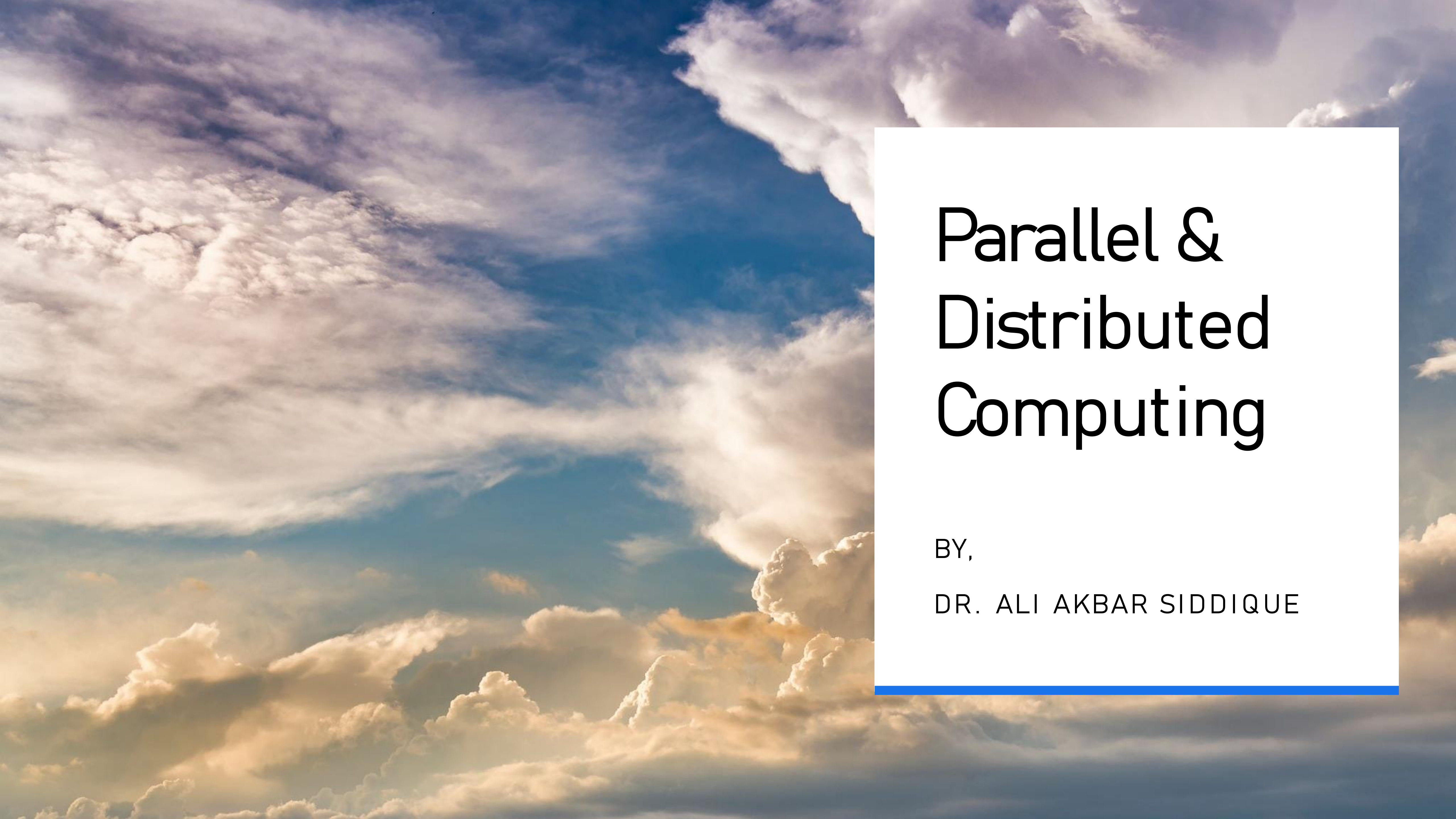


Parallel & Distributed Computing

BY,

DR. ALI AKBAR SIDDIQUE

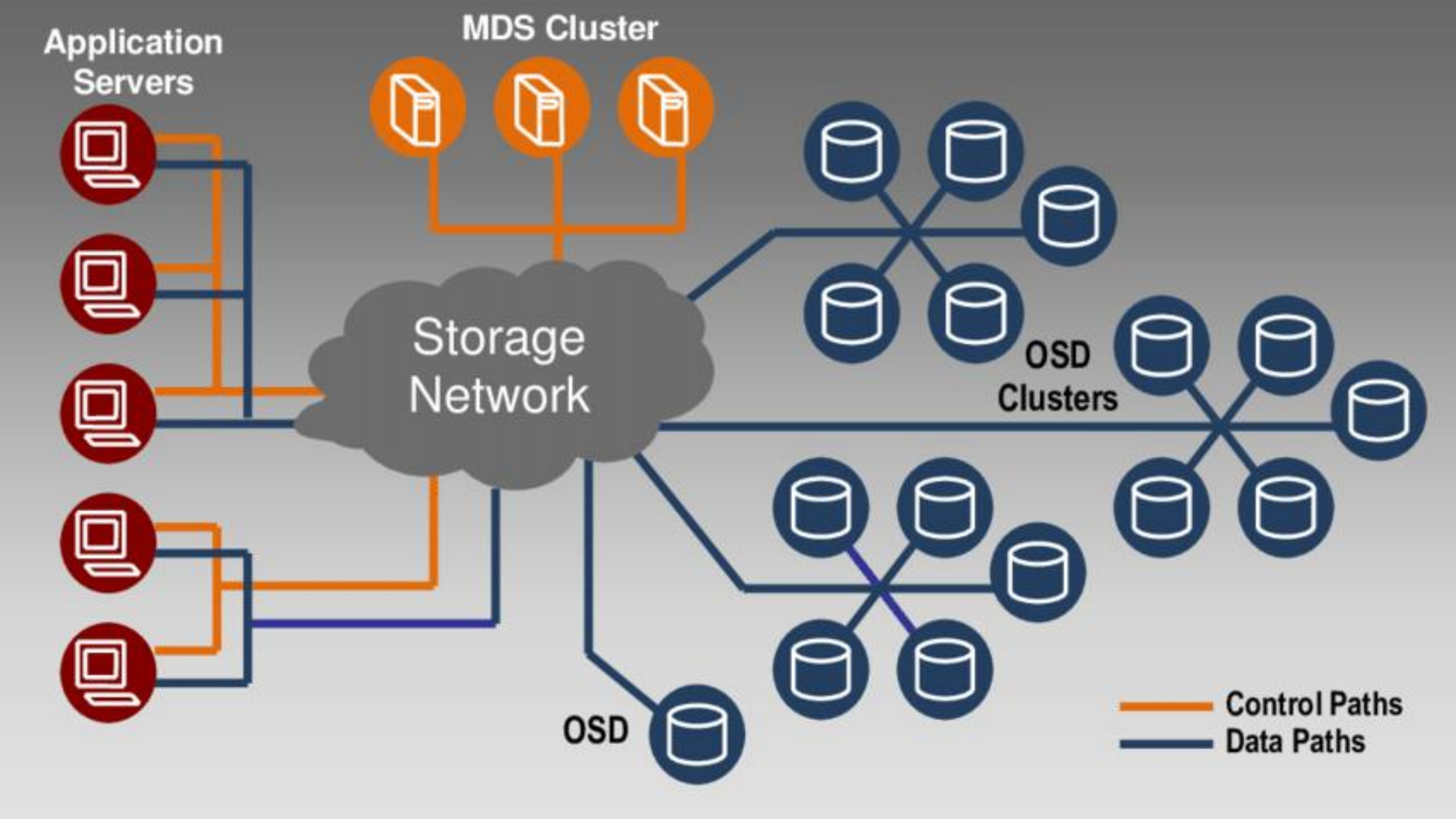
Introduction to Distributed Storage

Challenge	Why Distributed Storage Helps
Scalability	Easily adds more storage nodes as data grows.
Fault Tolerance	Redundancy ensures system reliability even if some nodes fail.
High Availability	Data is accessible even during partial system failures.
Performance	Parallel access and storage improve I/O throughput.

- Distributed storage systems store, manage, and access data across multiple networked nodes or machines, making them fundamental to modern large-scale computing environments.

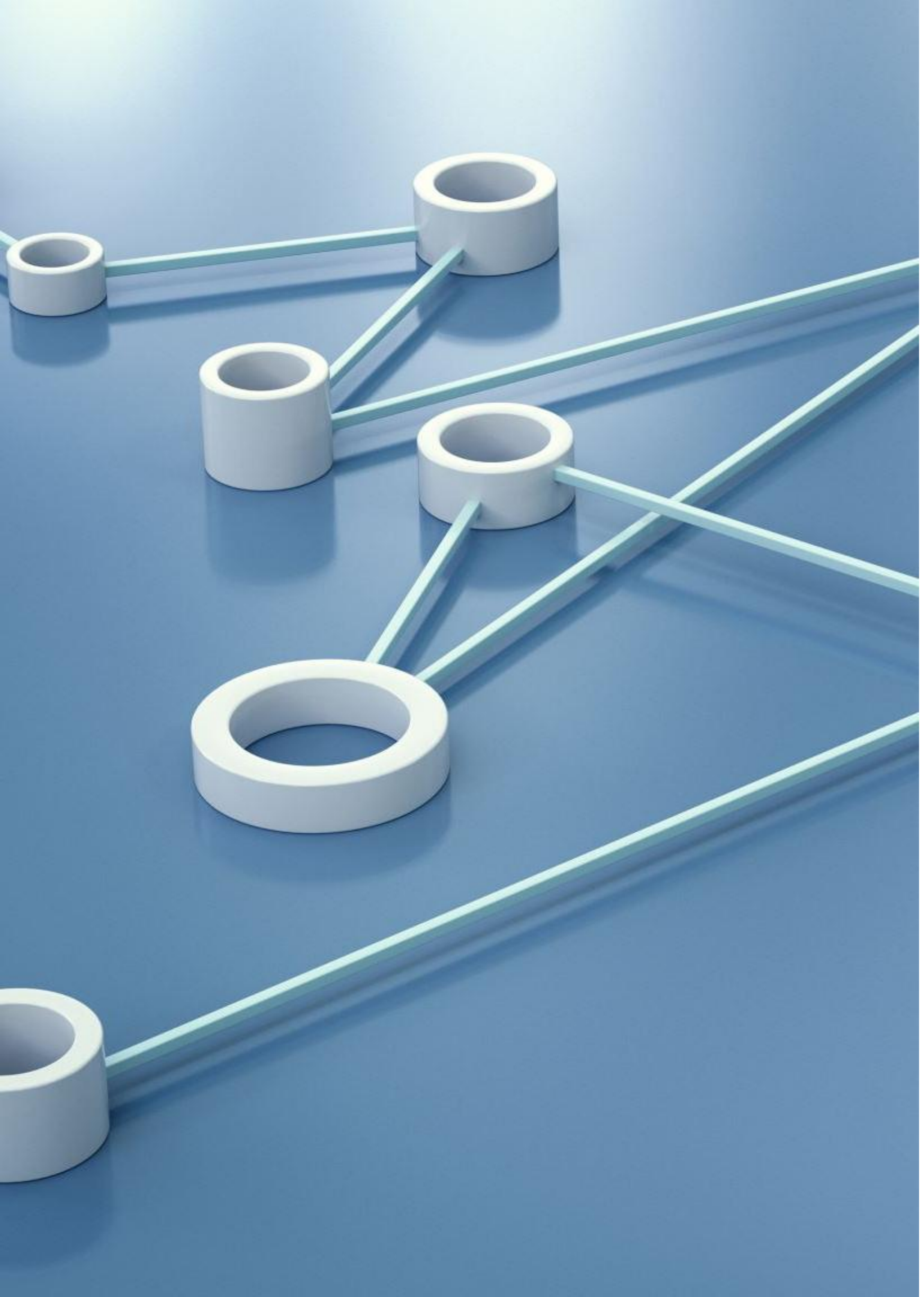
Example Use Cases

- **Big Data Processing**: Hadoop, Spark
- **Cloud Services**: Google Cloud Storage, AWS S3
- **Scientific Computing**: CERN, NASA data grids


Application Servers
MDS Cluster
Storage Network
OSD Clusters
OSD
Control Paths
Data Paths

Distributed File System (DFS)

- A **Distributed File System (DFS)** is a file system that **allows access to files stored across multiple machines** as if they were on a single local disk.



Core Functions of DFS

- **Storage Abstraction**: Hides the complexity of storage spread across many machines.
- **Metadata Management**: Keeps track of file names, directories, permissions, and locations.
- **Fault Tolerance**: Replicates data to prevent data loss in case of node failure.
- **Scalability**: Easily grows by adding more nodes without service disruption.

