Al-Mustaqbal University
College of Sciences
Intelligent Medical System Department

جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قسم الانظمة الطبية الذكية

Lecture (3): Hadoop Distributed File System (HDFS)

Subject: Big Data Analysis in Healthcare
Level: Fourth
Lecturer: Asst. Lecturer Qusai AL-Durrah
Duration: Two hours

1. Introduction

The Hadoop Distributed File System (HDFS) is the foundational storage layer of Apache Hadoop.
It is designed to hold enormous datasets—often petabytes—in a reliable, cost-effective, and highly scalable manner.
Unlike traditional enterprise storage, HDFS runs on clusters of commodity hardware and provides *fault tolerance*, *high throughput*, and *data locality* for analytical workloads.
In healthcare, HDFS underpins large-scale applications such as:

- Archiving and retrieving terabytes of medical imaging (MRI, CT) data.
- Storing high-velocity sensor streams from intensive-care units.
- Managing genome-sequencing files for personalized medicine.

This lecture explores HDFS architecture, operations, and relevance to healthcare big-data environments.

2. Learning Outcomes

By the end of this lecture, students will be able to:

1. **Explain** the design goals and key characteristics of HDFS.
2. **Describe** HDFS architecture, including blocks, NameNode, and DataNode roles.
3. **Illustrate** file read/write data flows and fault-tolerance mechanisms.
4. **Use** the command-line interface for basic HDFS operations.

5. **Discuss** advanced features such as federation and high availability.
6. **Evaluate** HDFS applications in healthcare analytics.

---

3. Design Principles of HDFS

HDFS is modeled after Google's GFS and optimized for large, streaming data access.
Its fundamental principles are:

- **Streaming Data Access:** Optimized for high-throughput, large-file reads rather than low-latency random access.
- **Write Once, Read Many:** Files are typically written once and read repeatedly, simplifying consistency.
- **Commodity Hardware:** Runs on inexpensive servers; the system assumes failures are common and must recover automatically.
- **Data Locality:** Computation is scheduled near the data to minimize network traffic.

4. Core Concepts

4.1 Blocks

- HDFS stores files as *blocks* (default size 64 MB or 128 MB).
- Each block is replicated (default factor = 3) across different nodes for durability.
- Large block size reduces metadata overhead and improves sequential I/O.

4.2 NameNode and DataNodes

- **NameNode:**
  - Manages the filesystem namespace and metadata (file hierarchy, block locations).
  - Holds all metadata in memory for fast access.
- **DataNodes:**
  - Store actual data blocks.
  - Periodically send *heartbeats* and *block reports* to the NameNode for health monitoring.

4.3 HDFS Federation and High Availability

- **Federation:** Allows multiple independent NameNodes to scale namespace capacity.
- **High Availability (HA):** Provides standby NameNodes for automatic failover, eliminating the single-point-of-failure problem.

5. HDFS Architecture

5.1 Master–Servant Model

- One (or more, in HA) **NameNode** acts as master.
- Many **DataNodes** act as servants, serving read/write requests and block creation or deletion.

5.2 Metadata Management

- Metadata includes file permissions, directory structure, and mapping of files to blocks.
- Stored in RAM for speed; persistent copies kept in the *fsimage* and *edit log* on disk.

5.3 Communication

- All client–DataNode and client–NameNode interactions use TCP/IP.
- Clients obtain block location information from the NameNode, then interact directly with DataNodes for I/O.

6. Data Flow in HDFS

6.1 Anatomy of a File Read

1. Client queries NameNode for block locations.
2. NameNode returns a list of DataNodes hosting each block replica.
3. Client reads blocks directly from the nearest DataNodes, streaming sequentially.

6.2 Anatomy of a File Write

1. Client requests file creation from NameNode.
2. NameNode allocates a block and selects a pipeline of DataNodes for replication.

3. Client writes to the first DataNode, which forwards data to the next nodes in the pipeline.
4. Upon successful replication, the client is notified of completion.

6.3 Coherency Model

- HDFS provides *write-once* semantics: once a file is closed, it is immutable.
- Appends are supported but less common in analytic workloads.

7. Command-Line Interface (CLI)

HDFS offers shell commands similar to traditional UNIX utilities:

- hdfs dfs -ls / – List directory contents.
- hdfs dfs -put localfile /path – Upload a file.
- hdfs dfs -get /path localfile – bring a file for processing.
- hdfs dfs -rm /path – Delete a file or directory.

These operations enable administrators and analysts to manage data without complex programming.

Code example:

```
# 1. List contents of the root directory in HDFS

hdfs dfs -ls /

# 2. Upload a local file (e.g., data.txt) from your local machine to HDFS under /user directory
```

```
hdfs dfs -put data.txt /user/

# 3. Verify that the file was uploaded by listing the /user directory

hdfs dfs -ls /user/

# 4. Download the same file from your local system back to HDFS

hdfs dfs -get /user/data.txt ./downloaded_data.txt

# 5. Remove the file from HDFS

hdfs dfs -rm /user/data.txt

# 6. Confirm deletion

hdfs dfs -ls /user/
```

8. Data Ingestion and Balancing

- **Flume and Sqoop:** Tools for ingesting streaming data and relational database exports into HDFS.
- **DistCp:** Efficient parallel copy for large dataset migration.
- **Balancer:** Redistributes data evenly across nodes to prevent storage hot spots.

9. Fault Tolerance and Reliability

- **Replication:** Default triple replication ensures data remains available even if nodes fail.
- **Heartbeat Monitoring:** DataNodes send heartbeats; if a heartbeat is missed, blocks are re-replicated elsewhere.

- **Rack Awareness:** Replicas are spread across racks to survive rack-level failures.

10. HDFS in Healthcare Analytics

HDFS is particularly suited to healthcare scenarios requiring secure, high-volume storage:

- **Medical Imaging Archives:** Storing and parallel processing of DICOM files for AI-driven diagnostics.
- **Genomic Data Pipelines:** Petabyte-scale sequence alignment and variant calling.
- **Real-Time Monitoring:** Retention of continuous sensor feeds for retrospective analysis.
- **Research Collaboration:** Enables sharing of large public health datasets across institutions.

10. Conclusion

HDFS provides the reliable, scalable storage essential for Big Data analytics in healthcare. Its master–servant architecture, block-based design, and built-in fault tolerance enable medical institutions to store and analyze multi-petabyte datasets efficiently and securely.

References

- Tom White, *Hadoop: The Definitive Guide*, 3rd Edition, O'Reilly Media, 2012 – **Chapter 3: The Hadoop Distributed Filesystem**.