Hadoop MapReduce

Unit II Syllabus

- Data Product, Building Data Products at Scale with Hadoop, Data Science Pipeline and Hadoop Ecosystem

  Ref: (Chapter 1 - Data Analytics with Hadoop By Benjamin Bengfort & Jenny Kim)

- Operating System for Big Data: Concepts, Hadoop Architecture, Working with Distributed file system, Working with Distributed Computation

  Ref: (Chapter 2 - Data Analytics with Hadoop By Benjamin Bengfort & Jenny Kim)

- Framework for Python and Hadoop Streaming, Hadoop Streaming, MapReduce with Python, Advanced MapReduce.

  Ref: (Chapter 3 - Data Analytics with Hadoop By Benjamin Bengfort & Jenny Kim)

- In-Memory Computing with Spark, Spark Basics, Interactive Spark with PySpark, Writing Spark Applications

  Ref: (Chapter 4 - Data Analytics with Hadoop By Benjamin Bengfort & Jenny Kim)

Today's Topics

Chapter 2 Operating System for Big Data

- Concepts
- Hadoop Architecture
- Working with Distributed file system
- Working with Distributed Computation
- Hadoop Revision – Why Hadoop, Feature of Hadoop, Components of Hadoop :HDFS & YARN
- Example – HDFS – Splitting data, Replication of Data, Operations,
- HDFS – Read/Write Architecture

Ref: (Chapter 2, 3 - Data Analytics with Hadoop By Benjamin Bengfort & Jenny Kim)

Chapter 3 Framework for Python and Hadoop Streaming,

- Hadoop Streaming
- MapReduce with Python
- Advanced MapReduce.

Hadoop

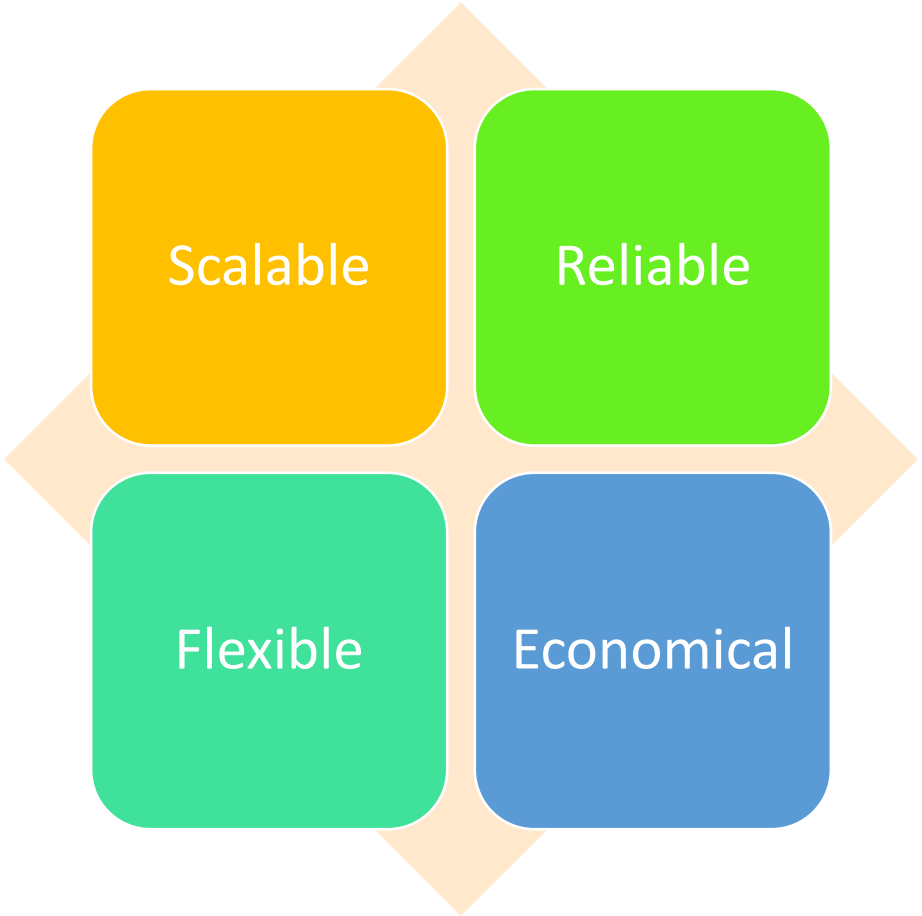
- **Hadoop** is an open-source software framework used for storing and processing **Big Data** in a distributed manner on large clusters of commodity hardware.
- Hadoop is licensed under the Apache v2 license.
- **Hadoop** was developed, based on the paper written by Google on the **MapReduce** system and it applies concepts of functional programming.
- Hadoop is written in the **Java programming language** and ranks among the highest-level Apache projects.
- Hadoop was developed by **Doug Cutting** and **Michael J. Cafarella.**

Why Hadoop?

Hadoop is required to

- Store Huge amount of data
- Store variety of data
- Process data faster by moving processing unit to data

Features of Hadoop



Components of Hadoop

HDFS

- Hadoop Distributed File System – Scalable Storage Unit

YARN

- Yet Another Resource Network – Processes Data

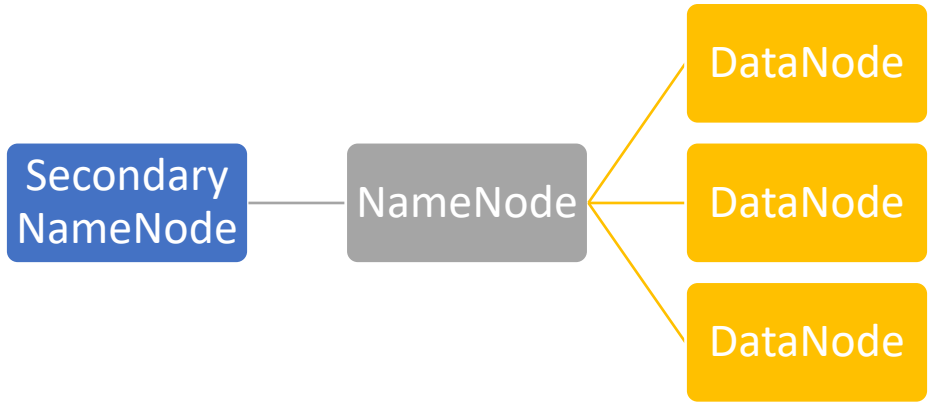
Goals of HDFS

- **Handling the hardware failure -** The HDFS contains multiple server machines. Anyhow, if any machine fails, the HDFS goal is to recover it quickly.

- **Streaming data access -** The HDFS applications usually run on the general-purpose file system. This application requires streaming access to their data sets.

- **Coherence Model -** The application that runs on HDFS require to follow the write-once-ready-many approach.
  - So, a file once created need not to be changed. However, it can be appended and truncate.

HDFS

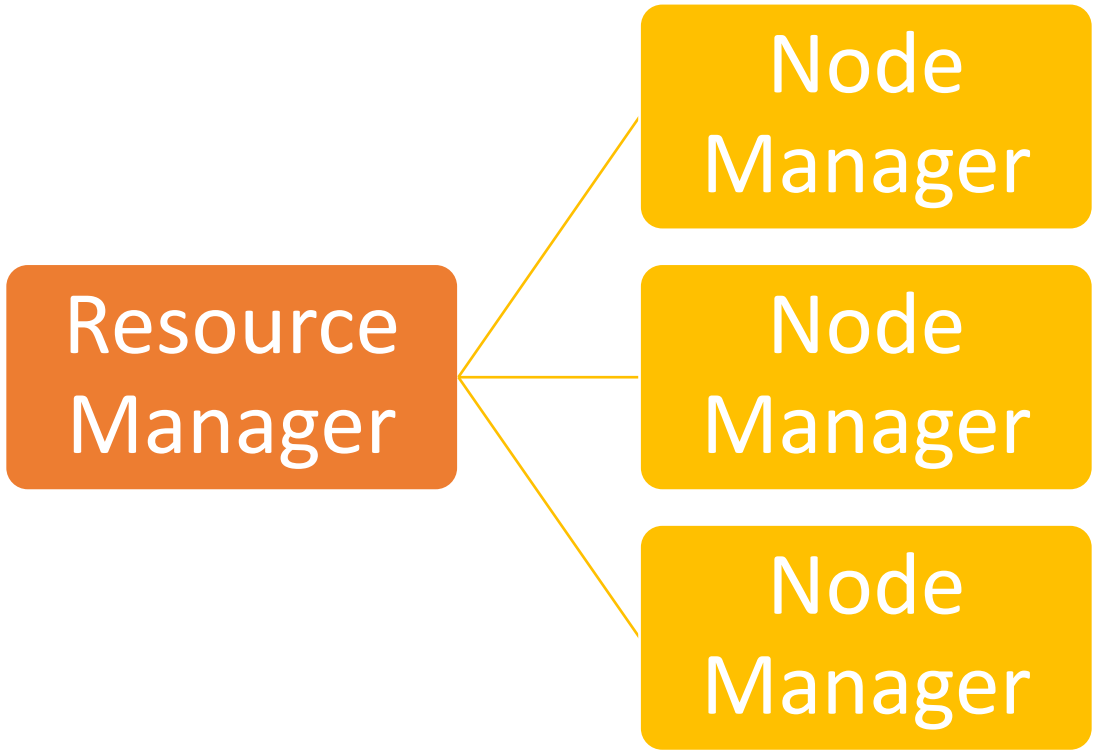


HDFS –Features

- Highly Scalable
- Cost Effective
- Reliability & Fault Tolerance
- Data Integrity /Replication
- High throughput
- Data Locality

- NameNode
  - Master Deamon
  - Maintains & Manages DataNodes
  - Records Metadata
  - Receives heartbeat and block report from all the DataNodes
- DataNode
  - Slave deamons
  - Stores actual data
  - Serves read & write request

YARN

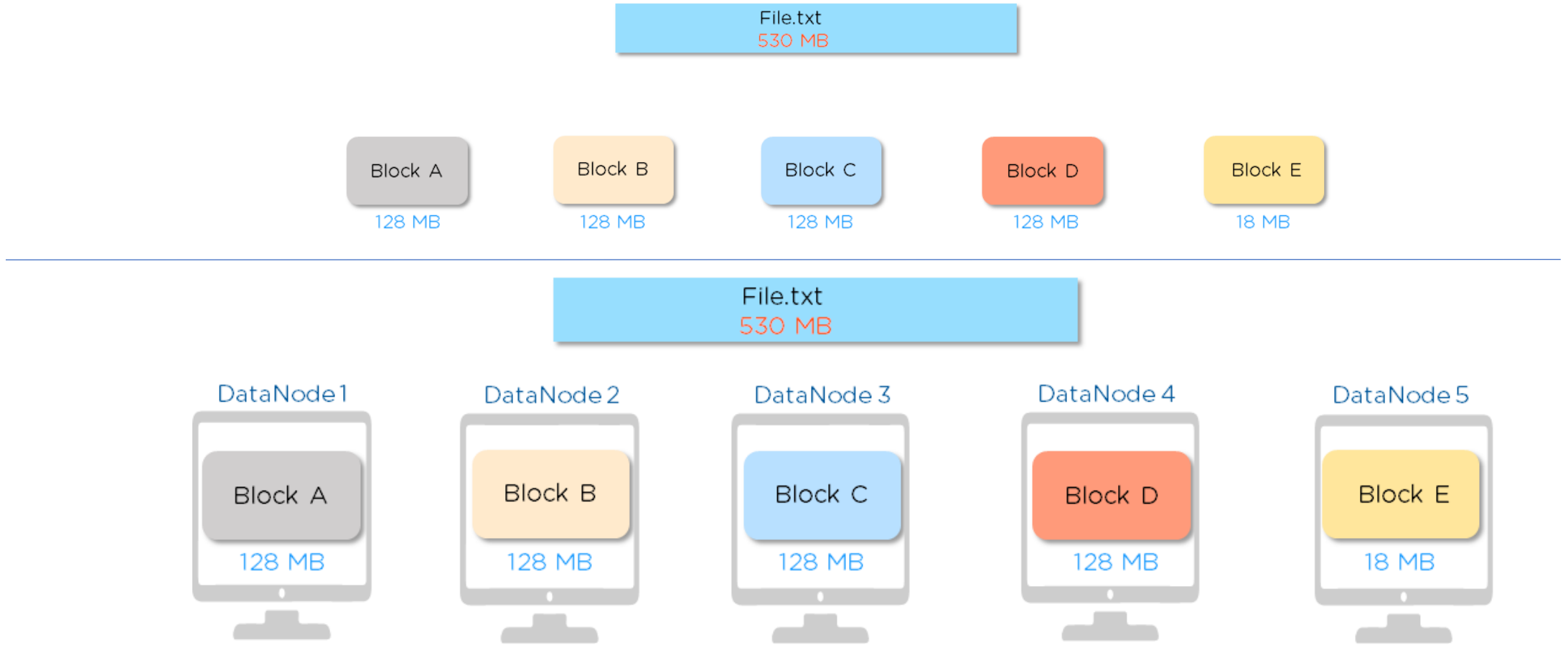


- Resource Manager
  - Receives the processing request
  - Passes the parts of request to corresponding Node Managers
- Node Manager
  - Installed on every DataNode
  - Responsible for execution of task on every single DataNode
- Client
  - For submitting MapReduce jobs
- MapReduce application Master
  - Checks tasks running the MapReduce job

