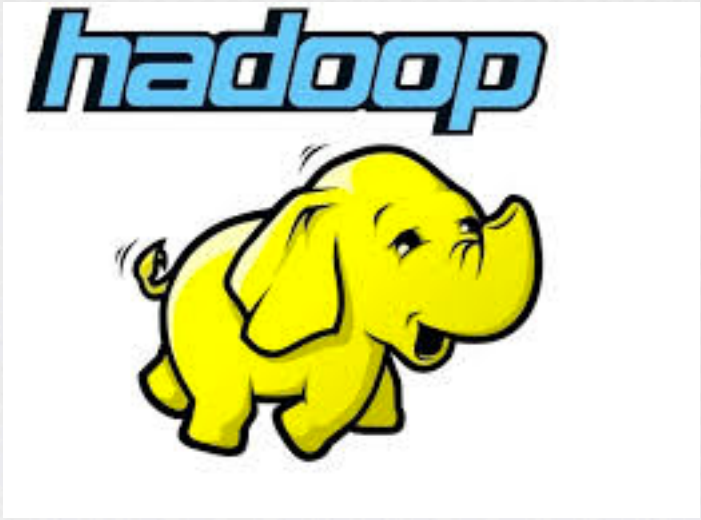
UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS



BIG DATA and AI
for business

Lecture 4 (02/12, 02/14): Hadoop

Decisions, Operations & Information Technologies
Robert H. Smith School of Business
Spring, 2018

What we'll cover…

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Overview
- Architecture
  - Hadoop distributed file system (HDFS)
- Hands-on
  - Installation and configuration
  - Running Hadoop jobs

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

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- 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

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- 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

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- 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?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

Social

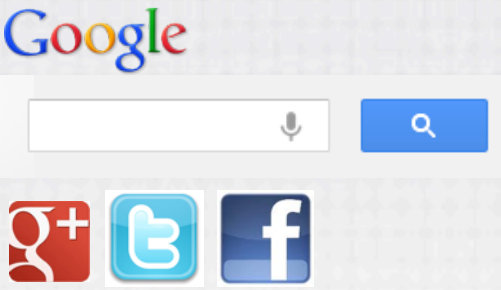
User Tracking & Engagement

Homeland Security

eCommerce

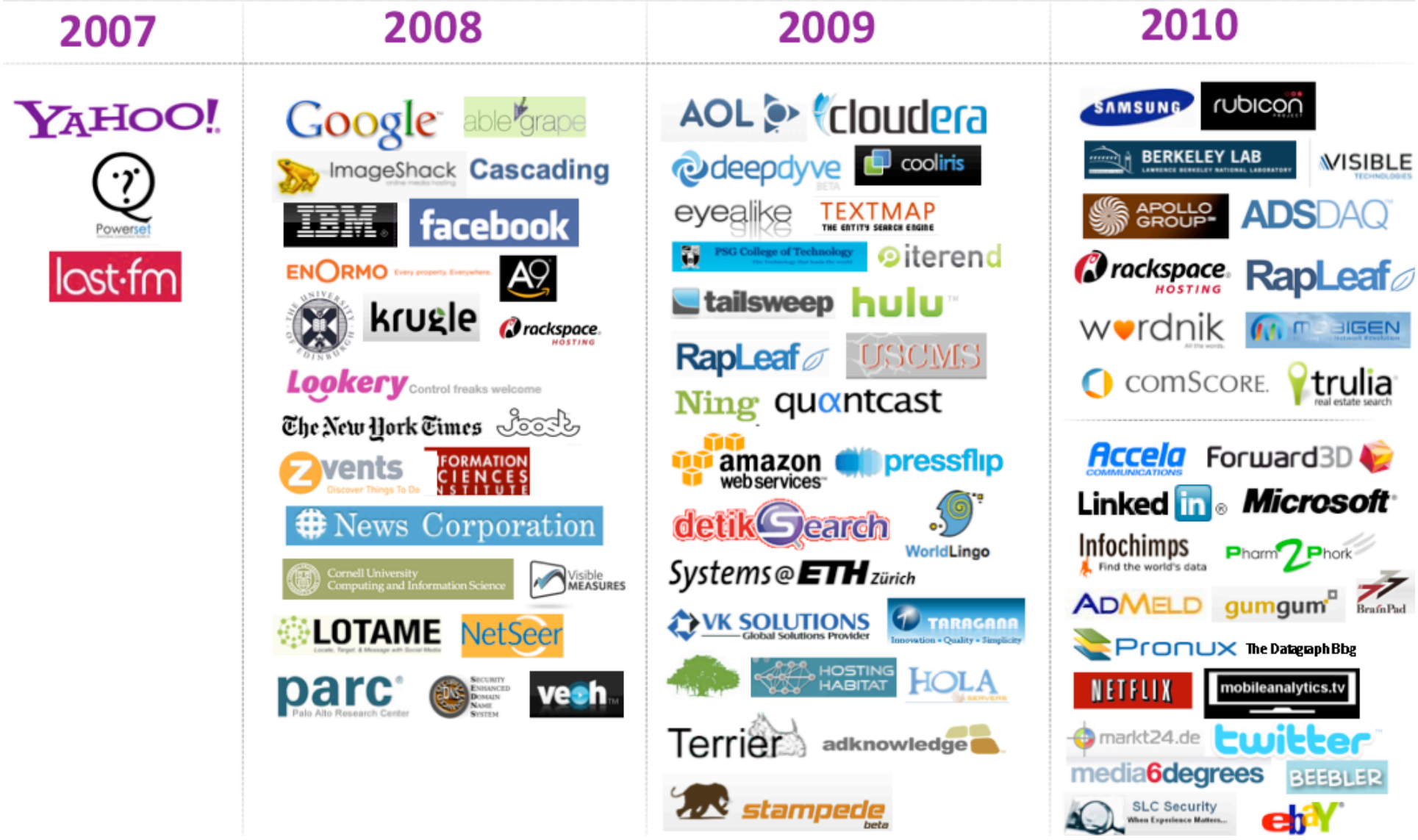
Financial Services

Real Time Search



Who uses Hadoop?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS



Why HDFS?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- **Problem 1:** Data is too big to store on one machine.
- **HDFS:** Store the data on multiple machines!

Why HDFS?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

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

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

Why HDFS?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

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

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

Why HDFS?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

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

- **HDFS:** Replicate the data!

Why HDFS?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

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

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

Another reason why Hadoop

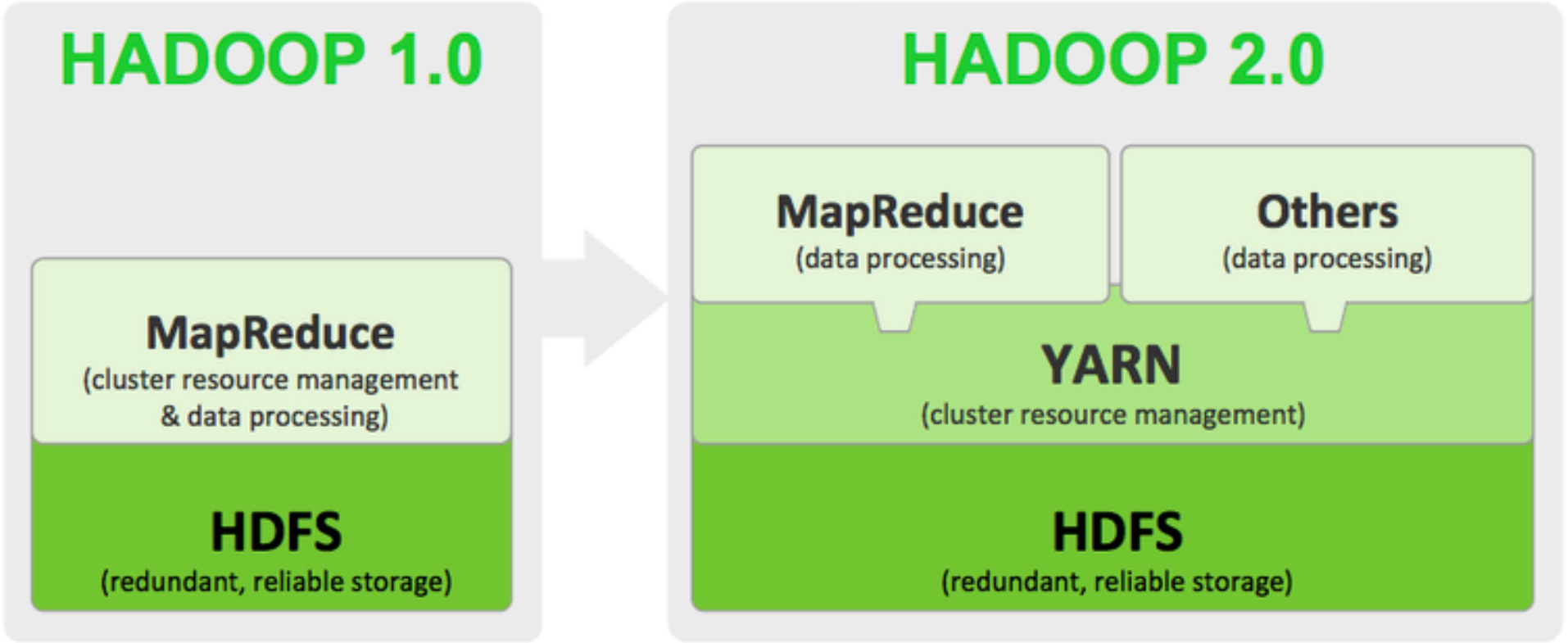
- **Scan** 100TB datasets on 1000 node cluster
  - Remote storage @ 10MB/s = 165 mins
  - Local storage @ 50-200MB/s = 33-8 mins
- Moving computation is more efficient than moving data
- Need fault tolerant store with reasonable availability guarantees
  - Handle hardware faults transparently

Hadoop goals

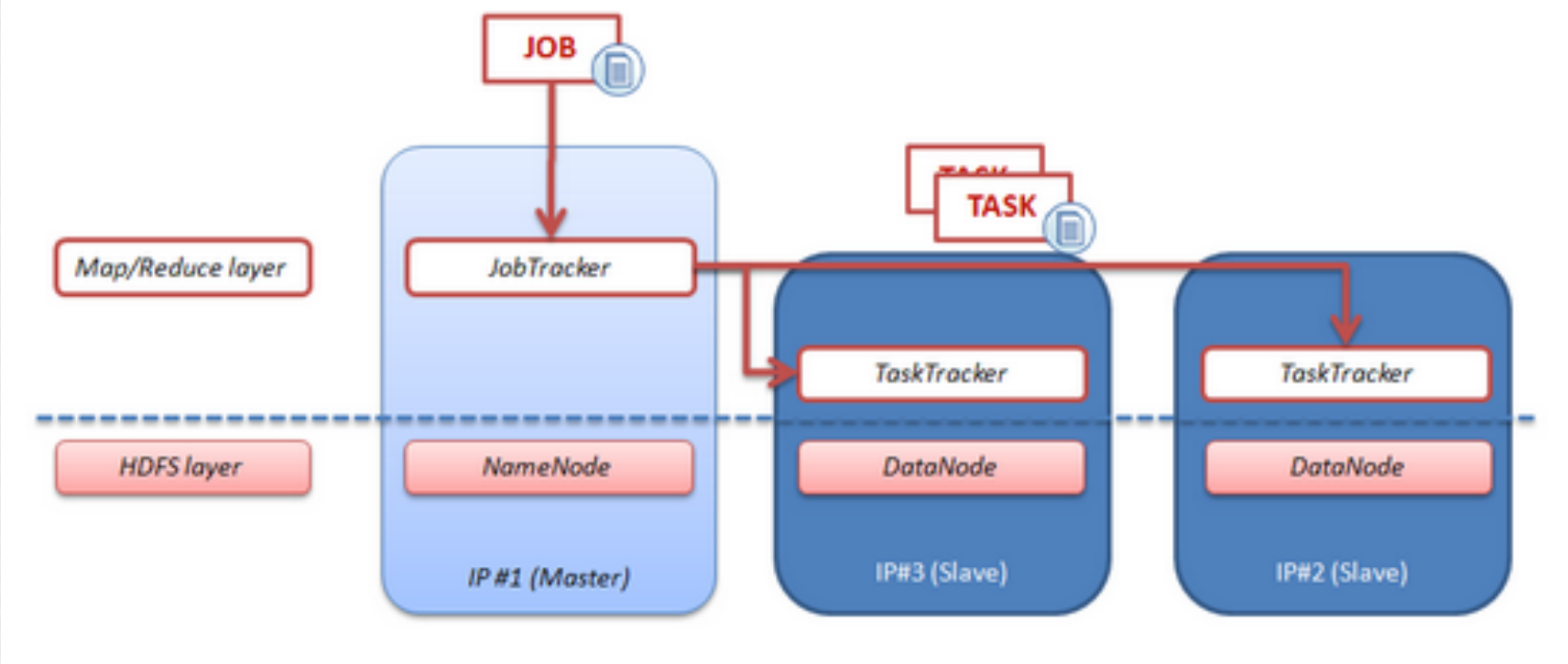
UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

