CS202 – Computer Security

Distributed Filesystems

March 10, 2025

Moving Computation is Cheaper than Moving Data

What is Hadoop?

- Flexible infrastructure for large scale computation & data processing on a network of commodity hardware
- Completely written in Java
- Open source & distributed under Apache license
- Hadoop Common, HDFS, & MapReduce

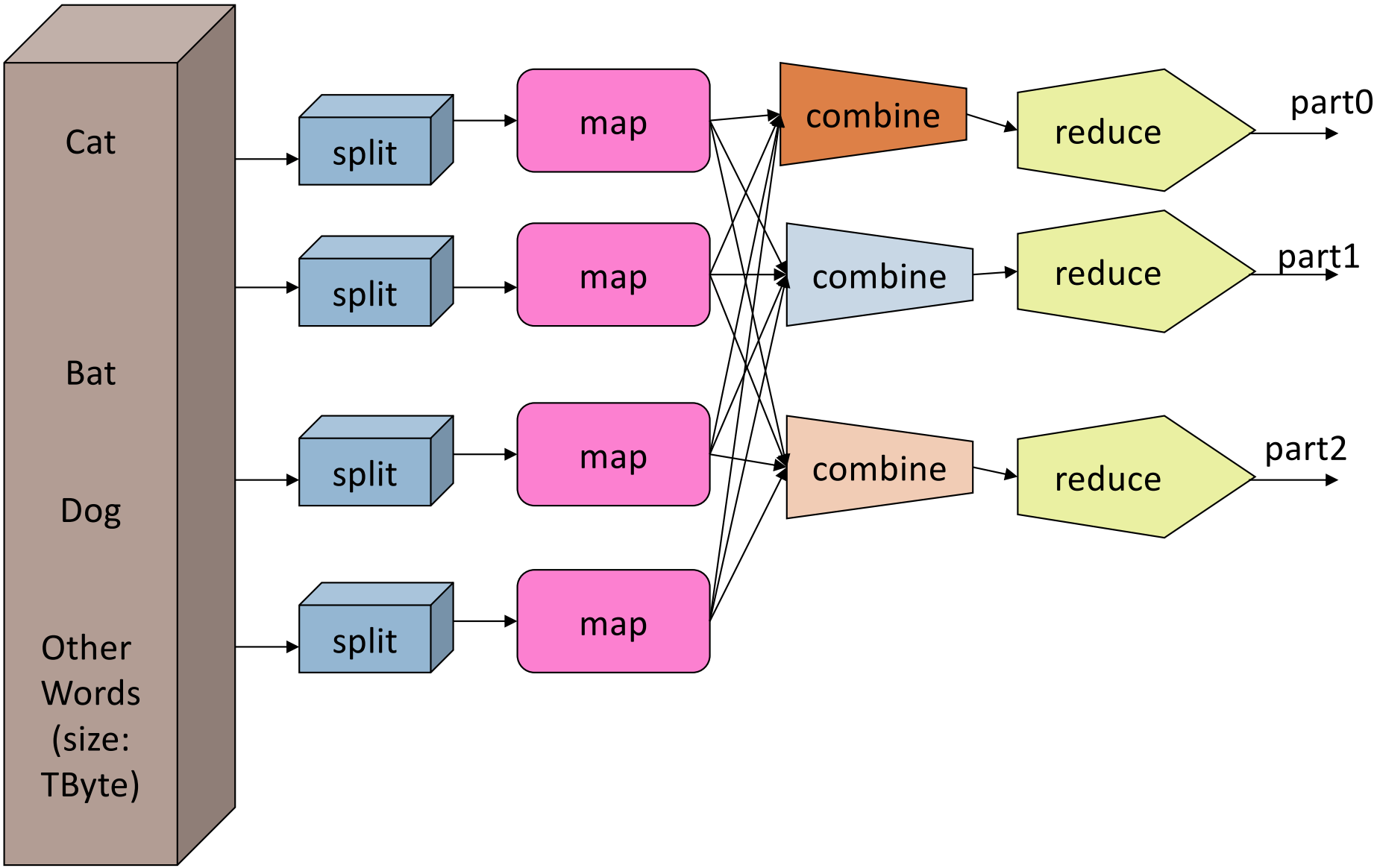
Origins of Hadoop

- Google File System 2003
- MapReduce paper in OSDI 2004 (Google)
  - New programming paradigm to handle data at internet scale
  - Google developed GFS
- Hadoop/HDFS open-source version of MapReduce
  - Hadoop started as a part of the Nutch project.
  - In Jan 2006 Doug Cutting started working on Hadoop at Yahoo
  - Yahoo gave it to Apache
- First release of Apache Hadoop in September 2007
  - Jan 2008 - Hadoop became a top-level Apache project

Motivation: Big Data!

- What is Big Data?
  - Google (over 20~ PB/day)
- Where does it come from?
- Why so much data?
  - Information is everywhere
  - Existing systems (vertical scalability)
  - Why Hadoop (horizontal scalability)?
- MapReduce!

MapReduce


Cat
Bat
Dog
Other Words (size: TByte)
split
split
split
split
map
map
map
map
combine
combine
combine
reduce
reduce
reduce
part0
part1
part2

Workload Assumptions

- Hardware failure common
  - Norm, not exception, using commodity hardware
- Files are large, and numbers are limited (millions not billions)
  - Large data sets
- Two main types of reads: large streaming reads and small random reads
  - Files rarely modified (simplifies coherence)
  - Sequential writes that append to files
- High sustained bandwidth preferred to low latency

Dictates Problems

- How to store very large files?
  - Different problem than faced by traditional distributed filesystems (e.g., NFS)
- How to ensure reliable storage of large files?
  - Of file metadata and data
- Filesystem operations to support
  - Should we support the traditional POSIX API?
- Attain high throughput for computations
  - What can we do to improve performance?

Secondary Problems

- Paper also discusses…
- Ensure operational consistency
  - Across a huge number of nodes
- Balance the use of storage
  - Try to avoid running out on one node when other has free space
- Detect failures
  - Of nodes and blocks
- And may be concerned about…
  - Ensuring data security – secrecy, integrity, etc.

Very Large Files

- How do we store and access very large files?
  - Operate with modest number of very large files
  - Mostly read and occasionally appended