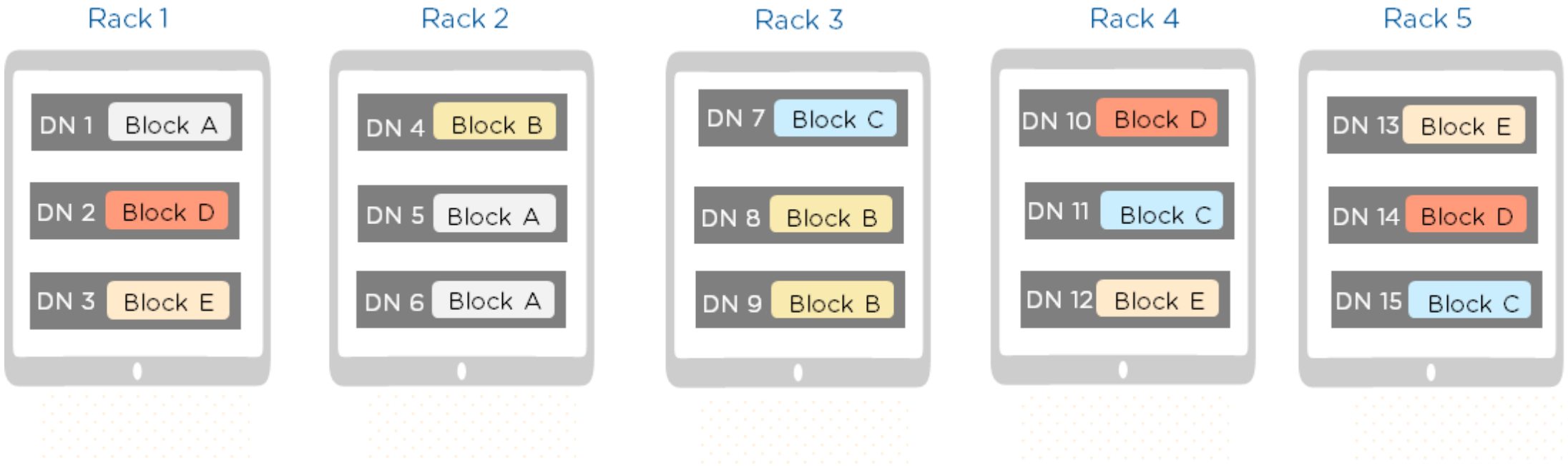
Advantages of YARN

- **Scalability**
  - Map Reduce 1 hits ascalability bottleneck at 4000 nodes and 40000 task, but Yarn is designed for 10,000 nodes and 1 lakh tasks
- **Utilization**
  - Node Manager manages a pool of resources, rather than a fixed number of the designated slots thus increasing the utilization
- **Multitenancy**
  - Different version of MapReduce can run on YARN, which makes the process of upgrading MapReduce more manageable

