Case Study: OS Hot Topics

Instructor: Dr. Liting Hu

Big Data

- What is Big Data
  - A bunch of data?
  - A technical term?
  - An industry?
  - A trend?
  - A set of skills?
  - …?

What is Big Data?

- Wikipedia big data
  - An all-encompassing term for any collection of data sets so **large** and **complex** that it becomes **difficult** to process using on-hand data management tools or traditional data processing applications.

How big is big?

- 2008: Google processes 20 PB a day
- 2009: Facebook has 2.5 PB user data + 15 TB /day
- 2011: Yahoo! has 180-200 PB of data
- 2012: Facebook ingests 500 TB/day
- 2013: YouTube 1000 PB video storage; 4 billion views/day

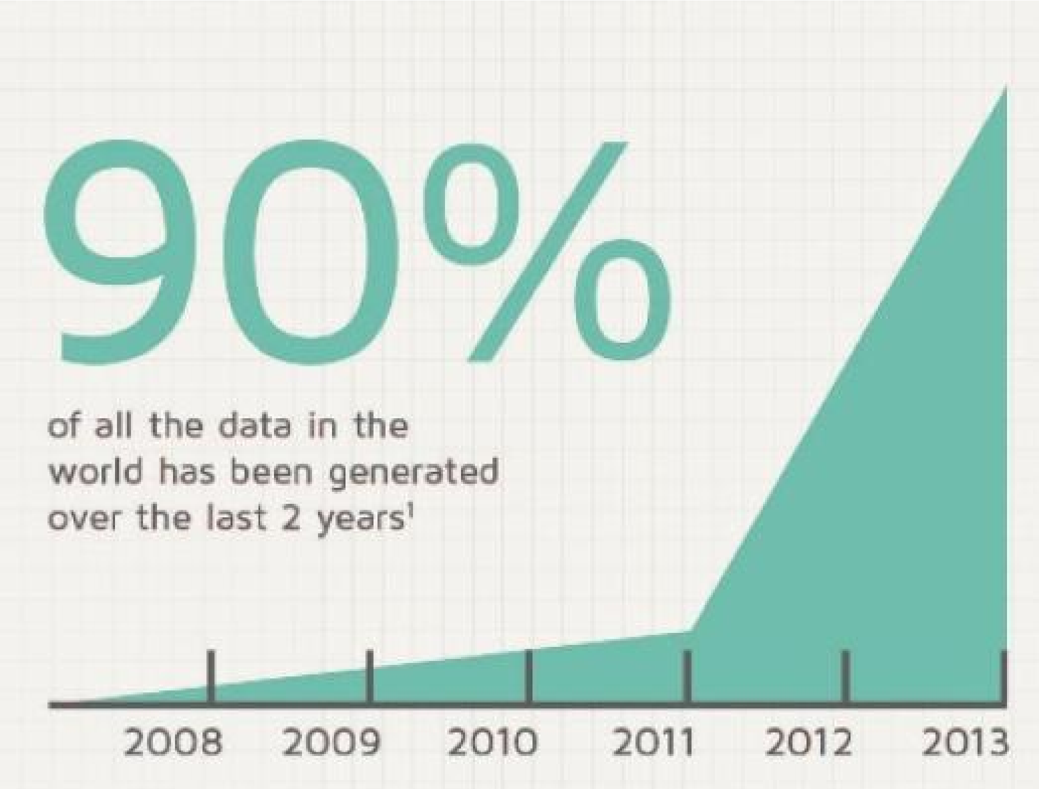
1 PB=10^3 TB=10^6 GB=10^15 B

1 Exabyte = 1000 PB

Zettabyte, Yottabyte …

How big is big?

- The percentage of all data in the word that has been generated in **last 2 years**?



Who is generating Big Data?

Social

Twitter for Business

Small business advertisers, learn how to get $100 in free Twitter advertising from American Express.

