BIG DATA and AI for business

Hadoop

Decisions, Operations & Information Technologies
Robert H. Smith School of Business
Fall, 2020

Distributed processing is non-trivial

- How to assign tasks to different workers in an efficient way?
- What happens if tasks fail?
- How do workers exchange results?
- How to synchronize distributed tasks allocated to different workers?

Big data storage is challenging

- Data volumes are massive
- Reliability of storing PBs of data is challenging
- All kinds of failures: Disk/Hardware/Network Failures
- Probability of failures simply increase with the number of machines …

One popular solution: Hadoop

Hadoop Cluster at Yahoo!

Hadoop offers

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

Hadoop offers

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

Programmers

No longer need to worry about

Q: Where file is located?

Q: How to handle failures & data lost?

Q: How to divide computation?

Q: How to program for scaling?

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

Who uses Hadoop?

Social

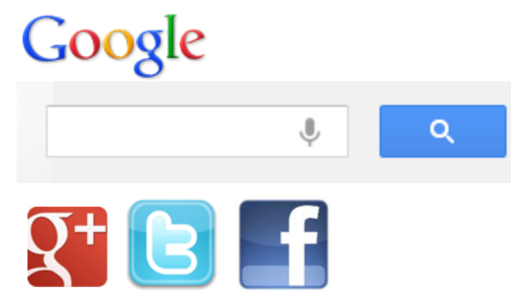
User Tracking & Engagement

Homeland Security

eCommerce

Financial Services

Real Time Search



Who uses Hadoop?

2007

- Yahoo!
- Powerset
- last.fm

2008

- Google
- ablegrape
- ImageShack
- Cascading
- IBM
- facebook
- ENORMO
- A9
- The University of Edinburgh
- krugle
- rackspace hosting
- Lookery
- The New York Times
- Joost
- Zvents
- Information Sciences Institute
- News Corporation
- Cornell University Computing and Information Science
- Visible Measures
- LOTAME
- NetSeer
- parc Palo Alto Research Center
- Security Enhanced Domain Name System
- veoh

2009

- AOL
- cloudera
- deepdyve
- cooliris
- eyealike
- TEXTMAP
- PSG College of Technology
- iterend
- tailsweep
- hulu
- RapLeaf
- USCMS
- Ning
- quantcast
- amazon web services
- pressflip
- detikSearch
- WorldLingo
- Systems@ETH Zürich
- VK SOLUTIONS
- TARAGANA
- HOSTING HABITAT
- HOLA
- Terrier
- adknowledge
- stampede

2010

- Samsung
- rubicon
- Berkeley Lab
- Visible Technologies
- Apollo Group
- ADSDAQ
- rackspace hosting
- RapLeaf
- wordnik
- mobigen
- comScore
- trulia
- Accela Communications
- Forward3D
- LinkedIn
- Microsoft
- Infochimps
- Pharm2Phork
- AdMeld
- gumgum
- BrainPad
- Pronux
- The Datagraph Blog
- Netflix
- mobileanalytics.tv
- markt24.de
- twitter
- media6degrees
- Beebler
- SLC Security
- eBay

Why HDFS?

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

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

Why HDFS?

- **Problem 2:** Very high end machines are too expensive

- **HDFS:** Run on commodity hardware!

Why HDFS?

- **Problem 3:** Commodity hardware can fail

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

Why HDFS?

- **Problem 4:** What happens to the data if the machine storing the data fails?

- **HDFS:** Replicate the data!

Why HDFS?

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

- **HDFS:** Master-Slave Architecture!

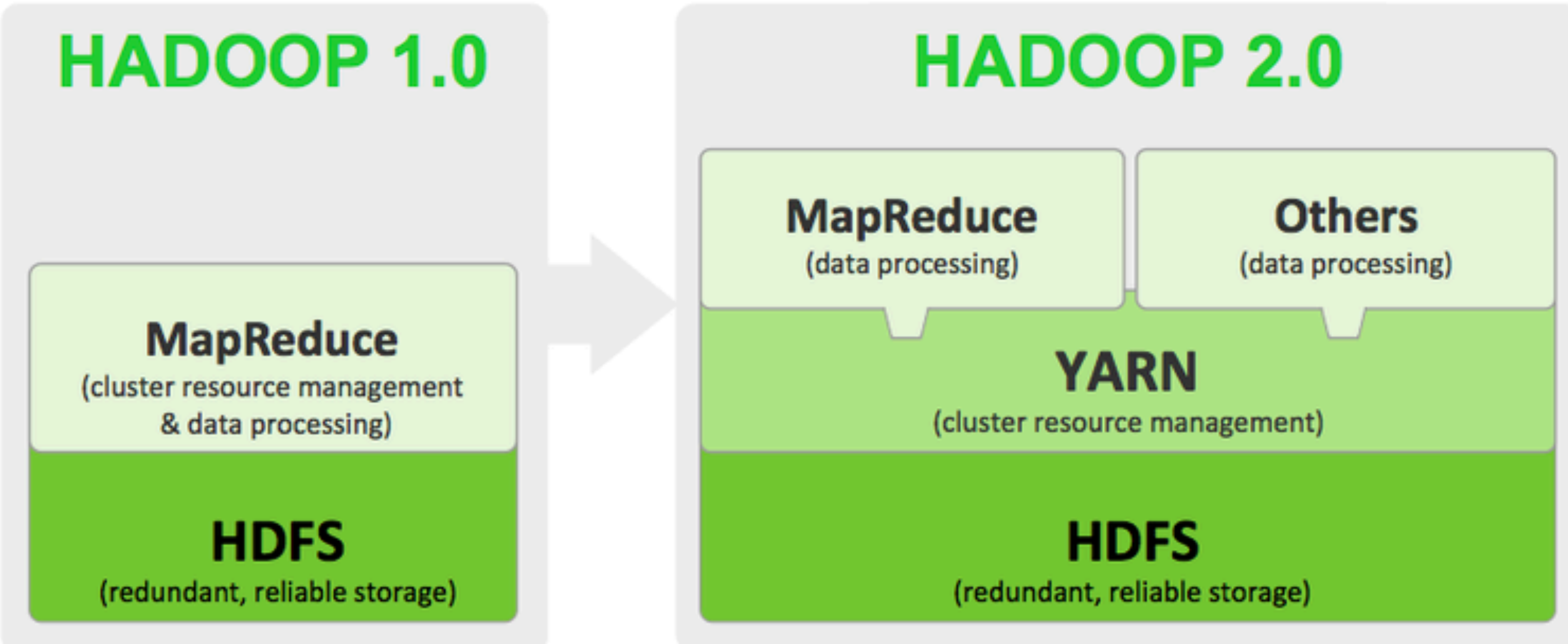
Another reason why Hadoop

- **Scan** 100TB datasets on a 1000-node cluster
  - Remote storage @ 10MB/s = 165 mins
  - Local storage @ 50-200MB/s = 33-8 mins
- Moving computation is more efficient than moving data
- Need fault tolerant store with reasonable availability guarantees
  - Handle hardware faults transparently