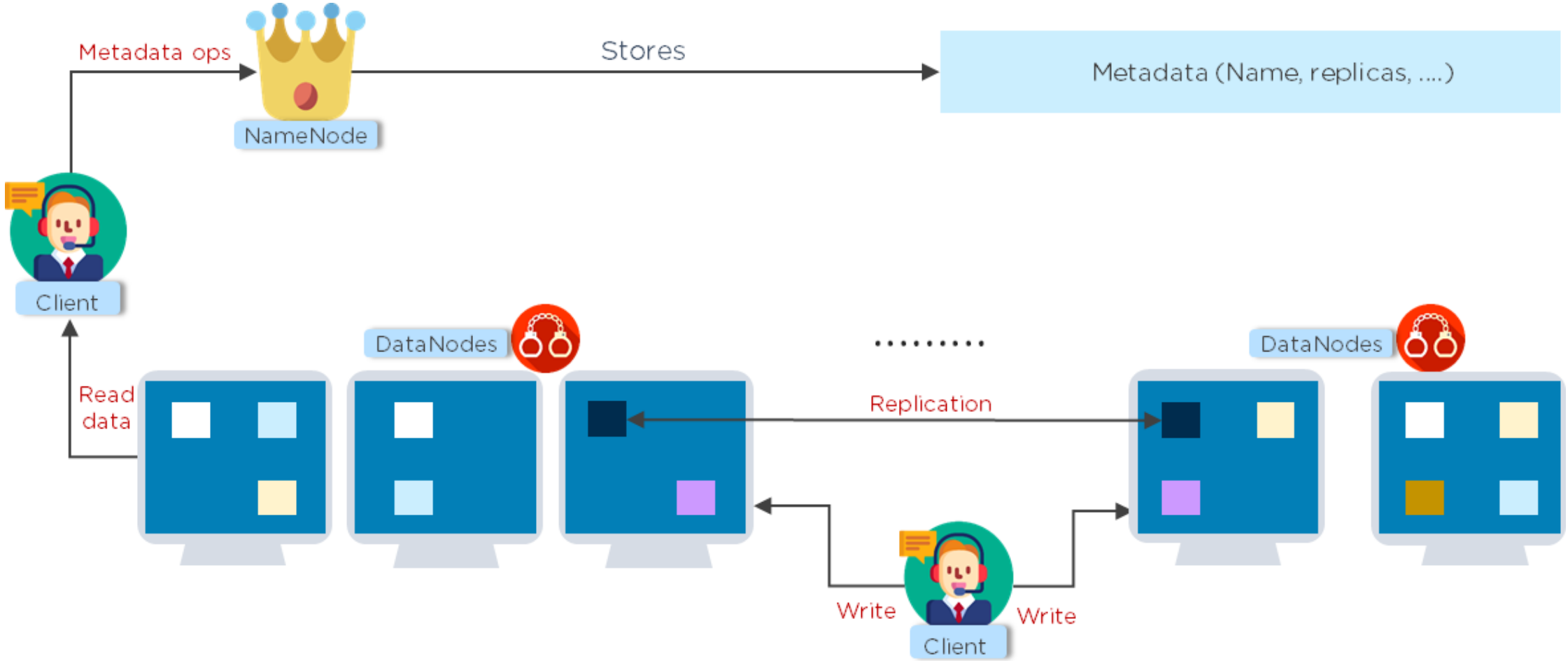
Example – HDFS – Splitting Data



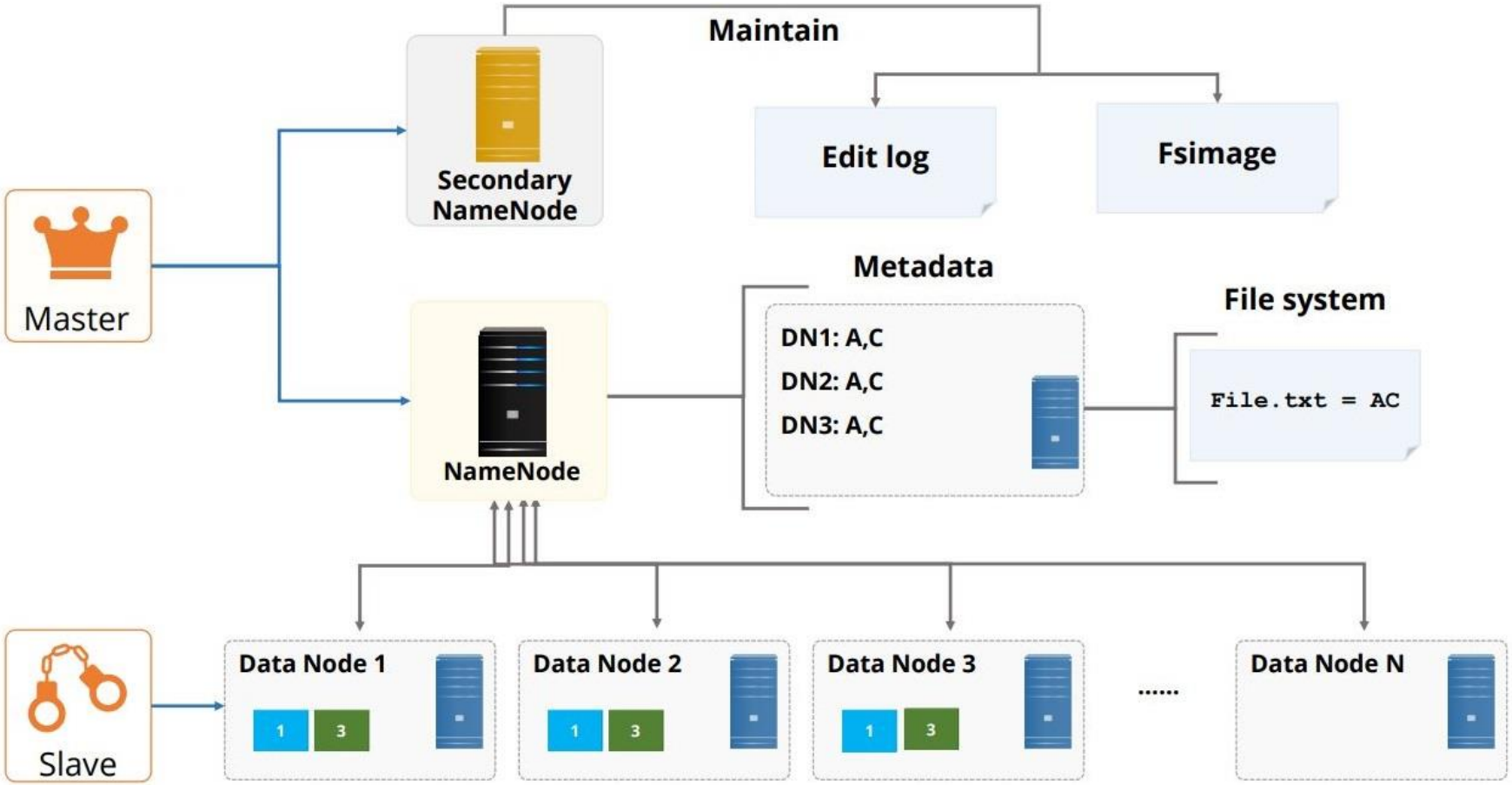
Example – HDFS - Replication of Data



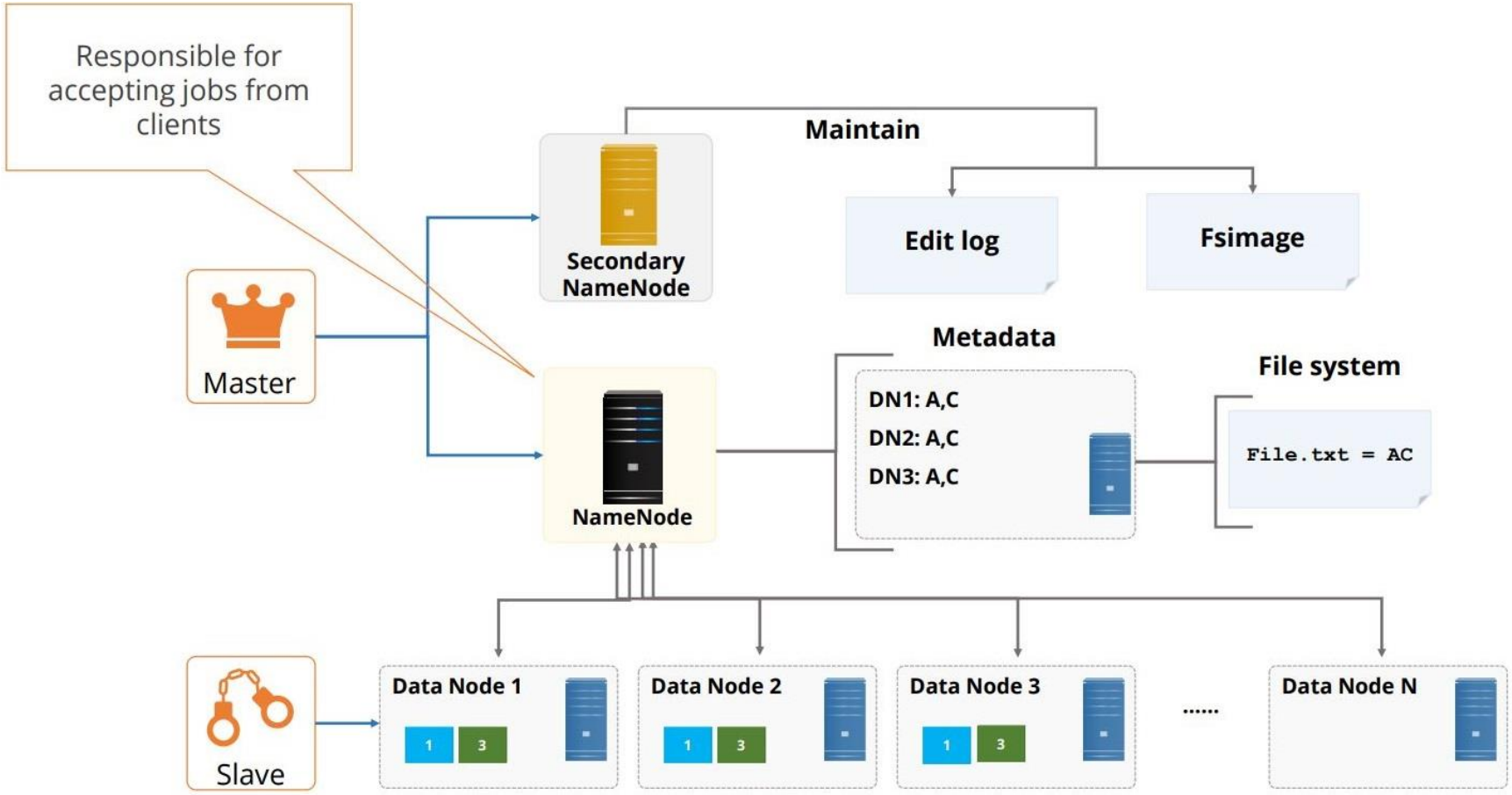
Example –HDFS – Operation



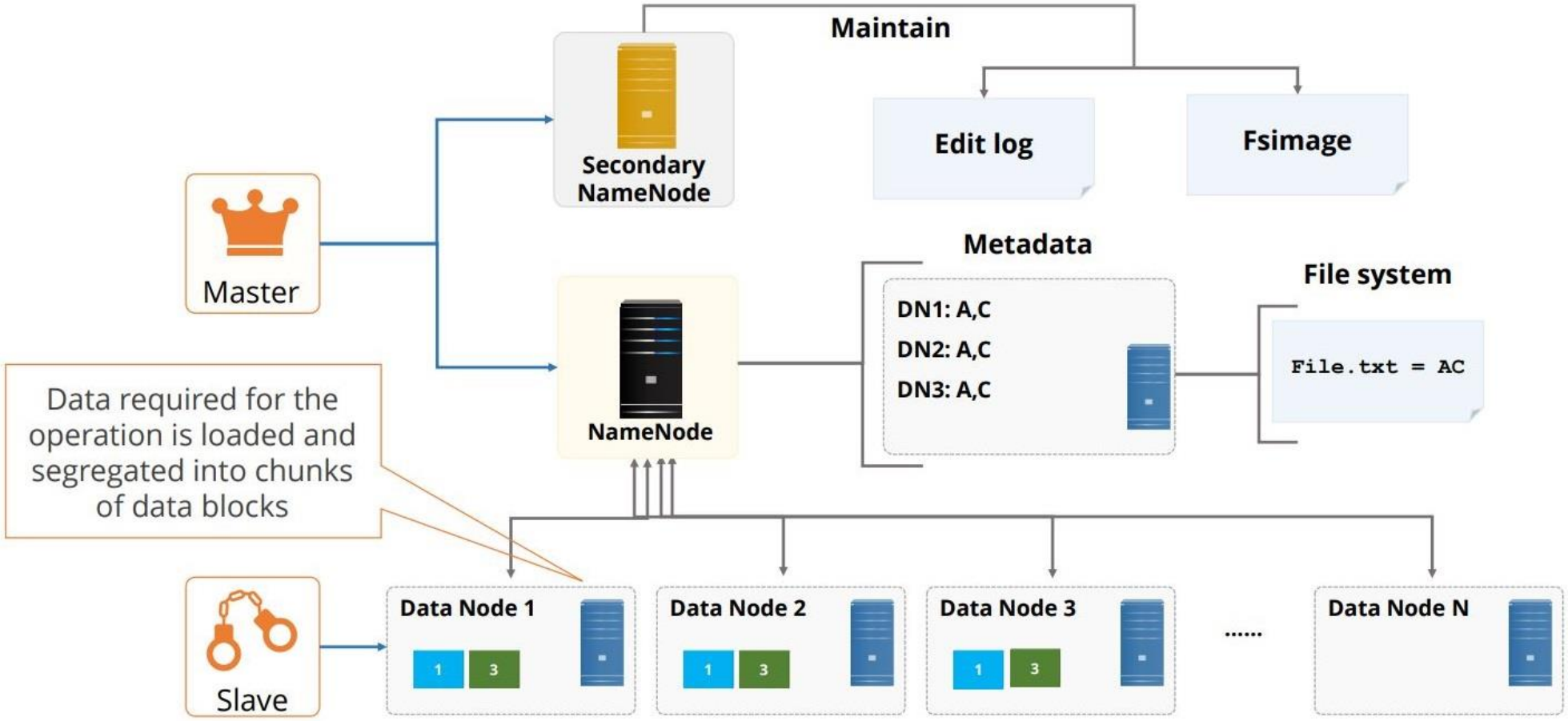
HDFS Architecture and Components



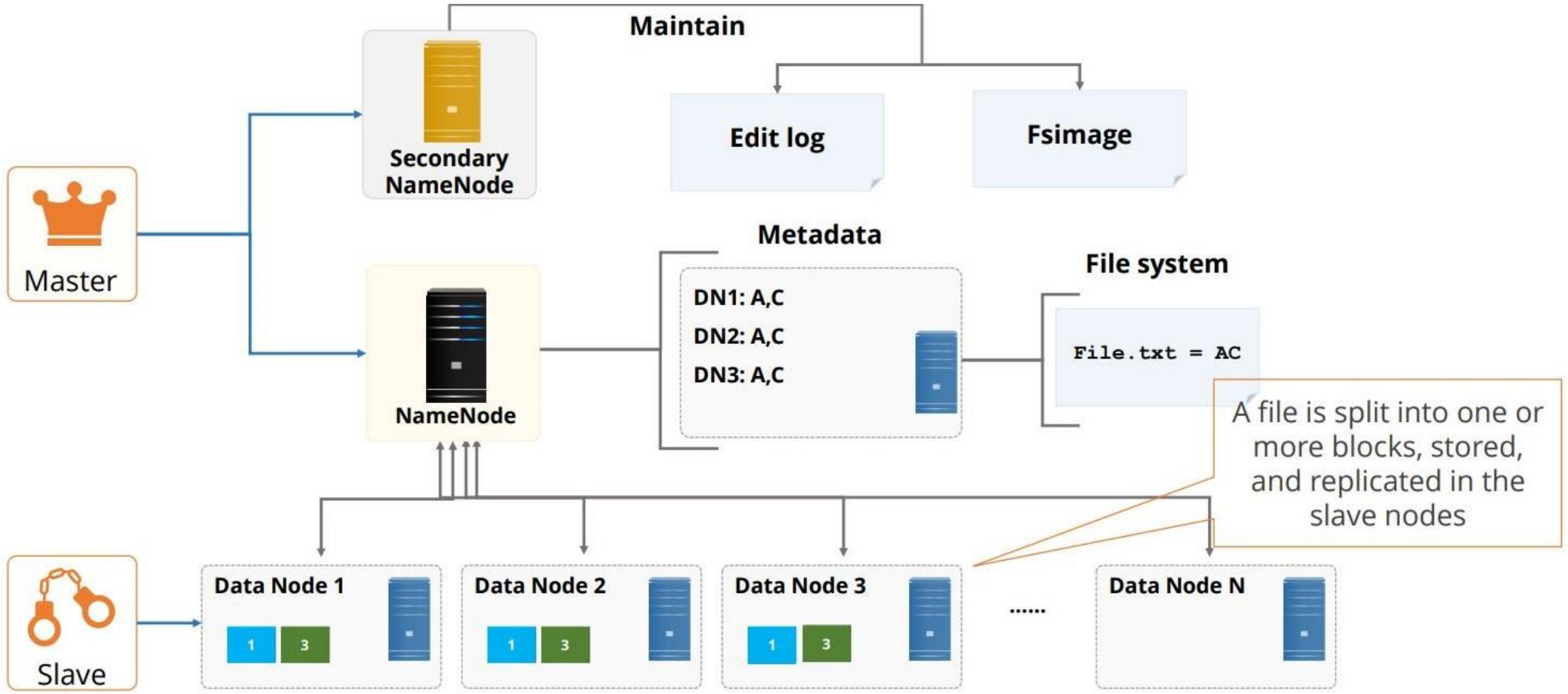
HDFS Architecture and Components



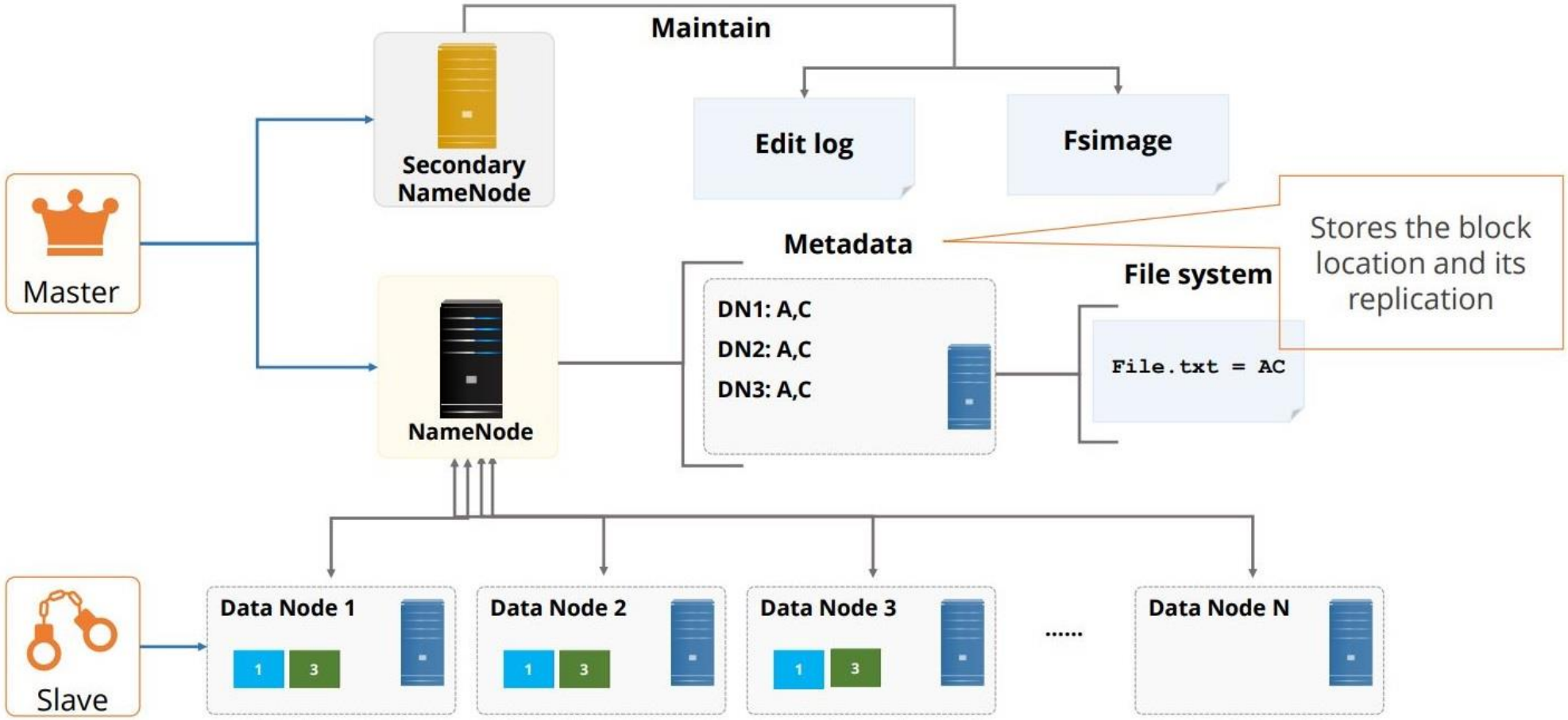
HDFS Architecture and Components



HDFS Architecture and Components

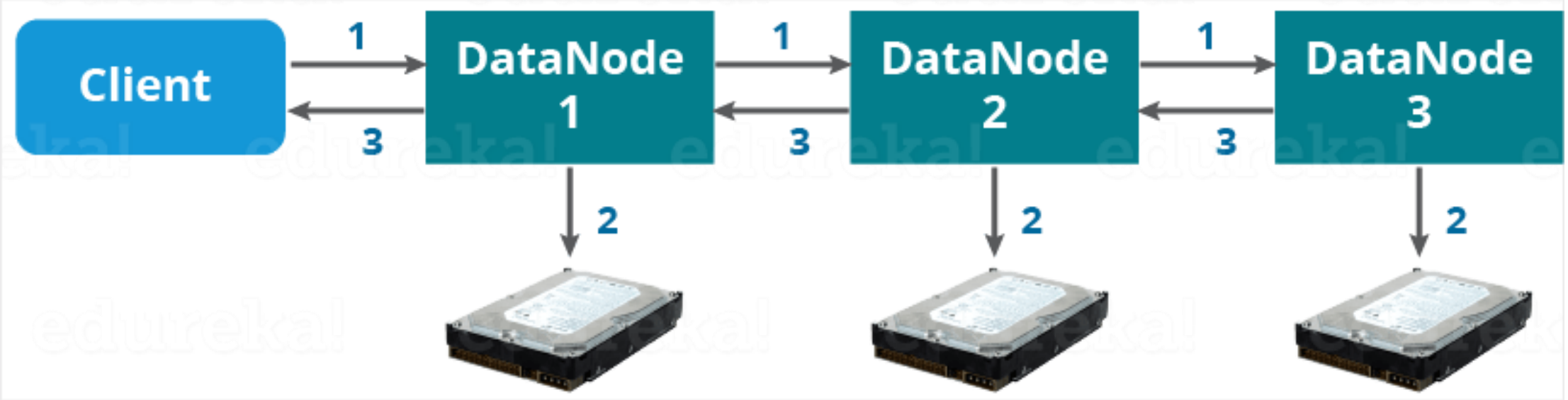