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

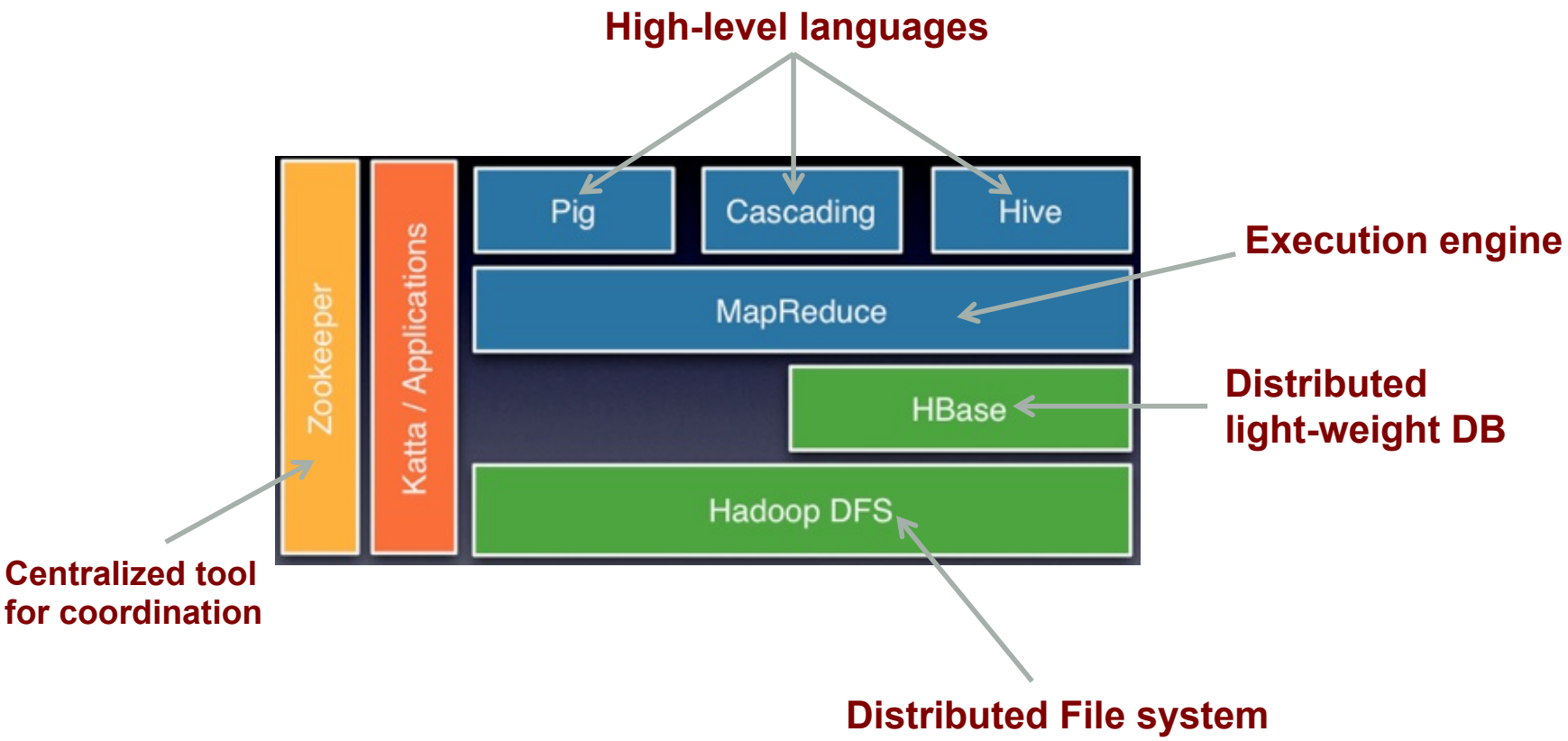
Hadoop big picture



High-level architecture of Hadoop

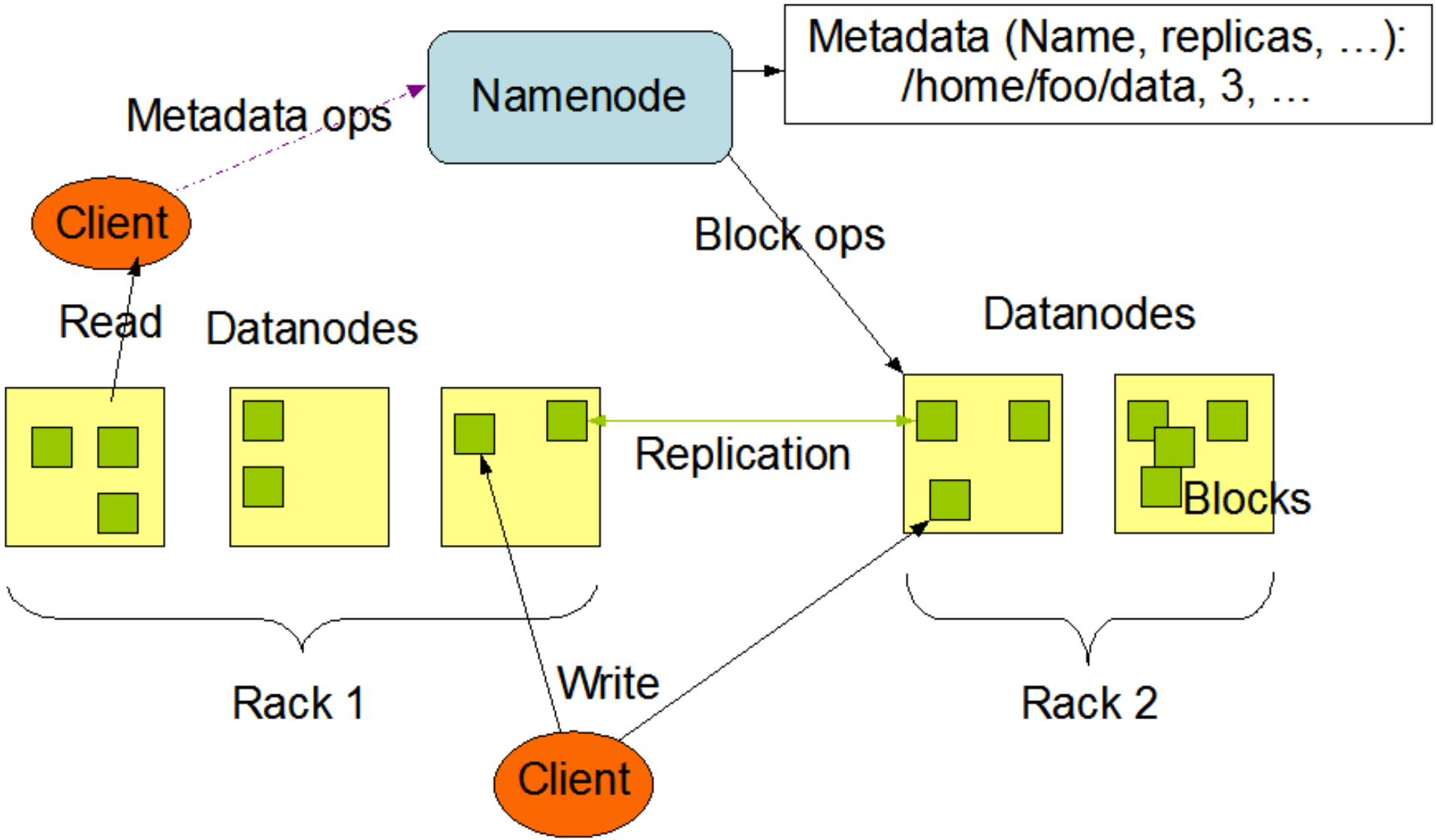


Hadoop big picture

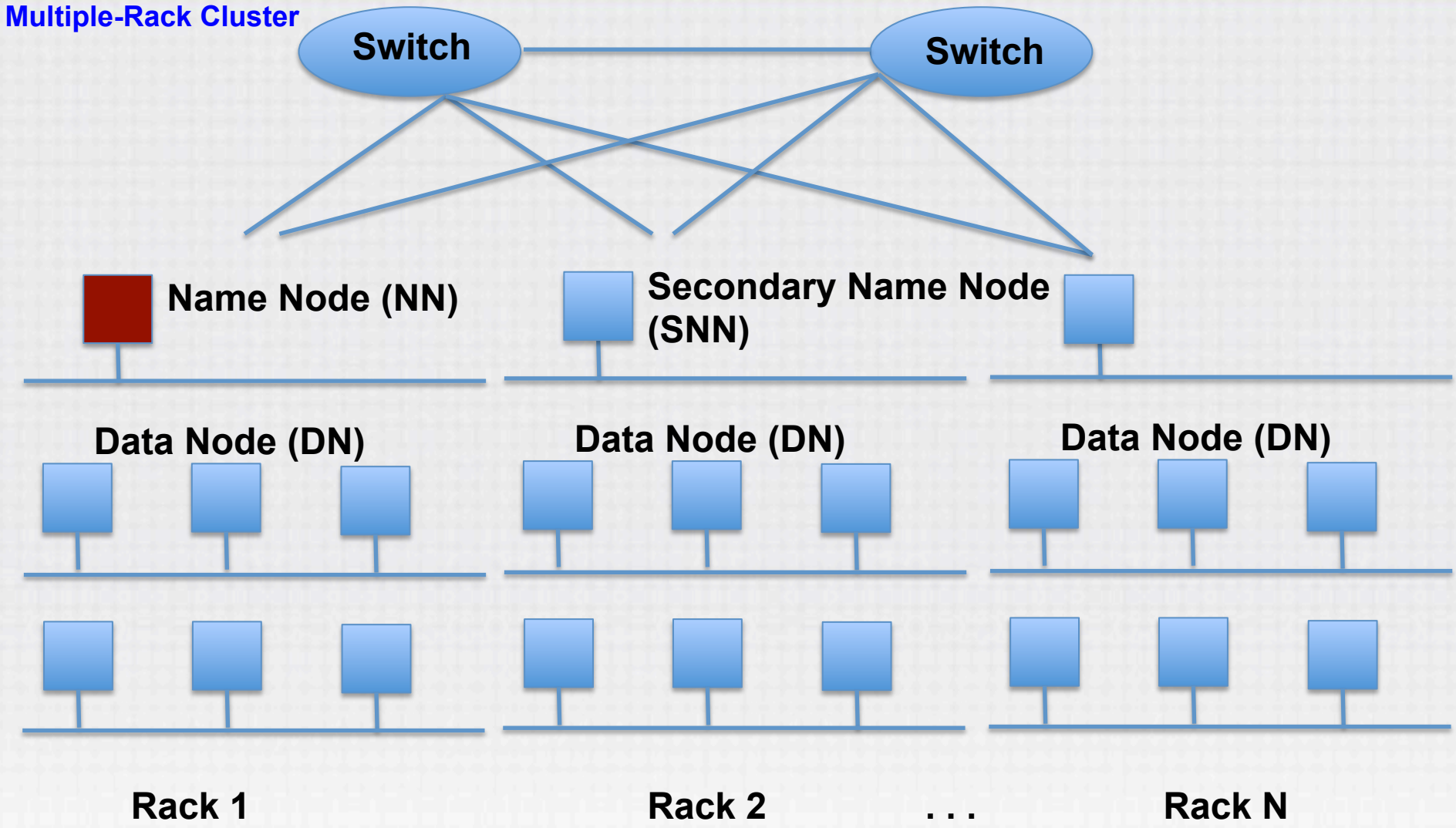


HDFS + MapReduce are enough to have things working

HDFS Architecture



HDFS architecture



HDFS

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- **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 (default: 64MB)
- 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)

