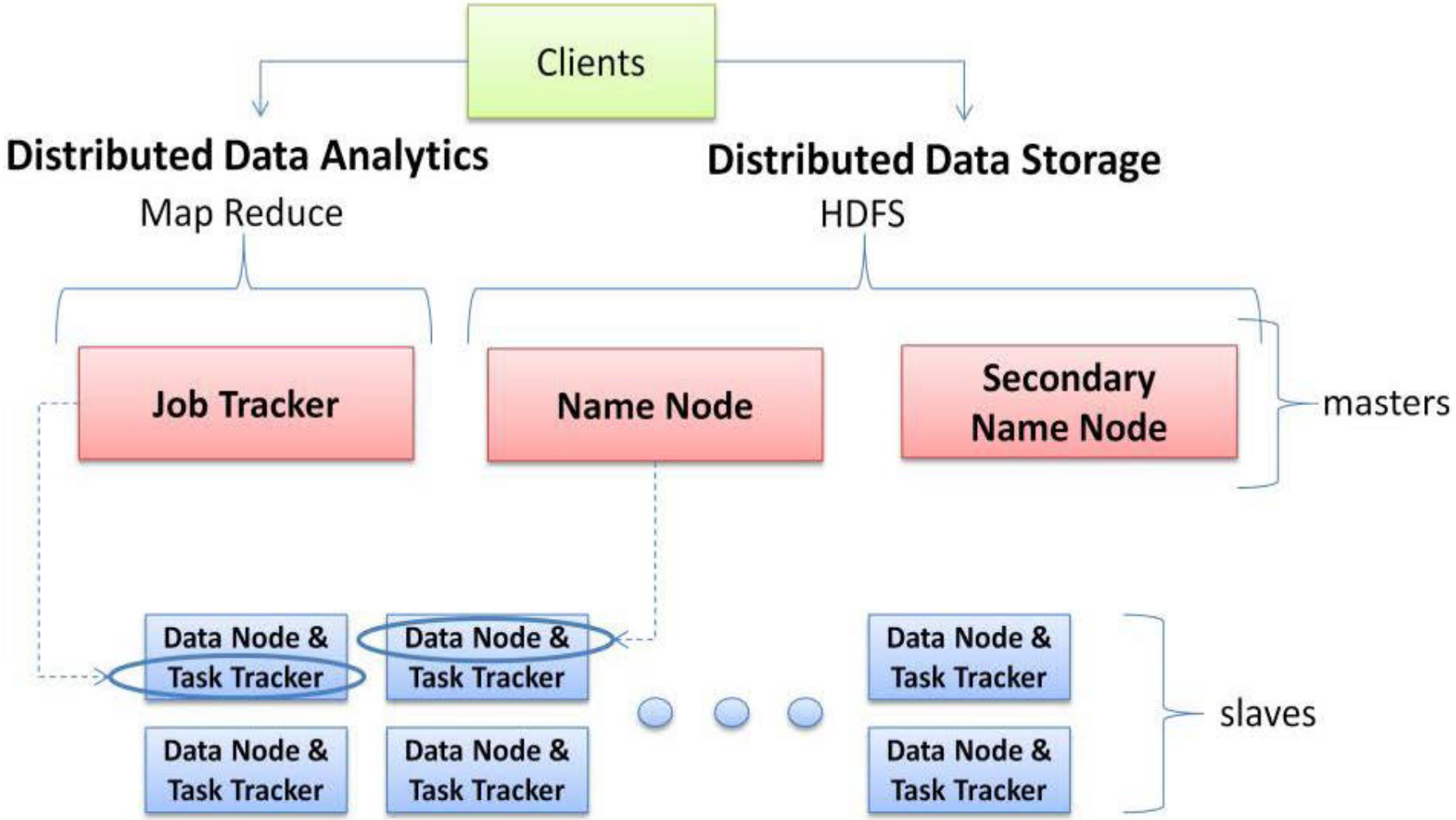
Session 2

Hadoop Distributed File System (HDFS)

What For Today!!!

- ✓ *HDFS Features & Design Goals*
- ✓ *HDFS Operation Principle*
- ✓ *Data Locality, Rack Awareness*
- ✓ *Writing and Reading Files*
- ✓ *NameNode Memory Considerations*
- ✓ *Secondary NameNode – FSImage & EditLog*
- ✓ *Data Node – Heartbeats & Block Report*

Hadoop Server Roles



HDFS Goals

- Store millions of large files (GBs) totaling petabytes
- Scale out with linearly as more nodes are added
- JBOD instead of RAID
- Optimized for large, streaming r + w, instead of low latency access to small files. Batch > Interactive
- Self-healing, recover automatically from disk/node failures
- Support MapReduce processing

HDFS Background

- Based on Google's GFS paper.
- Provides cheap redundant storage for massive amounts of data.
- Operates 'on top of' existing file systems
- At ingestion data blocks are distributed across the nodes.
- Each block is typically 64Mb or 128Mb in size.
- Each block is replicated 3x times by default.
- Replicas are stored on different nodes.

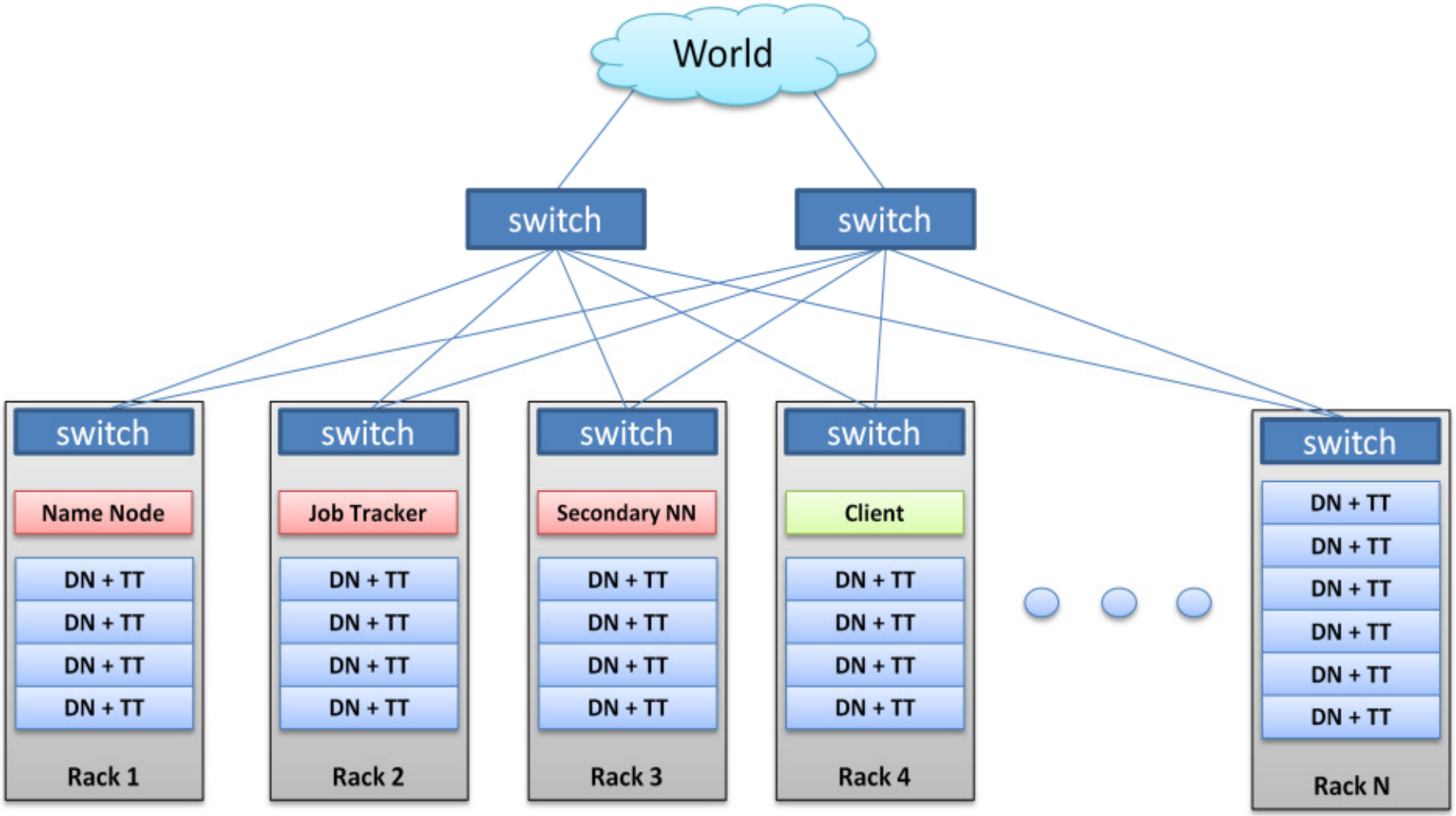
HDFS Background (Contd)

