Hadoop Distributed File System

GVGD: VÕ THỊ HỒNG TUYẾT

Nội dung

1. HDFS
2. Architecture
3. NameNode, DataNodes, HDFS Client, CheckpointNode, BackupNode, Snapshots
4. File I/O Operations and Replica Management
5. File Read and Write, Block Placement, Replication management, Balancer,

HDFS

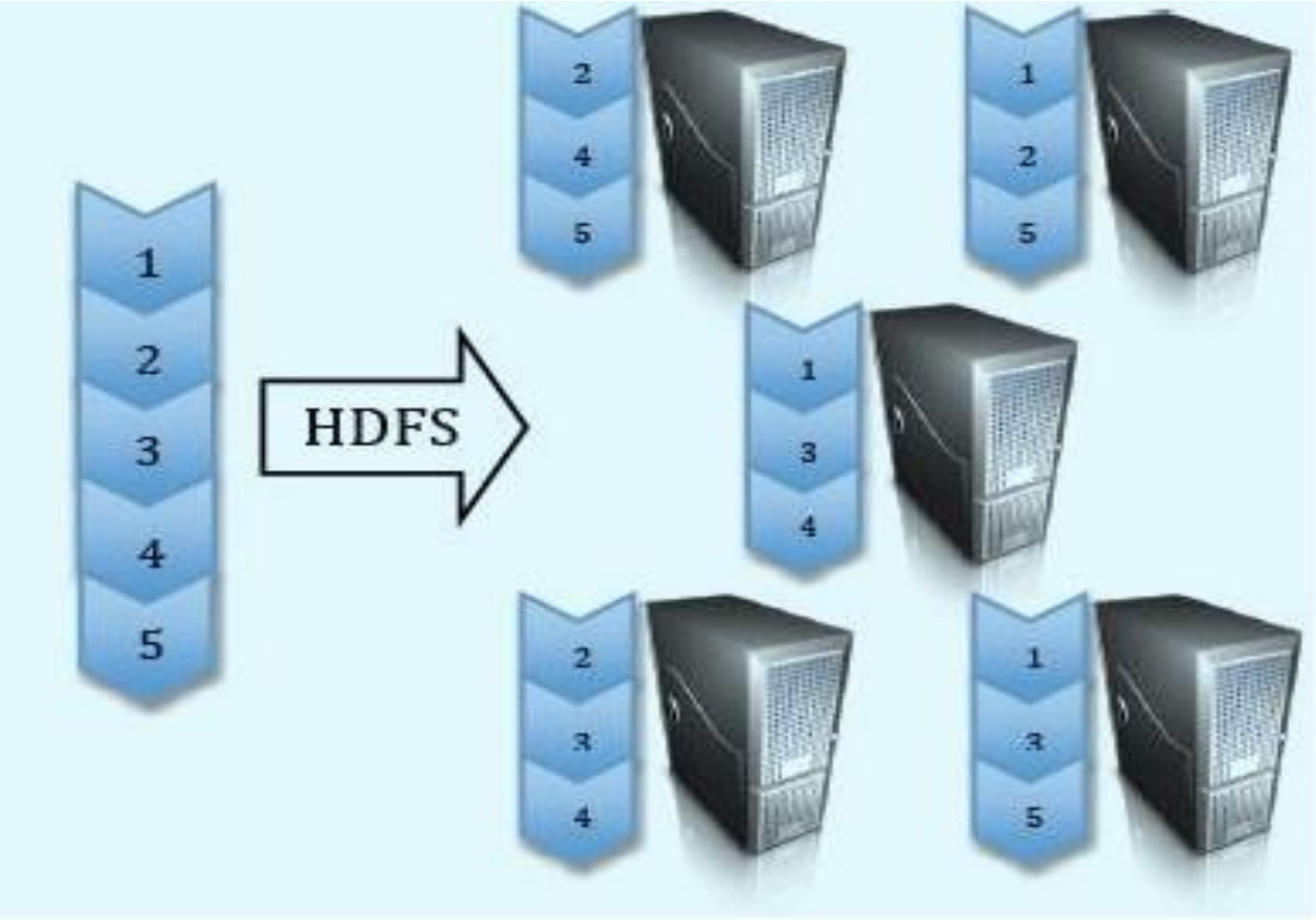
The Hadoop Distributed File System (HDFS) is the file system component of Hadoop.

It is designed to store very large data sets (1) *reliably*, and to stream those data sets (2) at *high bandwidth* to user applications. These are achieved by **replicating file content** on multiple machines(DataNodes).

Architecture

- HDFS is a block-structured file system: Files broken into blocks of 128MB (per-file configurable).
- A file can be made of several blocks, and they are stored across a cluster of one or more machines with data storage capacity.
- Each block of a file is replicated across a number of machines, To prevent loss of data.

Architecture



Architecture

NameNode and DataNodes

- HDFS stores file system metadata and application data separately.
- **Metadata** refers to file metadata(attributes such as permissions, modification, access times, namespace and disk space quotas)called "inodes"+list of blocks belong to the file.
- HDFS stores metadata on a dedicated server, called the NameNode.(Master) Application data are stored on other servers called DataNodes.(Slaves)
- All servers are fully connected and communicate with each other using TCP-based protocols.(RPC)

Architecture

Single Namenode:

- Maintain the namespace tree(a hierarchy of files and directories) operations like opening, closing, and renaming files and directories.
- Determine the mapping of file blocks to DataNodes (the physical location of file data).
- File metadata (i.e. “inode”) .
- Authorization and authentication.
- Collect block reports from Datanodes on block locations.
- Replicate missing blocks.
- HDFS keeps the entire namespace in RAM, allowing fast access to the metadata.

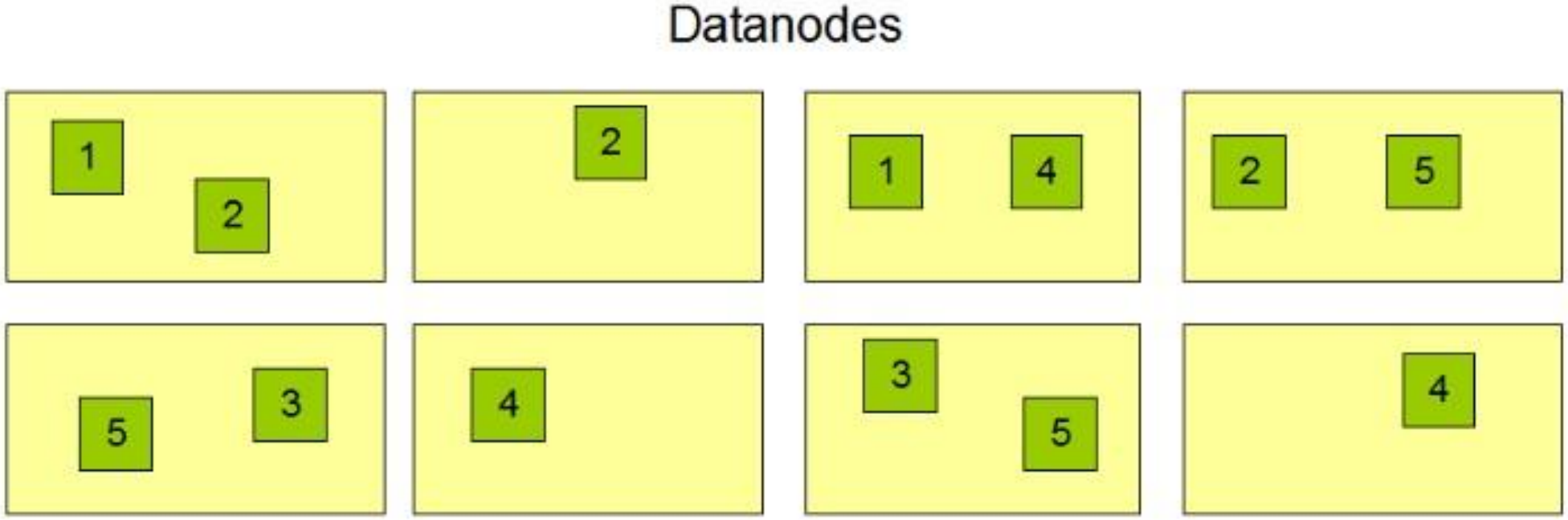
Architecture

DataNodes:

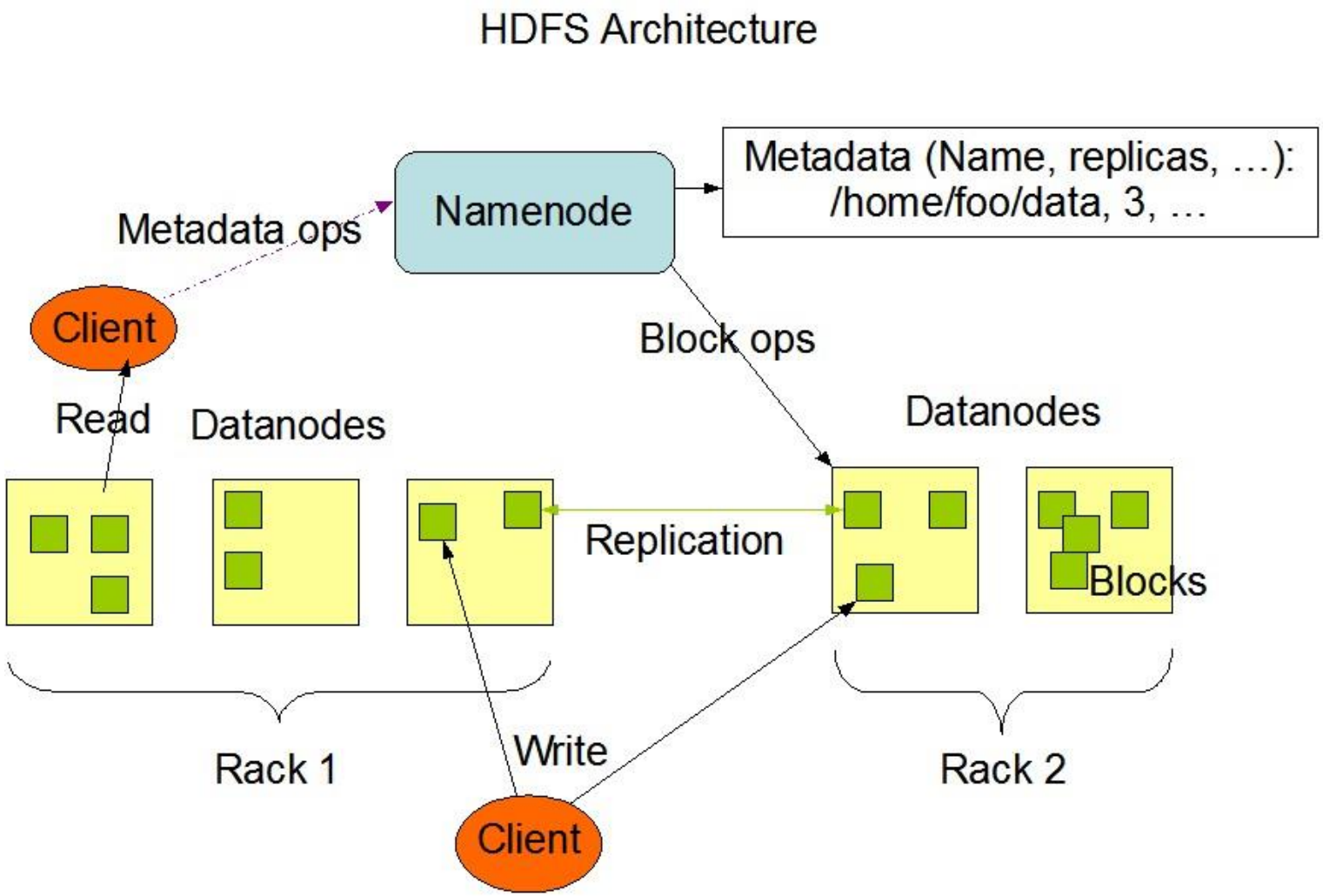
- The DataNodes are responsible for serving read and write requests from the file system's clients.
- The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.
- Data nodes periodically send block reports to Namenode.

Architecture

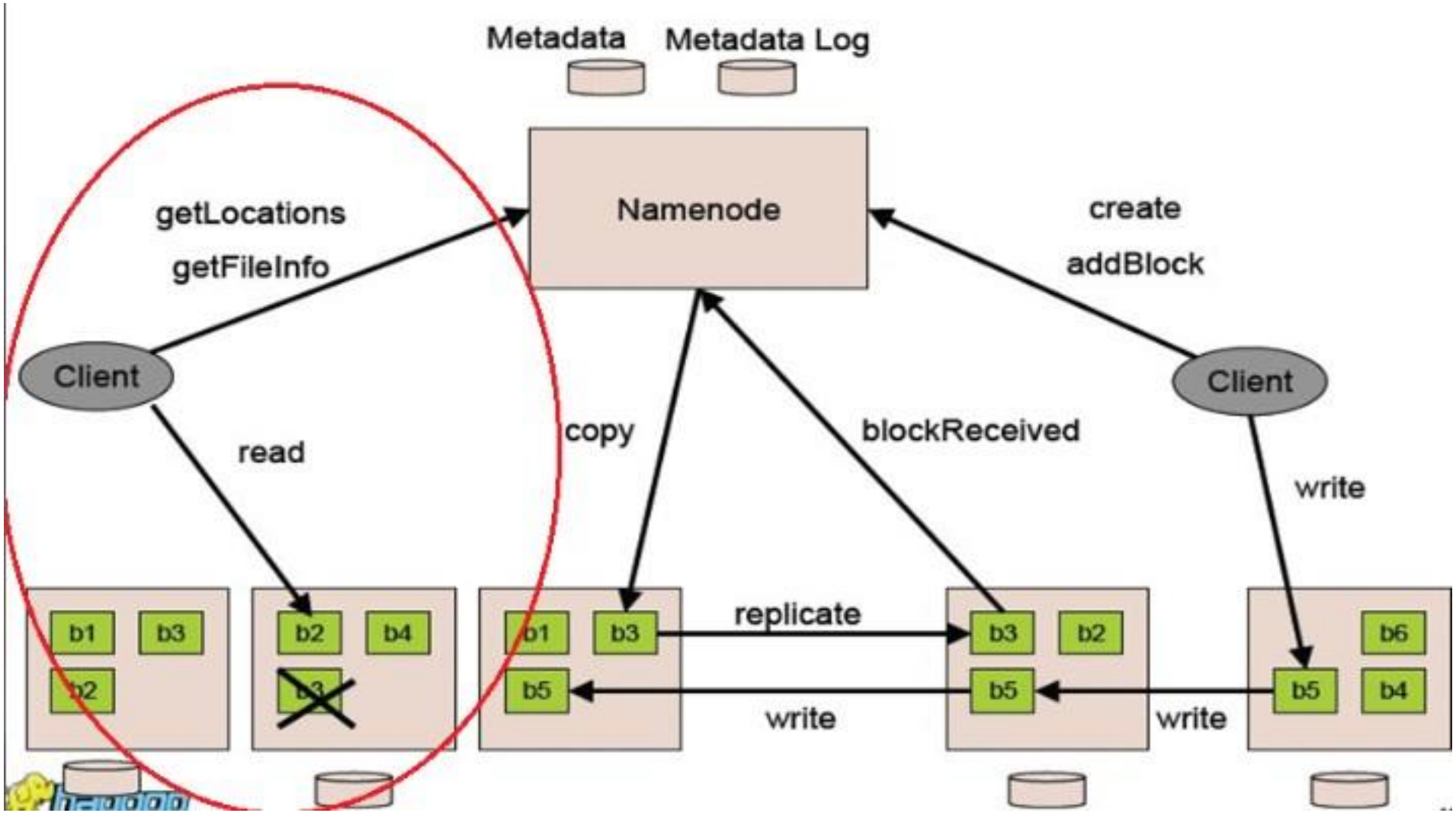
Namenode (Filename, numReplicas, block-ids, …)
/users/sameerp/data/part-0, r:2, {1,3}, …
/users/sameerp/data/part-1, r:3, {2,4,5}, …



