// DISTRIBUTED FILE SYSTEM

==========

- **Distributed system** is a collection of independent computers that appears to its users as a single coherent system
  - Scalability
  - Reliability
  - Availability
  - Communication

- **Cluster Systems**
  - Master node: - Master node controls storage allocation and job scheduling
  - Computational nodes: - does job , have similar OS, similar computers

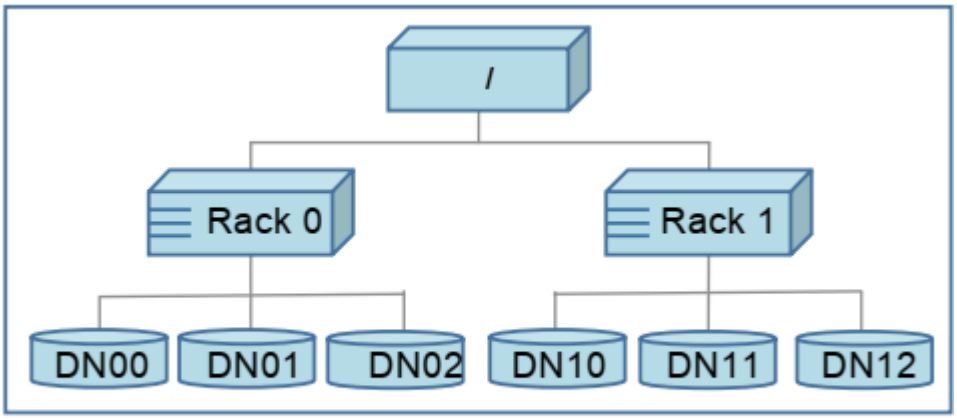
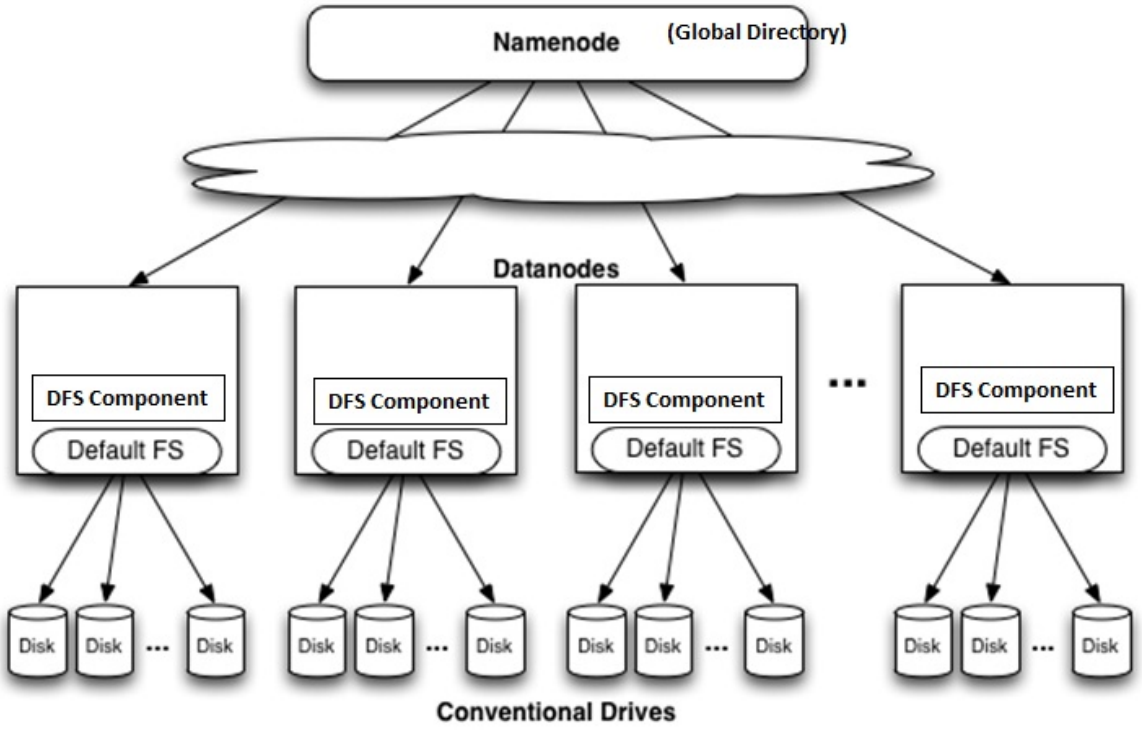


Figure 3. Cluster topology example

DISTRIBUTED FILE SYSTEM



- **<u>Distributed FIle System</u>**
  - **client / server-based application that allows clients to access and process data stored on the server**

// NAMENODE

==========

- **NameNode (refer above image)**
  - The **NameNode** maintains the **metadata** (**inode data** and the **mapping of file blocks to DataNodes** **)**
  - **inode data**
    - Files and directories are represented on the NameNode by **inodes**, which record attributes like permissions, modification and access times, **namespace(a hierarchy of files and directories)** and disk space quotas.
- **Image**
  - A snapshot of the metadata information is called image