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- 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

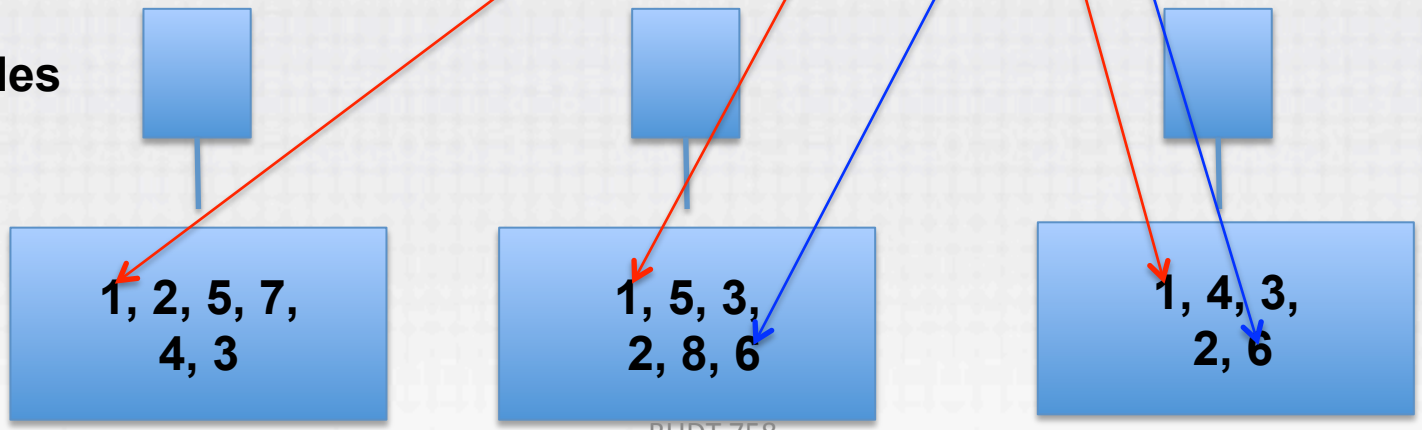
Name Node

Snapshot of FS

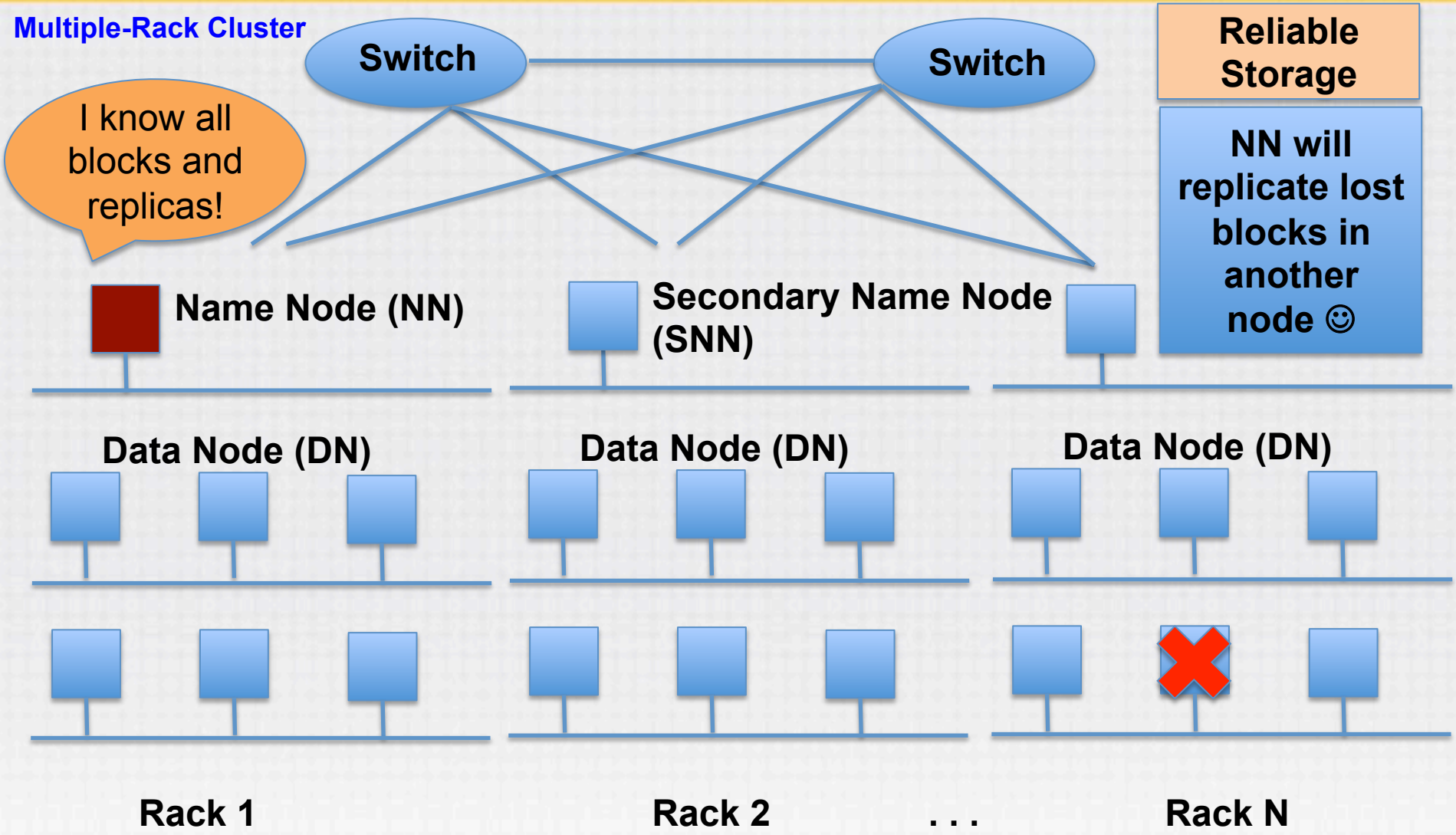
Edit log: record changes to FS

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

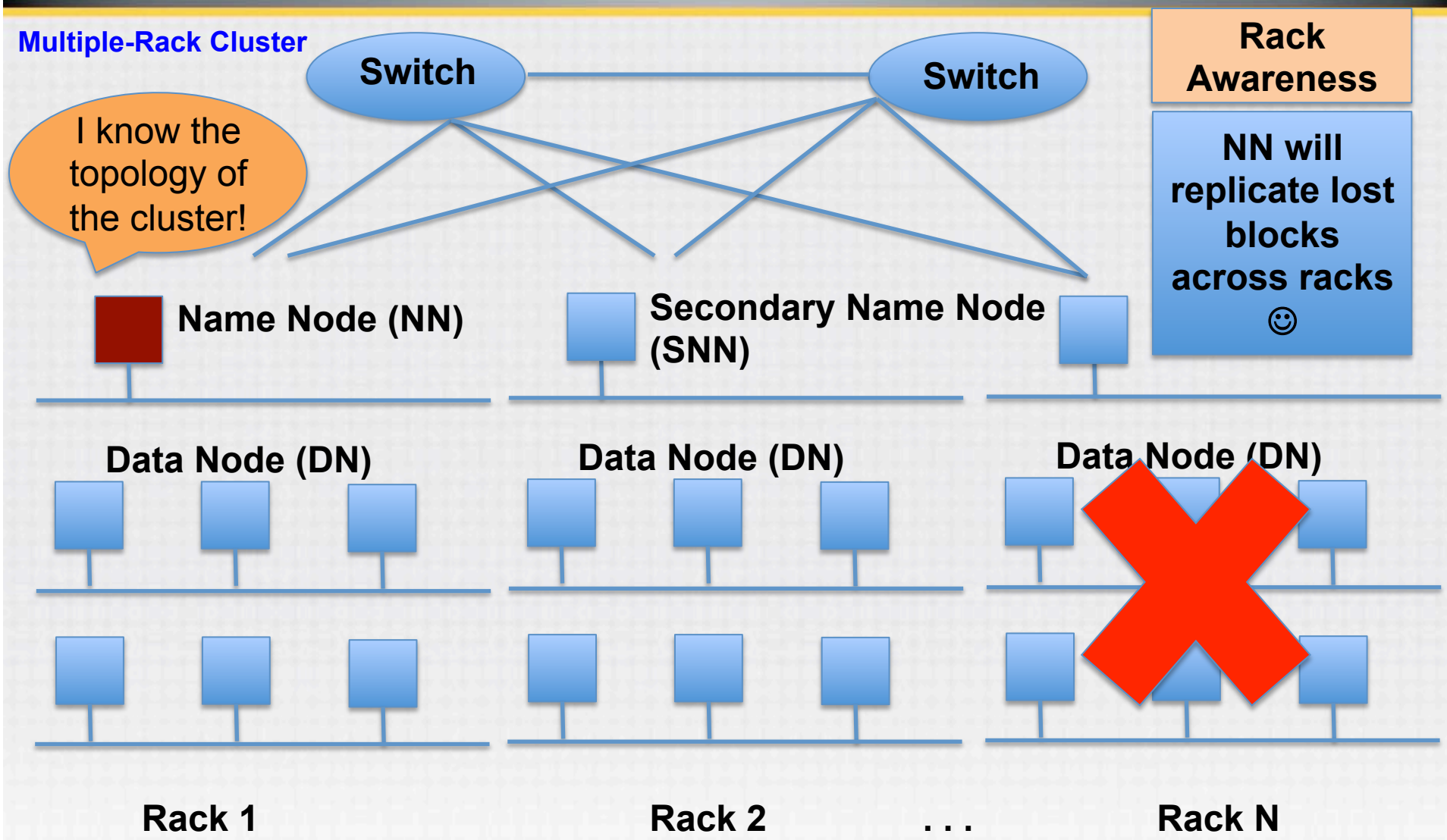
Data Nodes



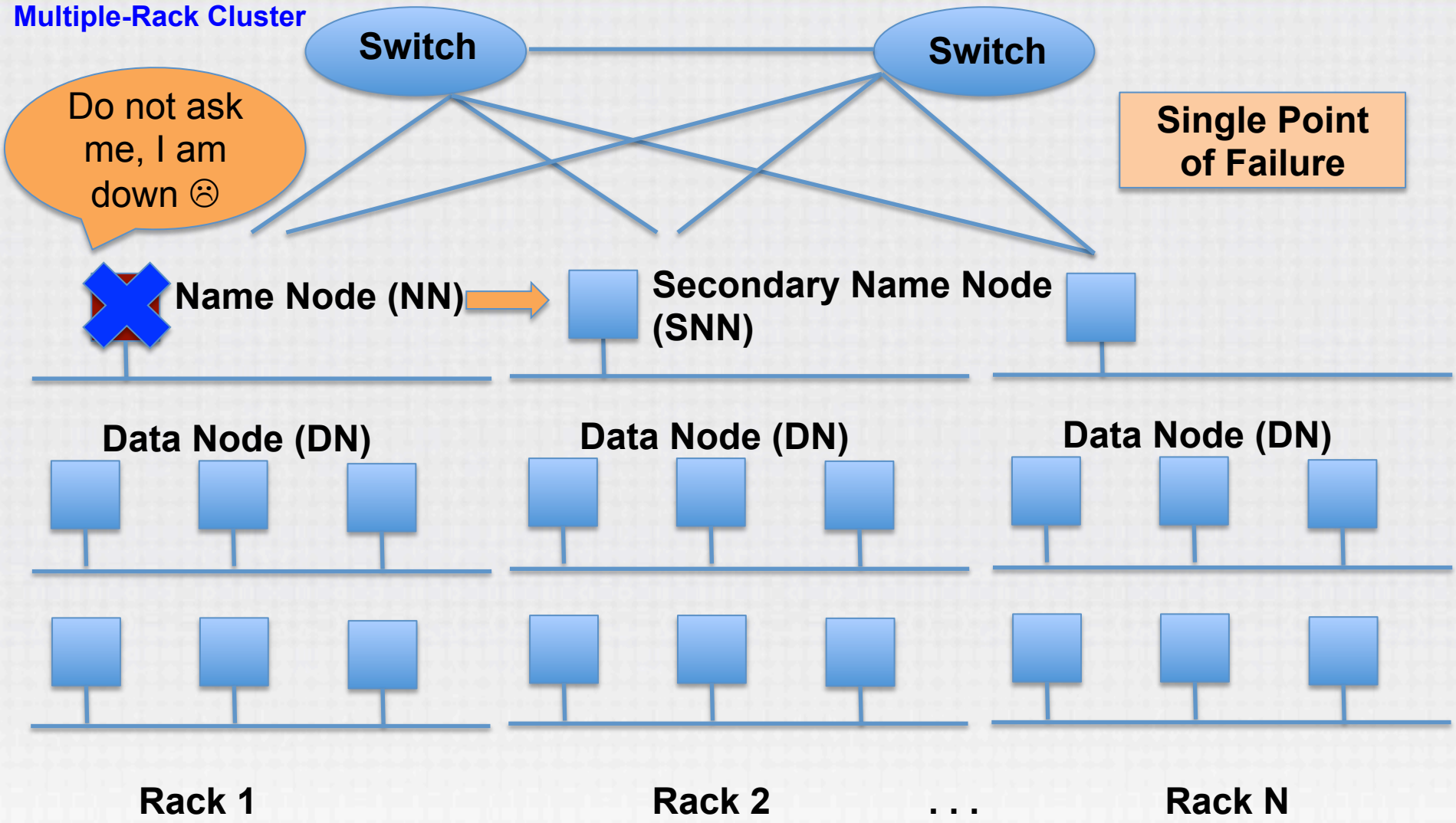
HDFS architecture



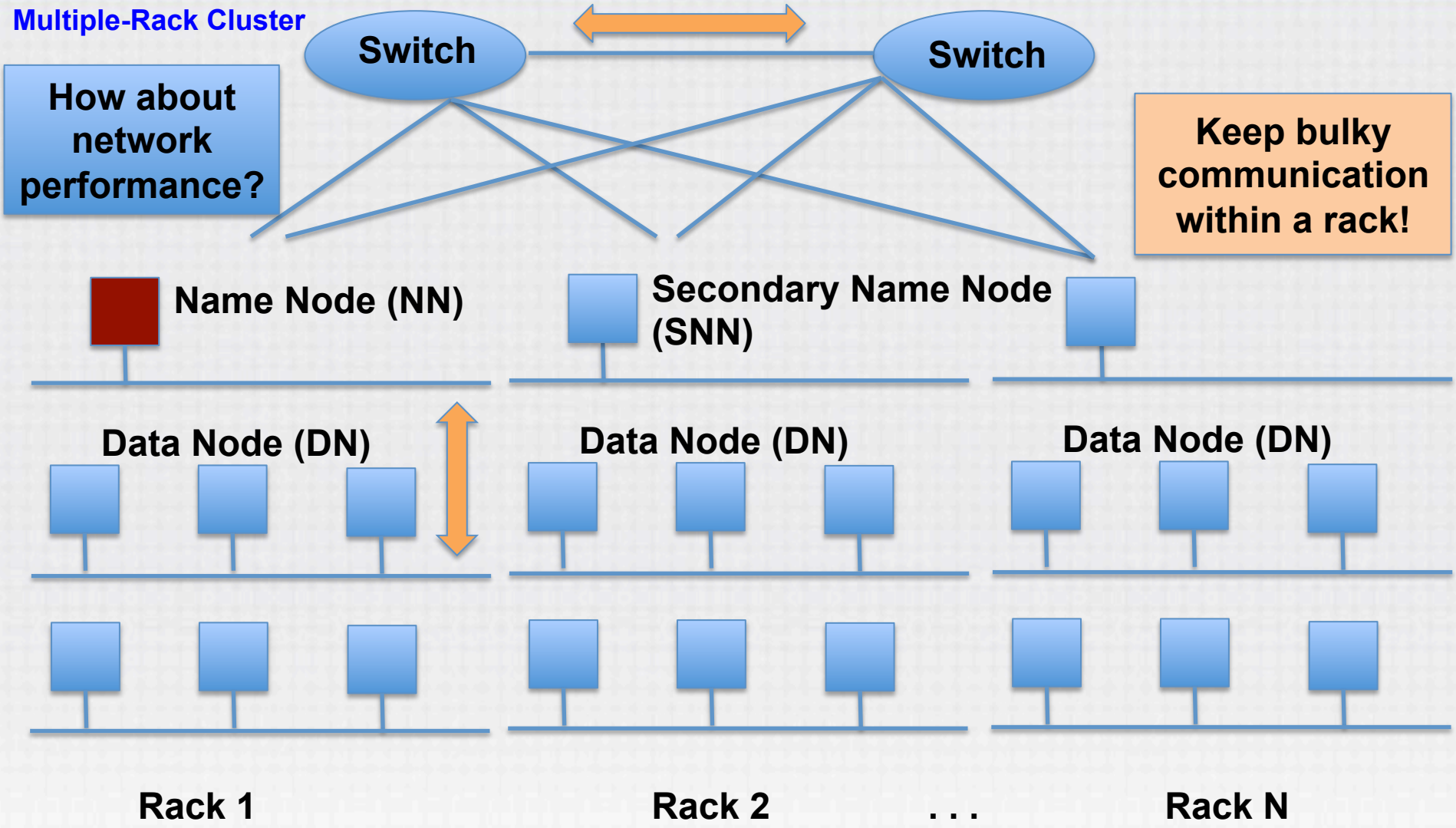
HDFS architecture