Learn the Basics

What is Twitter?
Twitter Glossary
Best Practices
Twitter on the Go

Optimize

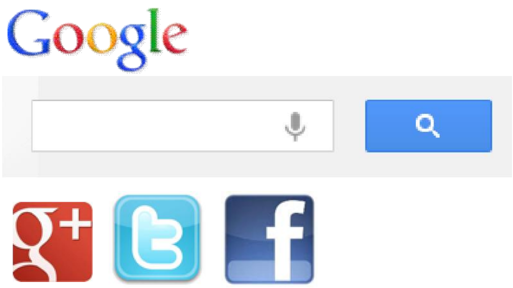
User Tracking & Engagement

Homeland Security

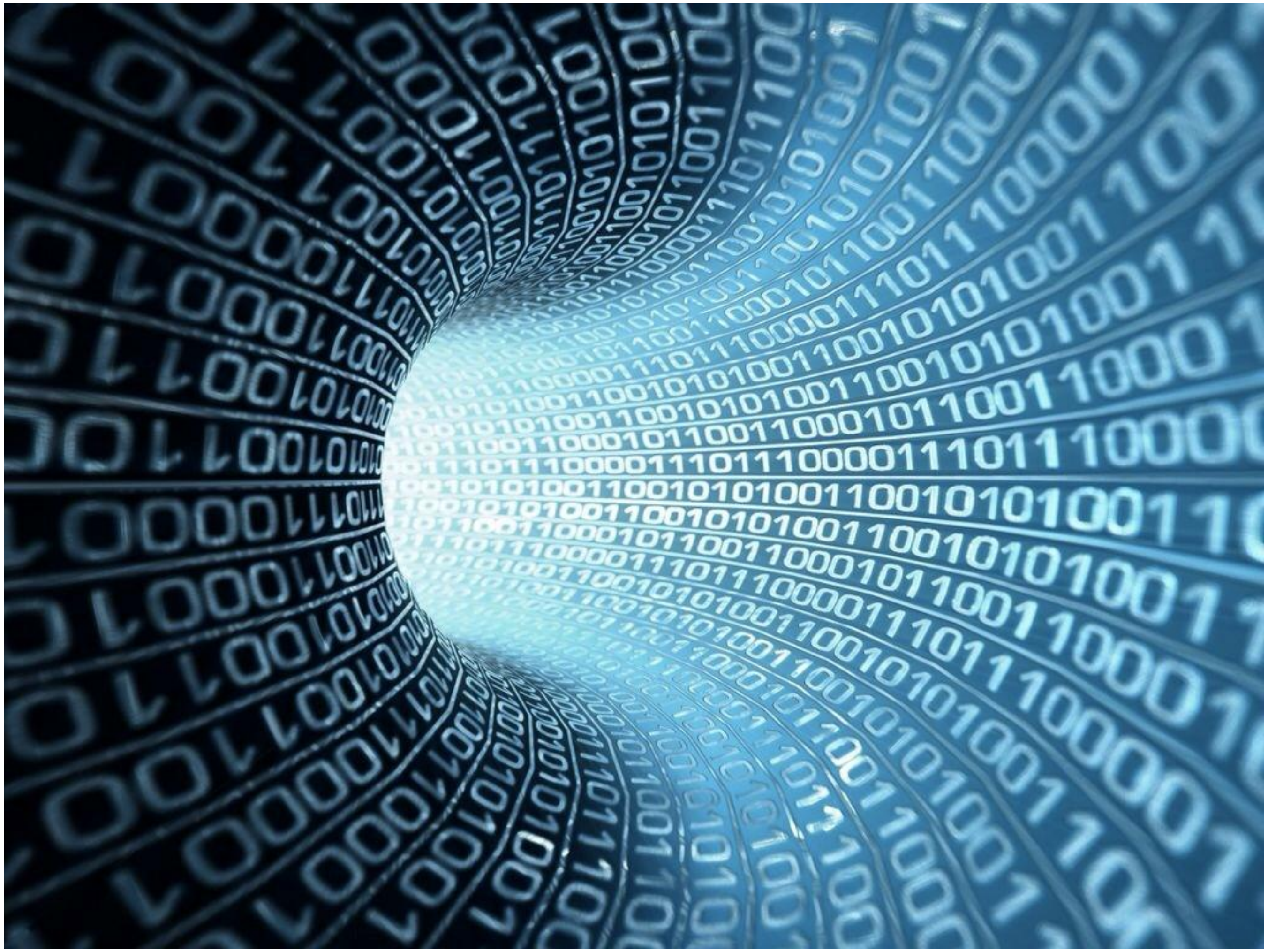
eCommerce

Financial Services

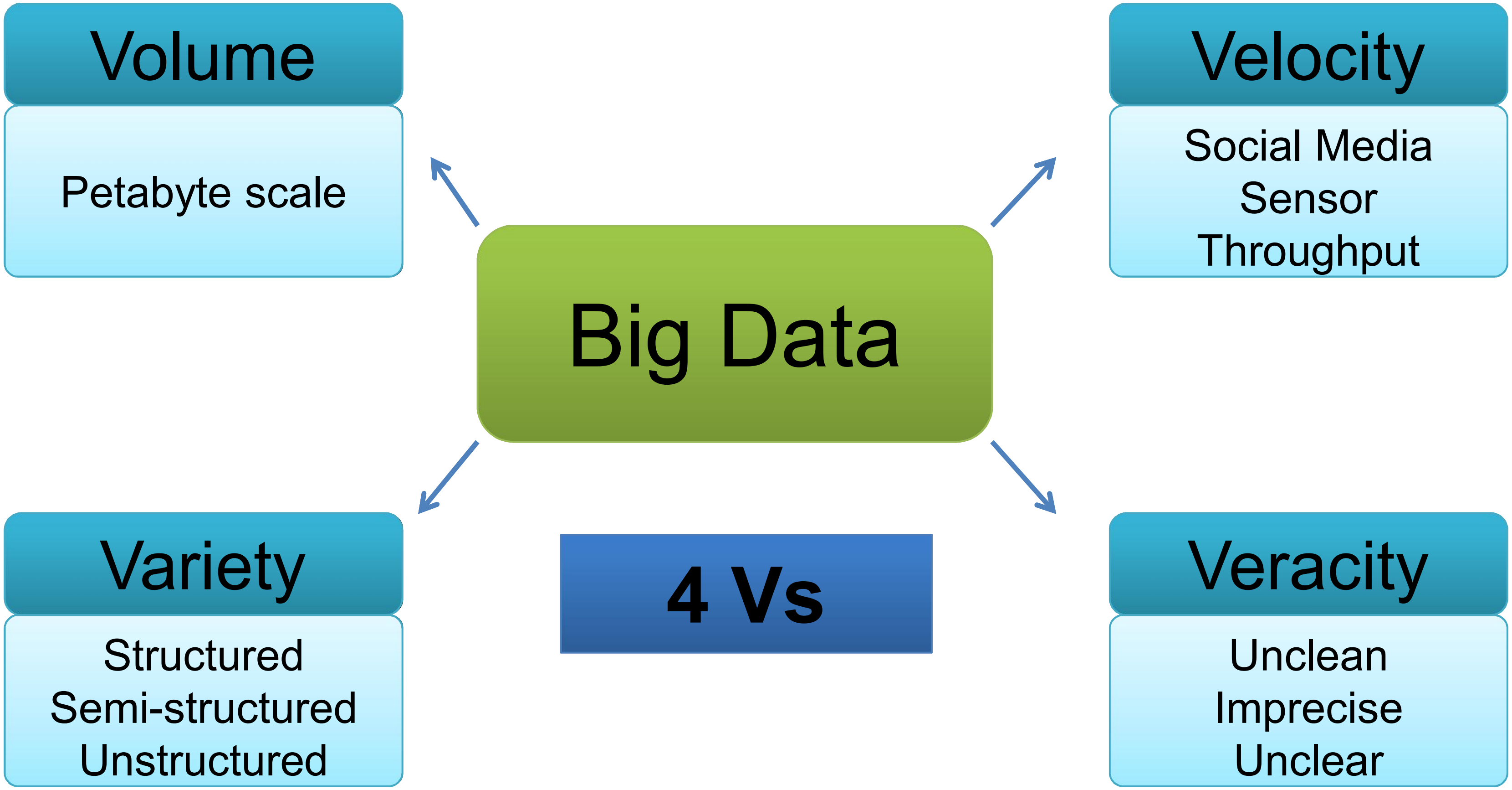
Real Time Search



That is a lot of data …



What are Key Features of Big Data?



Big Data

Volume

Petabyte scale

Velocity

Social Media
Sensor
Throughput

Variety

Structured
Semi-structured
Unstructured

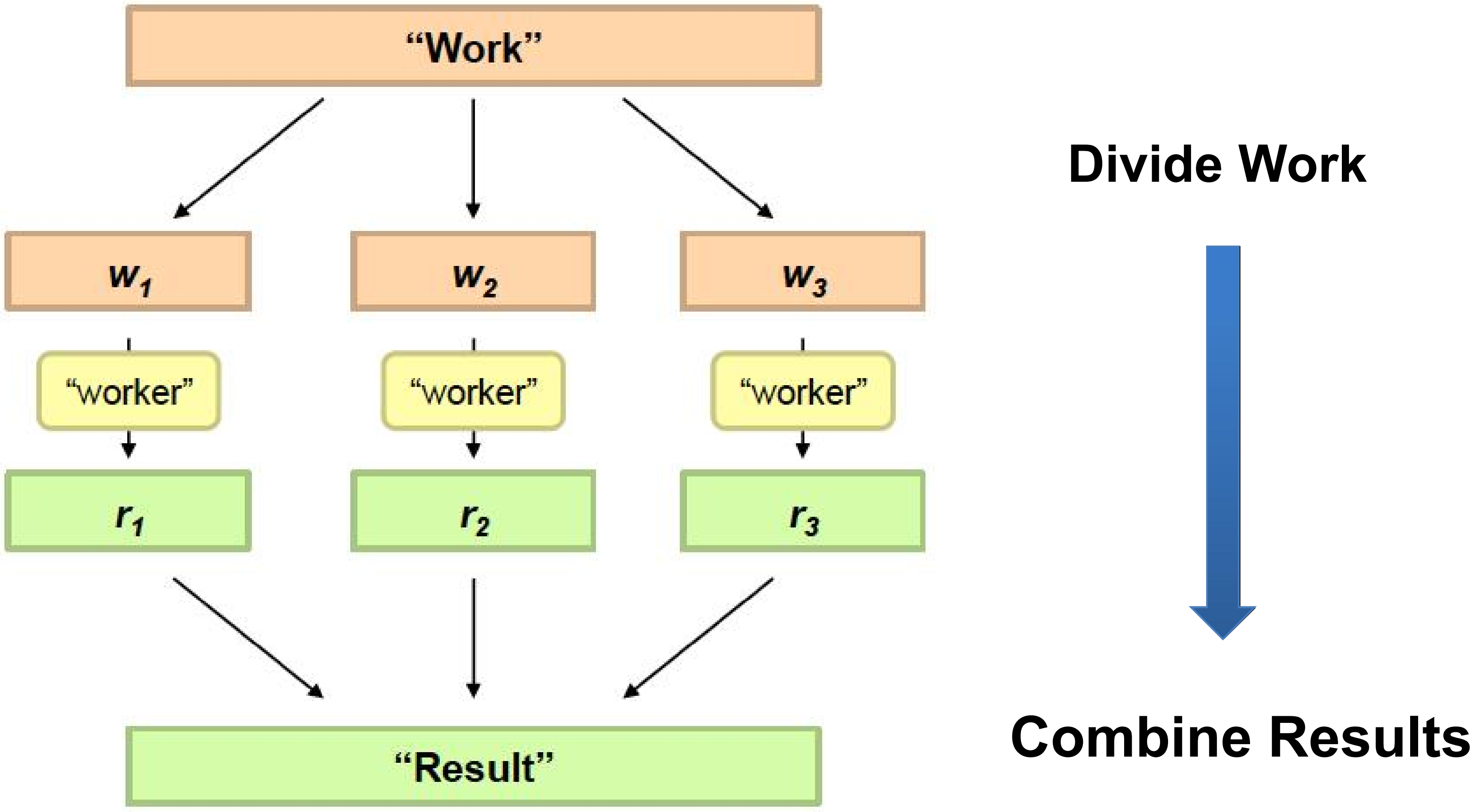
Veracity

Unclean
Imprecise
Unclear

4 Vs

Philosophy to Scale for Big Data?

Divide and Conquer



"Work"

w_1 w_2 w_3

"worker" "worker" "worker"

r_1 r_2 r_3

"Result"

Divide Work

Combine Results

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?

Image courtesy of Master isolated images at FreeDigitalPhotos.net

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! (Credit: Yahoo)

Hadoop offers

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

HDFS: Hadoop Distributed File System

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 real world example of New York Times

- **Goal:** Make entire archive of articles available online: 11 million, from 1851
- **Task:** Translate 4 TB TIFF images to PDF files
- **Solution:** Used Amazon Elastic Compute Cloud (EC2) and Simple Storage System (S3)
- **Time:** ?
- **Costs:** ? $

