



Bigdata Fundamentals

Agenda

- ▶ Need of Bigdata Architecture?
- ▶ History of Bigdata
- ▶ Architecture of Bigdata
- ▶ Bigdata Storage module
- ▶ Bigdata Processing Module
- ▶ Hands-On labs
- ▶ Bigdata Storage Module
- ▶ Bigdata Processing Module
- ▶ Spark, Hive

Need of Bigdata Architecture?

- ▶ Data is too large and Complex to Compute

3 V Approach in Bigdata

- ▶ Volume
- ▶ Variety
- ▶ Velocity

Volume

- ▶ Compare the ratio of the data consumption from 2009 to 2020?
- ▶ According to International data corporation 44x times it got increased
- ▶ From 0.8 Zettabytes to 35 Zettabytes
- ▶ By 2022 60 Zettabytes
- ▶ By 2022 there will be 5200 gb of data that every person will consume

Volume

- ▶ 12 + tb twitter
- ▶ 25+ tb Facebook
- ▶ 30 billion RFID tags
- ▶ 4.6 billion camera phone
- ▶ 100s of millions gps enabled
- ▶ 2+ billion active users available by end 2011

Variety

- ▶ CSV
- ▶ TSV
- ▶ JSON – linkedin, facebook
- ▶ AVRO - bigdata
- ▶ ORC
- ▶ Parquet
- ▶ XML – linkedin, facebook

Velocity

- ▶ Because of data its been growing exponentially, our storage and processing medium that should need to support
- ▶ Speed of data generation

Veracity

- ▶ Break up to 11.20

Hadoop Components

- ▶ Storage Module -> HDFS(Hadoop Distributed File System) storage concept that will run on multiple computers
 - ▶ EX: 2 TB data(4 systems with 500 GB hard drive)(Logically it will combine all the storage and treat as single logical entity)
 - ▶ Data – logical binaries (File system – Method to store and retrieve the data efficiently)
 - ▶ File system approach to store and retrieve the large volume of data in the distributed file system
- ▶ Processing Module -> Map Reduce

History

- ▶ OCT, 2003 -> Google File System Published(Theory)
- ▶ Dec, 2004 -> Jeffrey Dean & Sanjay Ghemawat from Google, published another paper MapReduce: Simplified Data Processing on Large Cluster(Theory)
- ▶ Jan, 2006 -> Inspired by above papers that google published, Doug Cutting, a yahoo employee, developed an opensource implementation of MapReduce Framework
- ▶ Apr, 2006 -> Hadoop 0.1.0 Released
- ▶ May, 2006 -> Yahoo deploys 300 Machine Hadoop Cluster
- ▶ Apr, 2007 -> Yahoo Deploys 2 Cluster of 1000 Machines
- ▶ Jul, 2008 -> Hadoop wins Terabyte sort Benchmark

History

- ▶ Jun, 2010, Yahoo 4000 Nodes/ 70 Petabytes
- ▶ Dec, 2011 -> Apache Hadoop Release 1.0.0 available
- ▶ Oct 2013 -> Apache Hadoop Release 2.2.0 (Yarn)
- ▶ Dec 2015 Apache Hadoop 2.6.3 Released

Hadoop was named after Doug Cuttings son toy elephant

DFS – Distributed File System

Read 1TB Data



1 Machine
4 I/O Channels
Each channel - 100MB/s



43 Minutes



10 Machines
4 I/O Channels
Each channel - 100MB/s



4.3 Minutes

Hadoop Components

2 Main Hadoop Components

Storage

Processing



Installation

Java Commands

Hadoop Distributions

- ▶ 1.0 -> it is suitable only for Java Programmers
 - ▶ 2.0 -> API -> user friendliness
 - ▶ 3.0
-
- ▶ Some Software is working on group of computers at the particular time to execute on particular task
 - ▶ Master
 - ▶ Slave

Hadoop 1.0

- ▶ HDFS
 - ▶ Name Node(Master)
 - ▶ Data Node(Slave)
 - ▶ Secondary Name Node(Secondary Master)
- ▶ MapReduce
 - ▶ Job Tracker(Master)
 - ▶ Task Tracker (Slave)