- Single Name Node stores metadata and co-ordinates data access
- Actual data is stored by Data Nodes
- Files in HDFS are ‘write once’; append also available
- Instead of bringing data to processors, it brings the processing to the data
- Earlier Hadoop releases had Name Node HA as SPOF

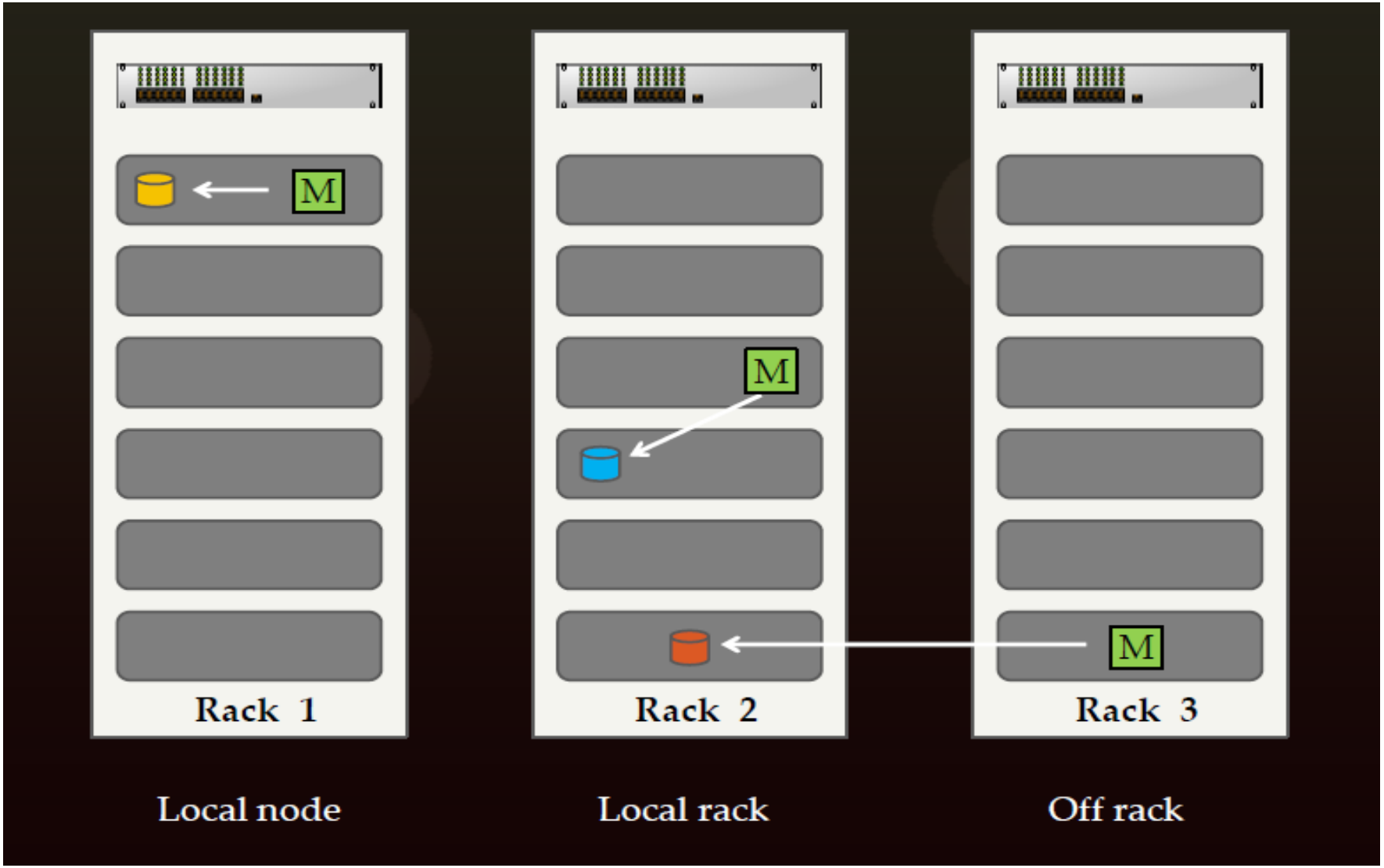
HDFS Daemons

Daemon	# per cluster	Purpose
NameNode	1	Store file system metadata, file to block mappings, provide a global picture of file system
Secondary NameNode	1	Perform WAL checkpointing for NN (combines Journal + Checkpoint and rewrites a new Checkpoint)
DataNode	Many	Store and retrieves block data (file contents)