The
New York
Times

A real world example of New York Times

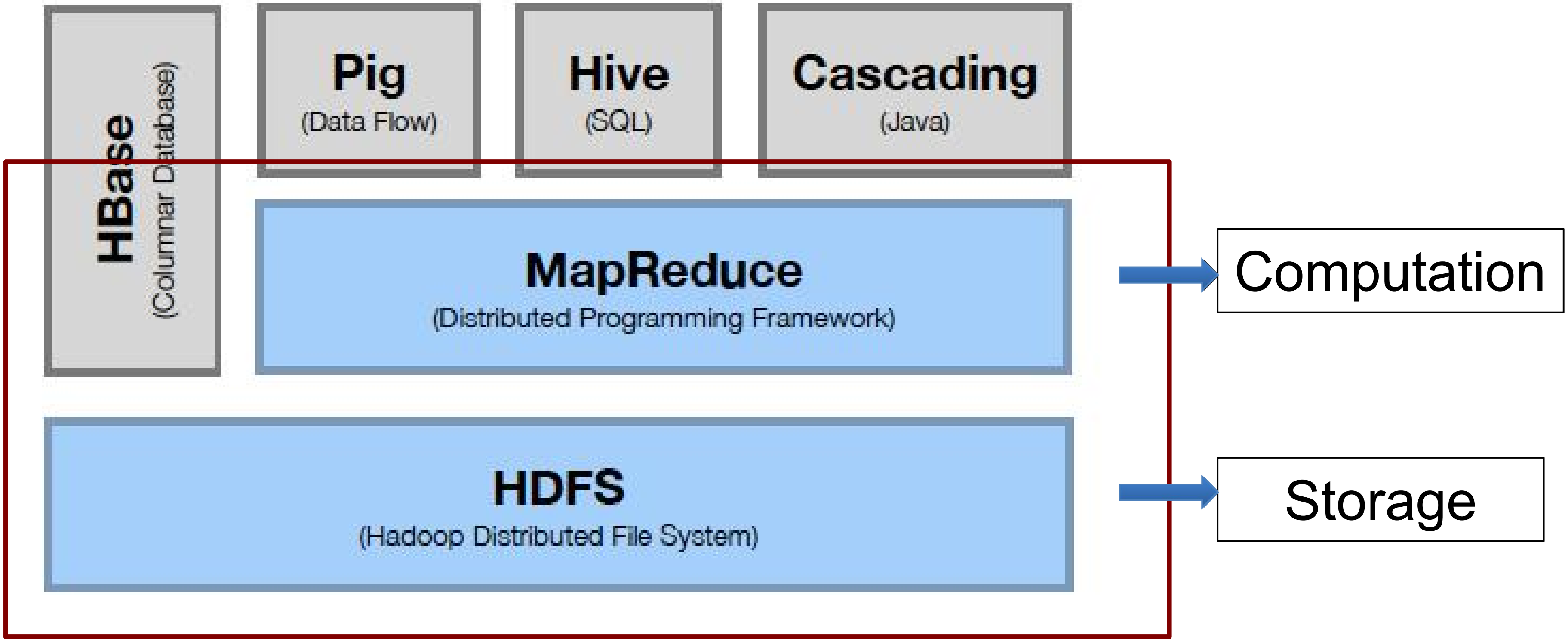
- **Goal:** Make entire archive of articles available online: 11 million, from 1851
- **Task:** Translate 4 TB TIFF images to PDF files
- **Solution:** Used Amazon Elastic Compute Cloud (EC2) and Simple Storage System (S3)
- **Time:** **< 24 hours**
- **Costs:** **$240**

So we are

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 Stack



HBase (Columnar Database)

Pig (Data Flow)

Hive (SQL)

Cascading (Java)

MapReduce (Distributed Programming Framework)

HDFS (Hadoop Distributed File System)

Computation

Storage

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/

- Hadoop Definitive Guide:

http://www.amazon.com/Hadoop-Definitive-Guide-Tom-White/dp/1449311520

HDFS Outline

- Motivation
- Architecture and Concepts
- Inside
- User Interface

Motivation Questions

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

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

Motivation Questions

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

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

Motivation Questions

- **Problem 3:** Commodity hardware will fail!

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

Motivation Questions

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

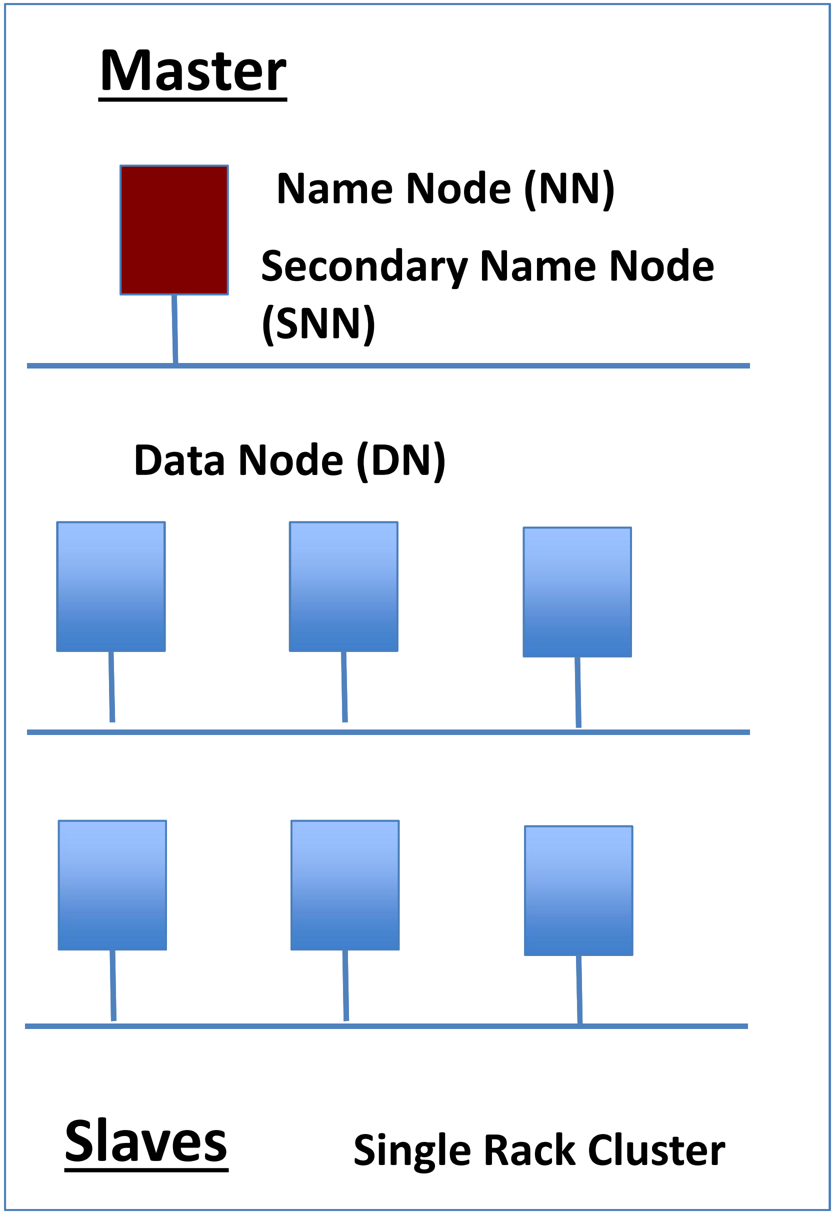
- **HDFS:** Replicate the data!

Motivation Questions

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

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

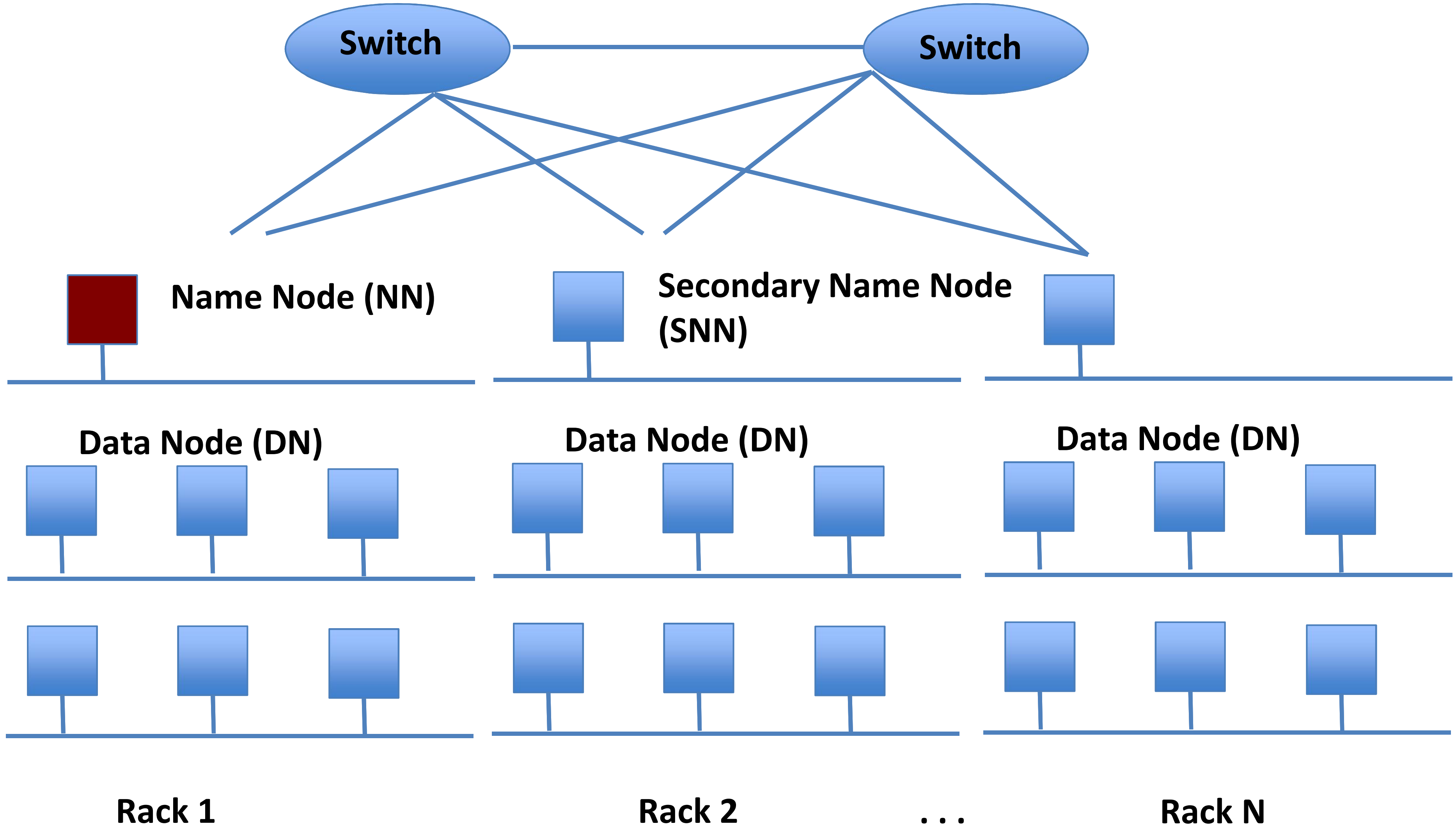
HDFS Architecture: Master-Slave



- Name Node: Controller
  - File System Name Space Management
  - Block Mappings
- Data Node: Work Horses
  - Block Operations
  - Replication
- Secondary Name Node:
  - Checkpoint node