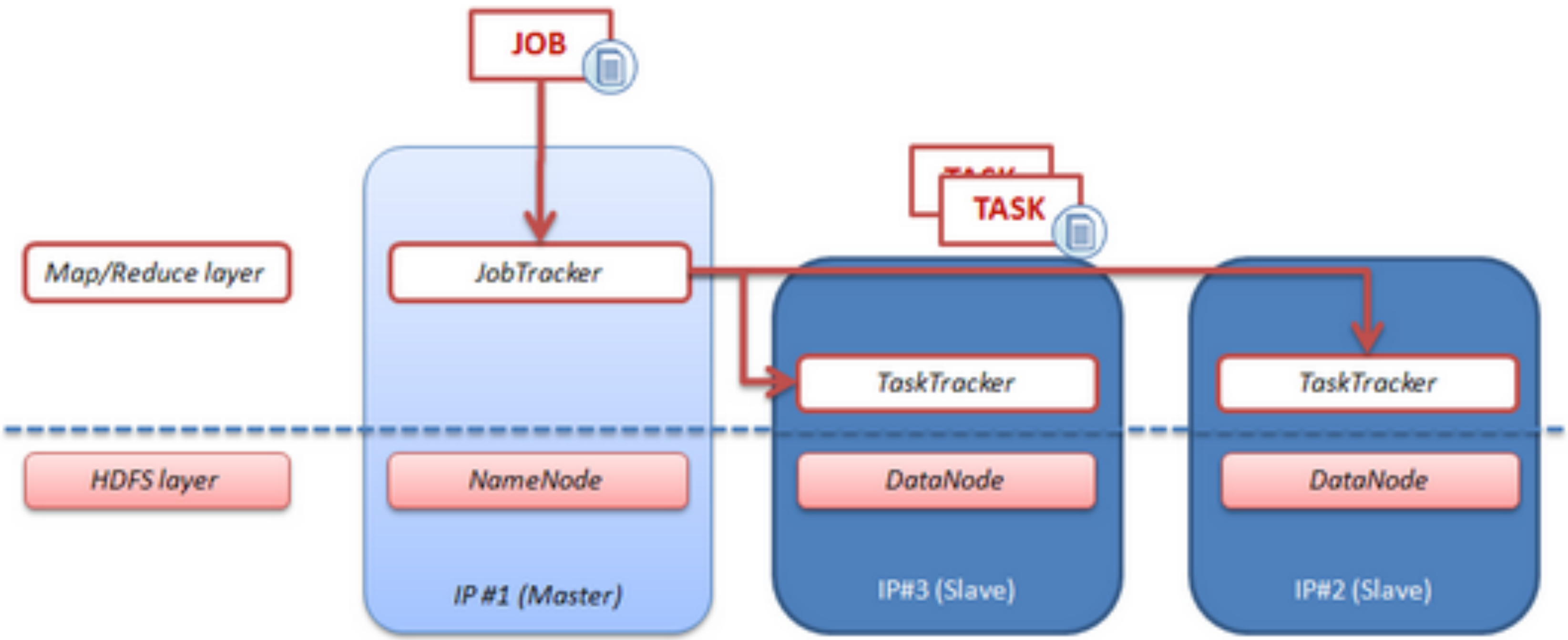
Hadoop goals

- **Scalable**: Petabytes (10^{15} Bytes) of data on thousands on nodes
- **Economical**: Commodity components only
- **Reliable**: fault tolerance

