BITS Pilani presentation

BITS Pilani
Hyderabad Campus

D. Powar
Lecturer,
BITS-Pilani, Hyderabad Campus

innovate achieve lead

BITS Pilani
Hyderabad Campus

SSZG527

Cloud Computing

Agenda:

- Introduction to file system
- Distributed File System (DFS)
- Case Studies
  - GFS
  - HDFS
  - Reading and writing files (HDFS)
  - Cloud storage

File system

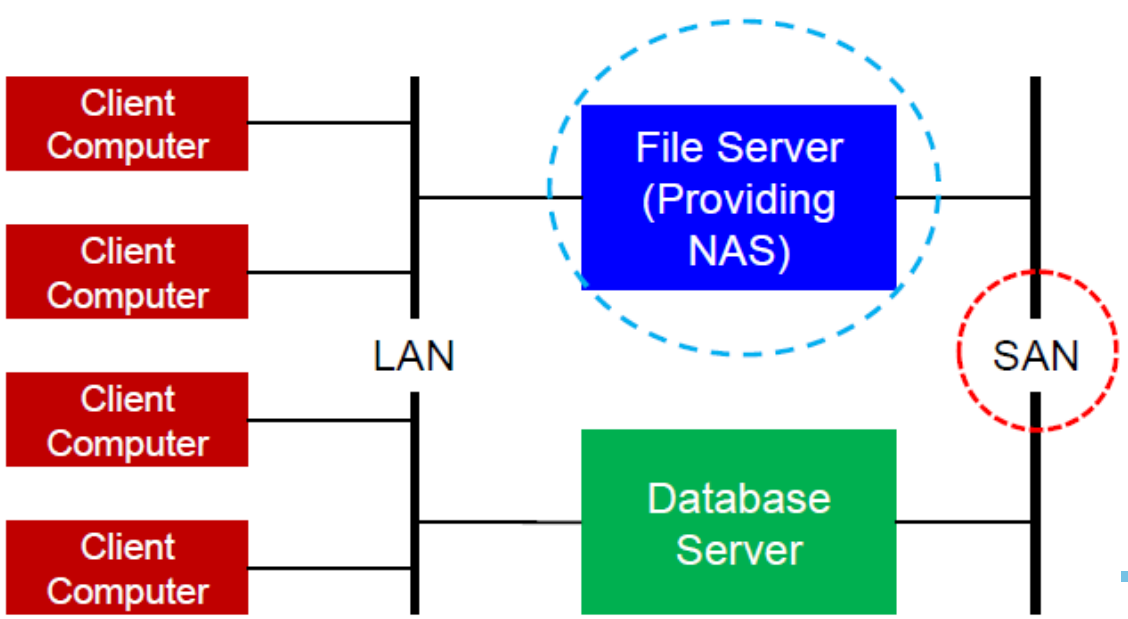
- **File system** is a type of data store which can be used to
  - organize data (as files)
  - provide a means for applications to *store, access, and modify* data
- Windows makes use of the FAT, NTFS, exFAT and ReFS file systems (the latter is only supported and usable in Windows Server 2012)
- Ext, ext2, ext3, ext4 for linux
- HFS, HFS+ for Mac OS
- CDFS
- GFS/HDFS

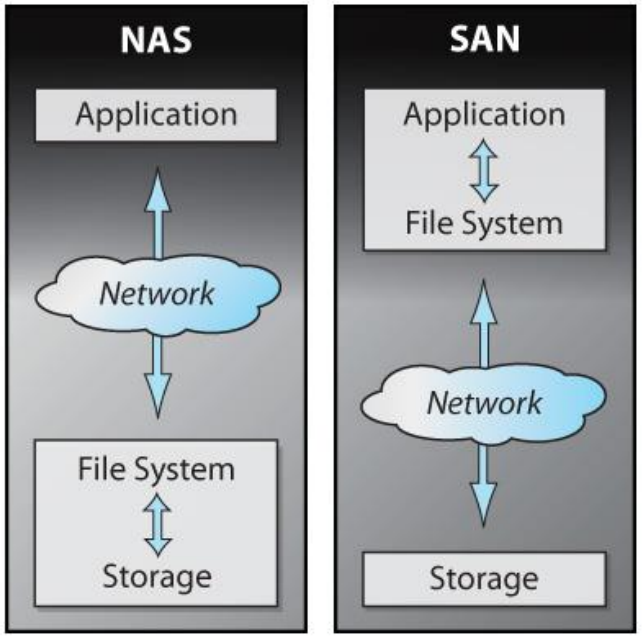
Distributed File System (DFS)?

- Big data continues to grow
- In contrary to a local file system, a distributed file system (DFS) can hold big data and provide access to this data to many clients distributed across a network

NAS versus SAN

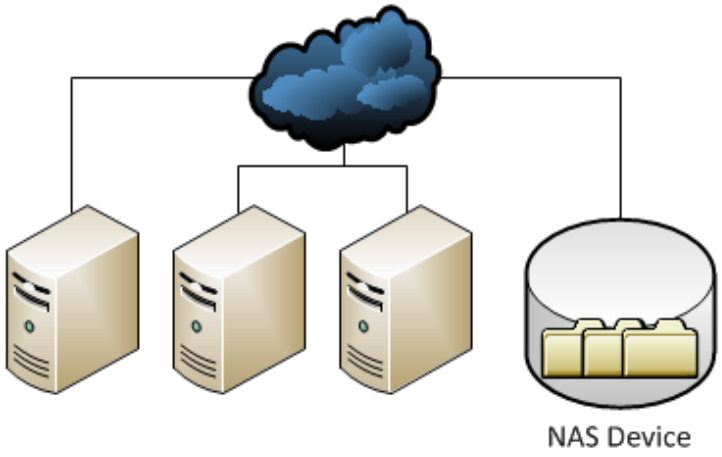
- Another term for DFS is *network attached storage* (NAS), referring to attaching storage to network servers that provide file systems
- A similar sounding term that refers to a very different approach is *storage area network* (SAN)
  - SAN makes storage devices (not file systems) available over a network





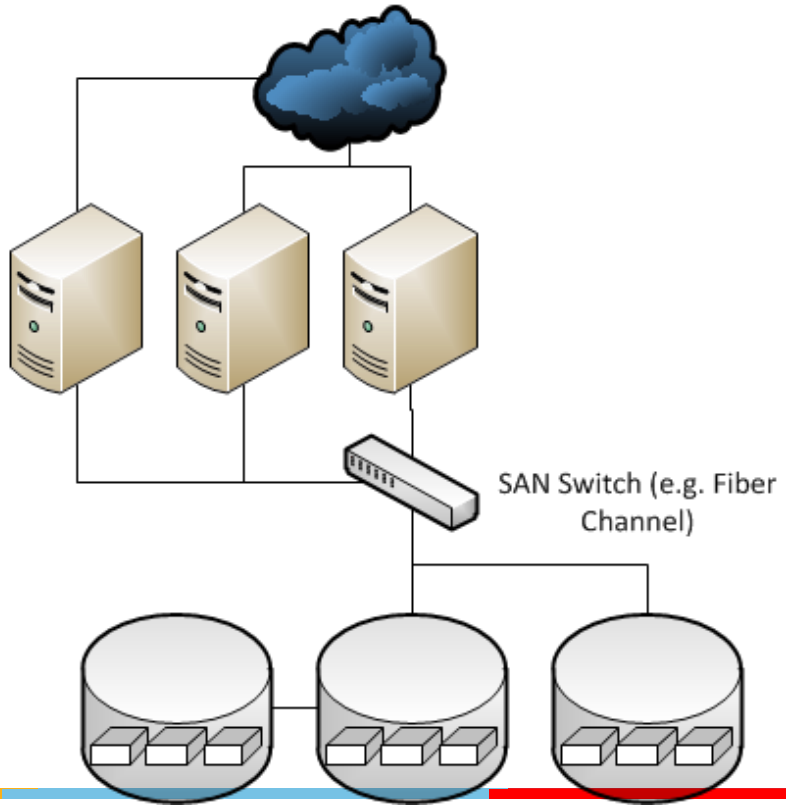
Network Attached Storage

- Shared storage over shared network
- File system
- Easier management



Storage Area Network

- Shared storage over dedicated network
- Raw storage
- Fast, but costly



Benefits of DFSs

DFSs provide:

1. File sharing over a network: without a DFS, we would have to exchange files by e-mail or use applications such as the Internet's FTP

2. Transparent files accesses: A user's programs can access remote files as if they are local. The remote files have no special APIs; they are accessed just like local ones

3. Easy file management: managing a DFS is easier than managing multiple local file systems

DFS Architectures

1. Client-Server Distributed File Systems
2. Cluster-Based Distributed File Systems
3. Symmetric Distributed File Systems

Cluster-Based Distributed File Systems

- The cluster-based file system is a key component for providing scalable data-intensive application performance
- The cluster-based file system divides and distributes big data, using file striping techniques, for allowing concurrent data accesses
- The cluster-based file system could be either a cloud computing or an HPC oriented distributed file system
  - GFS, HDFS, S3, etc are examples of cloud computing DFSs
  - Parallel Virtual File System (PVFS) and IBM's General Parallel File System (GPFS) are examples of HPC DFSs