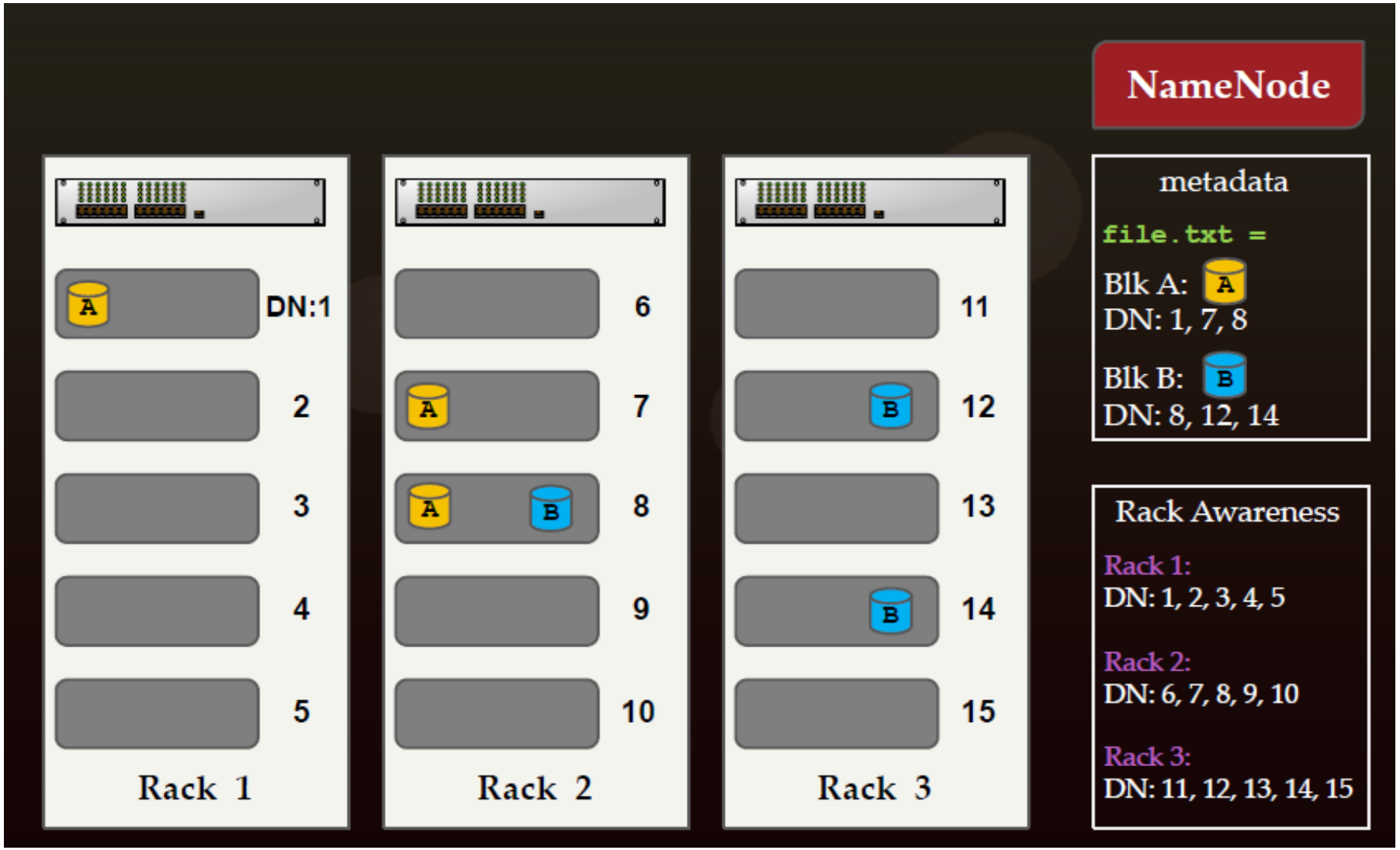
Network Layout



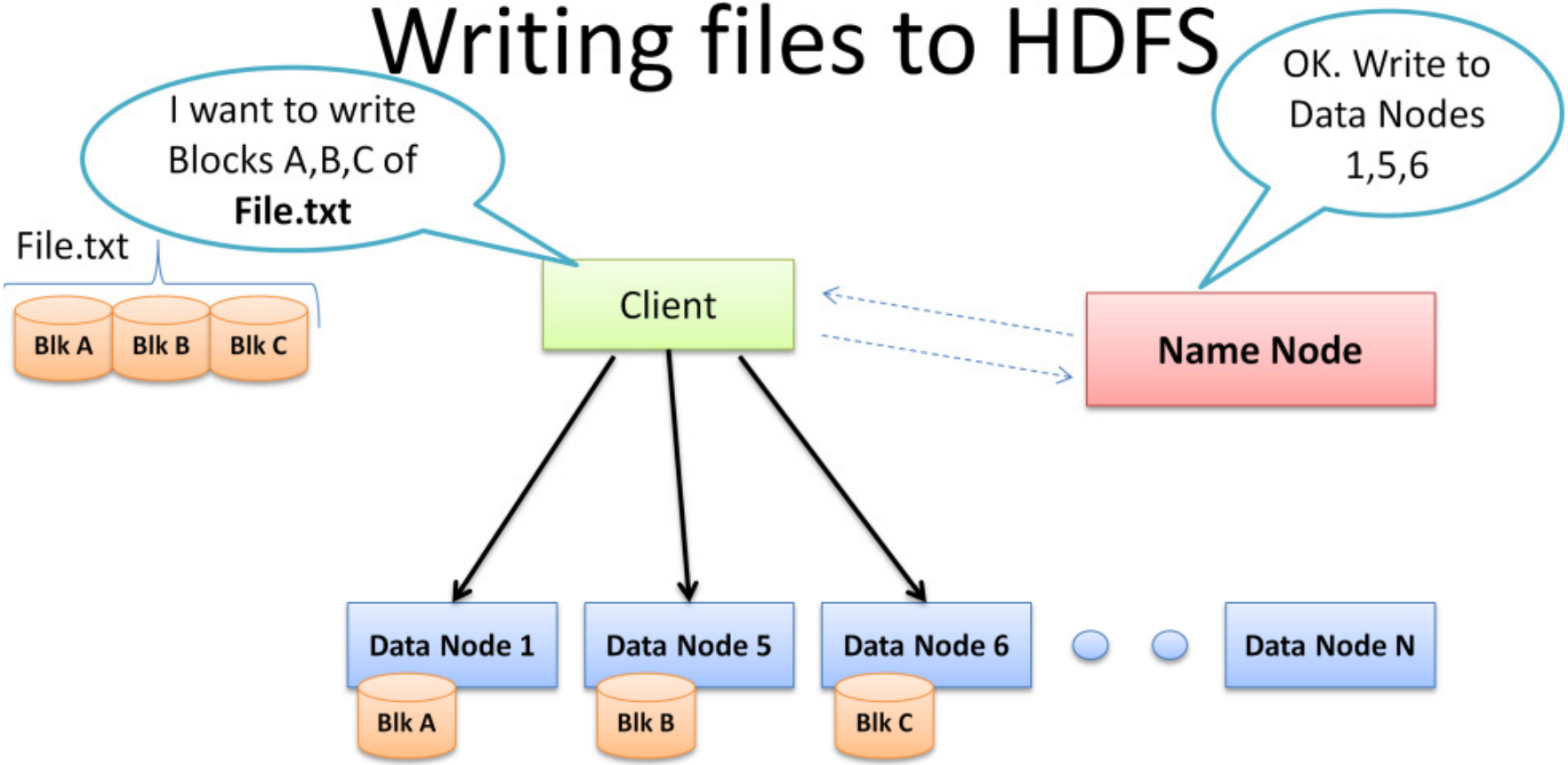
Data Locality



Rack Awareness

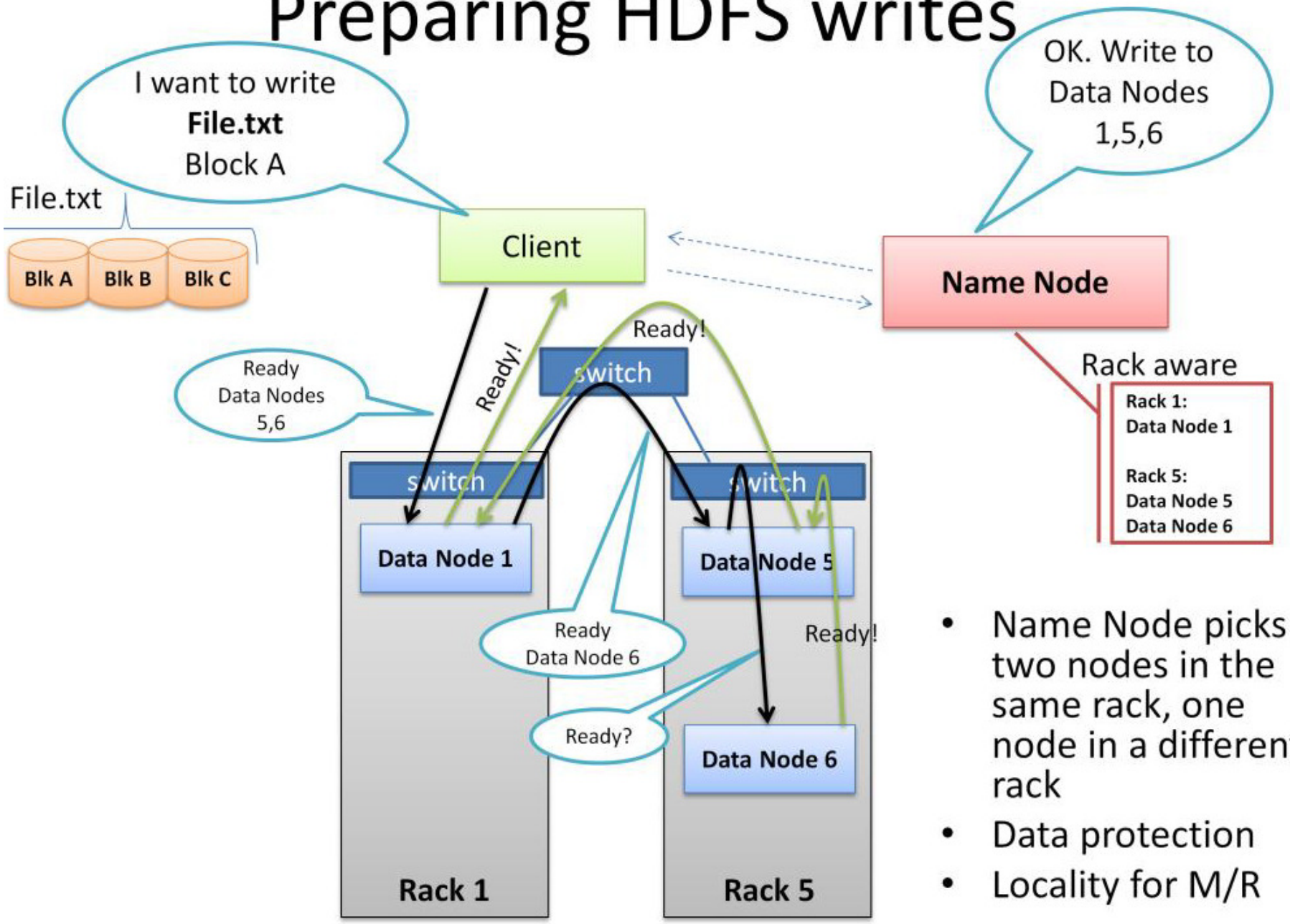


Writing files to HDFS