Hadoop 2.0

- ▶ HDFS
 - ▶ Name Node(Master)
 - ▶ Data Node (Slave)
 - ▶ Secondary Name Node (Secondary Master)
- ▶ MapReduce
 - ▶ Resource Manager (Master)
 - ▶ Node Manager (Slave)

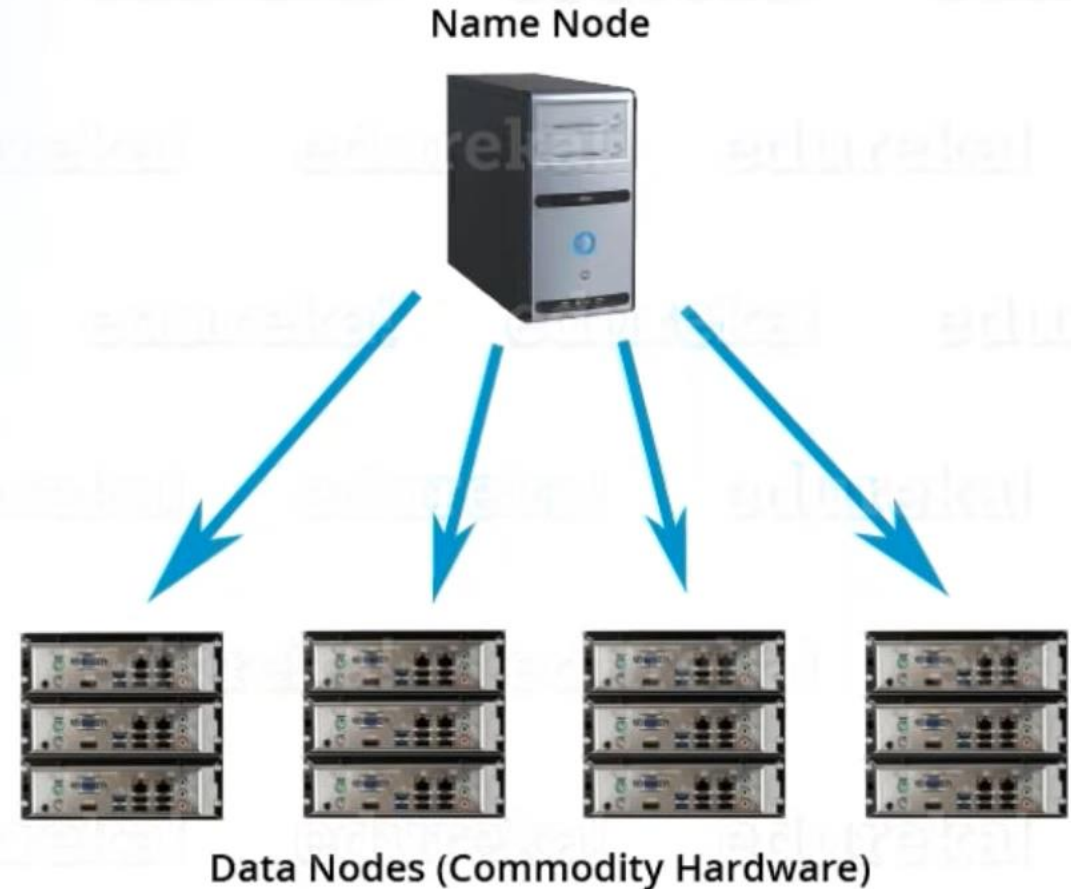
NameNode and DataNode


NameNode

- Master daemon
- Maintains and Manages DataNodes
- Records metadata 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



- 
- ▶ The basic storage unit of HDFS is Blocks
 - ▶ Block Size of HDFS 1.0 is 64MB
 - ▶ Block Size of HDFS 2.0 is 128MB
 - ▶ Ex: 500MB
 - ▶ 1.0 -> 8 Blocks
 - ▶ 2.0 -> 4 Blocks
 - ▶ 1024MB file
 - ▶ 1.0 -> 16
 - ▶ 2.0 -> 8

