INFORMATION TECHNOLOGY UNIVERSITY

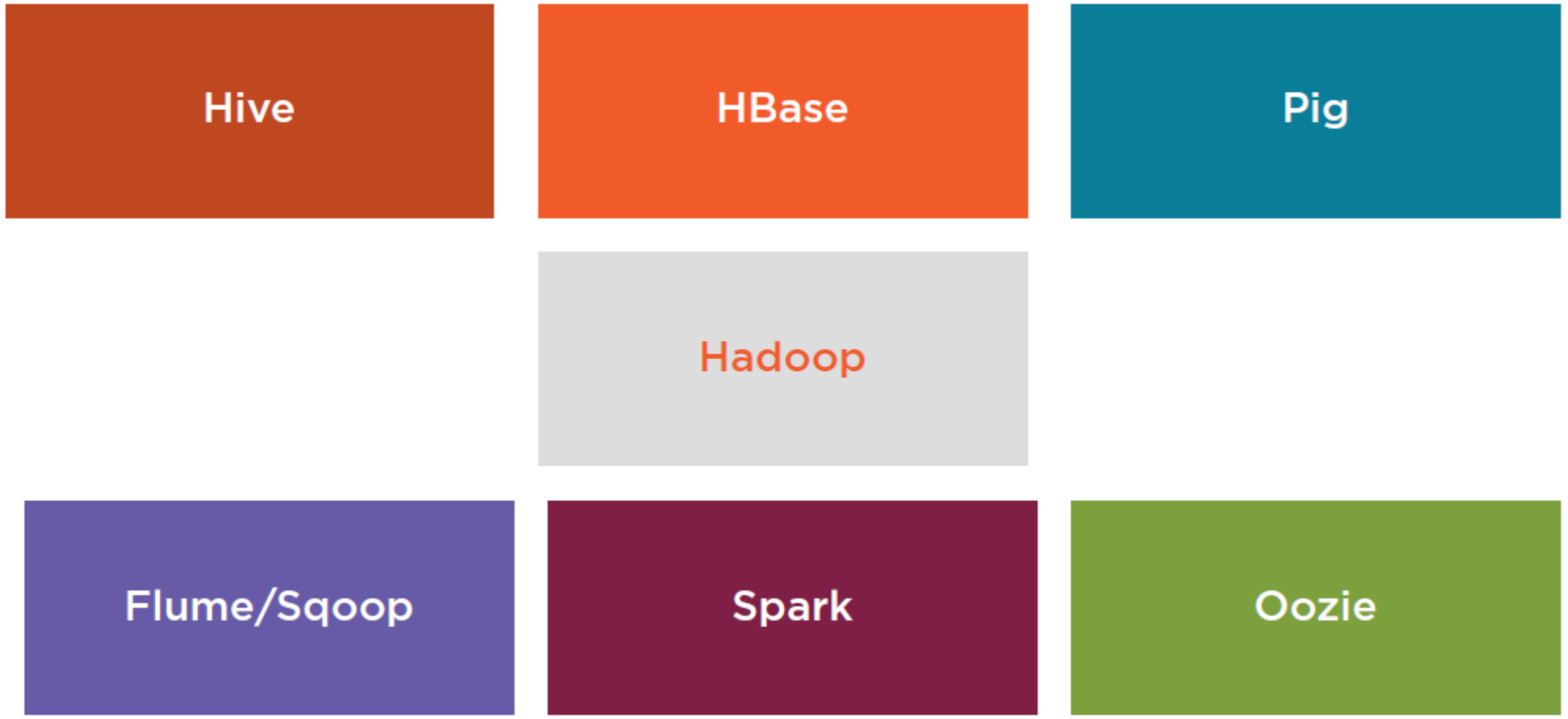
DATA SCIENCE LAB

Lecture 8

CS 537- Big Data Analytics

Dr. Faisal Kamiran

Other Modules – The Hadoop Ecosystem



Hadoop Ecosystem: Hive

- A distributed data warehouse for data that is stored in HDFS
- Provides an SQL interface to Hadoop called HiveQL

Hadoop Ecosystem: HBASE

- A distributed, NoSQL columnar database built on top of Hadoop
- Can store large datasets with flexible schemas
- Key-value store
- Based on Google Big Table
- Can hold extremely large data
- Dynamic data model
- Not a Relational DBMS

Hadoop Ecosystem: Pig

- A data manipulation language named Pig Latin
- Contains an infrastructure layer consisting of a compiler that produces of MapReduce programs

Hadoop Ecosystem: Spark

- A distributed computing engine
- Provides interactive shell to quickly process datasets
- Contains libraries for machine learning, stream processing and graph processing
- Multi-stage in-memory primitives
- provides performance up to 100 times

Hadoop Ecosystem: Oozie

- A workflow management tool that can handle the scheduling and chaining together of Hadoop applications

Hadoop Ecosystem: Sqoop

- Tool to transfer large amounts of data between Hadoop and other systems

Hadoop Ecosystem: Zookeeper

- Provides operational services for a Hadoop cluster group services
- Centralized service for: maintaining configuration information, naming services, providing distributed synchronization and providing group services

Hadoop Ecosystem: Flume

- Distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data

Hadoop Ecosystem: Impala

- Cloudera's open source massively parallel processing (MPP) SQL query engine Apache Hadoop

Hadoop Distributions

Many companies have taken important components from the Hadoop ecosystem and bundled them together into a complete package