// CHECKPOINT AND JOURNAL
==========

- **Checkpoint**
  - Image stored in localhost file system is called a checkpoint

- **Journal**
  - Modification Log of the Image stored in localhost file system is called journal

- **Note**
  - During restarts the NameNode restores the **namespace (a hierarchy of files and directories)** by reading the namespace(restored from checkpoint) and replaying the journal.
  - **If either the checkpoint or the journal is missing, or becomes corrupt, the namespace information will be lost partly or entirely.**
  - **In order to preserve this critical information HDFS can be configured to store the checkpoint and journal in multiple storage directories.**
  - **Recommended practice is to place the directories on different volumes**
  - **A role of NameNode : Checkpoint Node** : periodically combines the existing checkpoint and journal to create a new checkpoint and an empty journal
  - **Another Role of NameNode: Backup Node :** The BackupNode is capable of creating periodic checkpoints, but in addition it maintains an in memory, up-to-date image of the file system namespace that is always synchronized with the state of the NameNode

// DATA NODE
==========

- Data Node(refer above image)
  - Stores Data (obviously)
  - Files are divided into 128 byte blocks and stored.
  - **each block of the file** is independently **replicated at multiple DataNodes** (typically three)

- **Periodic Checks by Data Node**
    - **During startup** each DataNode connects to the NameNode and performs a **handshake**. The purpose of the handshake is to **verify the namespace ID and the software version of the DataNode**. If either does not match that of the NameNode the DataNode automatically shuts down
    - **Every hour**, a DataNode sends a **block report** about replicas A block report contains the **block id, the generation stamp and the length for each block replica** the server hosts
    - **Every 3 seconds** , DataNodes send **heartbeats** to the NameNode to **confirm that the DataNode is available**
      Heartbeat contains several types of information
      If the NameNode does not receive a heartbeat from a DataNode in ten minutes the NameNode considers the DataNode to be out of service and the block replicas hosted by that DataNode to be unavailable.

// NAMENODE TO DATANODE COMMUNICATION
==========

- NameNode to DataNode Communication
    - It uses replies to heartbeats to send instructions to the DataNodes. The instructions include commands to:
        - replicate blocks to other nodes
        - remove local block replicas;
        - re-register or to shut down the node;
        - send an immediate block report.

// FILE READ AND WRITE
==========

- **Hadoop File System Client (simply Client)(UI)**
    - refers to the software library or set of tools that allow applications and users to interact with the Hadoop Distributed File System

- **Read and Write Facts**
    - HDFS implements a single-writer, multiple-reader model.
    - The HDFS client that opens a file for writing is **granted a lease for the file, no other client can write to the file**.

- The writing client periodically renews the lease by sending a heartbeat to the NameNode.
- When the **file is closed, the lease is revoked**

- **Read using HDFS**
  - **Sends request** for each block **to read** NameNode
  - Name node **returns addresses of** DataNodes containing **replicas** of requested file blocks (usually 3). Usually closest Data node is chosen for reading

- **Write using HDFS**
  - **Sends request** for each block of data to write NameNode
  - If there is a need of new block, NameNode allocates a unique block with an id
  - Name node **returns addresses of** DataNodes to store **replicas** of requested file blocks (usually 3). **Write happens in all 3.**
  - Writing happens in the form of a pipeline, where DataNodes are arranged in the order of distance from NameNode
  - Bytes are pushed to the pipeline as a sequence of packets.

// REPLICATION MANAGEMENT
==========

- **Replication management by NameNode**

  - Analyze block report every hour to find whether a block is under replicated or over replicated
  - If over replicated, removes a replica
  - Also makes sure that all replicas are not in one rack

//EXTRA TOOLS

==========

- **Balancer(Moves replicas around)**
  - It iteratively moves replicas from DataNodes with **higher space utilization to DataNodes with lower space utilization**
- **Block Scanner**
  - Run By DataNode
  - **Scans Replicas, take their checksum , and ensures they match with original**
  - If a replica is corrupt, it notifies the namenode.
  - The NameNode marks the replica as corrupt, but does not schedule deletion of the replica immediately. Instead, it starts to replicate a good copy of the block.
  - This policy aims to preserve data as long as possible. So even if all replicas of a block are corrupt, the policy allows the user to retrieve its data from the corrupt replicas
- **Cluster Administrator**
  - **specifies which Datanodes can join the cluster** by listing the host addresses of Datanodes.
  - Can Decommission DataNodes, A present DataNode of the cluster that becomes excluded is marked for decommissioning.
  - To decommission a datanode, all Datanode data is replicated, then decommissioned, it will be active until then.

- **DistCp**
  - For large inter/intra-cluster parallel copying. It is a MapReduce job; each of the map tasks copies a portion of the source data into the destination file system.
  - The MapReduce framework automatically handles parallel task scheduling, error detection and recovery