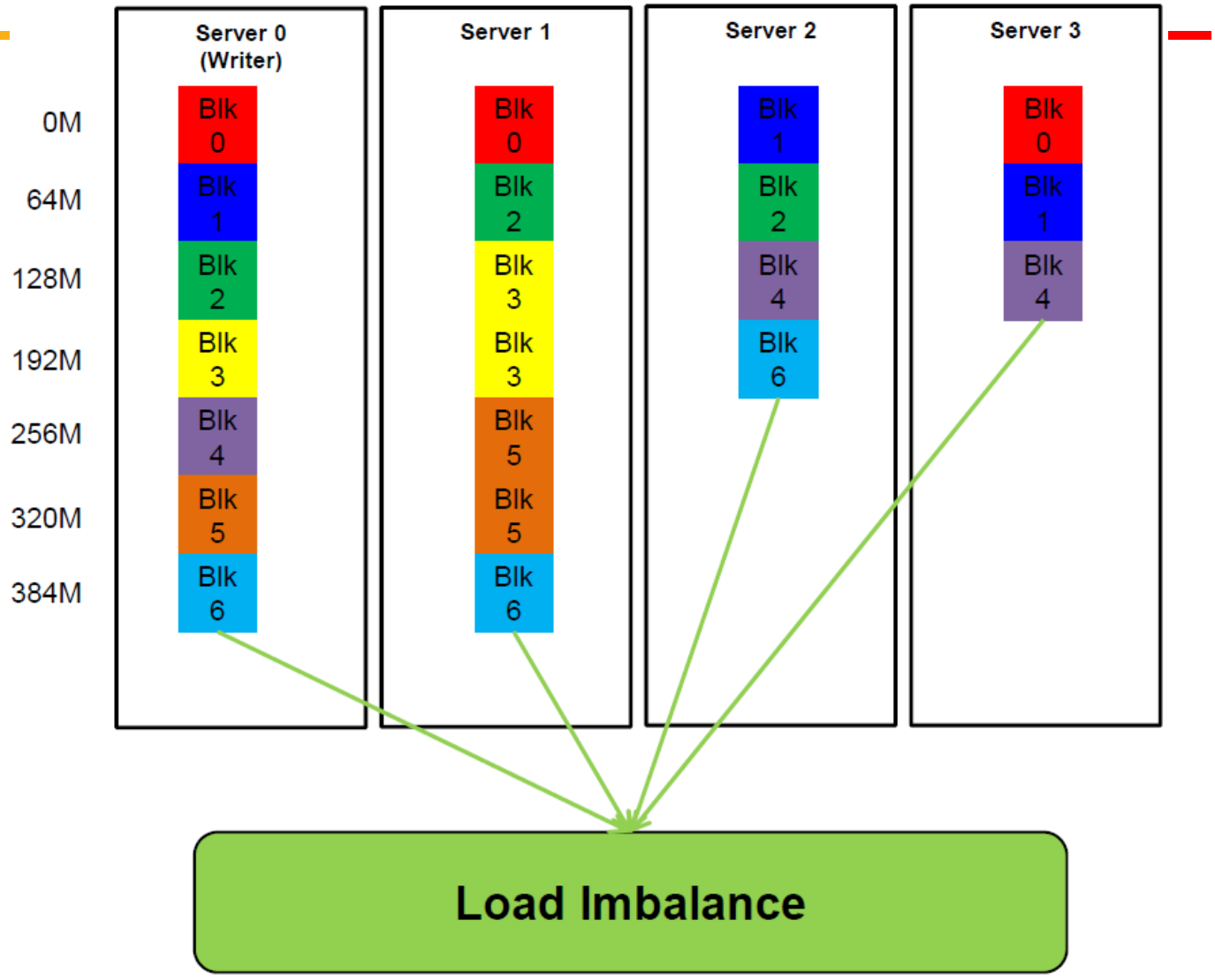
Google File System (GFS)

- The Google File System (GFS) is a cloud computing based scalable DFS for large distributed data intensive applications
- GFS divides large files into multiple pieces called chunks or blocks (by default 64MB) and stores them on different data servers
  - This design is referred to as block-based design
- Each GFS chunk has a unique 64-bit identifier and is stored as a file in the lower layer local file system on the data server
- GFS distributes chunks across cluster data servers using a random distribution policy

Google File System

- GFS stores a huge number of files, totaling many terabytes of data
- Individual file characteristics
  - Very large, multiple gigabytes per file
  - Files are updated by appending new entries to the end (faster than overwriting existing data)
  - Files are virtually never modified (other than by appends) and virtually never deleted.
  - Files are mostly read-only

GFS Random Distribution Policy

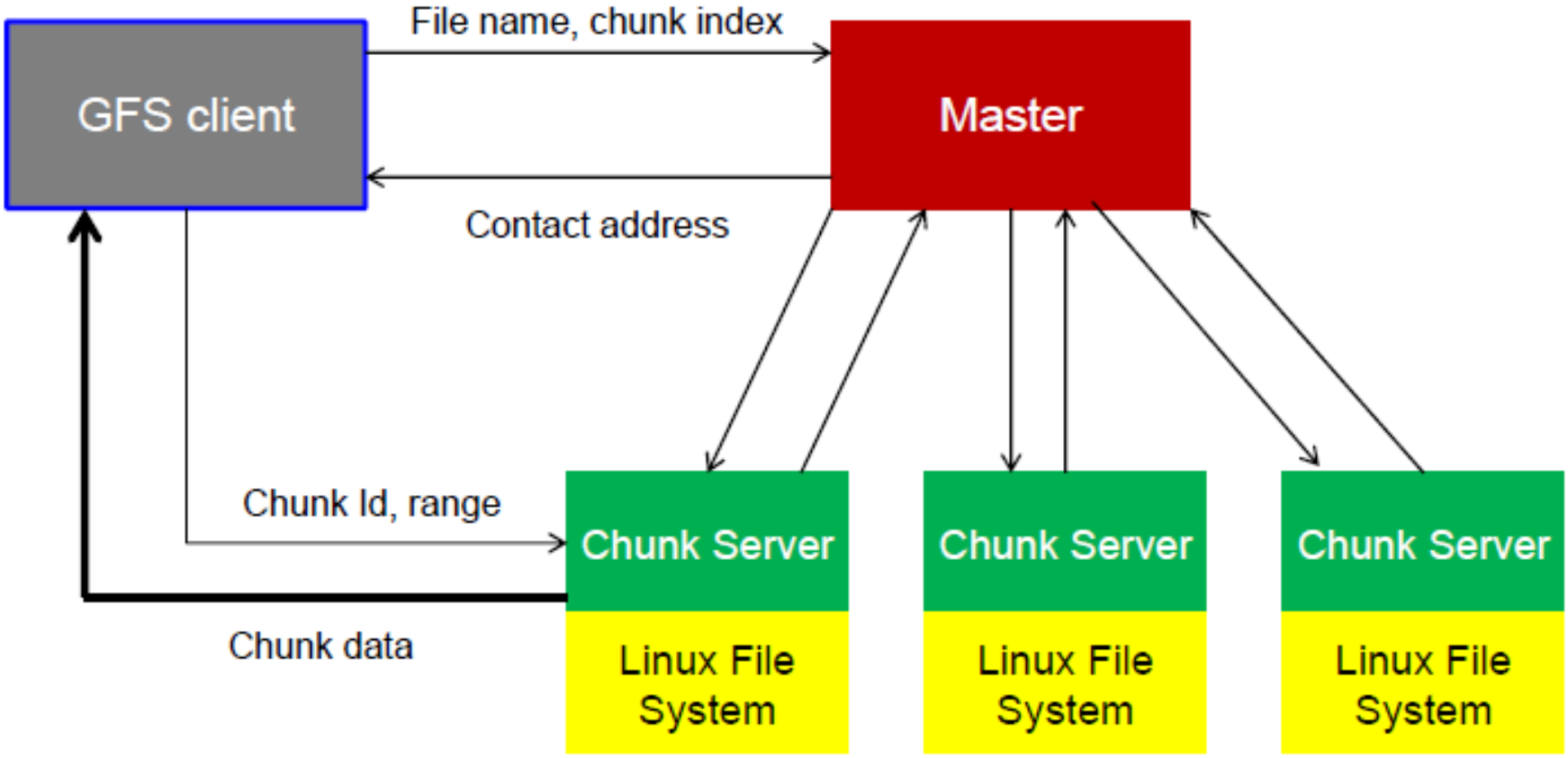


How to replicate?

- Two machines in the same rack have more bandwidth and lower latency between each other than two machines in two different racks

- **For every block of data, two copies will exist in one rack, another copy in a different rack.**

GFS Architecture



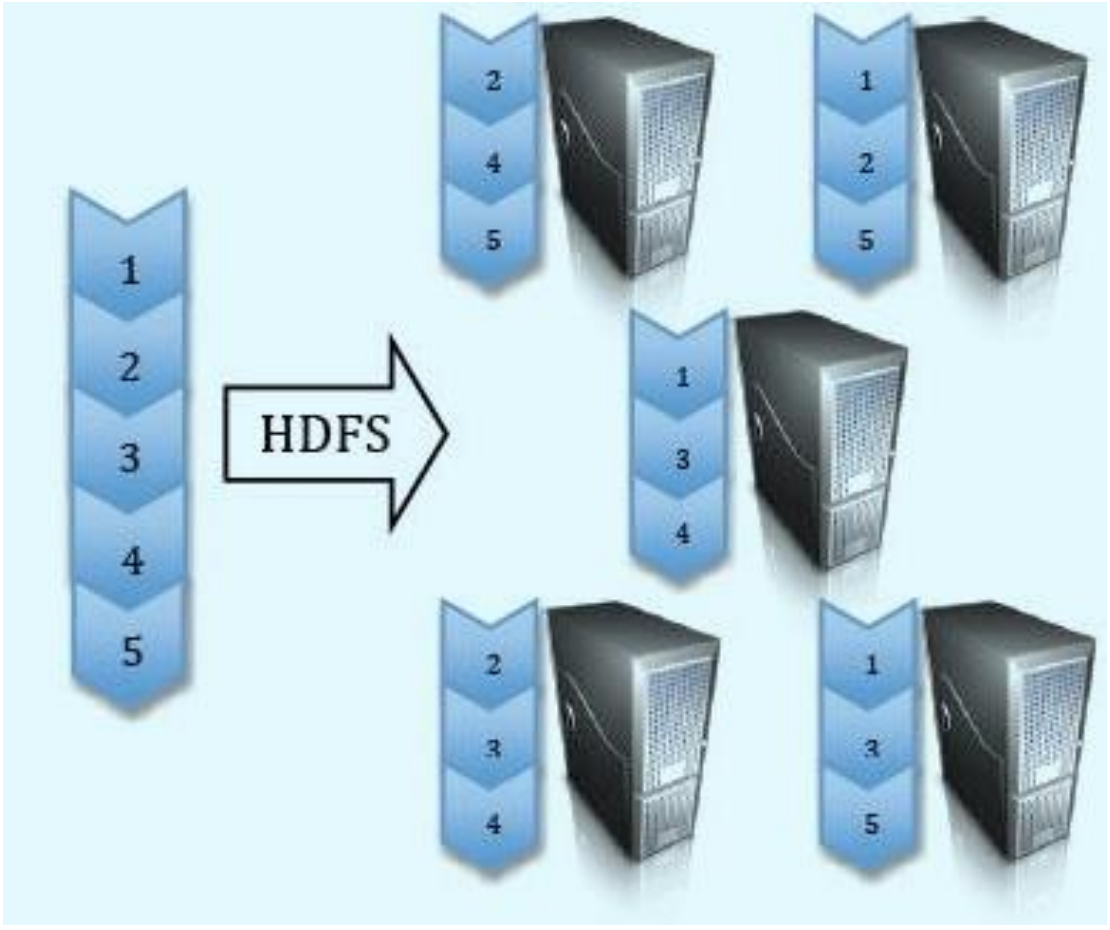
Master and Chunk Server Responsibilities