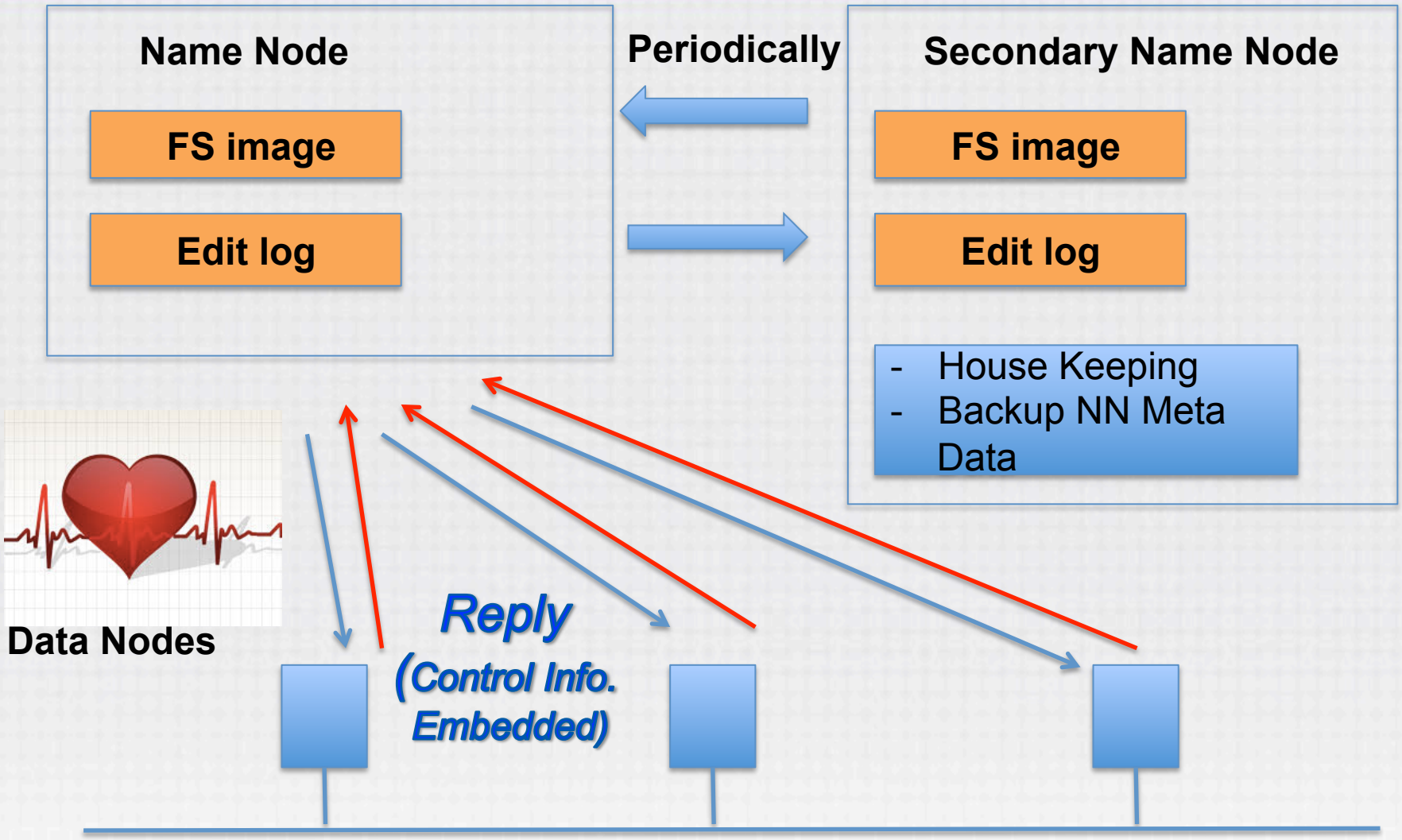
HDFS architecture



HDFS architecture



HDFS Inside: Name Node



HDFS Inside: Blocks

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

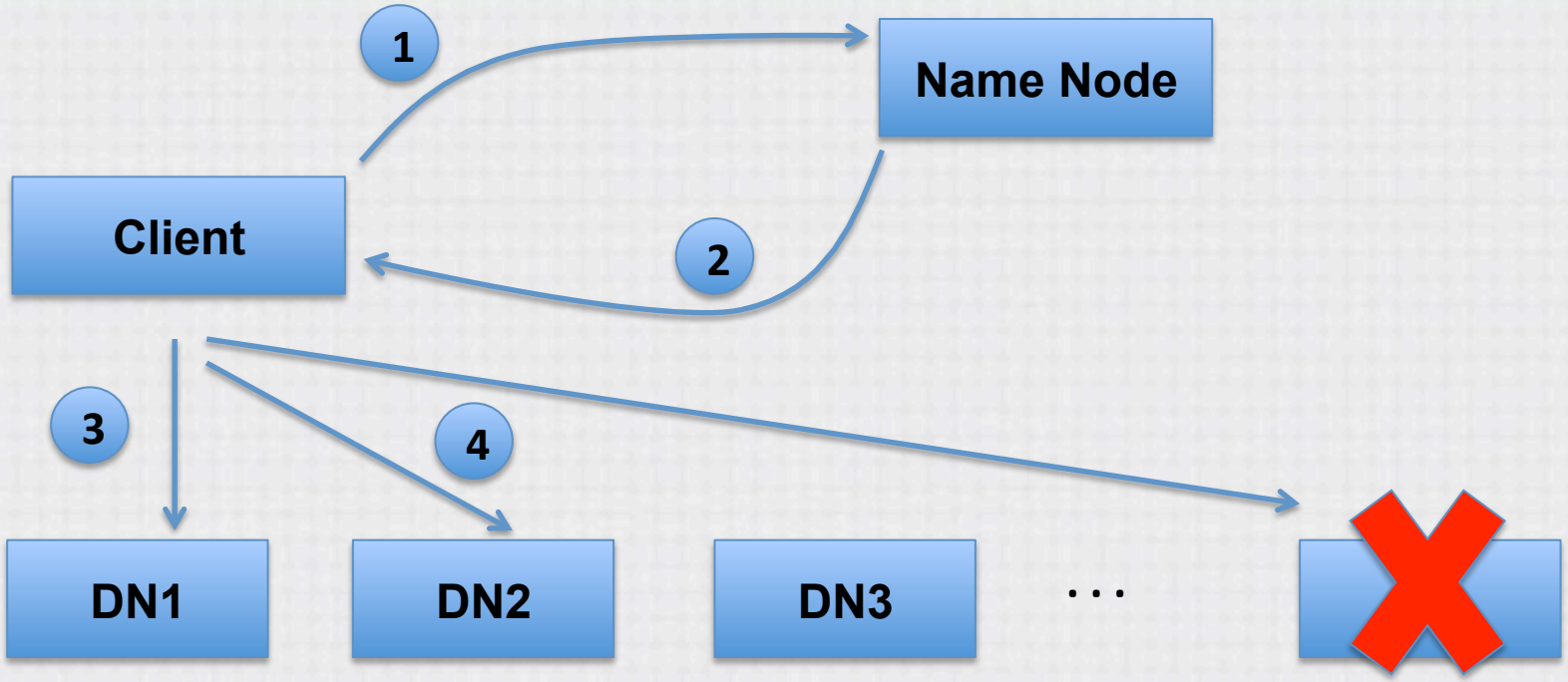
- Q: Why do we need the abstraction "Blocks" in addition to "Files"?
- Reasons:
  - File can be larger than a single disk
  - Block is of fixed size, easy to manage and manipulate
  - Easy to replicate and do more fine grained load balancing