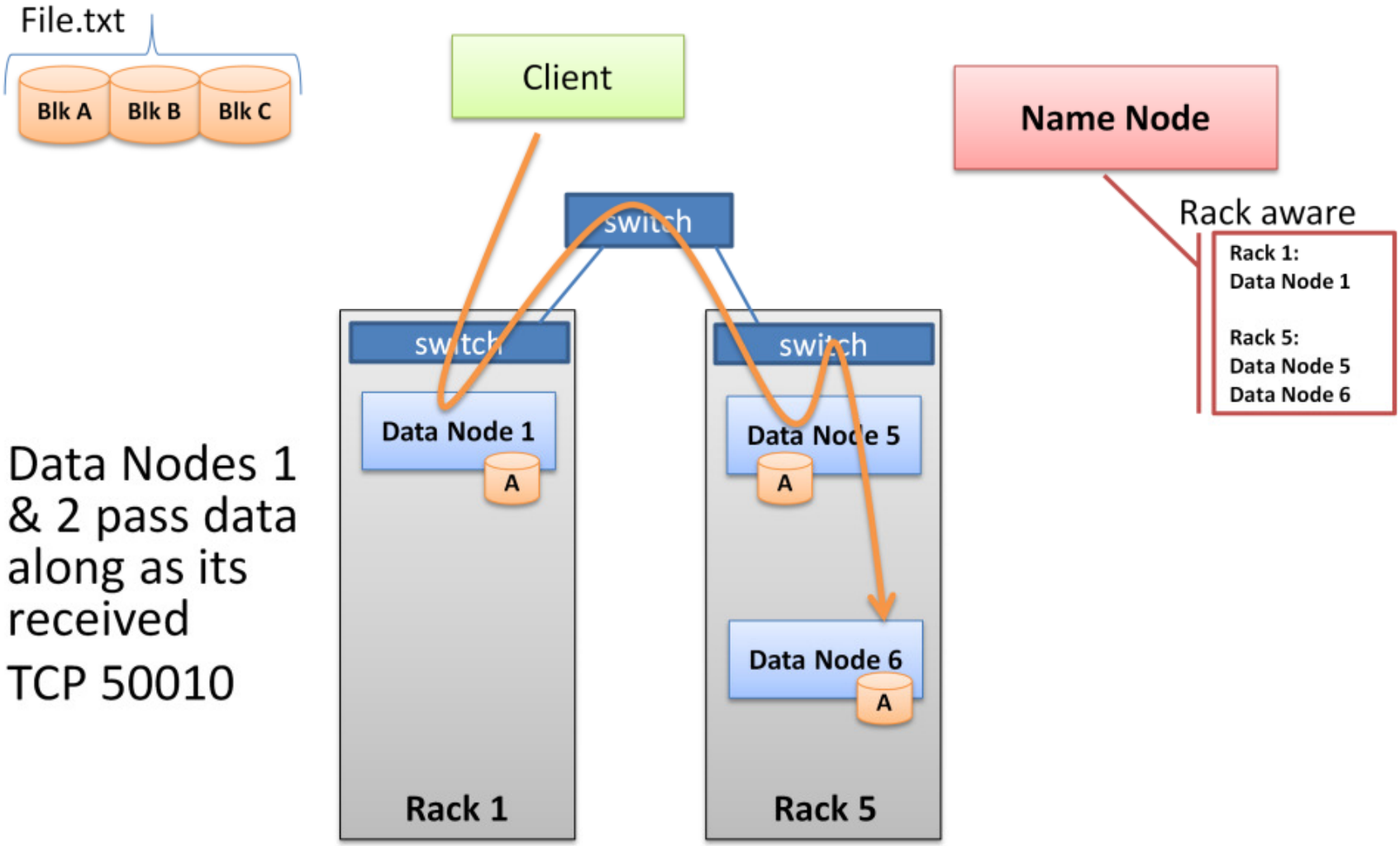
- Client consults Name Node
- Client writes block directly to one Data Node
- Data Nodes replicates block
- Cycle repeats for next block