Benefits

- Installation
- Packaging
- Maintenance
- Support

Hadoop Distributions

On-premise Distributions

- Cloudera
- Hortonworks
- MapR

Cloud Distributions

- Amazon's Elastic MapReduce
- Azure HDInsight

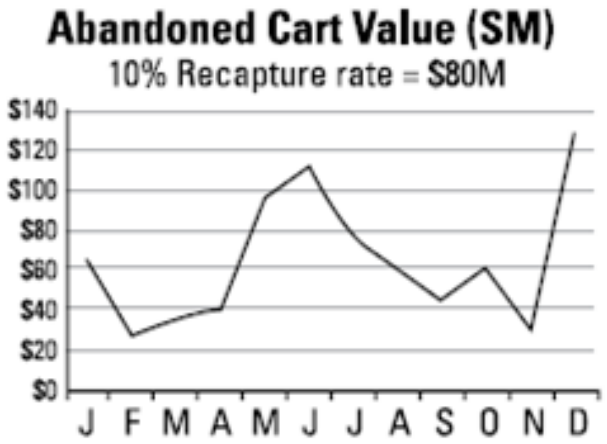
Cloud services offer cost-effective solutions in terms of infrastructure setup and maintenance

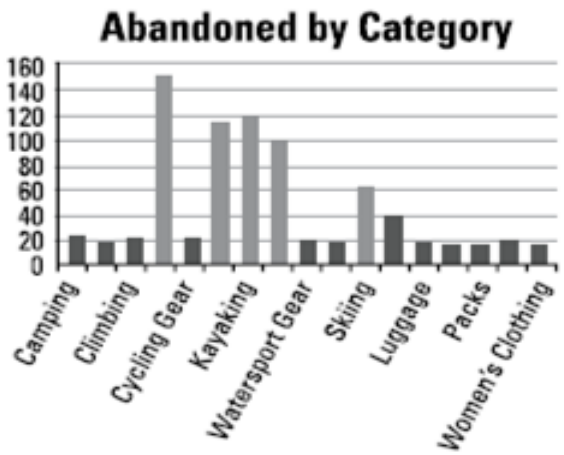
Big Data application with Hadoop – Shopping website log data analysis

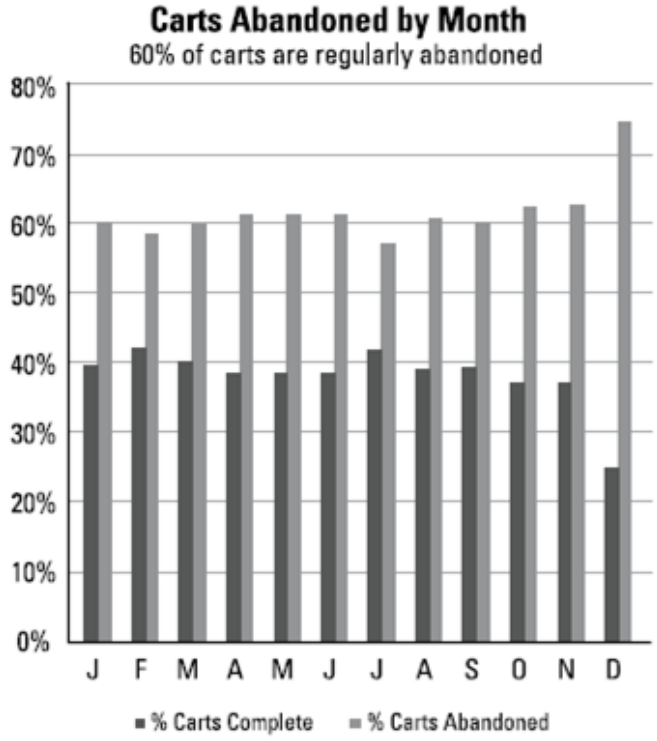
- Shopping websites store clickstream data of its users e.g.
  - A user surfs the site looking for items to buy
  - He views the description of some products
  - He adds some items to his shopping cart and proceeds to buy
  - After seeing the shipping costs, he changes his mind and closes the browser
- Tremendous amount of clickstream data is generated daily
- With traditional systems, this data was retained for a few weeks and then discarded