Design Principles

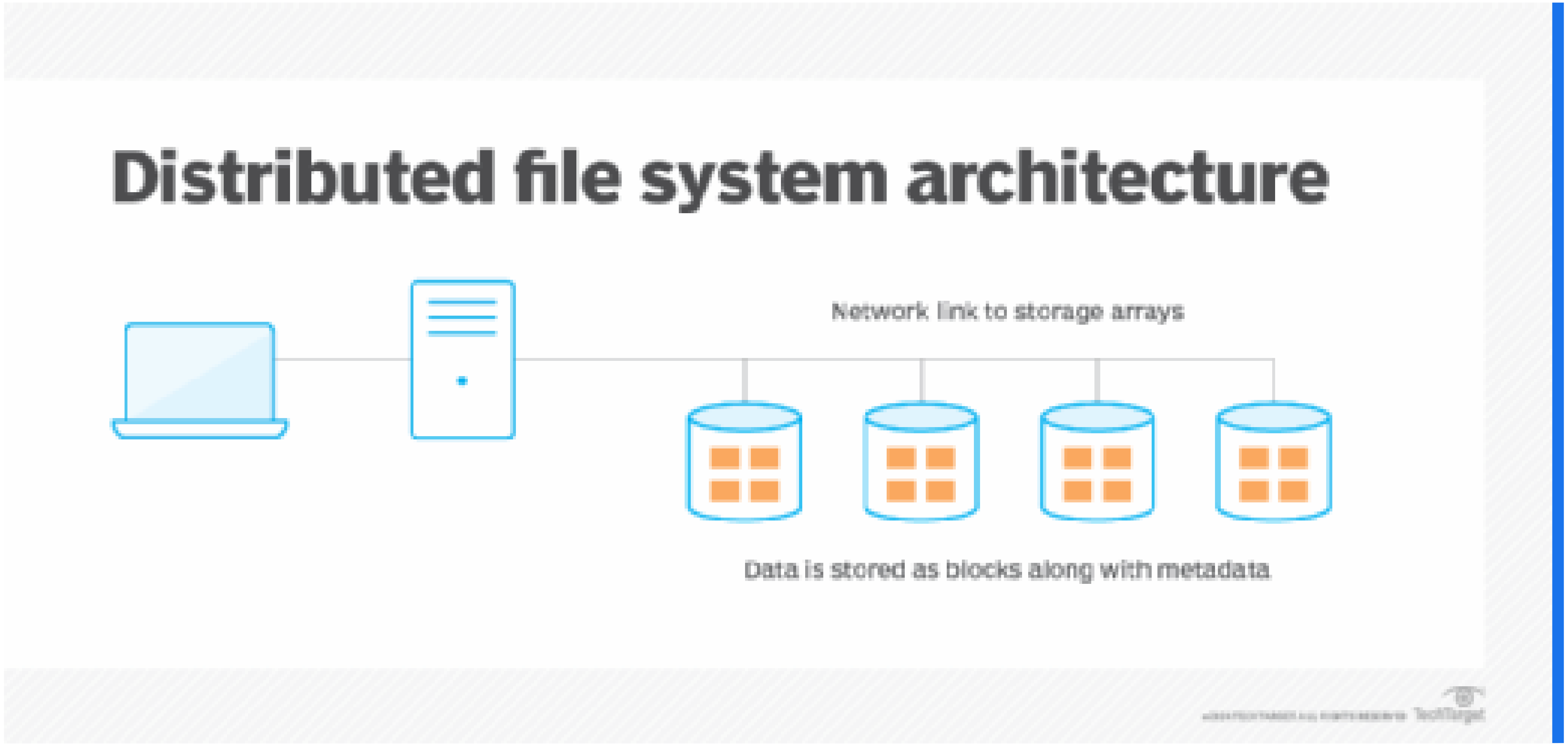
Principle	Explanation
Transparency	Users don't need to know where files reside.
Scalability	Handles increasing numbers of users and large datasets efficiently.
Reliability	Continues operation in the event of hardware or network failure.
Consistency	Ensures file contents remain synchronized across replicas or versions.
Concurrency Control	Allows multiple clients to access and modify files without conflict.

