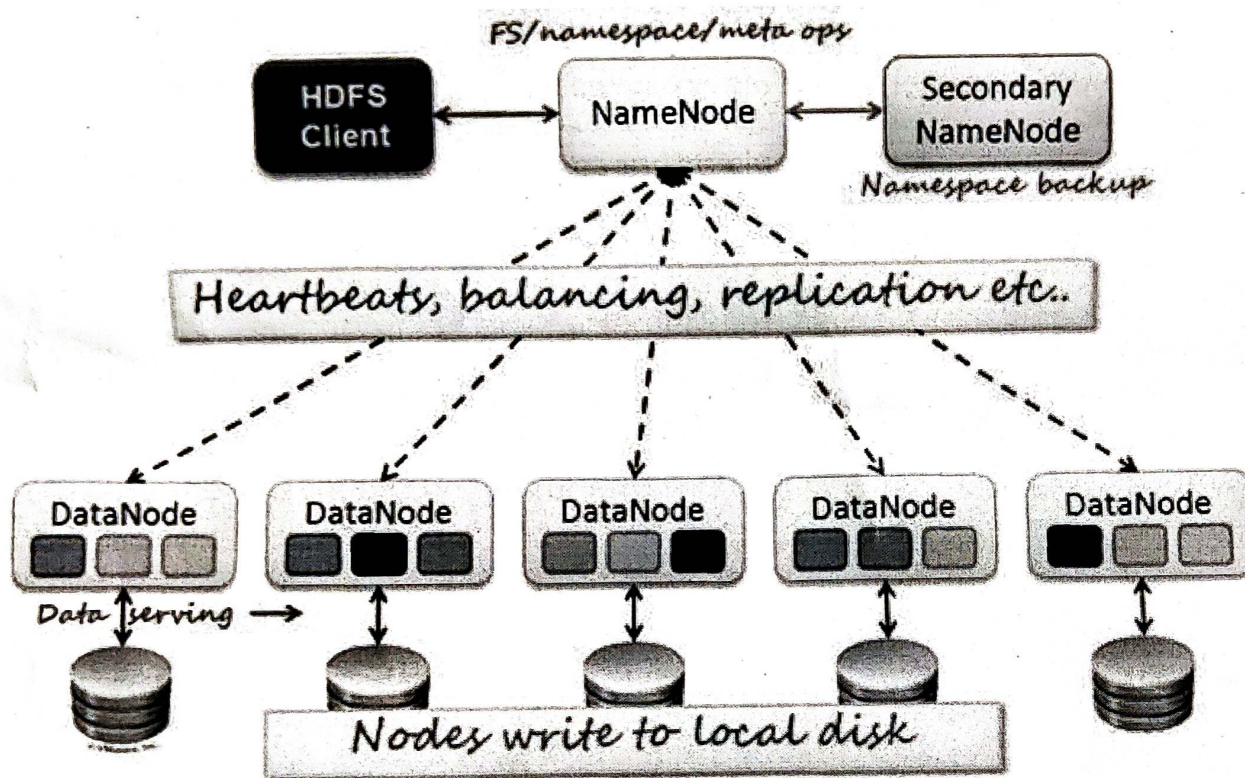


HDFS is a block-structured file system: individual files are broken into blocks of a fixed size. These blocks are stored across a cluster of one or more machines with data storage capacity. Individual machines in the cluster are referred to as DataNodes. A file can be made of several blocks, and they are not necessarily stored on the same machine; the target machines which hold each block are chosen randomly on a block-by-block basis. Thus access to a file may require the cooperation of multiple machines, but supports file sizes far larger than a single-machine DFS; individual files can require more space than a single hard drive could hold.

If several machines must be involved in the serving of a file, then a file could be rendered unavailable by the loss of any one of those machines. HDFS combats this problem by replicating each block across a number of machines (3, by default).



Most block-structured file systems use a block size on the order of 4 or 8 KB. But, the default block size in HDFS is 128MB for fast streaming reads of data, by keeping large amounts of data sequentially laid out on the disk. Thus, HDFS expects to store a modest number of very large files: hundreds of megabytes, or gigabytes each. Files on your computer may also frequently be accessed "randomly," with applications cherry-picking small amounts of information from several different locations in a file which are not sequentially laid out. By contrast, HDFS expects to read a block start-to-finish for a program.

Because HDFS stores files as a set of large blocks across several machines, these files are not part of the ordinary file system. HDFS runs in a separate namespace, isolated from the contents of local files. The files inside HDFS (or more accurately: the blocks that make them up) are stored in a particular directory managed by the DataNode service, but the files will named only with block ids.

It is important for this file system to store its metadata reliably. Furthermore, while the file data is accessed in a write once and read many model, the metadata structures (e.g., the names of files and directories) can be modified by a large number of clients concurrently. It is important that this information is never desynchronized. Therefore, it is all handled by a single machine, called the NameNode. The NameNode

