

Big Data Analytics and Visualization (CDSC 715)

Addisu G. Semie, PhD
Asst. Prof., Computational Science Program,
Addis Ababa University
Email: addisu.semie@aau.edu.et

Big Data with Hadoop

- The core modules of Hadoop
- Hadoop MapReduce
- Hadoop YARN

Big Data Tools

These are some of the following tools used for Big Data Analytics: Hadoop, Pig, Apache Hbase, Apache Spark, Talend, Splunk, Apache Hive, Kafka



Hadoop

Big data storage is challenging

- Data Volumes are massive
- Reliability of Storing PBs of data is challenging
- All kinds of failures:
Disk/Hardware/Network Failure
- Probability of failures simply increase with the number of machines ...



One popular solution: Hadoop



Hadoop Cluster at Yahoo! (Credit: Yahoo)

Hadoop

Hadoop offers

- Redundant, Fault-tolerant data storage
- Parallel computation framework
- Job coordination



Hadoop

A little history on Hadoop:

- Hadoop is an open-source implementation based on **Google File System (GFS)** and **MapReduce** from Google
- Hadoop was created by **Doug Cutting** and **Mike Cafarella** in 2005
- Hadoop was donated to **Apache** in 2006



Hadoop: who are using it?

Social Media



User Tracking & Engagement



Homeland Security



eCommerce



Financial Services



Real Time Search

Google



Hadoop Resources

Hadoop at ND:

<http://ccl.cse.nd.edu/operations/hadoop/>

Apache Hadoop Documentation:

<http://hadoop.apache.org/docs/current/>

Data Intensive Text Processing with Map-Reduce

<http://lintool.github.io/MapReduceAlgorithms/>

HDFS - Hadoop Distributed File System

Problem 1: Data is too big to store on one machine.

HDFS - Hadoop Distributed File System

Problem 1: Data is too big to store on one machine.

***HDFS:** Store the data on multiple machines!*

HDFS - Hadoop Distributed File System

Problem 2: Very high end machines are too expensive

HDFS - Hadoop Distributed File System

Problem 2: Very high end machines are too expensive

HDFS: Run on commodity hardware!

HDFS - Hadoop Distributed File System

Problem 3: Commodity hardware will fail!

HDFS - Hadoop Distributed File System

Problem 3: Commodity hardware will fail!

***HDFS:** Software is intelligent enough to handle hardware failure!*

HDFS - Hadoop Distributed File System

Problem 4: What happens to the data if the machine stores the data fails?

HDFS - Hadoop Distributed File System

Problem 4: What happens to the data if the machine stores the data fails?

HDFS: Replicate the data!

HDFS - Hadoop Distributed File System

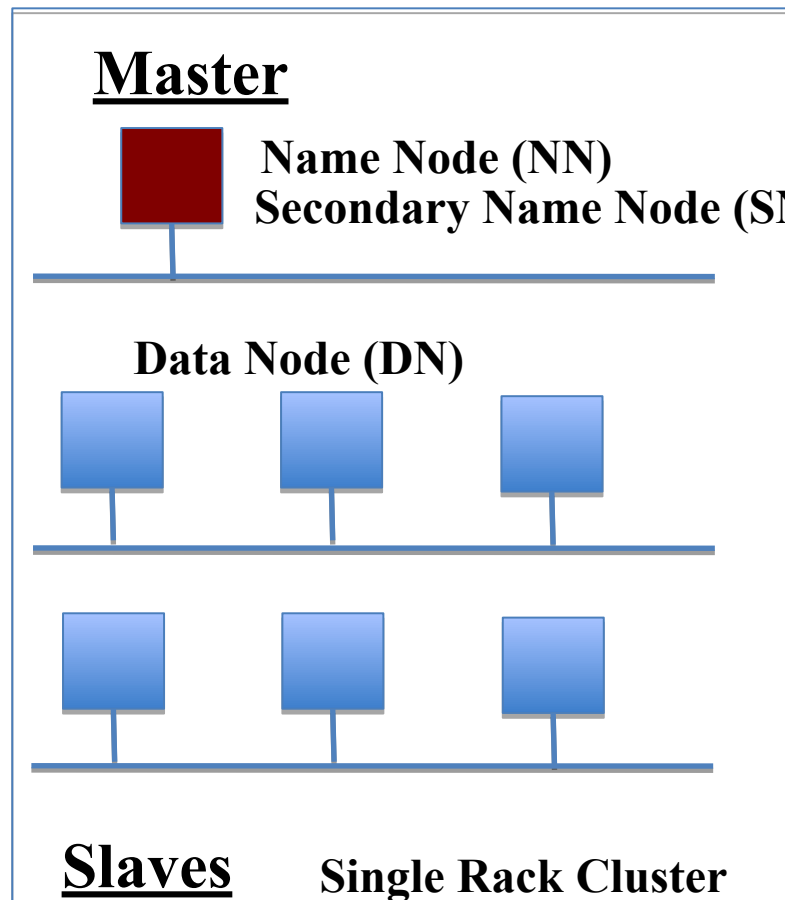
Problem 5: How can distributed machines organize the data in a coordinated way?

HDFS - Hadoop Distributed File System

Problem 5: How can distributed machines organize the data in a coordinated way?

HDFS: Master-Slave Architecture!