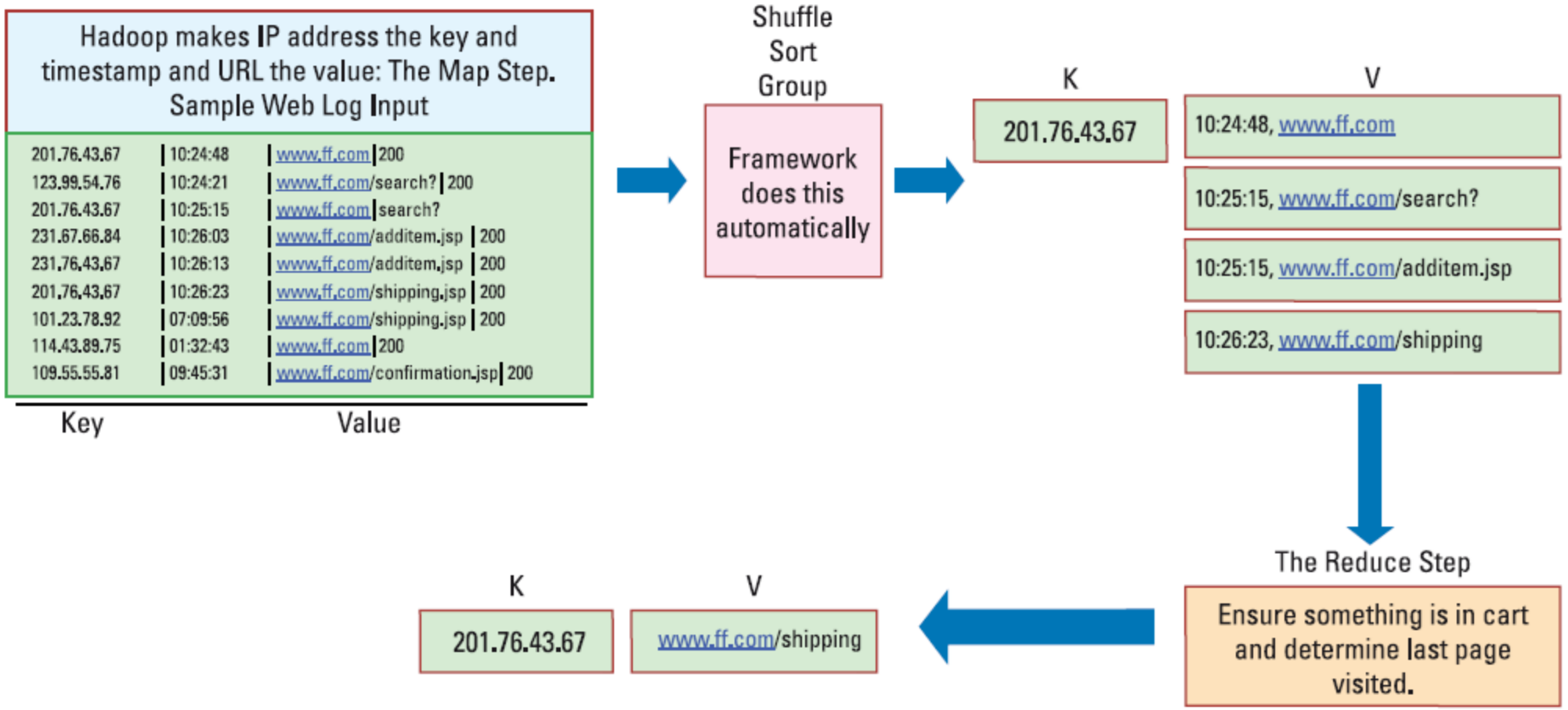
Big Data application with Hadoop – Shopping website log data analysis

- Valuable insights can be gained from this data e.g.
  - "How much revenue can be recaptured if you decrease cart abandonment by 10 percent?"
  - "Are certain products abandoned more than others?"









Building user sessions from clickstream log data and calculating the last page visited for sessions where a shopping cart is abandoned.

Big Data application with Hadoop – Shopping website log data analysis

Hadoop enables this data to be processed quickly with MapReduce and find for each user:

- In the Map stage we find:
  - The final page he visited
  - The list of items in his shopping cart
- In the Reduce stage, aggregations are done to total the number and value of carts abandoned per month and the most common final pages one viewed before abandoning a session

Big Data application with Hadoop – Shopping website log data analysis

Hadoop enables this data to be processed quickly with MapReduce and find for each user:

- In the Map stage we find:
  - The final page he visited
  - The list of items in his shopping cart
- In the Reduce stage, aggregations are done to total the number and value of carts abandoned per month and the most common final pages one viewed before abandoning a session

HDFS

Hadoop Distributed File System

Basic Features: HDFS

- Highly fault-tolerant
- High throughput
- Suitable for applications with large data sets
- Can be built out of commodity hardware

Fault tolerance

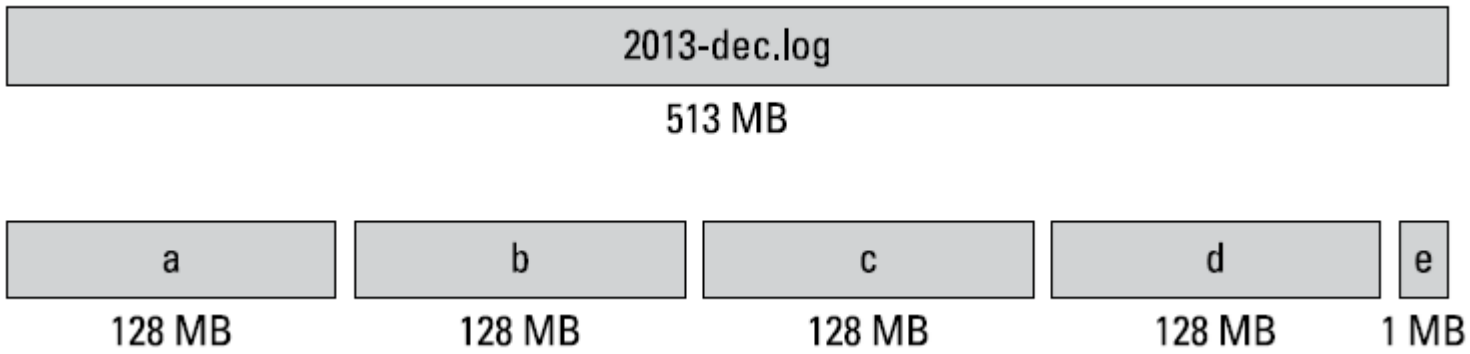
- Failure is the **norm** rather than **exception**
- A HDFS instance may consist of **thousands** of **server machines**, each storing part of the file system's data.
- Since we have huge number of components and that each component has **non-trivial probability** of failure means that there is always some component that is non-functional.
- **Detection** of **faults** and **quick**, automatic **recovery** from them is a core architectural goal of HDFS.