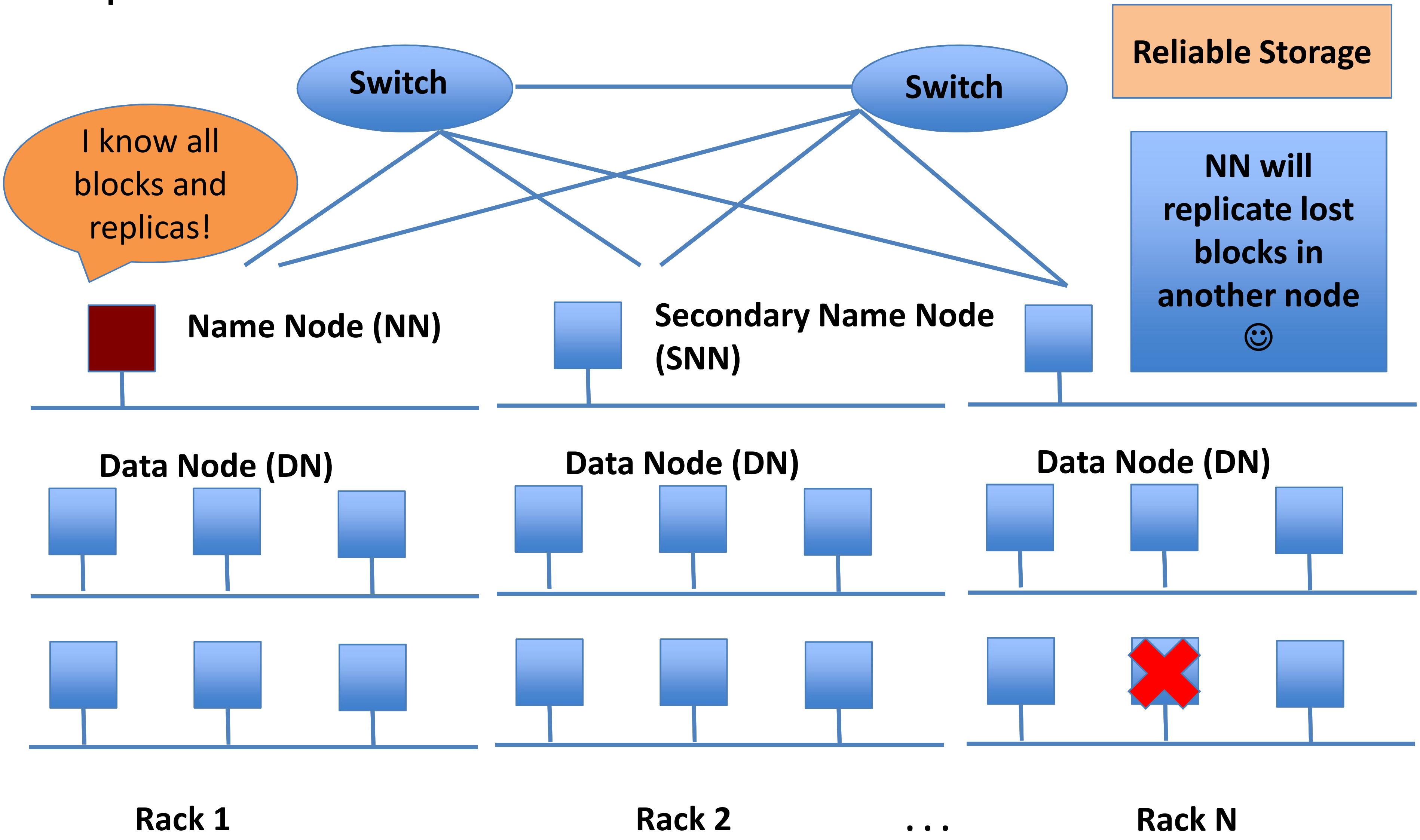
HDFS Architecture: Master-Slave

Multiple-Rack Cluster



HDFS Architecture: Master-Slave

Multiple-Rack Cluster



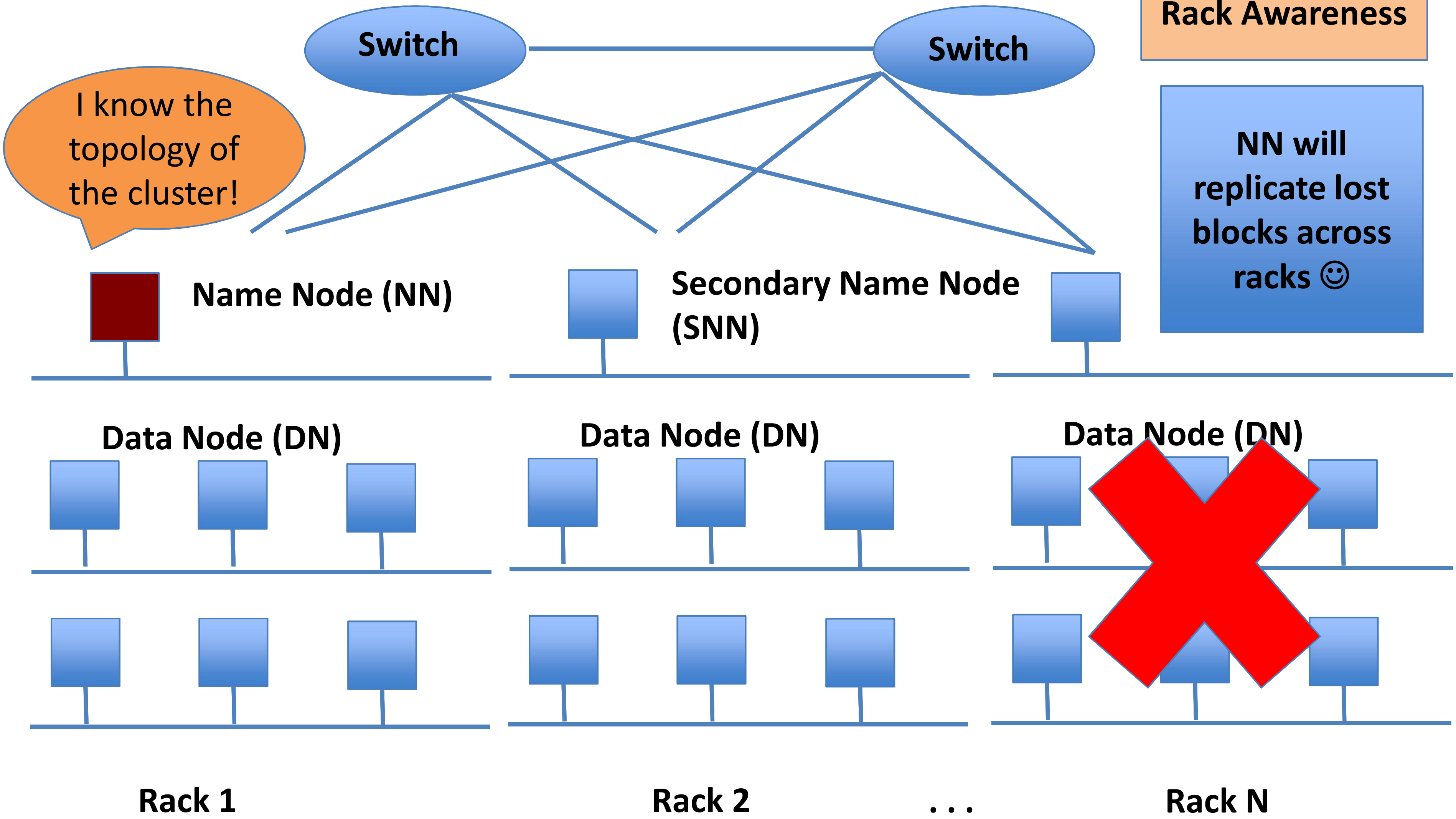
Reliable Storage

I know all blocks and replicas!