HDFS Architecture: Master-Slave



Name Node: Controller

- File System Name Space Management
- Block Mappings

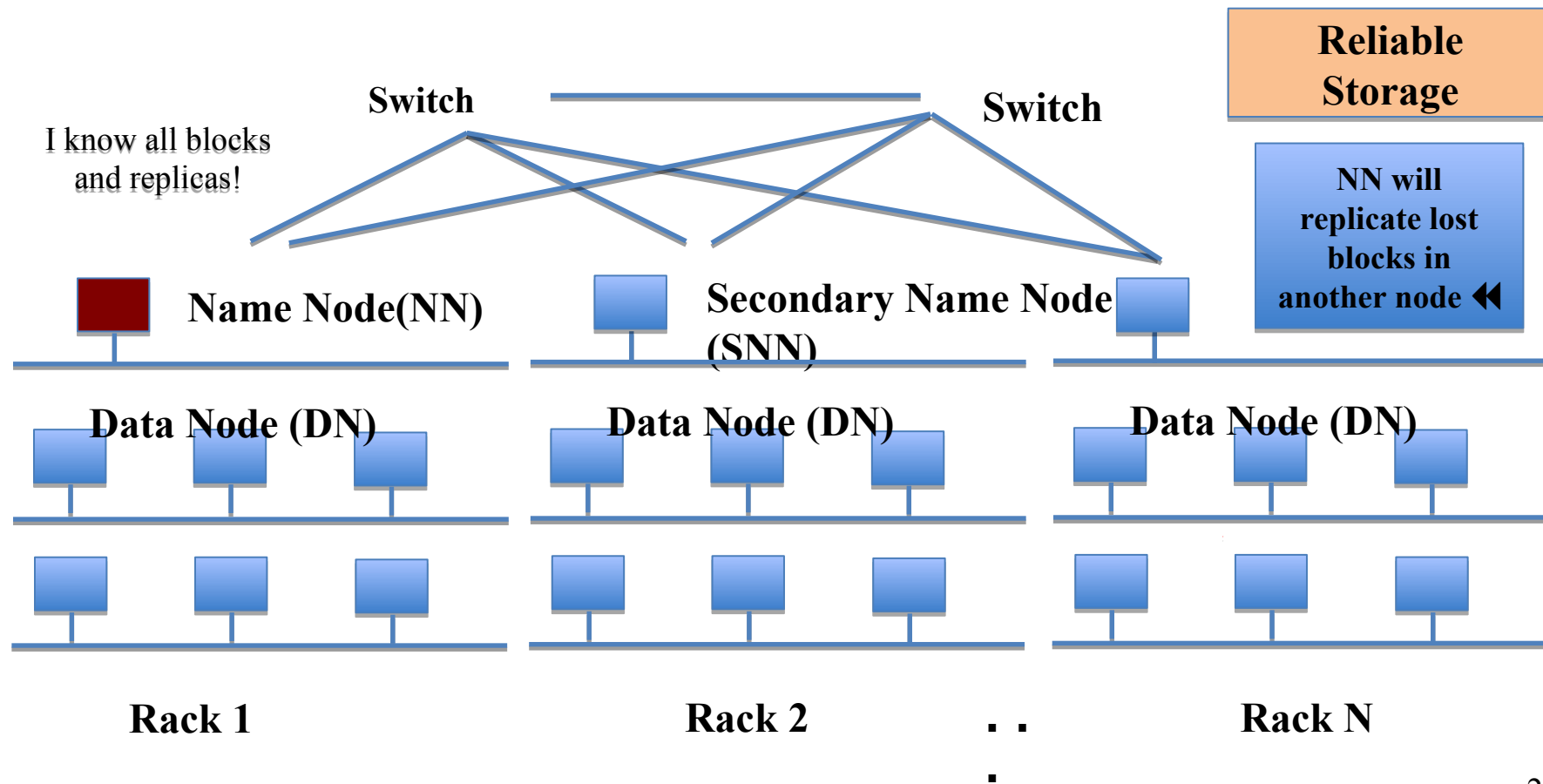
Data Node: Work Horses

- Block Operations
- Replication

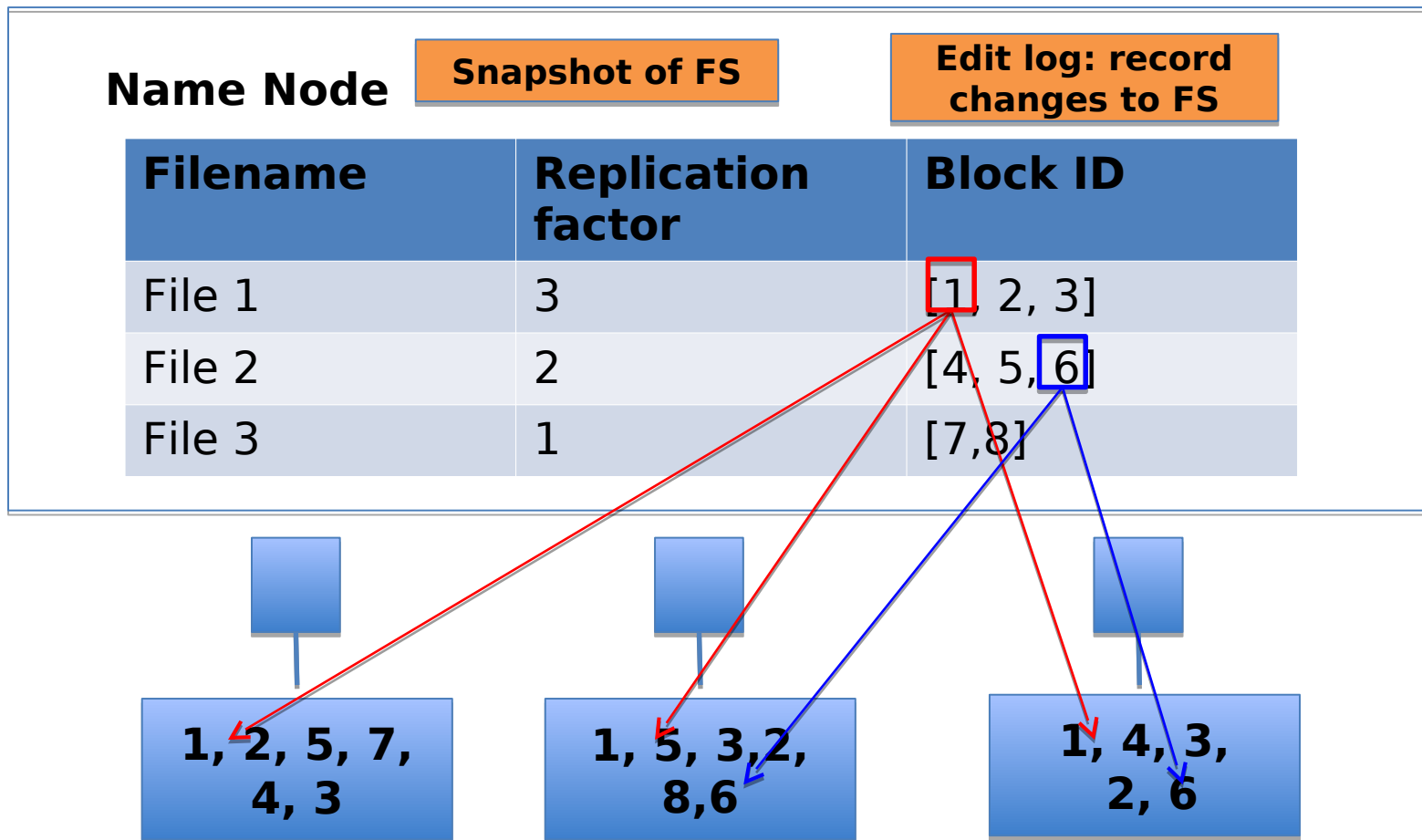
Secondary Name Node:

- Checkpoint node

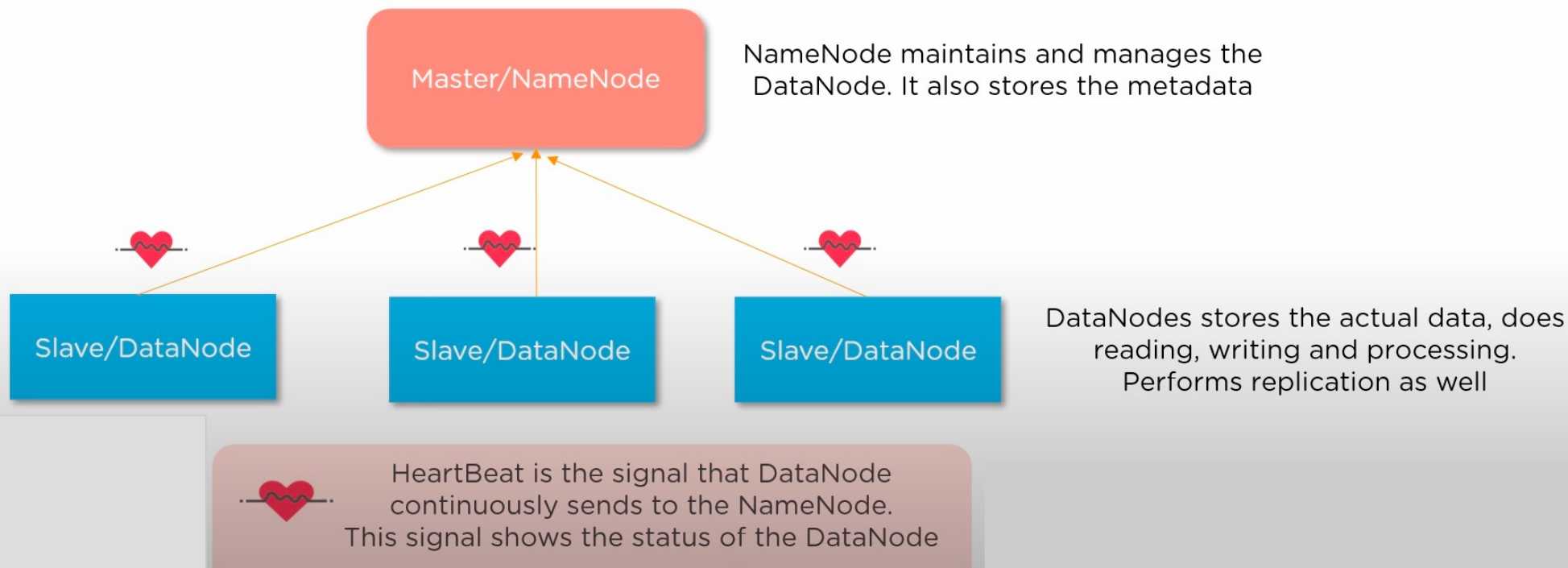
HDFS Architecture: Master-Slave



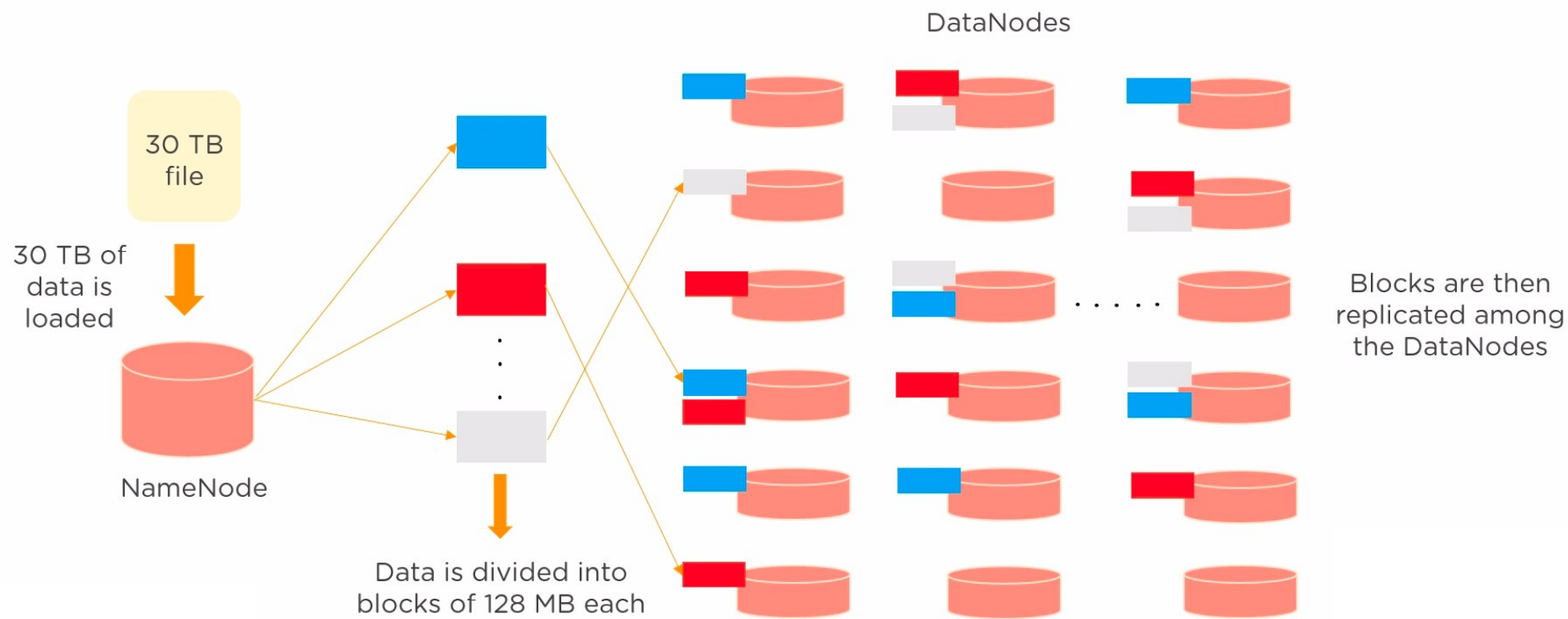
HDFS Inside: Name Node



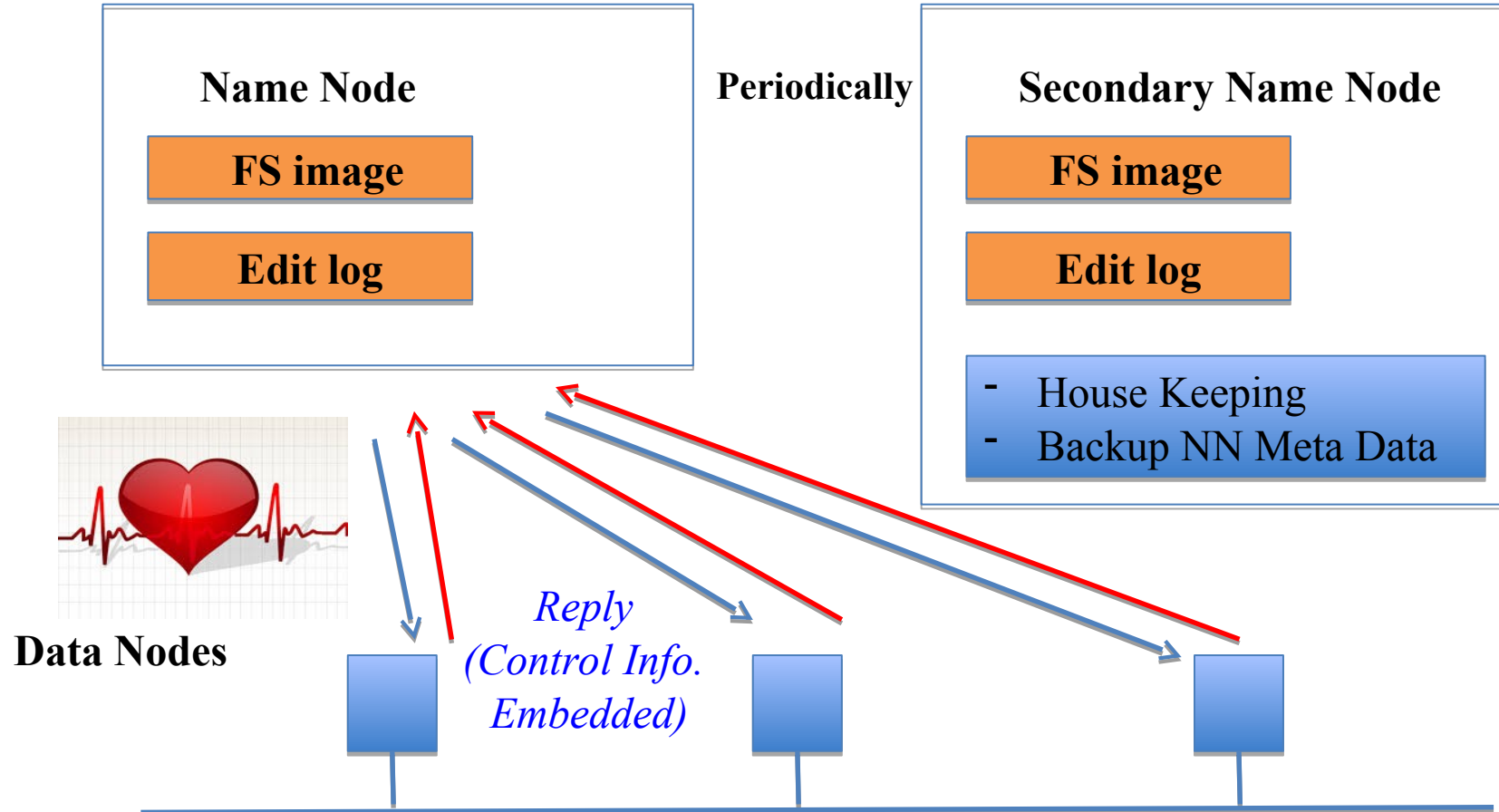
HDFS Architecture: Master-Slave



HDFS Architecture: Master-Slave



HDFS Inside: Name Node



HDFS Inside: Blocks

- Why do we need the abstraction “Blocks” in addition to “Files”?

