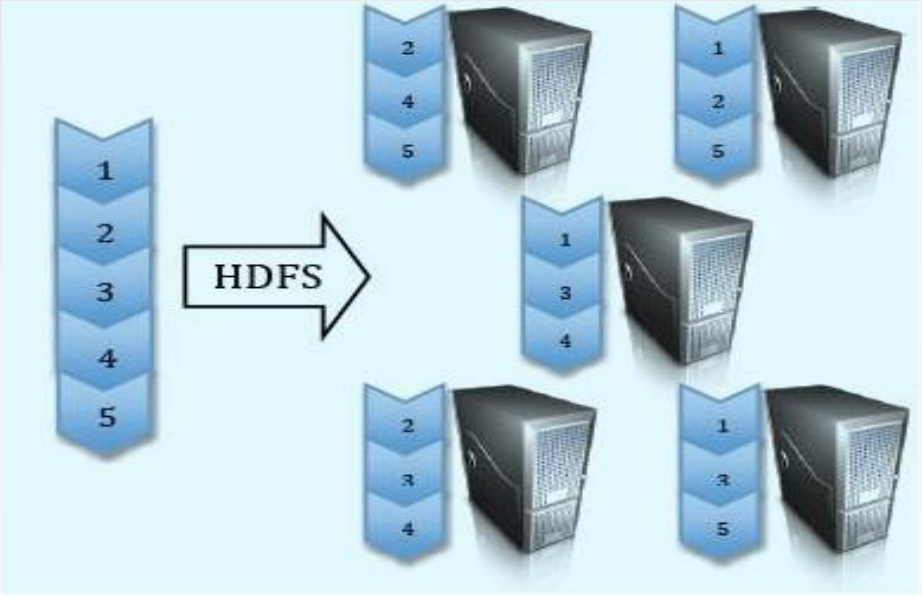
# HDFS Architecture

- HDFS is a block-structured file system: Files broken into blocks of 64MB or 128MB
- 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.