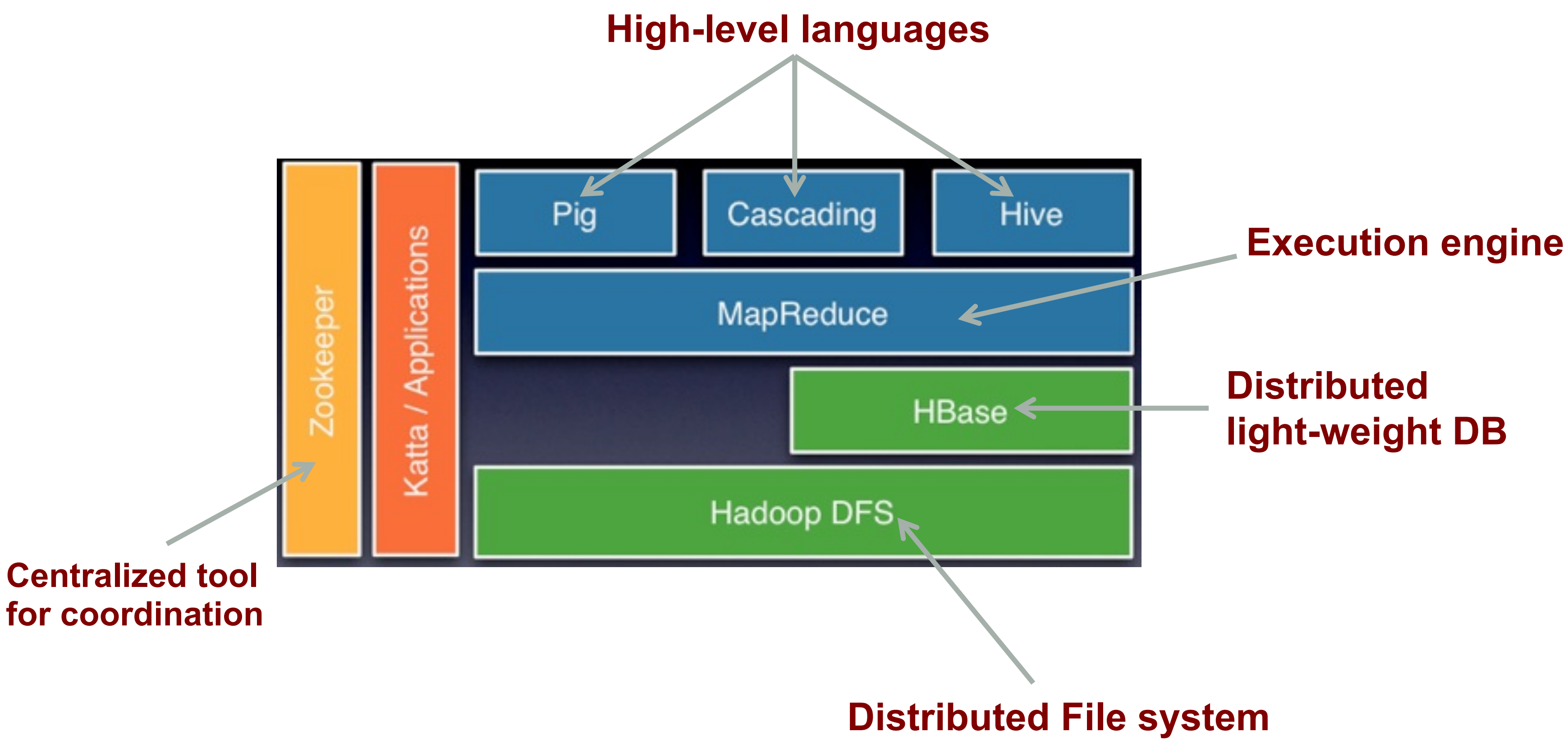
Hadoop big picture



High-level architecture of Hadoop

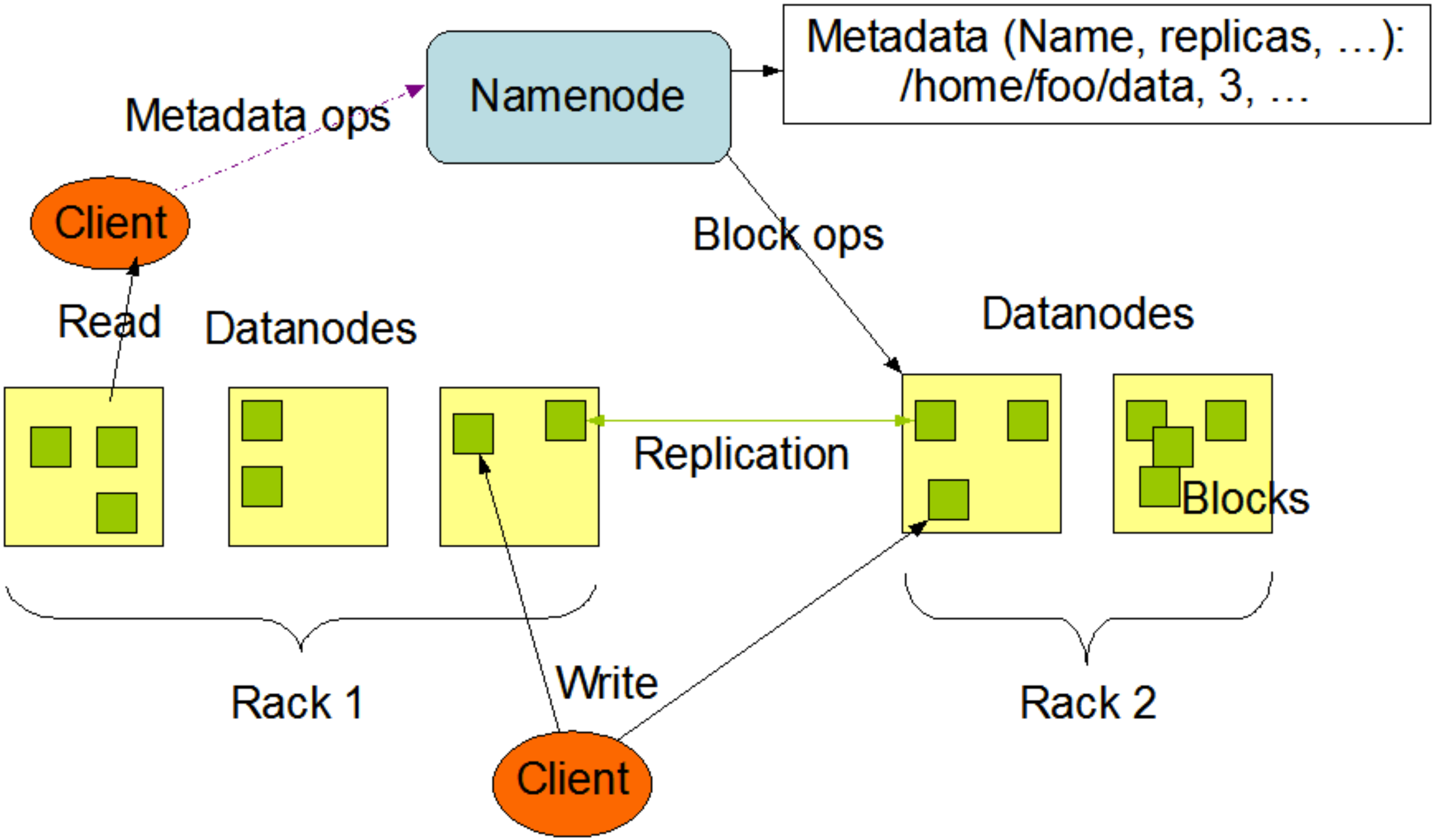


Hadoop big picture

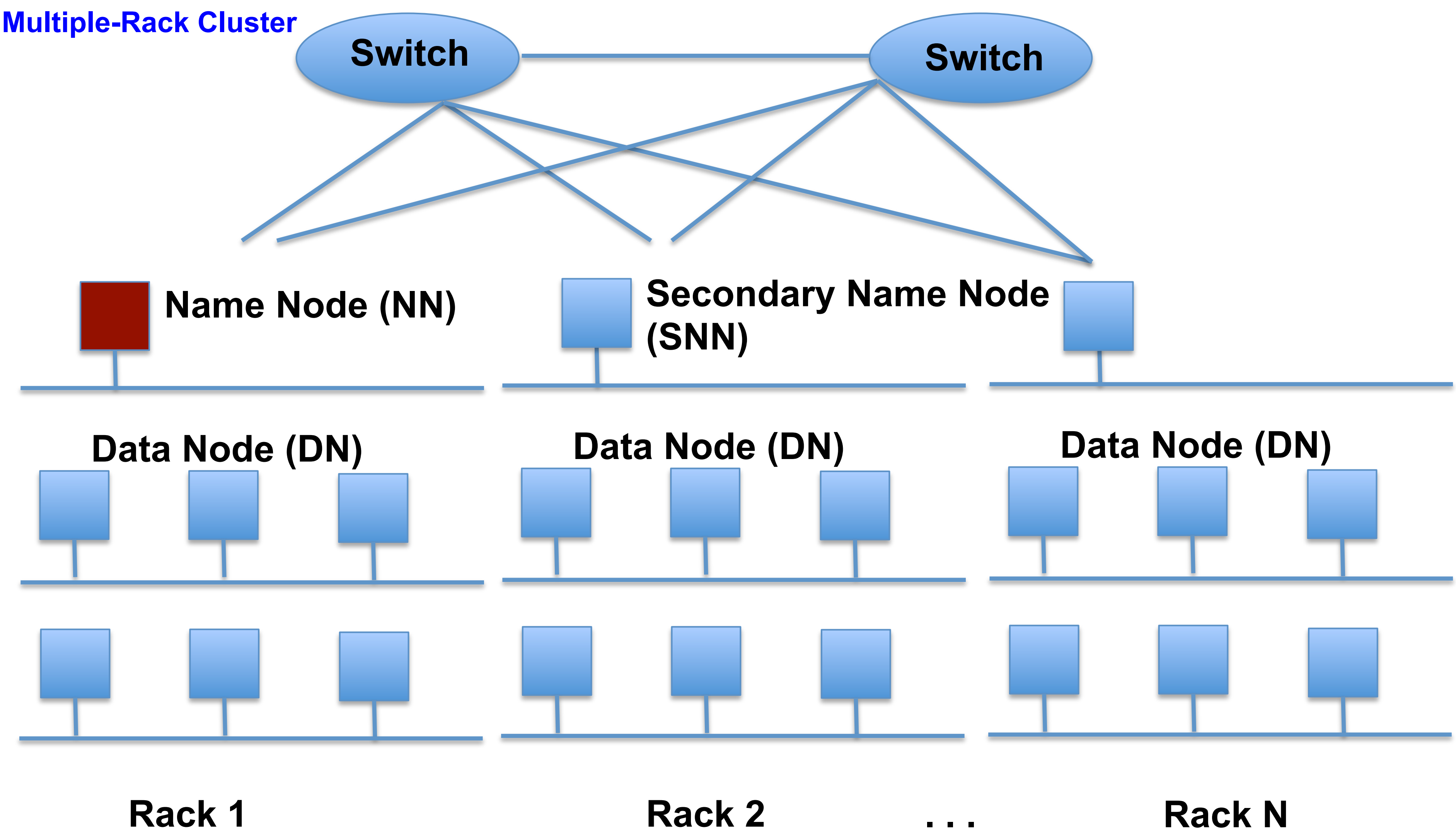


HDFS + MapReduce are enough to have things working

HDFS Architecture



HDFS architecture



HDFS

- **Master-Slave** architecture
- Single NameNode
  - ❑Sometimes a backup: secondary NameNode
- Many (Thousands) DataNodes
- Files are split into fixed sized blocks and stored on data nodes
- Data blocks are replicated for fault tolerance and fast access (default: 3)

HDFS – Master (NameNode)

- Manages file system (FS) namespace
- File metadata
- Mapping file to list of blocks
- Authorization & Authentication
- Mapping of datanode to list of blocks
- Monitor datanode health
- Replicate missing blocks
- Keeps ALL namespace in memory

HDFS – Slave (DataNode)

- Handle block storage on multiple volumes & block integrity
- Clients access the blocks directly from data nodes
- Periodically send heartbeats and block reports to NameNode
- Blocks are stored as underlying OS's files

