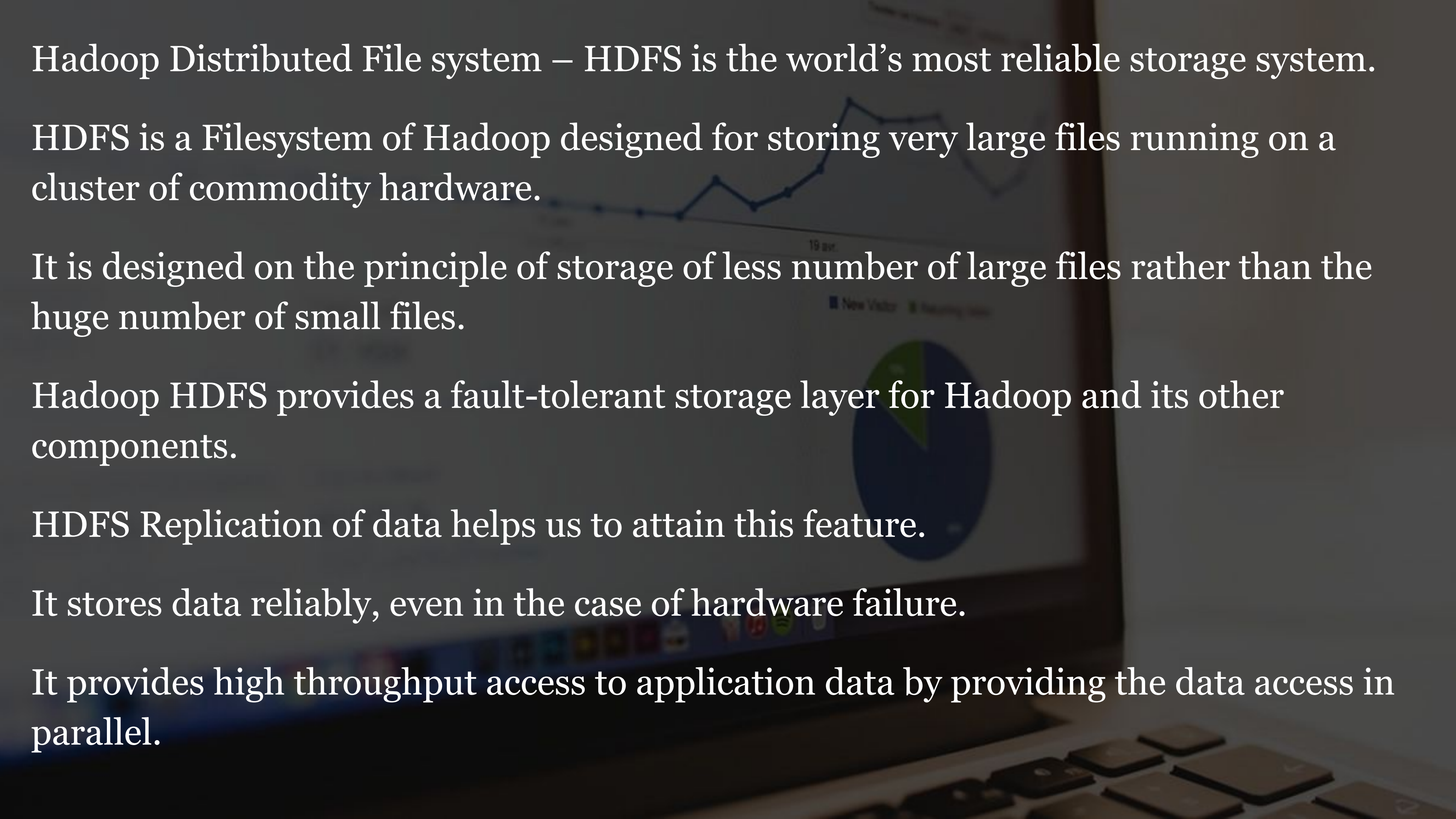
What is HDFS ?

Hadoop Distributed File System



Hadoop Distributed File system – HDFS is the world's most reliable storage system.

HDFS is a Filesystem of Hadoop designed for storing very large files running on a cluster of commodity hardware.