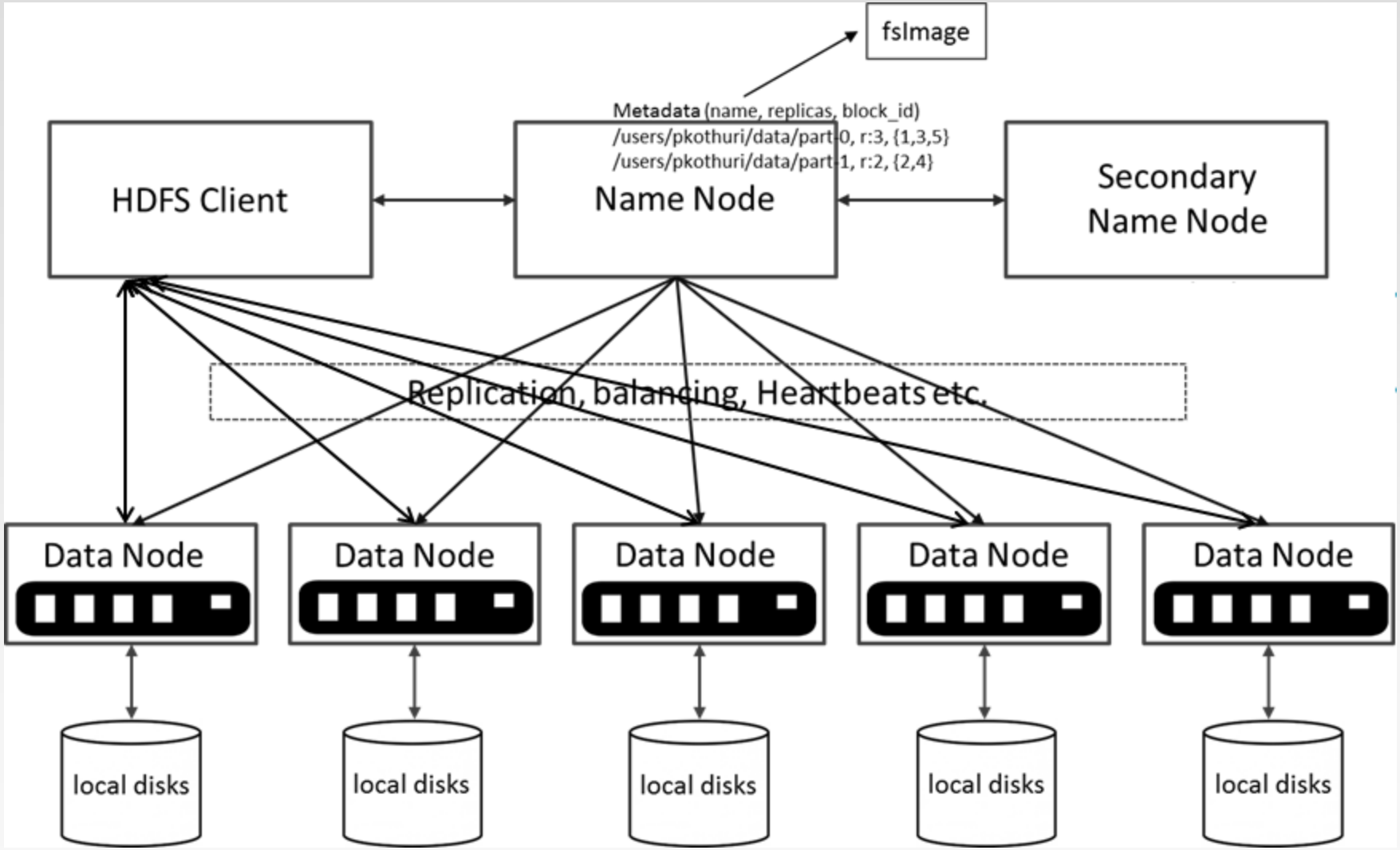


# HDFS Architecture

- HDFS has a master/slave architecture.

- There are two types (and a half) of machines in a HDFS cluster
  - NameNode: the heart of an HDFS filesystem,  it maintains and manages the file system metadata. E.g., what blocks make up a file, and on which datanodes those blocks are stored.
    - Only one in an HDFS cluster
  - DataNode: where HDFS stores the actual data. Serves read, write requests, performs block creation, deletion, and replication upon instruction from Namenode
    - A number of DataNodes usually one per node in a cluster.
    - A file is split into one or more blocks and set of blocks are stored in DataNodes.
  - Secondary NameNode: **NOT** a backup of NameNode!!
    - Checkpoint node. Periodic merge of Transaction log
    - Help NameNode start up faster next time

# HDFS Architecture



# Functions of a NameNode

- Managing the file system namespace:
  - Maintain the namespace tree operations like opening, closing, and renaming files and directories.
  - Determine the mapping of file blocks to DataNodes (the physical location of file data).
  - Store file metadata.
- Coordinating file operations:
  - Directs clients to DataNodes for reads and writes
  - No data is moved through the NameNode
- Maintaining overall health:
  - Collect block reports and heartbeats from DataNodes
  - Block re-replication and rebalancing
  - Garbage collection

# NameNode Metadata

- HDFS keeps the entire namespace in RAM, allowing fast access to the metadata.
  - 4GB of local RAM is sufficient
- Types of metadata
  - List of files
  - List of Blocks for each file
  - List of DataNodes for each block
  - File attributes, e.g. creation time, replication factor
- A Transaction Log (EditLog)
  - Records file creations, file deletions etc

