of work for mapper
- Main() method is the entry point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

Sample Text File

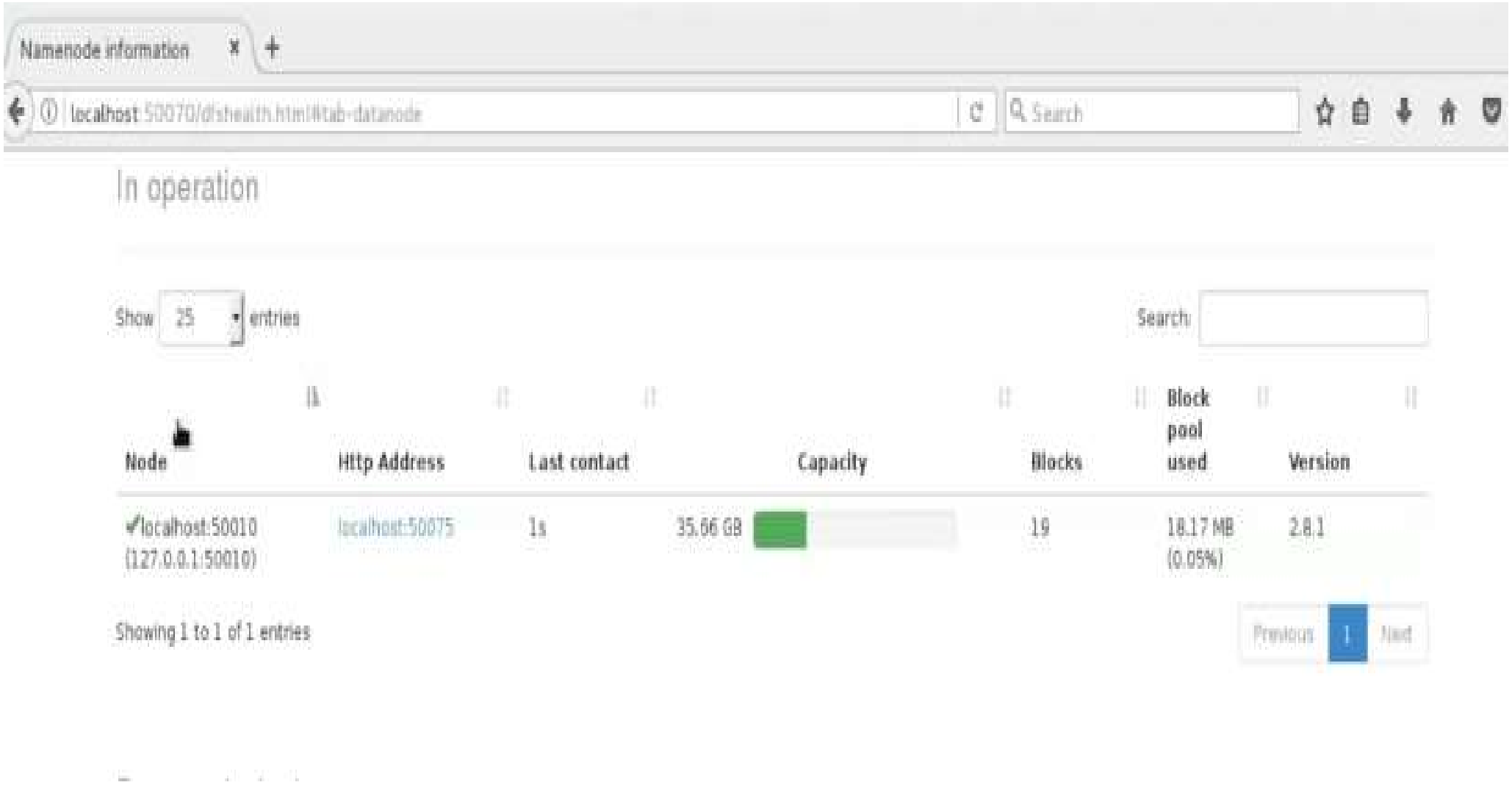


Hadoop, 2017.