How HDFS will handle Read and Write Operation

- ▶ Up to 02.05 pm

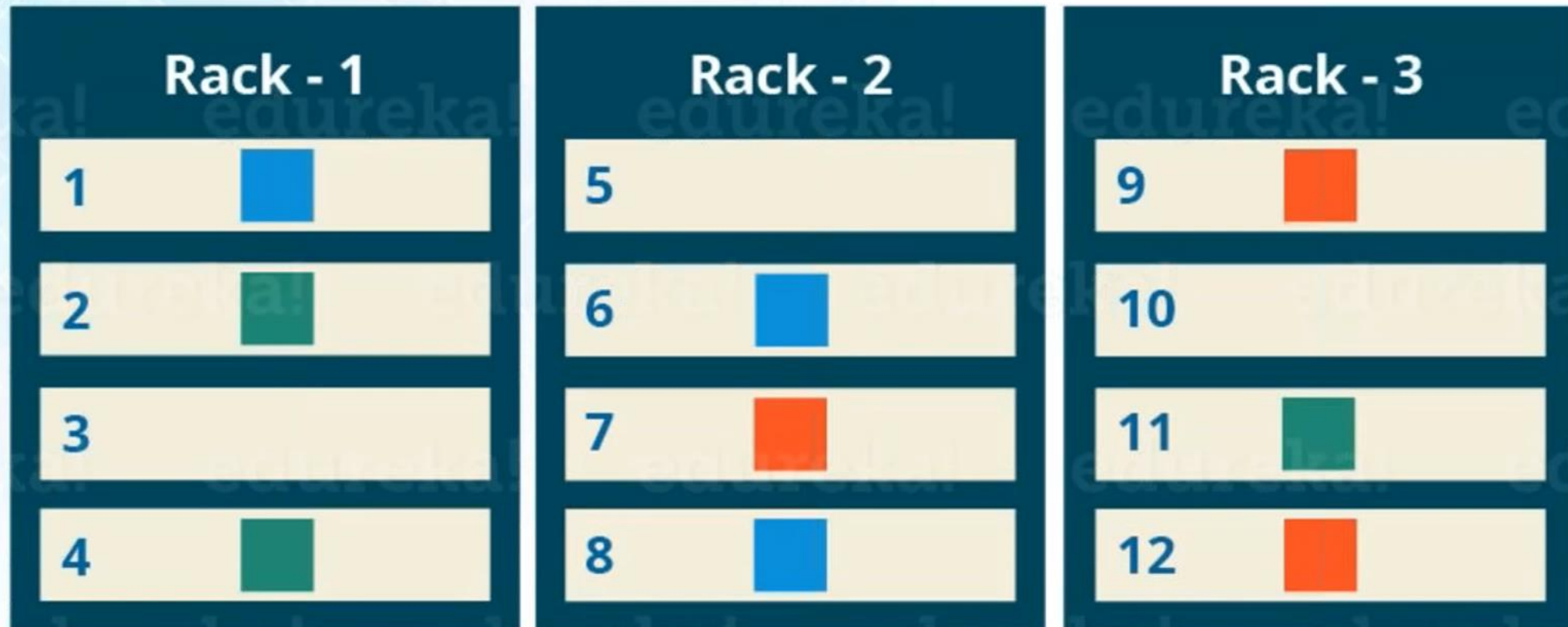
HDFS Replication Factor


- ▶ By default HDFS will maintain 1+2 Replication factor
- ▶ 2 PB, if I want to save it in Hadoop ecosystem means I need 6PB.
- ▶ In Hadoop ecosystem processing module is different and storage module is different?
- ▶ Storage module is costlier or processing module(ram + cpu)

Hadoop Architecture: Rack Awareness

Rack Awareness Algorithm

Block A :  Block B:  Block C: 