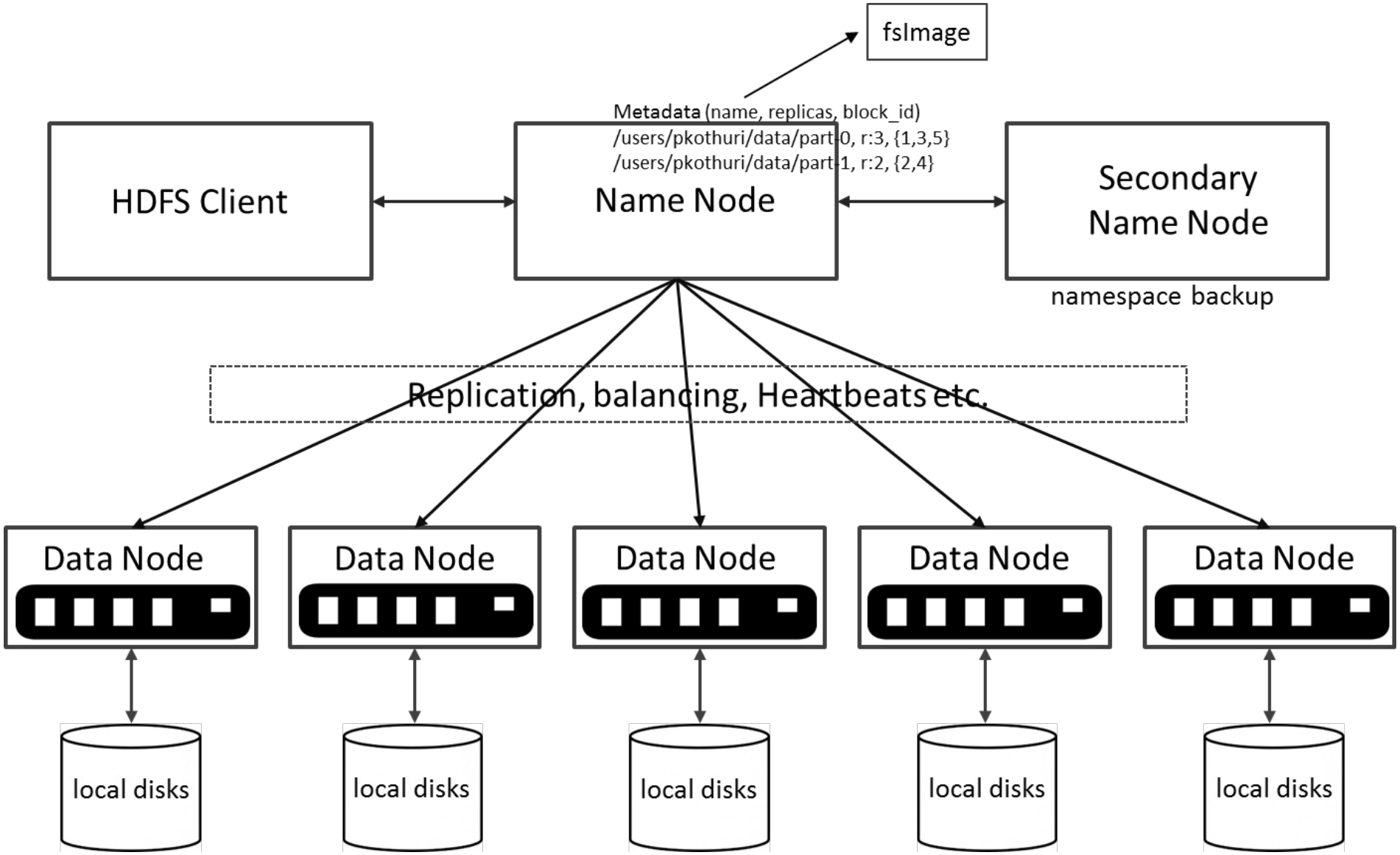
Very Large Files Approach

- Separate metadata from data
- Store metadata centrally – not so large
- Distribute data across distributed nodes
  - May be many for very large files
- Replicate storage for reliability
- Keep it all consistent and keep costs under control