HDFS Architecture and Components



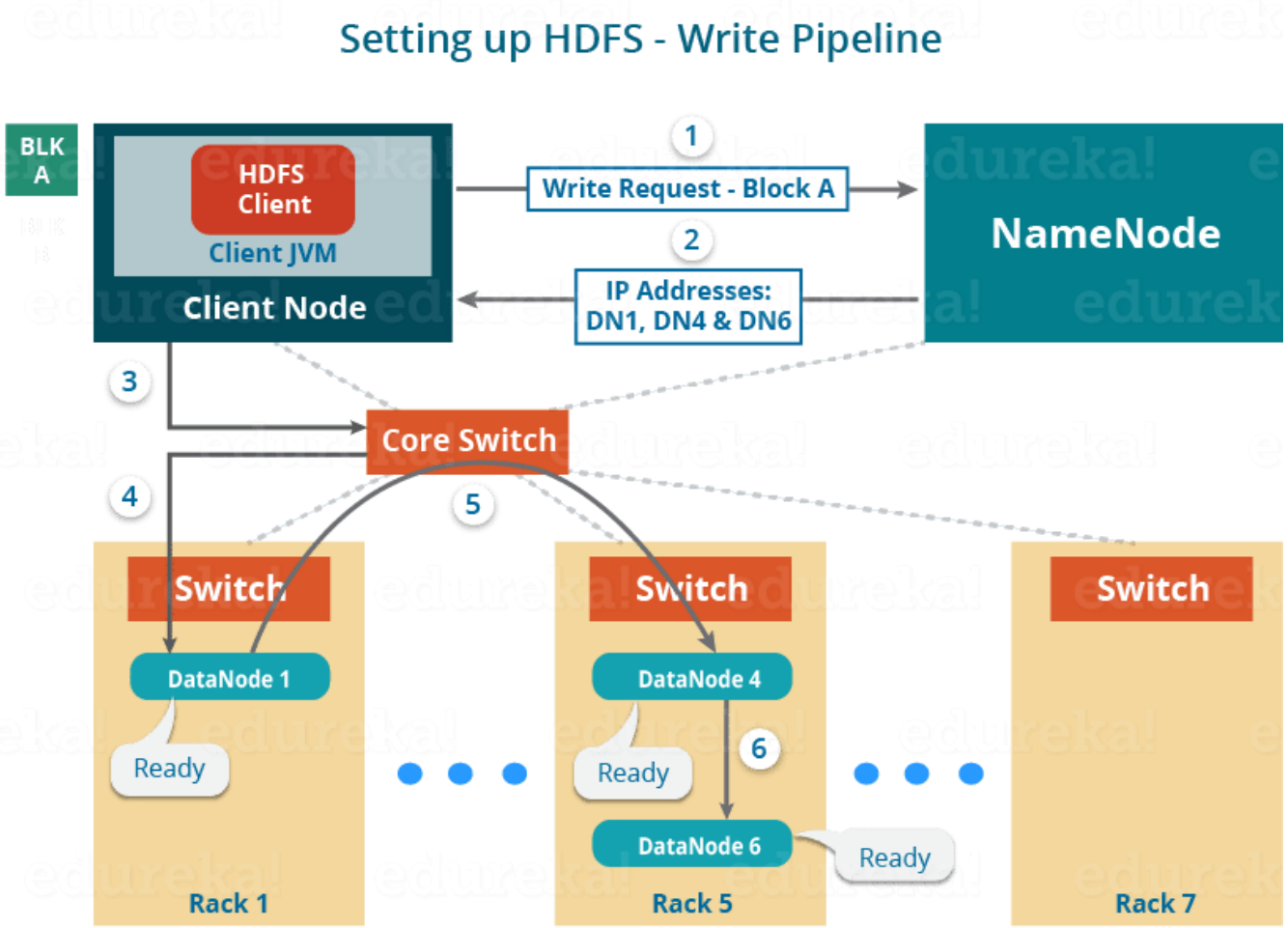
HDFS - Read/Write Architecture

HDFS - Read/Write Architecture

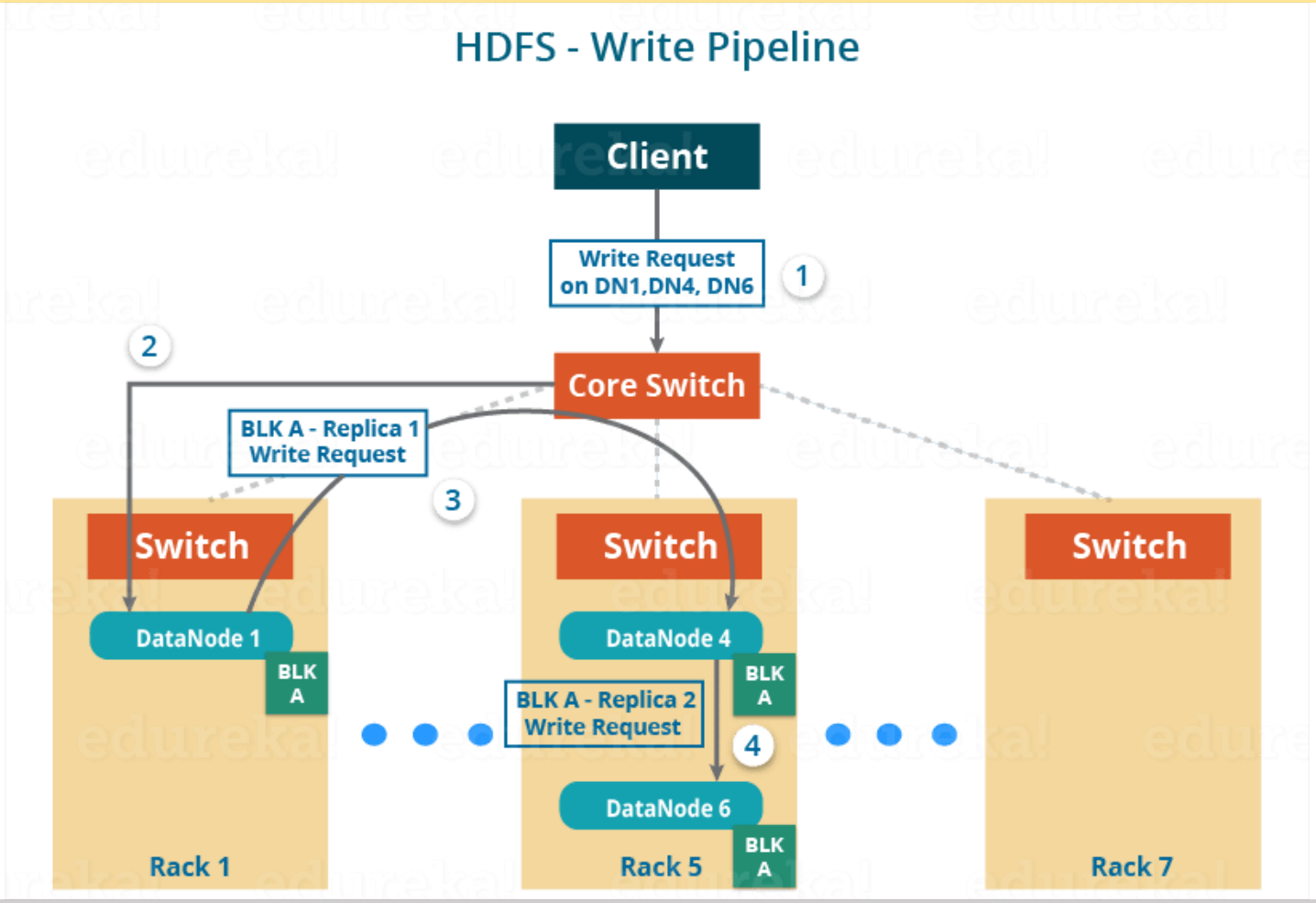


1. Set up Pipeline
2. Data Streaming & Replication
3. Shutdown of pipeline

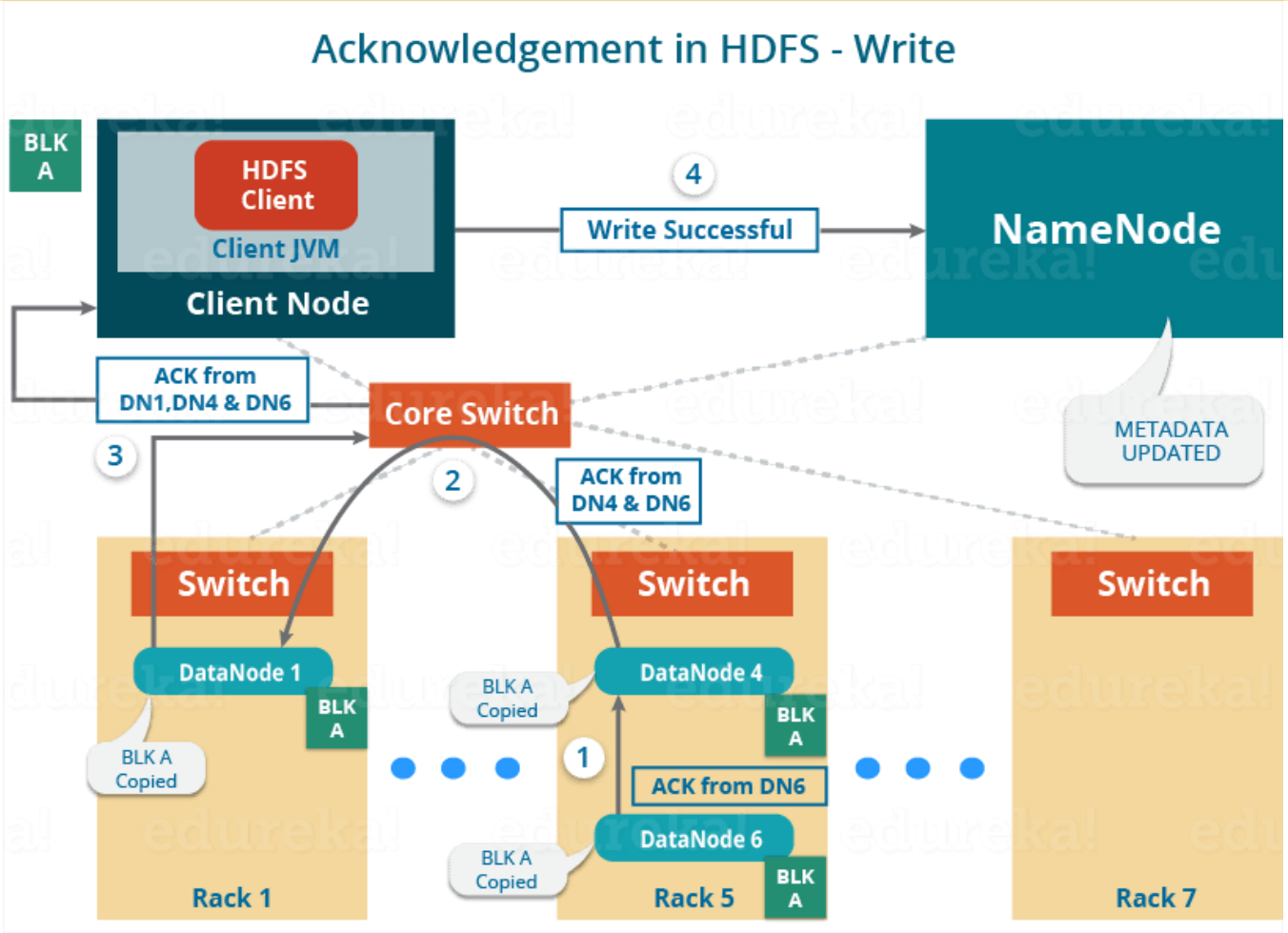
Set up Pipeline



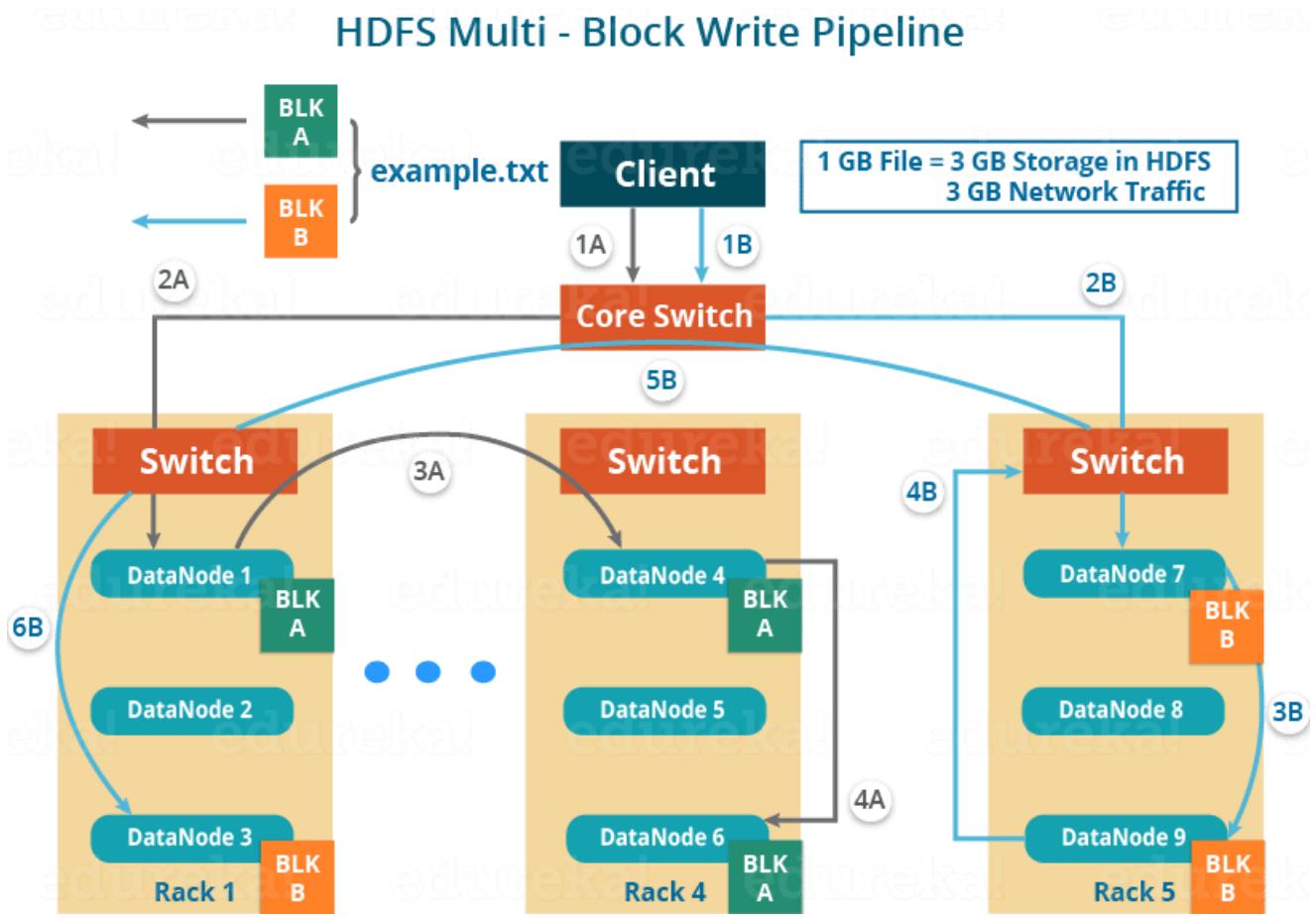
Data Streaming



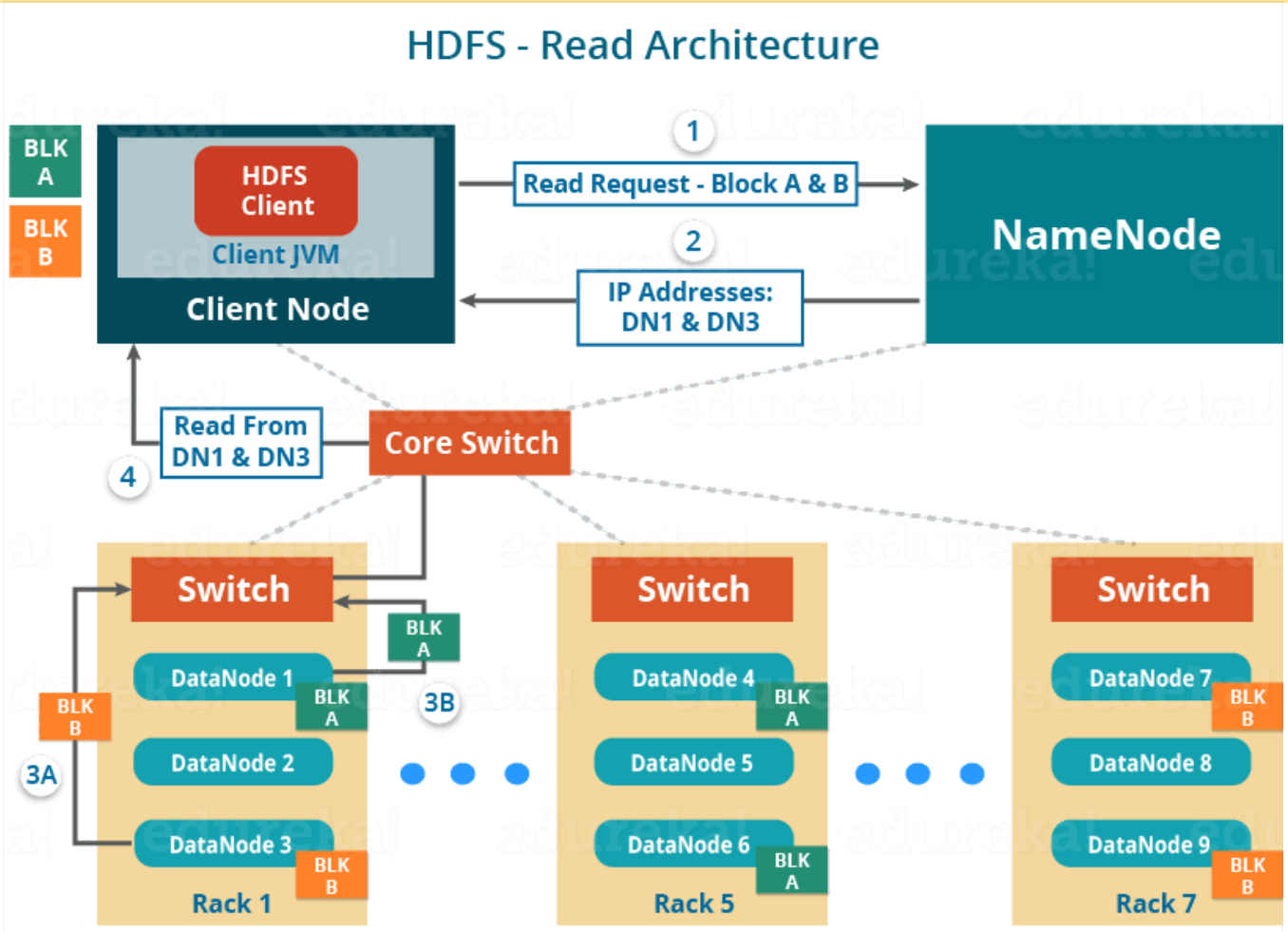
Acknowledgement in HDFS - Write



HDFS Multi – Block Write pipeline

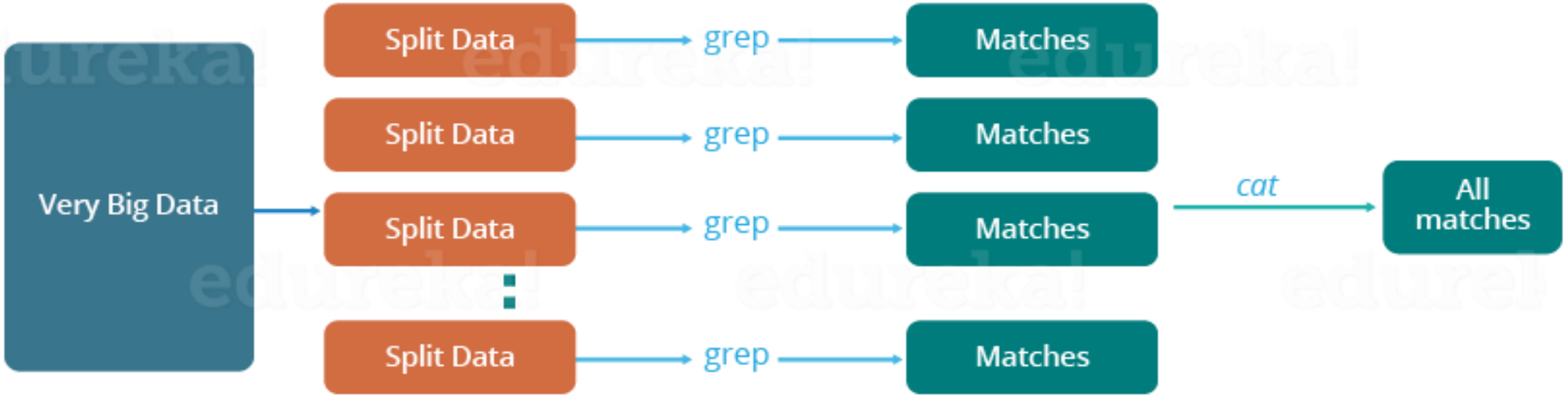