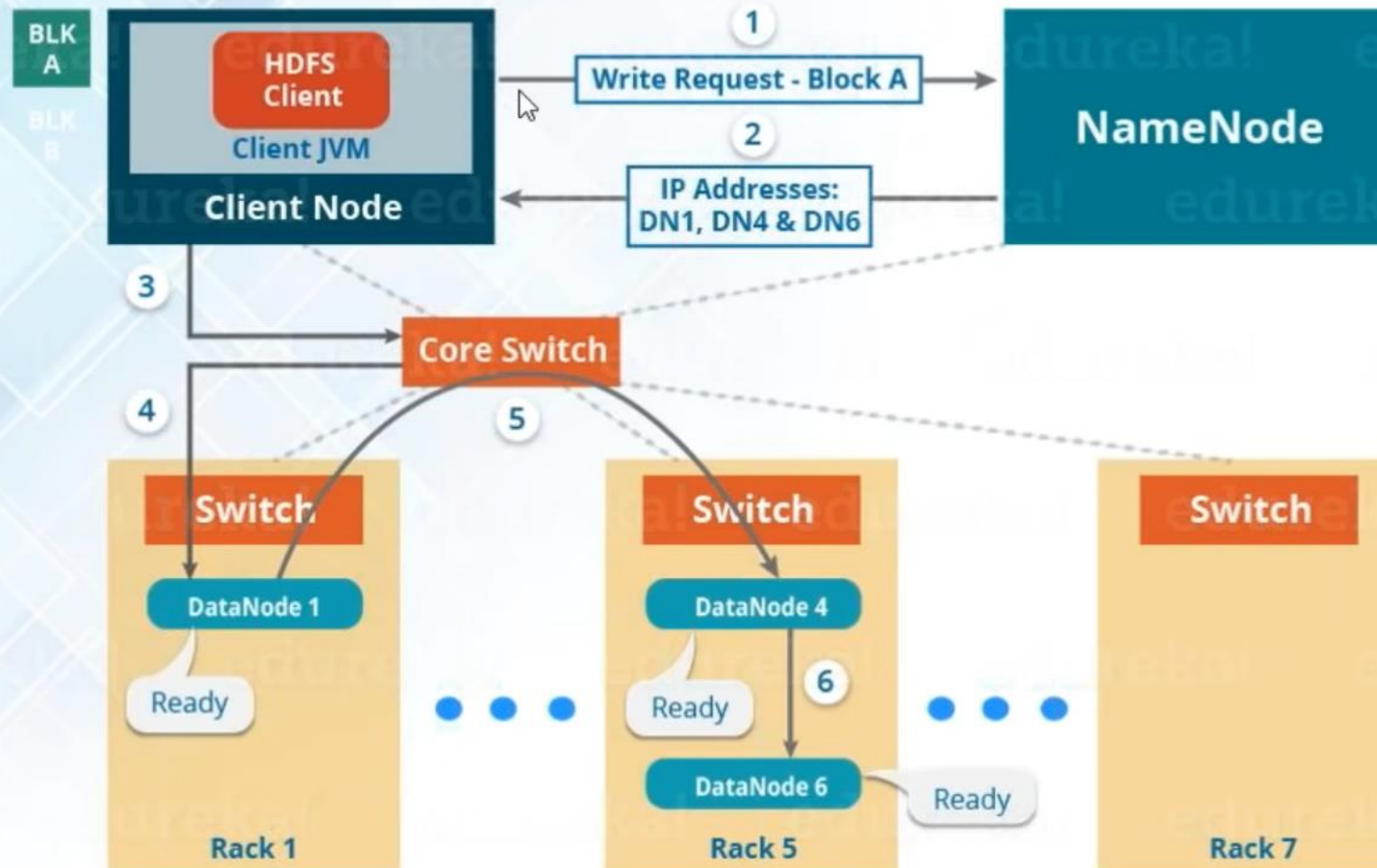
- 
- ▶ Rack– It the collection of machines around 40-50. All these machines are connected using the same network switch and if that network goes down then all machines in that rack will be out of service. Thus we say rack is down.
 - ▶ Rack Awareness was introduced by Apache Hadoop to overcome this issue. In Rack Awareness, NameNode chooses the DataNode which is closer to the same rack or nearby rack. NameNode maintains Rack ids of each DataNode to achieve rack information. Thus, this concept chooses Datanodes based on the rack information. NameNode in hadoop makes ensures that all the replicas should not stored on the same rack or single rack. Rack Awareness Algorithm reduces latency as well as Fault Tolerance.