NN will replicate lost blocks in another node ☺

HDFS Architecture: Master-Slave

Multiple-Rack Cluster

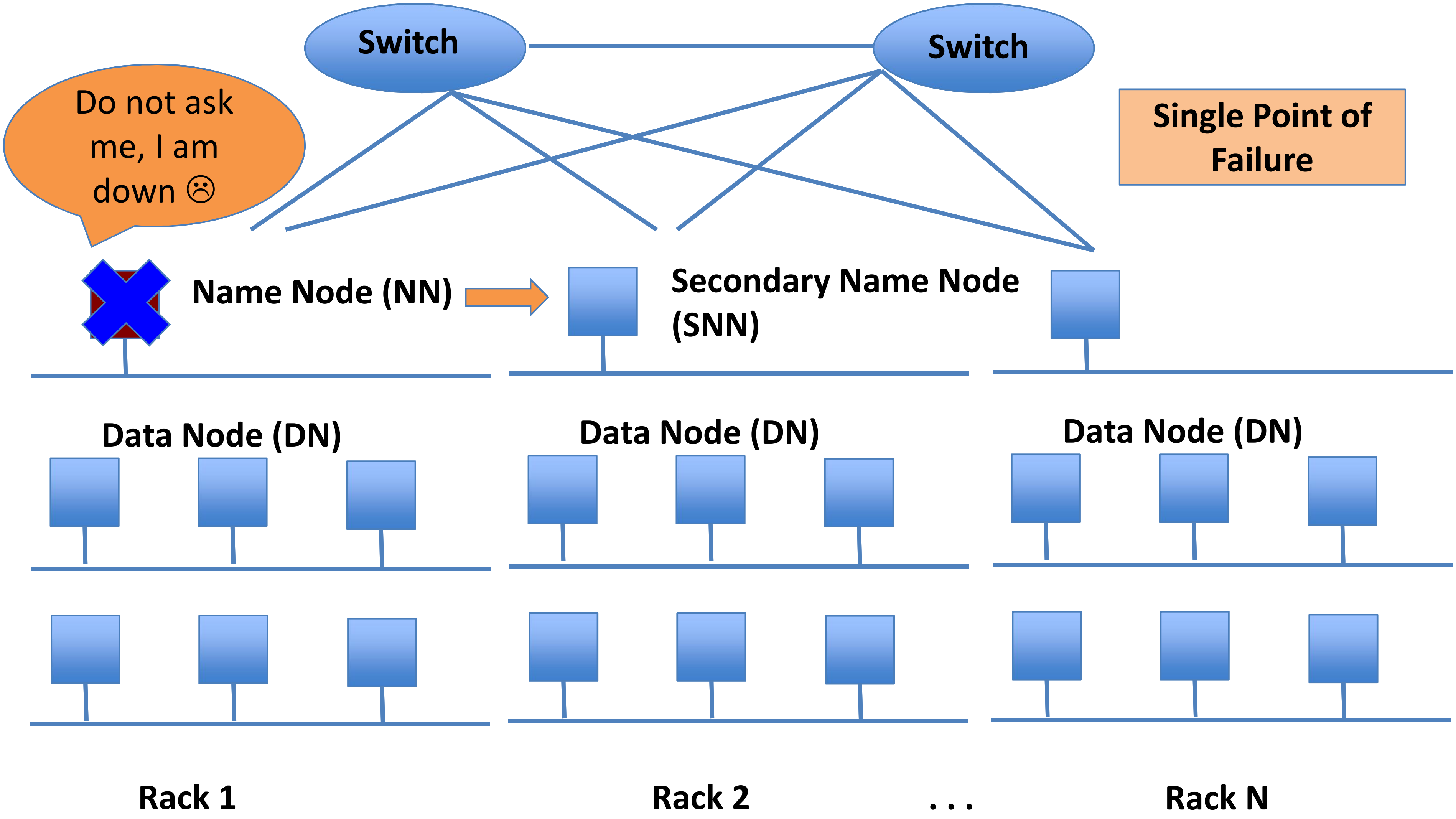


Rack Awareness

NN will replicate lost blocks across racks ☺

HDFS Architecture: Master-Slave

Multiple-Rack Cluster



Switch

Switch

Do not ask me, I am down ☹

Single Point of Failure

Name Node (NN)

Secondary Name Node (SNN)

Data Node (DN)

Data Node (DN)

Data Node (DN)

Rack 1

Rack 2

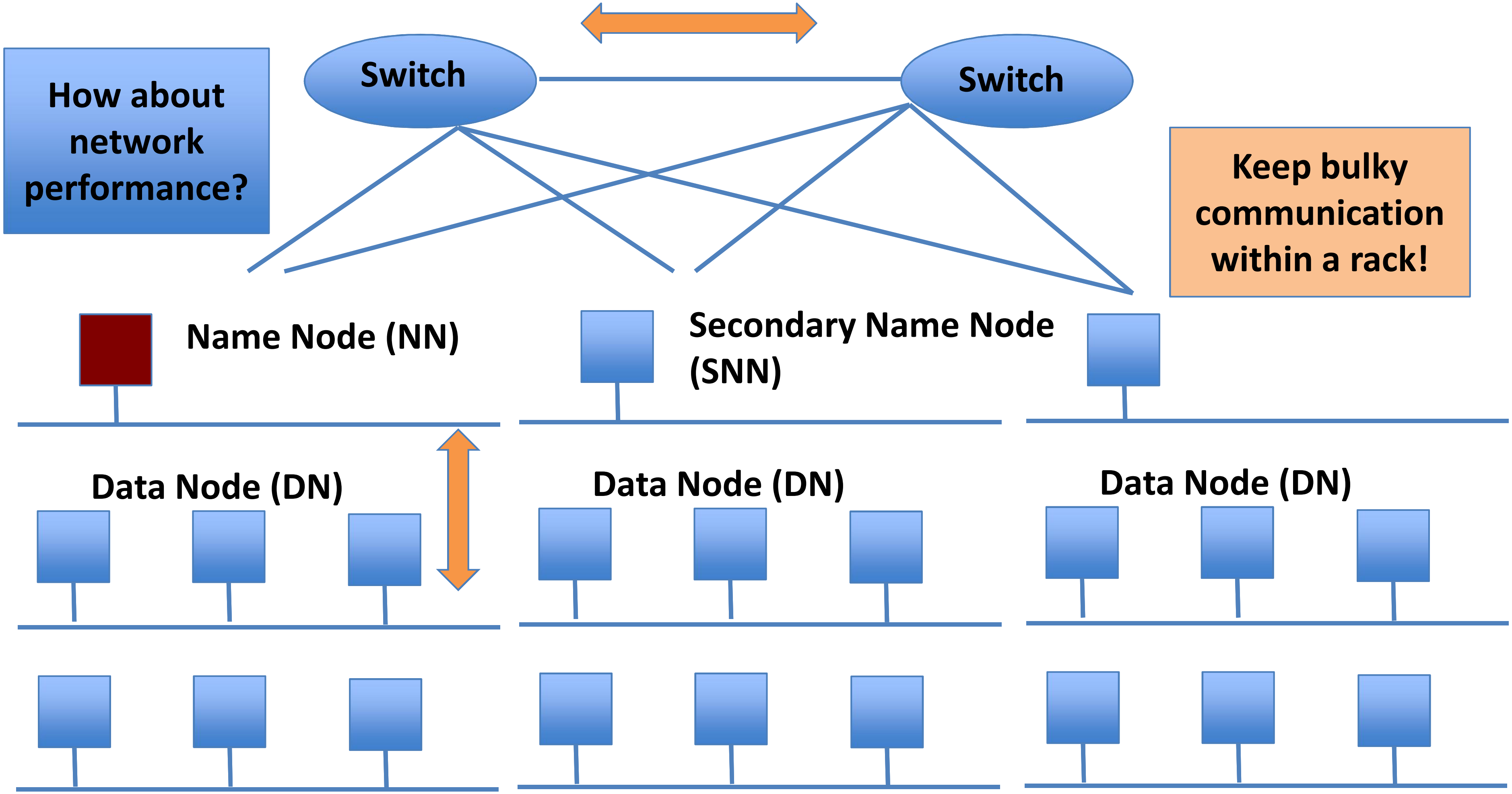
. . .