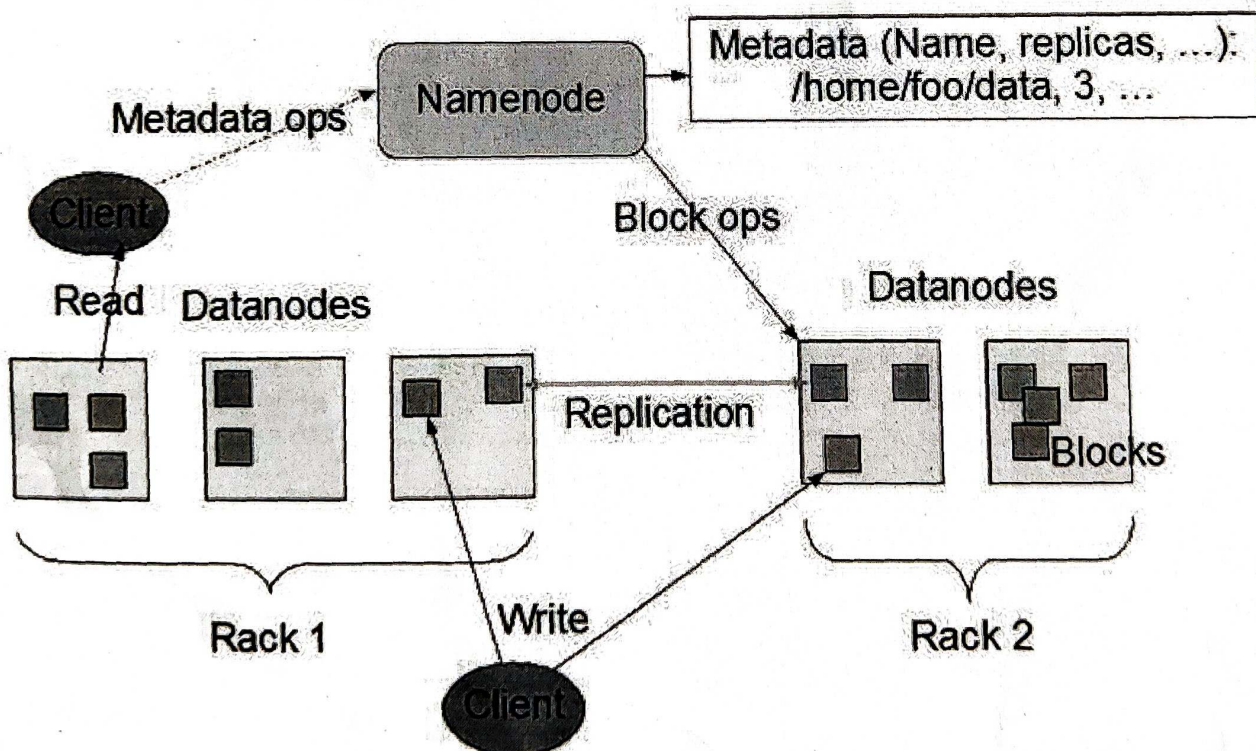


Stores all the metadata for the file system. Because of the relatively low amount of metadata per file (it only tracks file names, permissions, and the locations of each block of each file), all of this information can be stored in the main memory of the NameNode machine, allowing fast access to the metadata.

To open a file, a client contacts the NameNode and retrieves a list of locations for the blocks that comprise the file. These locations identify the DataNodes which hold each block. Clients then read file data directly from the DataNode servers, possibly in parallel. The NameNode is not directly involved in this bulk data transfer, keeping its overhead to a minimum.

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

### HDFS Architecture



The existence of a single NameNode in a cluster greatly simplifies the architecture of the system. The NameNode is the arbitrator and repository for all HDFS metadata. The system is designed in such a way that user data never flows through the NameNode.

HDFS supports a traditional hierarchical file organization. A user or an application can create directories and store files inside these directories. The file system namespace hierarchy is similar to most other existing file systems; one can create and remove



, move a file from one directory to another, or rename a file. HDFS does not yet implement user quotas. HDFS does not support hard links or soft links. However, the HDFS architecture does not preclude implementing these features.

The NameNode maintains the file system namespace. Any change to the file system namespace or its properties is recorded by the NameNode. An application can specify the number of replicas of a file that should be maintained by HDFS. The number of copies of a file is called the replication factor of that file. This information is stored by the NameNode.

## Metadata Persistence

The NameNode uses a transaction log called the EditLog to persistently record every change that occurs to file system metadata. For example, creating a new file in HDFS causes the NameNode to insert a record into the EditLog indicating this. Similarly, changing the replication factor of a file causes a new record to be inserted into the EditLog. The NameNode uses a file in its local host OS file system to store the EditLog. The entire file system namespace, including the mapping of blocks to files and file system properties, is stored in a file called the FsImage. The FsImage is stored as a file in the NameNode's local file system too.

The NameNode keeps an image of the entire file system namespace and file Blockmap in memory. This key metadata item is designed to be compact, such that a NameNode with 4 GB of RAM is plenty to support a huge number of files and directories. When the NameNode starts up, it reads the FsImage and EditLog from disk, applies all the transactions from the EditLog to the in-memory representation of the FsImage, and flushes out this new version into a new FsImage on disk. It can then truncate the old EditLog because its transactions have been applied to the persistent FsImage. This process is called a checkpoint. In the current implementation, a checkpoint only occurs when the NameNode starts up. Work is in progress to support periodic checkpointing in the near future.

The DataNode stores HDFS data in files in its local file system. The DataNode has no knowledge about HDFS files. It stores each block of HDFS data in a separate file in its local file system. The DataNode does not create all files in the same directory. Instead, it uses a heuristic to determine the optimal number of files per directory and creates subdirectories appropriately. It is not optimal to create all local files in the same directory because the local file system might not be able to efficiently support a huge number of files in a single directory. When a DataNode starts up, it scans through its local file system, generates a list of all HDFS data blocks that correspond to each of these local files and sends this report to the NameNode: this is the Blockreport.

## Communication Protocol

All HDFS communication protocols are layered on top of the TCP/IP protocol. A client establishes a connection to a configurable TCP port on the NameNode machine. It talks the ClientProtocol with the NameNode. The DataNodes talk to the NameNode using the DataNode Protocol. A Remote Procedure Call (RPC) abstraction wraps both the Client Protocol and the DataNode Protocol. By design, the NameNode never initiates any RPCs. Instead, it only responds to RPC requests issued by DataNodes or clients.