HDFS Write Mechanism

- ▶ Successful commination is nothing but successful transaction of sync + ack request

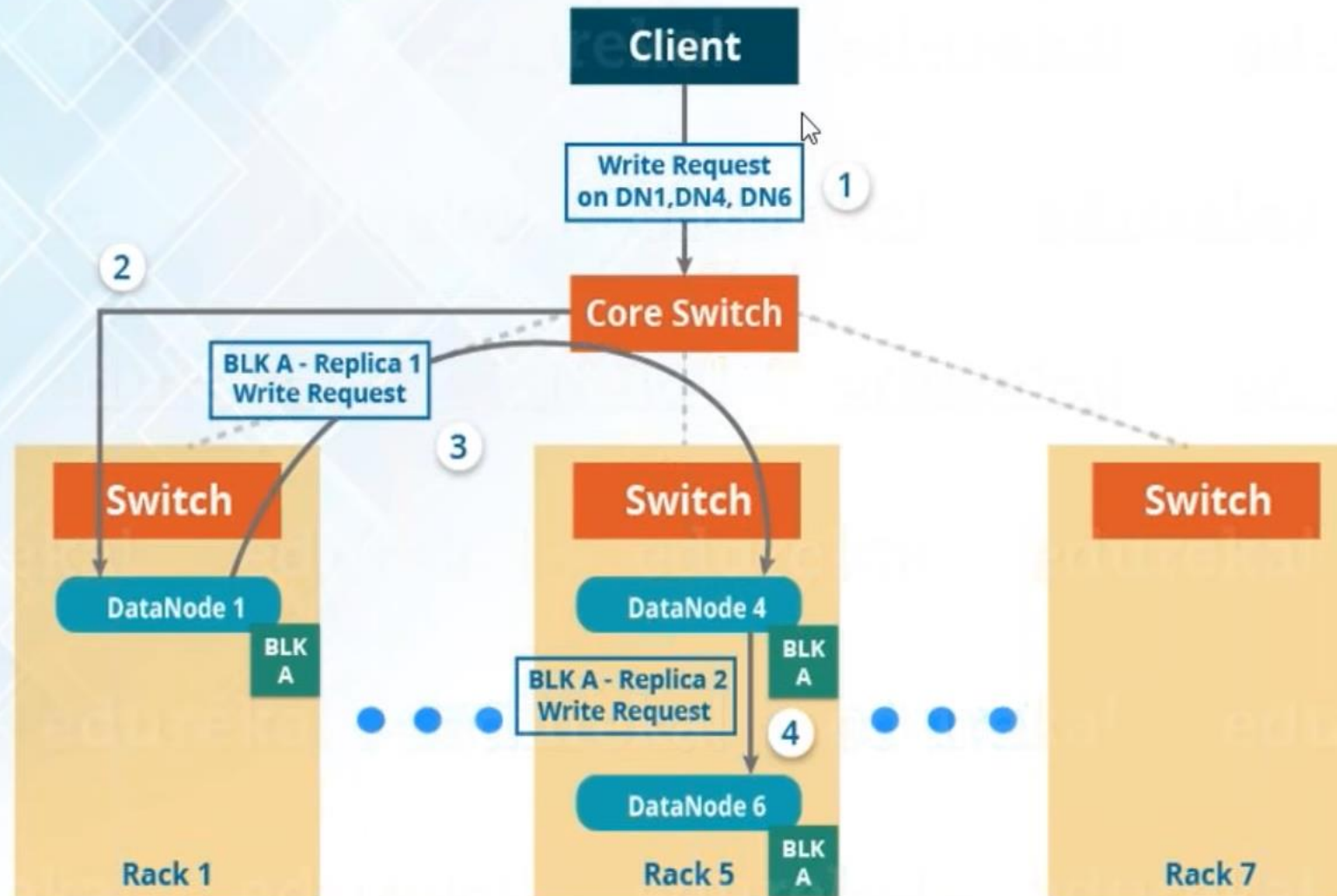
HDFS Write Mechanism – Pipeline Setup

Setting up HDFS - Write Pipeline