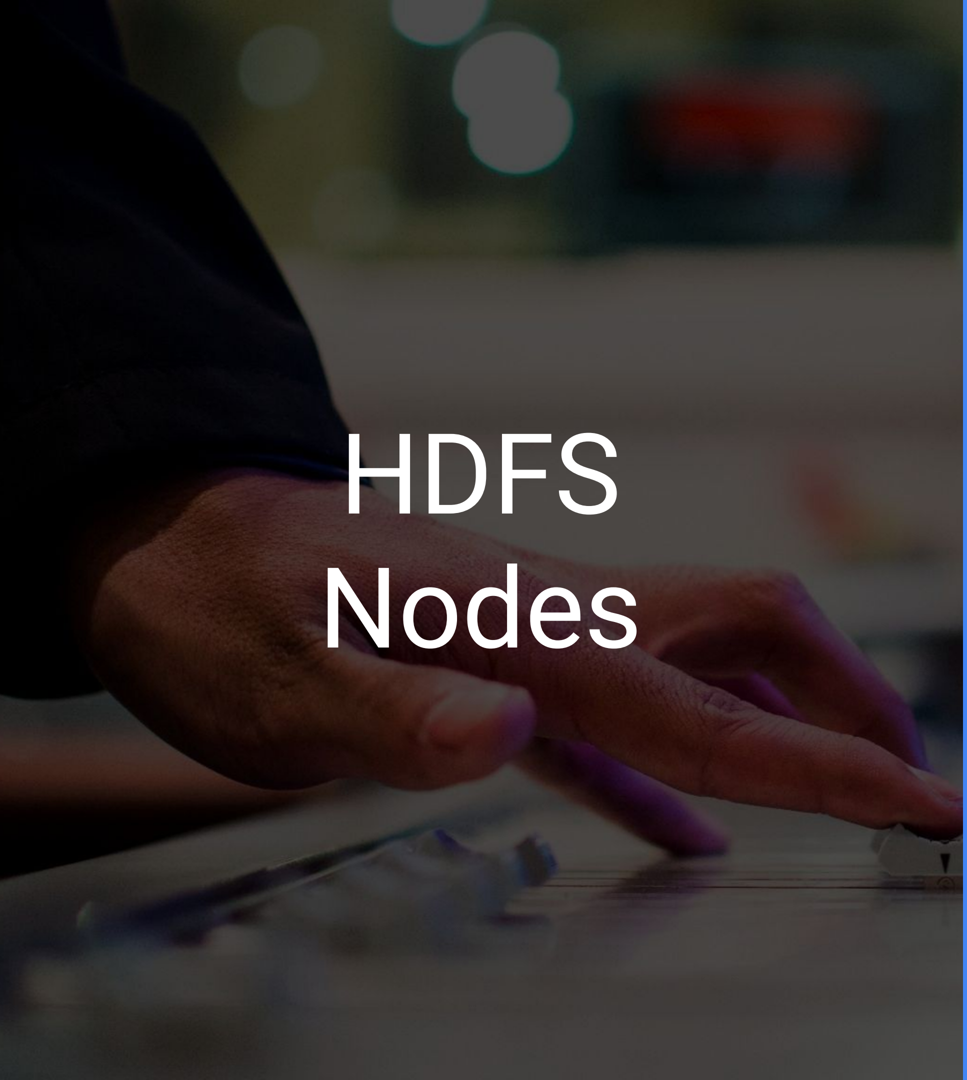
It is designed on the principle of storage of less number of large files rather than the huge number of small files.

Hadoop HDFS provides a fault-tolerant storage layer for Hadoop and its other components.

HDFS Replication of data helps us to attain this feature.

It stores data reliably, even in the case of hardware failure.

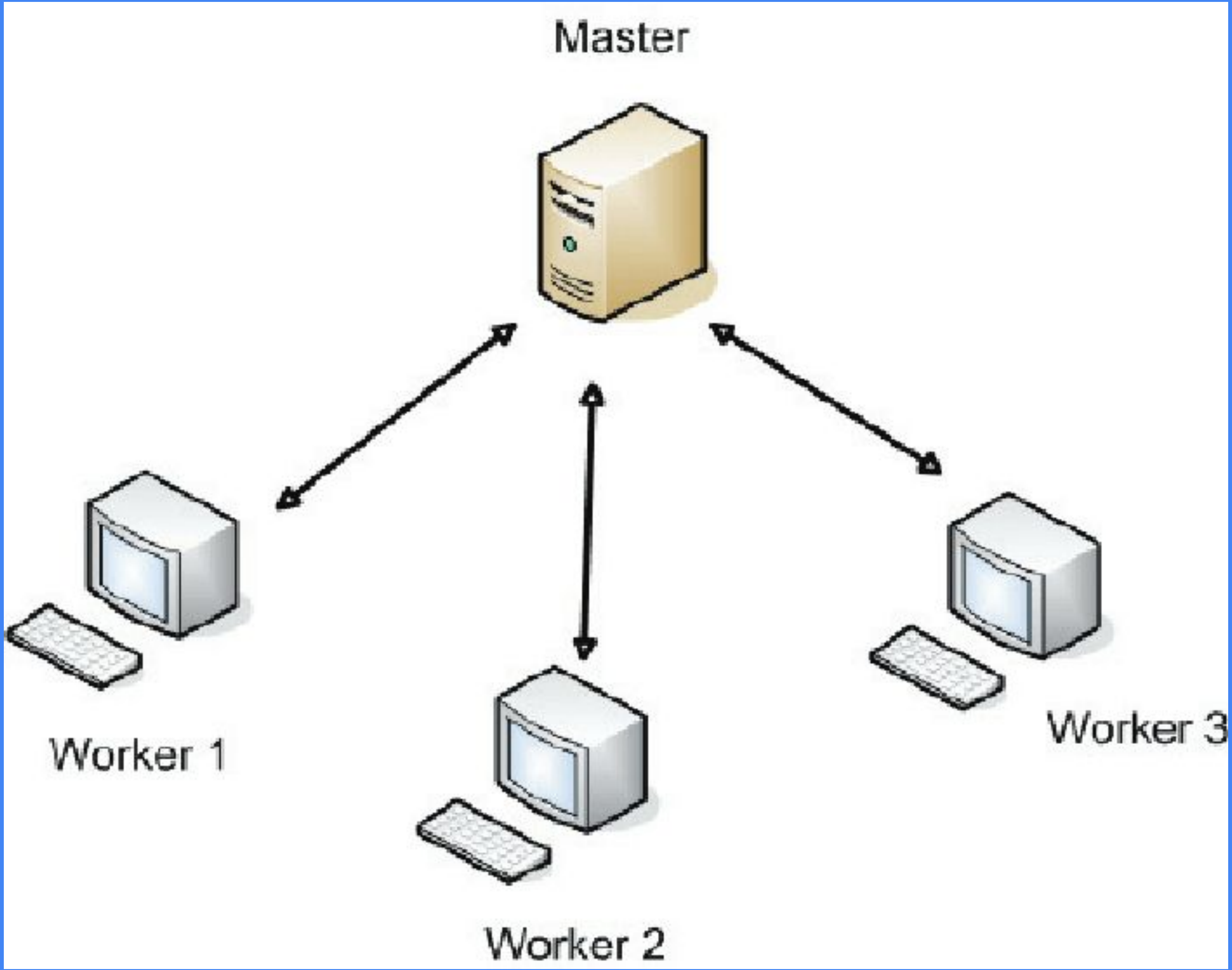
It provides high throughput access to application data by providing the data access in parallel.