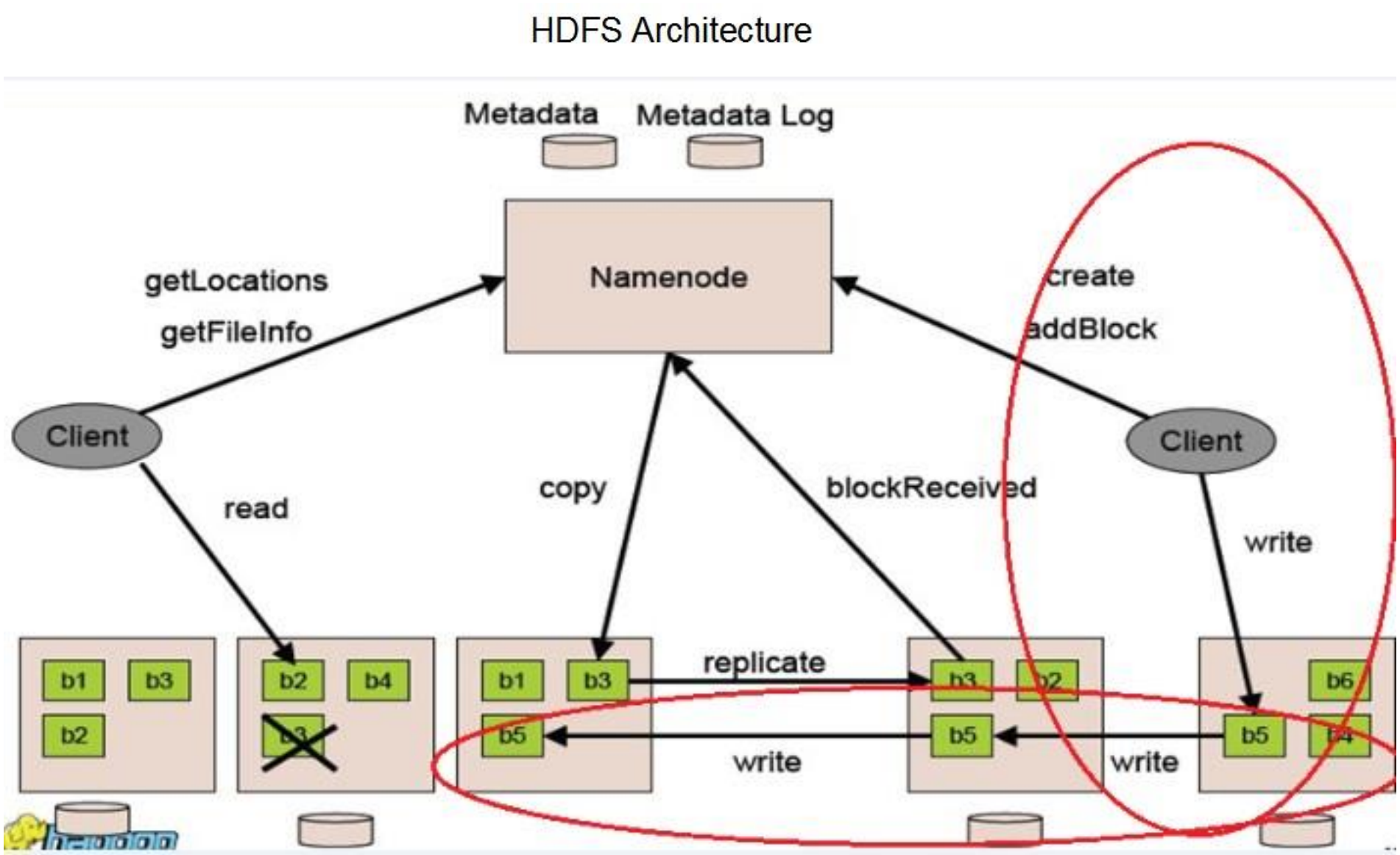
Architecture



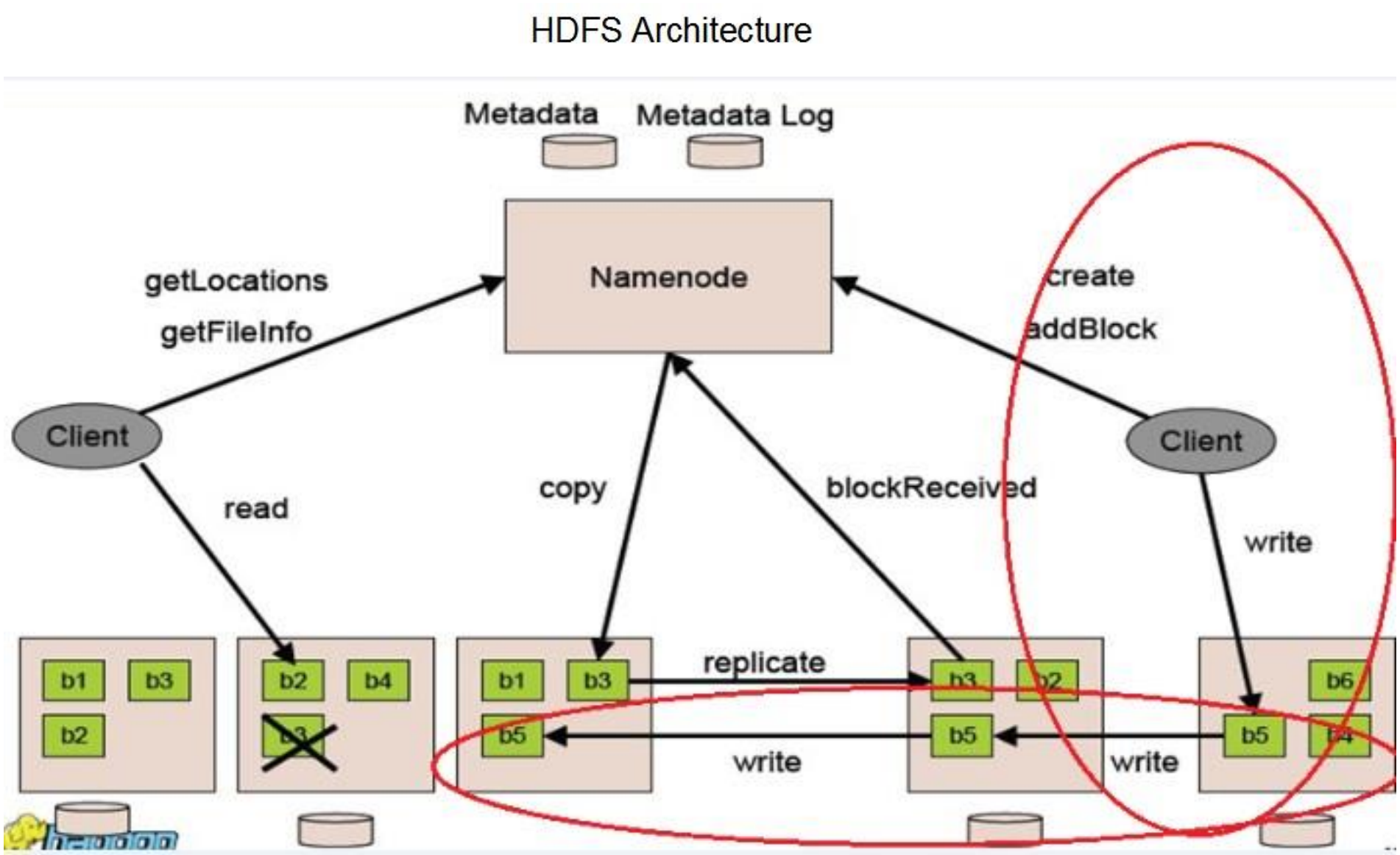
Architecture



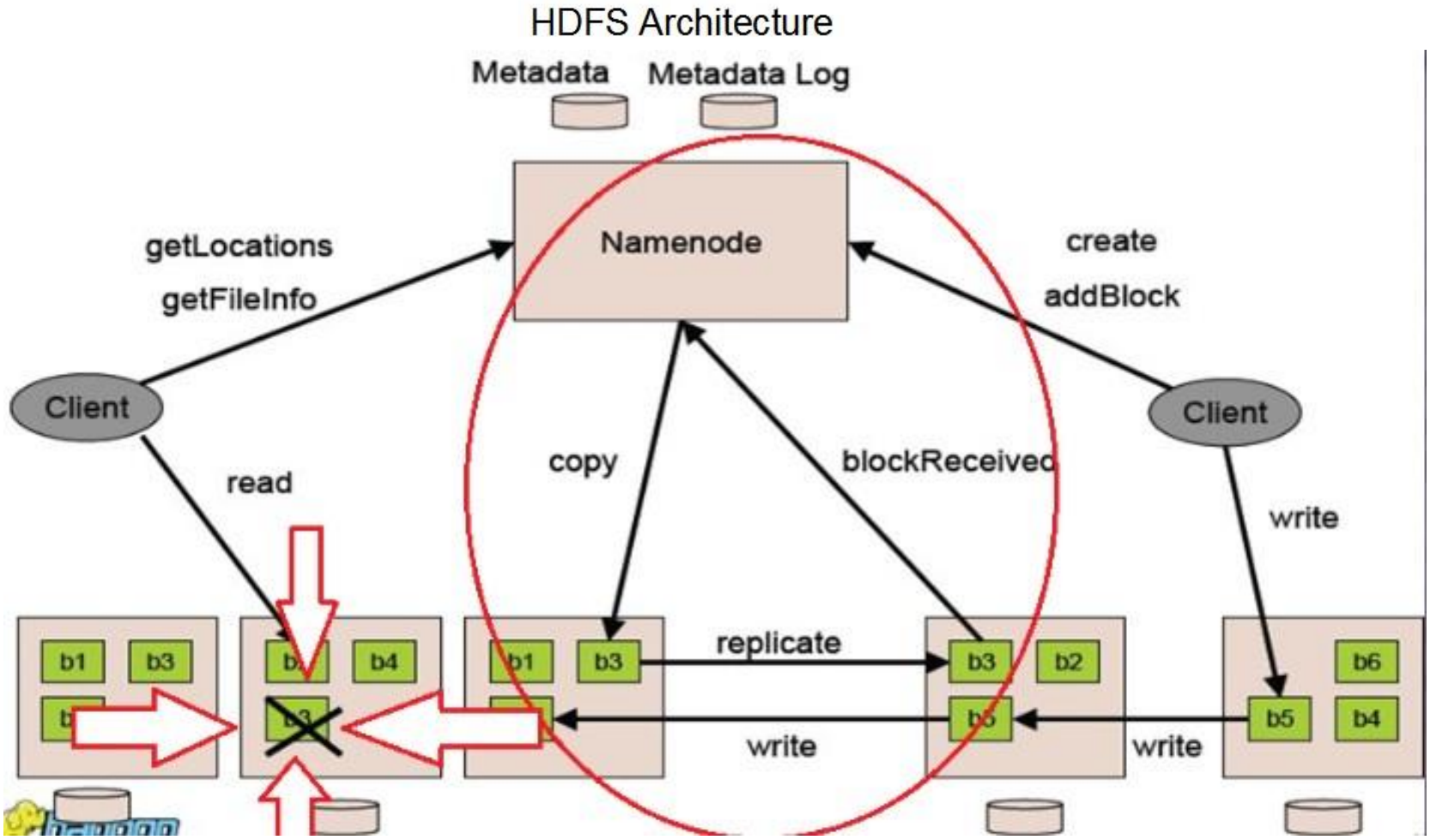
Architecture



Architecture



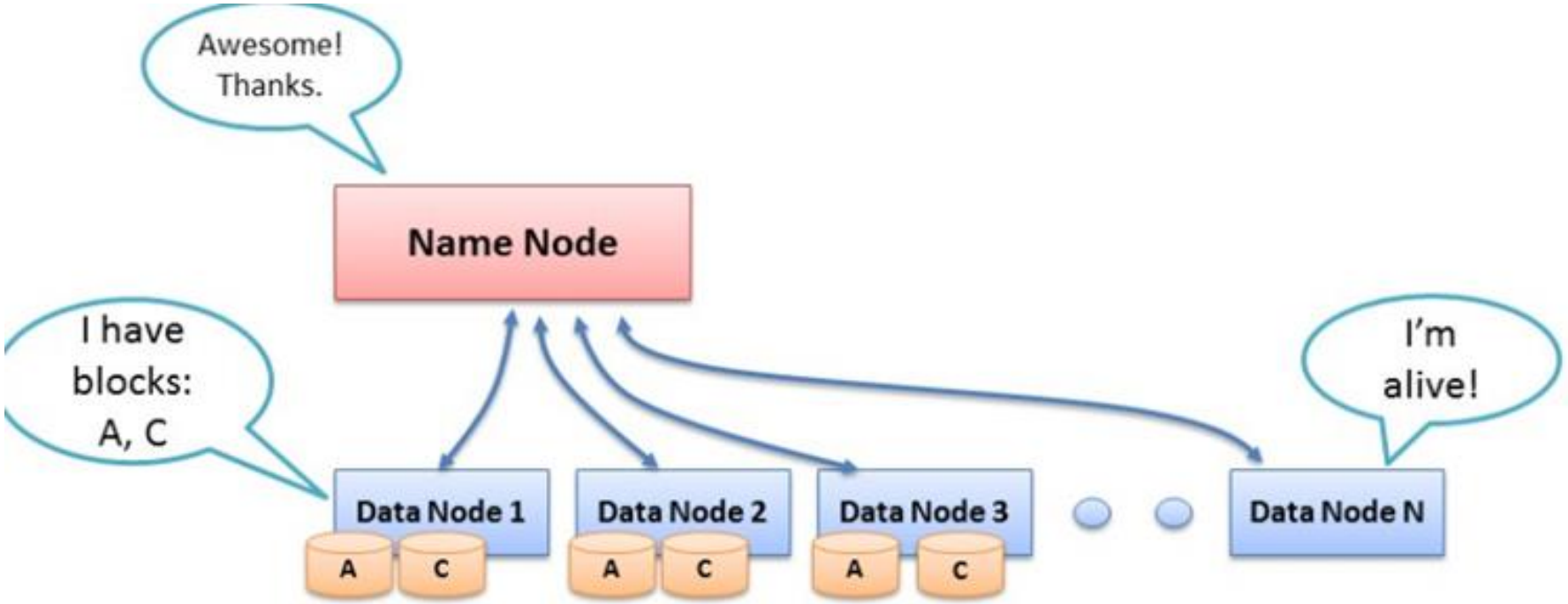
Architecture



Architecture

- NameNode and DataNode communication: *Heartbeats.*
- DataNodes send *heartbeats* to the NameNode to confirm that **the DataNode is operating** and **the block replicas it hosts are available.**

Architecture



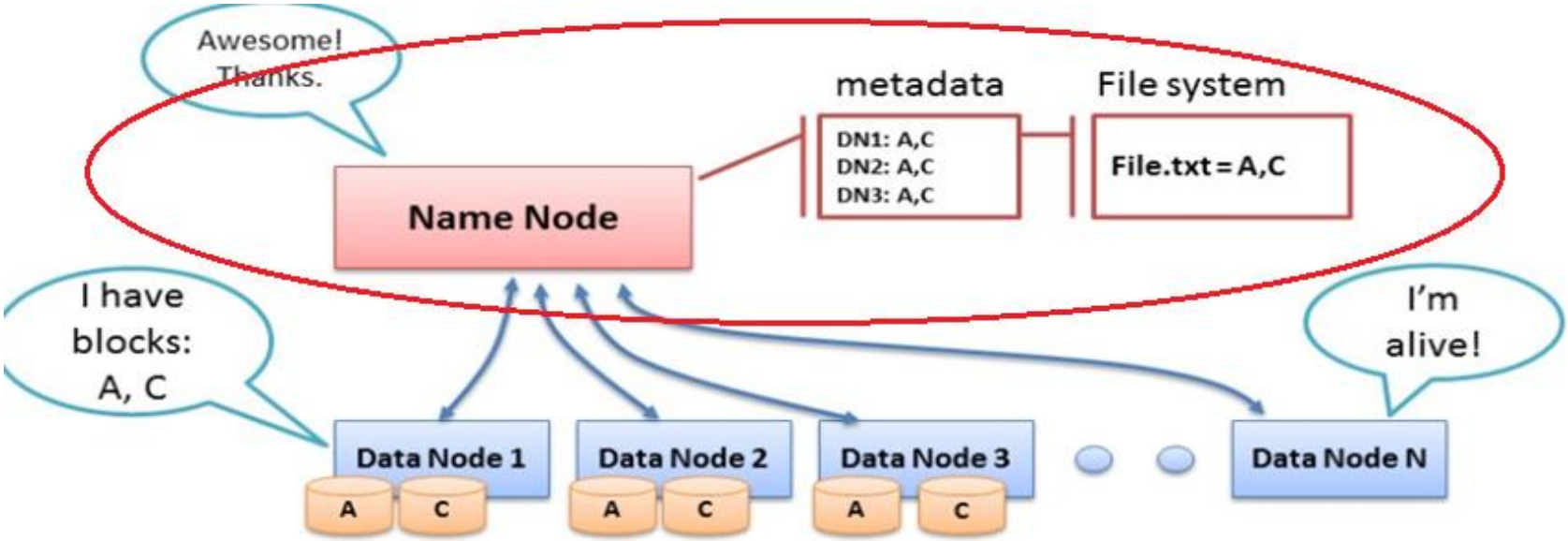
- Data Node sends Heartbeats
- Every 10th heartbeat is a Block report
- Name Node builds metadata from Block reports
- TCP – every 3 seconds

Architecture

Blockreports:

- A DataNode identifies block replicas in its possession to the NameNode by sending a *block report*. A block report contains the ***block id***, the ***generation stamp*** and the **length for each block replica** the server hosts.