Preparing HDFS writes



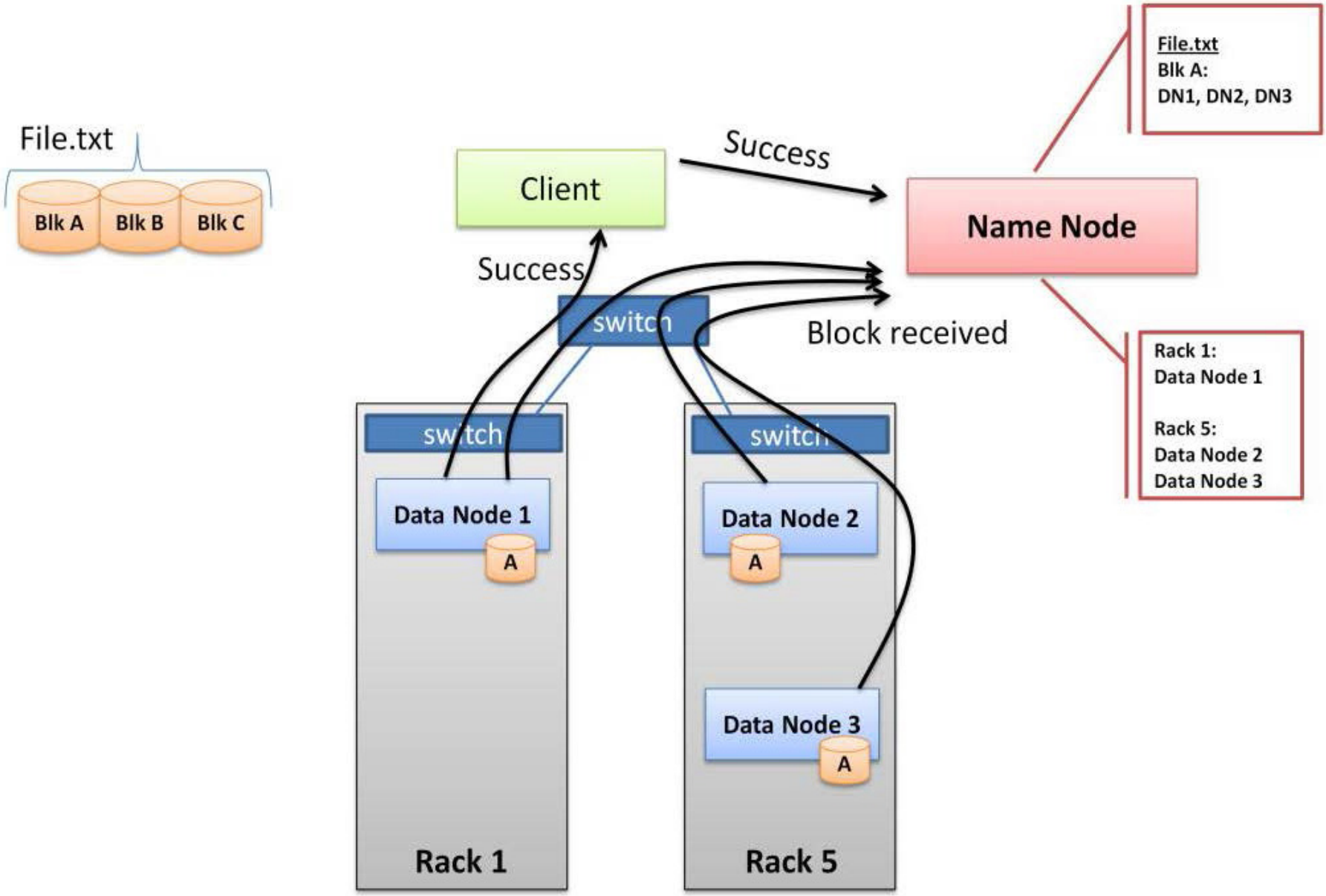
- Name Node picks two nodes in the same rack, one node in a different rack
- Data protection
- Locality for M/R