HDFS Nodes

As we know, Hadoop works in master-slave fashion, HDFS also has two types of nodes that work in the same manner.

These are the NameNode(s) and the DataNodes.


Master
Worker 1
Worker 2
Worker 3

HDFS Master Node (Namenode)

Master Node is the **Namenode**, it contains the metadata.

NameNode regulates file access to the clients.

It maintains and manages the worker nodes and assigns tasks to them.

NameNode executes file system namespace operations like opening, closing, and renaming files and directories.

HDFS Worker Node (DataNode)

There are n number of slaves (where n can be up to 1000) or **DataNodes** in the Hadoop Distributed File System that manages storage of data. These slave nodes are the actual **worker nodes** that do the tasks and serve read and write requests from the file system's clients.

They perform block creation, deletion, and replication upon instruction from the **NameNode**. Once a block is written on a **DataNode**, it replicates it to other **DataNode**, and the process continues until creating the required number of replicas (the replication factor by default is 3).

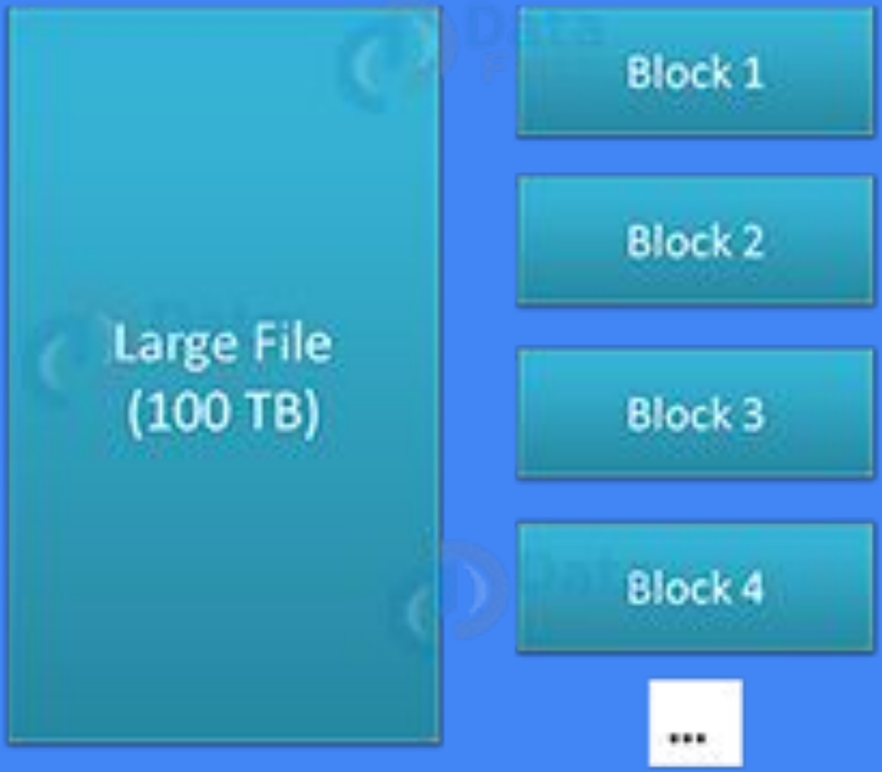
HDFS Daemons:

There are two daemons which run on HDFS for data storage:

Namenode: This is the daemon that runs on all the masters. NameNode stores metadata like filename, the number of blocks, number of replicas, a location of blocks, block IDs, etc.
This metadata is available in memory in the master for faster retrieval of data. In the local disk, a copy of the metadata is available for persistence. So NameNode memory should be high as per the requirement.