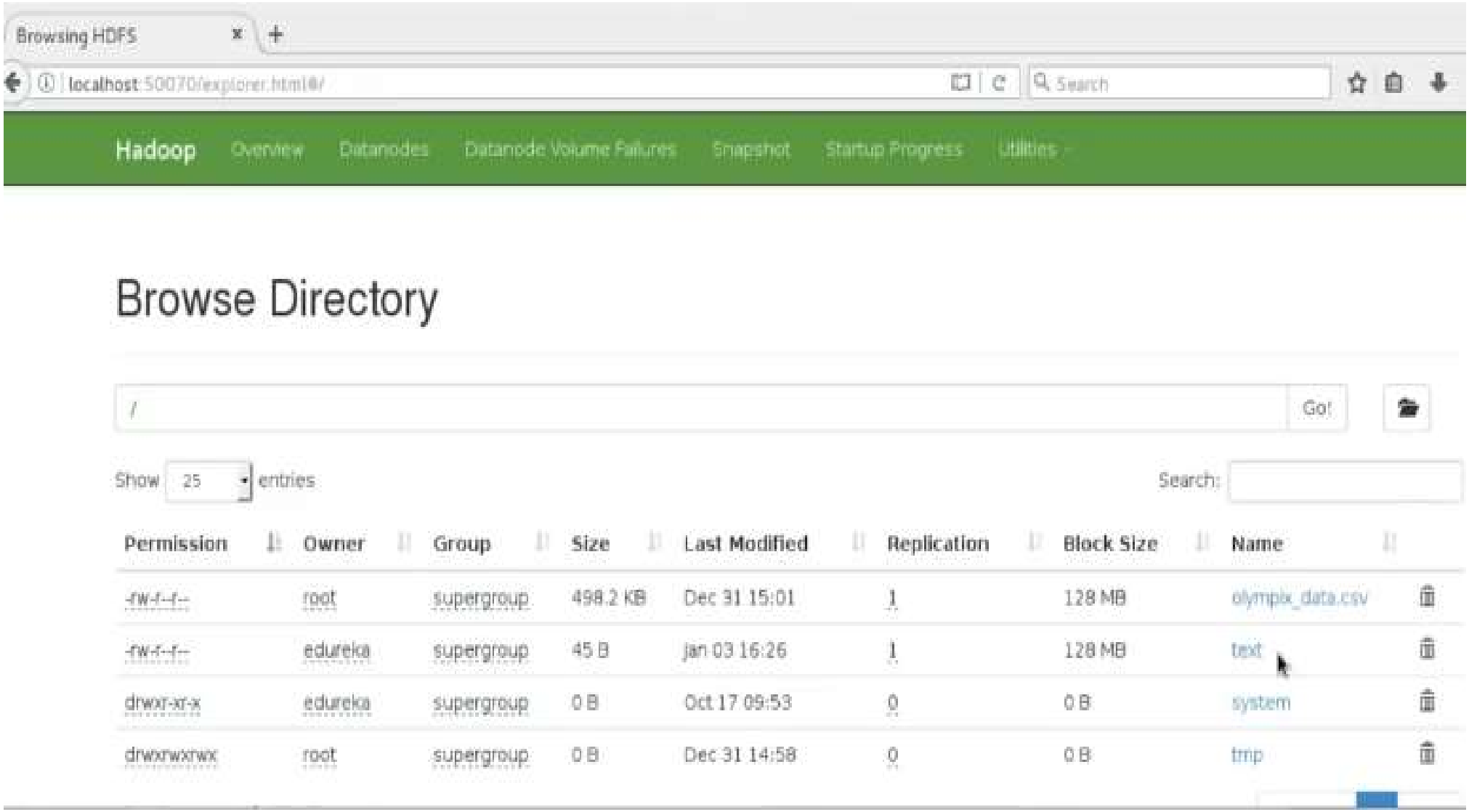
NameNode Summary



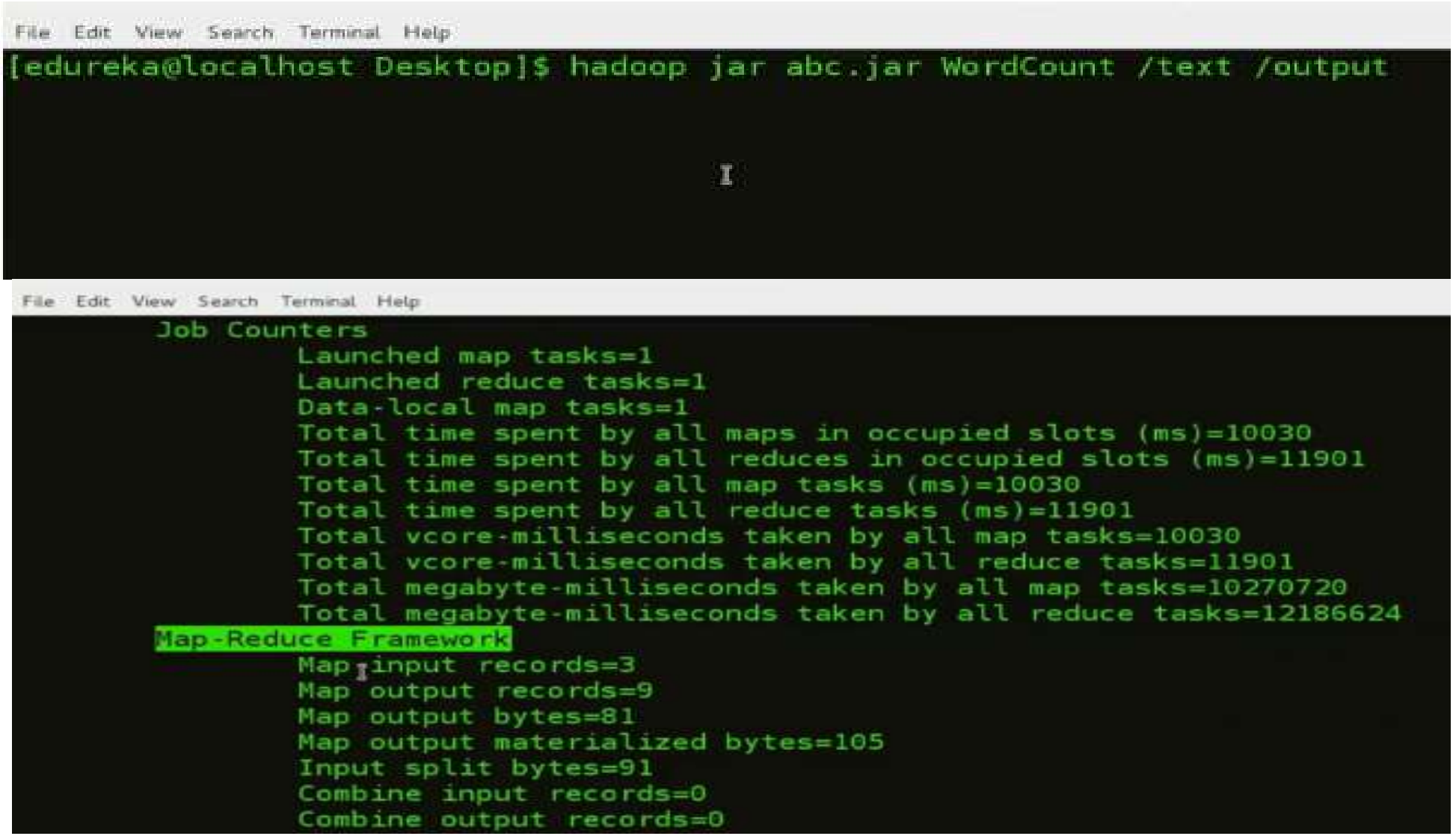
Live Node Details



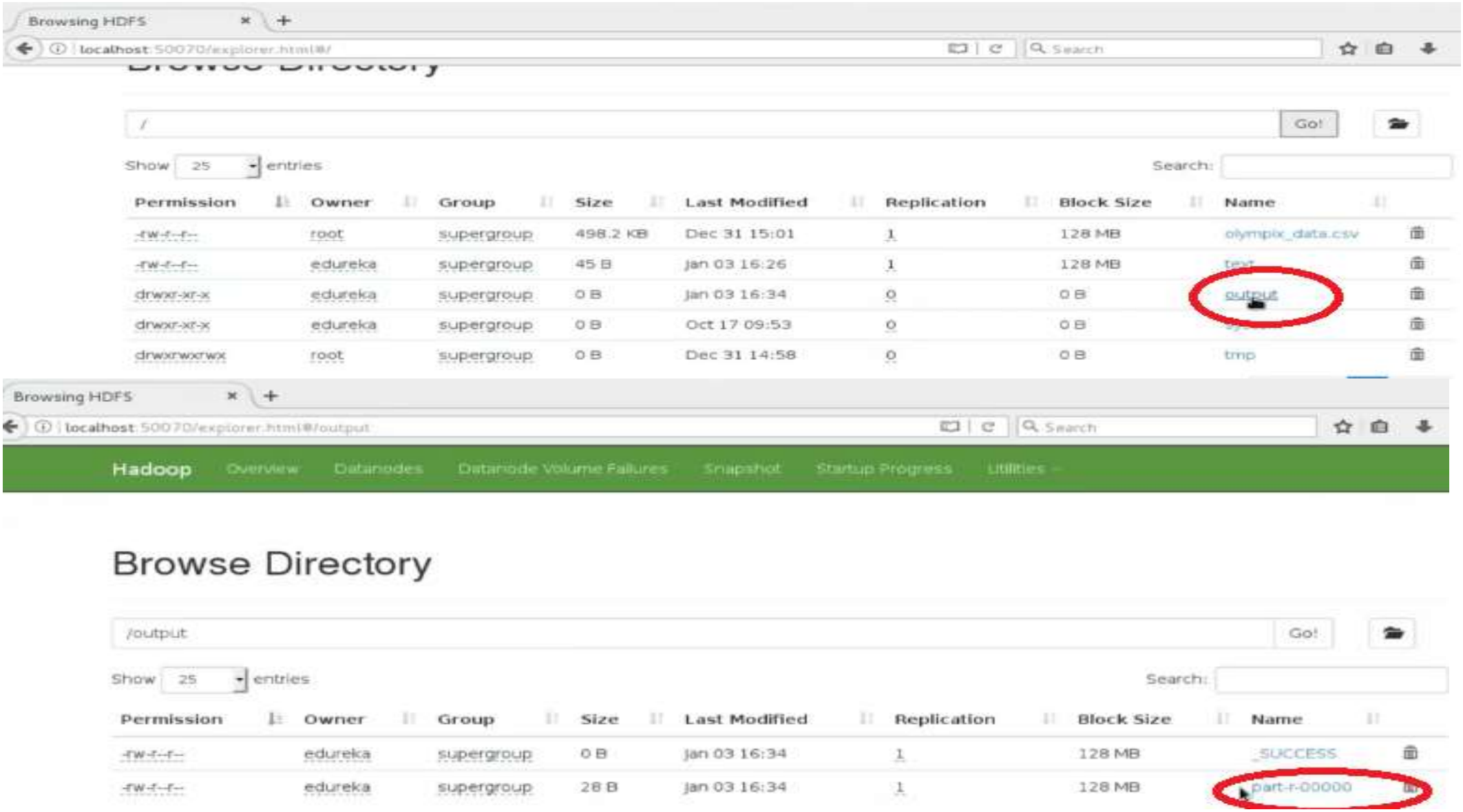
NameNode Files