HDFS Inside: Blocks

- Why do we need the abstraction “Blocks” in addition to “Files”?

Reasons:

- *File can be larger than a single disk*
- *Block is of fixed size, easy to manage and manipulate*
- *Easy to replicate and do more fine grained load balancing*

HDFS Inside: Blocks

- HDFS Block size is by default 64 MB, why it is much larger than regular file system block?

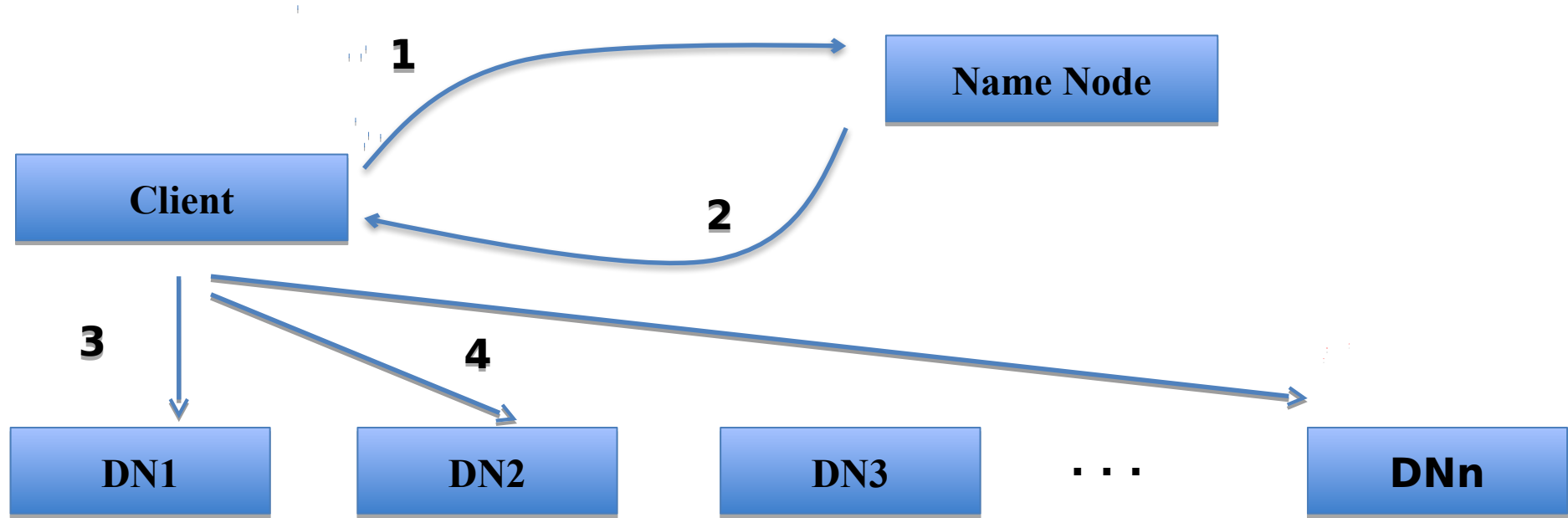
HDFS Inside: Blocks

- HDFS Block size is by default 64 MB, why it is much larger than regular file system block?

Reasons:

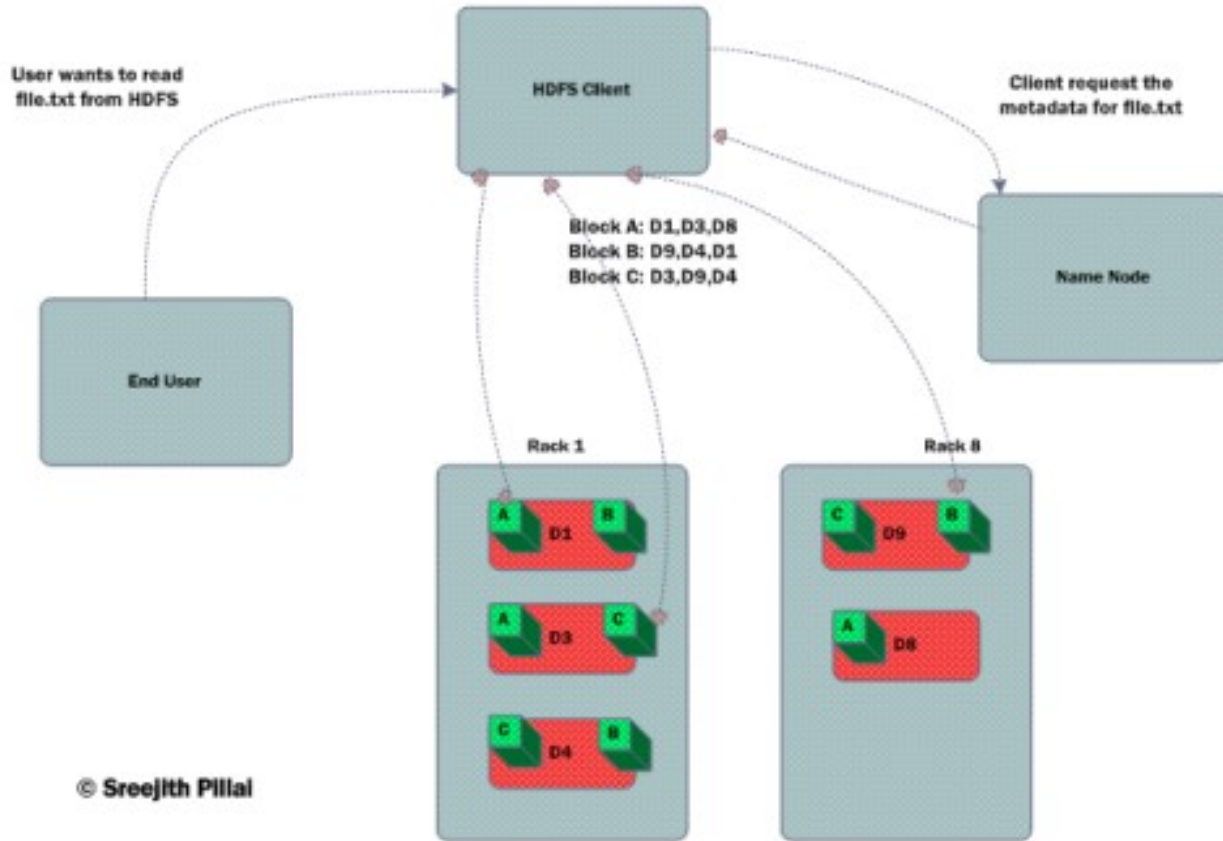
- Minimize overhead: disk seek time is almost constant
- Example: seek time: 10 ms, file transfer rate: 100MB/s, overhead (seek time/a block transfer time) is 1%
 - The ratio $\text{seekTime} / \text{transferTime}$ small (close to .01 in the text), it means we are reading data from the disk almost as fast as the physical limit imposed by the disk, with minimal time spent looking for information.

HDFS Inside: Read



1. Client connects to NN to read data
2. NN tells client where to find the data blocks
3. Client reads blocks directly from data nodes (without going through NN)
4. In case of node failures, client connects to another node that serves the missing block

HDFS Inside: Read



HDFS Inside: Read

- Why does HDFS choose such a design for read?
Why not ask client to read blocks through NN?