Data Characteristics

- **Batch processing** rather than interactive user access.
- Large data sets and files: gigabytes to **terabytes** size
- High aggregate data **bandwidth**
- **Scale** to hundreds of nodes in a cluster
- Tens of millions of files in a single instance
- **Write-once-read-many:** a file once created, written and closed need not be changed – this assumption simplifies coherency
- A **map-reduce** application application fits perfectly with this model.

Data Blocks

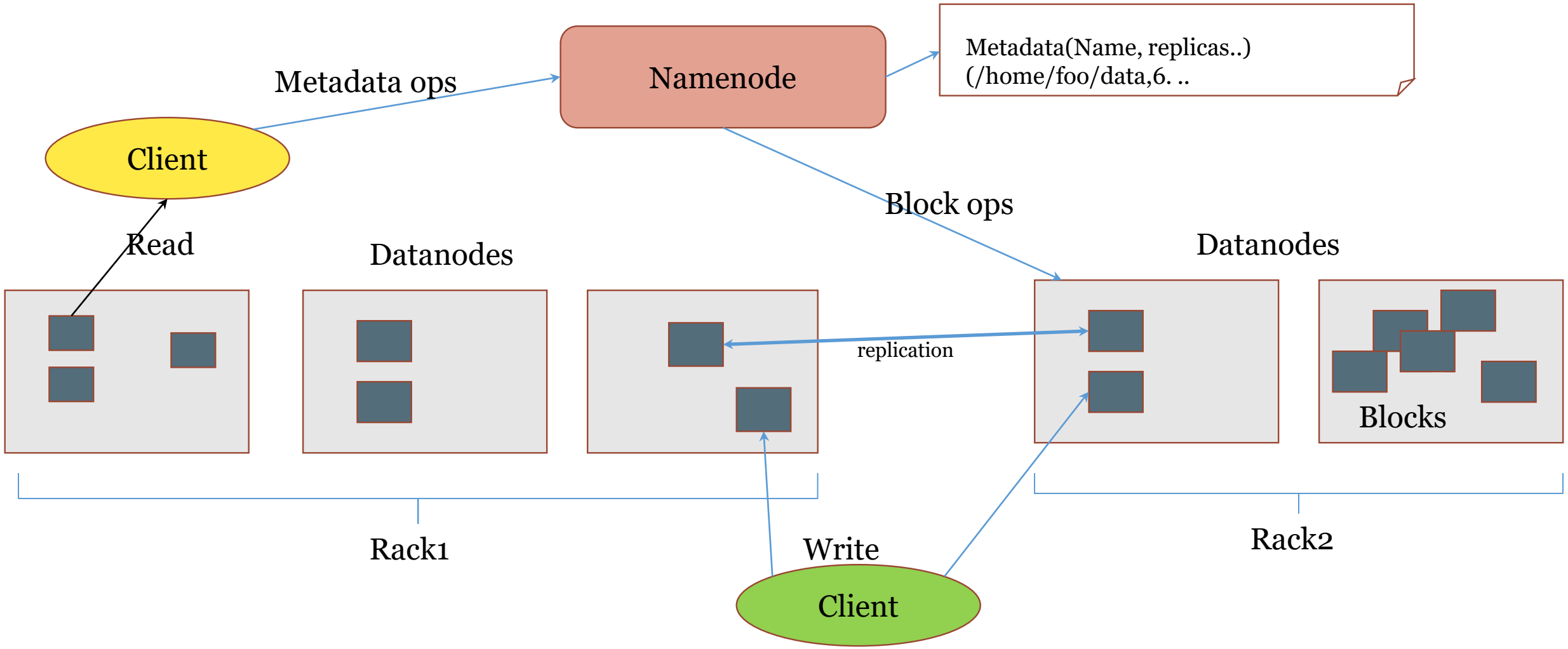
- HDFS support write-once-read-many with reads at high speeds.
- A data file is split into blocks.
- A typical block size is 128MB.
- A file is chopped into 128MB chunks and stored.



