HDFS Fundamentals

Hadoop Distributed File System

Professor Anis Koubaa

SE 446
Alfaisal University

Spring 2026

Outline

1. HDFS Overview
2. HDFS Architecture
3. HDFS Read/Write Operations
4. Blocks and Replication
5. File Formats
6. HDFS Commands
7. Summary

What is HDFS?

Definition

HDFS (Hadoop Distributed File System) is a distributed storage system designed to store very large files across multiple machines.

Key Features:

- **Distributed**: Files split across many machines
- **Fault-tolerant**: Data replicated for reliability
- **Scalable**: Add more machines as needed
- **Cost-effective**: Uses commodity hardware

HDFS Design Philosophy

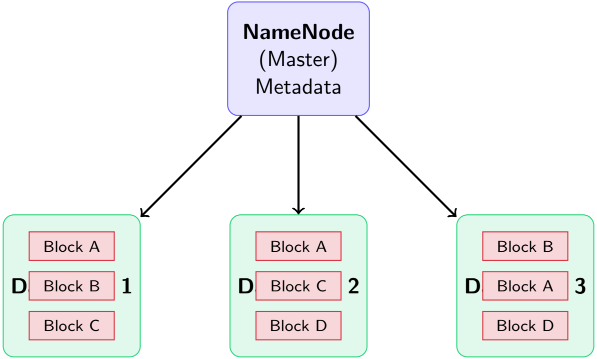
Optimized For

- Very large files (GB to PB)
- Streaming data access
- Write once, read many
- Commodity hardware

NOT Optimized For

- Low-latency access
- Many small files
- Random writes
- Interactive queries

HDFS Architecture: Master-Slave



NameNode
(Master)
Metadata

Block A
D. Block B 1
Block C

Block A
D. Block C 2
Block D

Block B
D. Block A 3
Block D

Stores metadata

Store actual data

NameNode: The Master

Responsibilities

- Manages file system **namespace** (directory tree)
- Stores **metadata**: file names, permissions, block locations
- Handles **client requests** for file operations
- Tracks **health** of DataNodes via heartbeats

Critical Component!

If NameNode fails, **the entire cluster is unavailable**.
Solutions: Secondary NameNode, HA (High Availability) mode

NameNode Variants: Understanding the Differences

Component	Purpose	When Used
Secondary NameNode	• Periodically merges fsimage + edits • Creates checkpoint metadata • NOT a backup!	Single NameNode deployments
Standby NameNode	• Hot standby for failover • Synchronized metadata • Automatic takeover on failure	High Availability (HA) mode

Production Recommendation

DataNode: The Workers

Responsibilities

- Store actual **data blocks** on local disk
- Serve **read/write requests** from clients
- Perform **block replication** as instructed by NameNode
- Send **heartbeats** to NameNode every 3 seconds
- Send **block reports** every hour

Commodity Hardware

DataNodes are designed to run on **cheap, commodity servers**.
Failures are expected and handled automatically.

Monitoring DataNodes: Heartbeats & Block Reports

Mechanism	Frequency	Purpose
Heartbeat	Every 3 sec	• Confirms DataNode is alive • Reports available storage • Receives commands from NameNode
Block Report	Every 1 hour	• Lists all blocks on this DataNode • Enables metadata verification • Triggers cluster balancing

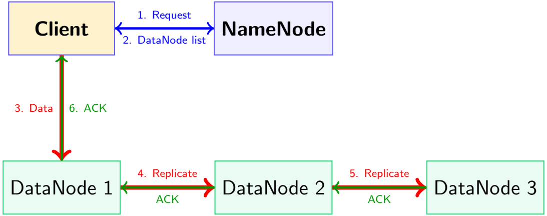
HDFS Write Path: How Data Gets Stored

1. **Client requests** to write file to NameNode
2. **NameNode checks** permissions, creates metadata
3. **NameNode selects** DataNodes for block replicas
4. **Client streams data** directly to first DataNode
5. **Replication pipeline**: DataNode-1 → DataNode-2 → DataNode-3
6. **Acknowledgment** flows back to client
7. **NameNode updates** metadata

Key Insight

Data never flows through NameNode!
Client writes directly to DataNodes for scalability.

HDFS Write Pipeline Visualization



Pipeline replication: Each DataNode forwards to next while writing locally

HDFS Read Path: How Data Gets Retrieved

1. **Client requests** file from NameNode
2. **NameNode returns** block locations (DataNode addresses)
3. **Client selects** nearest DataNode for each block
4. **Client streams** data directly from DataNodes
5. **Parallel reads** for different blocks
6. **Checksum verification** on each block chunk
7. **Automatic failover** to replica if checksum fails

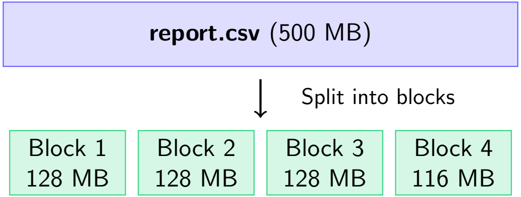
Performance Benefit

Locality awareness: Client reads from closest DataNode.
Maximizes network bandwidth utilization.