HDFS Name Node

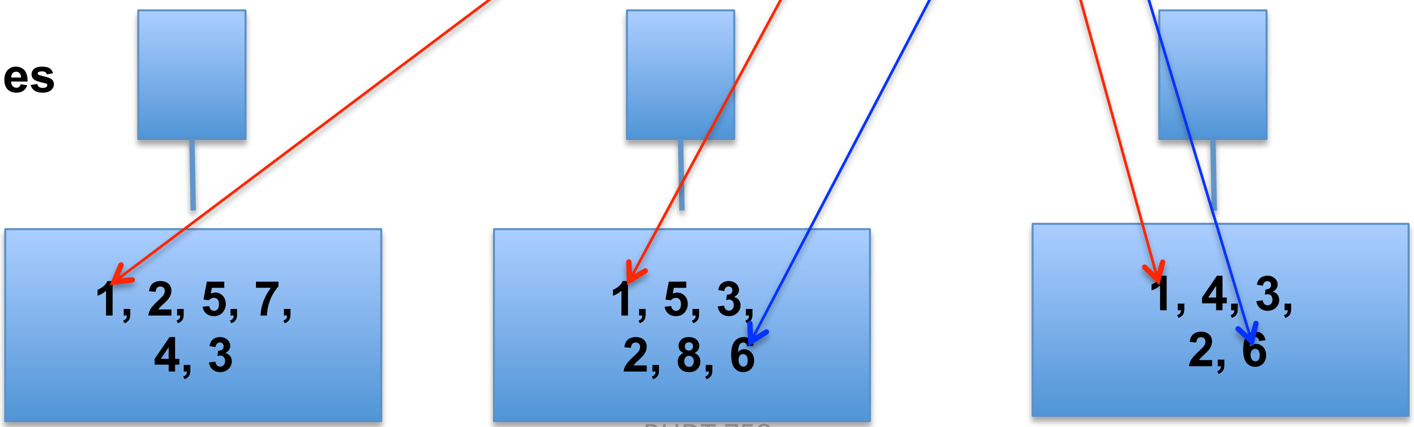
Name Node

Snapshot of FS

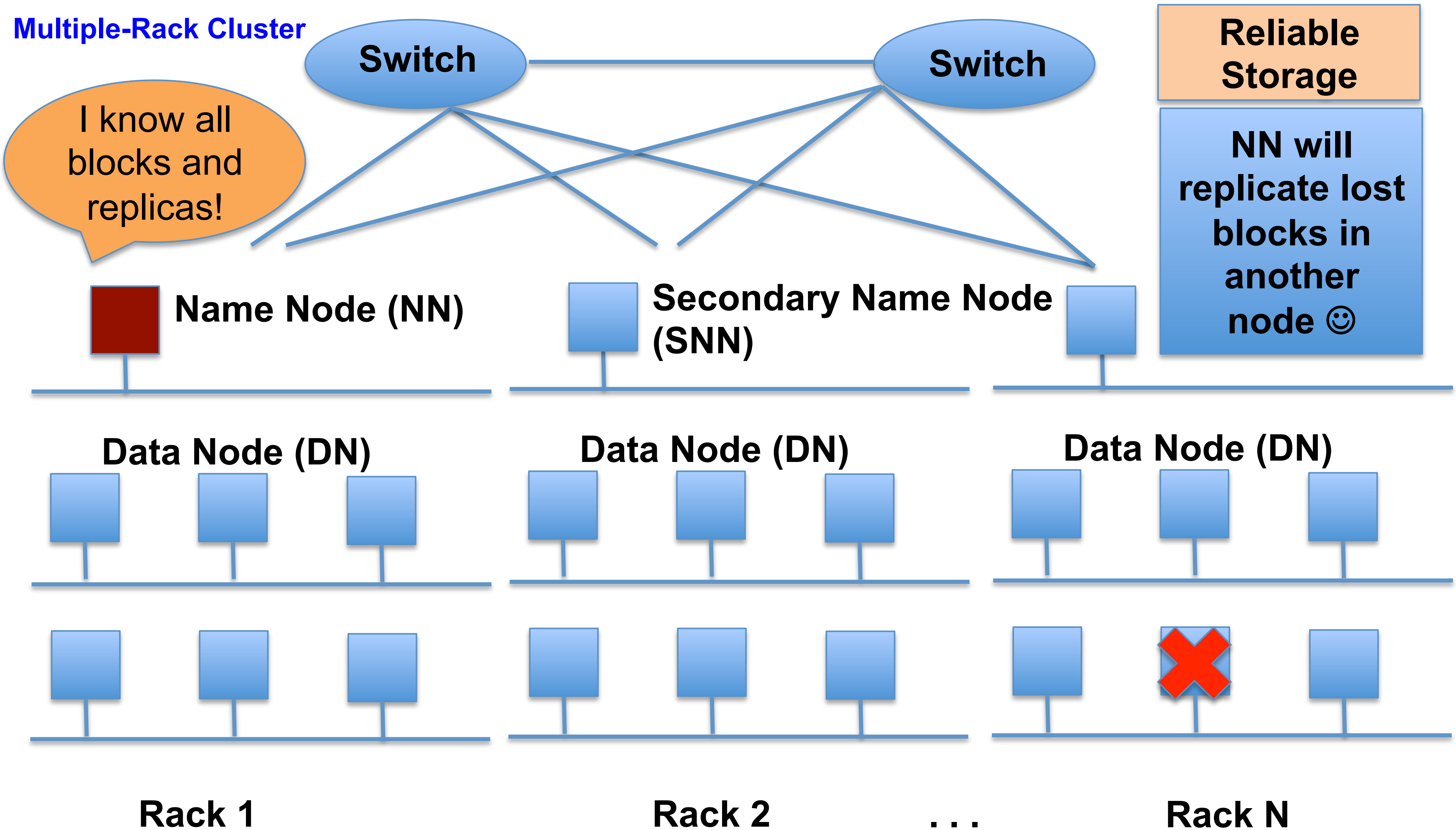
Edit log: record changes to FS

Filename	Replication factor	Block ID
File 1	3	[1, 2, 3]
File 2	2	[4, 5, 6]
File 3	1	[7,8]