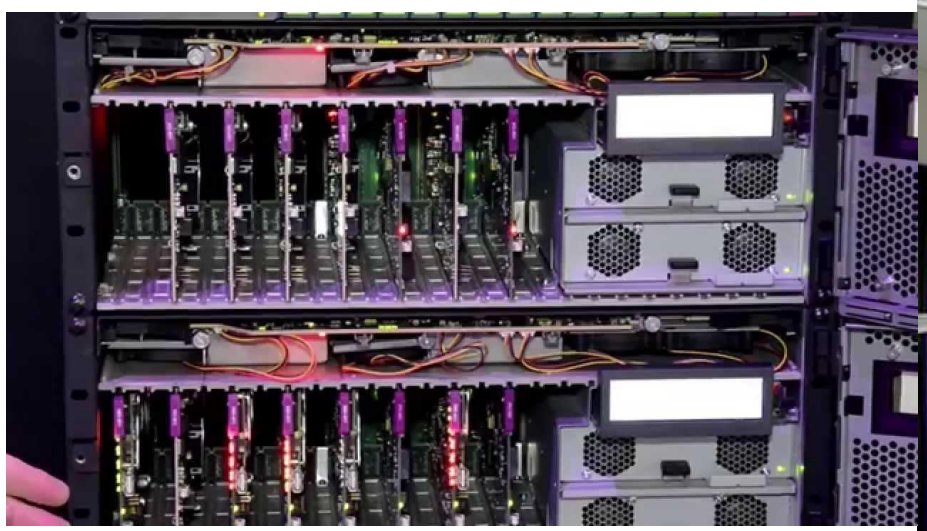
Namenode and Datanodes

- Master/slave architecture
- HDFS cluster consists of a single Namenode, a master server that manages the file system namespace and regulates access to files by clients.
- There are a number of DataNodes usually one per node in a cluster.
- The DataNodes manage storage attached to the nodes that they run on.
- HDFS exposes a file system namespace and allows user data to be stored in files.
- A file is split into one or more blocks and set of blocks are stored in DataNodes.
- DataNodes: serves read, write requests, performs block creation, deletion, and replication upon instruction from Namenode.