- Blockreports provide the NameNode *with an up-to-date view* of where *block replicas* are located on the cluster and nameNode constructs and maintains latest metadata from blockreports.

Architecture



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

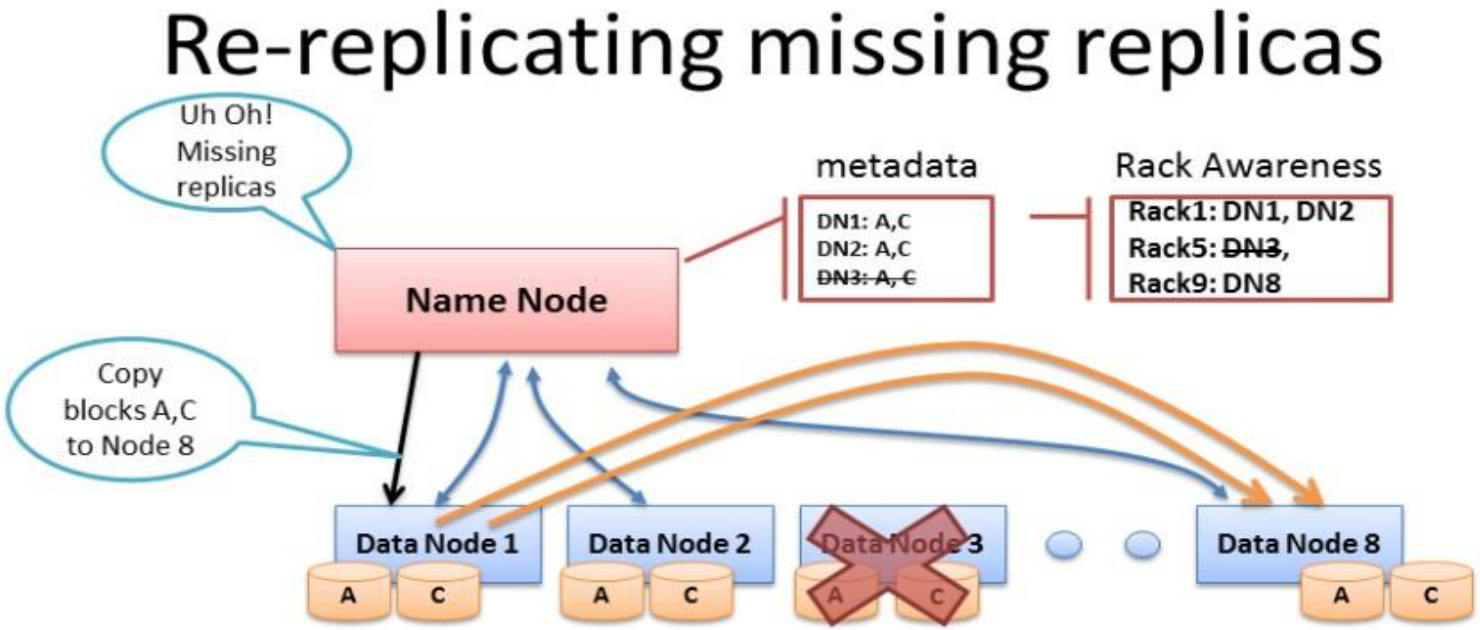
Architecture

Failure recovery

- The NameNode does not directly call DataNodes. It uses replies to heartbeats to send instructions to the DataNodes. The instructions include commands to:

- replicate blocks to other nodes:
  - DataNode died.
  - copy data to local.
- remove local block replicas;
- re-register or to shut down the node;

Architecture



- 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

Architecture

failure recovery

So when dataNode died, NameNode will notice and instruct other dataNode to replicate data to new dataNode. What if NameNode died?