HDFS Inside: Blocks

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- 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

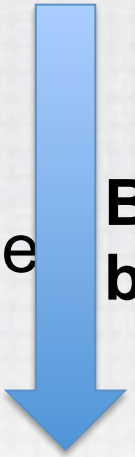
- Q: Why does HDFS choose such a design for read? Why not ask client to read blocks through NN?
- Reasons:
  - Prevent NN from being the bottleneck of the cluster
  - Allow HDFS to scale to large number of concurrent clients
  - Spread the data traffic across the cluster

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

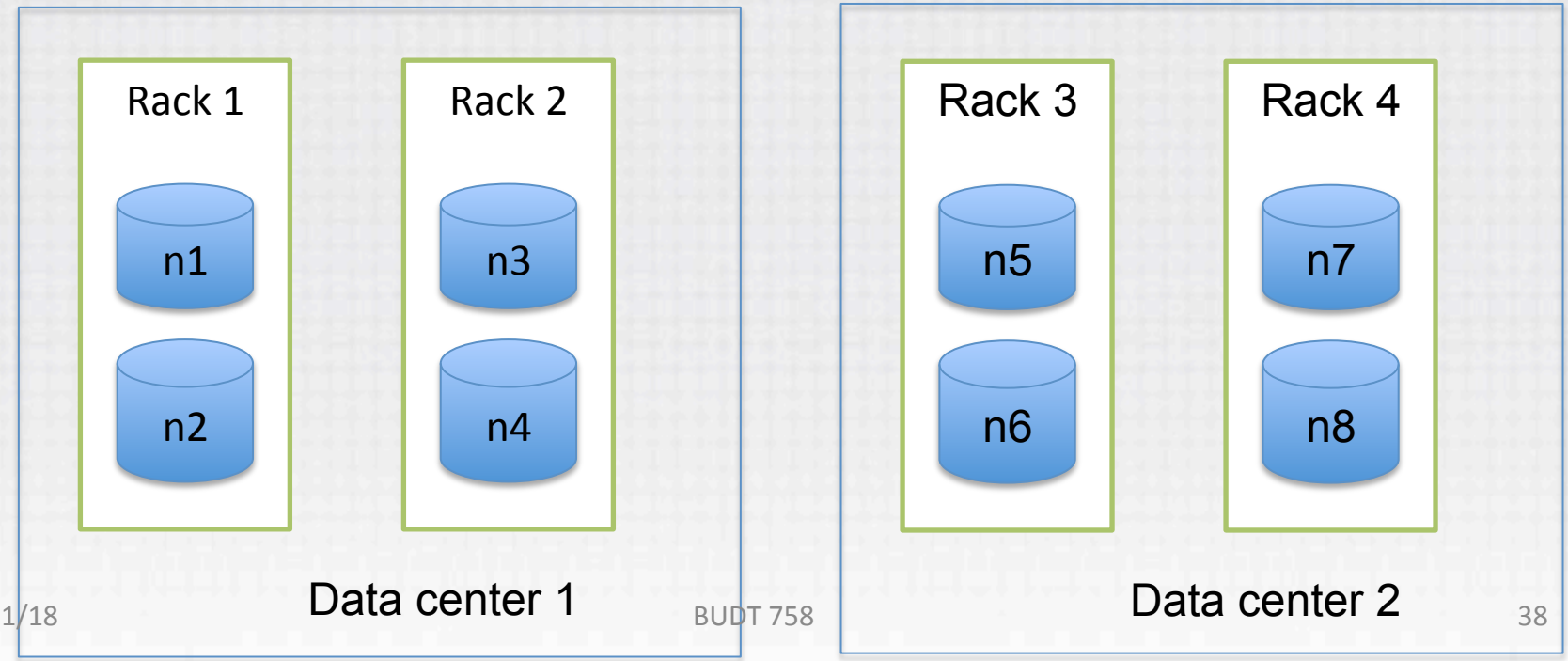
- The critical resource in HDFS is **bandwidth**, distance is defined based on that
- Measuring bandwidths between any pair of nodes is too complex and **does not scale**
- **Basic Idea:**
  - Processes on the same node
  - Different nodes on the same rack
  - Nodes on different racks in the same data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

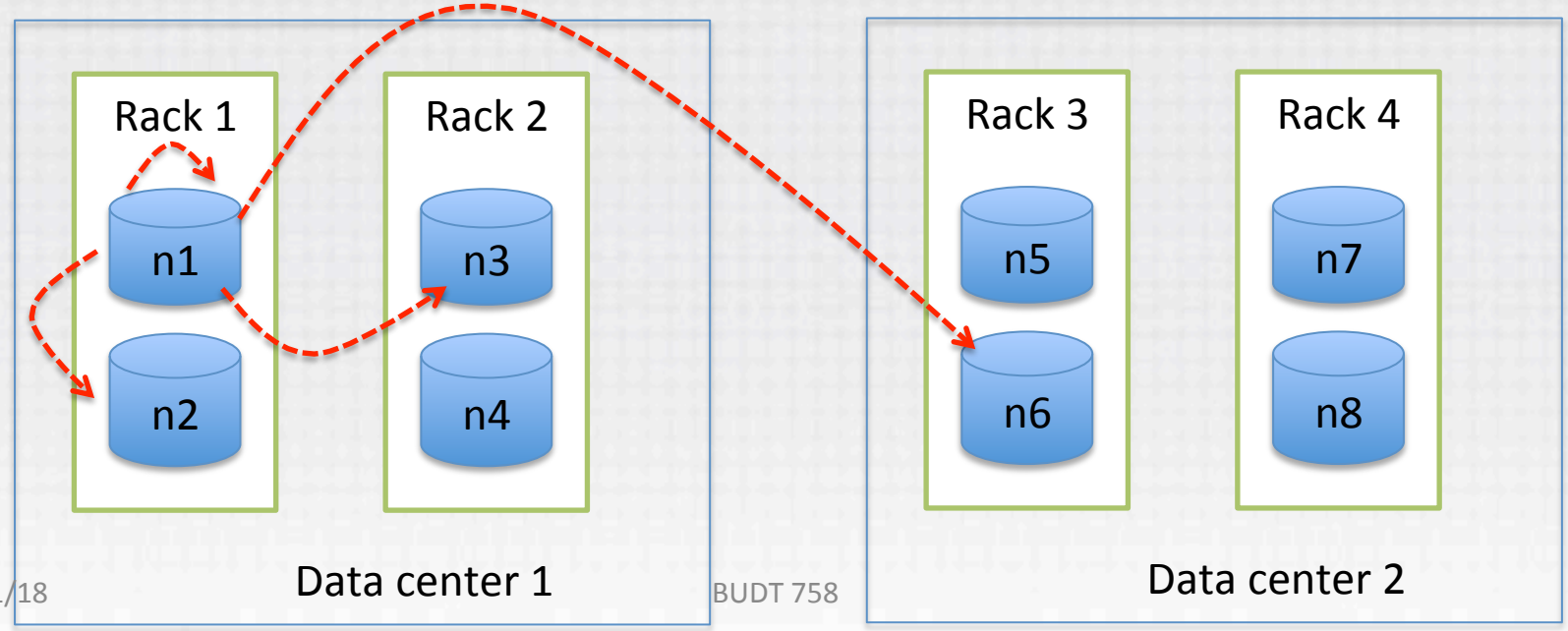
- What are the distance of the following pairs:

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

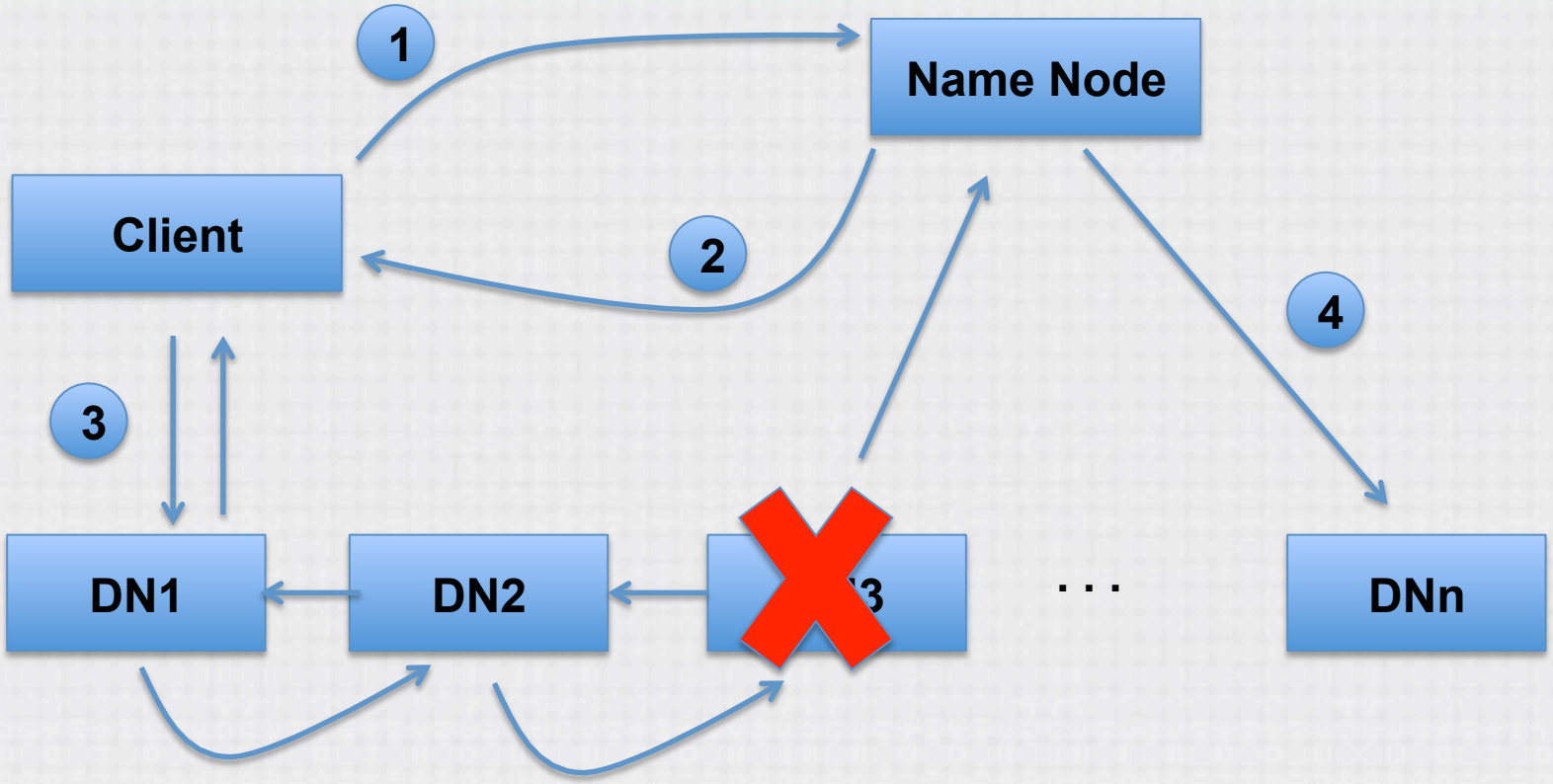
Dist(d1/r1/n1, d1/r1/n2)= **2**

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

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



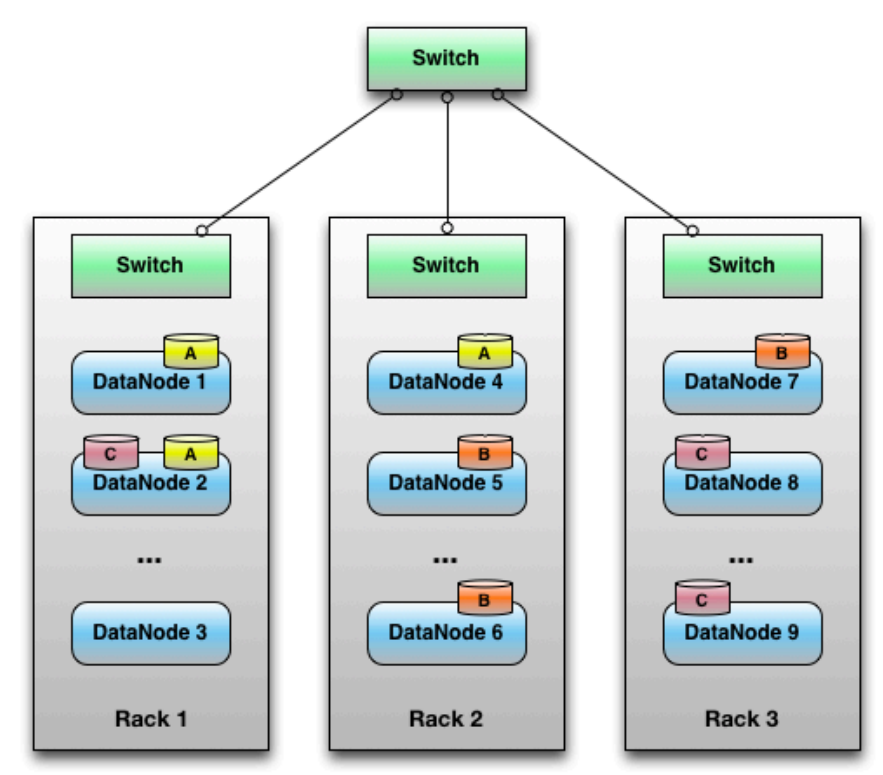
HDFS Inside: Write



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

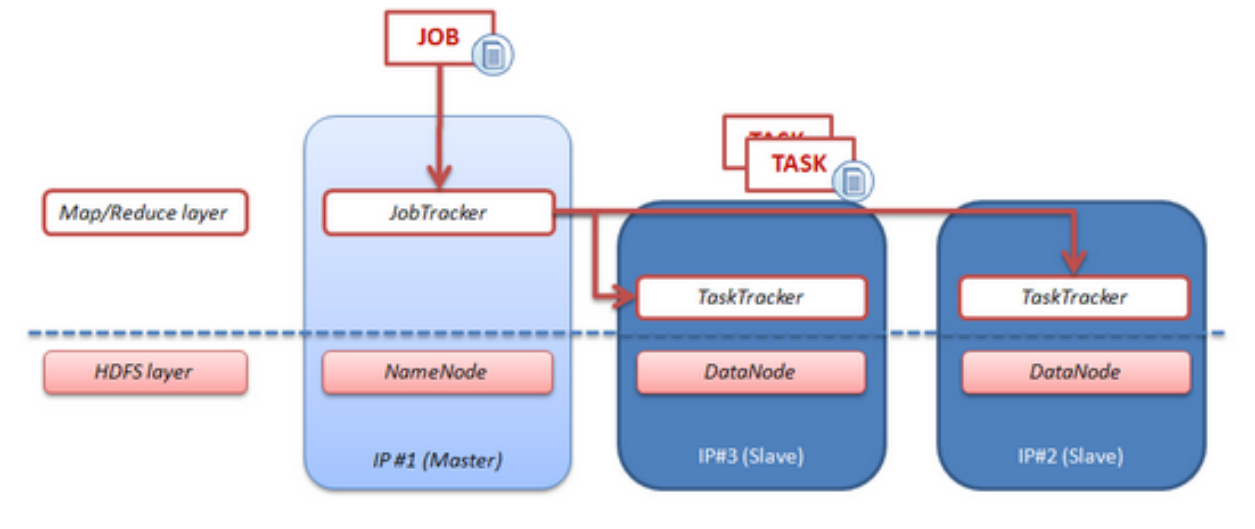
UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- 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

HDFS filesystem commands

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

1. List the contents of a directory
   - $hadoop fs -ls
2. Create a directory in HDFS at given path(s)
   - $hadoop fs -mkdir <directory name>

3. Upload and download a file in HDFS
   - Upload: $hadoop fs -put <local file> <remote path>
   - Download: $hadoop fs -get <file in HDFS> <local path>

4. See contents of a file
   - $hadoop fs -cat <filename>
5. Delete a file/directory in HDFS
   - $hadoop fs -rm/rmr <file or directory>

HDFS filesystem commands

6. Move file from source to destination
   - $hadoop fs -mv <src> <dst>
7. Report the amount of space used and availability
   - $hadoop fs -df hdfs:/
8. How much space a directory occupies
   - $hadoop fs -du -s -h <dir name>
9. Change permission of files
   - $sudo hadoop fs -chmod 600 <file>
10. Change owner and group of files
   - $sudo hadoop fs -chown root:root <file>

HDFS admin commands

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- DFSAdmin command
  - -report: reports basic statistics of HDFS
  - -safemode: though usually not required, an administrator can manually enter or leave safemode
    - enter, leave, get, wait
  - -refreshNodes: updates the set of hosts allowed to connect to namenode

Usage: hadoop dfsadmin [-report] [-safemode enter | leave | get | wait] [-refreshNodes] [-finalizeUpgrade] [-upgradeProgress status | details | force] [-metasave filename] [-setQuota <quota> <dirname>...<dirname>] [-clrQuota <dirname>...<dirname>] [-help [cmd]]

Installation and configuration

- Use the stable version, e.g., 1.0.3
- Local (standalone) mode
- **Fully-distributed mode**
- **Pseudo-distributed mode**

Fully distributed mode

- Assume we have three machines with following configurations

OS	Machine name	IP address
Ubuntu 13.04	Master.Hadoop	192.168.1.141
Ubuntu 9.11	Slave1.Hadoop	192.168.1.142
Fedora 17	Slave2.Hadoop	192.168.1.137

- Create a hadoop user account on for each machine

Modify machine names

- For different OS
  - Ubuntu OS

```
hadoop@Master:~
$ vi /etc/hostname
hadoop@Master:~
$ hostname
Master.Hadoop
hadoop@Master:~
```

  - Fedora OS

```
[hadoop@Slave2 conf]$ vi /etc/sysconfig/network
[hadoop@Slave2 conf]$ hostname
Slave2.Hadoop
[hadoop@Slave2 conf]$
```

```
NETWORKING=yes
HOSTNAME=Slave2.Hadoop
NTPSERVERARGS=iburst
~
```

  - Mac OS
    - $ sudo scutil --set HostName Master.Hadoop

Configure DNS

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Modify /etc/hosts under the master machine

```
hadoop@Master:~
$ ping 192.168.1.142
PING 192.168.1.142 (192.168.1.142) 56(84) bytes of data.
64 bytes from 192.168.1.142: icmp_req=1 ttl=64 time=7.51 ms
64 bytes from 192.168.1.142: icmp_req=2 ttl=64 time=0.974 ms
^C
--- 192.168.1.142 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.974/4.244/7.515/3.271 ms
hadoop@Master:~
$ ping Slave1.Hadoop
ping: unknown host Slave1.Hadoop
hadoop@Master:~
```

```
#127.0.0.1      localhost
127.0.0.1       Master.Hadoop

192.168.1.141  Master.Hadoop
192.168.1.142  Slave1.Hadoop
192.168.1.137  Slave2.Hadoop
```

Required software

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

1. JDK

   Download:
   http://www.oracle.com/technetwork/java/javase/index.html

   e.g., jdk-7u25-linux-i586.tar.gz

2. Hadoop

   Download:
   http://hadoop.apache.org/common/releases.html

   e.g., hadoop-1.0.3.tar.gz

3. SSH
   - $ sudo apt-get install openssh-server (for ubuntu OS)
   - $ yum install openssh-server (for Fedora OS)

Passwordless login for SSH (1)

- $ ssh-kegen –t rsa –P ‘’ (no spaces between single quotes; ‘P’ is uppercase)

```
root@Master:/usr# ssh-keygen -t rsa -P ''
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
e5:bf:4a:ca:88:8b:34:a9:d4:0a:bc:ff:b1:51:74:dc root@Master.Hadoop
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|        . .      |
|       . o.E     |
|      . .o       |
|       .S .      |
|. ..  .    .     |
|.o+. o    . .    |
|oooo .+o o   .   |
|.oo.++. o ...    |
+-----------------+
```

- $ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
- $ chmod 600 ~/.ssh/authorized_keys

Passwordless login for SSH (2)

- $ sudo vi /etc/ssh/sshd_confid
- Make sure the following lines are uncommented

```
# Authentication:
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile      %h/.ssh/authorized_keys
```

- Restart ssh service
  - $ sudo service ssh restart

Passwordless login for SSH (3)

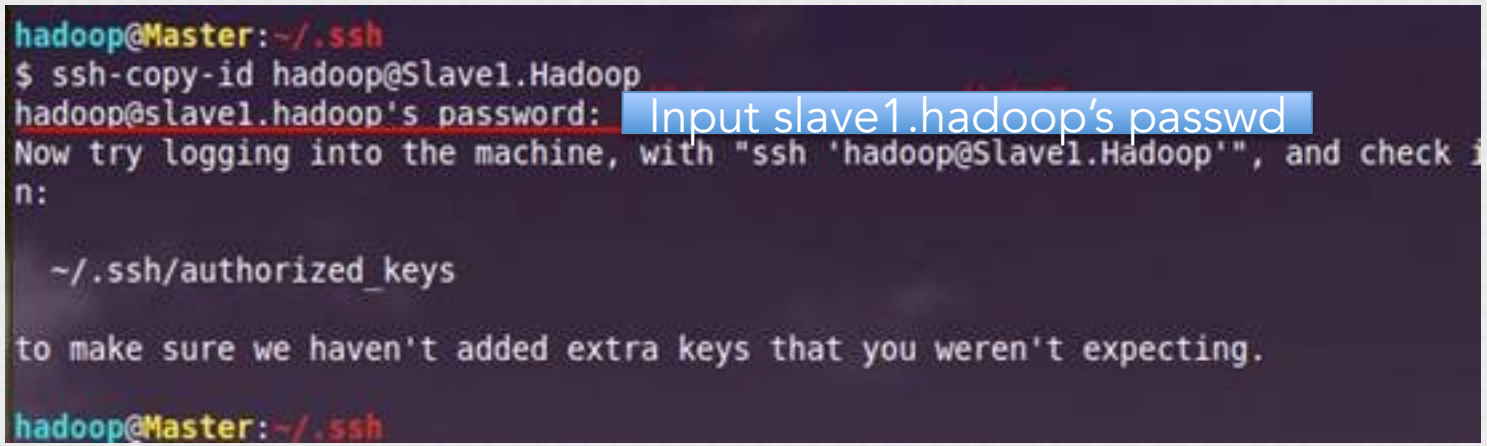
UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Do the ssh-add if ssh is still not working

```
seed@Master:~/.ssh
$ ssh localhost
Agent admitted failure to sign using the key.
seed@localhost's password:

seed@Master:~/.ssh
$ ssh-add ~/.ssh/id_rsa
Identity added: /home/seed/.ssh/id_rsa (/home/seed/.ssh/id_rsa)
seed@Master:~/.ssh
$ ssh localhost
Welcome to Ubuntu 11.04 (GNU/Linux 2.6.38-8-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release 'oneiric' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Aug 27 10:31:16 2013 from localhost
```

Passwordless login for SSH (4)

- Copy the generated public key to all slave machines in order passwordless access from the master to slaves



Passwordless login for SSH (5)

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Test if it is successful to access slave machines without password