- **Master Node**
  - Holds all metadata
    - Namespace
    - Current locations of chunks
    - All in RAM for fast access
  - Manages chunk leases to chunk servers
  - Garbage collects orphaned chunks
  - Migrates chunks between chunk servers
  - Polls chunk servers at startup
  - Use heartbeat messages to monitor servers
- **Chunk Servers**
  - Simple
  - Stores Chunks as files
  - Chunks are 64MB size
  - Chunks on local disk using standard filesystem
  - Read write requests specify chunk handle and byte range
  - Chunks replicated on configurable chunk servers

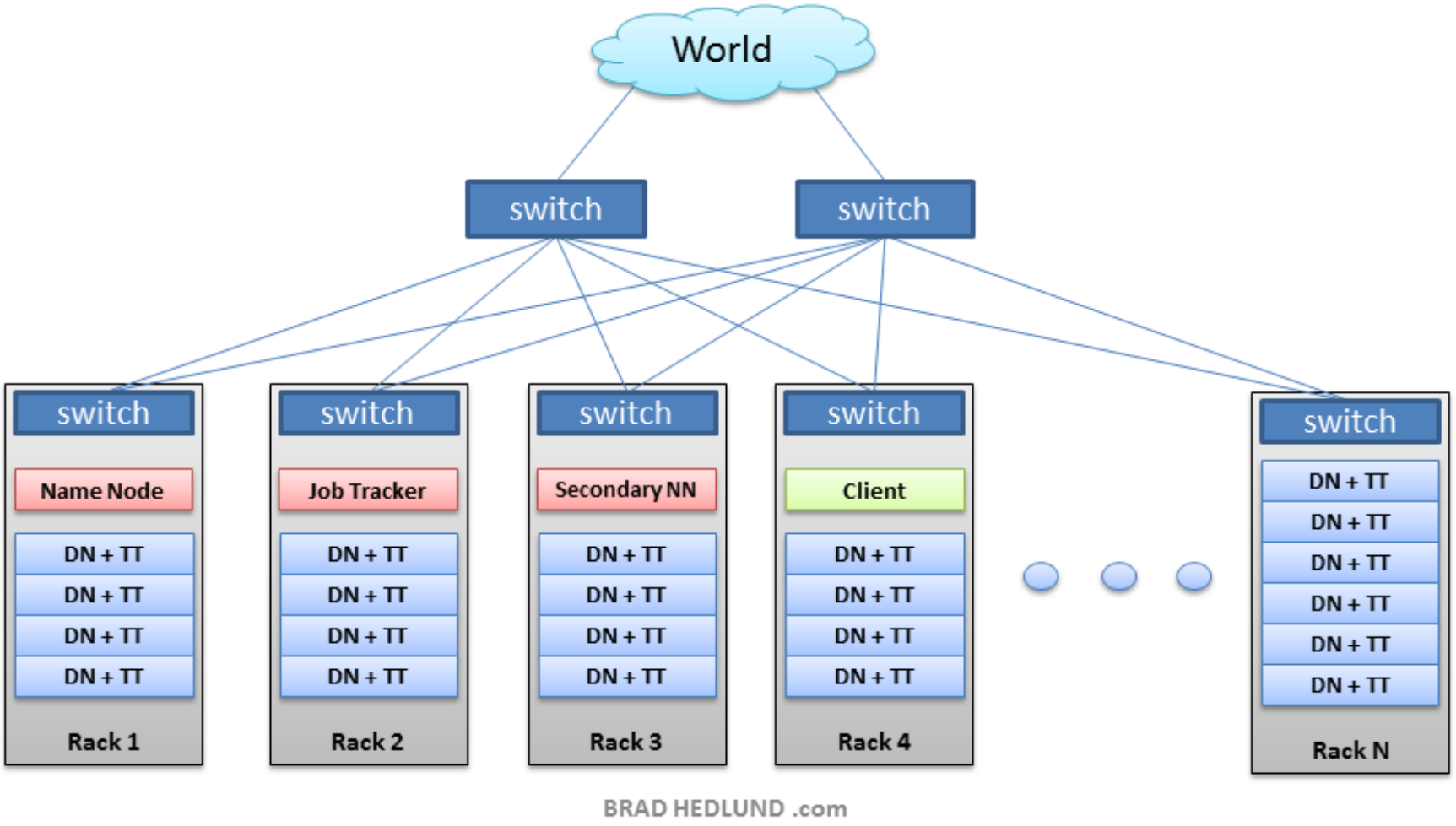
GFS

- Chunks are replicated within a cluster for fault tolerance, using a primary/backup scheme.
- Periodically the master polls all its chunk servers to find out which chunks each one stores
  - This means the master doesn't need to know each time a new server comes on board, when servers crash, etc.
- Polling occurs often enough to guarantee that master's information is "good enough".

Hadoop File system



Hadoop Cluster



Source: http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network

HDFC??

- Hadoop's Distributed File System is designed to reliably store very large files across machines in a large cluster.
- It is inspired by the Google File System.
- Hadoop DFS stores each file as a sequence of blocks, all blocks in a file except the last block are the same size.
- Blocks belonging to a file are replicated for fault tolerance. The block size and replication factor are configurable per file. Files in HDFS are "write once" and have strictly one writer at any time.

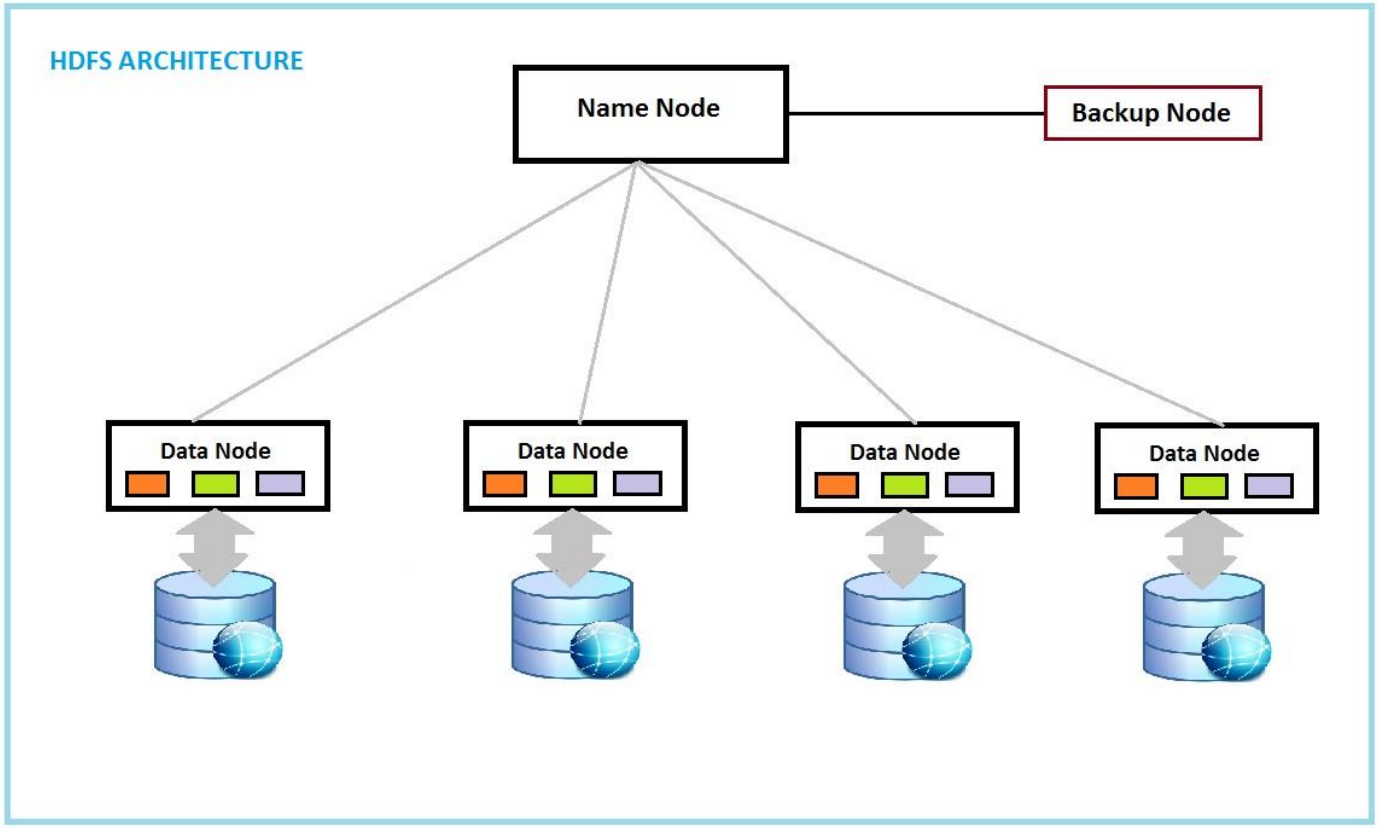
Hadoop Distributed File System – Goals:

- Store large data sets
- Cope with hardware failure
- Emphasize streaming data access

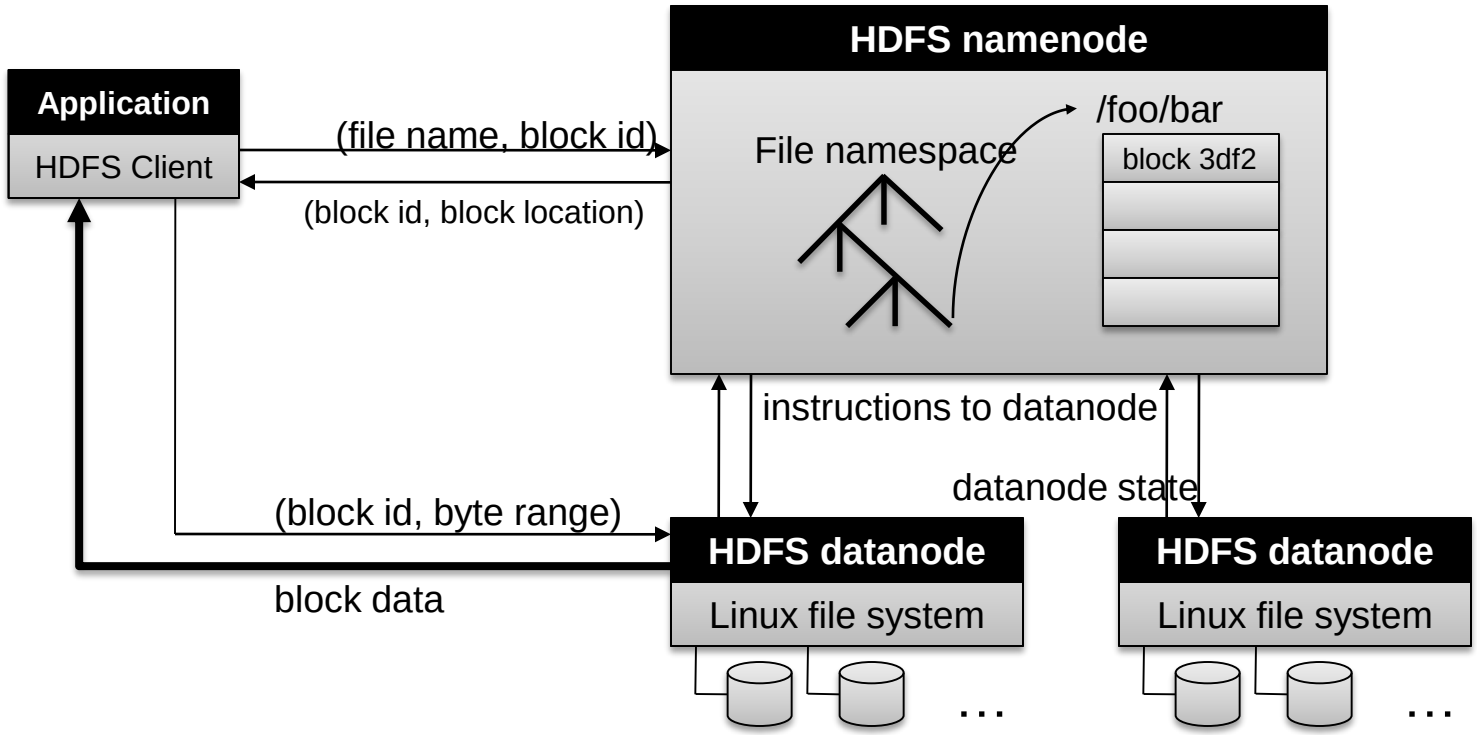
From GFS to HDFS

Terminology differences:

- GFS master = Hadoop namenode
- GFS chunkservers = Hadoop datanodes



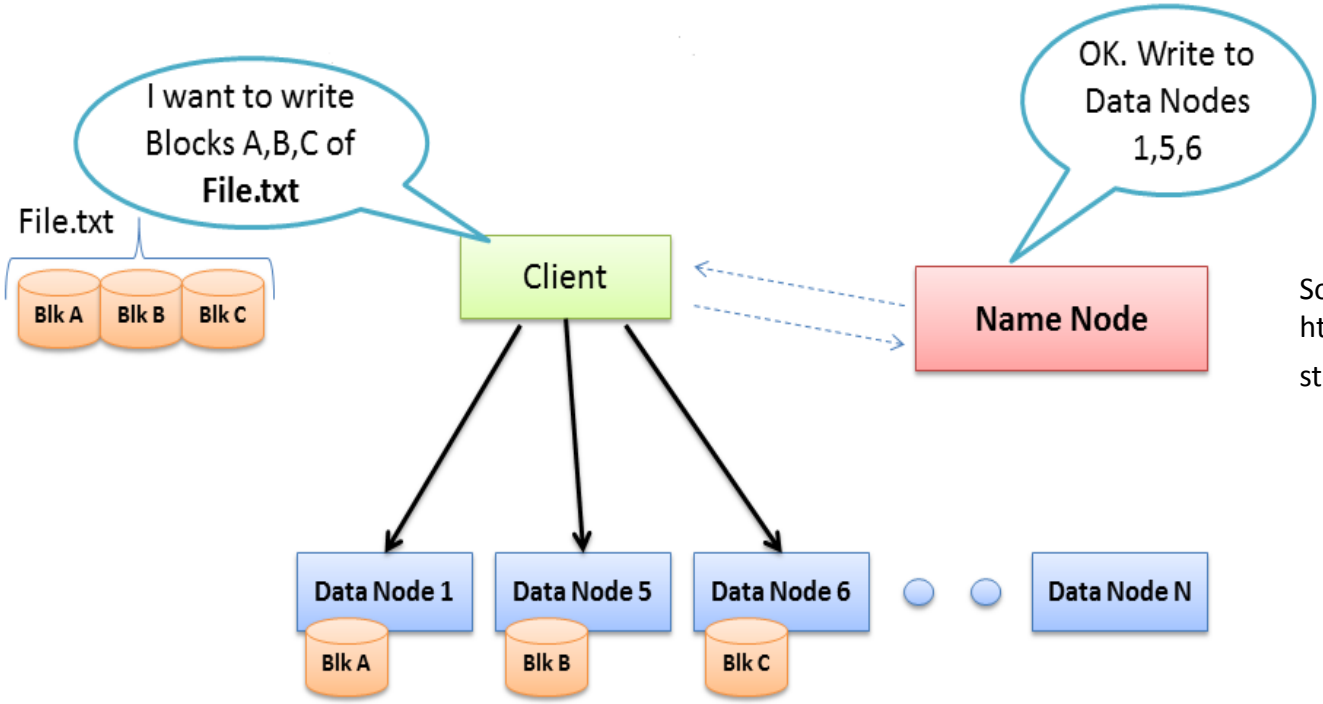
HDFS Architecture



Namenode Responsibilities

- Managing the file system namespace:
  - Holds file/directory structure, metadata, file-to-block mapping, access permissions, etc.
- Coordinating file operations:
  - Directs clients to datanodes for reads and writes
  - No data is moved through the namenode
- Maintaining overall health:
  - Periodic communication with the datanodes
  - Garbage collection