Rack N

HDFS Architecture: Master-Slave

Multiple-Rack Cluster

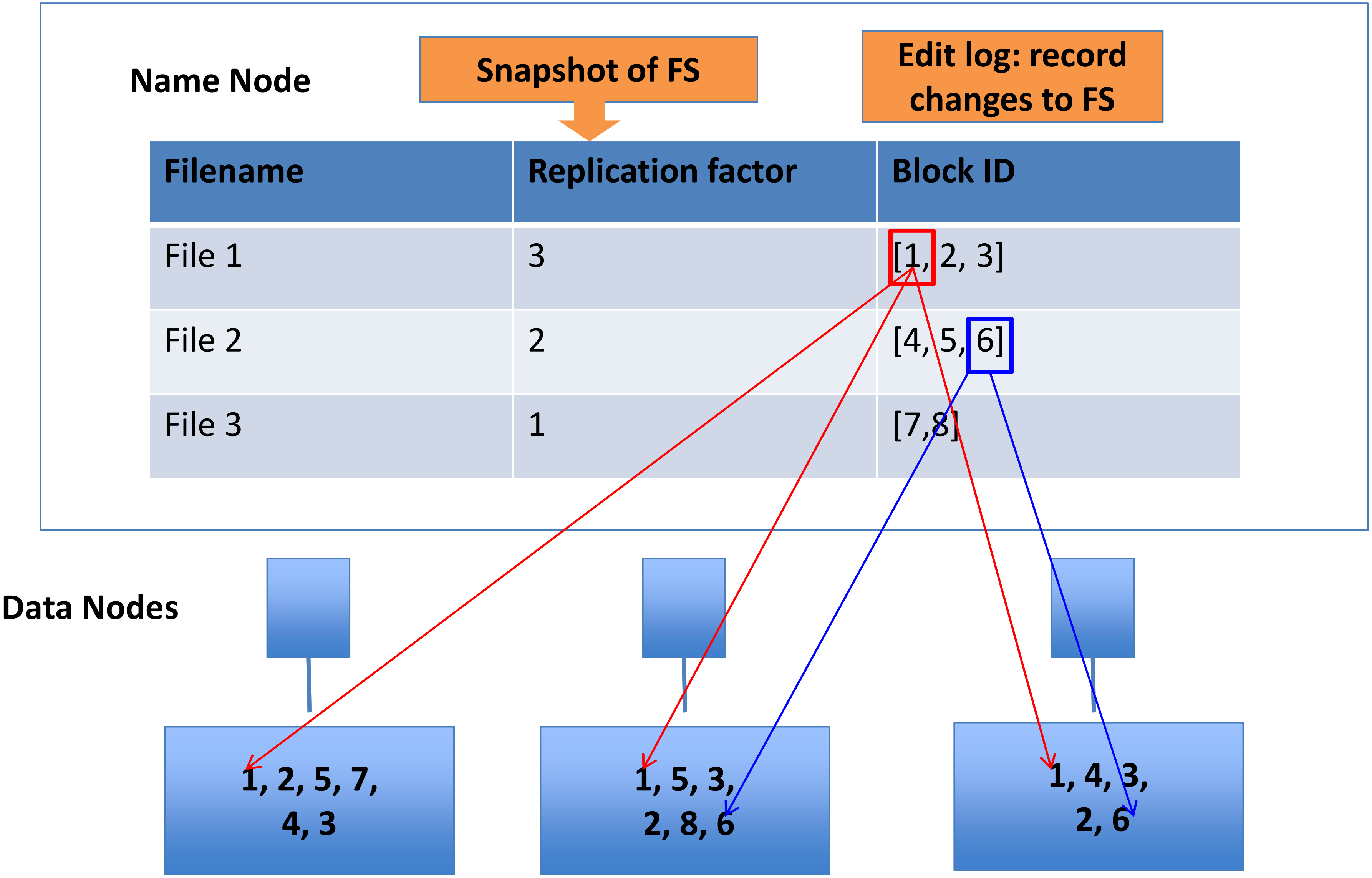
How about network performance?



Keep bulky communication within a rack!

Rack 1 Rack 2 . . . Rack N

HDFS Inside: 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]