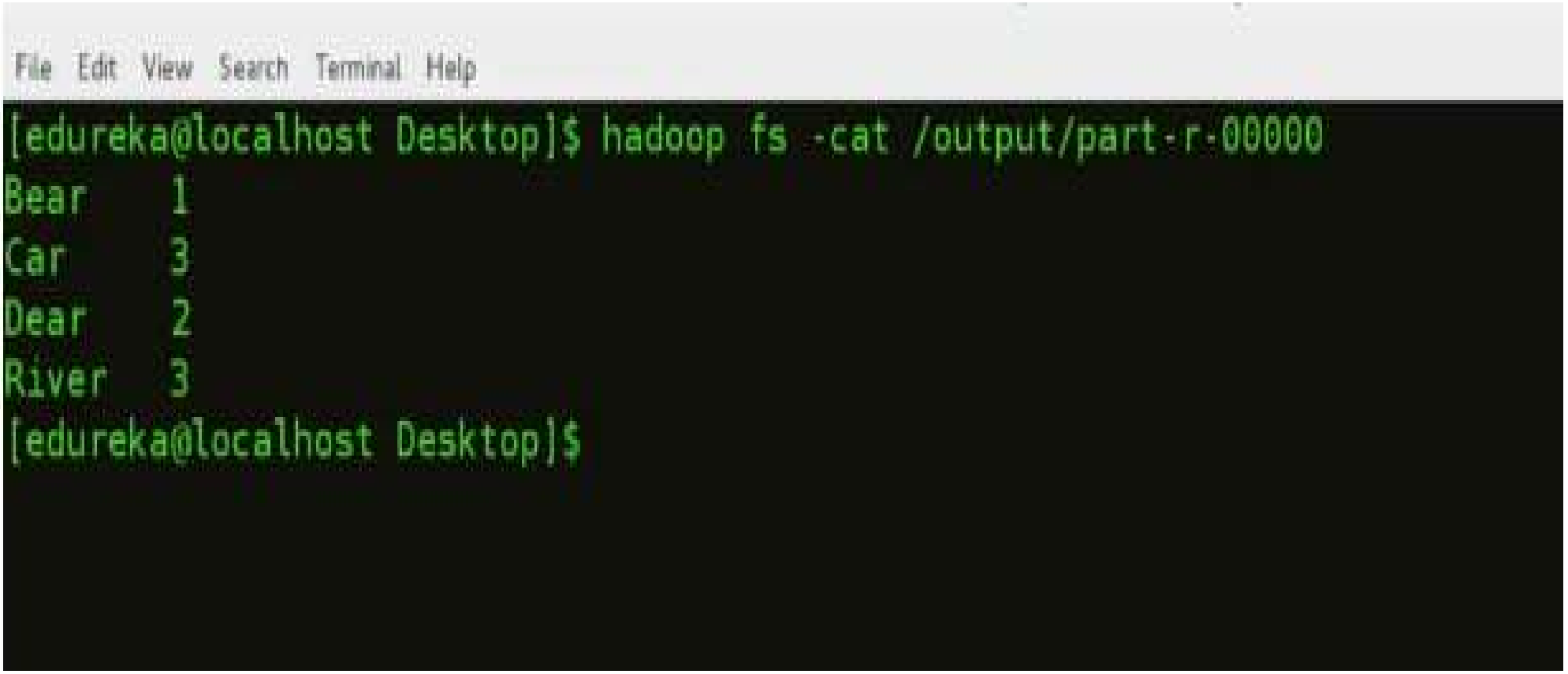
Execution


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

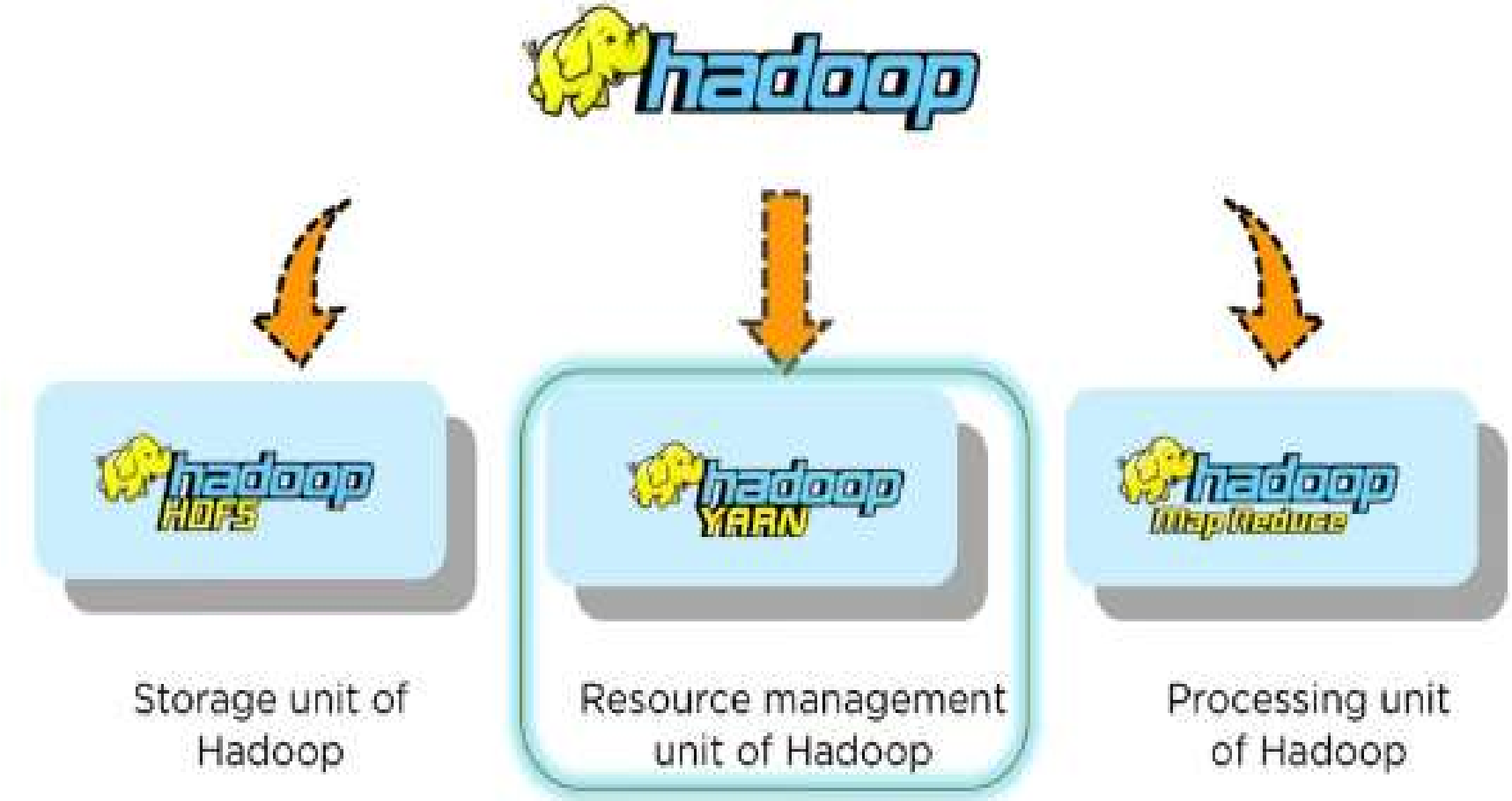
Result


Browsing HDFS
localhost:50070/explorer.html#/
Search
/
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 Jan 03 16:34 0 0 B output
drwxr-xr-x edureka supergroup 0 B Oct 17 09:53 0 0 B
drwxrwxrwx root supergroup 0 B Dec 31 14:58 0 0 B tmp
Browsing HDFS
localhost:50070/explorer.html#/output
Search
Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities
Browse Directory
/output
Go!
Show 25 entries
Search:
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



Basic Hadoop-2 Architecture



YET

ANOTHER

RESOURCE

NEGOTIATOR

Coming Up……

That is all for the day

Thank you