Data Blocks

What is a Block?

Files in HDFS are split into fixed-size **blocks** (default: 128 MB).



report.csv (500 MB)

Split into blocks

Block 1 128 MB

Block 2 128 MB

Block 3 128 MB

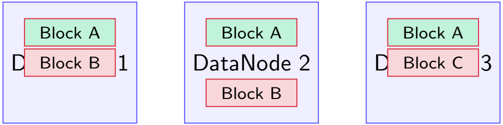
Block 4 116 MB

Why 128 MB? Minimize seek time, maximize throughput for large files.

Replication Factor

What is Replication?

Each block is copied to **multiple DataNodes** for fault tolerance.
Default replication factor: **3**



D Block A Block B 1

Block A DataNode 2 Block B

D Block A Block C 3

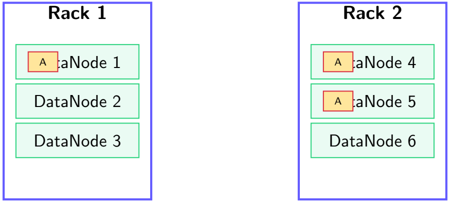
Block A is stored on 3 different nodes

Trade-off: More replicas = more fault tolerance, but 3× storage cost.

Rack Awareness: Smart Replica Placement

Why Rack Awareness?

In real data centers, servers are organized in **racks**.
Entire rack can fail (power, network switch failure).



Default Strategy (3 replicas):
Replica 1: Local rack — Replica 2: Remote rack — Replica 3: Same remote rack

Benefit

Survives entire rack failure while minimizing cross-rack traffic.

Storage Calculation

Example

File size: 500 MB

Replication factor: 3

Total storage used: $500 \times 3 = 1,500$ MB = **1.5 GB**

Formula

$$\text{Total Storage} = \text{File Size} \times \text{Replication Factor}$$

Question: If you store 10 TB of data with replication factor 3, how much disk space do you need?

Data Integrity: Checksums

How HDFS Ensures Data Correctness

HDFS computes **CRC32 checksums** for every 512-byte chunk of data.

Checksum Verification Process:

1. **On Write**: Compute checksum, store with data
2. **On Read**: Recompute checksum, compare with stored value
3. **If Mismatch**:
    - Report corruption to NameNode
    - Automatically read from another replica
    - NameNode schedules re-replication from good copy

Why It Matters

Protects against:

- Disk corruption (bit rot)
- Network transmission errors

File Formats Comparison

Format	**Type**	**Compression**	**Best For**
CSV	Row-based	None	Simple exchange
JSON	Row-based	None	APIs, configs
Parquet	**Column-based**	**Yes**	**Analytics**
Avro	Row-based	Yes	Streaming
ORC	Column-based	Yes	Hive

For Big Data Analytics

Use **Parquet** — columnar storage with excellent compression.

Why Columnar Storage?

Row-Based (CSV)

ID	Name	Age	Salary
1	Alice	30	75000
2	Bob	25	65000

Column-Based (Parquet)

ID	Name	Age	Salary
1	Alice	30	75000
2	Bob	25	65000

Query: SELECT AVG(Salary)
Only reads Salary column!

Parquet Benefits

1. **Read only needed columns**
   Faster queries, less I/O
2. **Better compression**
   Similar values in a column compress well (10x smaller)
3. **Schema embedded**
   No need for external schema files
4. **Predicate pushdown**
   Filter data at storage level, before loading

Common HDFS Commands

```
# List files in a directory
hdfs dfs -ls /user/data/

# Create a directory
hdfs dfs -mkdir /user/mydata/

# Upload a file from local to HDFS
hdfs dfs -put local_file.csv /user/mydata/

# Download a file from HDFS to local
hdfs dfs -get /user/mydata/file.csv ./local/

# View file contents
hdfs dfs -cat /user/mydata/file.txt

# Delete a file
hdfs dfs -rm /user/mydata/old_file.csv

# Check disk usage
```

Summary: Key Takeaways

1. **HDFS** = Distributed file system for Big Data
2. **NameNode** (master) stores metadata; **DataNode** (workers) store data
    - Secondary NameNode: Checkpoints — Standby: HA failover
    - Heartbeats (3s) + Block Reports (1h) for monitoring
3. **Write/Read paths**: Client streams directly to/from DataNodes
4. Files split into **blocks** (128 MB); **Replication factor 3**
    - Rack awareness for fault tolerance
    - Checksums ensure data integrity
5. Use **Parquet** for Big Data analytics (columnar, compressed)

Next Week

MapReduce: Distributed data processing paradigm

ExamGPT Quiz Topics

Be prepared to answer questions about:

- Role of NameNode vs DataNode
- Secondary vs Standby NameNode
- Heartbeats vs Block Reports
- HDFS Read/Write paths
- Rack awareness and replica placement
- Storage calculations with replication
- Data integrity (checksums)
- Block size and why it matters
- CSV vs Parquet comparison
- Basic HDFS commands

Quiz in 15 minutes!

Open ExamGPT and complete the Week 2B quiz.

Questions?

Prof. Anis Koubaa
akoubaa@alfaisal.edu