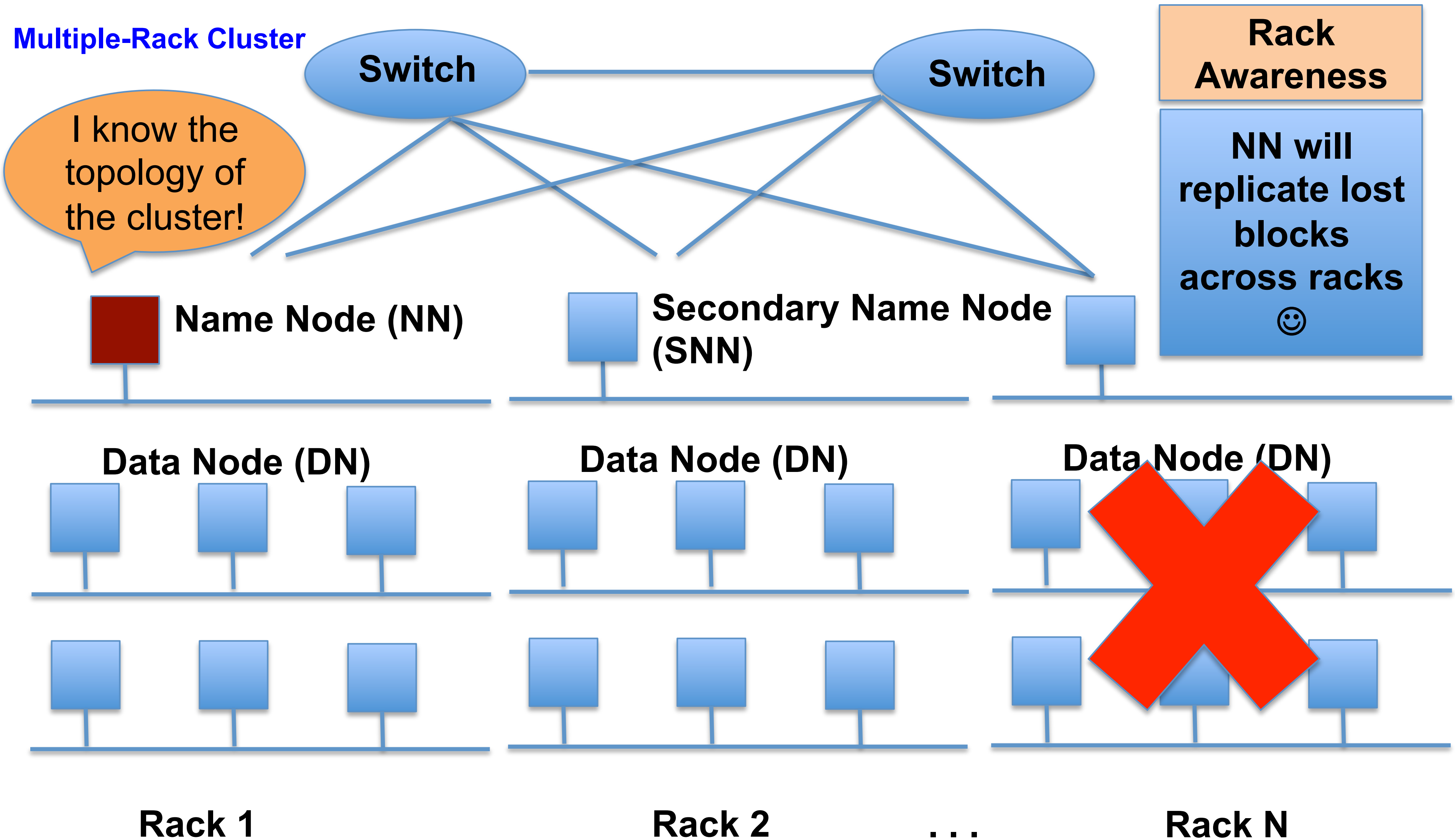
Data Nodes



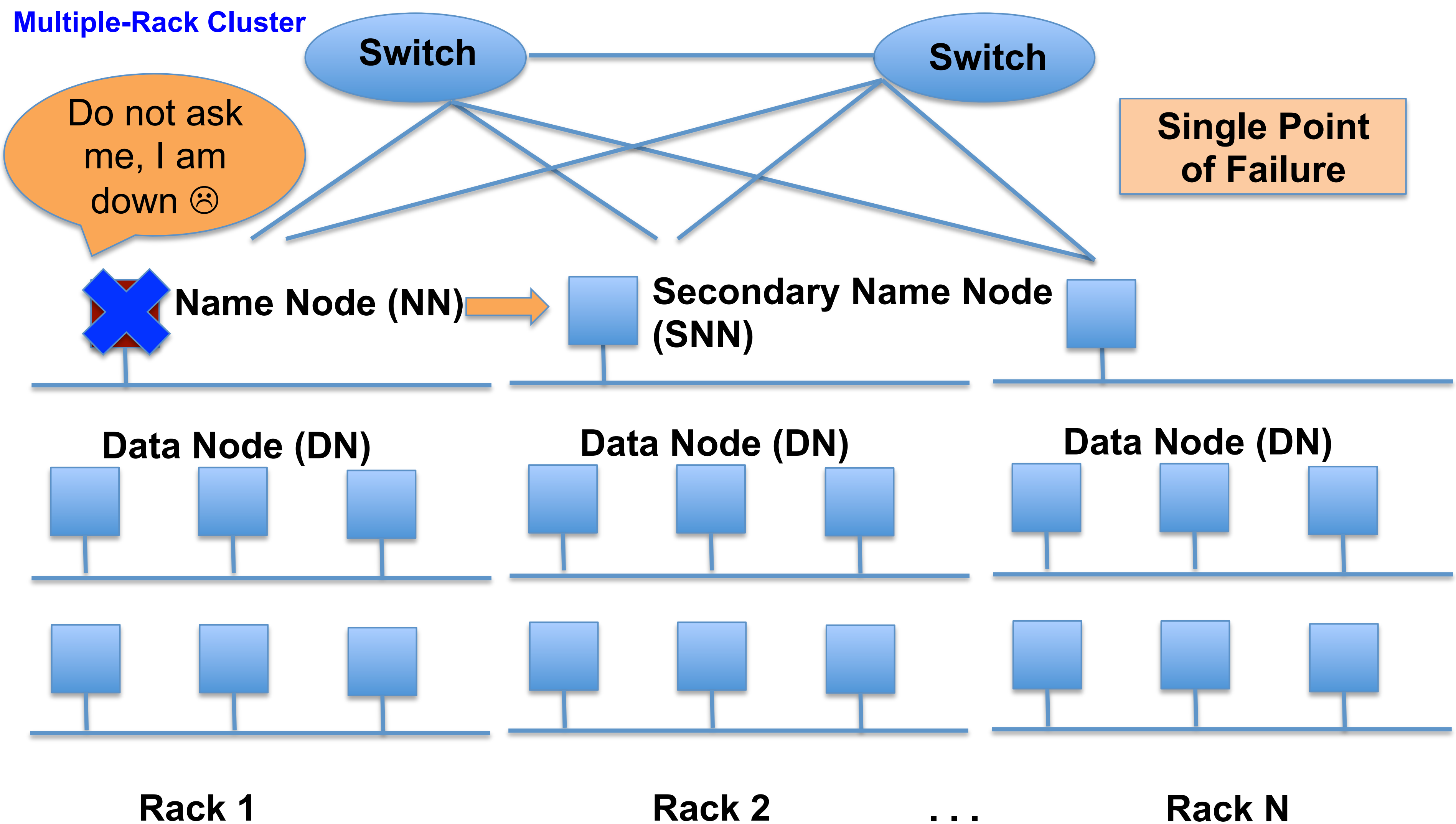
HDFS architecture



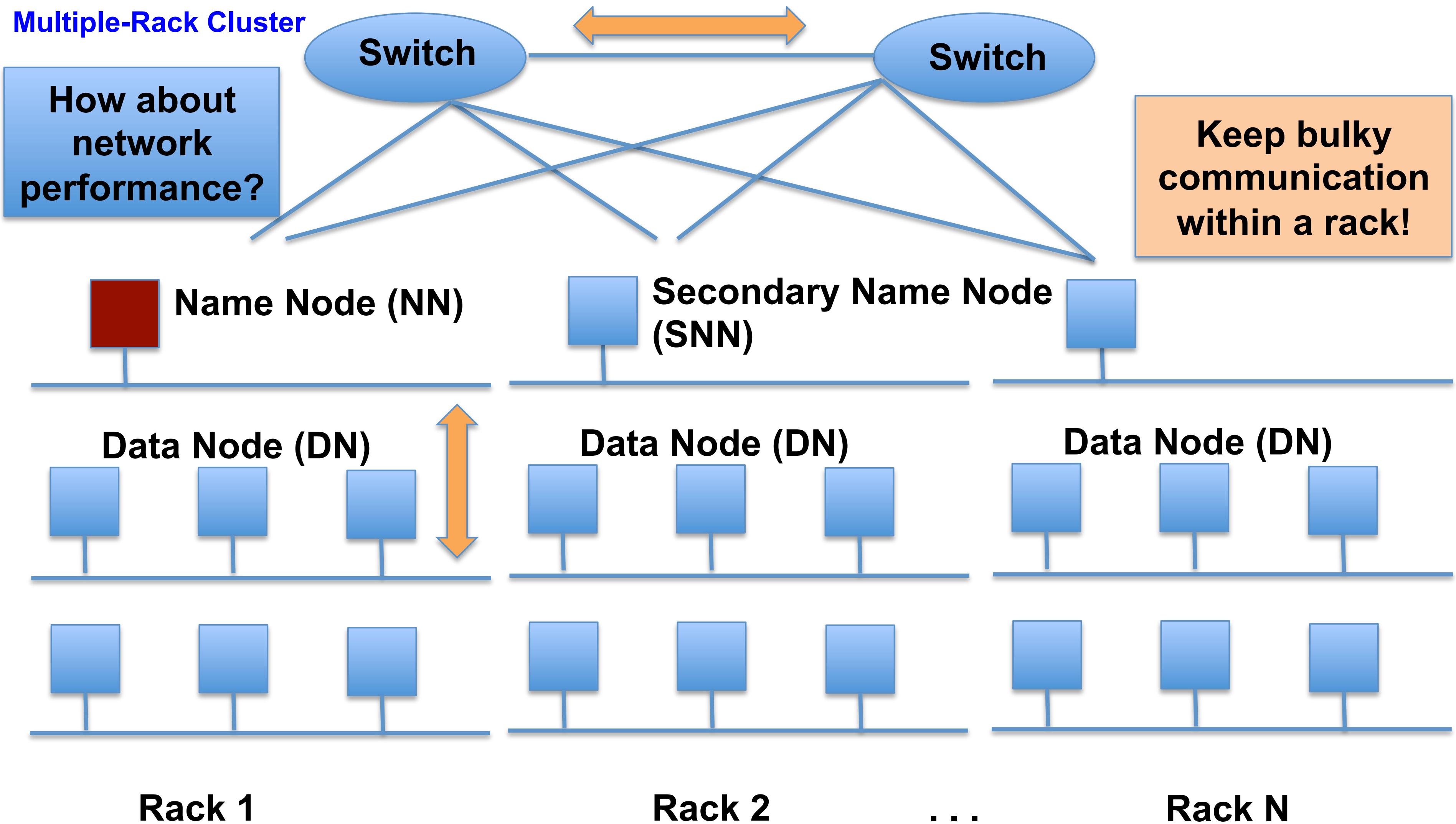
HDFS architecture



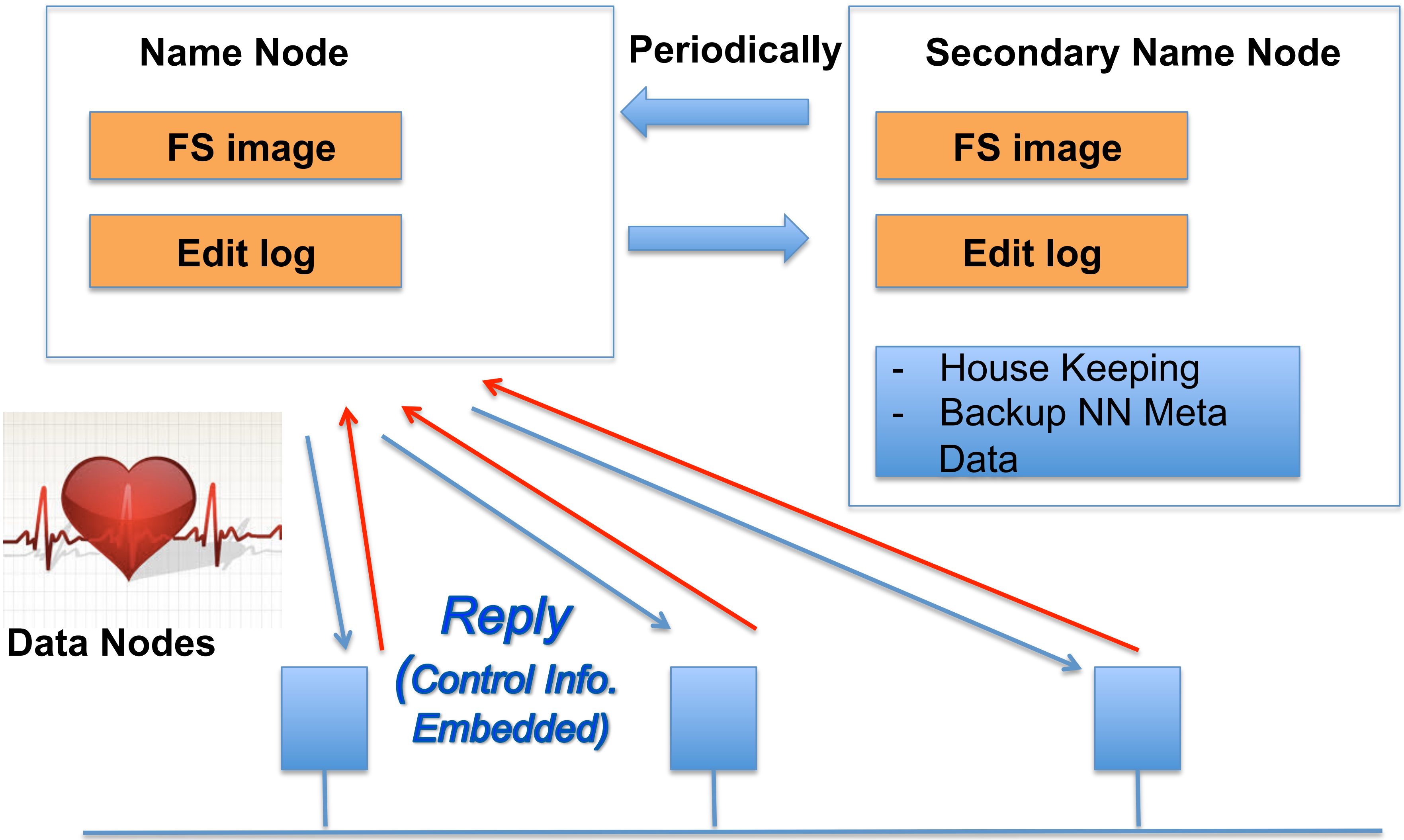
HDFS architecture