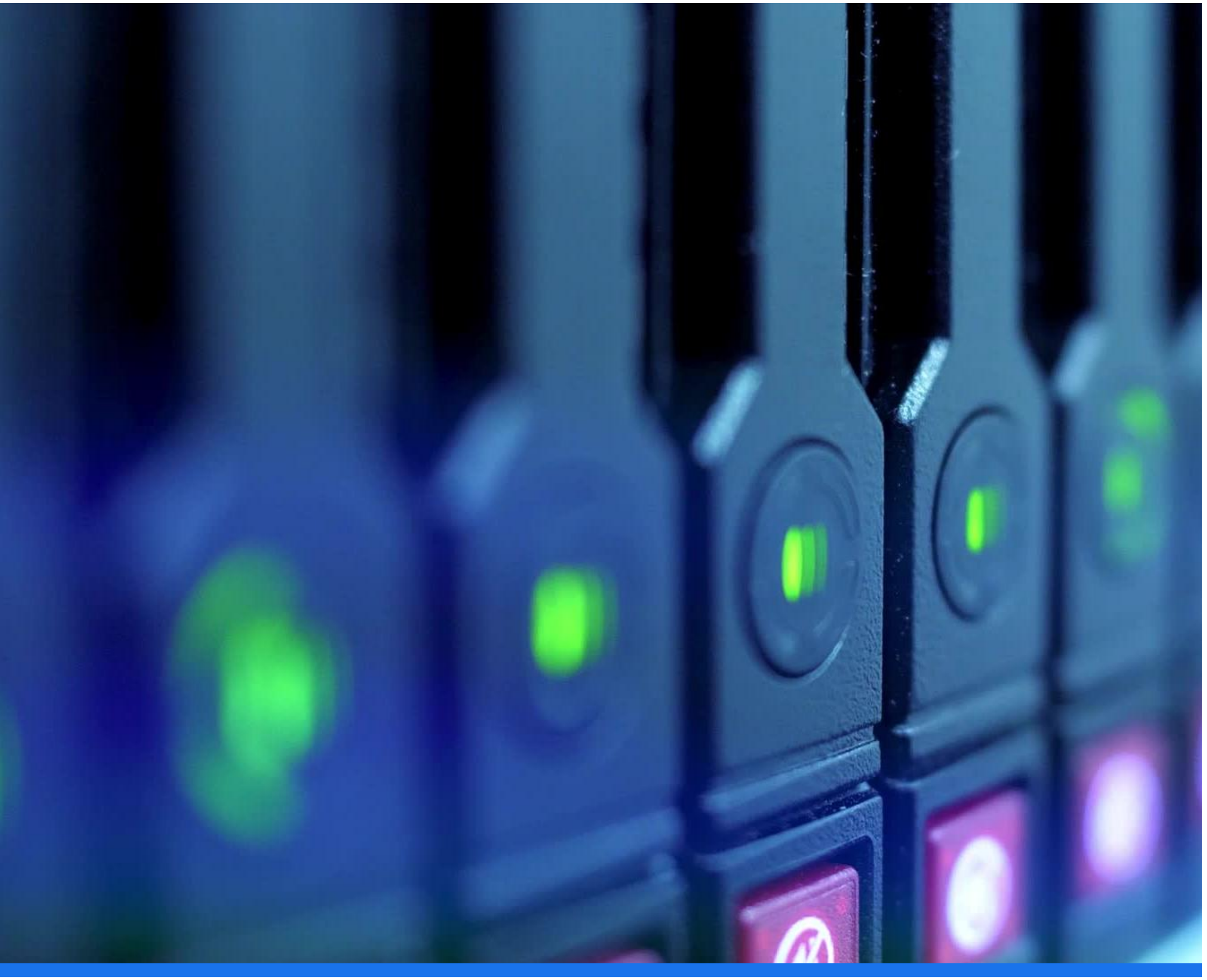
Example Features

- **Replication**: Files are stored in multiple locations for redundancy.
- **Access Control**: Permissions and user authentication ensure data security.
- **Client-Server Architecture**: Clients request files via a DFS protocol from one or more servers.

Real-World Use Cases

- **Big Data Analytics**: Hadoop uses HDFS to manage massive datasets.
- **Cloud Storage Systems**: Google File System, Amazon S3 (object storage with DFS characteristics).
- **Enterprise Systems**: Shared network drives across distributed teams.


Distributed file system architecture
Network link to storage arrays
Data is stored as blocks along with metadata
TechTarget



Hadoop Distributed File System (HDFS)

The **Hadoop Distributed File System (HDFS)** is the primary storage system used by the **Apache Hadoop** framework. It is optimized for:

- Storing large datasets (TBs to PBs)
- High-throughput data access
- Write-once, read-many workloads

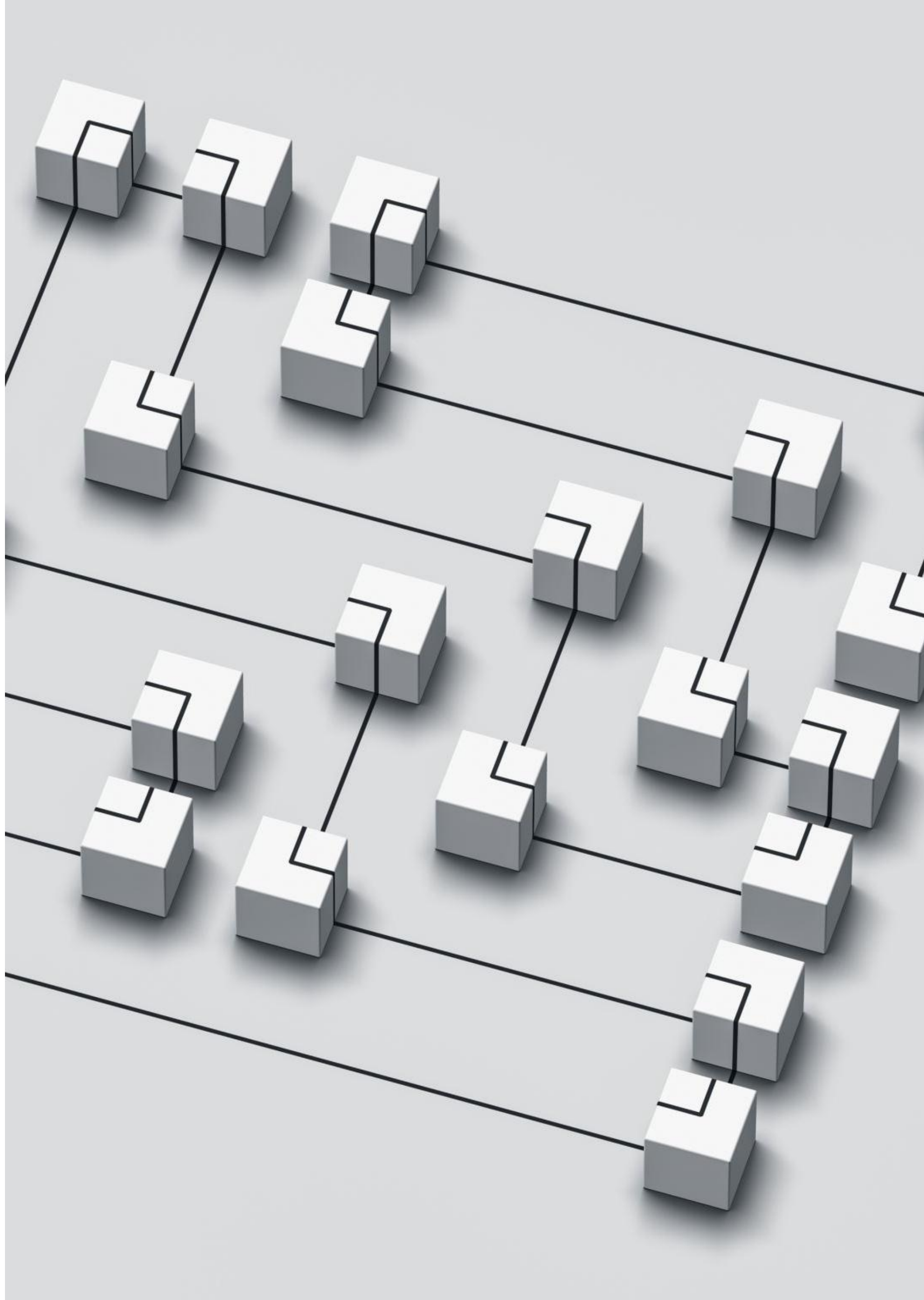
HDFS Architecture

Component	Description
NameNode	The master server that manages filesystem namespace and metadata (e.g., file-to-block mapping, permissions).
DataNodes	The worker nodes that store the actual data blocks. They send heartbeats and block reports to the NameNode.

➡️ Files are divided into large **blocks (default 128MB)** and distributed across multiple DataNodes.