HDFS Inside: Read

- Why does HDFS choose such a design for read?
Why not ask client to read blocks through NN?
- *Reasons:*
 - *Prevent NN from being the bottleneck of the cluster*
 - *Allow HDFS to scale to large number of concurrent clients*
 - *Spread the data traffic across the cluster*

HDFS Inside: Read

- Given multiple replicas of the same block, how does NN decide which replica the client should read?

HDFS Inside: Read

- Given multiple replicas of the same block, how does NN decide which replica the client should read?
- *HDFS Solution:*
 - *Rack awareness based on network topology*

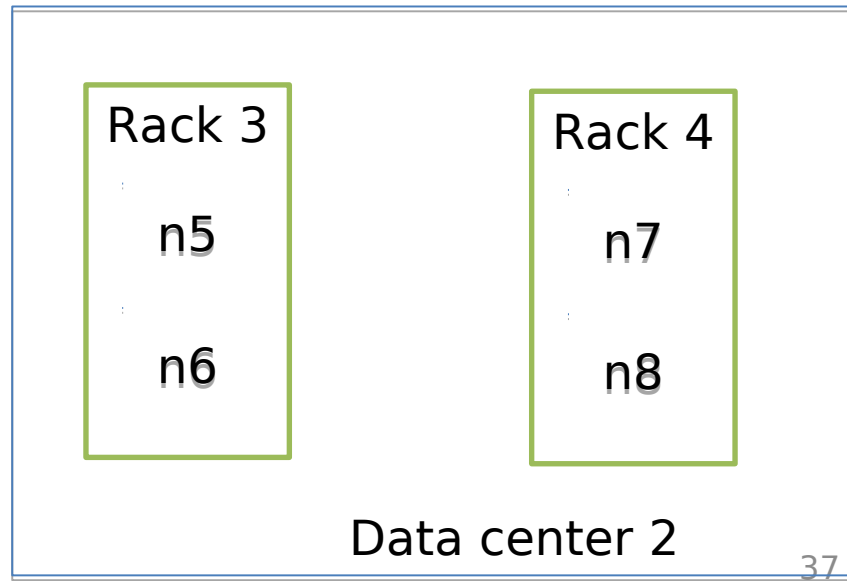
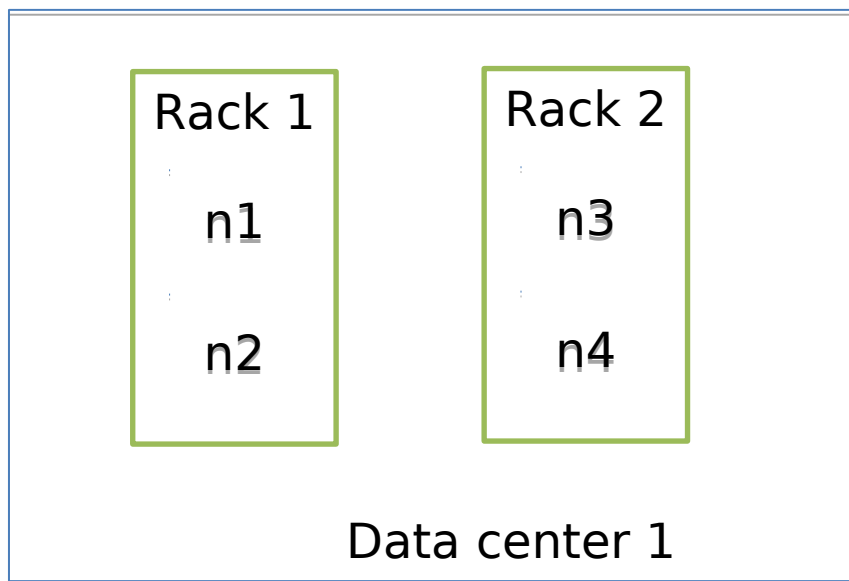
HDFS Inside: Network Topology

- The critical resource in HDFS is **bandwidth**, distance is defined based on that
- Measuring bandwidths between any pair of nodes is too complex and **does not scale**
- **Basic Idea:**
 - Processes on the same node
 - Different nodes on the same rack
 - Nodes on different racks in the same data center (cluster)
 - Nodes in different data centers

**Bandwidth
becomes less**

HDFS Inside: Network Topology

- HDFS takes a simple approach:
 - See the network as a tree
 - **Distance between two nodes is the sum of their distances to their closest common ancestor**



HDFS Inside: Network Topology

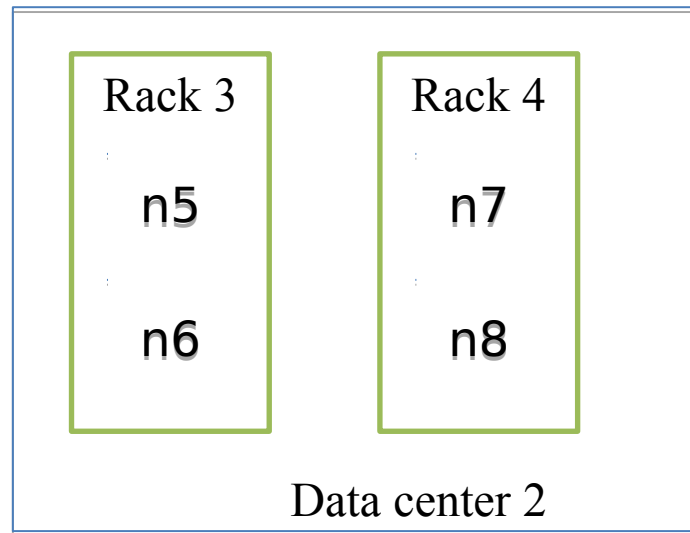
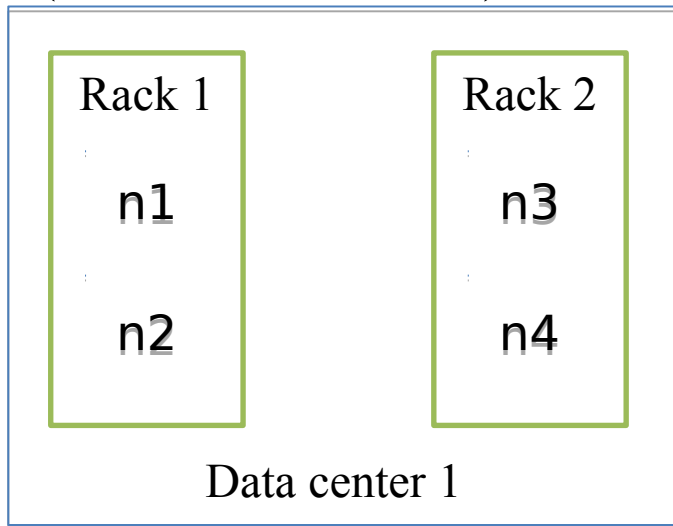
- What are the distance of the following pairs:

Dist (d1/r1/n1, d1/r1/n1) = 0

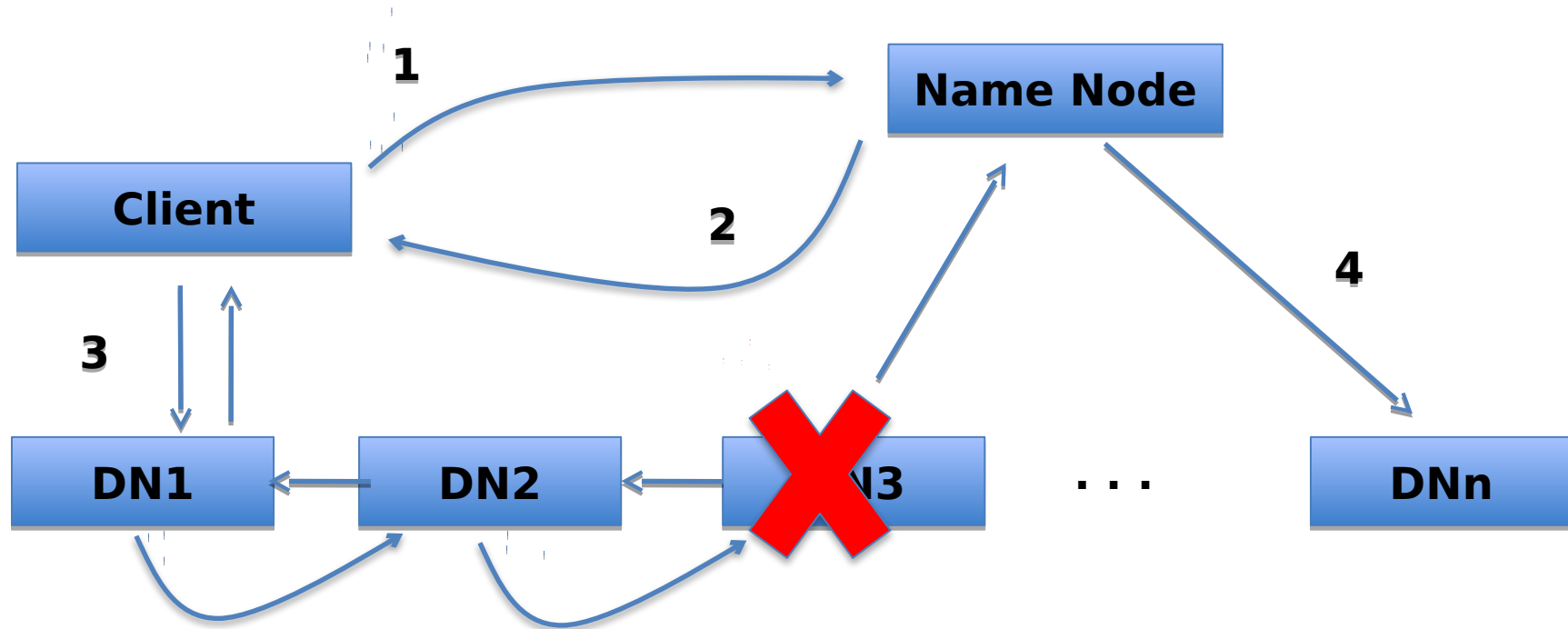
Dist(d1/r1/n1, d1/r1/n2) = 2

Dist(d1/r1/n1, d1/r2/n3) = 4

Dist(d1/r1/n1, d2/r3/n6) = 6

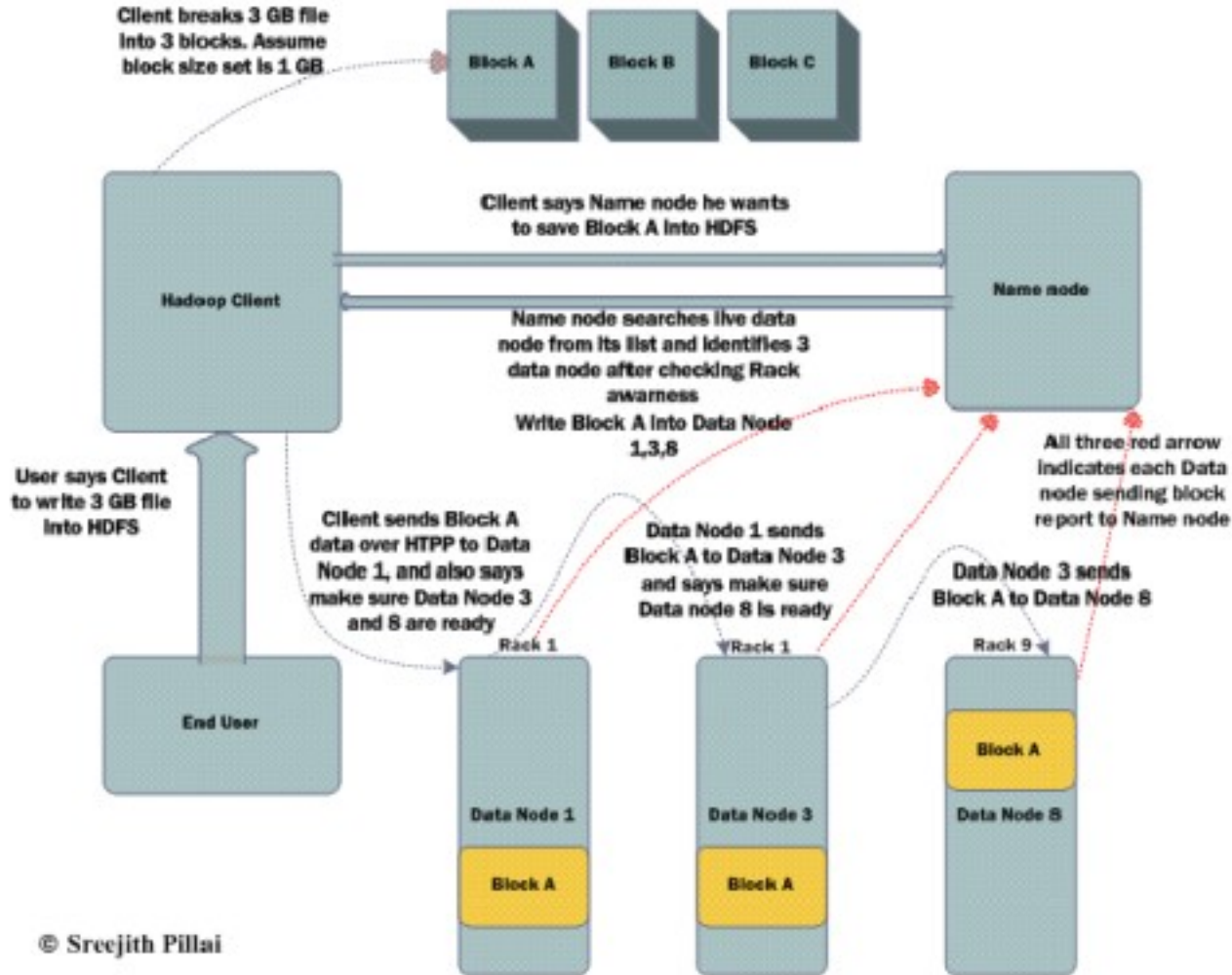


HDFS Inside: Write



1. Client connects to NN to write data
2. NN tells client write these data nodes
3. Client writes blocks directly to data nodes with desired replication factor
4. In case of node failures, NN will figure it out and replicate the missing blocks

HDFS Inside: Write



HDFS Inside: Write

- Where should HDFS put the three replicas of a block? What trade-offs we need to consider?

HDFS Inside: Write

- Where should HDFS put the three replicas of a block? What trade-offs we need to consider?
- *Trade-offs:*
 - *Reliability*
 - *Write Bandwidth*
 - *Read Bandwidth*

What are some possible strategies?