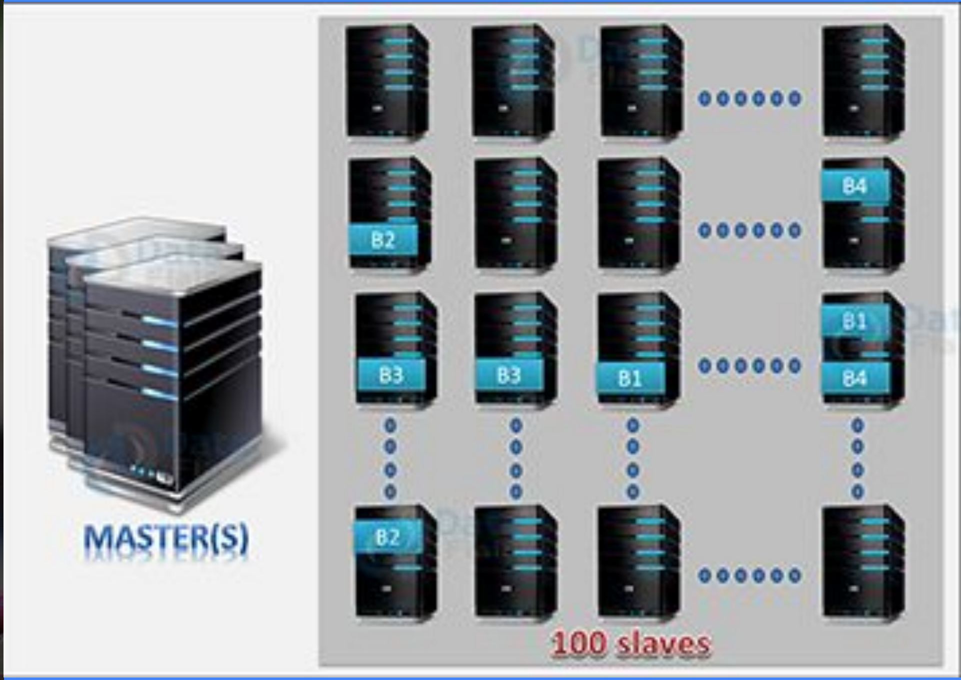
Datanode: This is the daemon that runs on the slave. These are actual worker nodes that store the data.

Hadoop **HDFS** broke the files into small pieces of data known as blocks.
The default block size in HDFS is 128 MB.
We can configure the size of the block as per the requirements.
These blocks are stored in the cluster in a distributed manner on different nodes.
This provides a mechanism for MapReduce to process the data in parallel in the cluster.



HDFS stores multiple copies of each block across the cluster on different nodes. This is a replication of data. By default, the HDFS replication factor is 3. Hadoop HDFS provides high availability, fault tolerance, and reliability.

HDFS splits a large file into n number of small blocks and stores them on different DataNodes in the cluster in a distributed manner. It replicates each block and stored them across different DataNodes in the cluster.



Rack Awareness in Hadoop HDFS

Hadoop runs on a cluster of computers spread commonly across many racks.

NameNode places replicas of a block on multiple racks for improved fault tolerance.

NameNode tries to place at least one replica of a block in a different rack so that if a complete rack goes down, then also the system will be highly available.

Optimize replica placement distinguishes **HDFS** from other distributed file systems. The purpose of a rack-aware replica placement policy is to improve data reliability, availability, and network bandwidth utilization.