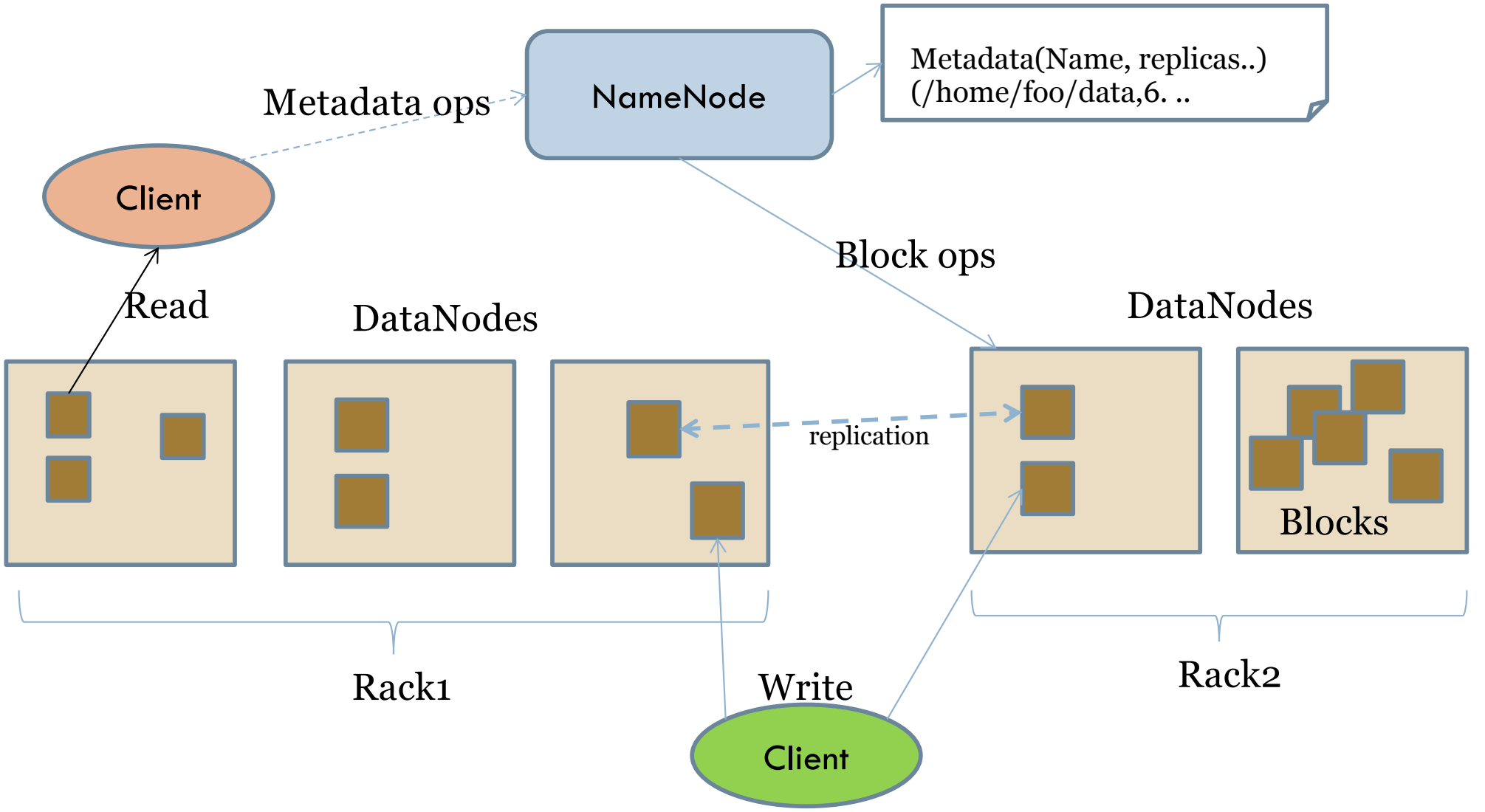
NameNodes and DataNodes

- HDFS exposes a file system interface of:
- A single NameNode
  - Maintains metadata and name space
  - Regulates access to files by clients
  - Carries out rebalancing and fault recovery
- Many DataNodes, usually in a cluster
  - Manage storage attached to the nodes that they run on
  - Serve read/write requests and perform block creation, deletion, and replication upon instruction from NameNode
  - Clients talk directly to DataNodes, once identified by NameNode