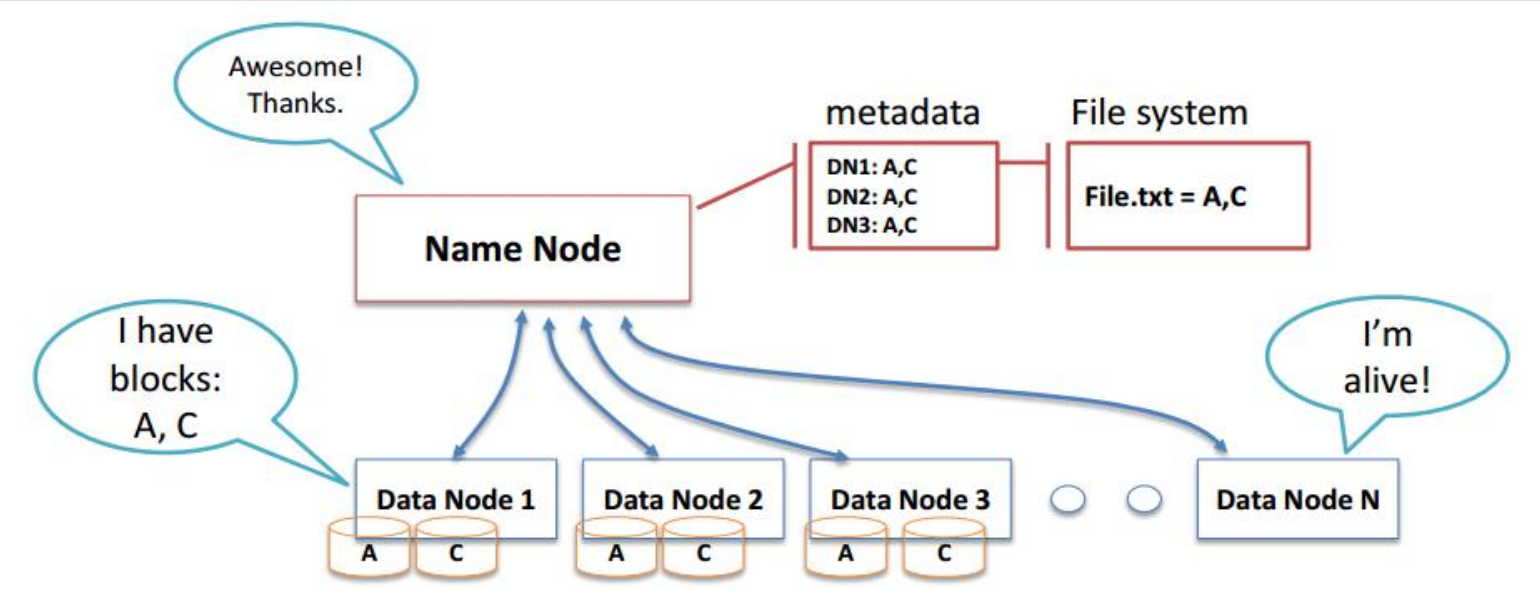
# Functions of DataNodes

- Responsible for serving read and write requests from the file system's clients.
- Perform block creation, deletion, and replication upon instruction from the NameNode.
- Periodically sends a report of all existing blocks to the NameNode (Blockreport)
- Facilitates Pipelining of Data
  - Forwards data to other specified DataNodes

# Communication between NameNode and DataDode

- Heartbeats
  - DataNodes send heartbeats to the NameNode to confirm that the DataNode is operating and the block replicas it hosts are available.
    - Once every 3 seconds
  - The NameNode marks DataNodes without recent Heartbeats as dead and does not forward any new IO requests to them
- Blockreports
  - A Blockreport contains a list of all blocks on a DataNode
- The Namenode receives a Heartbeat and a BlockReport from each DataNode in the cluster periodically

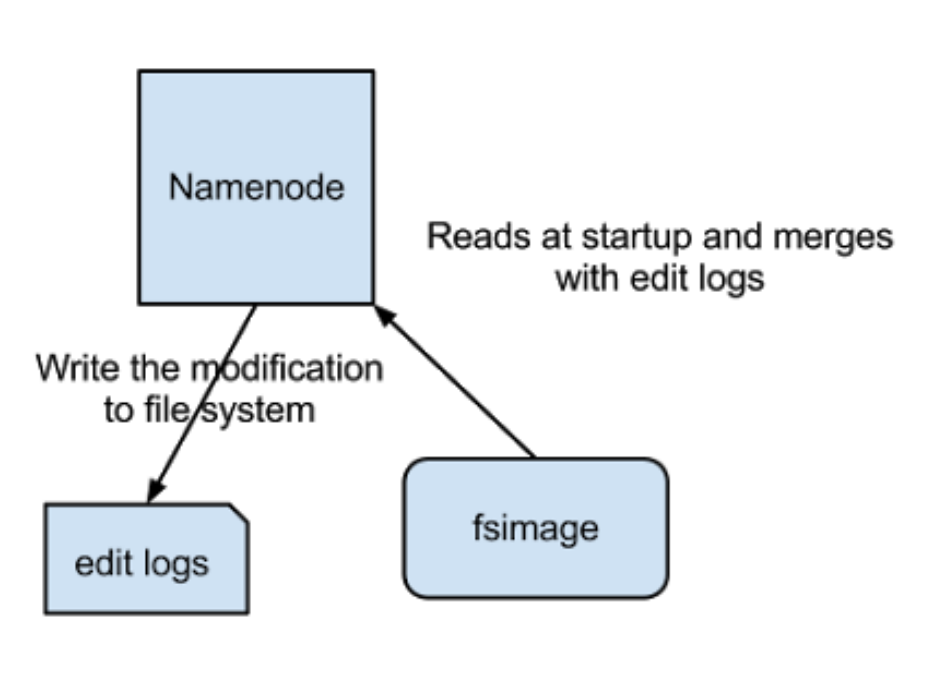
# Communication between NameNode and DataDode




- TCP – every 3 seconds a Heartbeat
- Every 10th heartbeat is a Blockreport
- Name Node builds metadata from Blockreports
- If Name Node is down, HDFS is down

# Inside NameNode

- FsImage - the snapshot of the filesystem when NameNode started
  - A master copy of the metadata for the file system
- EditLogs - the sequence of changes made to the filesystem after NameNode started