Pipelined Write

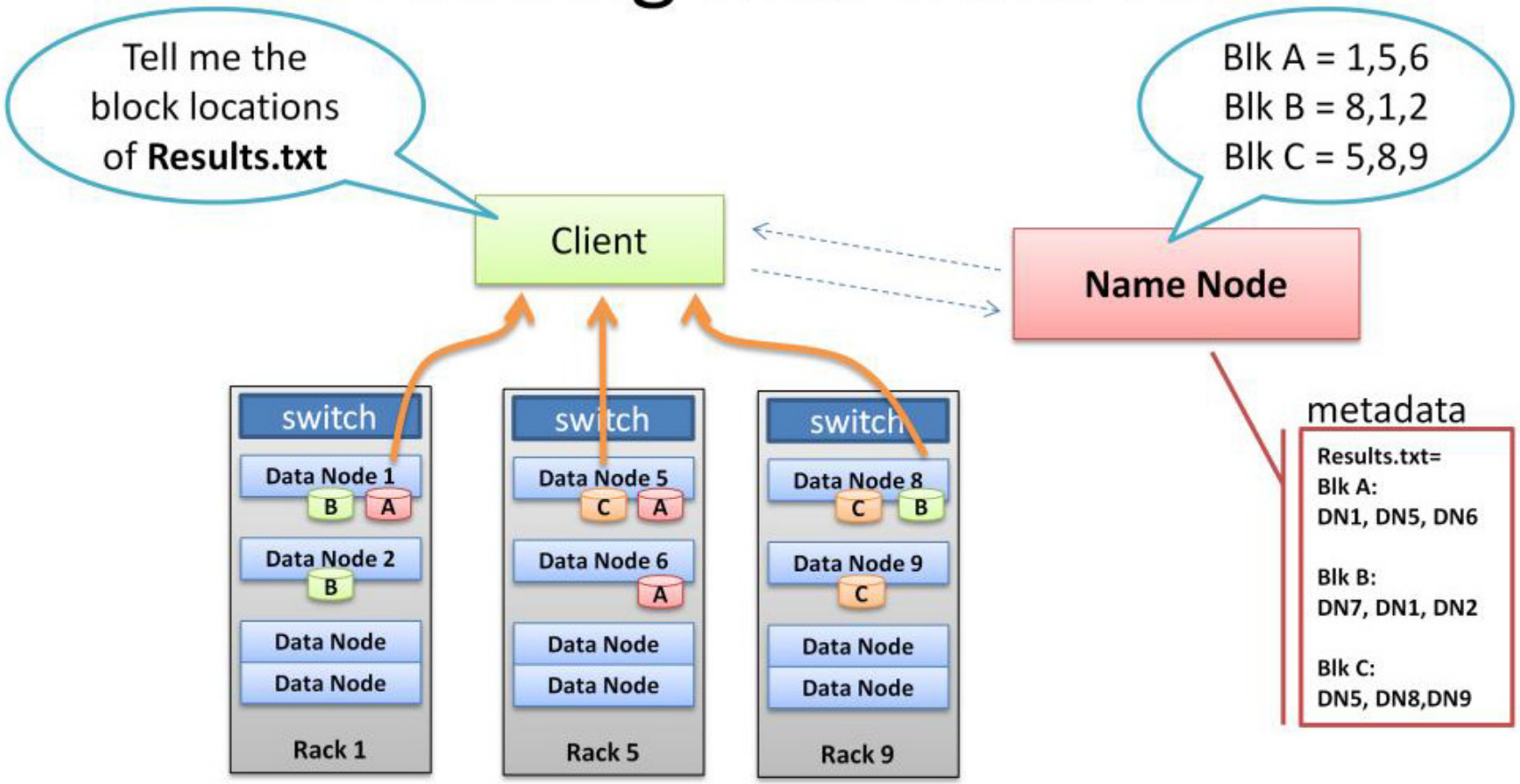


- Data Nodes 1 & 2 pass data along as its received
- TCP 50010

Pipelined Write

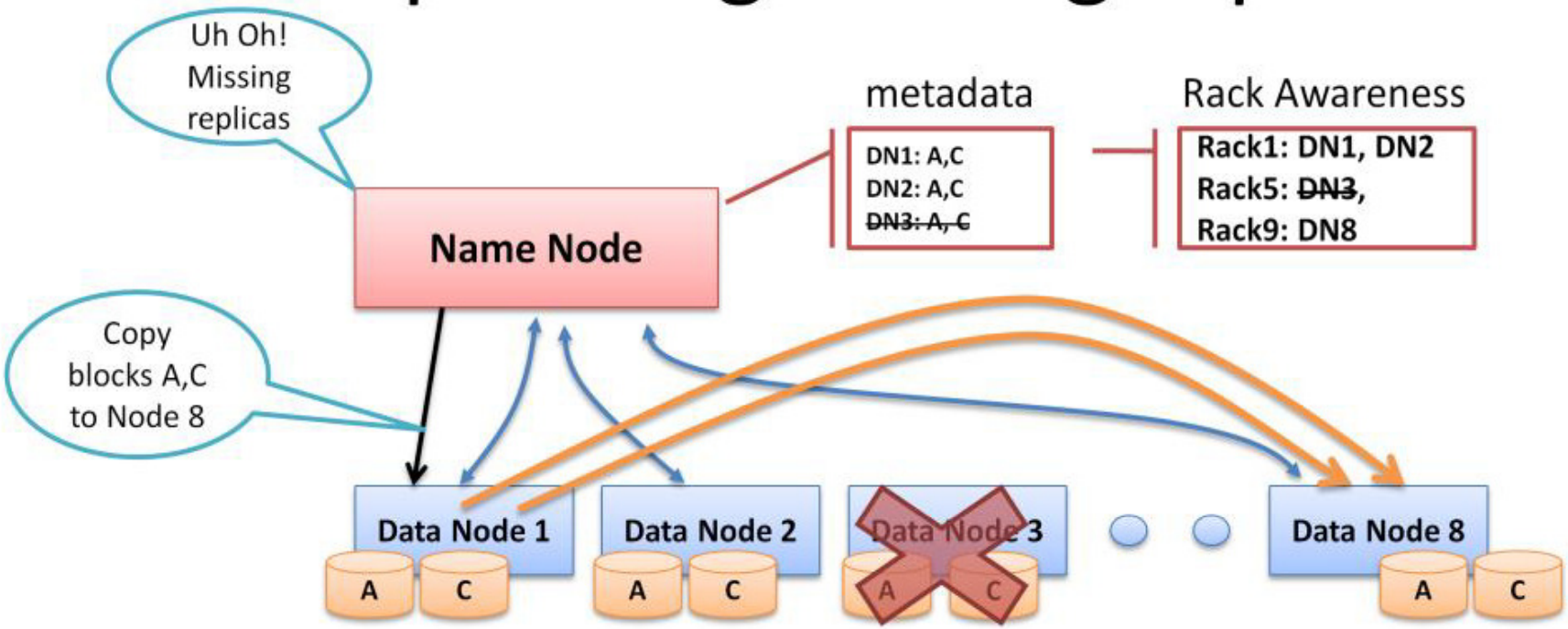


Client reading files from HDFS



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

Re-replicating missing replicas



- 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

Checkpoint and Journals

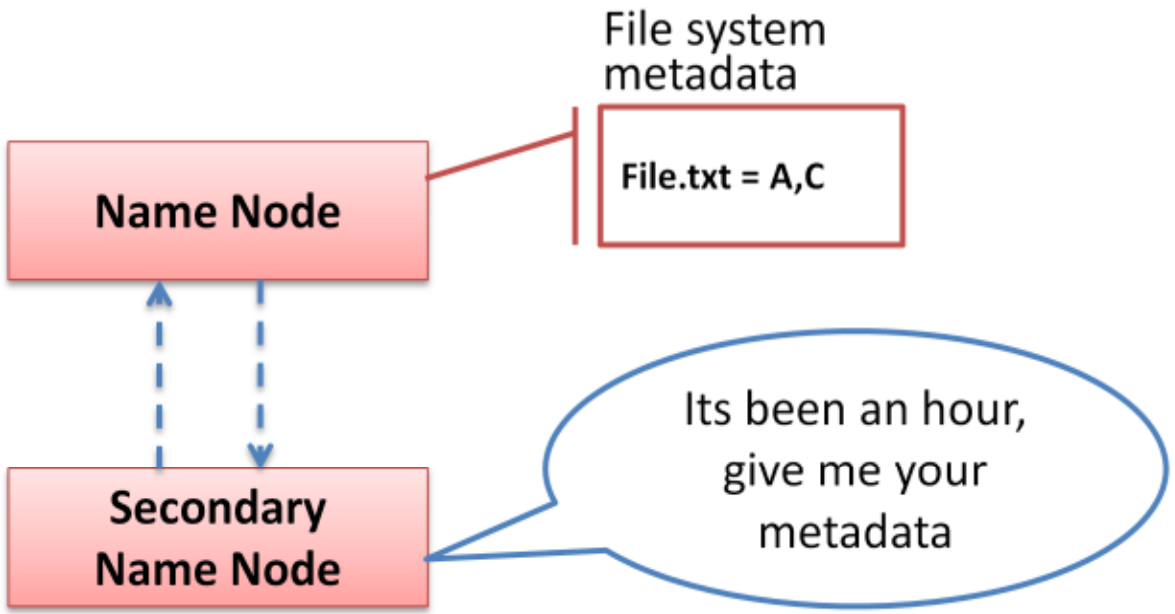
- Serves filesystem metadata entirely from RAM
- Rough estimate: metadata for 1000 blocks = 1GB

Checkpoint/fsimage: complete snapshot of FS metadata

Journals/edits/WAL: incremental modifications made to metadata

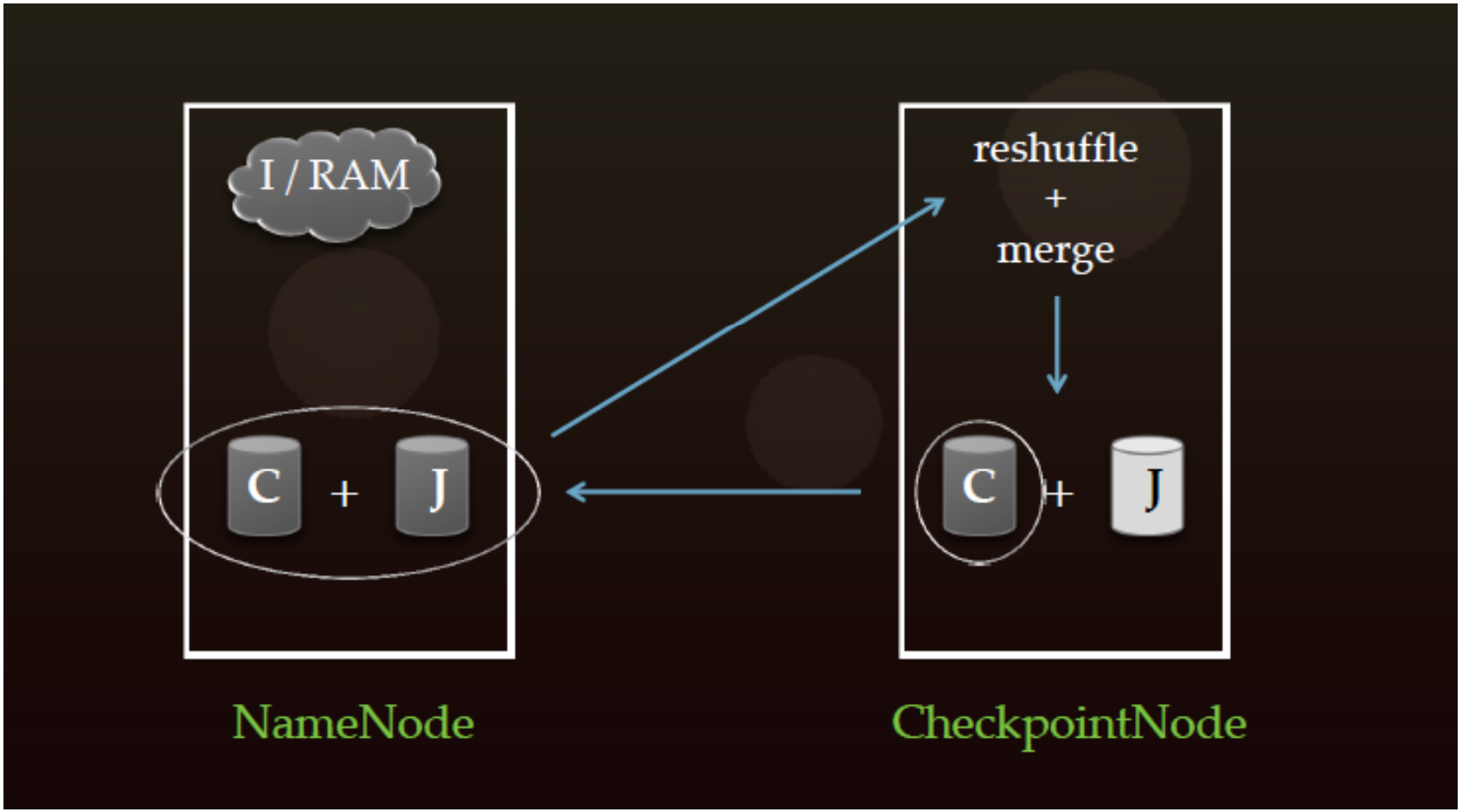
- **HDFS Metadata ::** list of Blocks + inodes (permissions, access times, mod. Times, namespace Q, diskspace Q) + Location of Replicas