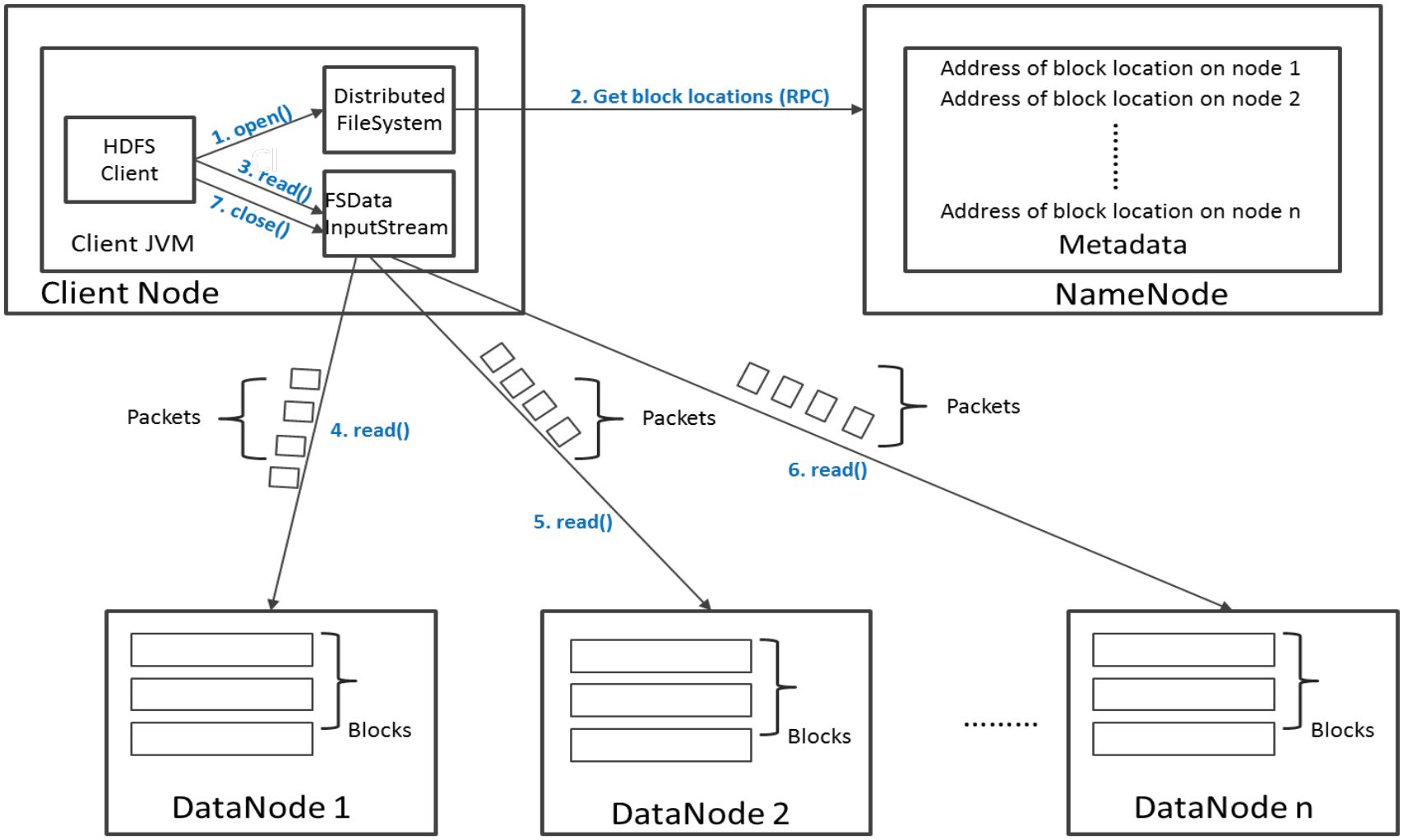
HDFS Architecture



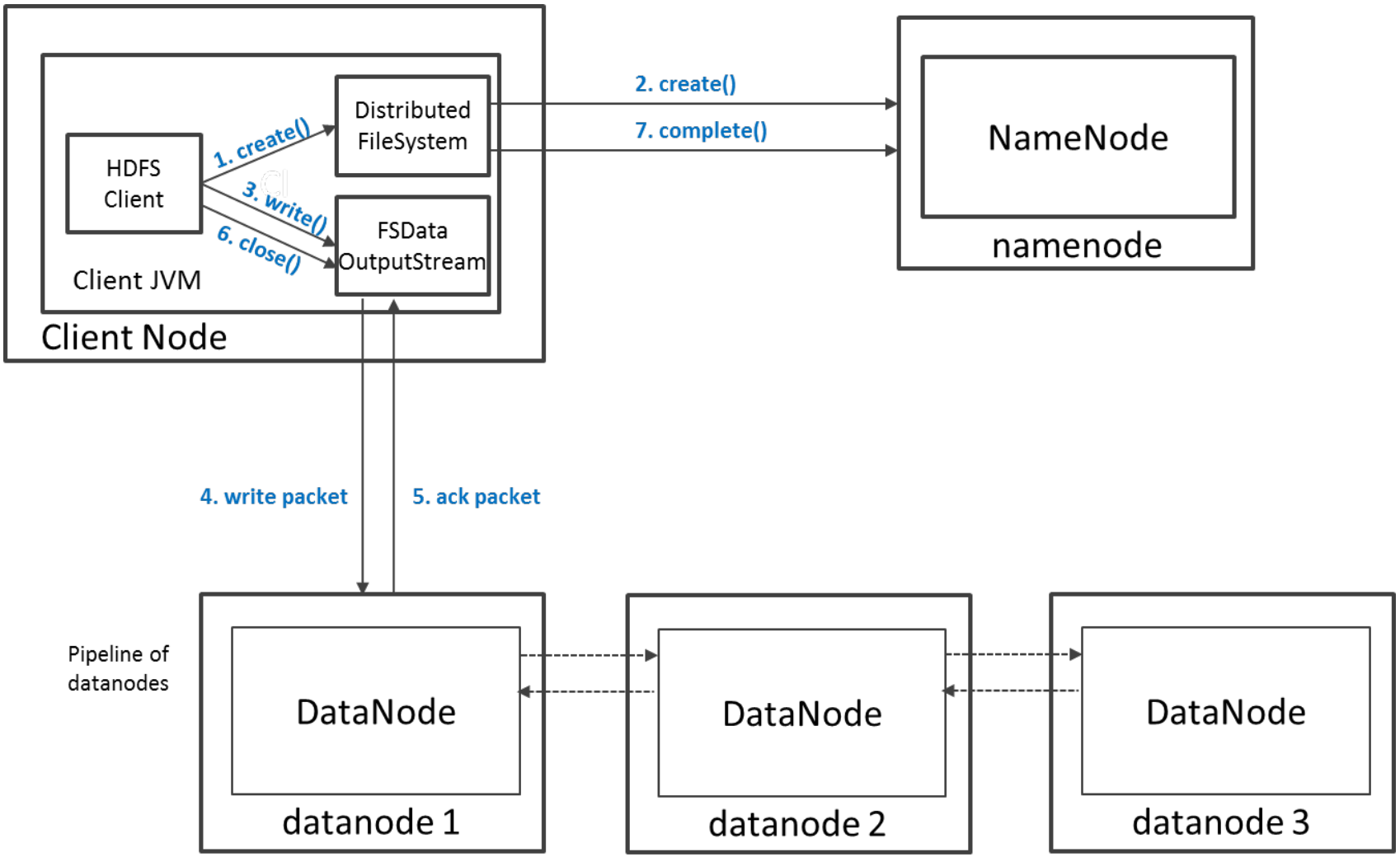
HDFS Architecture



HDFS Read Operation



HDFS Write Operation



HDFS Namespace

- Hierarchical file system with directories and files
  - Create, remove, move, rename etc.
- File system API started as UNIX compatible
  - “faithfulness to standards was sacrificed in favor of improved performance …”
- Any metadata changes to the file system recorded by the NameNode.