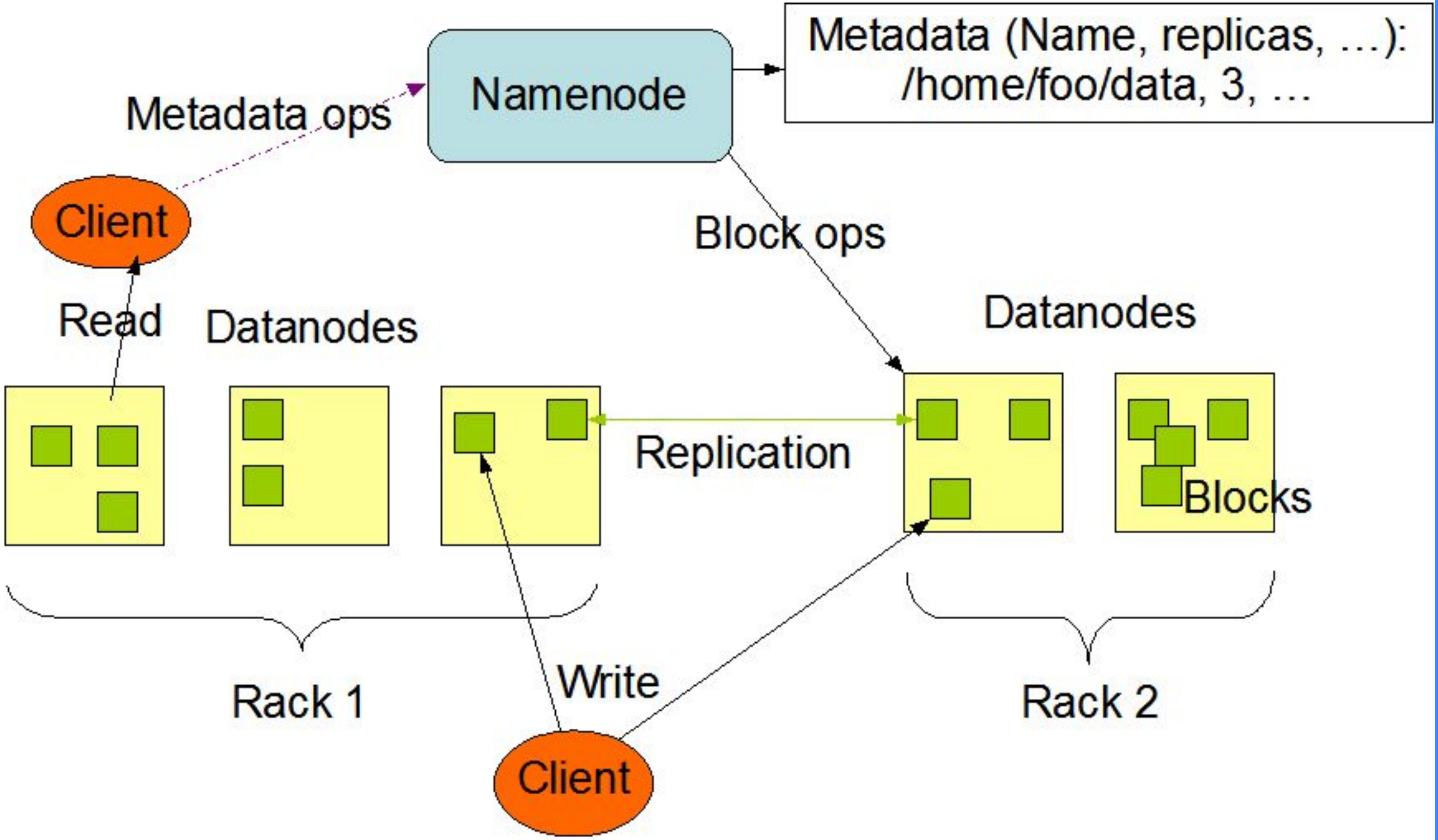
Hadoop HDFS Architecture

HDFS Architecture

This architecture gives you a complete picture of the Hadoop Distributed File System.There is a single NameNode that stores metadata, and there are multiple DataNodes that do actual storage work. Nodes are arranged in racks, and replicas of data blocks are stored on different racks in the cluster to provide fault tolerance.

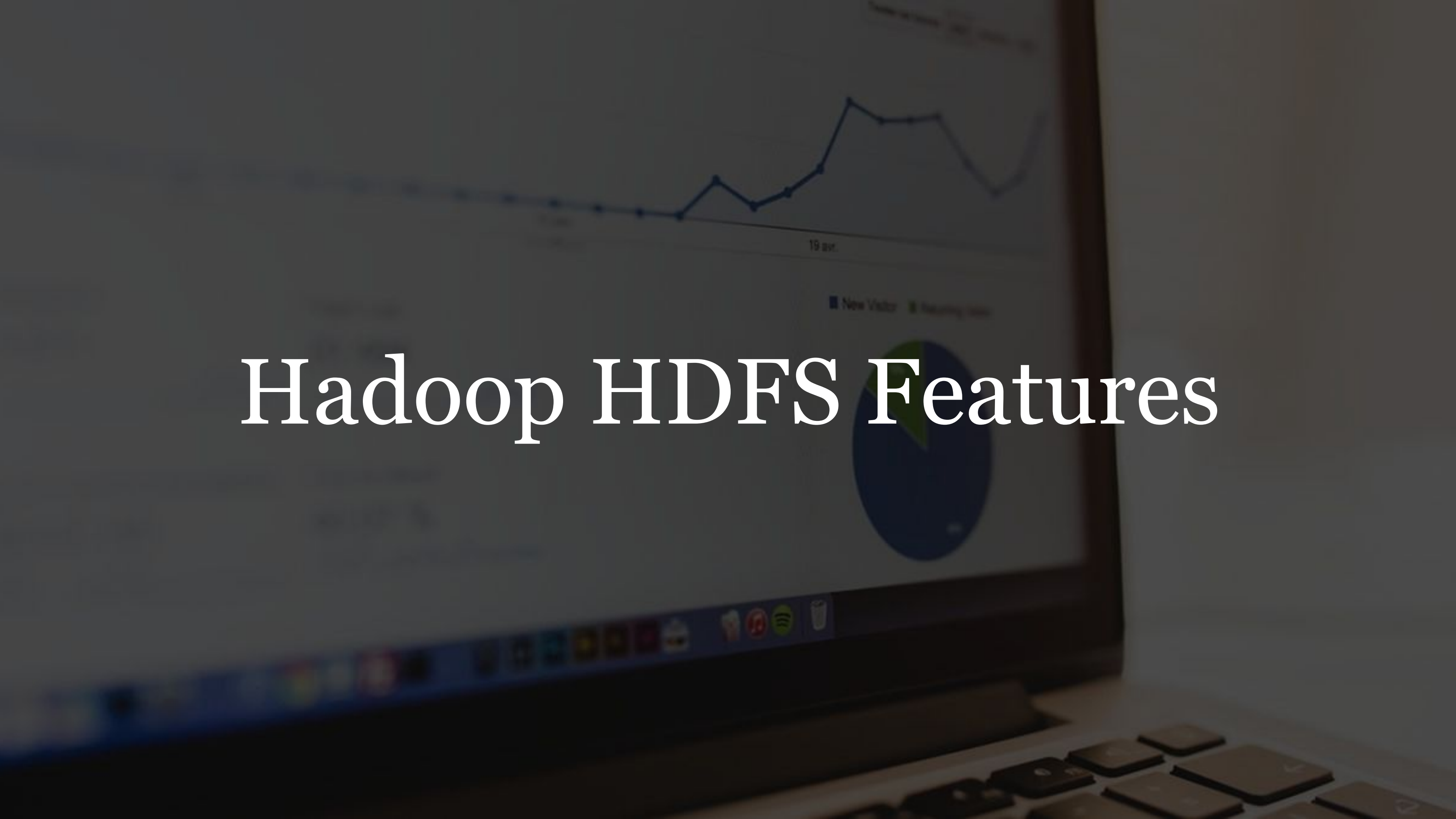
Later, we will see how read and write operations are performed in HDFS?To read or write a file in HDFS, the client needs to interact with NameNode. HDFS applications need a write-once-read-many access model for files. A file, once created and written, cannot be edited.