Architecture

failure recovery

- Keep journal (the modification log of metadata).
- Checkpoint: The persistent record of the metadata stored in the local host's native files system.

For example:

During restart, the NameNode initializes the namespace image from the checkpoint, and then replays changes from the journal until the image is up-to-date with the last state of the file system.

Architecture

failure recovery

- **CheckpointNode** and **BackupNode**--two other roles of NameNode
- **CheckpointNode:**
- When journal becomes too long, checkpointNode combines the existing checkpoint and journal to create a new checkpoint and an empty journal.

Architecture

failure recovery

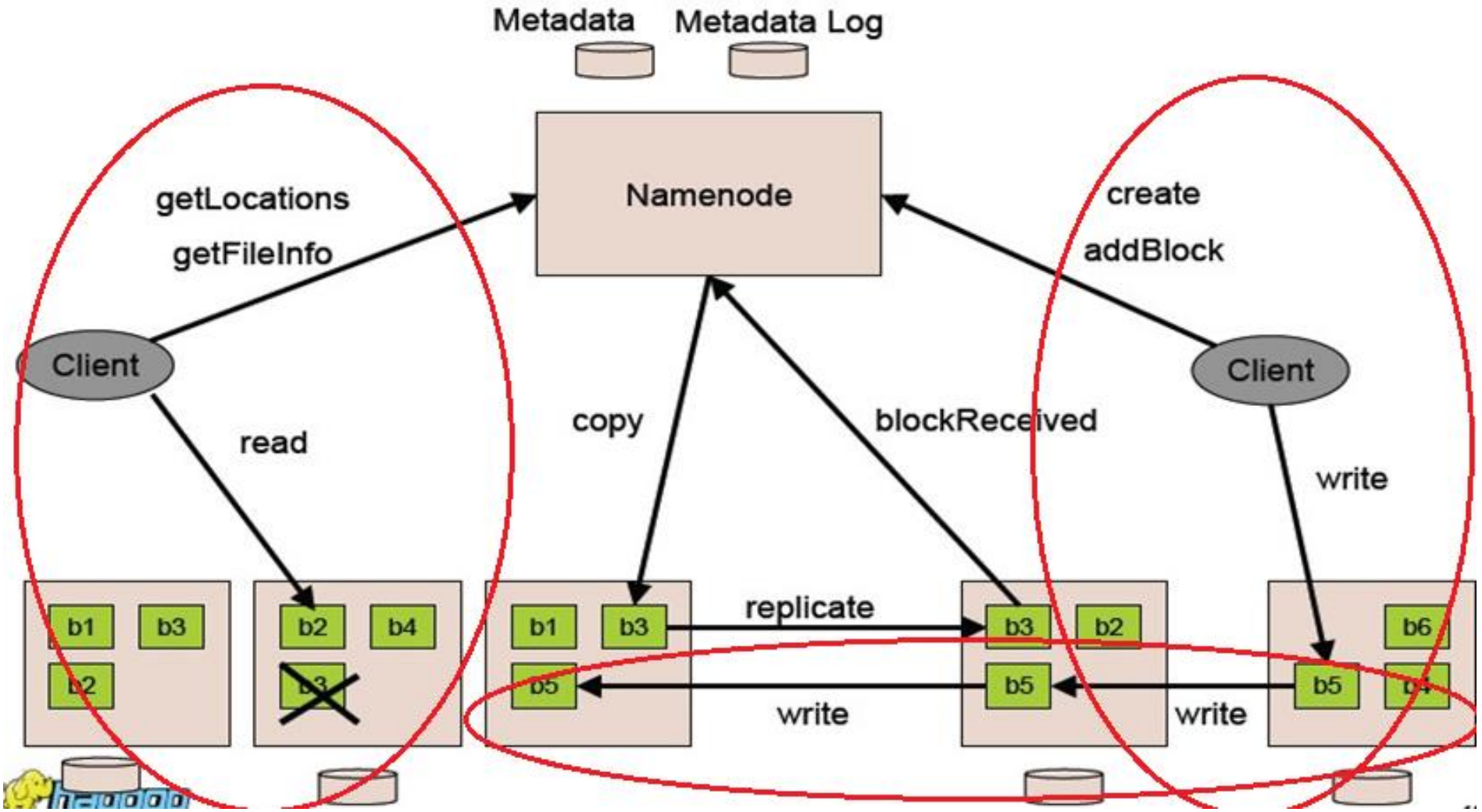
- **CheckpointNode** and **BackupNode**--two other roles of NameNode
- **BackupNode:** A read-only NameNode
- it maintains an *in-memory*, up-to-date image of the file system namespace that is always synchronized with the state of the NameNode.
- If the NameNode fails, the BackupNode's image in memory and the checkpoint on disk is a record of the latest namespace state.

Architecture

failure recovery

- Upgrades, File System Snapshots
- **The purpose of creating snapshots** in HDFS is to minimize potential damage to the data stored in the system during upgrades. During software upgrades the possibility of corrupting the system due to software bugs or human mistakes increases.
- The snapshot mechanism lets administrators **persistently save the current state of the file system(both data and metadata)**, so that if the upgrade results in data loss or corruption, it is possible to **rollback the upgrade** and return HDFS to the namespace and storage state as they were at the time of the snapshot.

File I/O Operations and Replica Management



File I/O Operations and Replica Management

- Hadoop has the concept of "Rack Awareness".

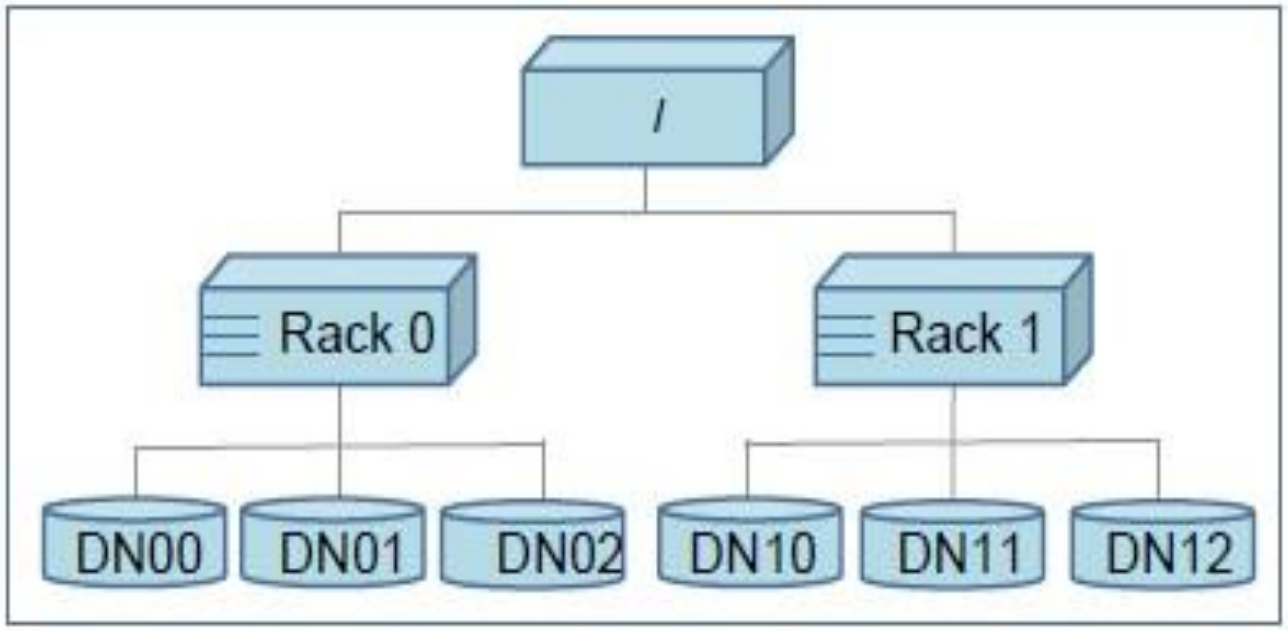


Figure 3. Cluster topology example

File I/O Operations and Replica Management

- Hadoop has the concept of “Rack Awareness”.