HDFS Inside: Write

- Replication Strategy vs Tradeoffs

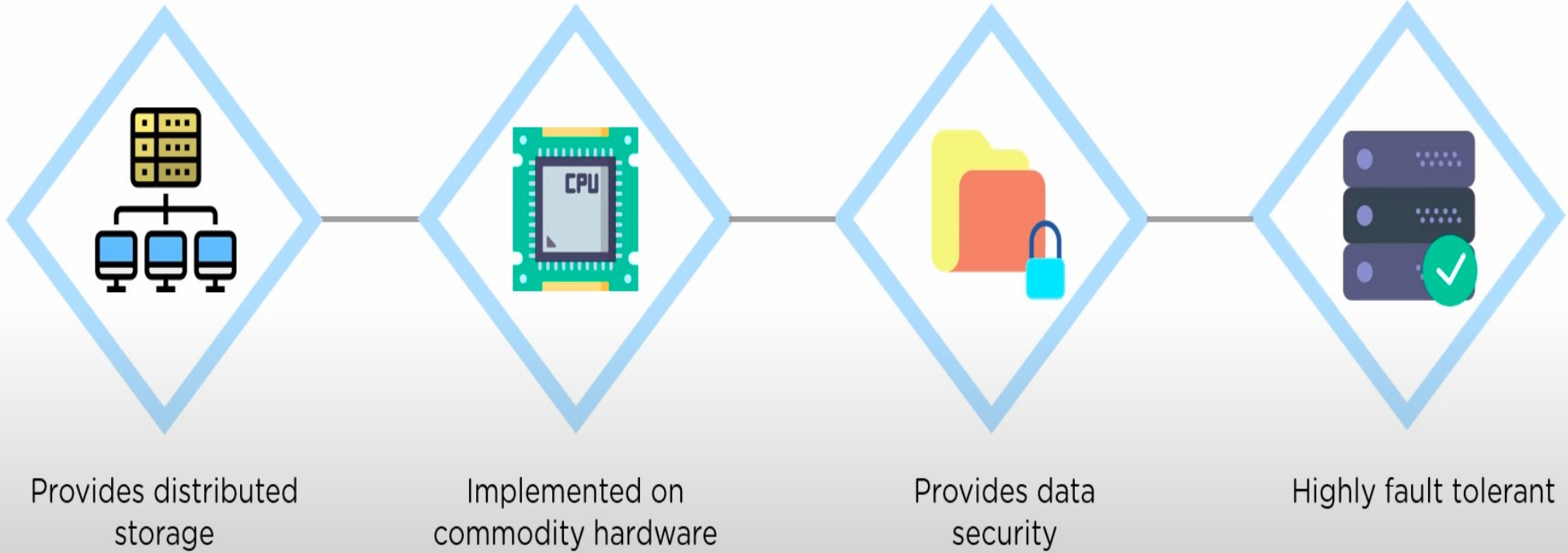
	Reliability	Write Bandwidth	Read Bandwidth
Put all replicas on one node			
Put all replicas on different racks			

HDFS Inside: Write

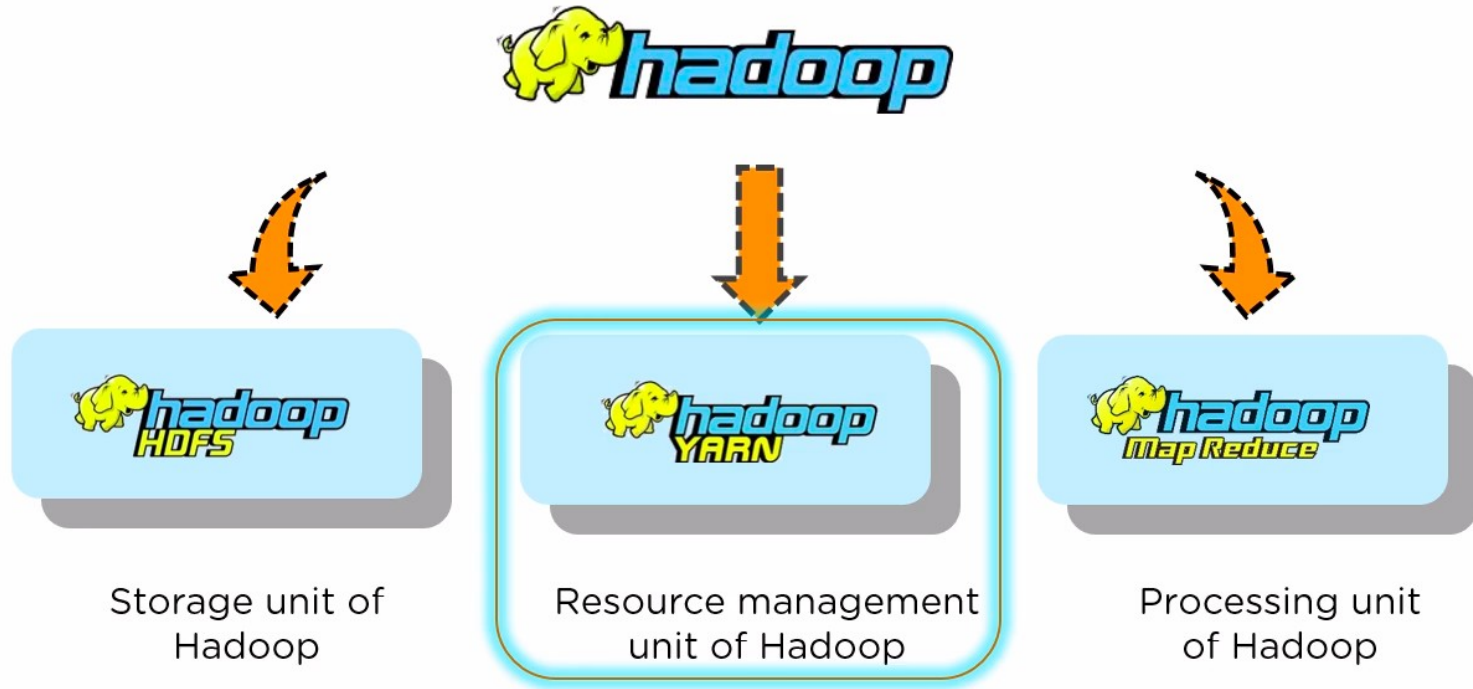
- Replication Strategy vs Tradeoffs

	Reliability	Write Bandwidth	Read Bandwidth
Put all replicas on one node			
Put all replicas on different racks			
HDFS: 1-> same node as client 2-> a node on different rack 3-> a different node on the same rack as 2			

Features of HDFS



Components of Hadoop version 2.0



Map Reduce

What is MapReduce?

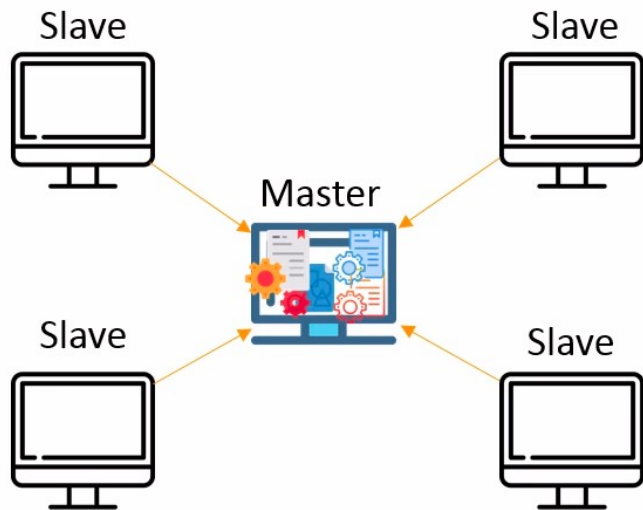
Hadoop MapReduce is a programming technique where huge data is processed in a parallel and distributed fashion



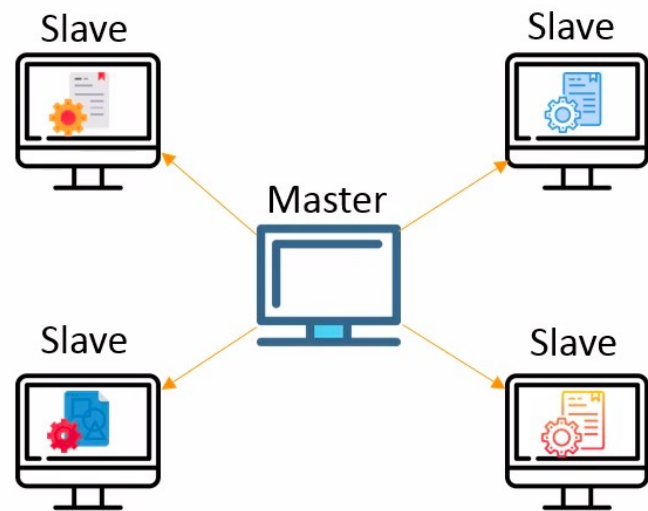
MapReduce is used for parallel processing of the Big Data, which is stored in HDFS

What is MapReduce?