HDFS Architecture



NameNode stores metadata, and DataNode stores actual data. The client interacts with NameNode for performing any tasks, as NameNode is the centerpiece in the cluster.

There are several DataNodes in the cluster which store HDFS data in the local disk. DataNode sends a heartbeat message to NameNode periodically to indicate that it is alive. Also, it replicates data to other DataNode as per the replication factor.



Hadoop HDFS Features

a) Distributed Storage

HDFS stores data in a distributed manner.
It divides the data into small pieces and stores it on different **DataNodes** in the cluster.
In this manner, the Hadoop Distributed File System provides a way to MapReduce to process a subset of large data sets broken into blocks, parallelly on several nodes.
MapReduce is the heart of Hadoop, but **HDFS** is the one who provides it all these capabilities.

b) Blocks:

HDFS splits huge files into small chunks known as blocks. Block is the smallest unit of data in a filesystem. We (client and admin) do not have any control on the block like block location. NameNode decides all such things.

HDFS default block size is 128 MB. We can increase or decrease the block size as per our need. This is unlike the OS filesystem, where the block size is 4 KB.

If the data size is less than the block size of HDFS, then block size will be equal to the data size.

For example, if the file size is 129 MB, then 2 blocks will be created for it. One block will be of default size 128 MB, and the other will be 1 MB only and not 128 MB as it will waste the space (here block size is equal to data size). Hadoop is intelligent enough not to waste the rest of 127 MB. So it is allocating 1 MB block only for 1 MB data.

The major advantage of storing data in such block size is that it saves disk seek time and another advantage is in the case of processing as mapper processes 1 block at a time. So 1 mapper processes large data at a time.

c) Replication

Hadoop HDFS creates duplicate copies of each block.

This is known as replication.

All blocks are replicated and stored on different DataNodes across the cluster.

It tries to put at least 1 replica in a different rack.

What do you mean by rack?

DataNodes are arranged in **racks**.

All the nodes in a rack are connected by a single switch, so if a switch or complete rack is down, data can be accessed from another rack.

We will see it further in the rack awareness section.

As seen earlier in this Hadoop HDFS tutorial, the default replication factor is 3, and this can be changed to the required values according to the requirement by editing the configuration files (hdfs-site.xml).

d) High Availability

Replication of data blocks and storing them on multiple nodes across the cluster provides high availability of data.
As seen earlier in this Hadoop HDFS tutorial, the default replication factor is 3, and we can change it to the required values according to the requirement by editing the configuration files (hdfs-site.xml).

e) Data Reliability

As we have seen in high availability in this HDFS tutorial, data is replicated in HDFS; It is stored reliably as well.
Due to replication, blocks are highly available even if some node crashes or some hardware fails.
If the DataNode fails, then the block is accessible from other DataNode containing a replica of the block.
Also, if the rack goes down, the block is still available on the different rack.
This is how data is stored reliably in HDFS and provides fault-tolerant and high availability.

f) Fault tolerant

HDFS provides a fault-tolerant storage layer for Hadoop and other components in the ecosystem.

HDFS works with commodity hardware (systems with average configurations) that has high chances of getting crashed at any time. Thus, to make the entire system highly fault-tolerant, HDFS replicates and stores data in different places.

g) Scalability

Scalability means expanding or contracting the cluster. We can scale Hadoop HDFS in 2 ways.

Vertical Scaling: We can add more disks on nodes of the cluster.
For doing this, we need to edit the configuration files and make corresponding entries of newly added disks. Here we need to provide downtime though it is very less. So people generally prefer the second way of scaling, which is horizontal scaling.

Horizontal Scaling: Another option of scalability is of adding more nodes to the cluster on the fly without any downtime. This is known as horizontal scaling.
We can add as many nodes as we want in the cluster on the fly in real-time without any

h) High throughput access to application data

Hadoop Distributed File System provides high throughput access to application data. Throughput is the amount of work done in a unit time. It describes how fast the data is getting accessed from the system, and it is usually used to measure the performance of the system.

In HDFS, when we want to perform a task or an action, then the work is divided and shared among different systems. So all the systems will be executing the tasks assigned to them independently and in parallel. So the work will be completed in a very short period of time.

Thus, by reading data in parallel HDFS gives good throughput.

Hadoop HDFS Operations

In Hadoop, we need to interact with the file system either by programming or by the command-line interface (CLI).

Hadoop Distributed File System has many similarities with the Linux file system. So we can do almost all the operations on the HDFS File System that we can do on a local file system like create a directory, copy the file, change permissions, etc.

It also provides different access rights like read, write, and execute to users, groups, and others.

We can browse the file system here by the browser that would be like http://master-IP:50070. By pointing the browser to this URL, you can get the cluster information like space used / available, the number of live nodes, the number of dead nodes, etc.

read
write

1 - HDFS Read Operation (part1)

Whenever a client wants to read any file from HDFS, the client needs to interact with NameNode as NameNode is the only place that stores metadata about DataNodes. NameNode specifies the address or the location of the slaves where data is stored.

The client will interact with the specified DataNodes and read the data from there. For security/authentication purposes, NameNode provides a token to the client, which it shows to the DataNode for reading the file.