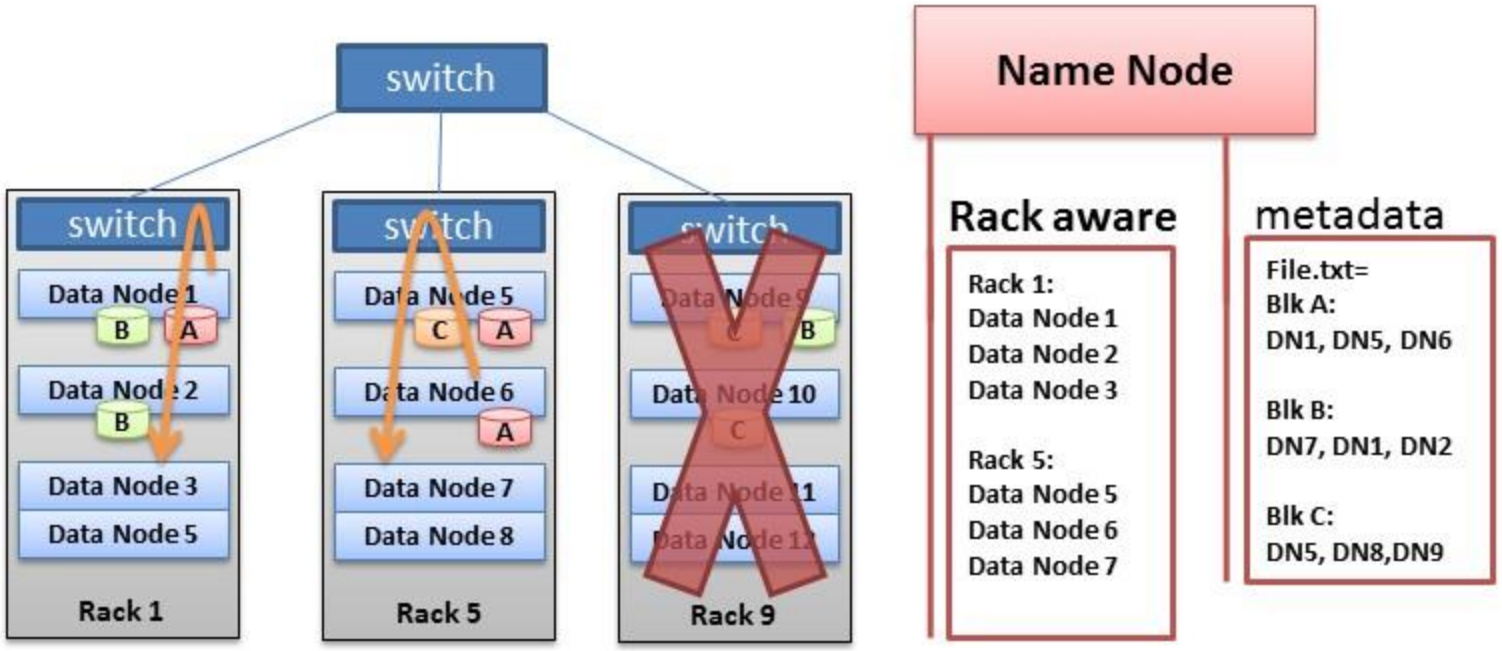
- The default HDFS replica placement policy can be summarized as follows:

  1. No Datanode contains more than one replica of any block.

  2. No rack contains more than two replicas of the same block, provided there are sufficient racks on the cluster.

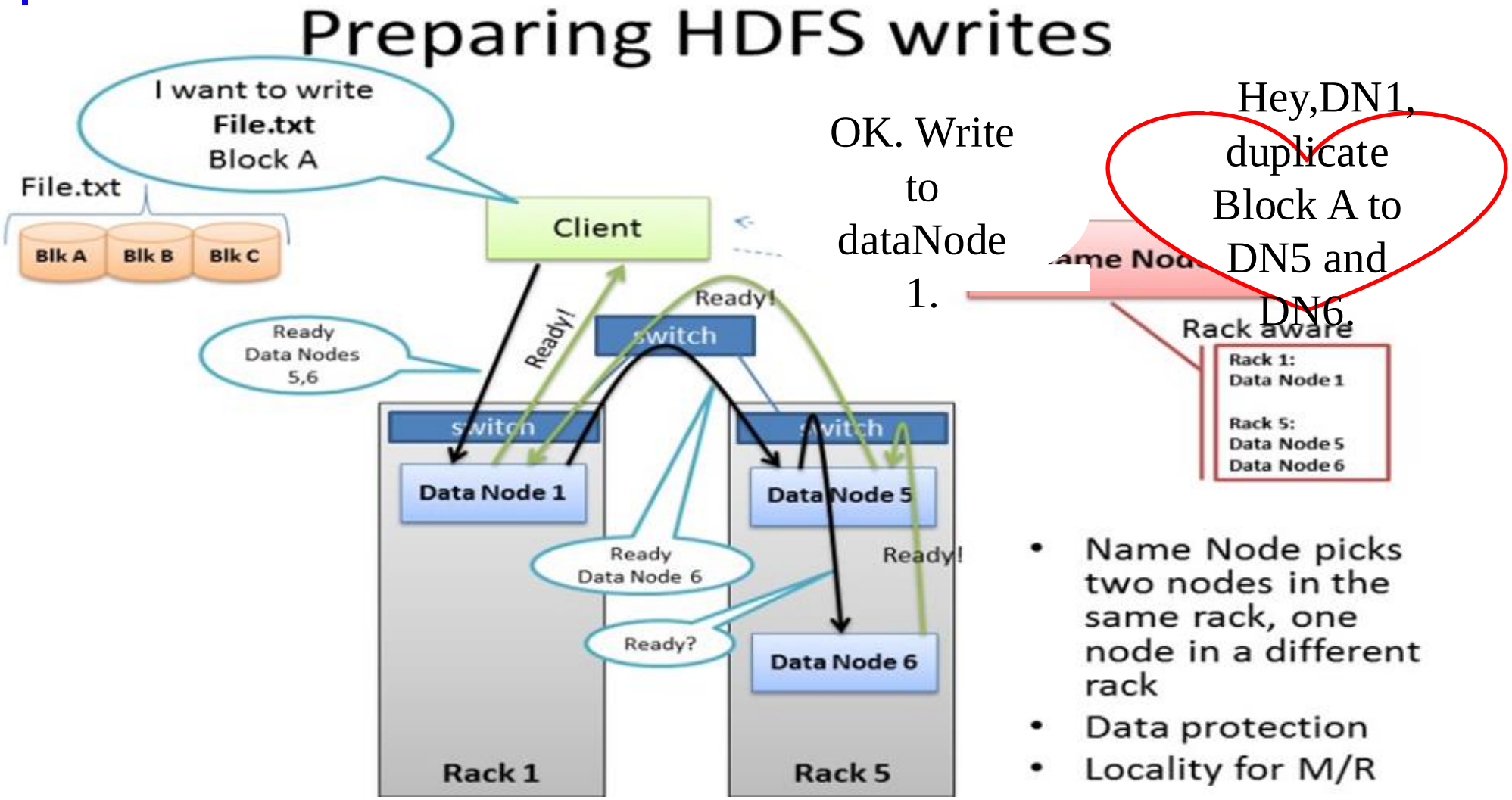
File I/O Operations and Replica Management

Hadoop Rack Awareness – Why?



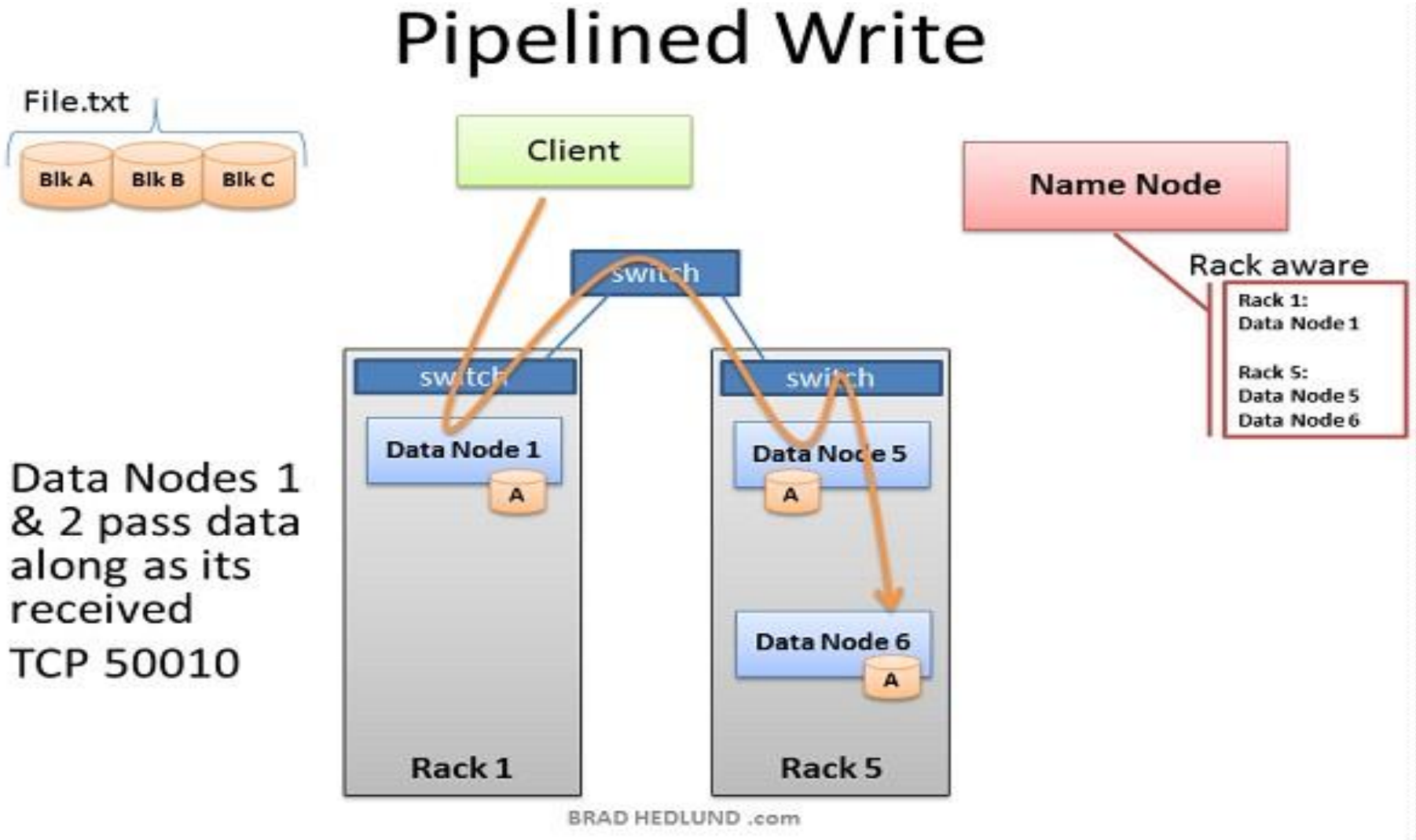
- Never loose all data if entire rack fails