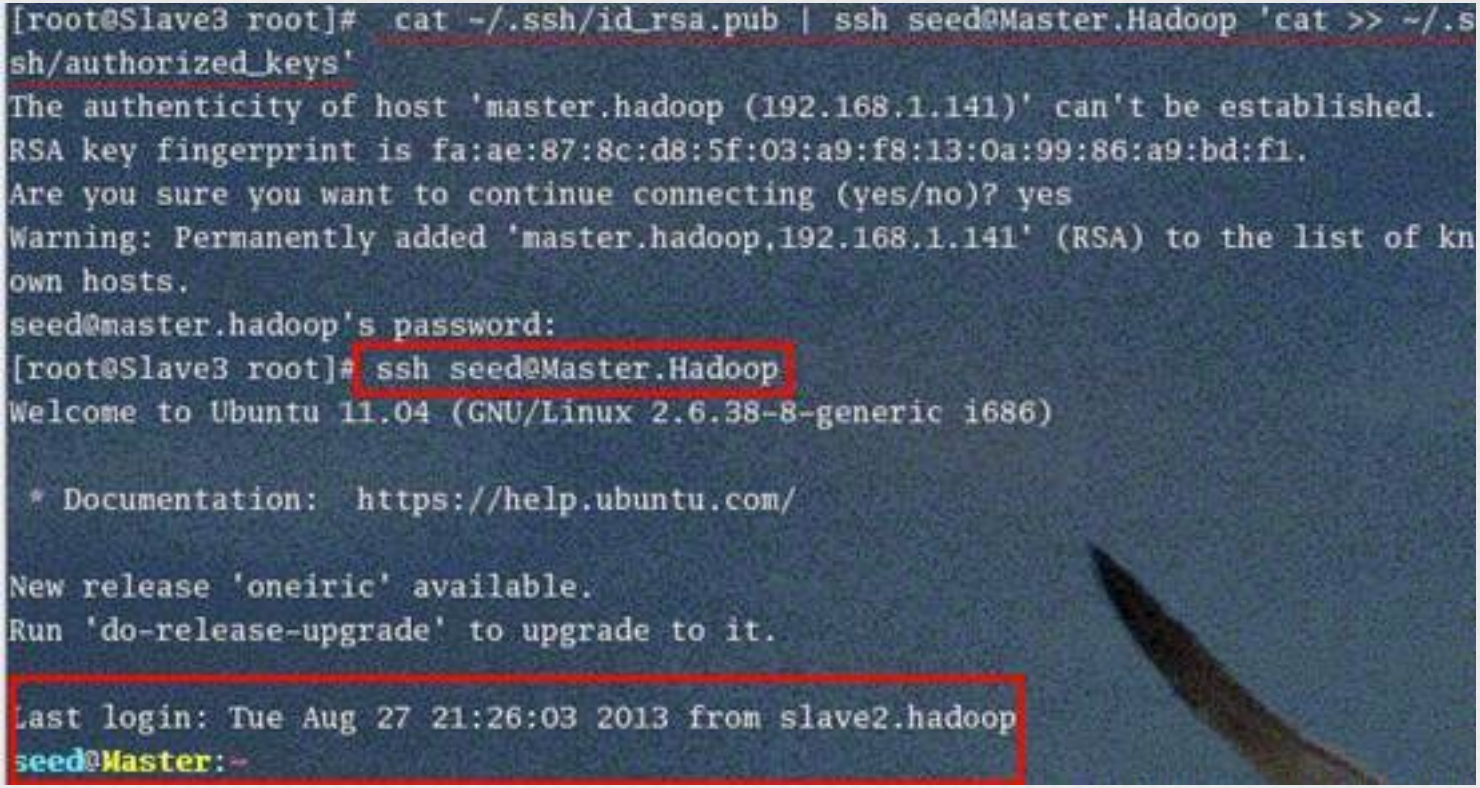
```
hadoop@Master:~/.ssh
$ ssh hadoop@Slave1.Hadoop
Linux Slave1.Hadoop 2.6.28-11-generic #42-Ubuntu SMP Fri Apr 17 01:57:59 UTC 200
9 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/

105 packages can be updated.
55 updates are security updates.

Last login: Tue Aug 27 21:47:57 2013 from slave3.hadoop
hadoop@Slave1:~$
```

Passwordless login for SSH (6)

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- To make passwordless access from slave machines to the master machine, we need to add public keys of slave machines into 'authorized_keys' on the master machine under the folder of .ssh using the following command

- $ cat ~/.ssh/id_rsa.pub | ssh hadoop@Master.Hadoop 'cat >> ~/.ssh/authorized_keys'
- It is equal to two commands:
  - $ scp ~/.ssh/id_rsa.pub hadoop@Master.Hadoop:/~
  - $ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys


[root@Slave3 root]# cat ~/.ssh/id_rsa.pub | ssh seed@Master.Hadoop 'cat >> ~/.s
sh/authorized_keys'
The authenticity of host 'master.hadoop (192.168.1.141)' can't be established.
RSA key fingerprint is fa:ae:87:8c:d8:5f:03:a9:f8:13:0a:99:86:a9:bd:f1.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'master.hadoop,192.168.1.141' (RSA) to the list of kn
own hosts.
seed@master.hadoop's password:
[root@Slave3 root]# ssh seed@Master.Hadoop
Welcome to Ubuntu 11.04 (GNU/Linux 2.6.38-8-generic i686)
* Documentation: https://help.ubuntu.com/
New release 'oneiric' available.
Run 'do-release-upgrade' to upgrade to it.
Last login: Tue Aug 27 21:26:03 2013 from slave2.hadoop
seed@Master:~

Set Java environment variables

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Edit /etc/profile (Ubuntu) or ~/.bashrc_profile (Mac)

```
umask 022

# set java environment
export JAVA_HOME=/usr/java/jdk1.7.0_25/
export JRE_HOME=/usr/java/jdk1.7.0_25/jre
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
```

- Make it work through
  - $ source /etc/profile or . /etc/profile

```
root@Master:/usr/java# vi /etc/profile
root@Master:/usr/java# . /etc/profile
root@Master:/usr/java#
```

Install Hadoop

```
cd /usr
tar –xzvf hadoop-1.1.2.tar.gz
mv hadoop-1.1.2 hadoop
chown –R hadoop:hadoop hadoop
rm -rf hadoop-1.1.2.tar.gz
```

- Modify /etc/profile to add some environment variables

```
umask 022

# set java environment
export JAVA_HOME=/usr/java/jdk1.7.0_25/
export JRE_HOME=/usr/java/jdk1.7.0_25/jre
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin

# set Hadoop environment
export HADOOP_HOME=/usr/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
:
```

Configure Hadoop (1)

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Edit /usr/hadoop/conf/hadoop-env.sh

```
# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  A
# optional.  When running a distributed configuration it
# set JAVA_HOME in this file, so that it is correctly def
# remote nodes.

# The java implementation to use.  Required.
export JAVA_HOME=/usr/java/jdk1.7.0_25
```

Configure Hadoop (2)

- Edit /usr/hadoop/conf/core-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/usr/hadoop/tmp</value>
    <description>A base for other temporary directories.</description>
  </property>
  <!-- file system properties -->
  <property>
    <name>fs.default.name</name>
    <value>hdfs://192.168.1.141:9000</value>
  </property>
</configuration>
```

Configure Hadoop (3)

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Edit /usr/hadoop/conf/hdfs-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
~
```

Configure Hadoop (4)

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Edit /usr/hadoop/conf/mapred-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>http://192.168.1.141:9001</value>
  </property>
</configuration>
~
```

Configure Hadoop (5)

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Edit 'masters' file

```
root@Master:/usr/hadoop/conf# vi masters
root@Master:/usr/hadoop/conf# cat masters
192.168.1.141
root@Master:/usr/hadoop/conf#
```

- Configure 'slaves' file (only needed on the master machine)

```
root@Master:/usr/hadoop/conf# vi slaves
root@Master:/usr/hadoop/conf# cat slaves
192.168.1.142
192.168.1.137
```

Start Hadoop (1)

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
hadoop@Master:/usr/hadoop
$ hadoop namenode -format
Warning: $HADOOP_HOME is deprecated.

13/08/28 03:31:47 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = Master.Hadoop/127.0.0.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 1.1.2
STARTUP_MSG:   build = https://svn.apache.org/repos/asf/hadoop/common/branches/b
ranch-1.1 -r 1440782; compiled by 'hortonfo' on Thu Jan 31 02:03:24 UTC 2013
************************************************************/
13/08/28 03:31:47 INFO util.GSet: VM type       = 32-bit
13/08/28 03:31:47 INFO util.GSet: 2% max memory = 19.33375 MB
13/08/28 03:31:47 INFO util.GSet: capacity      = 2^22 = 4194304 entries
13/08/28 03:31:47 INFO util.GSet: recommended=4194304, actual=4194304
13/08/28 03:31:49 INFO namenode.FSNamesystem: fsOwner=hadoop
13/08/28 03:31:50 INFO namenode.FSNamesystem: supergroup=supergroup
13/08/28 03:31:50 INFO namenode.FSNamesystem: isPermissionEnabled=true
13/08/28 03:31:50 INFO namenode.FSNamesystem: dfs.block.invalidate.limit=100
13/08/28 03:31:50 INFO namenode.FSNamesystem: isAccessTokenEnabled=false accessK
eyUpdateInterval=0 min(s), accessTokenLifetime=0 min(s)
13/08/28 03:31:50 INFO namenode.NameNode: Caching file names occuring more than
10 times
13/08/28 03:31:51 INFO common.Storage: Image file of size 112 saved in 0 seconds
.
13/08/28 03:31:51 INFO namenode.FSEditLog: closing edit log: position=4, editlog
=/usr/hadoop/tmp/dfs/name/current/edits
13/08/28 03:31:51 INFO namenode.FSEditLog: close success: truncate to 4, editlog
=/usr/hadoop/tmp/dfs/name/current/edits
13/08/28 03:31:51 INFO common.Storage: Storage directory /usr/hadoop/tmp/dfs/nam
e has been successfully formatted.
13/08/28 03:31:51 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at Master.Hadoop/127.0.0.1
************************************************************/
```

Start Hadoop (2)