Reliability

- How do we store large files reliably?
  - Because we are using a large number of commodity nodes
  - Nodes may crash and fail frequently

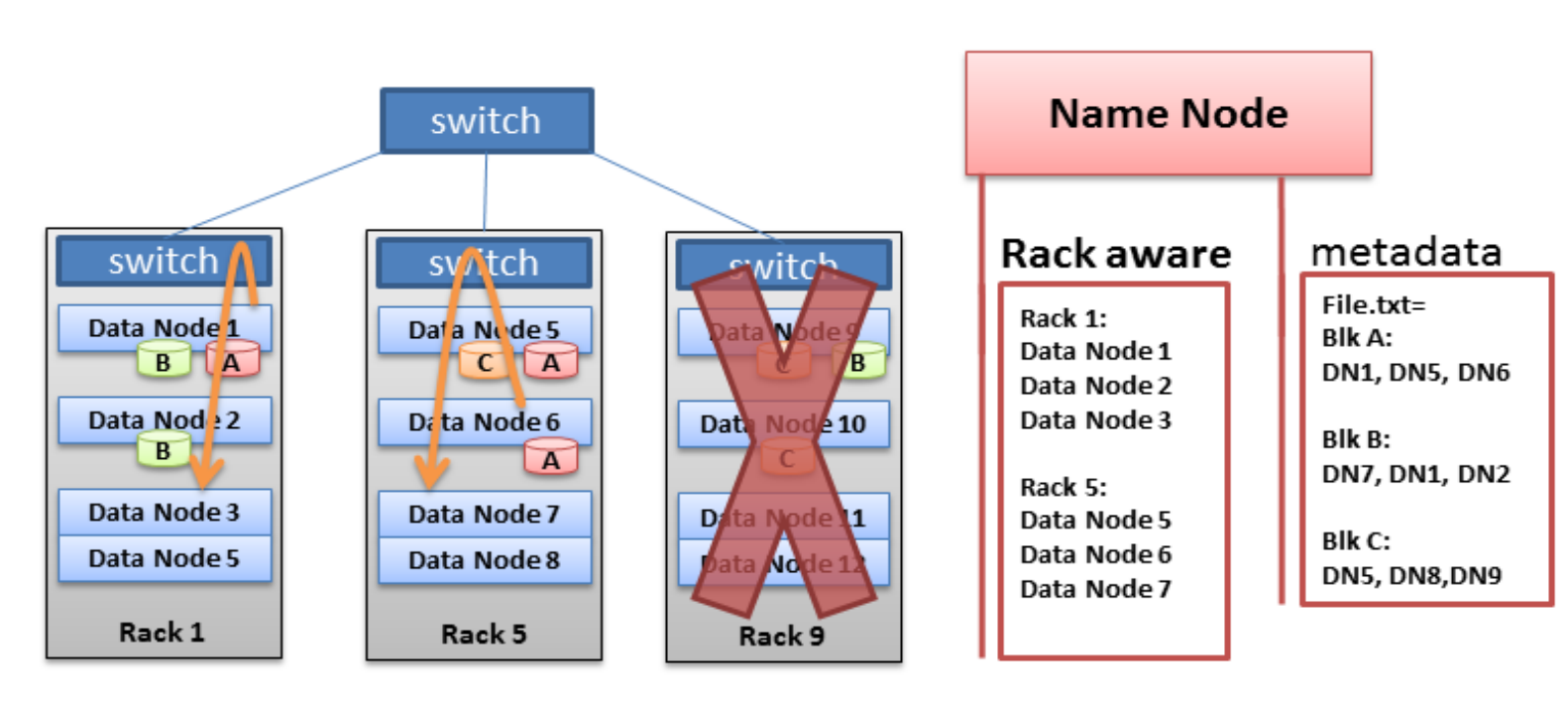
Data Replication

- HDFS is designed to store very large files across machines in a large cluster.
  - Each file is a sequence of blocks.
- Blocks are replicated for fault tolerance.
  - Block size and replicas are configurable per file.
- 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.
- Rack-aware replica placement:
  - Goal: improve reliability, availability and network bandwidth utilization
  - Research topic
- Many racks, communication between racks are through switches.
- Replicas are typically placed on unique racks
  - Simple but non-optimal
  - Writes are expensive
  - Replication factor is 3
- Replica placement strategy: 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.