File I/O Operations and Replica Management



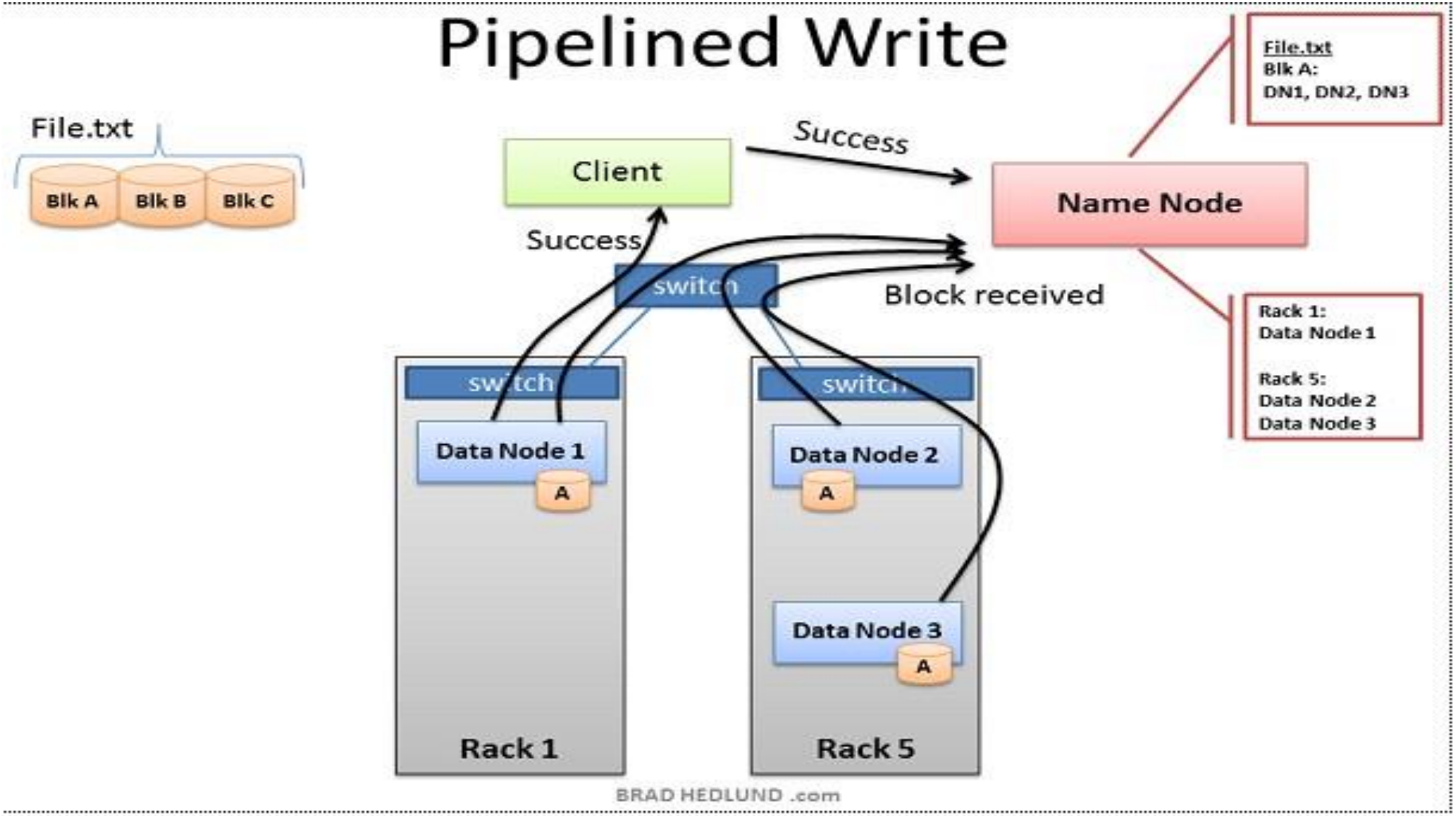
- Name Node picks two nodes in the same rack, one node in a different rack
- Data protection
- Locality for M/R

File I/O Operations and Replica Management



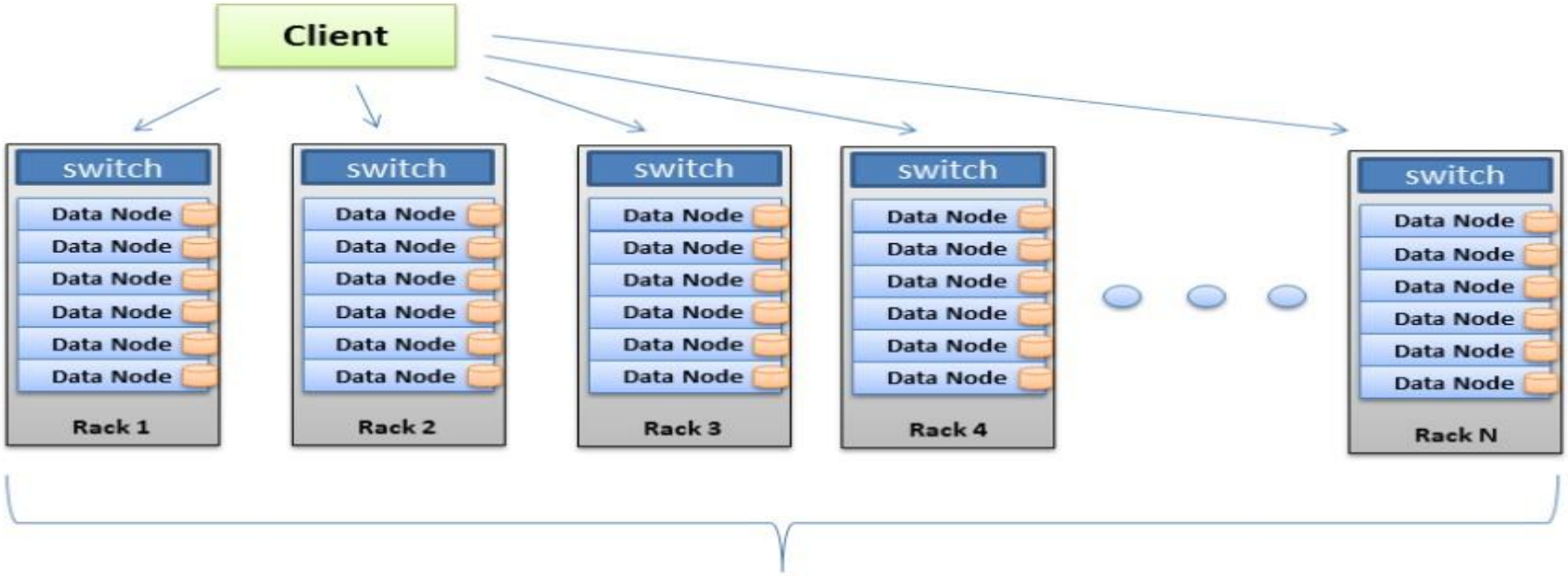
- Data Nodes 1 & 2 pass data along as its received
- TCP 50010