1 - HDFS Read Operation (part2)

In the Hadoop HDFS read operation, if the client wants to read data that is stored in HDFS, it needs to interact with NameNode first. So the client interacts with distributed file system API and sends a request to NameNode to send block location. Thus, NameNode checks if the client has sufficient privileges to access the data or not. If the client have sufficient privileges, then NameNode will share the address at which data is stored in the DataNode.

With the address, NameNode also shares a security token with the client, which it needs to show to DataNode before accessing the data for

1 - HDFS Read Operation (part3)

When a client goes to DataNode for reading the file, after checking the token, DataNode allows the client to read that particular block. A client then opens the input stream and starts reading data from the specified DataNodes. Hence, In this manner, the client reads data directly from DataNode.

During the reading of a file, if the DataNode goes down suddenly, then a client will again go to the NameNode, and the NameNode will share another location where that block is present.

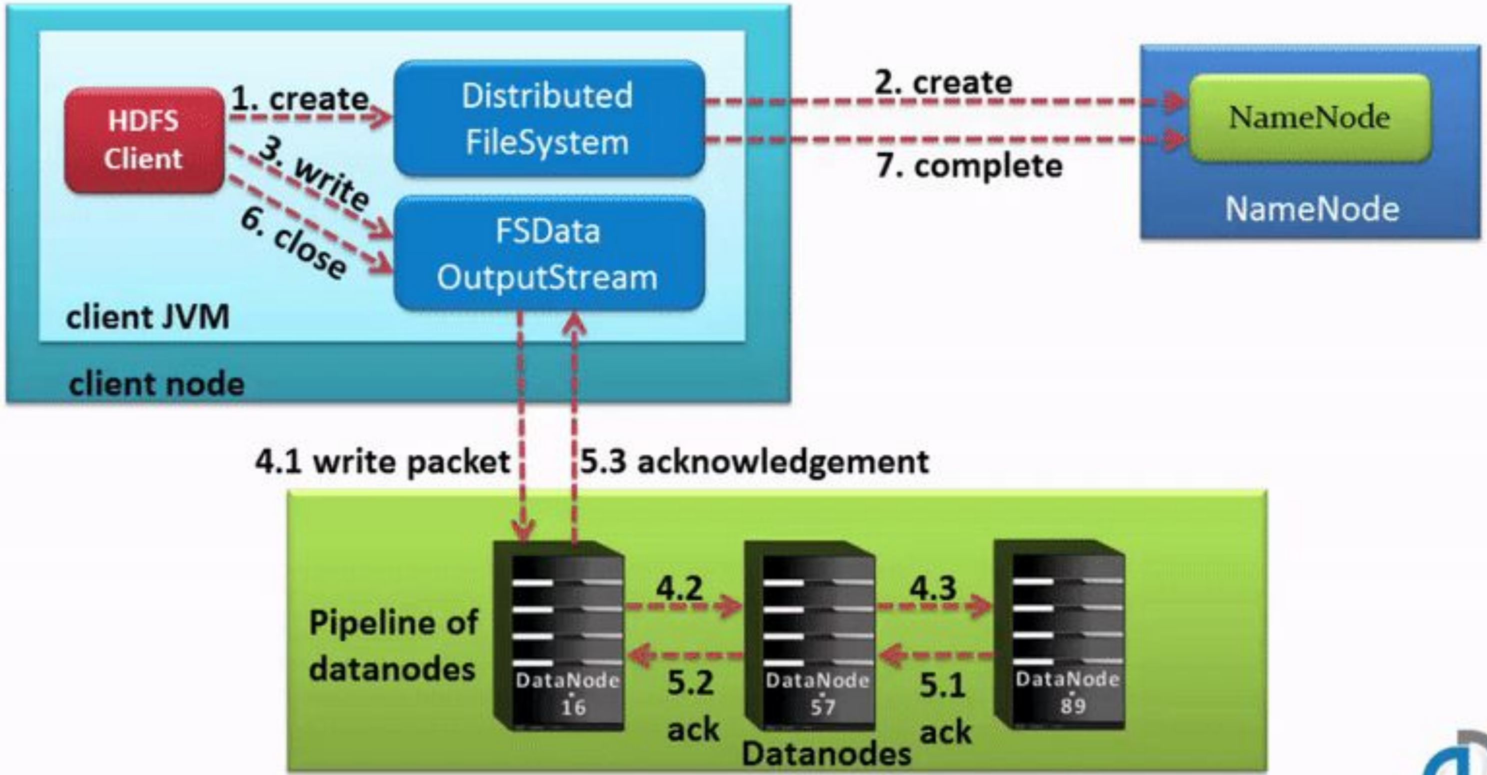
read
write

1 - HDFS Write Operation (part1)

As seen while reading a file, the client needs to interact with NameNode. Similarly, for writing a file, the client needs to interact with the NameNode.

NameNode provides the address of the slaves on which data has to be written by the client.

Once the client finishes writing the block, the slave starts replicating the block into another slave, which then copies the block to the third slave. This is the case when the default replication factor of 3 is used. After the required replicas are created, it sends a final acknowledgment to the client. The authentication process is similar, as seen in the read section.


HDFS Client
1. create
Distributed FileSystem
3. write
6. close
FSData OutputStream
client JVM
client node
2. create
7. complete
NameNode
NameNode
4.1 write packet
5.3 acknowledgement
Pipeline of datanodes
DataNode 16
4.2
5.2 ack
DataNode 57
4.3
5.1 ack
DataNode 89
Datanodes

1 - HDFS Write Operation (part2)

Whenever a client needs to write any data, it needs to interact with the NameNode. So the client interacts with distributed file system API and sends a request to NameNode to send a slave location.