Rack Awareness



Typically, large Hadoop clusters are arranged in racks and network traffic between multiple nodes within the same rack is much more desirable than network traffic across the racks. In addition, the NameNode tries to place replicas of each block on multiple racks for improved fault tolerance.

File System Metadata Reliability

- What about the metadata's reliability?

HDFS Metadata

- The HDFS namespace is stored by NameNode.
- NameNode uses a transaction log called the EditLog to record every change that occurs to the filesystem meta data.
  - For example, creating a new file.
  - Change replication factor of a file
  - EditLog is stored in the NameNode's local filesystem
  - Metadata only
- Entire filesystem namespace including mapping of blocks to files and file system properties is stored in a file FsImage. Stored in NameNode's local filesystem.

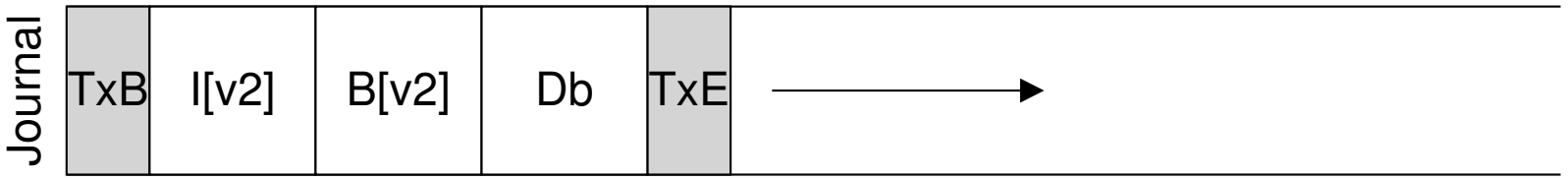
Aside: File System Journaling

- To create a new file, we need to update:
  - Directories
  - File control blocks (inodes)
  - Data blocks
  - Other metadata -- free counts

- What happens if there is a crash in the middle?

File System Journaling

- File system changes are applied in a transaction
  - E.g., write to a new data block



- Log writes in journal between transaction begin (TxB) and transaction end (TxE)
- Do we need to write the data block?
  - In the book (Chapter 42)

File System Journaling

- File system changes are applied in a transaction
  - Once these changes are written, user process can proceed
- Can then apply changes to actual file system structures later
  - On crash, can apply committed transactions
- What about those that were not completed?

HDFS Metadata Reliability

- Keeps image of entire file system namespace and file Blockmap in memory.
  - 4GB of local RAM is sufficient
- When NameNode starts up it gets the FsImage and Editlog from its local file system
  - update FsImage with EditLog information and then stores a copy of the FsImage on the filesytstem as a checkpoint.
- Periodic checkpointing is done. So that the system can recover back to the last checkpointed state in case of a crash.

Data Security in HDFS

- What might attackers want to do with HDFS data?

Data Security in HDFS

- What might attackers want to do with HDFS data?
  - Secrecy – steal secret data
  - Integrity – modify high value data

Data Security in HDFS

- How to protect data in distributed file systems
  - Secrecy – encrypt
  - Integrity – add authentication metadata
    - Message authentication code
    - Cryptographic signature
    - Checksum is not mathematically strong
  - Clients can do these things themselves

Data Security in HDFS

- Another threat to data security
  - Freshness – replay an old version of the stored data
  - Not protected on traditional HDFS systems
  - Clients need help

Data Security in HDFS

- Another threat to data security
  - Freshness – replay an old version of the stored data
  - Not protected on traditional HDFS systems
  - Clients need help

- Angel et al. Nimble: Rollback Protection for Confidential Cloud Services. OSDI 2023.

Conclusions

- We discussed the features of the Hadoop File System, a peta-scale file system to handle big-data sets.
- What discussed: Handling very large files and reliability under failures - at the design level
  - Briefly: journaling and performance/compatibility
- Missing element: Implementation
  - The Hadoop file system (internals)

Questions