HDFS Architecture

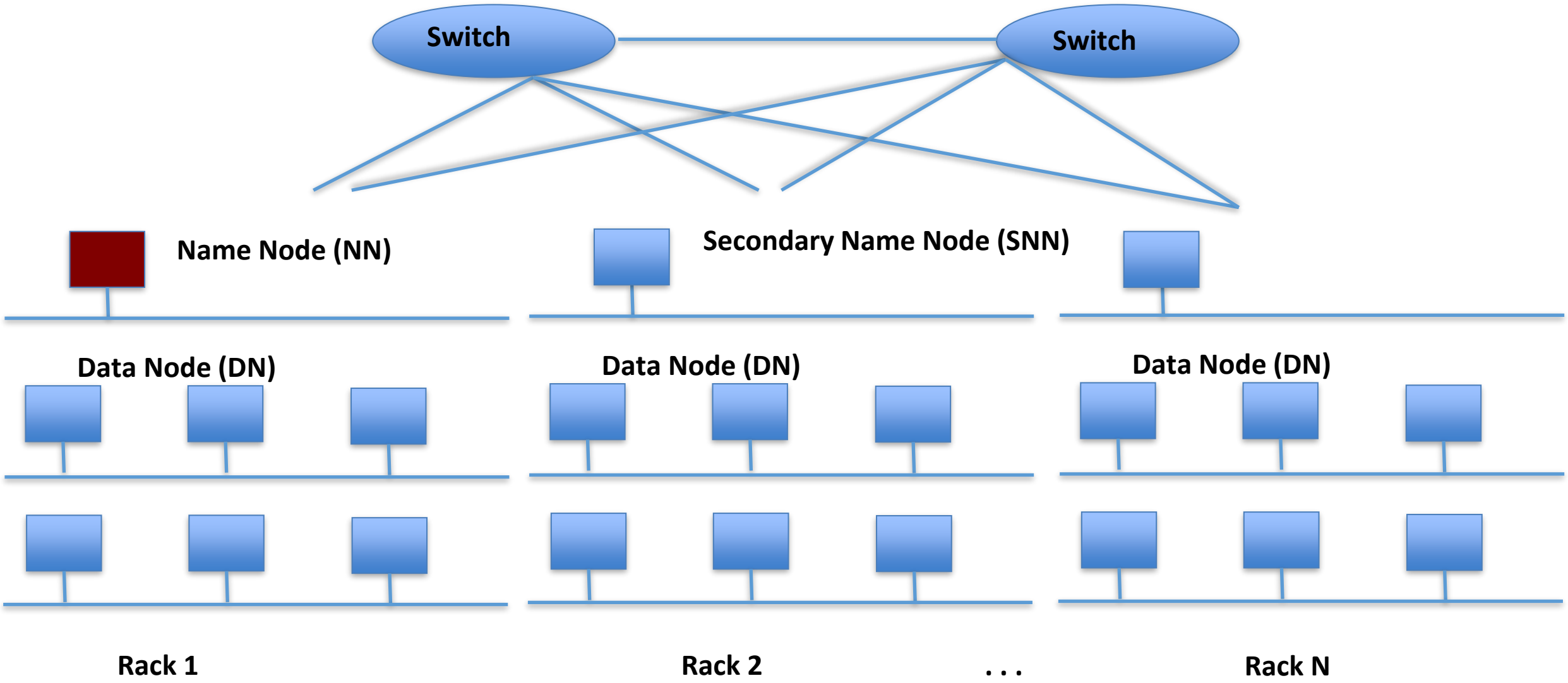


Data Center Environment

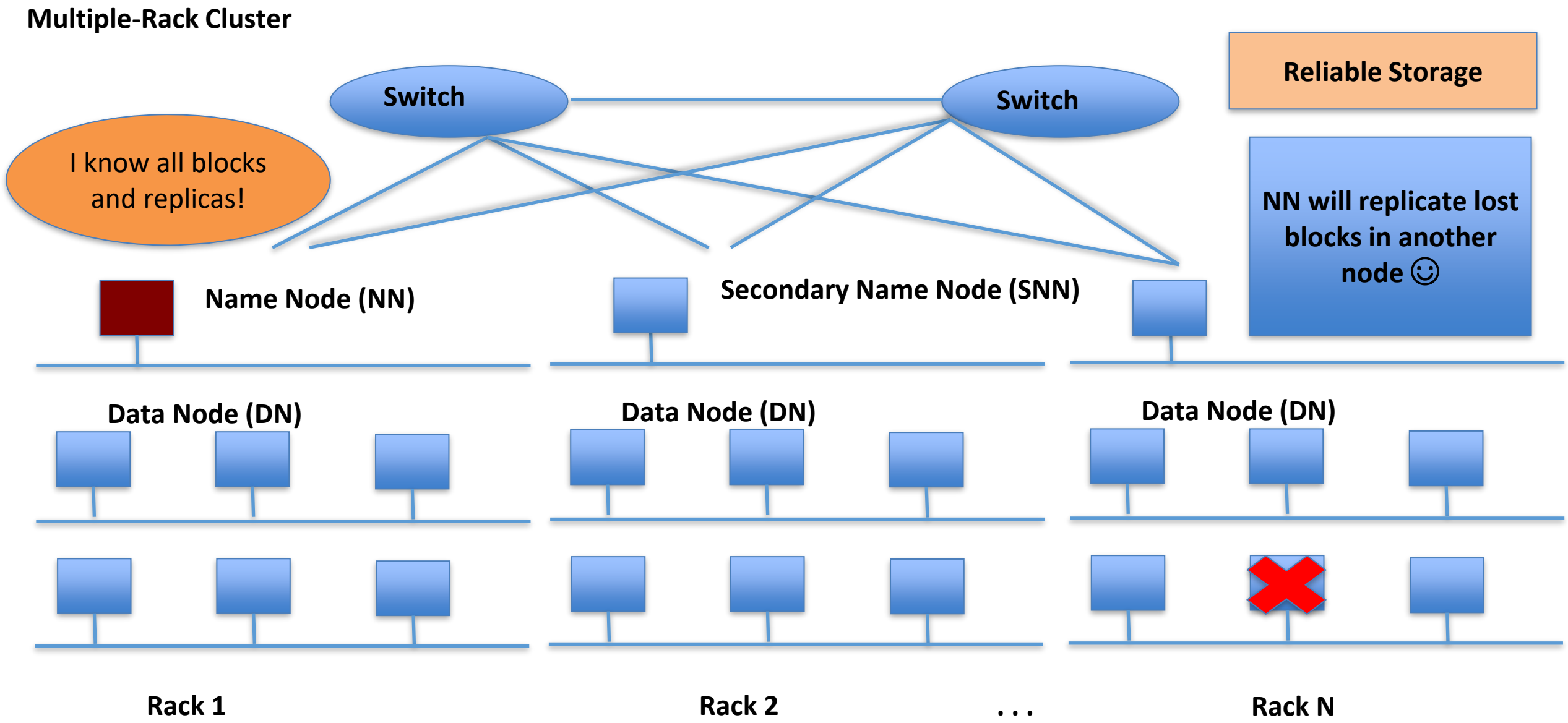


HDFS Architecture: Master-Slave

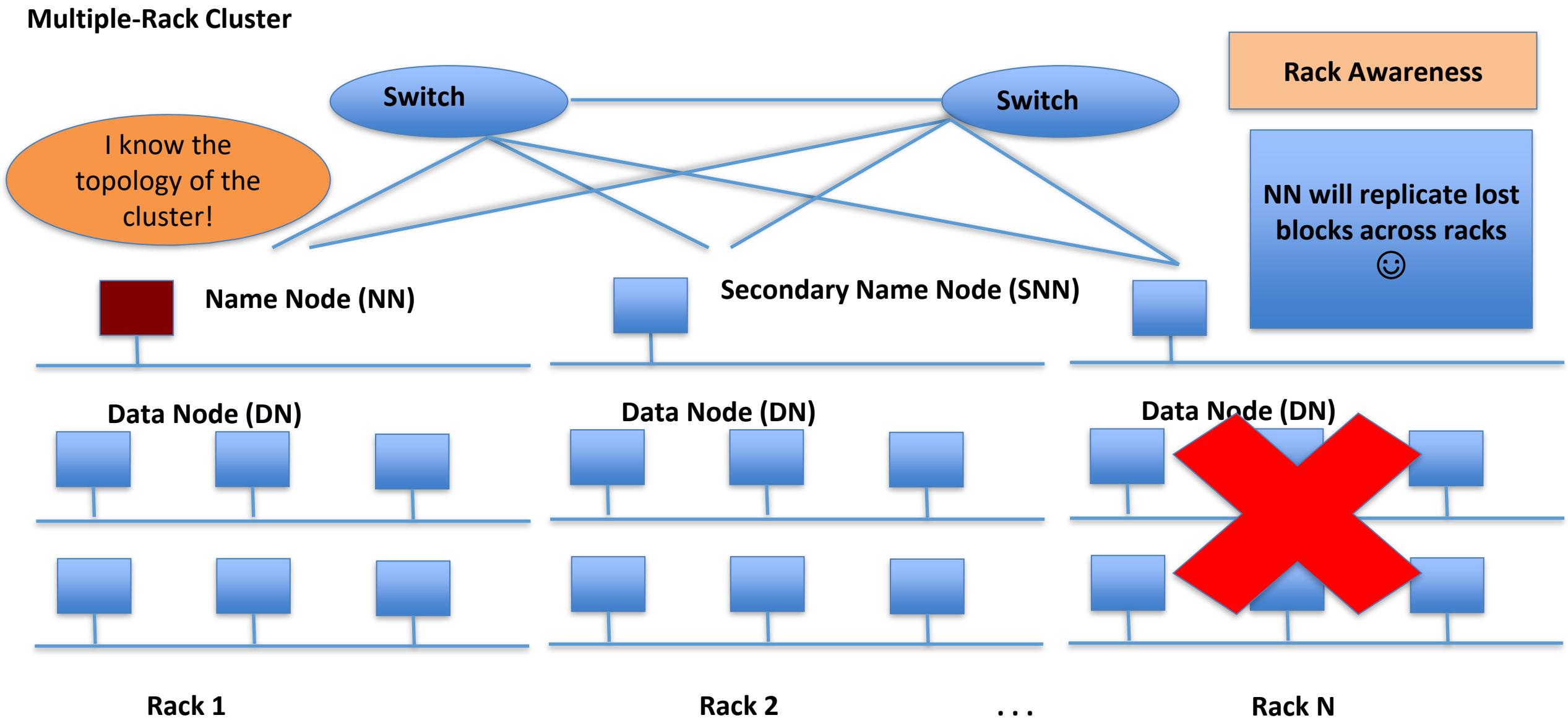
Multiple-Rack Cluster



HDFS Architecture: Master-Slave



HDFS Architecture: Master-Slave



File system Namespace

- Hierarchical file system with directories and files
- Create, remove, move, rename etc.
- Namenode maintains the file system
- Any meta information changes to the file system recorded by the Namenode.
- An application can specify the number of replicas of the file needed: replication factor of the file. This information is stored in the Namenode.

Data Replication

- HDFS is designed to store very large files across machines in a large cluster.
- Each file is a sequence of blocks.