HDFS - Read Architecture



MapReduce Functional Programming Model

Parallel & Distributed Processing

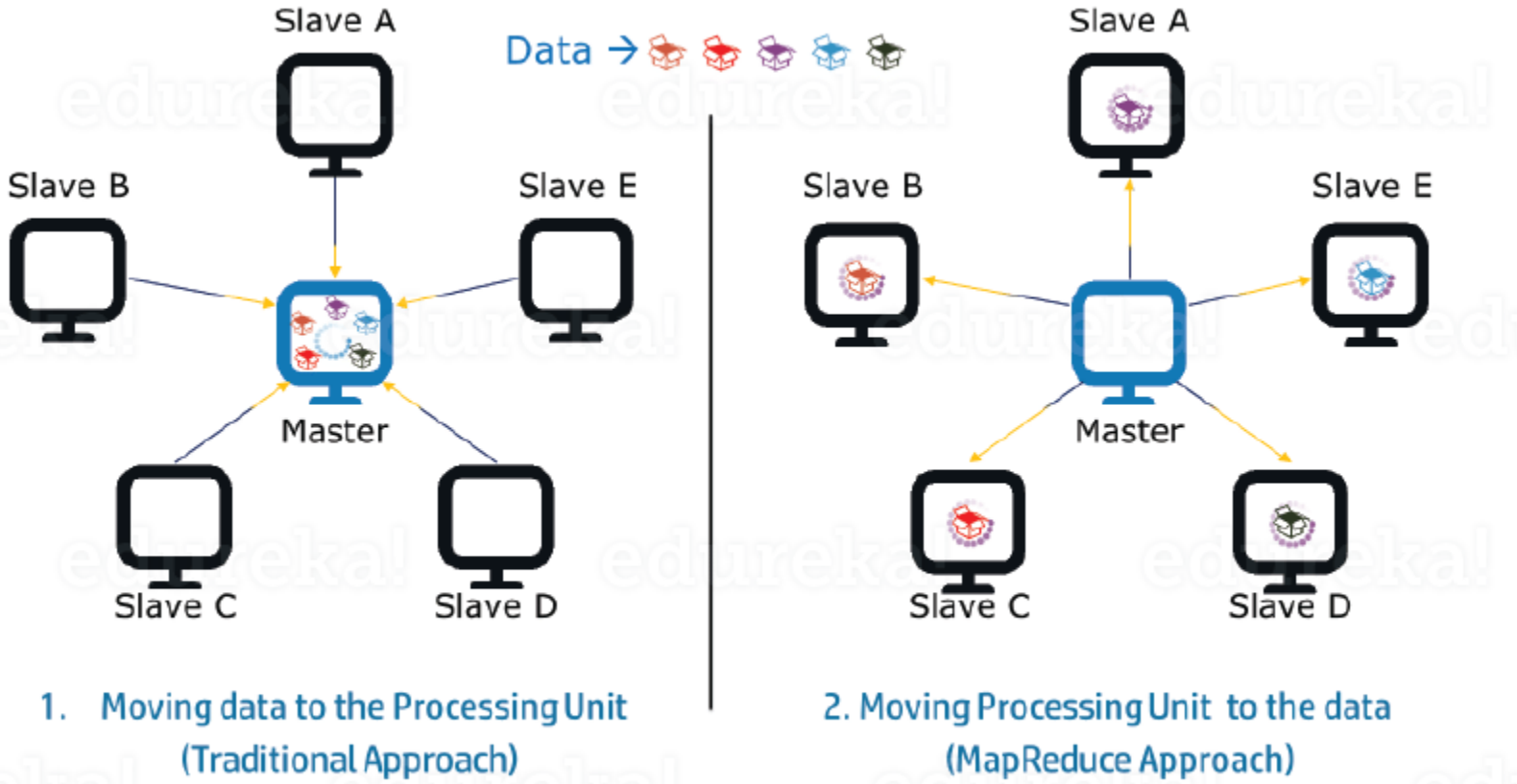


Challenges associated with this traditional approach –

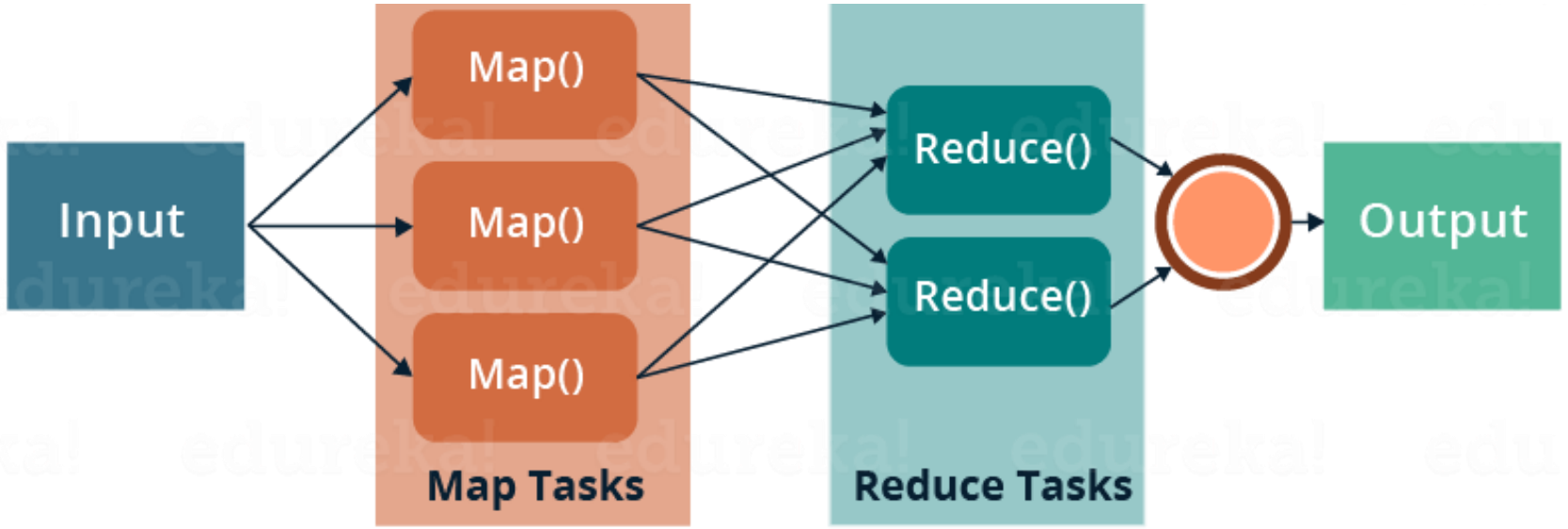
1. Critical Path Problem
2. Reliability Problem
3. Equal Split Issue
4. The Single Split Failure
5. Aggregation of Result

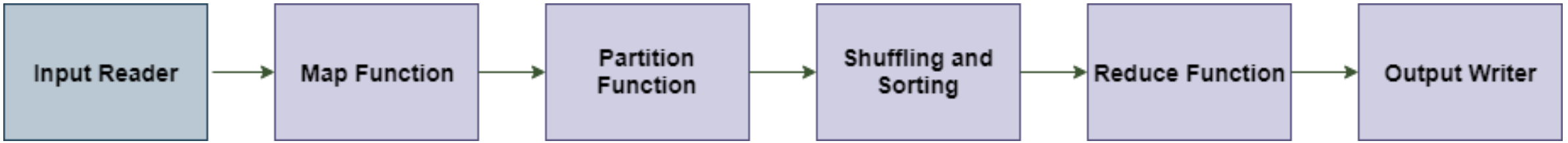
MapReduce

- Google released a paper on MapReduce technology in December 2004.
- This became the genesis of the Hadoop Processing Model.
- So, MapReduce is a programming model that allows us to perform parallel and distributed processing on huge data sets.
- **Advantages**
  - Parallel Processing
  - Data Locality



MapReduce


Input
Map()
Map()
Map()
Map Tasks
Reduce()
Reduce()
Reduce Tasks
Output


Input Reader
Map Function
Partition Function
Shuffling and Sorting
Reduce Function
Output Writer

MapReduce API

Mapper Class

RecordReader processes each Input record and generates the respective key-value pair.