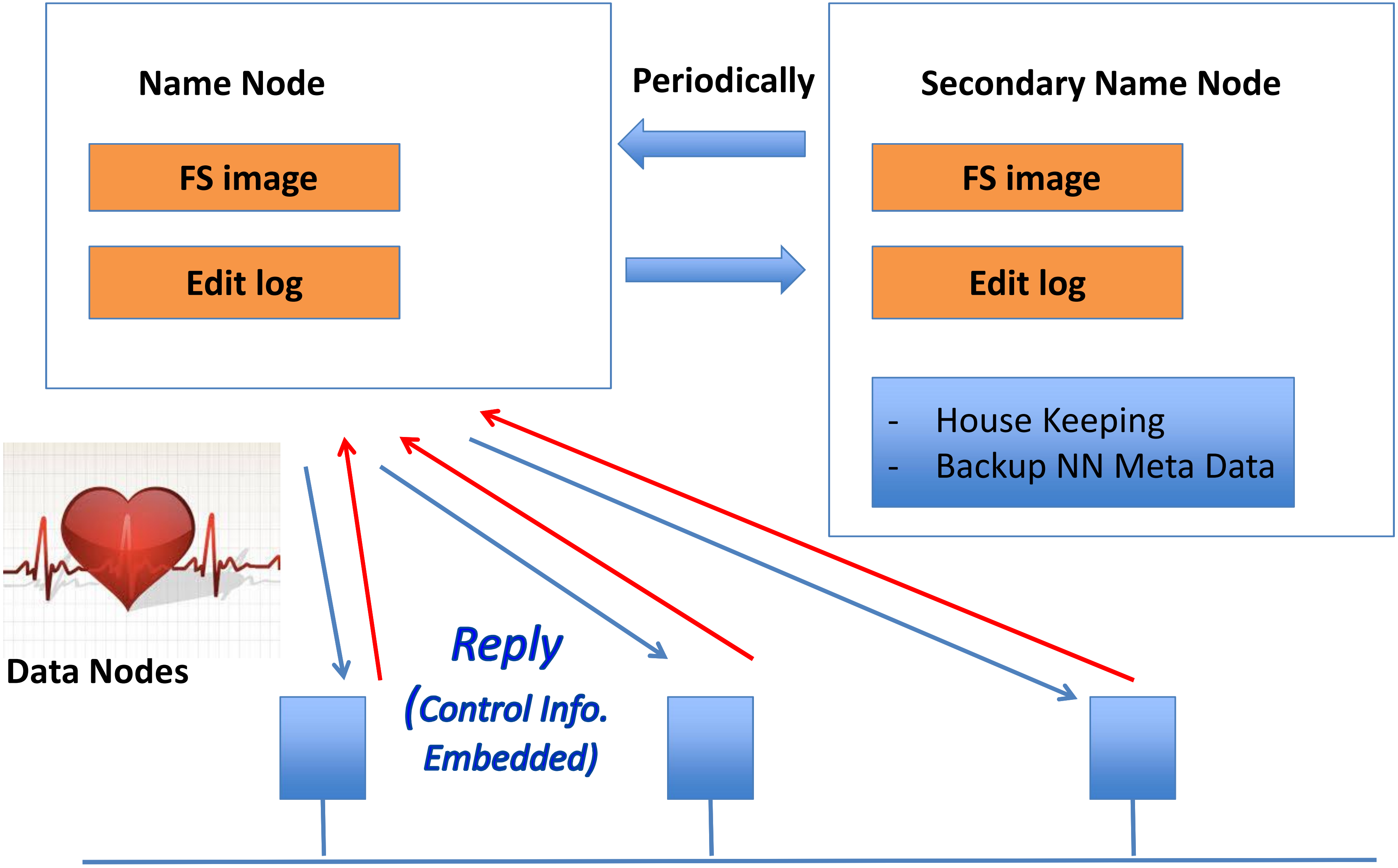
Data Nodes

1, 2, 5, 7, 4, 3

1, 5, 3, 2, 8, 6

1, 4, 3, 2, 6

HDFS Inside: Name Node



Name Node

FS image

Edit log

Periodically

Secondary Name Node

FS image

Edit log

- House Keeping
- Backup NN Meta Data

Data Nodes

Reply

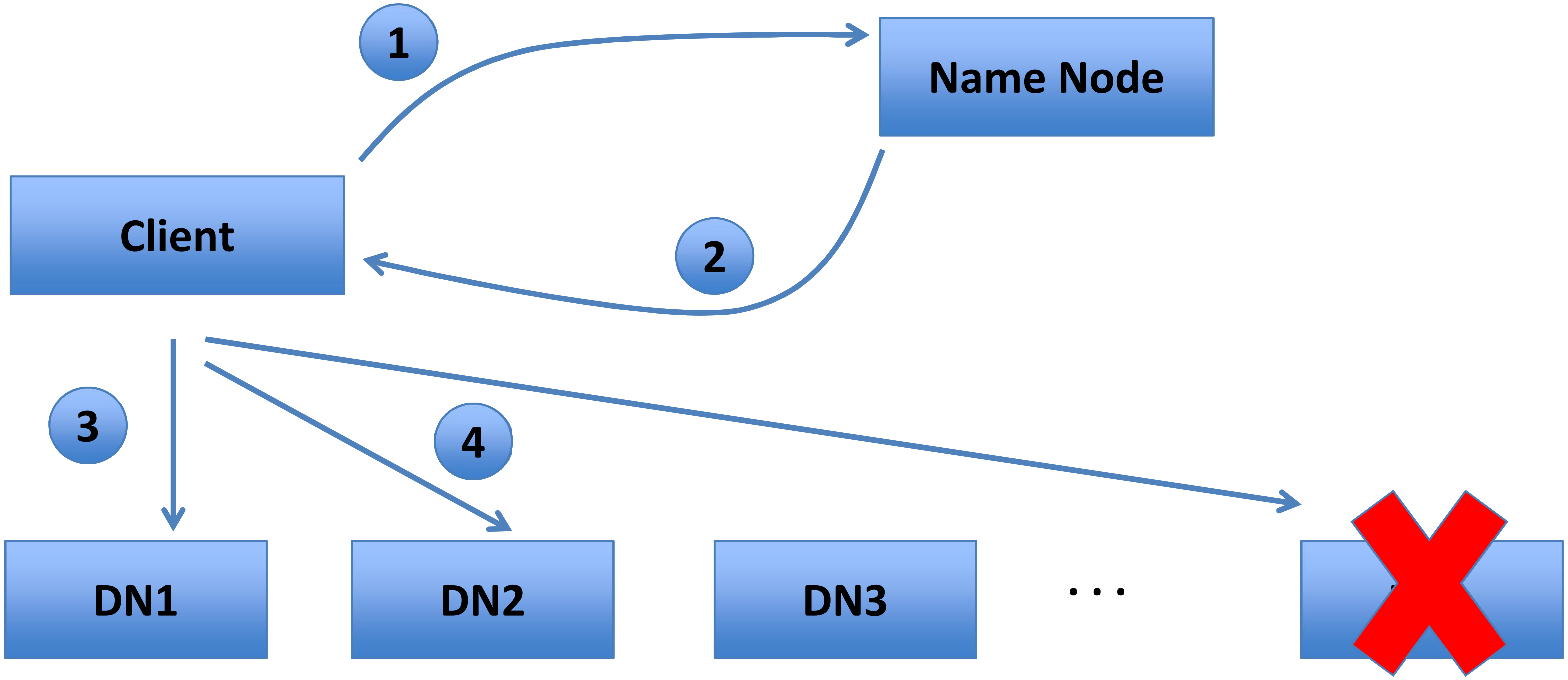
(Control Info. Embedded)

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: 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

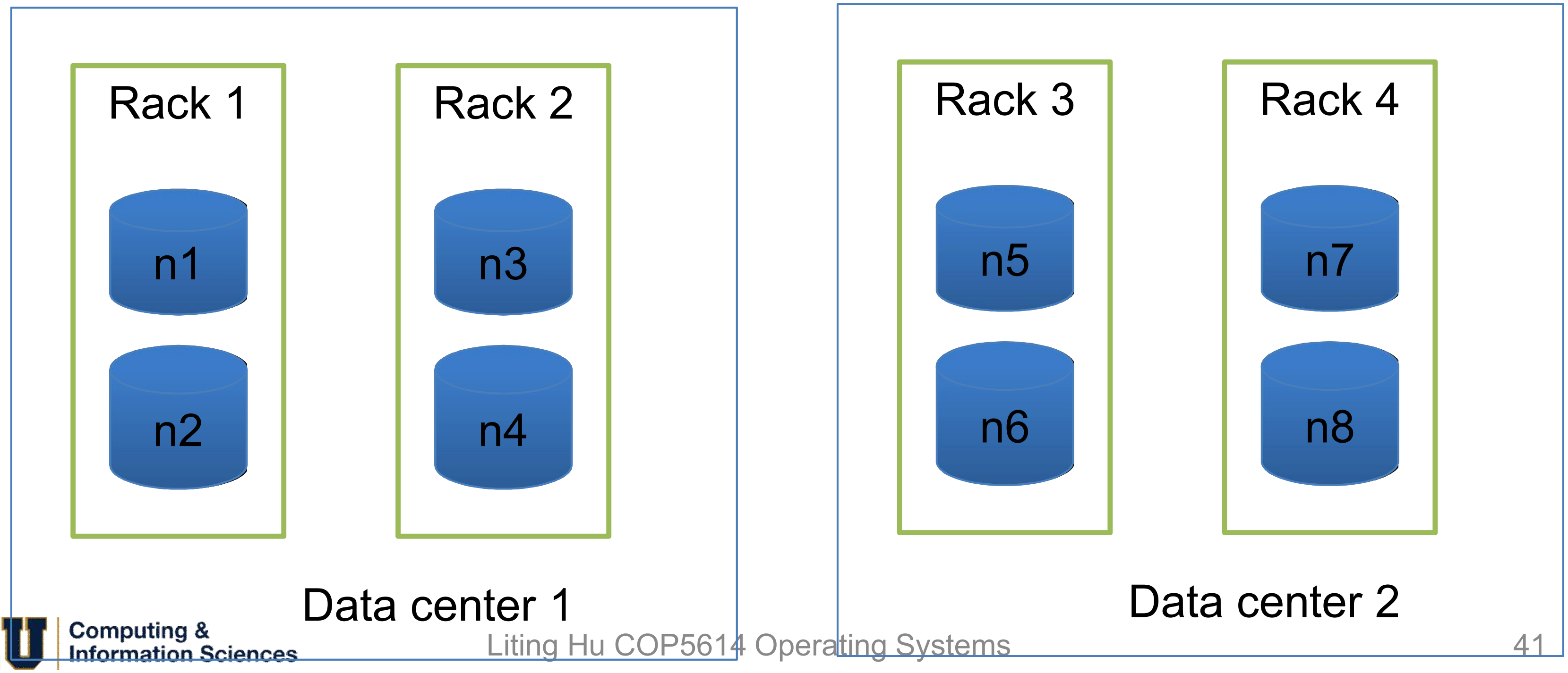
HDFS Inside: Network Topology

- 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**



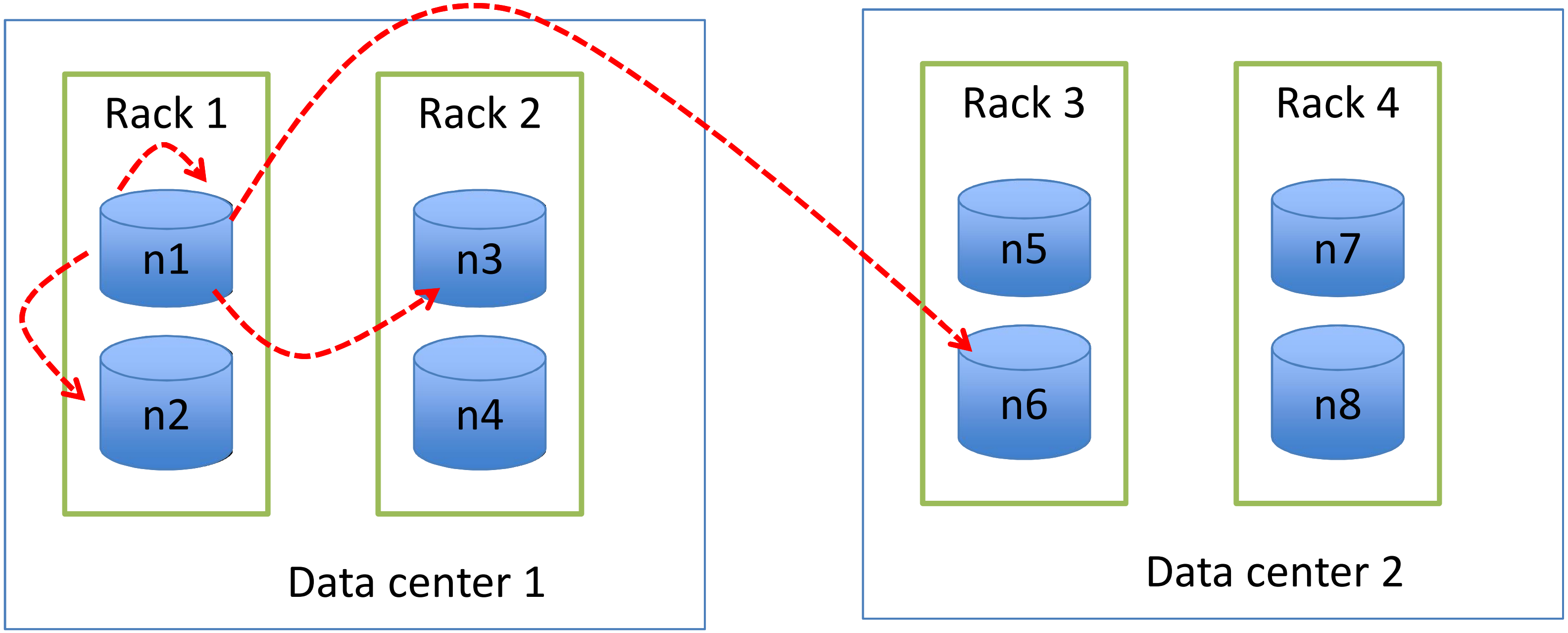
- What are the distance of the following pairs:

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

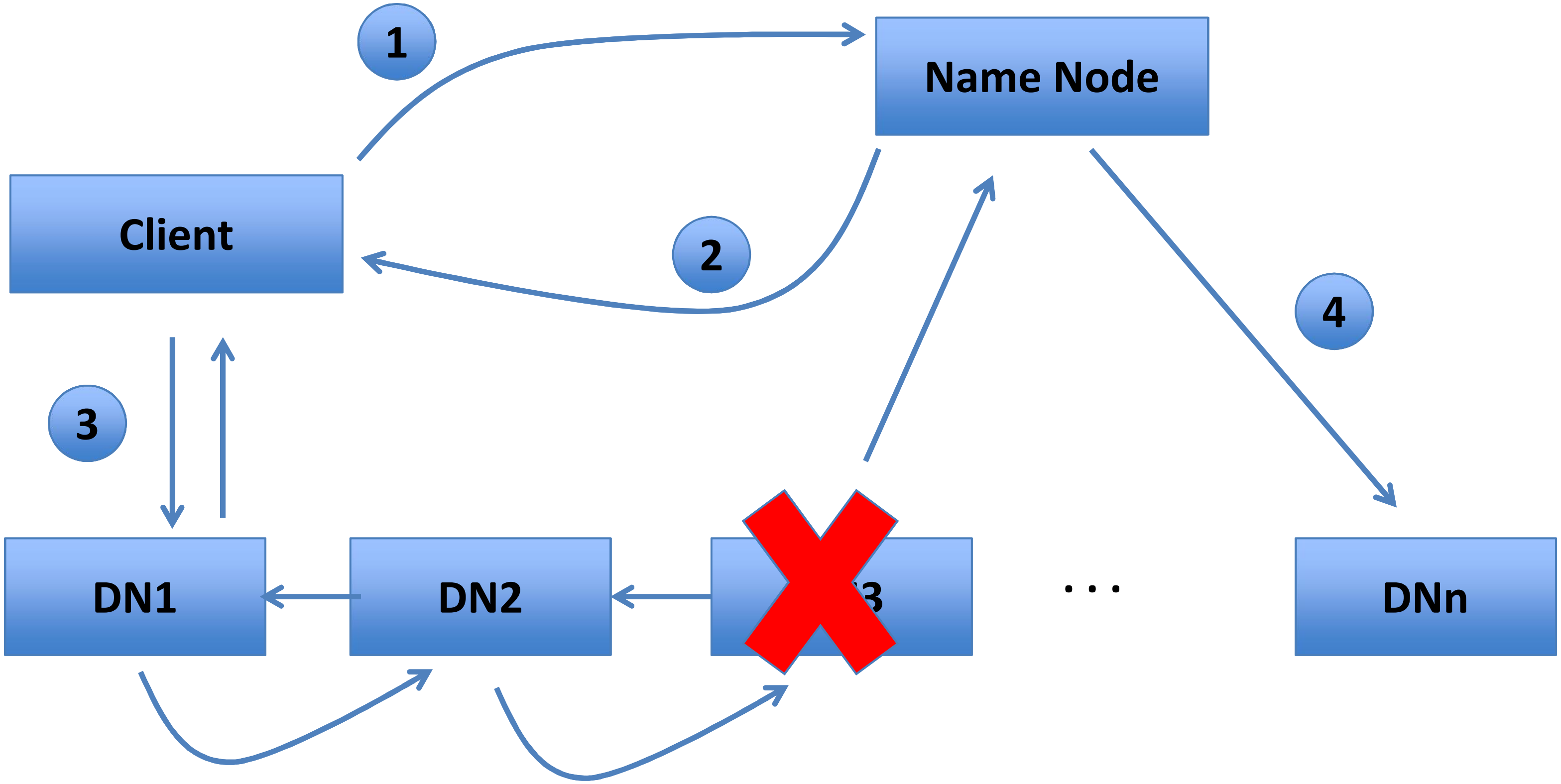
Dist(d1/r1/n1, d1/r1/n2)= **2**

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

- Q: Where should HDFS put the three replicas of a block? What tradeoffs we need to consider?
- Tradeoffs:
  - Reliability
  - Write Bandwidth
  - Read Bandwidth

Q: 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	☺	☹	☹

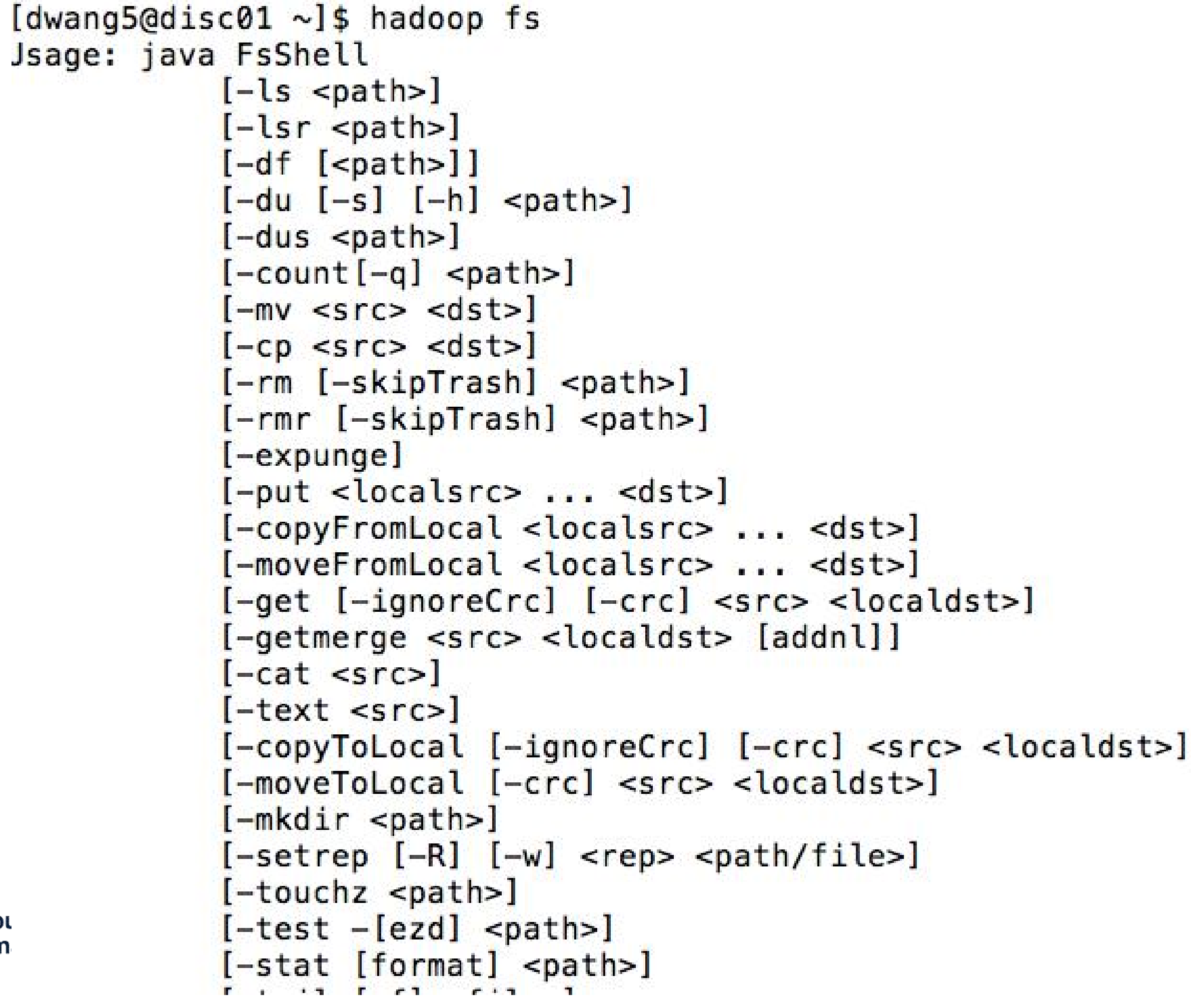
HDFS Inside: Write

- 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	☺	OK	OK

HDFS Command Line

- Hadoop Shell



```
[dwang5@disc01 ~]$ hadoop fs
Jsage: java FsShell
           [-ls <path>]
           [-lsr <path>]
           [-df [<path>]]
           [-du [-s] [-h] <path>]
           [-dus <path>]
           [-count[-q] <path>]
           [-mv <src> <dst>]
           [-cp <src> <dst>]
           [-rm [-skipTrash] <path>]
           [-rmr [-skipTrash] <path>]
           [-expunge]
           [-put <localsrc> ... <dst>]
           [-copyFromLocal <localsrc> ... <dst>]
           [-moveFromLocal <localsrc> ... <dst>]
           [-get [-ignoreCrc] [-crc] <src> <localdst>]
           [-getmerge <src> <localdst> [addnl]]
           [-cat <src>]
           [-text <src>]
           [-copyToLocal [-ignoreCrc] [-crc] <src> <localdst>]
           [-moveToLocal [-crc] <src> <localdst>]
           [-mkdir <path>]
           [-setrep [-R] [-w] <rep> <path/file>]
           [-touchz <path>]
           [-test -[ezd] <path>]
           [-stat [format] <path>]
```

Takeaways

- Big Data and Hadoop background
  - What and Why about Hadoop
  - 4 V challenge of Big Data

- Hadoop Distributed File System (HDFS)
  - Motivation: guide Hadoop design
  - Architecture: Single rack vs Multi-rack clusters
  - Reliable storage, Rack-awareness, Throughput
  - Inside: Name Node file system, Read, Write