In MapReduce approach, processing is done at the slave nodes and the final result is sent to the master node

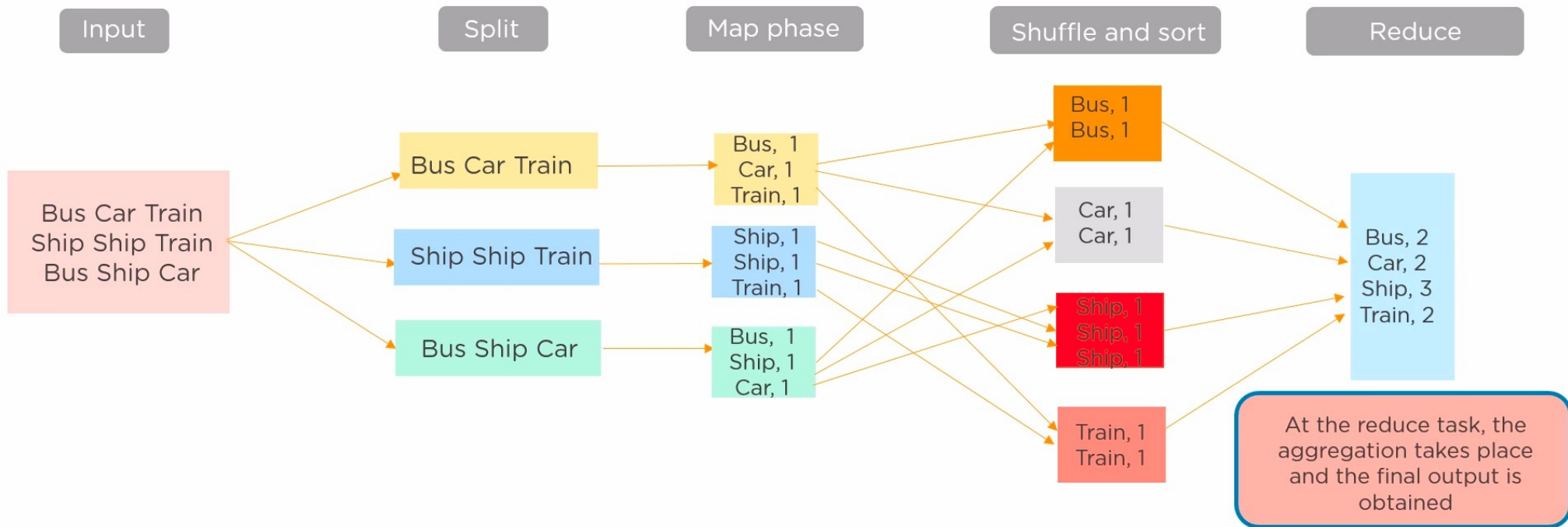


Traditional approach – Data is processed at the Master node



MapReduce approach – Data is processed at the Slave nodes

What is MapReduce?



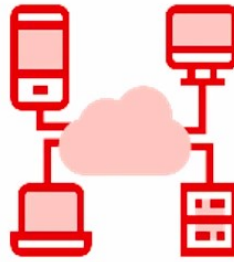
YARN – Yet Another Resource Negotiator

What is YARN?

YARN - Yet Another Resource Negotiator



Acts like an OS
to Hadoop 2



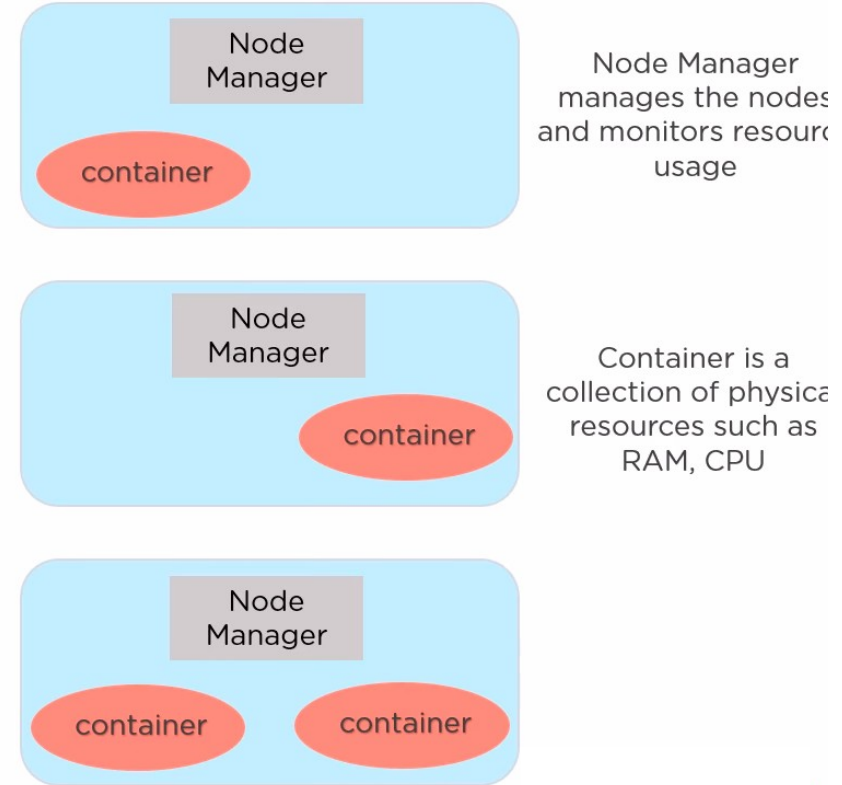
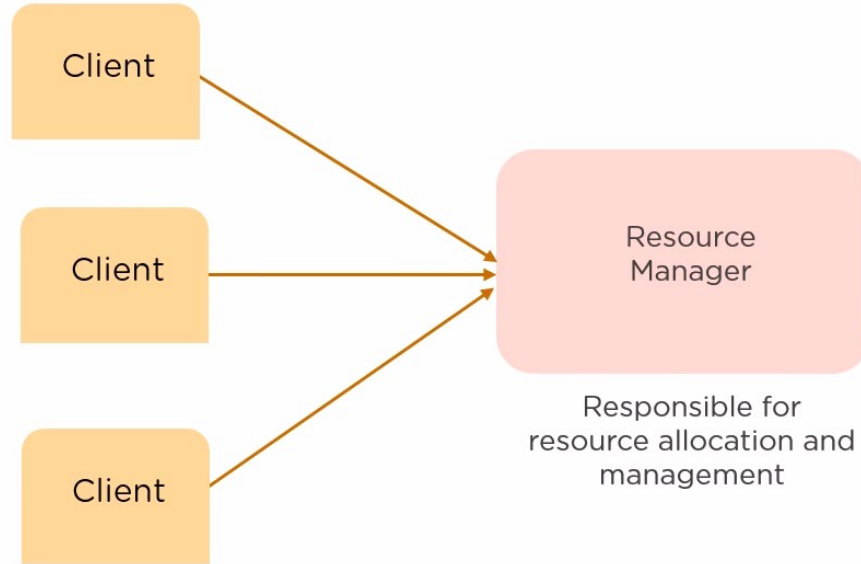
Responsible for managing
cluster resources



Does job scheduling

What is YARN?

Client submits
the job request



What is YARN?

Client submits
the job request