Input Split

RecordReader

Reducer Class

Processes the data (Mapper Output) & Generates the final output

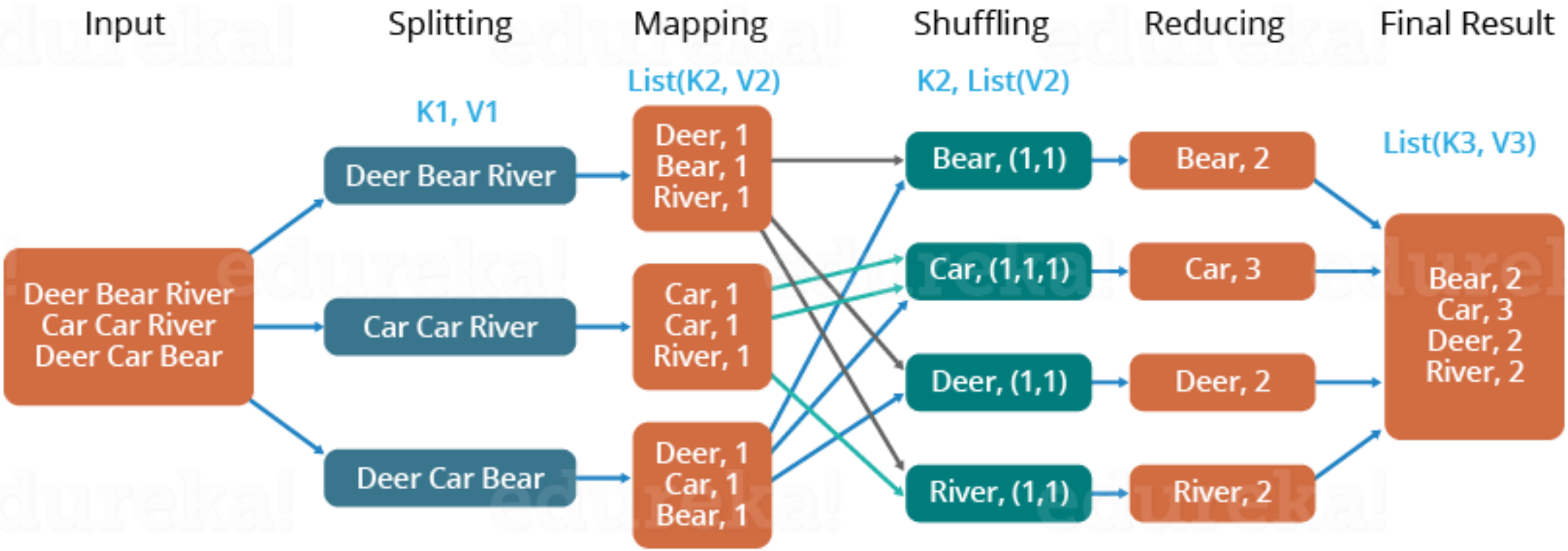
Reduces the set of intermediate values

Cleanup(), map(), run() & setup()

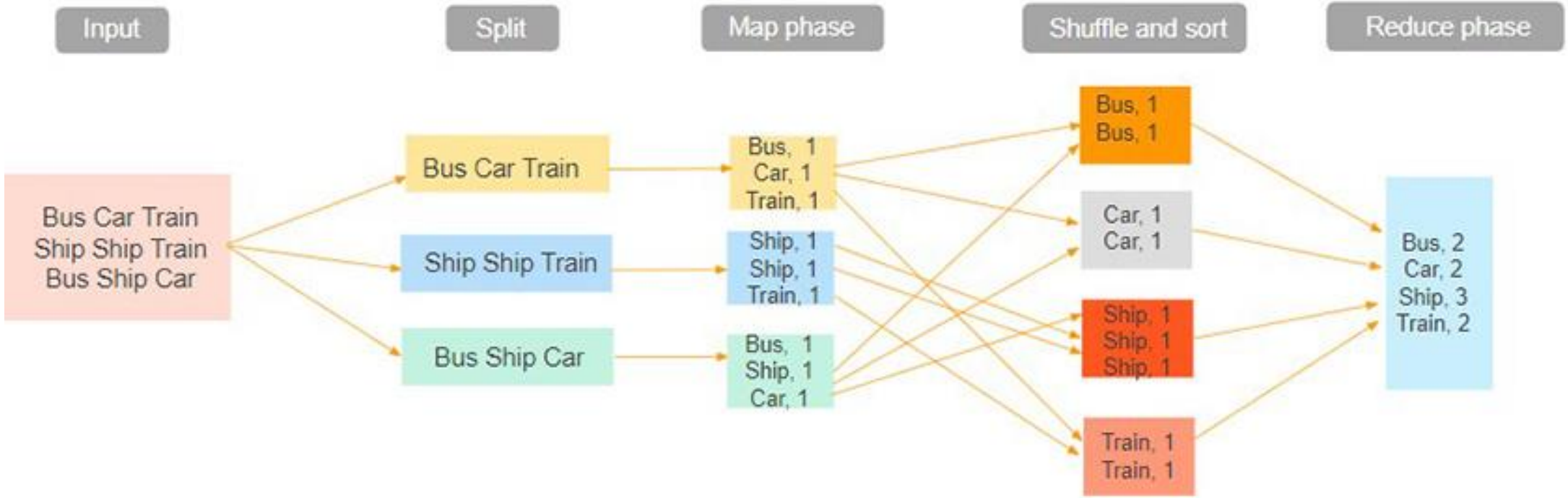
Driver Class

Responsible for setting up a MapReduce Job to run-in Hadoop

MapReduce Example 1 – Word Count



MapReduce Example 2 – Word Count



MapReduce – A Functional Programming Model

```
def map(key, value):
    # Perform processing
    return (intermed_key, intermed_value)

def reduce(intermed_key, values):
    # Perform processing
    return (key, output)
```

MapReduce – A Functional Programming Model

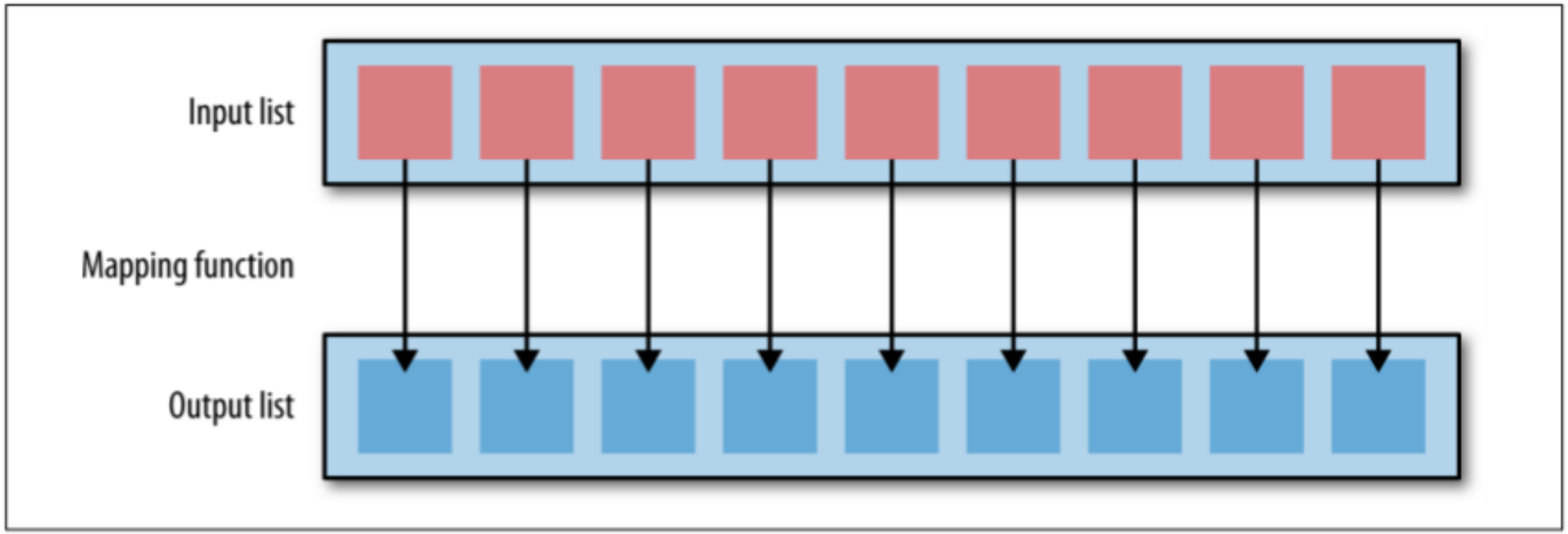


Figure 2-3. A map function takes as input a list of key/value pairs and operates singly upon each individual element in the list, outputting zero or more key/value pairs

MapReduce – A Functional Programming Model

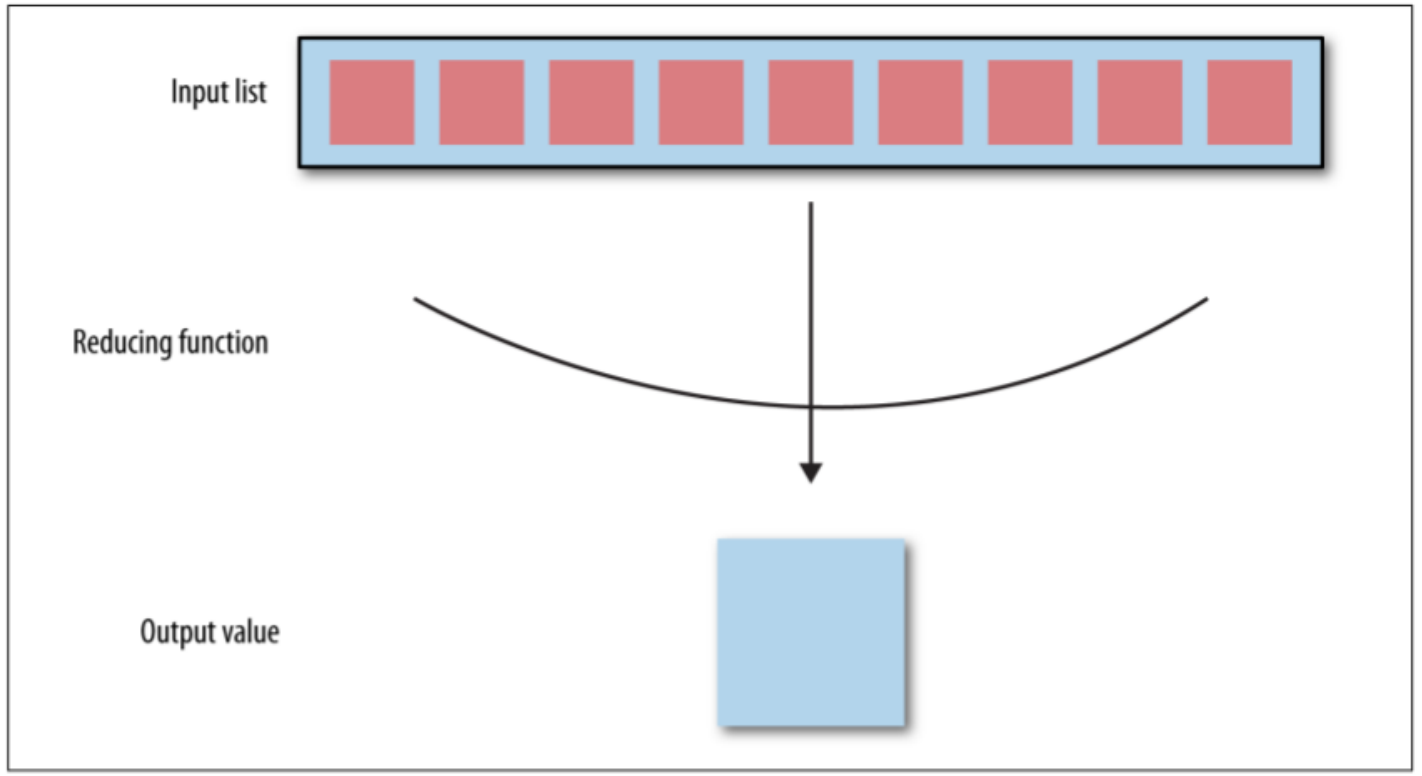


Figure 2-4. A reducer takes a key and a list of values as input, operates on the values list as a whole, usually through aggregation operations, and outputs zero or more key/value pairs