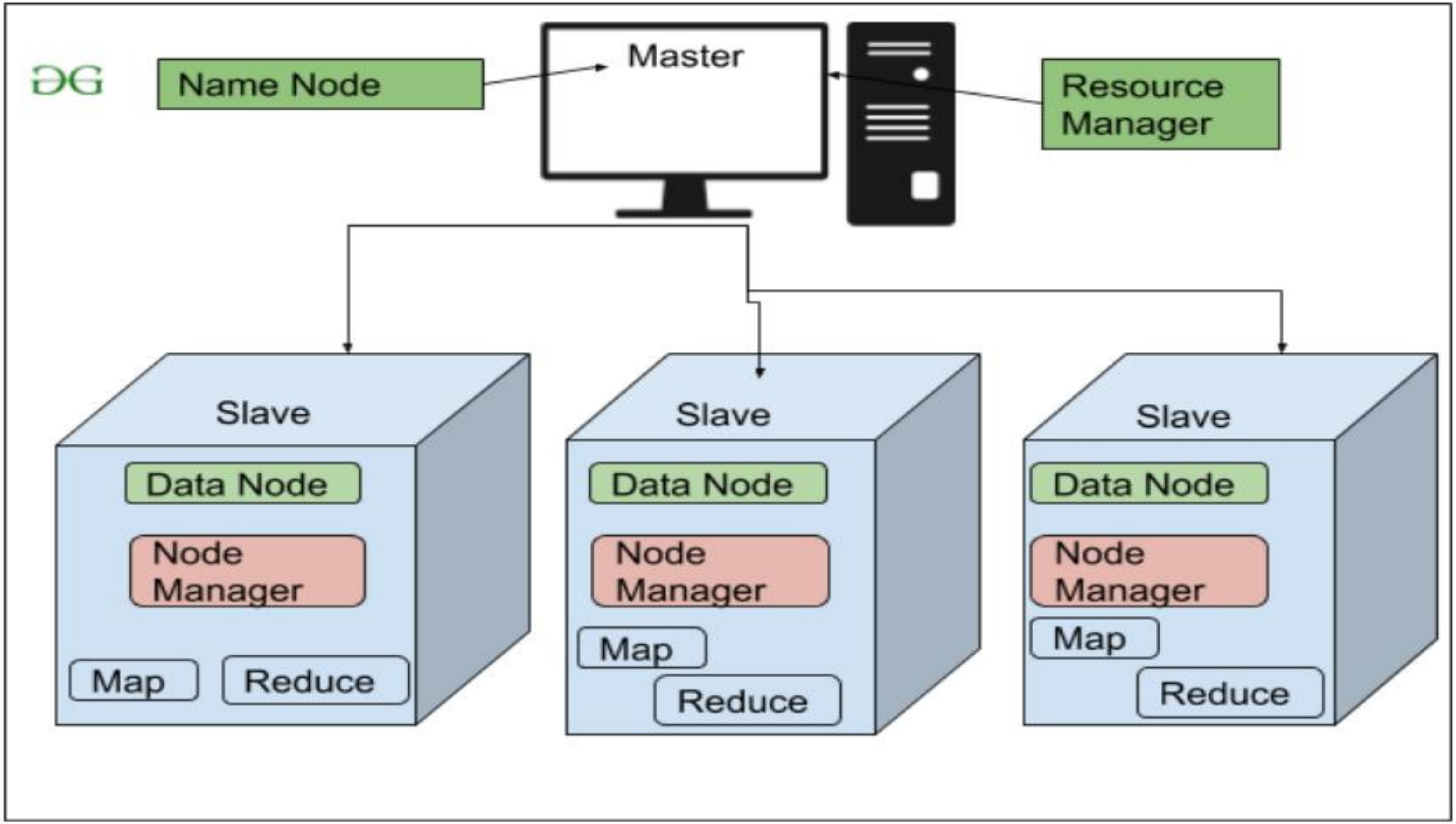
Data Flow in HDFS

- **Write Operation**:
  - Client contacts the NameNode to get block placement.
  - NameNode provides the DataNode sequence.
  - Client writes data in **pipeline fashion** to the DataNodes.
- **Read Operation**:
  - Client queries NameNode for block locations.
  - Client directly reads blocks from DataNodes.



Use Cases of HDFS

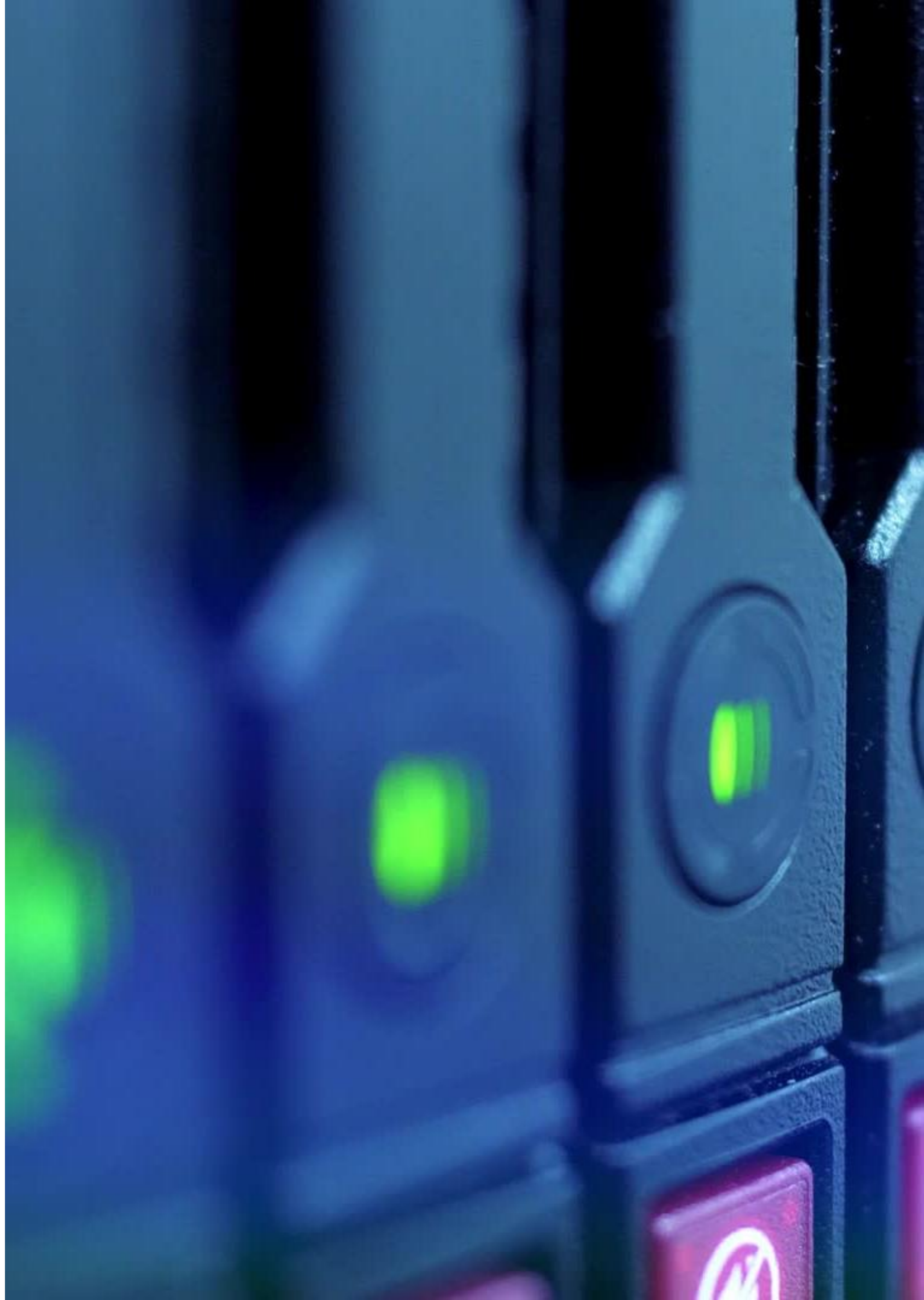
- Data Warehousing
- Web Log Processing
- Machine Learning Pipelines
- Social Media Analytics