Secondary 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

Secondary Name Node



Secondary Name Node

1) SNN instructs NN to roll its edits file and begin writing to edits.new
2) SNN copies NN's fsimage/checkpoint and edits/journal file to its local checkpoint directory
3) SNN loads fsimage into RAM and replays edits on top of it. SNN then writes a new, compacted fsimage to local disk.
4) SNN sends the new fsimage to the NN which adopts it
5) NN renames edits.new to edits

This process repeats every hour by default or when the NN's edits file reaches 64 MB.

File Leases

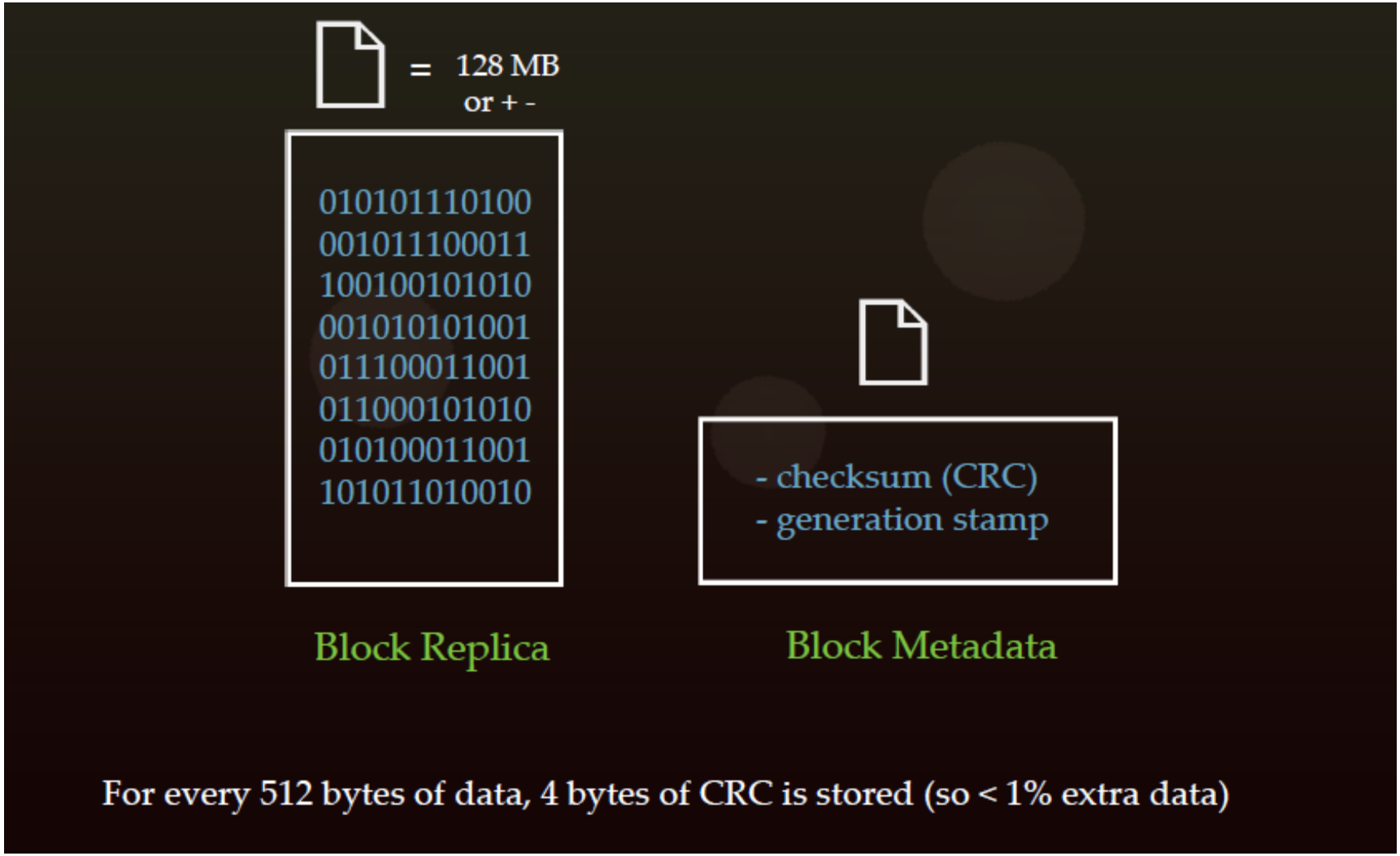
- Lease renewed by
- Write lease does not affect reads

Opening a file for write, generates 2 leases:

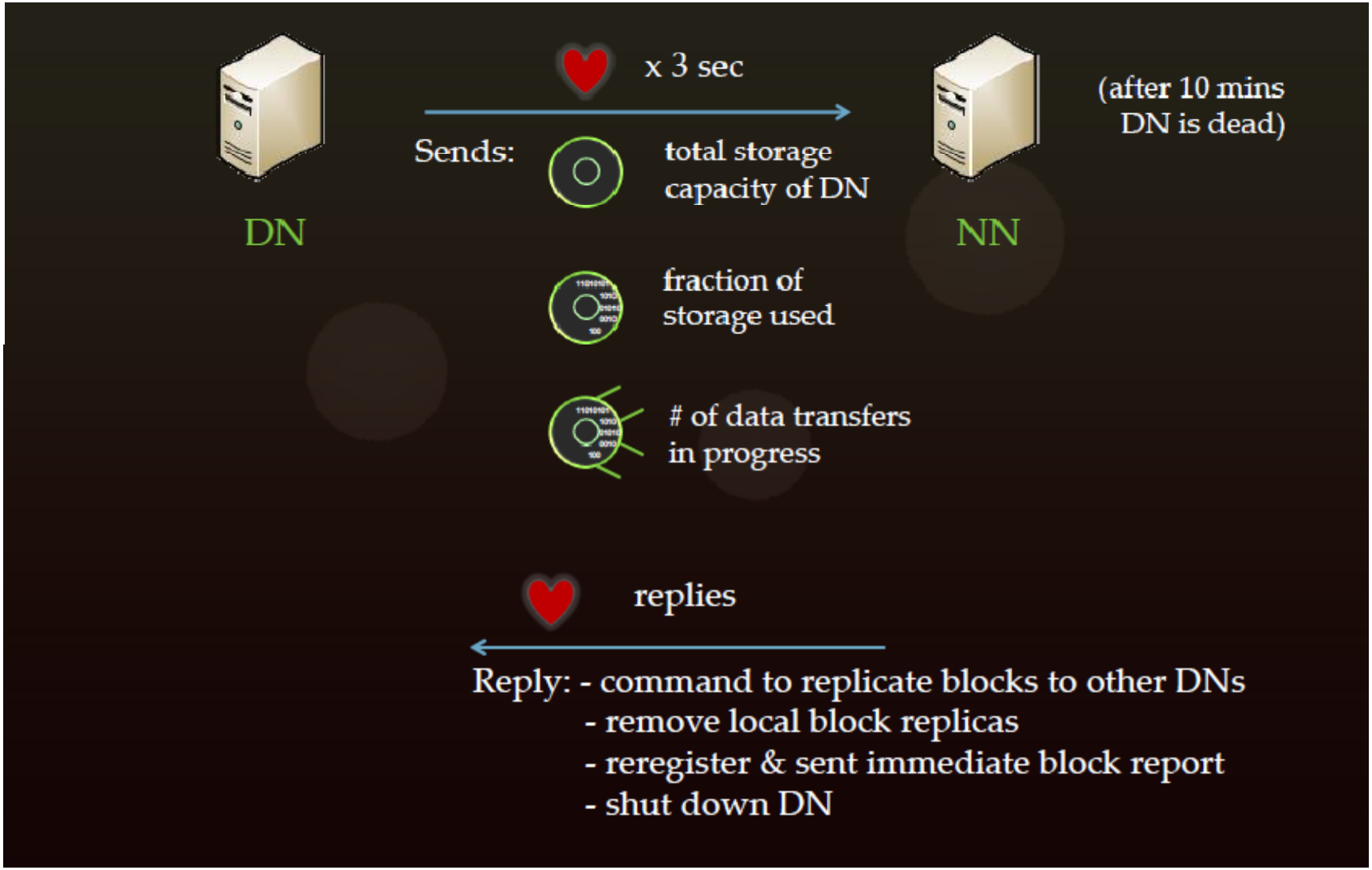
Soft Lease: while open, client has exclusive access to file
if expired, another client can preempt lease