Writing files to HDFS

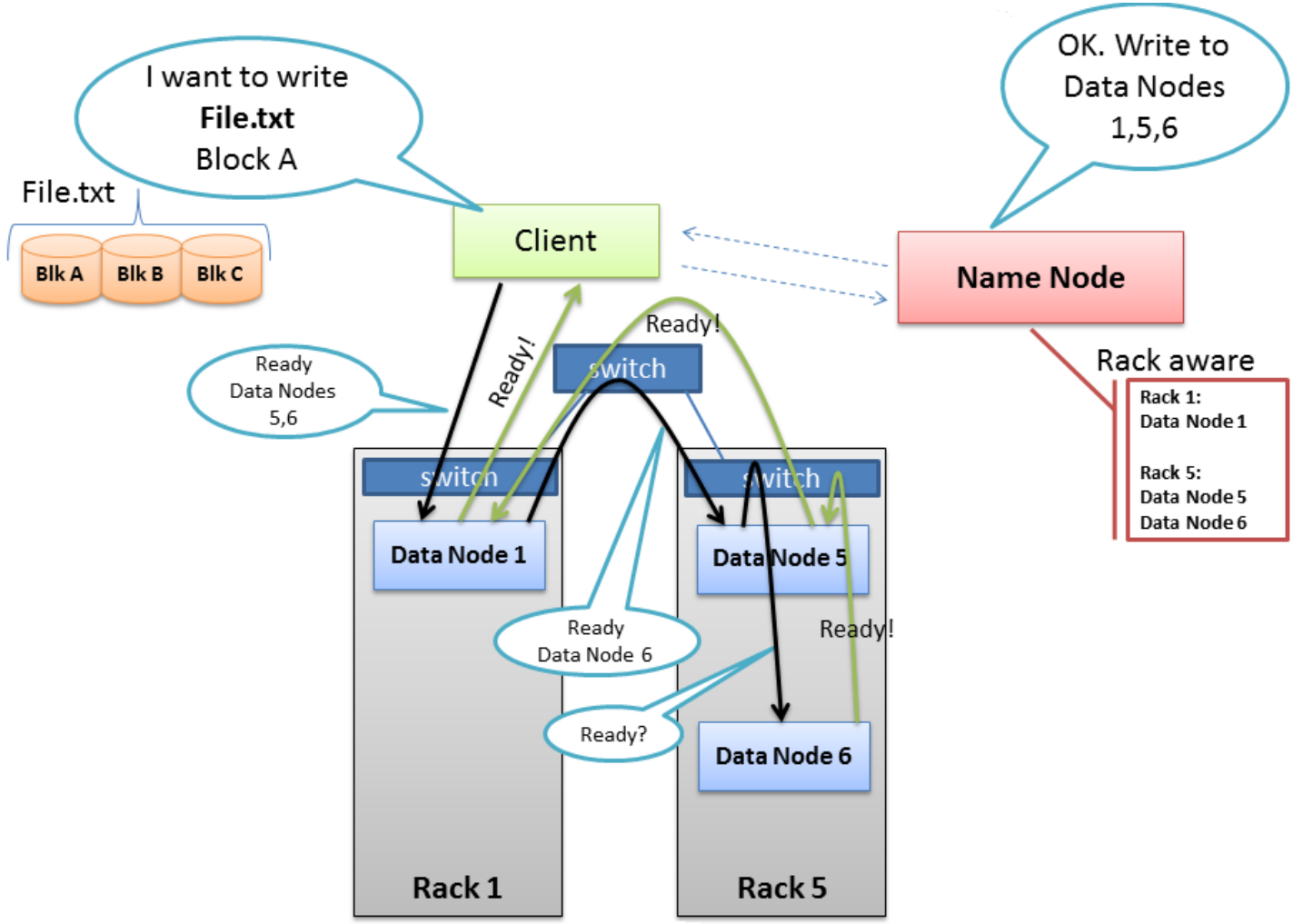


Source: http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network

- Client consults Name Node
- Client writes block directly to one Data Node
- Data Nodes replicates block
- Cycle repeats for next block

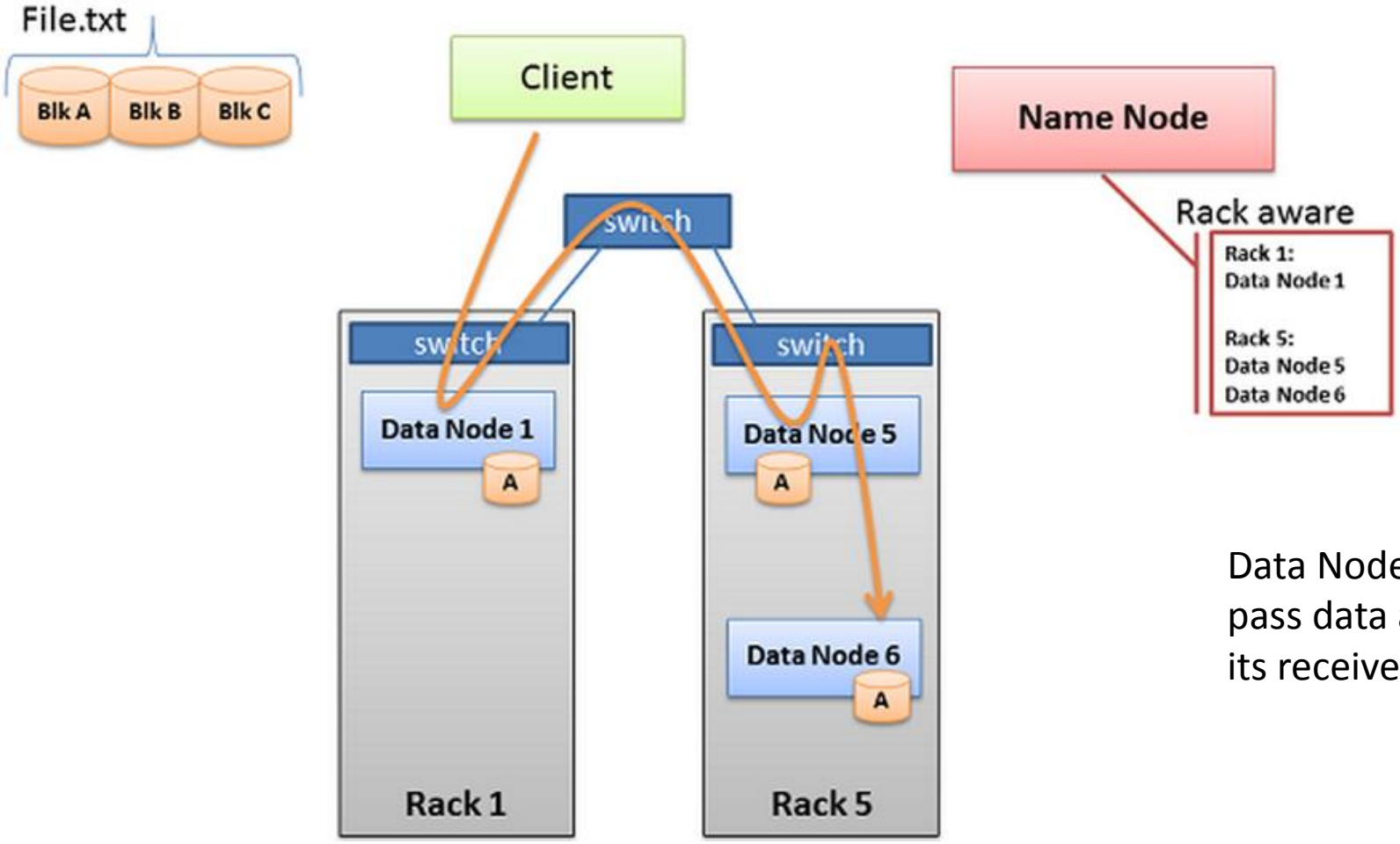
✓ **The Name Node is not in the data path. The Name Node only provides the map of where data is and where data should go in the cluster**

Preparing HDFS writes



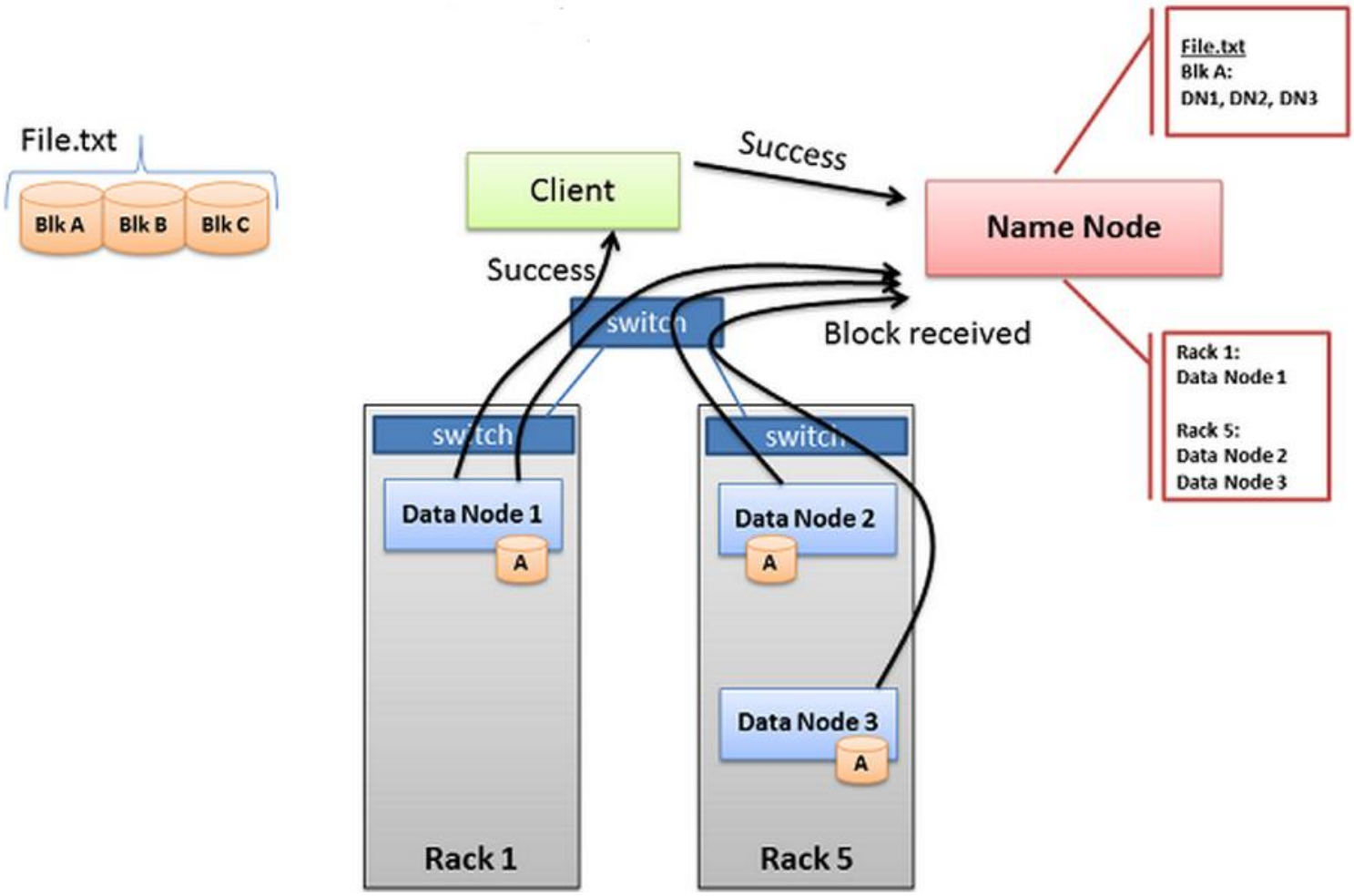
- Name Node picks two nodes in the same rack, one node in a different rack
- Data protection
- Locality for M/R