File I/O Operations and Replica Management



File I/O Operations and Replica Management

Client writes Span the HDFS Cluster

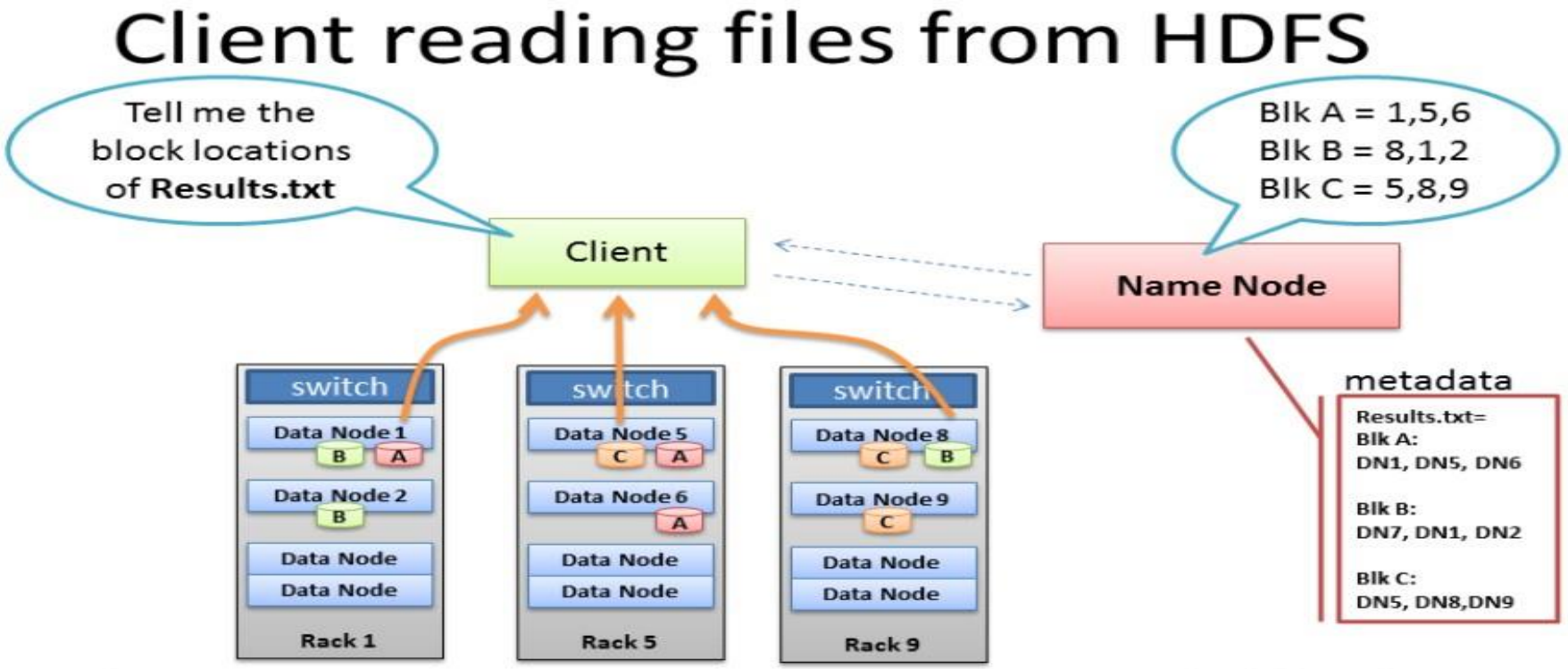


Factors:

- Block size
- File Size

More blocks = Wider spread

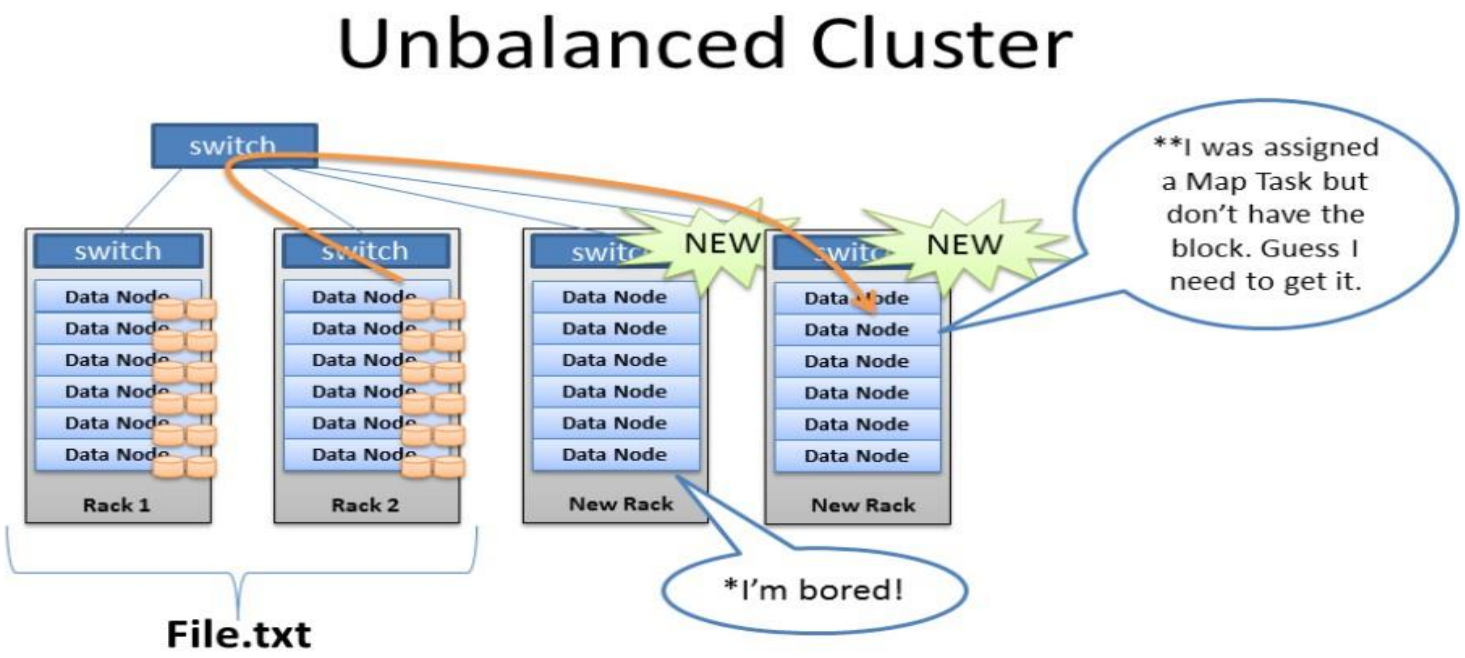
File I/O Operations and Replica Management



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

File I/O Operations and Replica Management

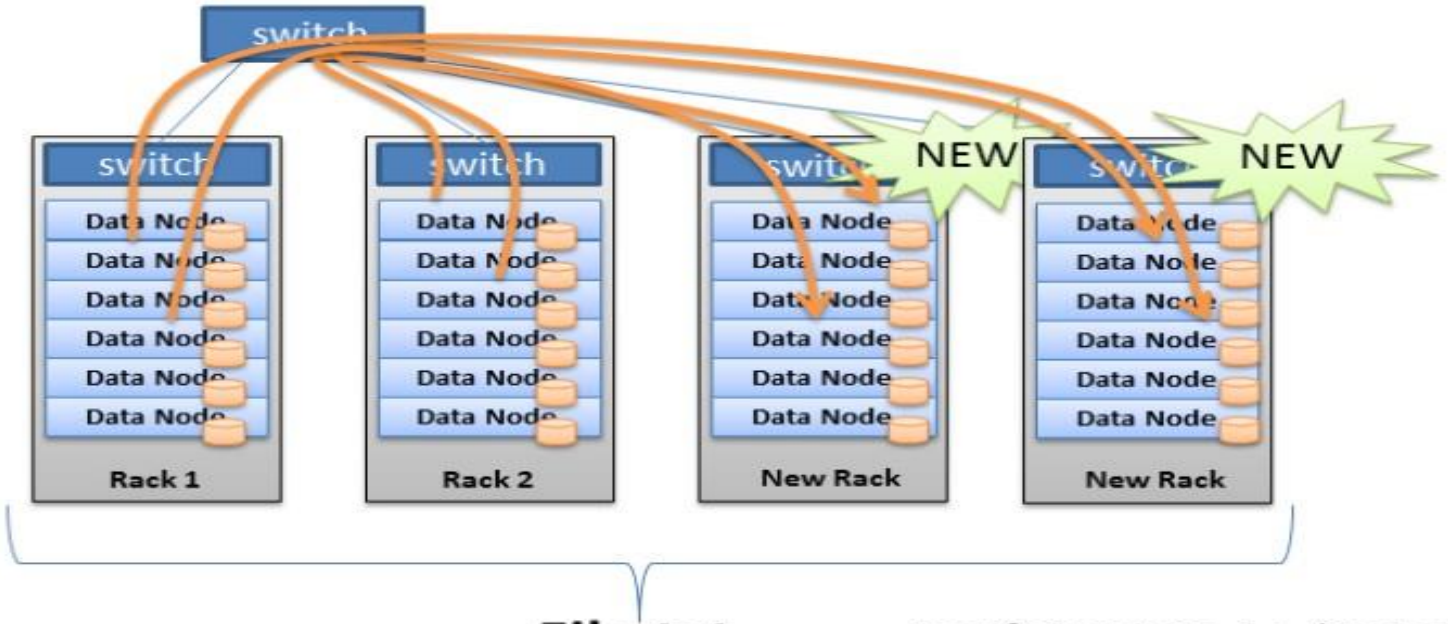
- Balancer



- Hadoop prefers local processing <u>if possible</u>
- New servers underutilized for Map Reduce, HDFS*
- More network bandwidth, slower job times**

File I/O Operations and Replica Management

Cluster Balancing



- Balancer utility (if used) runs in the background
- Does not interfere with Map Reduce or HDFS
- Default rate limit 1 MB/s