Name Node
Master
Resource Manager
Slave
Data Node
Node Manager
Map
Reduce
Slave
Data Node
Node Manager
Map
Reduce
Slave
Data Node
Node Manager
Map
Reduce

Google File System (GFS)

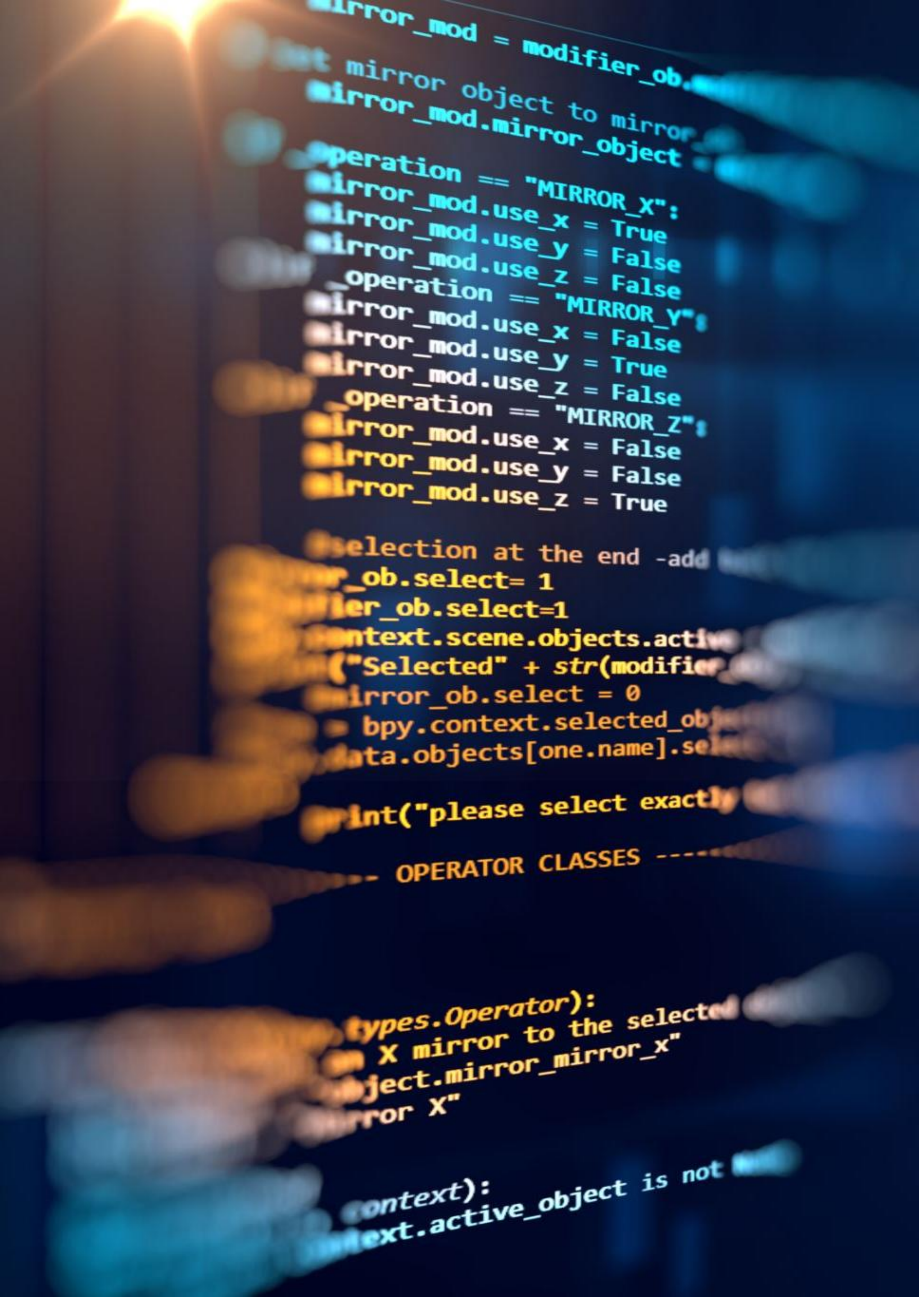
The **Google File System (GFS)** is a **proprietary distributed file system** developed by Google to meet the demands of:

- Large-scale data processing
- High fault tolerance
- Efficient handling of **very large files** (multi-GB+)