MapReduce – A Functional Programming Model

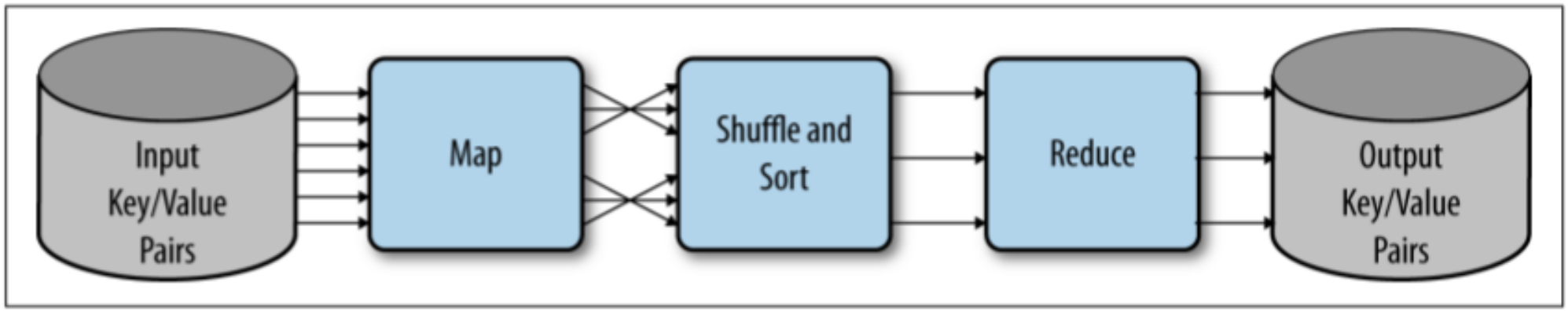


Figure 2-5. Broadly, MapReduce is implemented as a staged framework where a map phase is coordinated to a reduce phase via an intermediate shuffle and sort

MapReduce – A Functional Programming Model

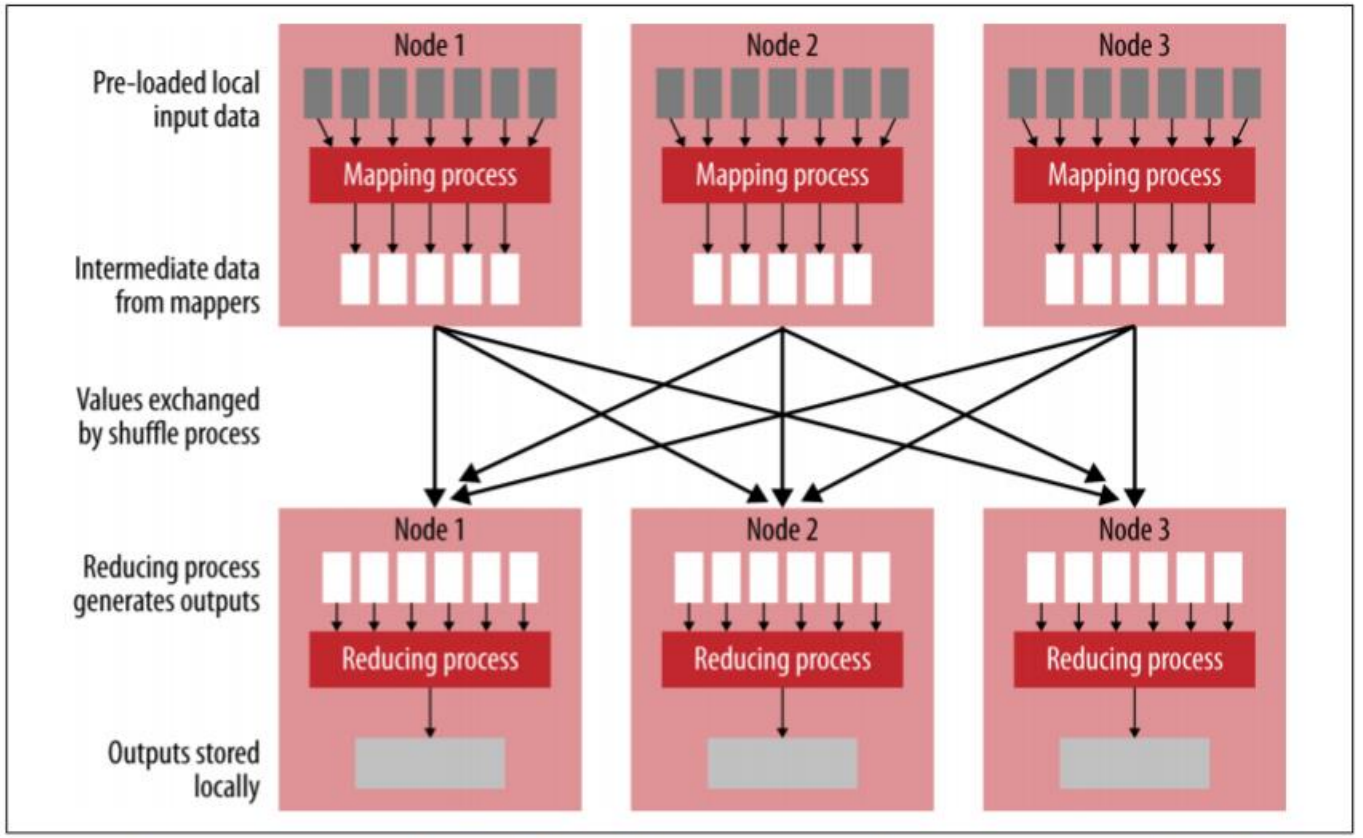


Figure 2-6. Data flow of a MapReduce job being executed on a cluster of a few nodes

MapReduce – A Functional Programming Model

```
# Input to WordCount mappers

(27183, "The fast cat wears no hat.")
(31416, "The cat in the hat ran fast.")

# Mapper 1 output

("The", 1), ("fast", 1), ("cat", 1), ("wears", 1),
("no", 1), ("hat", 1),(".", 1)

# Mapper 2 output

("The", 1), ("cat", 1), ("in", 1),
("hat", 1), ("ran", 1),("fast", 1),
```

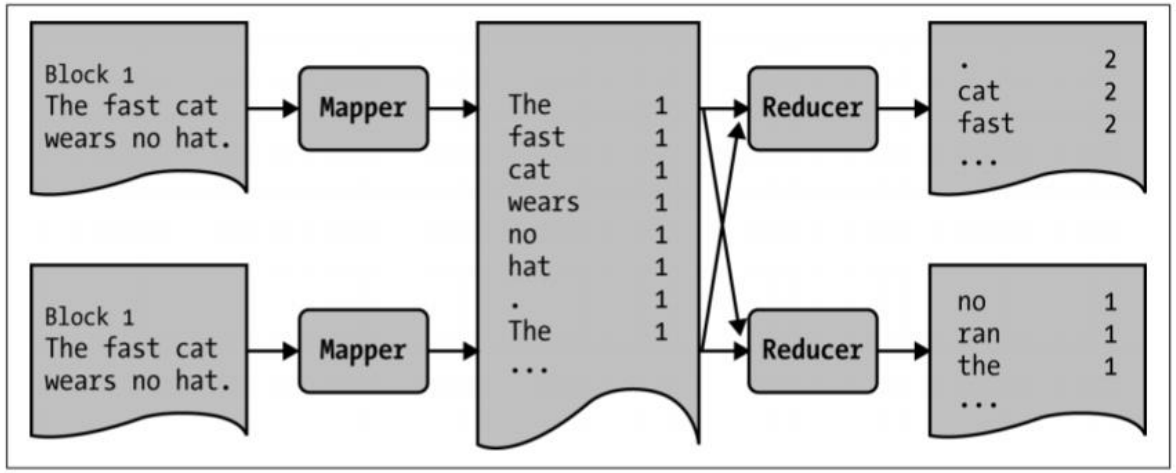


Figure 2-7. Data flow of the word count job being executed on a cluster with two mappers and two reducers

MapReduce – A Functional Programming Model

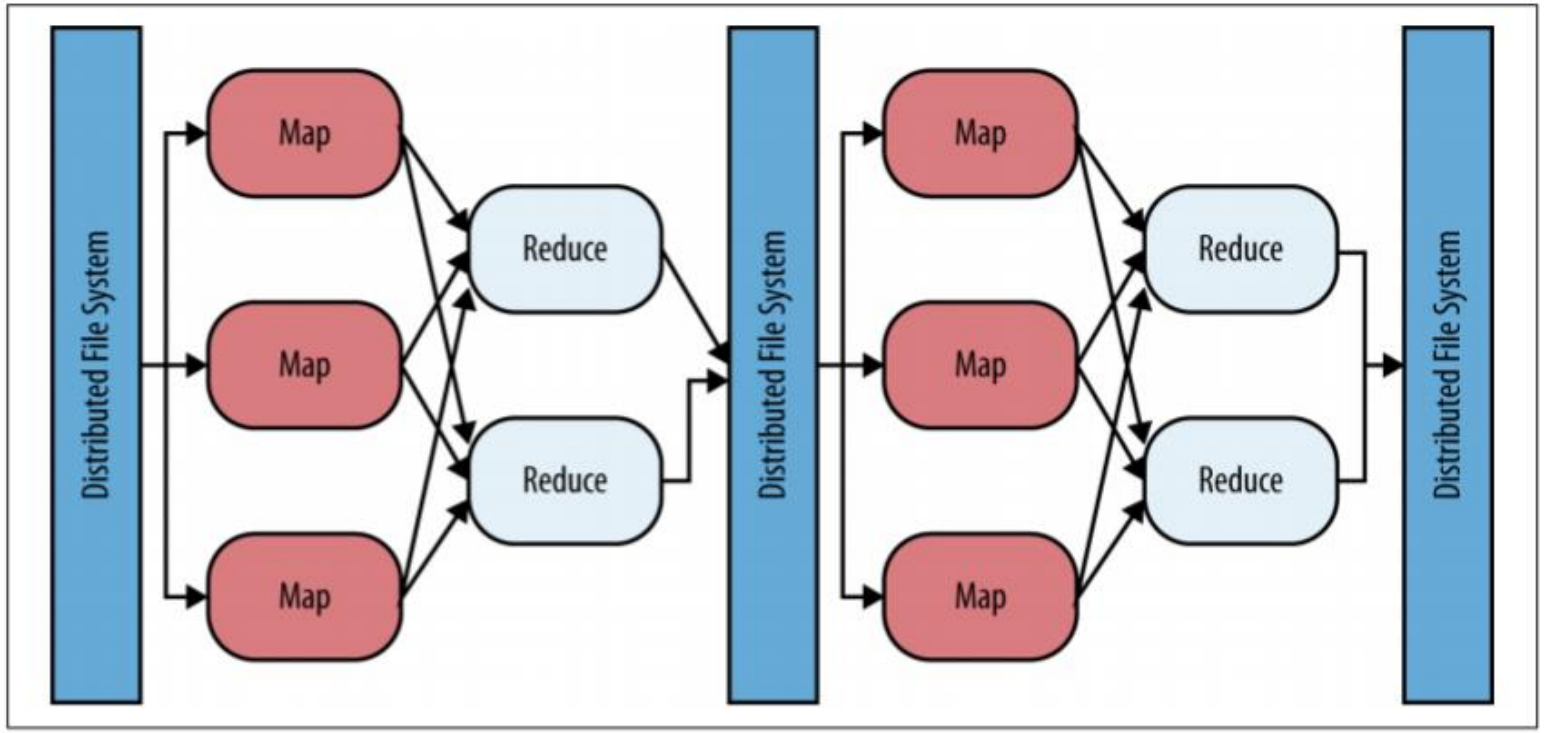


Figure 2-8. Complex algorithms or applications are actually made up through the chaining of MapReduce jobs where the input of a downstream MapReduce job is the output of a more recent one

Thank You….

Revise the topics from Syllabus References…

Fill Your Attendance Form….!

Syllabus References

1. Big Data and Analytics, Subhashini Chellappan Seema Acharya, Wiley

2. Data Analytics with Hadoop *An Introduction for Data Scientists*, Benjamin Bengfort and Jenny Kim, O'Reilly

3. Big Data and Hadoop, V.K Jain, Khanna Publishing

   https://books.google.co.in/books?id=i6NODQAAQBAJ&pg=PA122&source=gbs_toc_r&cad=4#v=onepage&q&f=true