Pipelined write



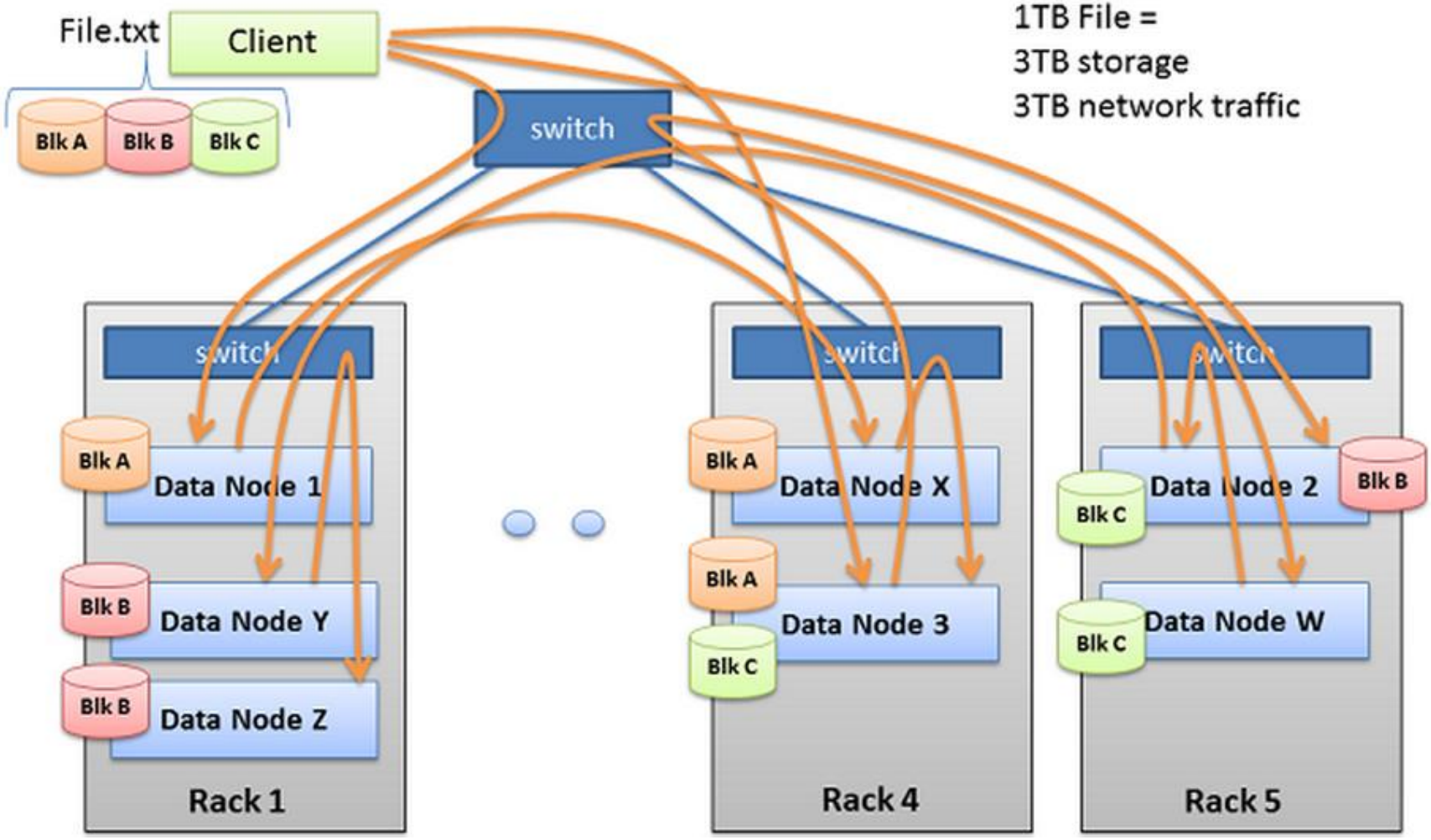
Data Nodes 1 & 5 pass data along as its received

Pipelined write (contd..)



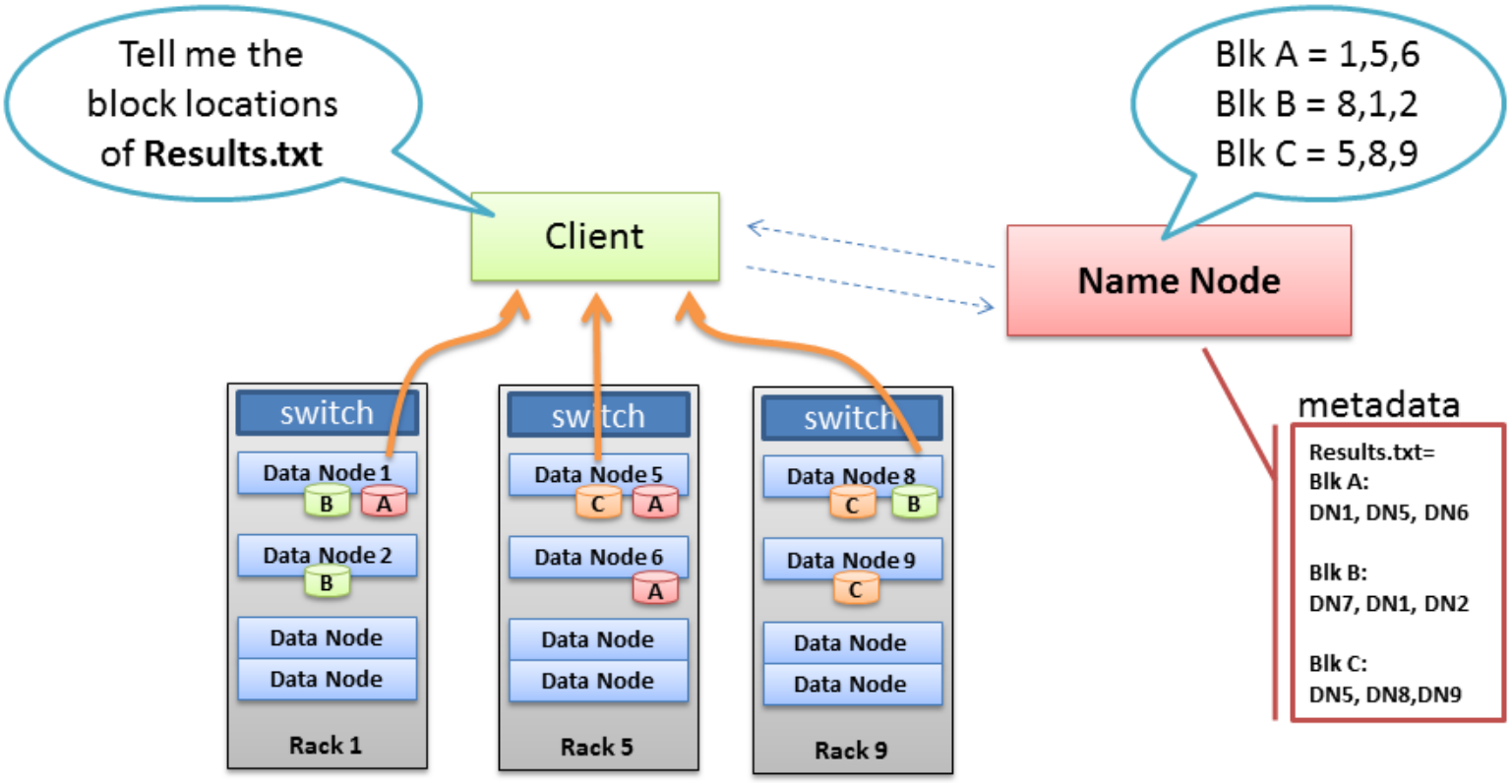
The Client is ready to start the pipeline process again for the next block of data

Multi-block replication pipeline



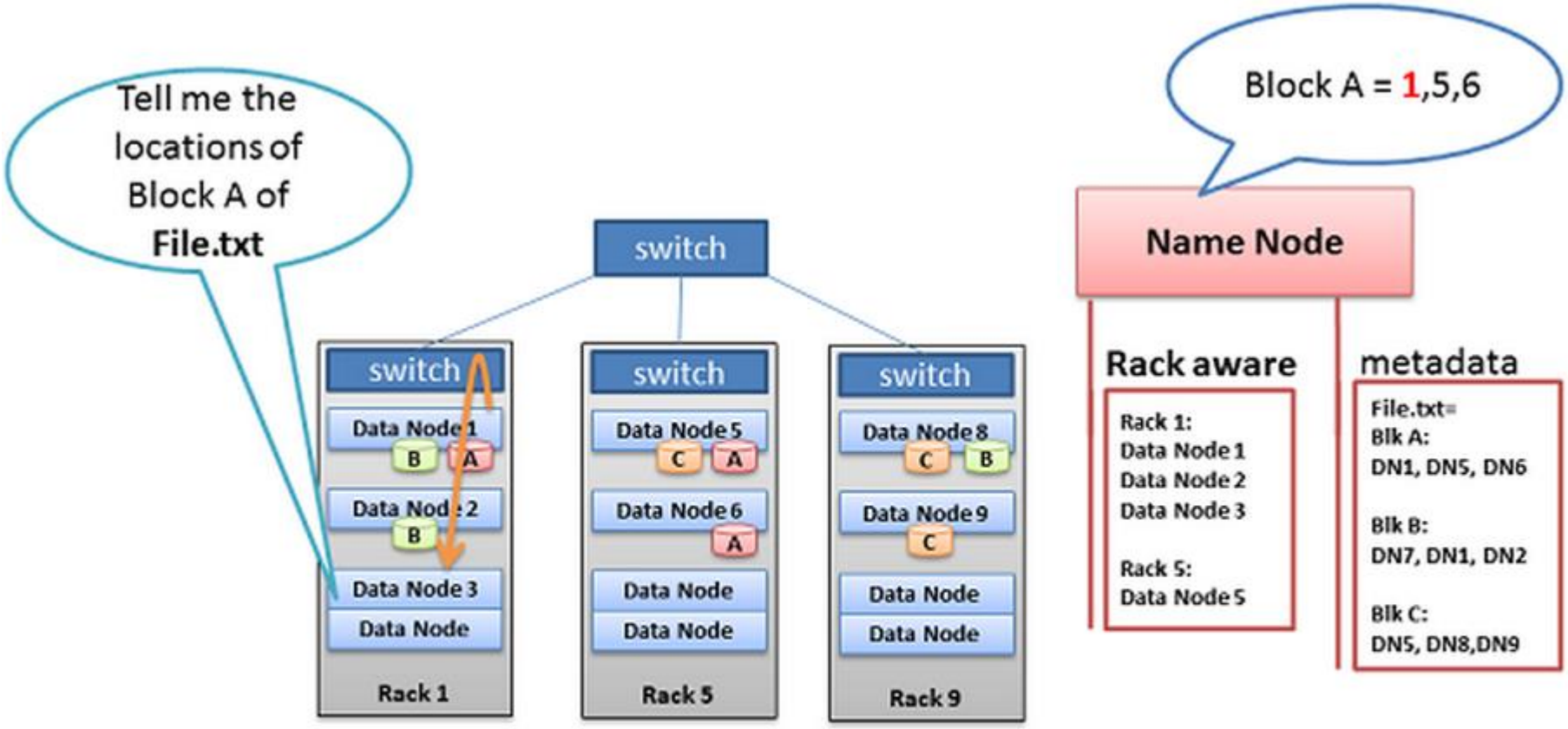
Note: The initial node in the pipeline will vary for each block