- All blocks in the file except the last are of the same size.
- Blocks are replicated for fault tolerance.
- The Namenode receives a Heartbeat and a BlockReport from each DataNode in the cluster.
- BlockReport contains all the blocks on a Datanode.

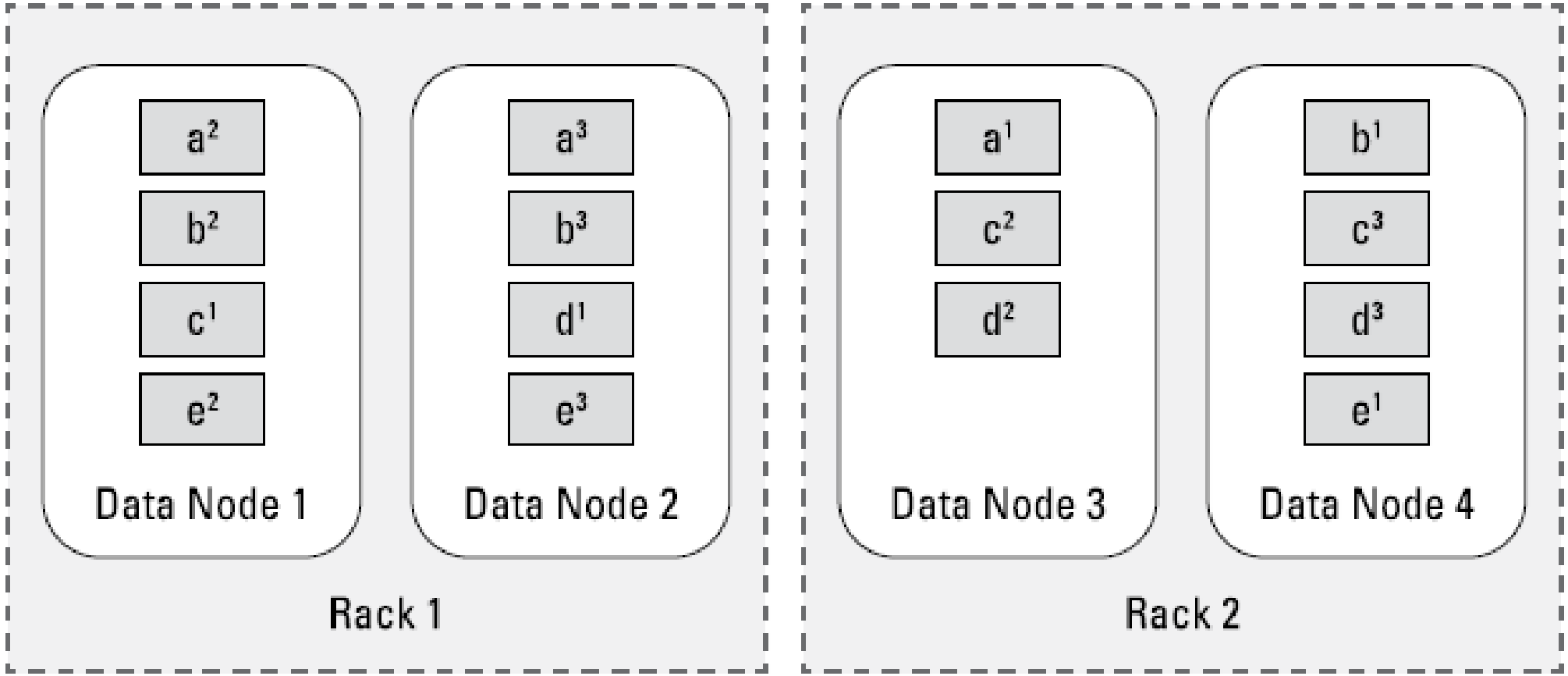
Replica Placement

- The placement of the replicas is critical to HDFS reliability and performance.
- Optimizing replica placement distinguishes HDFS from other distributed file systems.
- Rack-aware replica placement:
  - Goal: improve reliability, availability and network bandwidth utilization
- Many racks, communication between racks are through switches.
- Network bandwidth between machines on the same rack is greater than those in different racks.
- Namenode determines the rack id for each DataNode.