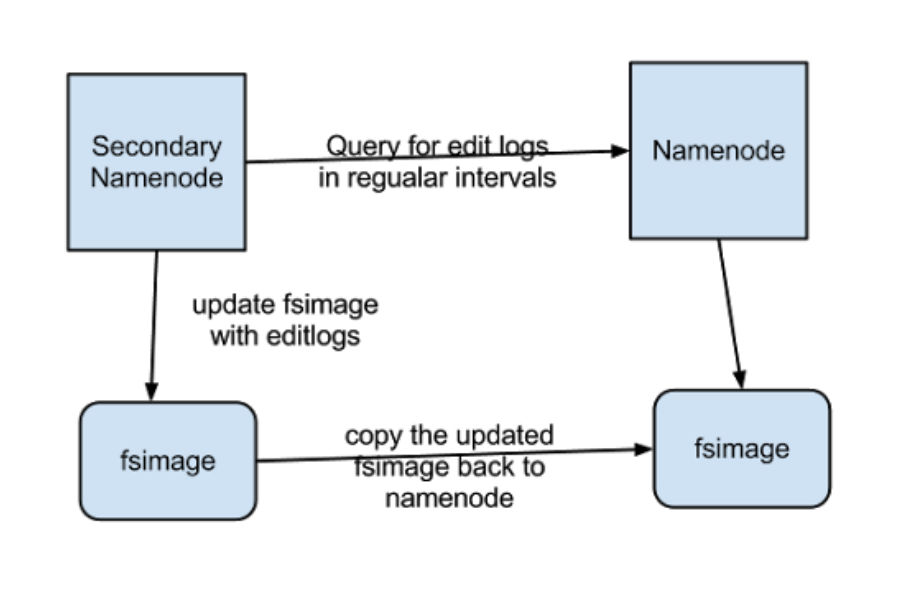
# Inside NameNode

- Only in the restart of NameNode, EditLogs are applied to FsImage to get the latest snapshot of the file system.
- But NameNode restart are rare in production clusters which means EditLogs can grow very large for the clusters where NameNode runs for a long period of time.
  - EditLog become very large , which will be challenging to manage it
  - NameNode restart takes long time because lot of changes has to be merged
  - In the case of crash, we will lose huge amount of metadata since FsImage is very old

- How to overcome this issue?

# Secondary NameNode

- Secondary NameNode helps to overcome the above issues by taking over responsibility of merging EditLogs with FsImage from the NameNode.
  - It gets the EditLogs from the NameNode periodically and applies to FsImage
  - Once it has new FsImage, it copies back to NameNode
  - NameNode will use this FsImage for the next restart, which will reduce the startup time



# 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 (EditLog).
- An application can specify the number of replicas of the file needed: replication factor of the file.

# HDFS Commands

- All HDFS commands are invoked by the bin/hdfs script. Running the hdfs script without any arguments prints the description for all commands.