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

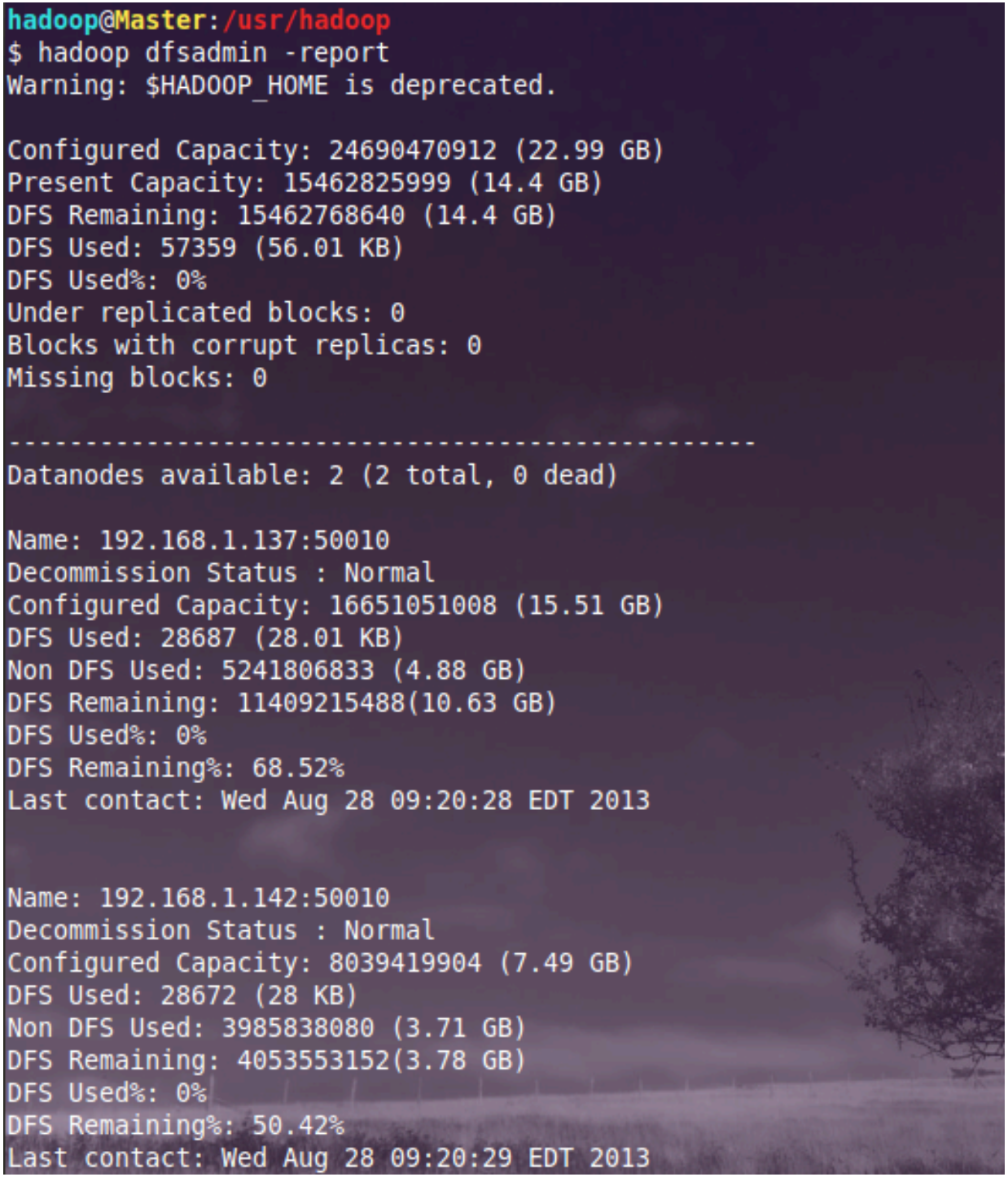
```
hadoop@Master:/usr/hadoop
$ start-all.sh
Warning: $HADOOP_HOME is deprecated.

starting namenode, logging to /usr/hadoop/libexec/../logs/hadoop-hadoop-namenode
-Master.Hadoop.out
192.168.1.137: starting datanode, logging to /usr/hadoop/libexec/../logs/hadoop-
hadoop-datanode-Slave2.Hadoop.out
192.168.1.142: starting datanode, logging to /usr/hadoop/libexec/../logs/hadoop-
hadoop-datanode-Slave1.Hadoop.out
192.168.1.141: starting secondarynamenode, logging to /usr/hadoop/libexec/../log
s/hadoop-hadoop-secondarynamenode-Master.Hadoop.out
starting jobtracker, logging to /usr/hadoop/libexec/../logs/hadoop-hadoop-jobtra
cker-Master.Hadoop.out
192.168.1.142: starting tasktracker, logging to /usr/hadoop/libexec/../logs/hado
op-hadoop-tasktracker-Slave1.Hadoop.out
192.168.1.137: starting tasktracker, logging to /usr/hadoop/libexec/../logs/hado
op-hadoop-tasktracker-Slave2.Hadoop.out
hadoop@Master:/usr/hadoop
```

Check Hadoop status

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
hadoop@Master:/usr/hadoop/tmp
$ jps
6231 Jps
5753 SecondaryNameNode
5834 JobTracker
5549 NameNode
hadoop@Master:/usr/hadoop/tmp
$
```

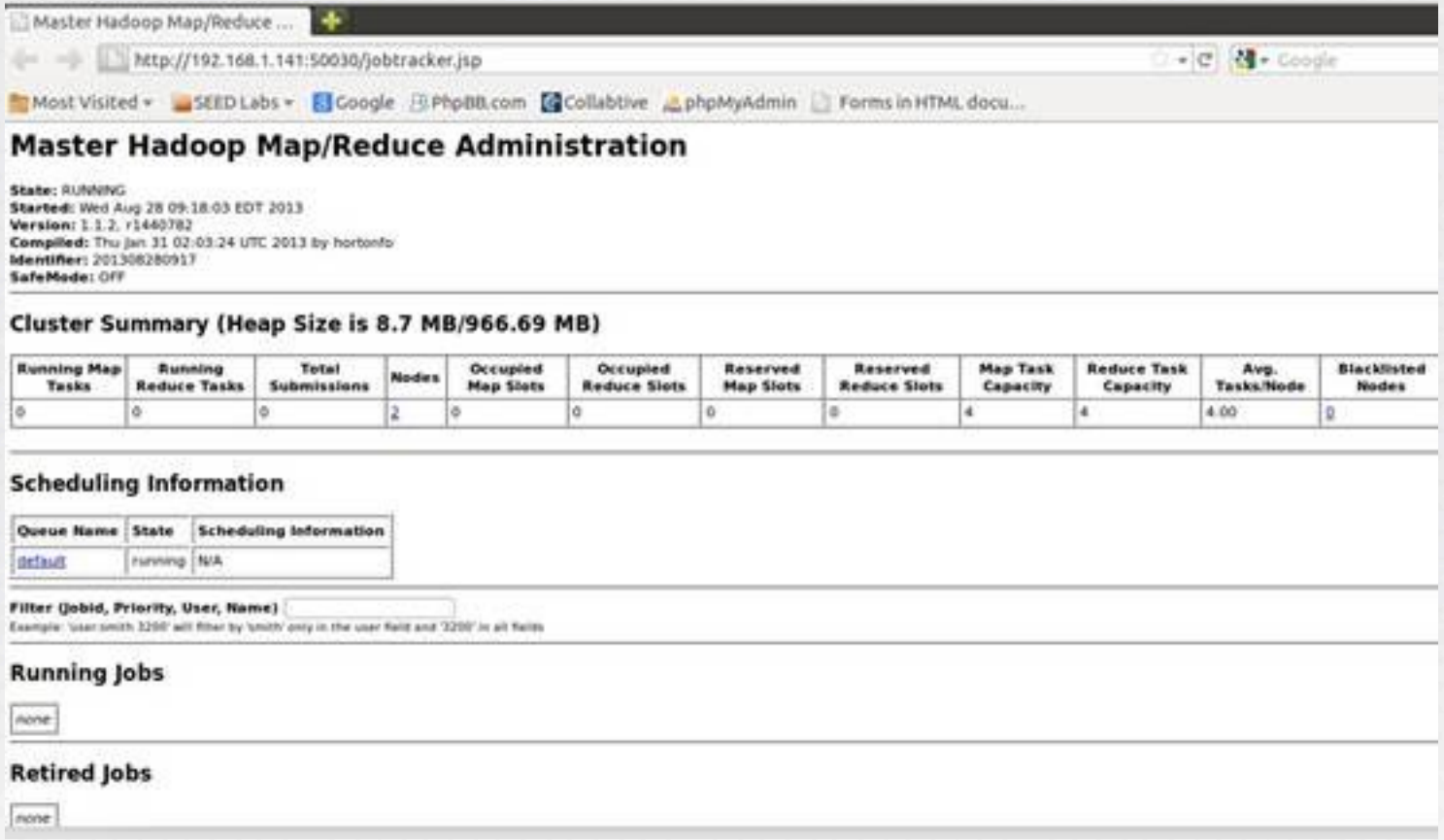
```
[hadoop@Slave2 tmp]$ jps
3454 DataNode
3636 Jps
3532 TaskTracker
[hadoop@Slave2 tmp]$
```


hadoop@Master:/usr/hadoop
$ hadoop dfsadmin -report
Warning: $HADOOP_HOME is deprecated.
Configured Capacity: 24690470912 (22.99 GB)
Present Capacity: 15462825999 (14.4 GB)
DFS Remaining: 15462768640 (14.4 GB)
DFS Used: 57359 (56.01 KB)
DFS Used%: 0%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0

Datanodes available: 2 (2 total, 0 dead)
Name: 192.168.1.137:50010
Decommission Status : Normal
Configured Capacity: 16651051008 (15.51 GB)
DFS Used: 28687 (28.01 KB)
Non DFS Used: 5241806833 (4.88 GB)
DFS Remaining: 11409215488(10.63 GB)
DFS Used%: 0%
DFS Remaining%: 68.52%
Last contact: Wed Aug 28 09:20:28 EDT 2013
Name: 192.168.1.142:50010
Decommission Status : Normal
Configured Capacity: 8039419904 (7.49 GB)
DFS Used: 28672 (28 KB)
Non DFS Used: 3985838080 (3.71 GB)
DFS Remaining: 4053553152(3.78 GB)
DFS Used%: 0%
DFS Remaining%: 50.42%
Last contact: Wed Aug 28 09:20:29 EDT 2013

Web access Hadoop

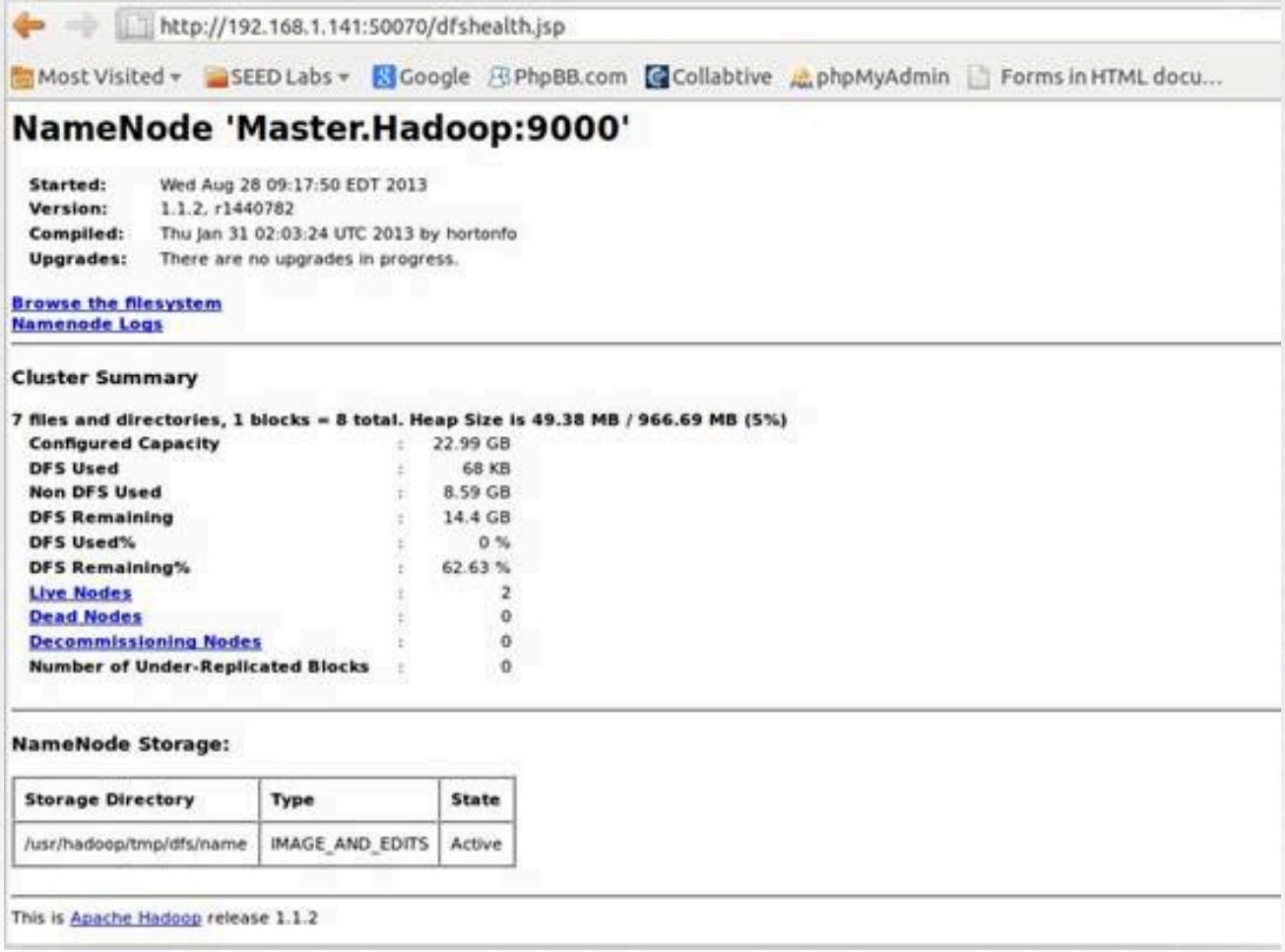
http://192.168.1.141:50030



Web access Hadoop

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

http://192.168.1.141:50070



Installation and configuration

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Local (standalone) mode
- Fully-distributed mode
- **Pseudo-distributed mode**

Please also refer to PDF file Hadoop_installation_and_configuration.pdf