HDFS architecture



HDFS Inside: Name Node



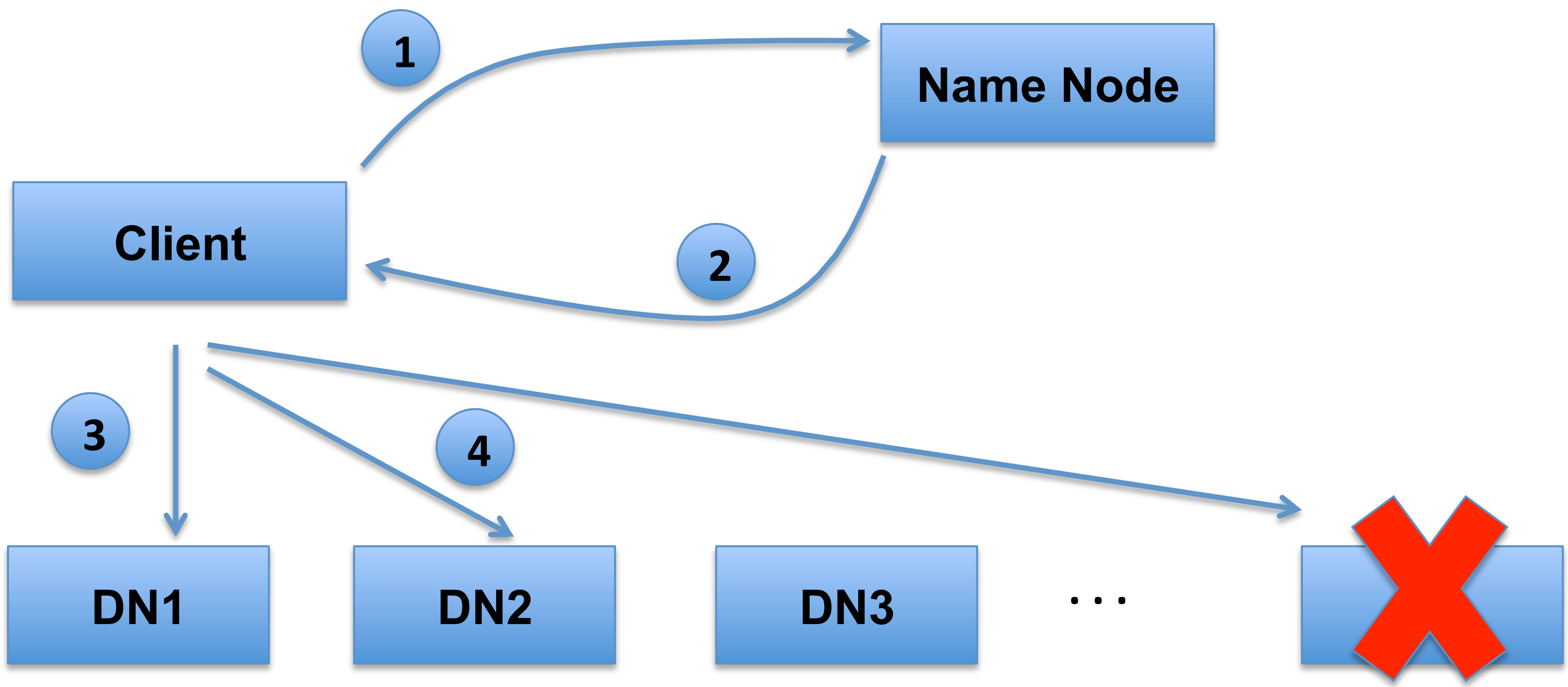
HDFS Inside: Blocks

- Q: 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?
- Reasons:
  - Minimize overhead: disk seek time is almost constant