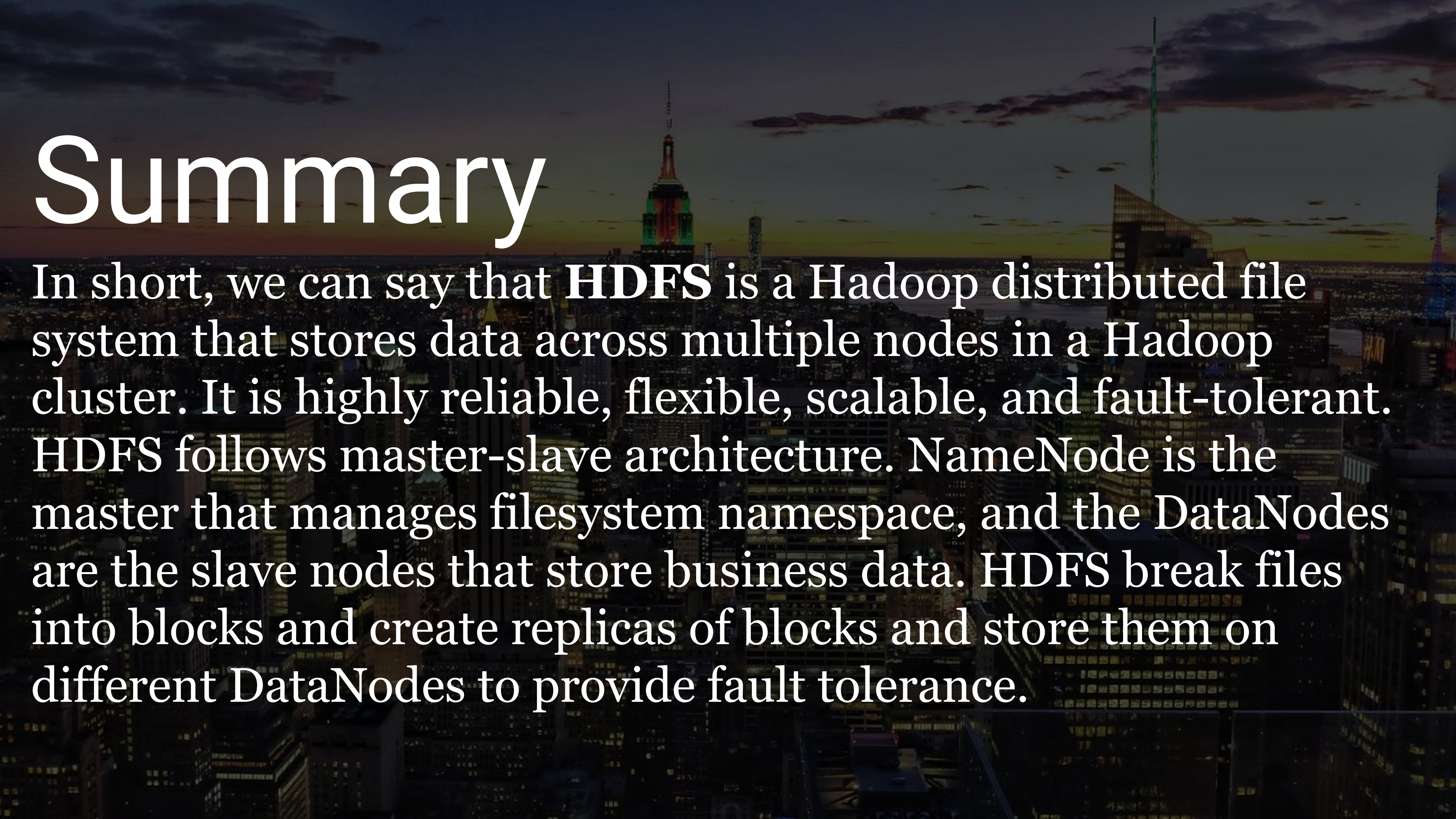
NameNode shares the location at which data has to be written.

Then the client interacts with the DataNode at which data has to be written and starts writing the data through the FS data output stream. Once the data is written and replicated, the DataNode sends an acknowledgment to the client informing that the data is written completely.

1 - HDFS Write Operation (part3)

As soon as the client finishes writing the first block, the first DataNode will copy the same block to other DataNode. Thus this DataNode after receiving the block starts copying this block to the third DataNode. Third sends an acknowledgment to second, the second DataNode sends an acknowledgment to the first DataNode, and then the first DataNode sends the final acknowledgment (in the case of default replication factor).

The client is sending just 1 copy of data irrespective of our replication factor, while DataNodes replicate the blocks. Hence, Writing of file in Hadoop HDFS is not costly as parallelly multiple blocks are getting written on several DataNodes.



Summary

In short, we can say that **HDFS** is a Hadoop distributed file system that stores data across multiple nodes in a Hadoop cluster. It is highly reliable, flexible, scalable, and fault-tolerant. HDFS follows master-slave architecture. NameNode is the master that manages filesystem namespace, and the DataNodes are the slave nodes that store business data. HDFS break files into blocks and create replicas of blocks and store them on different DataNodes to provide fault tolerance.