Client reading file from HDFS



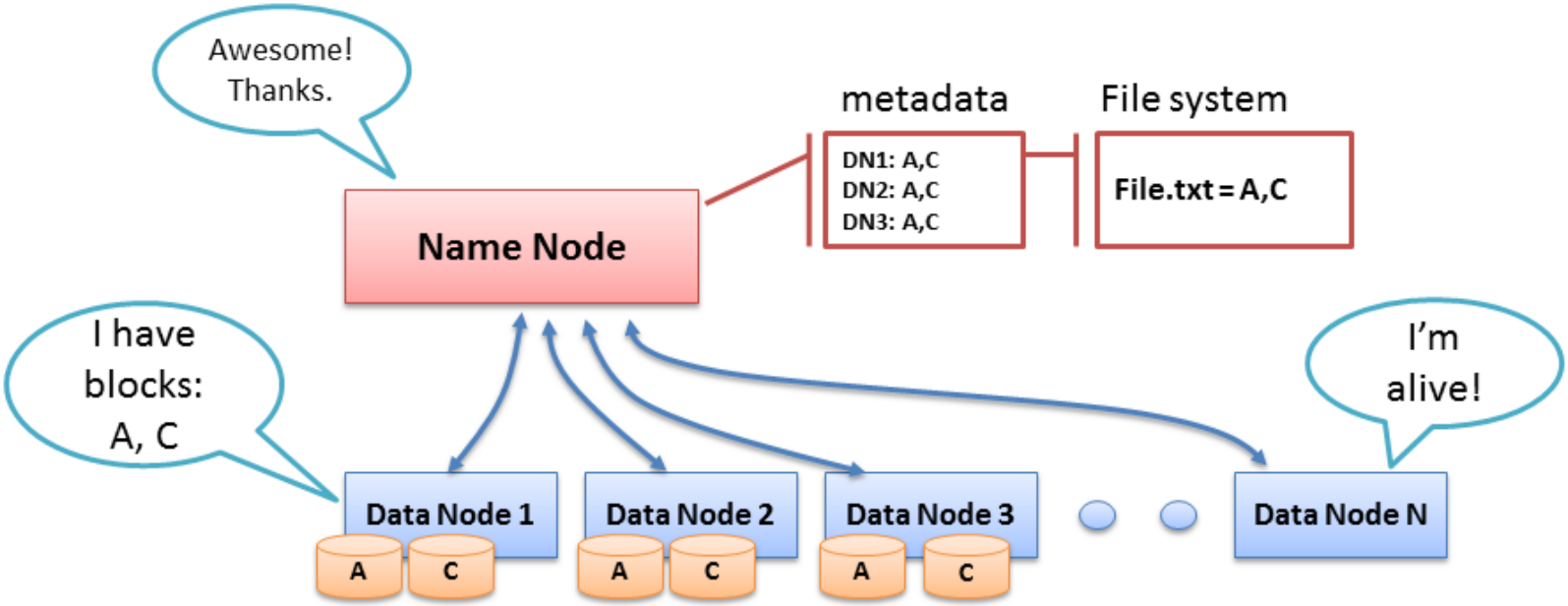
- Client receives Data Node list for each block
- Client picks first Data Node for each block
- Client reads blocks sequentially

Data Node reading file from HDFS



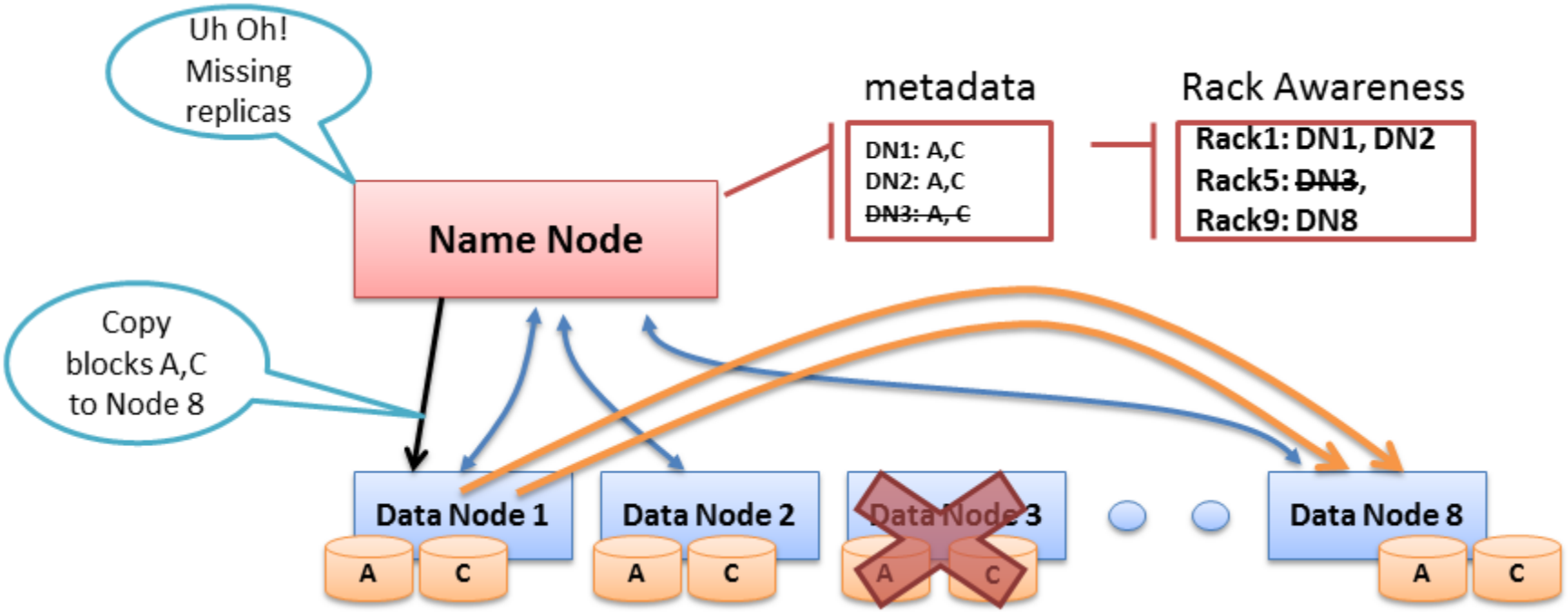
- Name Node provides rack local Nodes first
- Leverage in-rack bandwidth, single hop

Name node, Data node (heart beat)



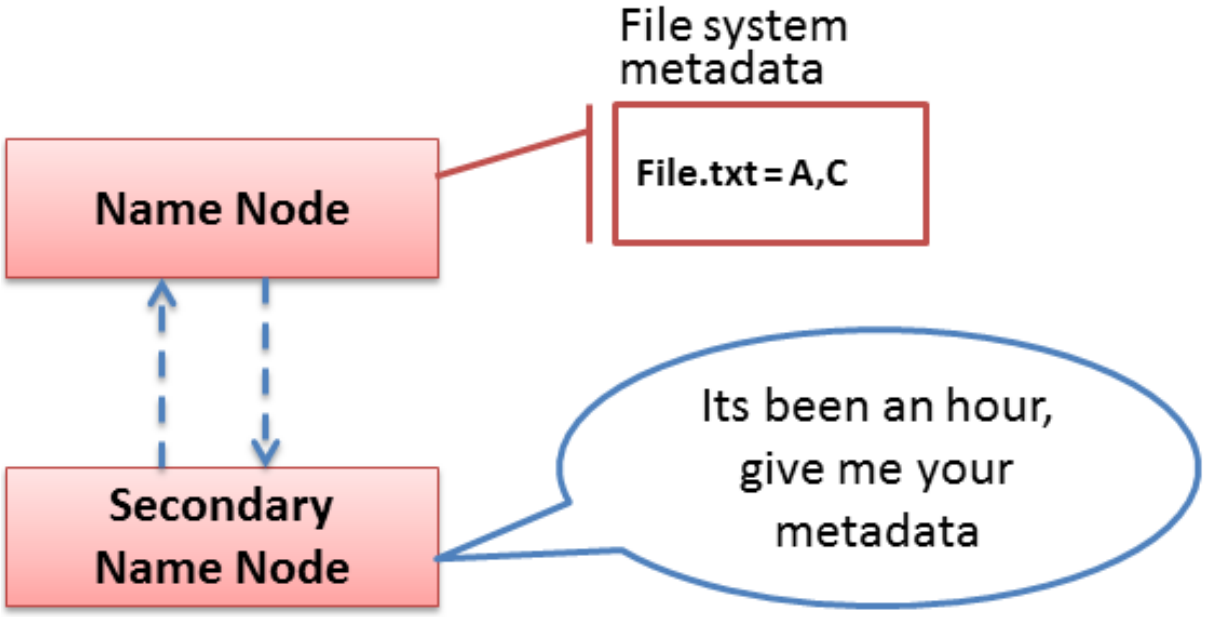
- Data Node sends Heartbeats
- Every 10th heartbeat is a Block report
- Name Node builds metadata from Block reports
- TCP – every 3 seconds
- If Name Node is down, HDFS is down