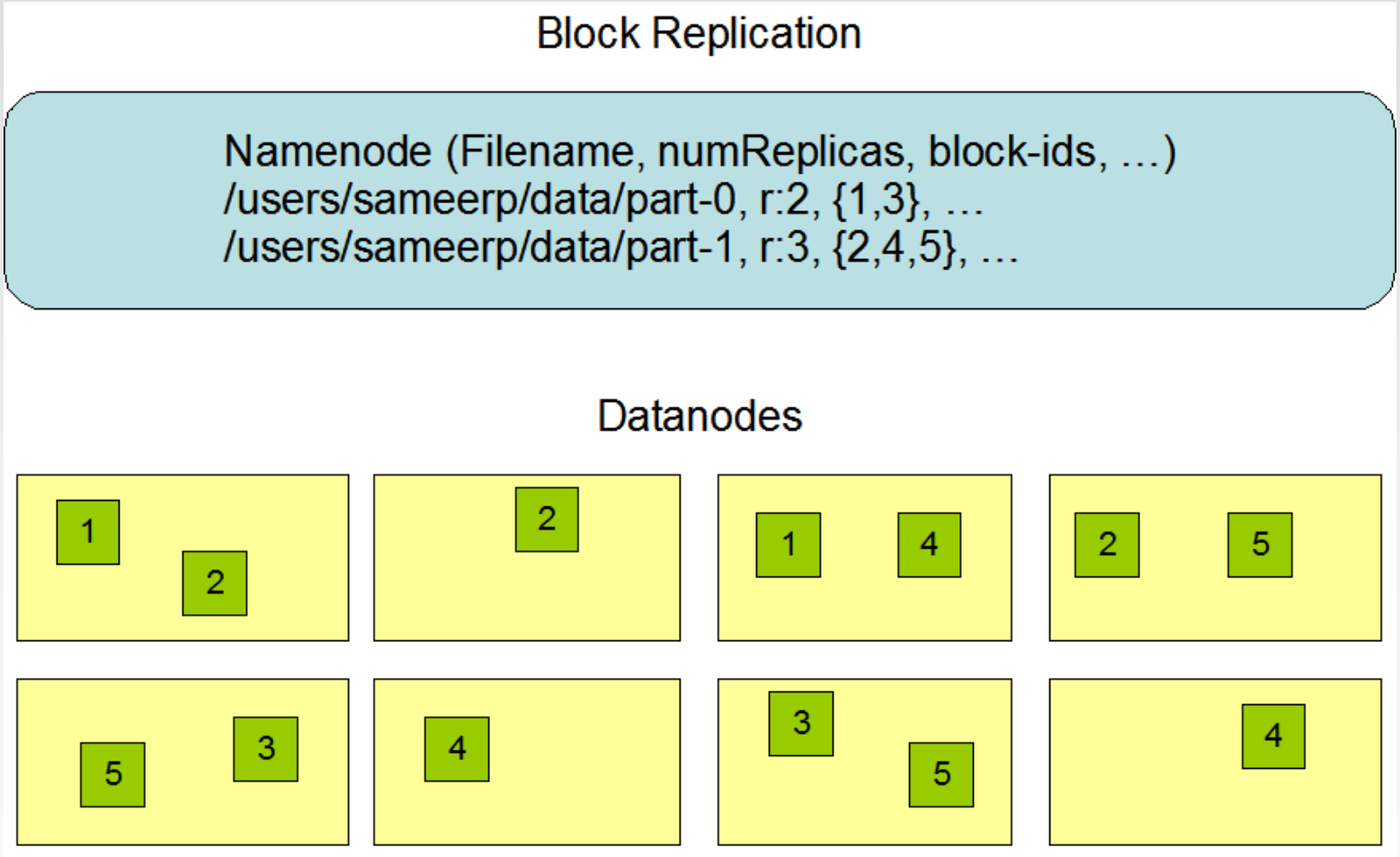
- Usage: hdfs [SHELL_OPTIONS] COMMAND [GENERIC_OPTIONS] [COMMAND_OPTIONS]

  - hdfs dfs [COMMAND [COMMAND_OPTIONS]]

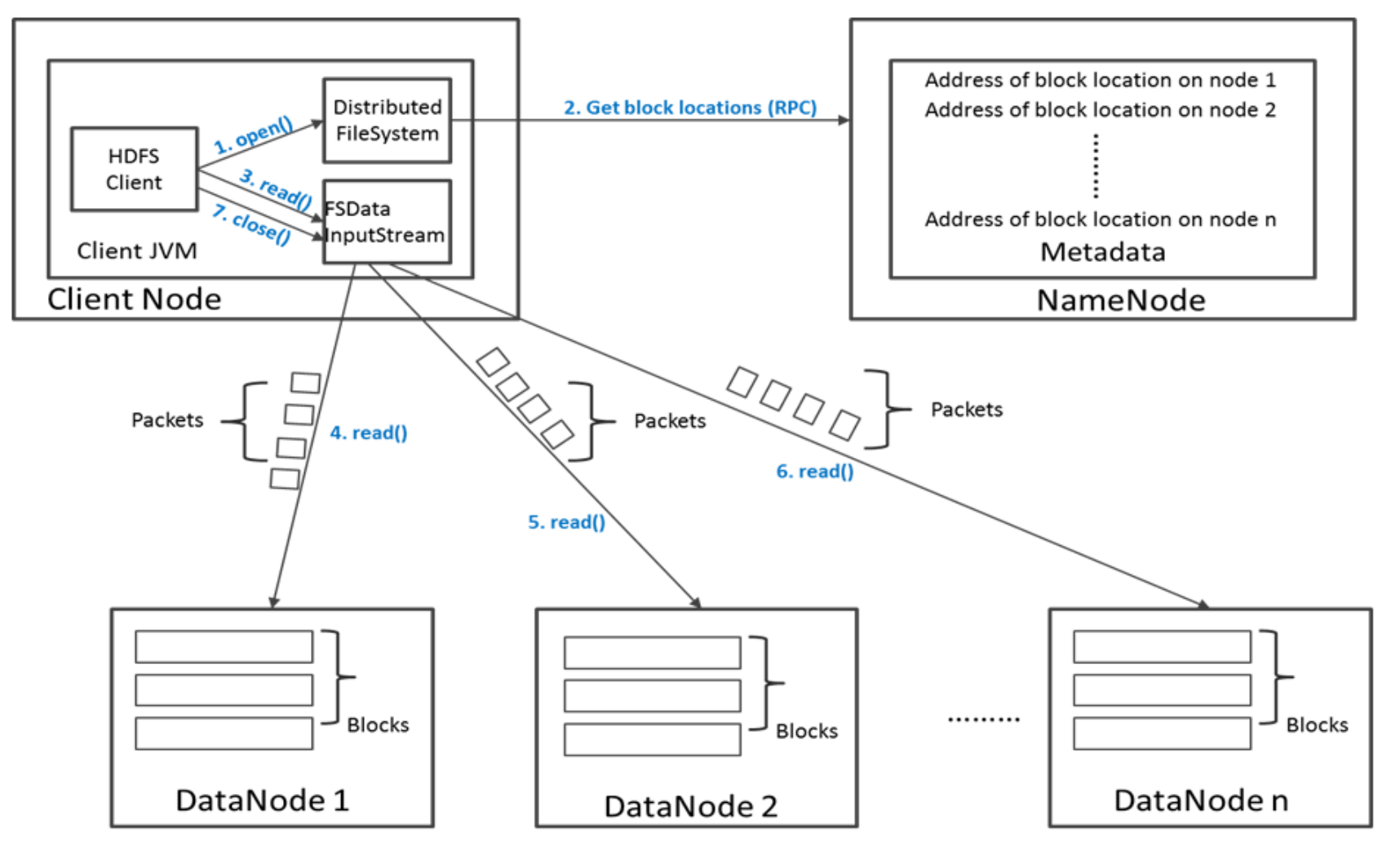
  - Run a filesystem command on the file system supported in Hadoop. The various COMMAND_OPTIONS can be found at File System Shell Guide.