Replica Placement

- Replicas are typically placed on unique racks
  - Simple but non-optimal
  - Writes are expensive
  - Replication factor is 3
- Replicas are placed: one on a node in a local rack, one on a different node in the local rack and one on a node in a different rack.
  - 1/3 of the replica on a node, 2/3 on a rack and 1/3 distributed evenly across remaining racks.

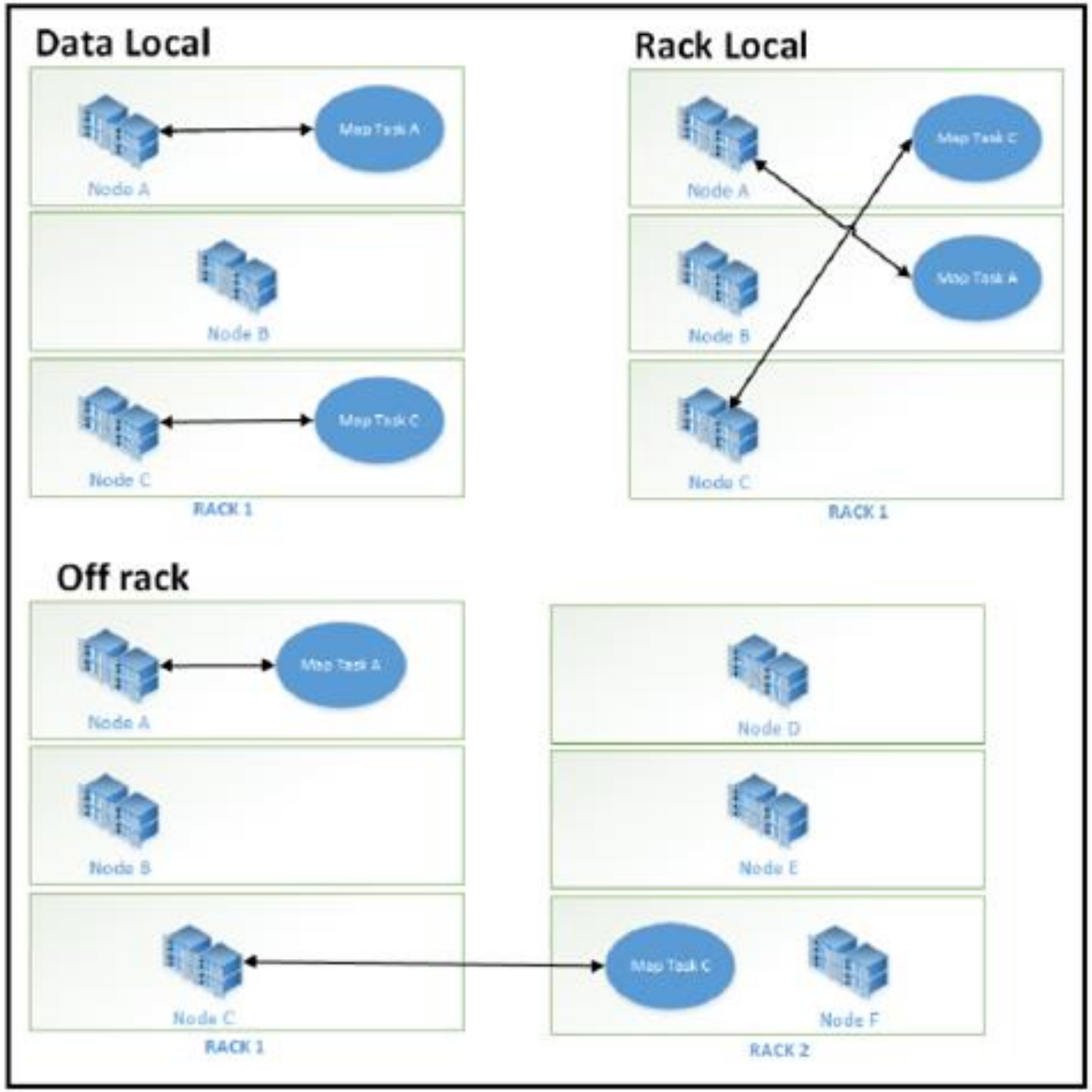
Data Replication Patterns



Replica Selection

- Replica selection for READ operation: HDFS tries to minimize the bandwidth consumption and latency.
- Replica selection criteria
  - **Data Local:** Replica available on the same node
  - **Rack Local:** Replica available on the same rack
  - **Off Rack:** Replica available on different rack
- Replica selection preference:
  Data Local > Rack Local > Off Rack (Worst Case)
- HDFS cluster may span multiple data centers: replica in the local data center is preferred over the remote one.

Replica Selection



That’s all for today.