Data Recovery



- Missing Heartbeats signify lost Nodes
- Name Node consults metadata, finds affected data
- Name Node consults Rack Awareness script
- Name Node tells a Data Node to re-replicate

Single point of failure (name node)

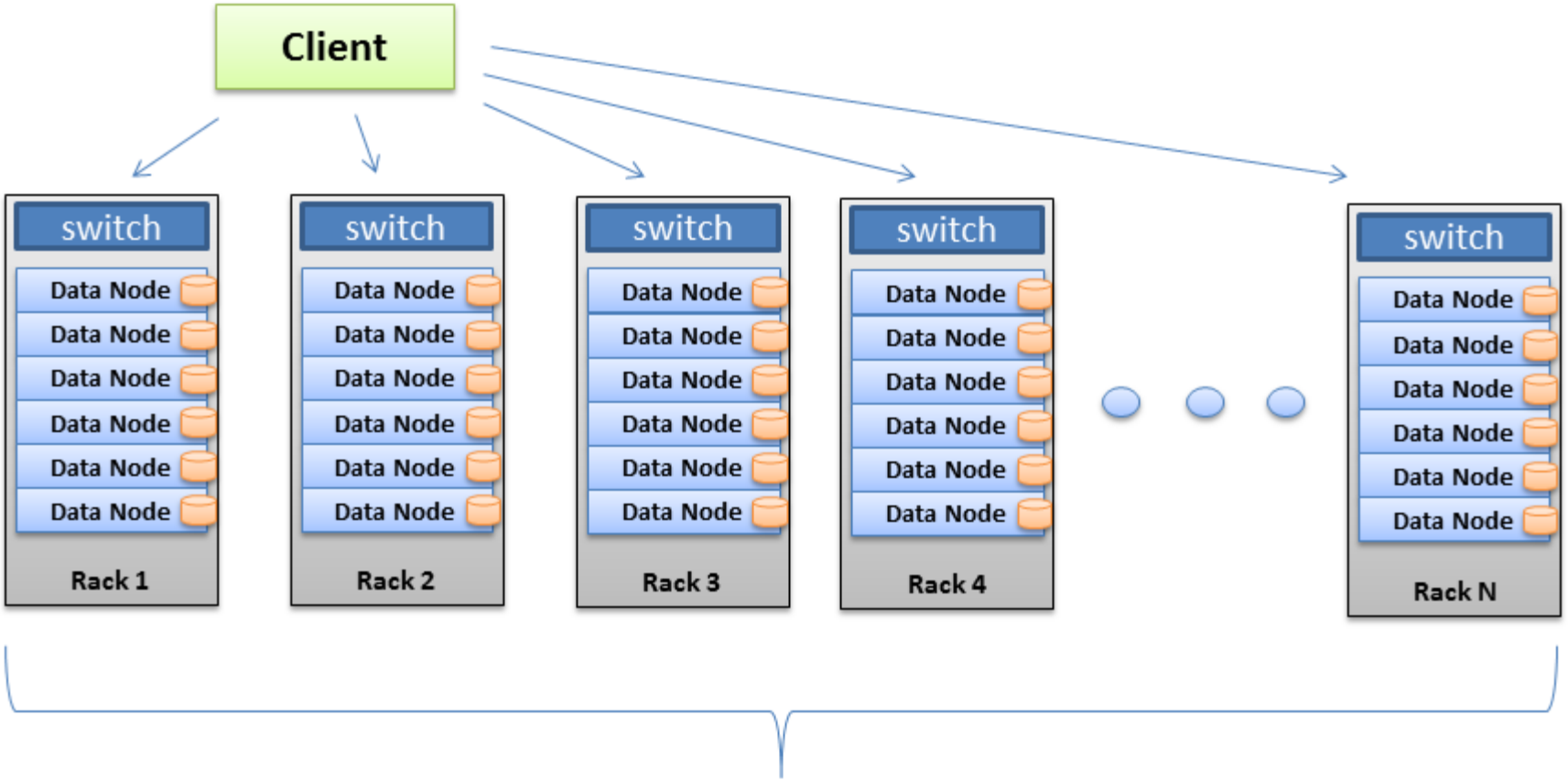


- Not a hot standby for the Name Node
- Connects to Name Node every hour
- Housekeeping, backup of Name Node metadata
- Saved metadata can rebuild a failed Name Node

Overall effect?

- As intended the file is spread in blocks across the cluster of machines, each machine having a relatively small part of the data
- The more blocks that make up a file, the more machines the data can potentially spread.
- The more CPU cores and disk drives that have a piece of your data mean more parallel processing power and faster results
- This is the motivation behind building large, wide clusters, to process more data, faster.

Scaling the cluster



Factors:

- Block size
- File Size

More blocks = Wider spread

Scaling the cluster (contd..)

Two approaches for scaling the cluster

- Wide
- Deep

Scaling the cluster (contd..)

Wide scaling:

- Cluster size increases by increasing the no. of nodes
- Network needs to scale appropriately

Deep:

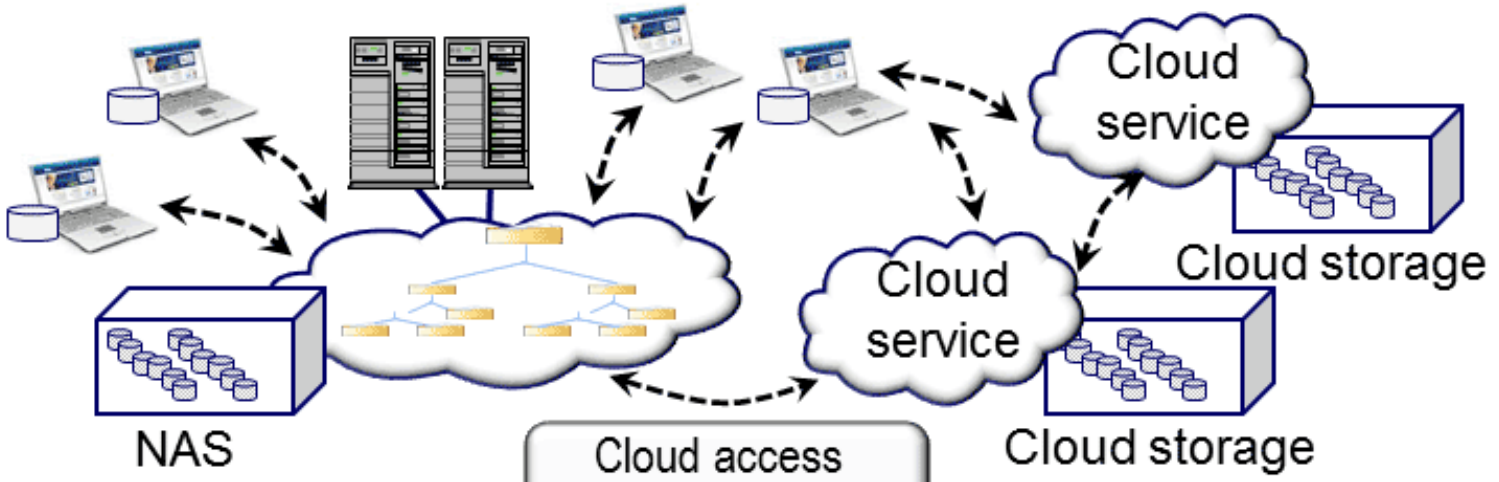
- Instead of increasing the number of machines you can look at increasing the density of each machine
- i.e., increasing each node capacity in terms of more CPUs, disk drives and RAM
- More network I/O requirements (fewer machines)

What does it have to do with cloud computing?

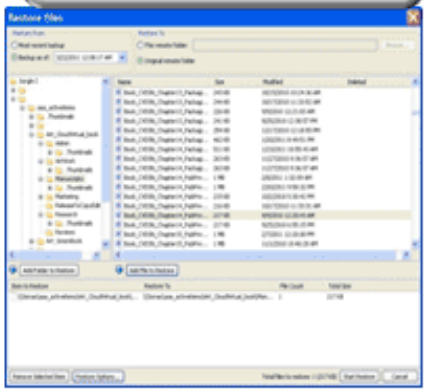
- Data is at the Heart of Cloud Computing Services
- Need File Systems that can fit the bill for large, scalable hardware and software
- GFS/HDFS and similar Distributed File Systems are now part and parcel of Cloud Computing Solutions.

Cloud???

- **Cloud storage** is a model of networked online storage where data is stored in virtualized pools of storage
- Companies operate large data centers, and people who require their data to be hosted, buy or lease storage capacity from them
- Cloud storage services may be accessed through a web service application programming interface (API), a cloud storage gateway or through a Web-based user interface
- It is difficult to pin down a canonical definition of cloud storage architecture, but object storage is reasonably analogous



- Backup/restore, archive, synchronize, replicate or copy data, files, folders, volumes to cloud storage
- Cloud data, file and storage sharing (public or private)
- List or display data or files stored in cloud



Source: Cloud and Virtual Data Storage Networking (CRC)

Common cloud API commands

PUT – Create container or update contents
POST – Update header information
LIST – Display information
GET – Retrieve container contents
HEAD – Access header meta data
DELETE – Remove container

Summary

- Introduction to file system
- Distributed File System (DFS)
- Case Studies
  - GFS
  - HDFS
  - Reading and writing files (HDFS)
  - Cloud storage