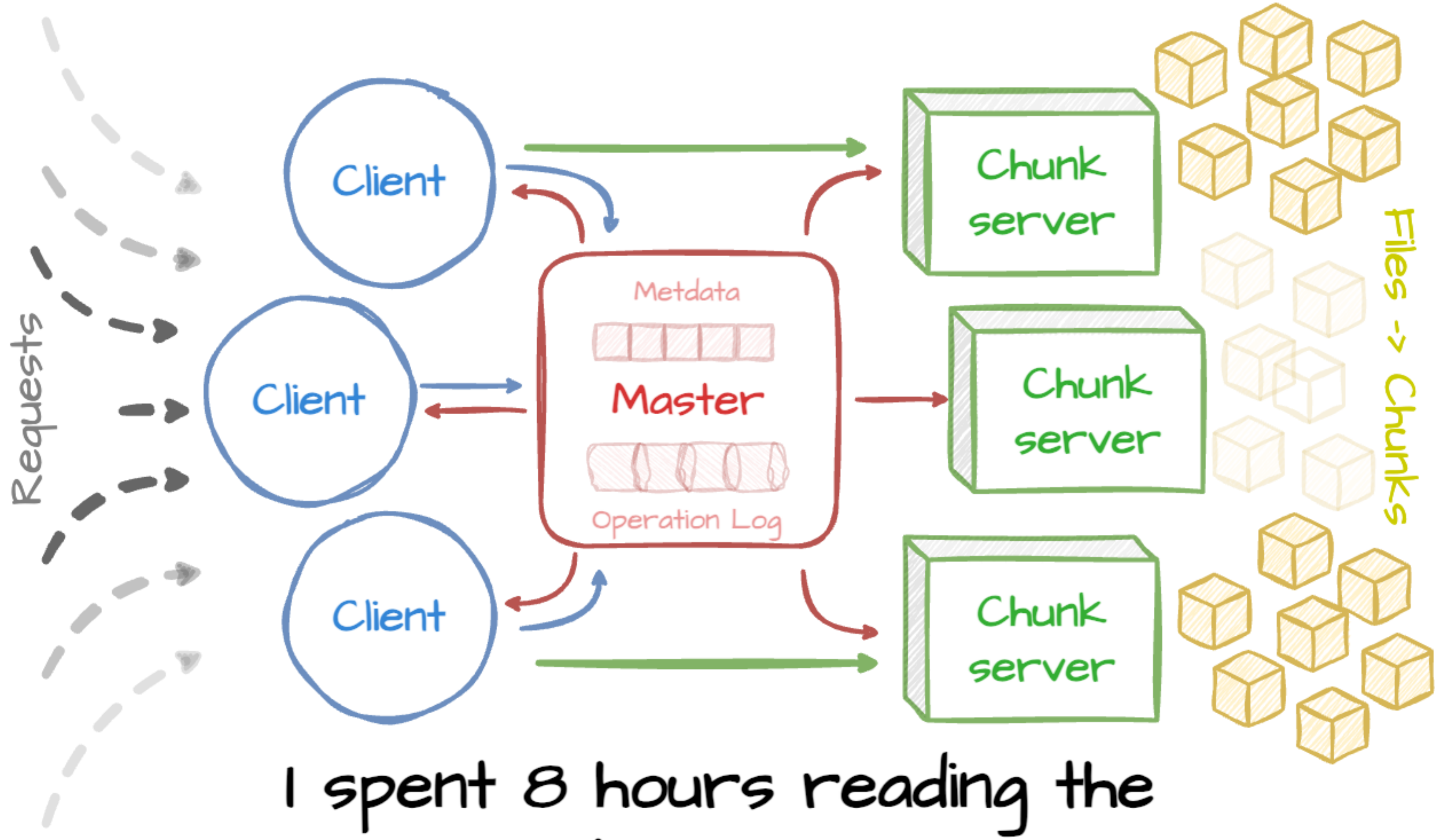
GFS Architecture

Component	Description
Master Server	Manages metadata (namespace, access control, file-to-chunk mapping).
Chunkservers	Store file data in **fixed-size chunks** (64MB each), identified by chunk handles.
Clients	Request file operations; interact with Master for metadata, and Chunkservers for actual data.



GFS Data Operations

- **File Creation**:
  - Client sends request to Master.
  - Master allocates chunk handles and returns Chunkserver locations.
- **Reading**:
  - Client fetches metadata from Master.
  - Reads data directly from Chunkservers.
- **Appending**:
  - Supports **record appends**, optimized for concurrent writers.
  - Ensures at-least-once semantics with optional application-level deduplication.



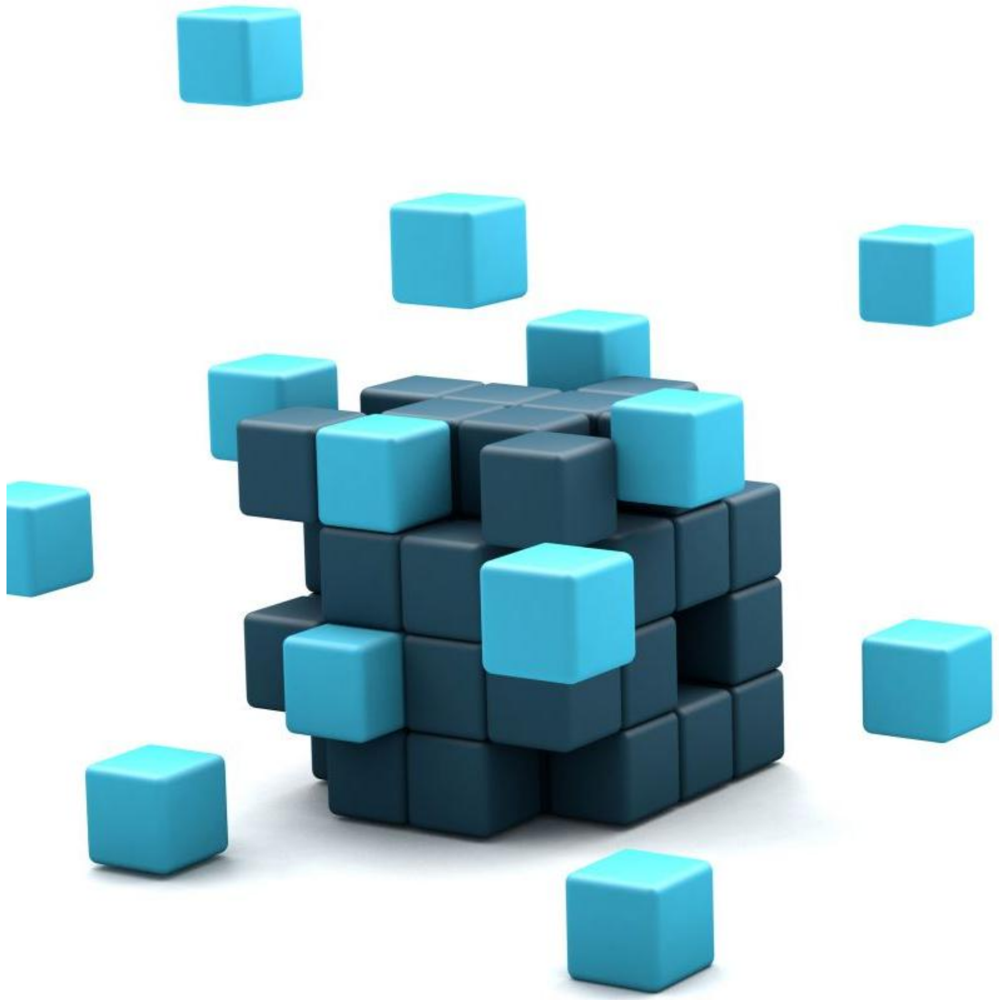
I spent 8 hours reading the paper about the Google File System