# Data Replication

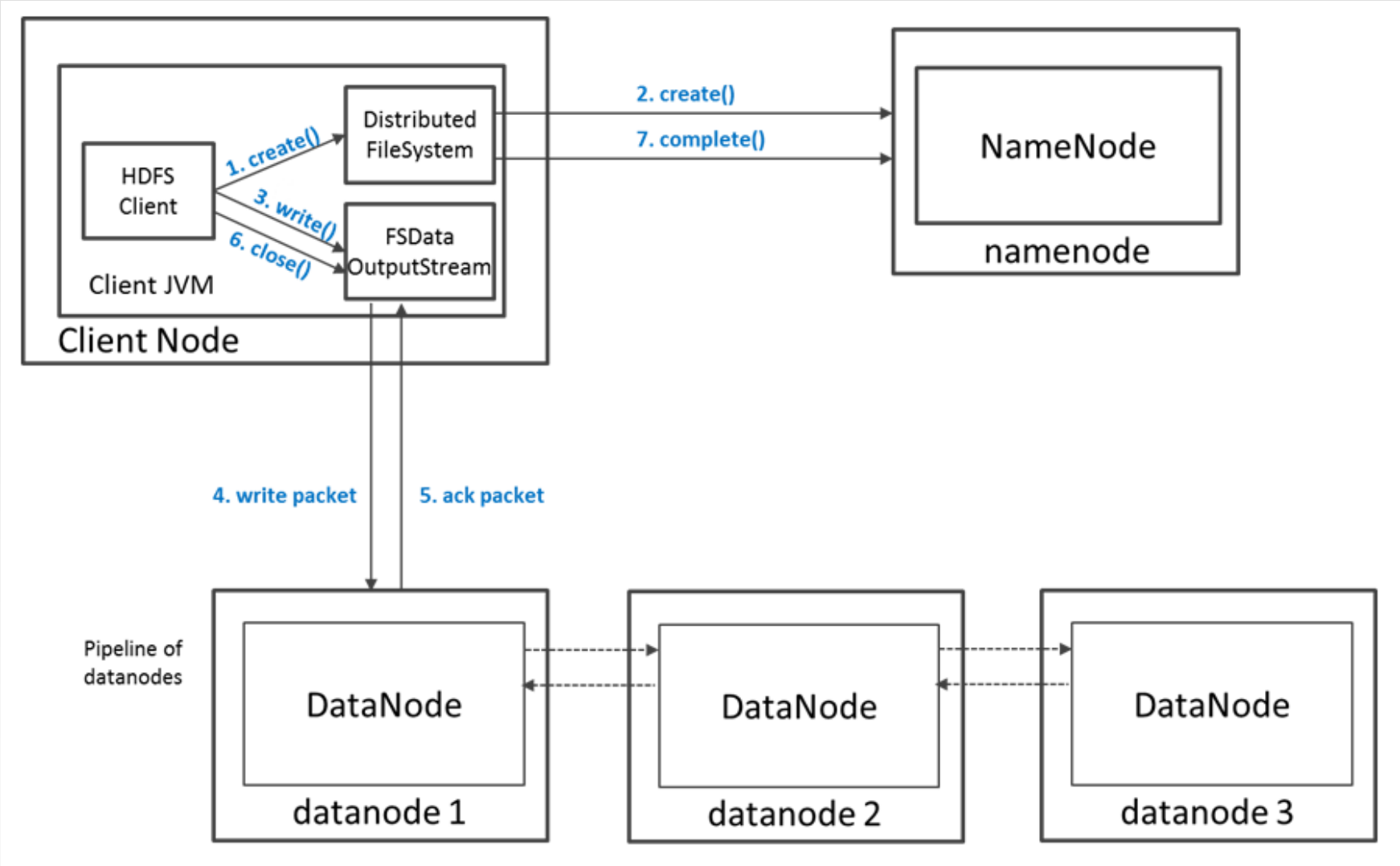
- The NameNode makes all decisions regarding replication of blocks.