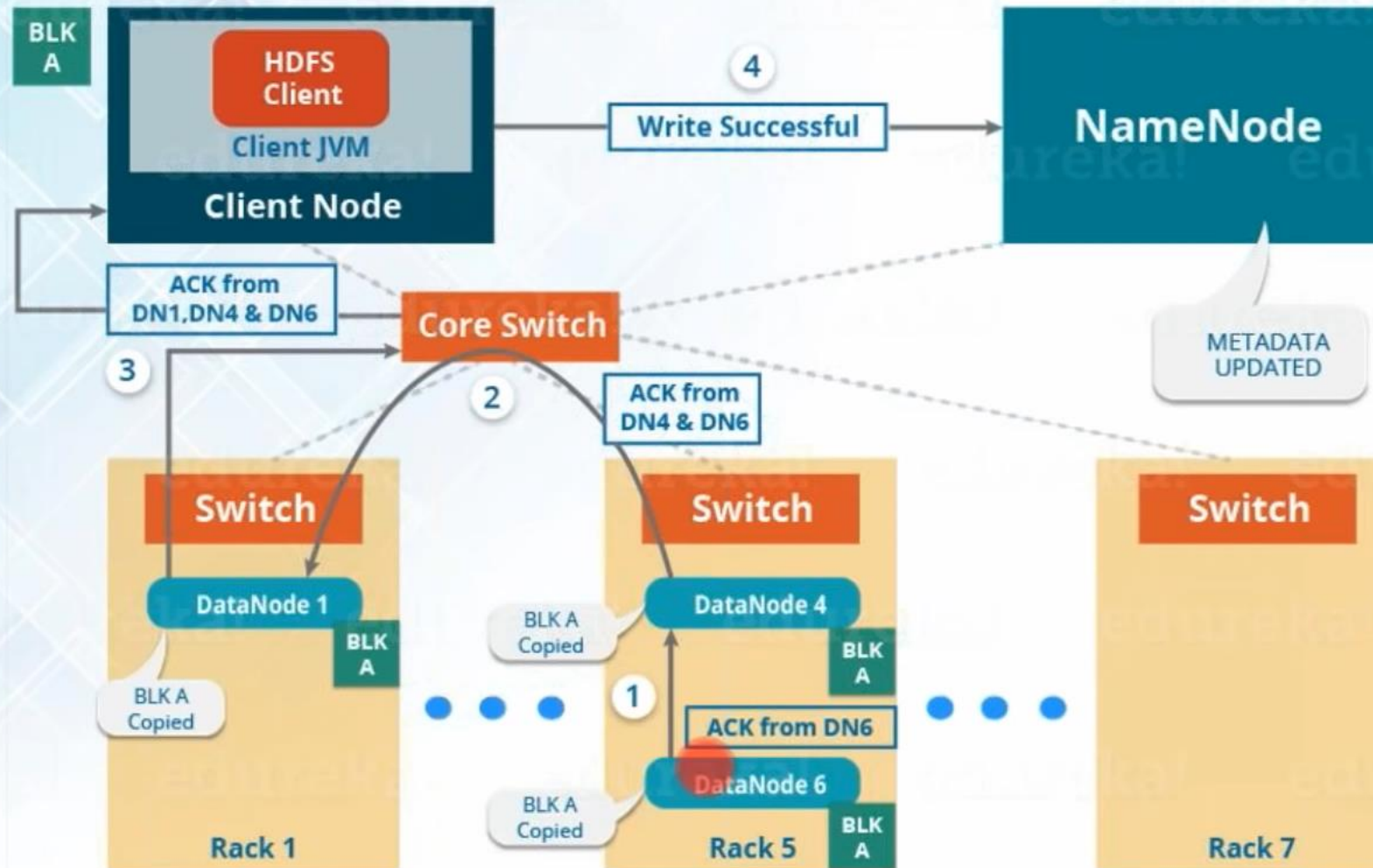
HDFS Write Mechanism – Writing a Block

HDFS - Write Pipeline



HDFS Write Mechanism - Acknowledgment

Acknowledgement in HDFS - Write



HDFS Read Mechanism

HDFS - Read Architecture