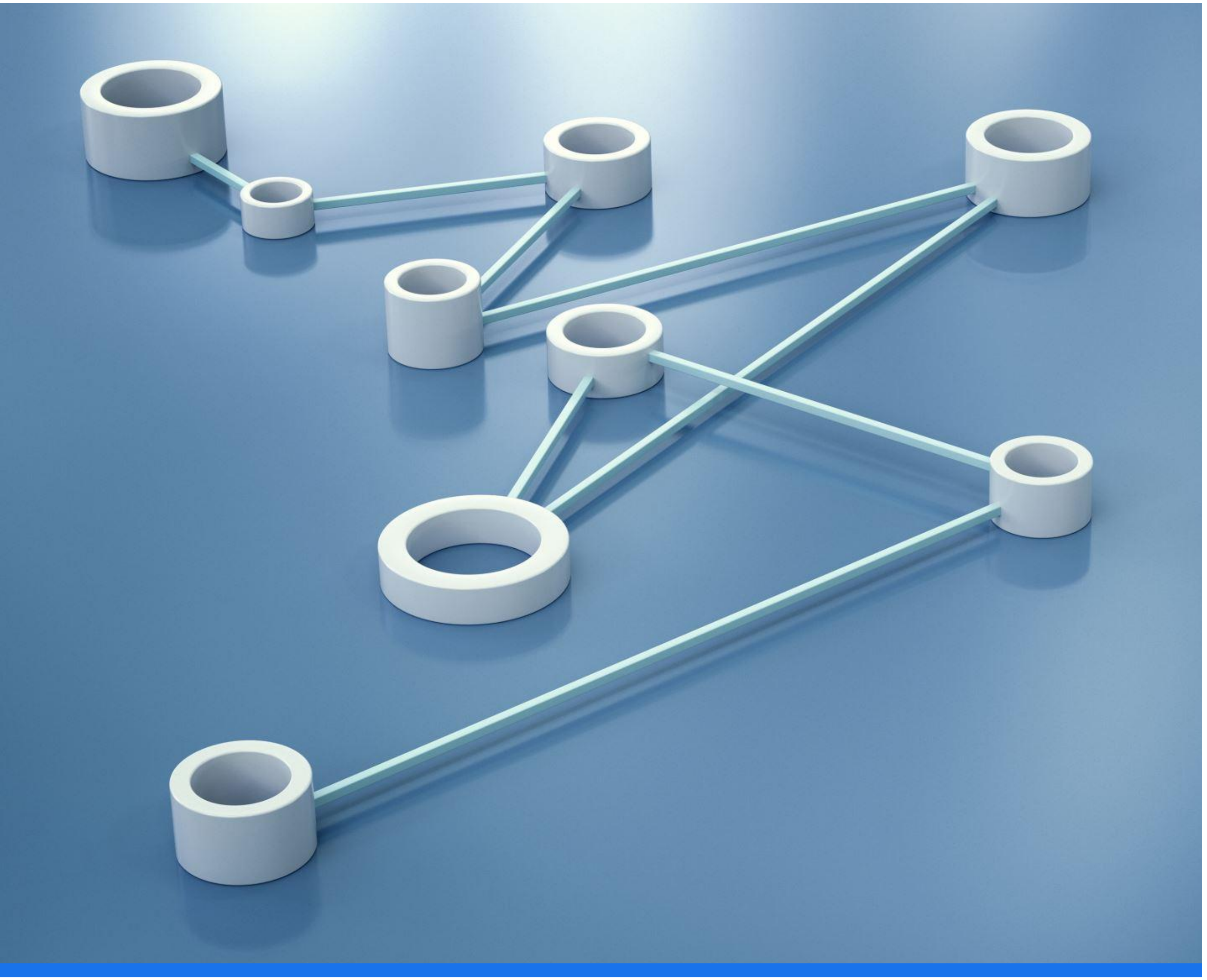
HDFS vs GFS

Feature	**HDFS (Hadoop DFS)**	**GFS (Google FS)**
Developed By	Apache Hadoop Community	Google
Purpose	Open-source DFS for big data processing	Internal DFS for Google services
Architecture	NameNode (Master), DataNodes (Workers)	Master Server, Chunkservers
Block/Chunk Size	128 MB (configurable)	64 MB (fixed)
File Access Pattern	Write-once, read-many	Append-only writes, read-many
Metadata Storage	Stored in memory on NameNode, also persisted	Stored in memory and periodically logged to disk
Replication	Default 3 replicas	Default 3 replicas
Fault Tolerance	Automatic failover with HA NameNode	Chunk versioning and replica checks
Consistency Model	Strong consistency (atomic operations)	Eventual consistency for appends
Scalability	Scales well with YARN and Hadoop ecosystem	Scales horizontally within Google's infrastructure
Open Source	✅ Yes	❌ No (proprietary)

Key Design Differences

- **Access Model**:
  - HDFS is optimized for batch processing and **MapReduce jobs**.
  - GFS is designed for **append-heavy workloads** with many concurrent clients.
- **Chunk vs. Block Management**:
  - HDFS uses larger default block sizes to optimize I/O throughput.
  - GFS chunks are slightly smaller but use versioning to manage consistency.
- **Metadata Handling**:
  - Both store metadata in memory, but HDFS has **HA (High Availability) configurations**, whereas GFS relies on periodic checkpoints and logs.





Introduction to Distributed Databases

- A Distributed Database is a collection of multiple, logically interrelated databases distributed over a network. They are managed by a Distributed Database Management System (DDBMS).

Types of Distributed Databases

Type	Description
Homogeneous	All sites use the same DBMS and schema. Easier to manage.
Heterogeneous	Sites may run different DBMS and schemas. More flexible but complex.

Centralized vs Distributed DBMS

Feature	Centralized DBMS	Distributed DBMS
Data Location	Single site	Multiple sites
Availability	Lower	Higher
Fault Tolerance	Minimal	Built-in via redundancy
Scalability	Limited	High