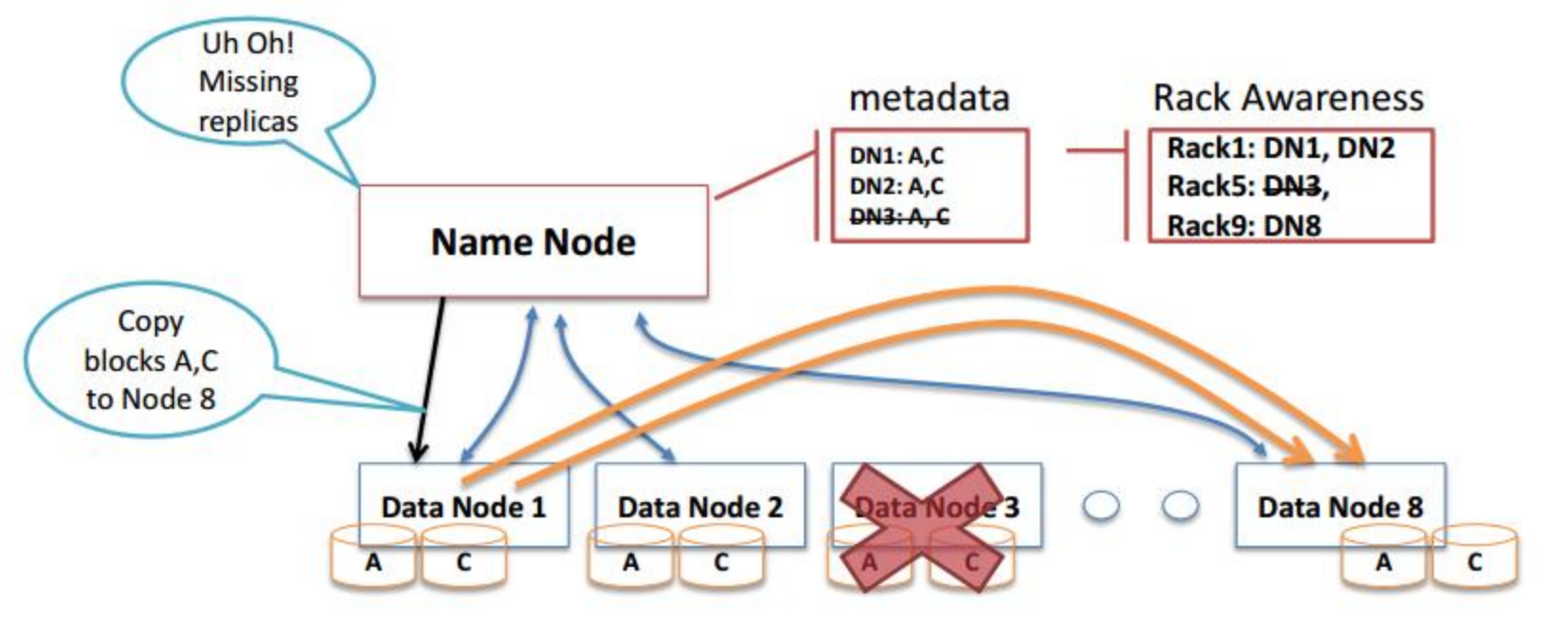
# File Read Data Flow in HDFS



# File Write Data Flow in HDFS



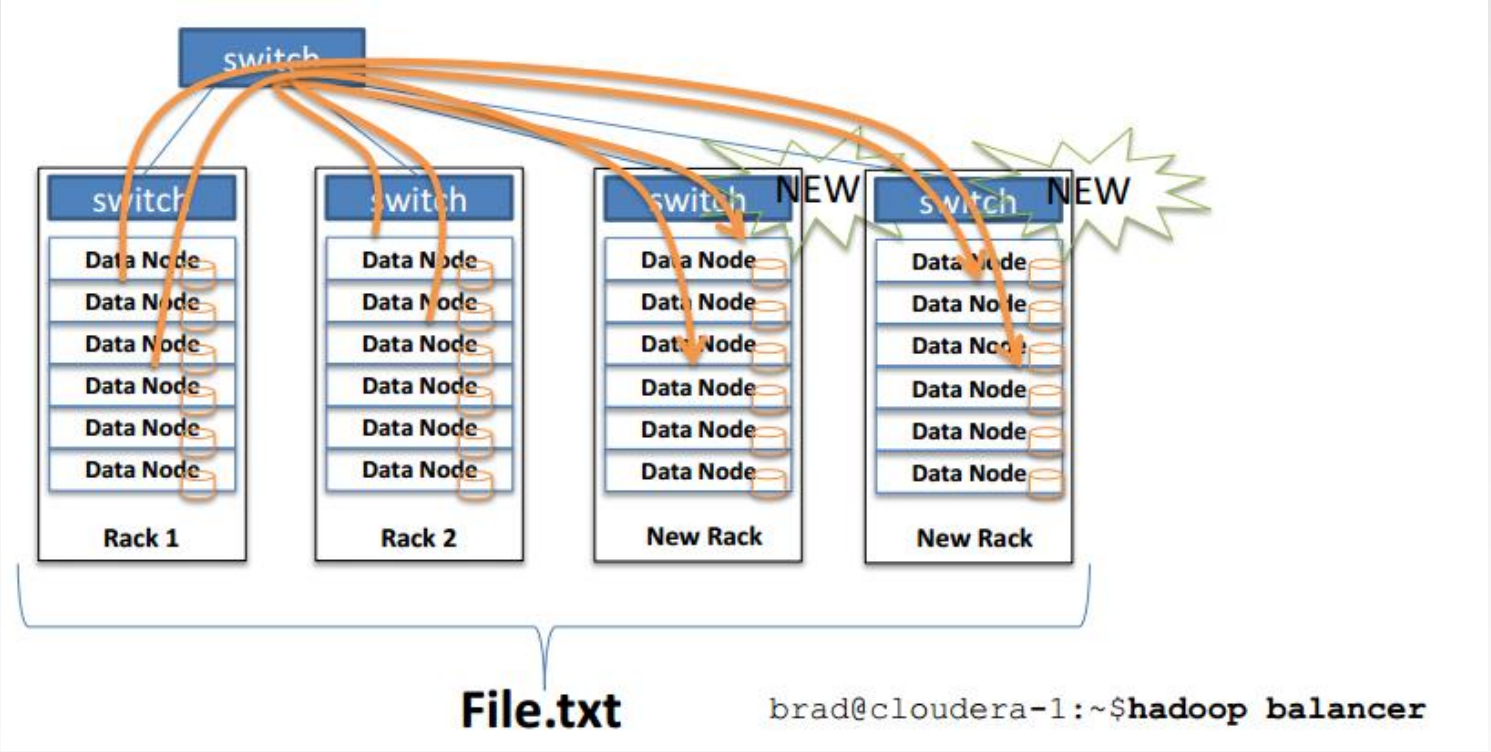
# Replication Engine




- NameNode detects DataNode failures
  - Missing Heartbeats signify lost Nodes
  - NameNode consults metadata, finds affected data
  - Chooses new DataNodes for new replicas
  - Balances disk usage
  - Balances communication traffic to DataNodes

# Cluster Rebalancing

- Goal: % disk full on DataNodes should be similar
  - Usually run when new DataNodes are added
  - Rebalancer is throttled to avoid network congestion
  - Does not interfere with MapReduce or HDFS
  - Command line tool



# 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.

# Metadata Disk Failure

- FsImage and EditLog are central data structures of HDFS. A corruption of these files can cause a HDFS instance to be non-functional.
  - A NameNode can be configured to maintain multiple copies of the FsImage and EditLog
  - Multiple copies of the FsImage and EditLog files are updated synchronously

# HDFS Erasure Coding

- Replication is expensive – the default 3x replication scheme in HDFS has 200% overhead in storage space and other resources.

- Therefore, a natural improvement is to use Erasure Coding (EC) in place of replication, which provides the same level of fault-tolerance with much less storage space.
  - Erasure Coding transforms a message of *k* symbols into a longer message with *n* symbols such that the original message can be recovered from a subset of the *n* symbols.
  - In typical Erasure Coding (EC) setups, the storage overhead is no more than 50%. Replication factor of an EC file is meaningless. It is always 1 and cannot be changed via -setrep command.