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

- Q: 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

- Q: 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

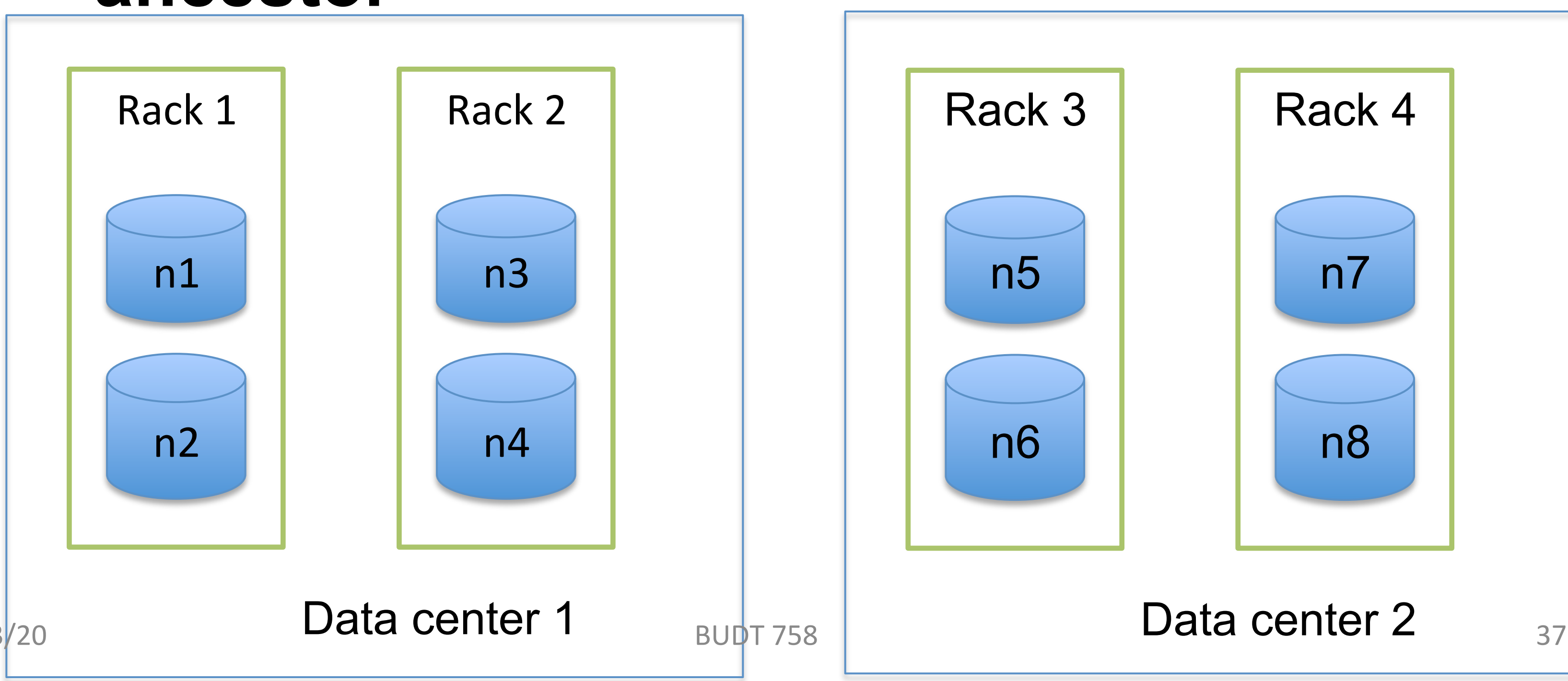
HDFS network topology

- 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 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 network topology

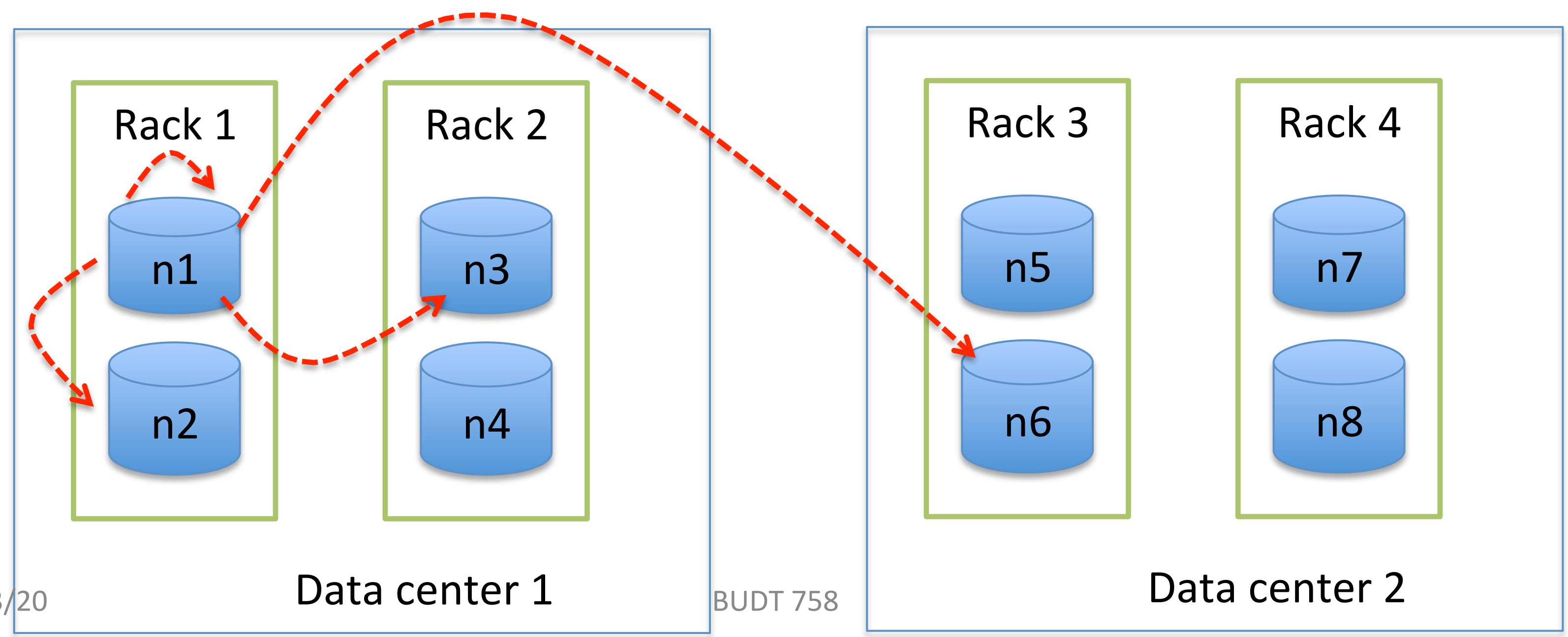
- What are the distance of the following pairs:

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

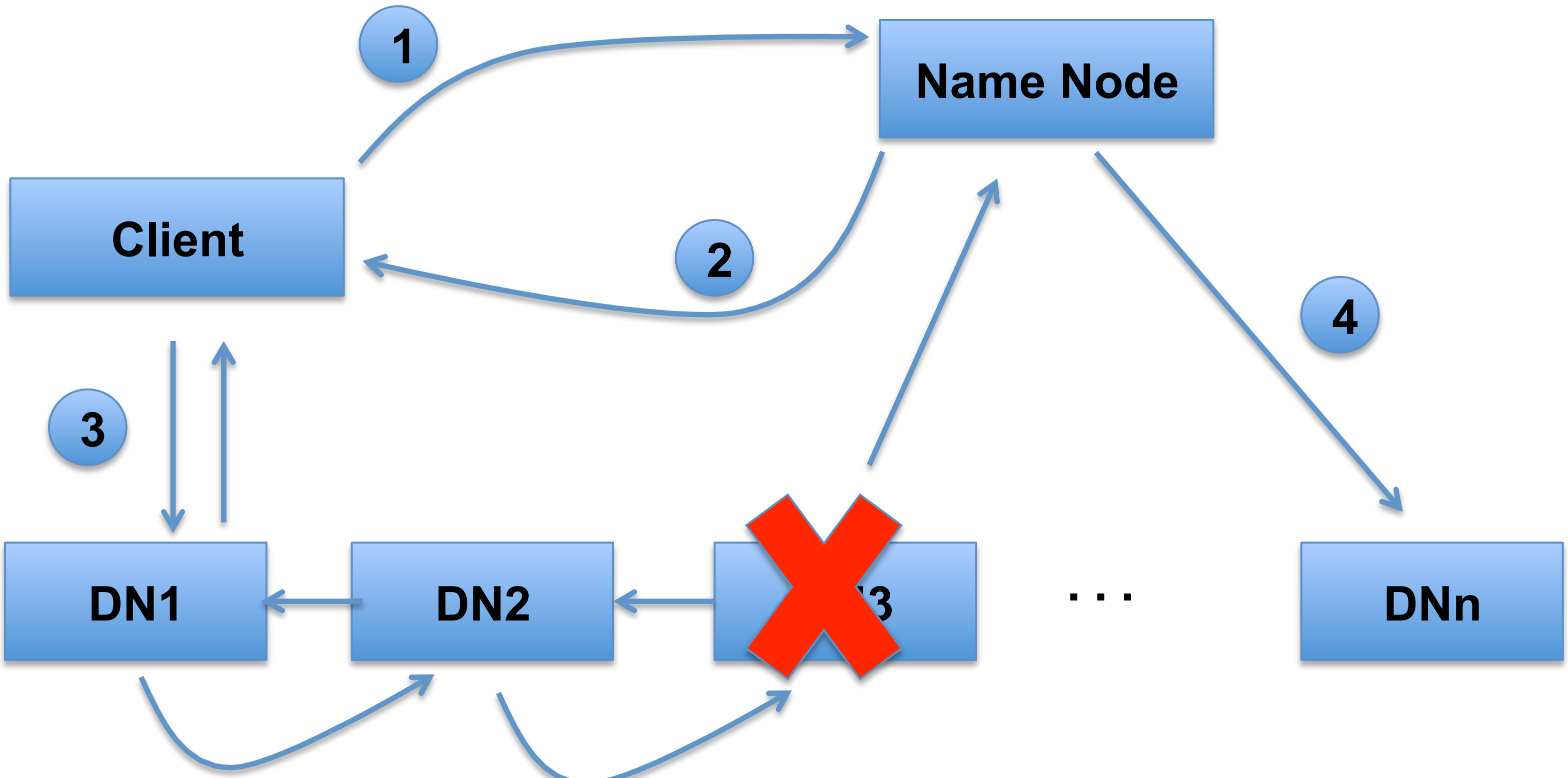
Dist(d1/r1/n1, d1/r1/n2)= **2**

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

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



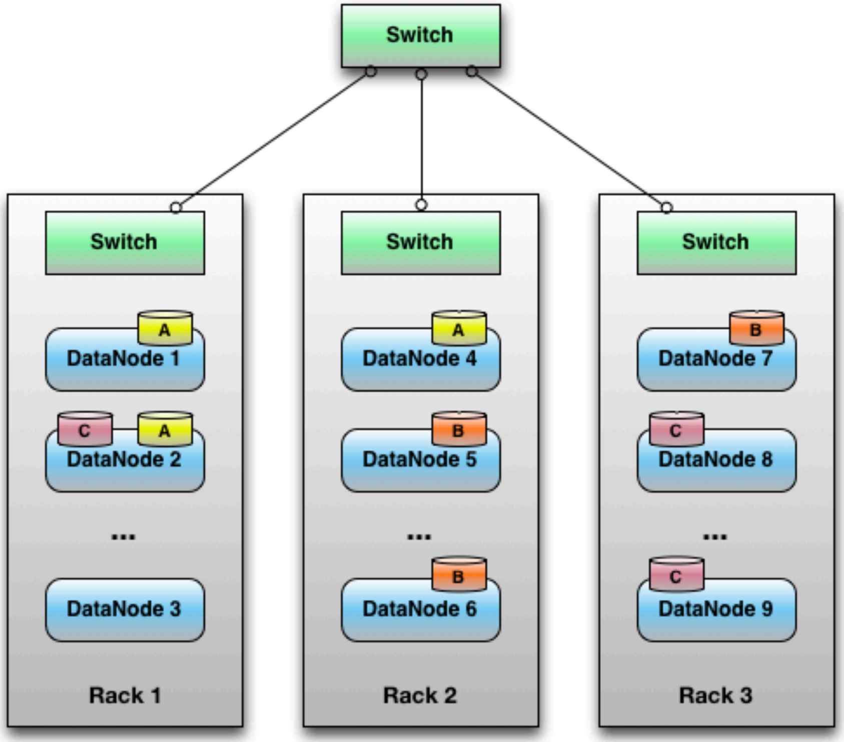
HDFS Inside: Write



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

Data replication

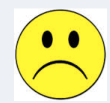
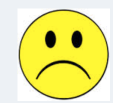
- **Frist copy** is written to the local node (write affinity).
- **Second copy** is written to a DataNode within a remote rack.
- **Third copy** is written to a DataNode in the same remote rack.
- **Additional** replicas are randomly placed.



Objectives: load balancing, fast access, fault tolerance.

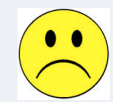
HDFS Inside: Write

- Replication Strategy vs Tradeoffs

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

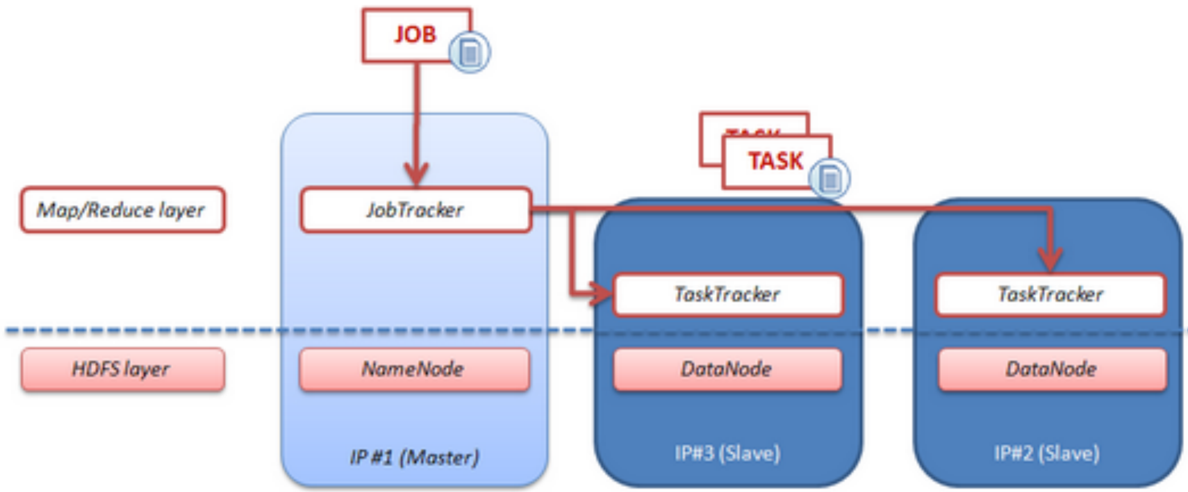
HDFS Inside: Write

- 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			

MapReduce: Hadoop execution layer

- **JobTracker** knows everything about submitted jobs
- Divides jobs into tasks and decides where to run each task
- Continuously communicating with TaskTracker



- **TaskTracker** execute task (multiple tasks per node)
- Monitors the execution of each task
- Continuously sending feedback to JobTracker