Hard Lease: expires in 1 hour
if not closed, HDFS forcefully recovers lease

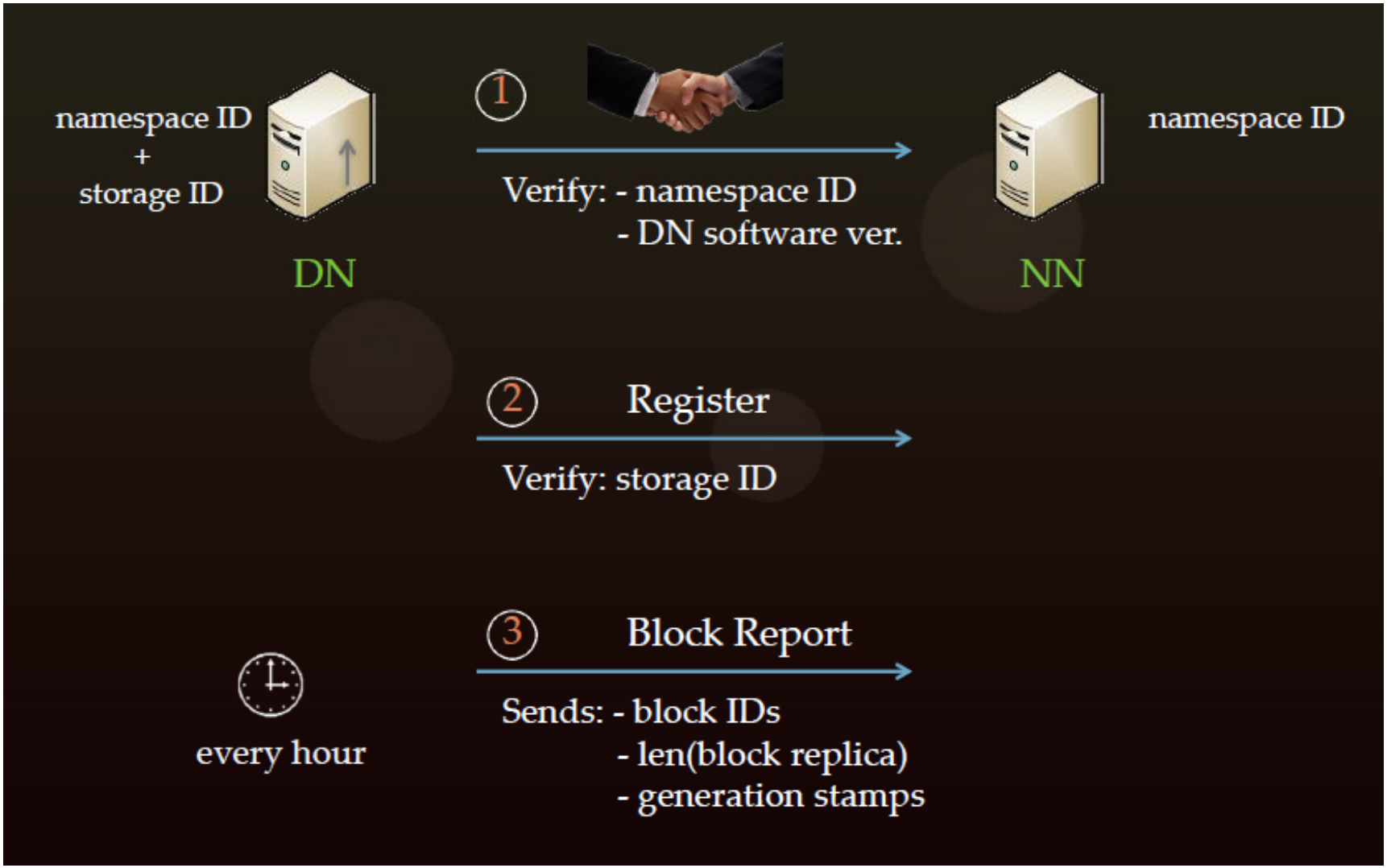
DataNode Internals



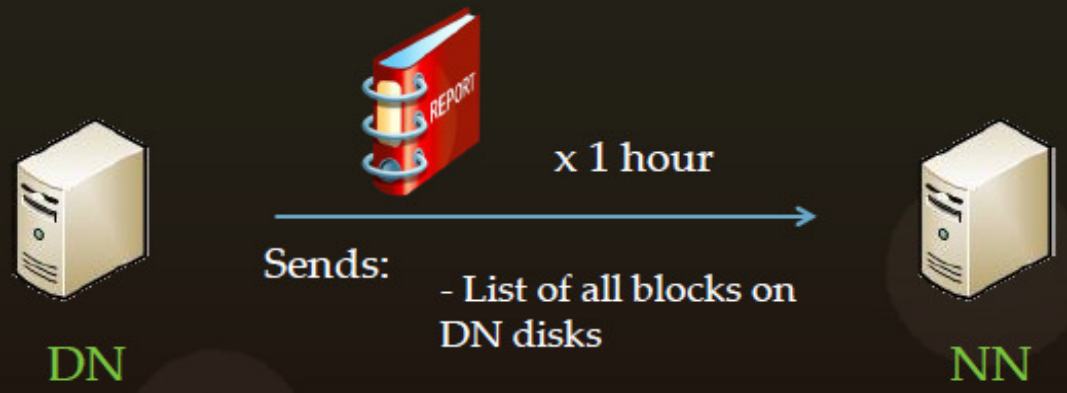
DataNode HeartBeats



DataNode StartUp



DataNode BlockReport



NN stores the file -> block mappings on disk, but not the location of blocks.

NN has to rebuild the location of blocks via the block reports every time it restarts. NN is in safe mode until it 99% of the block locations are accounted for

Thank You