BIG DATA ANALYTICS FRAMEWORK

Unit II

Fundamentals of Hadoop

HDFS is a distributed, fault tolerant, storage filesystem designed to store large quantities of data (100s of TB) on a cluster of nodes using commodity hardware

Data Replication - Creating and maintaining multiple copies of the same data in different locations to ensure data availability, reliability, and resilience

- Files stored in the HDFS are broken into blocks
- Blocks replicated across the cluster for fault tolerance, redundancy, and availability.
- The default block replication is three.
- The block replication may be set on a per-file basis.
- The maximum block replication is 512 by default and configured with the *dfs.replication.max property in hdfs-default.xml* .
- The minimal block replication is 1 and set with the *dfs. namenode.replication.min property.*

Blocks are replicated for the following reasons:

- Data durability. If one block replica is lost due to machine failure or block corruption, data is not lost as other block replicas are available.
- Fault-tolerance. If a block replica is lost, another block can be used and a client can access another copy of the block or MapReduce application accessing the file is still able to access the file. Fault-tolerance provides data reliability.
- Data locality. Data locality is the closeness of the map input data to the map task.

With more block replicas, data is local to more nodes.

- Speculative execution. If a task is slow, redundant copies of the task are launched

and the task that completes first is used. With block replicas speculative execution becomes feasible.

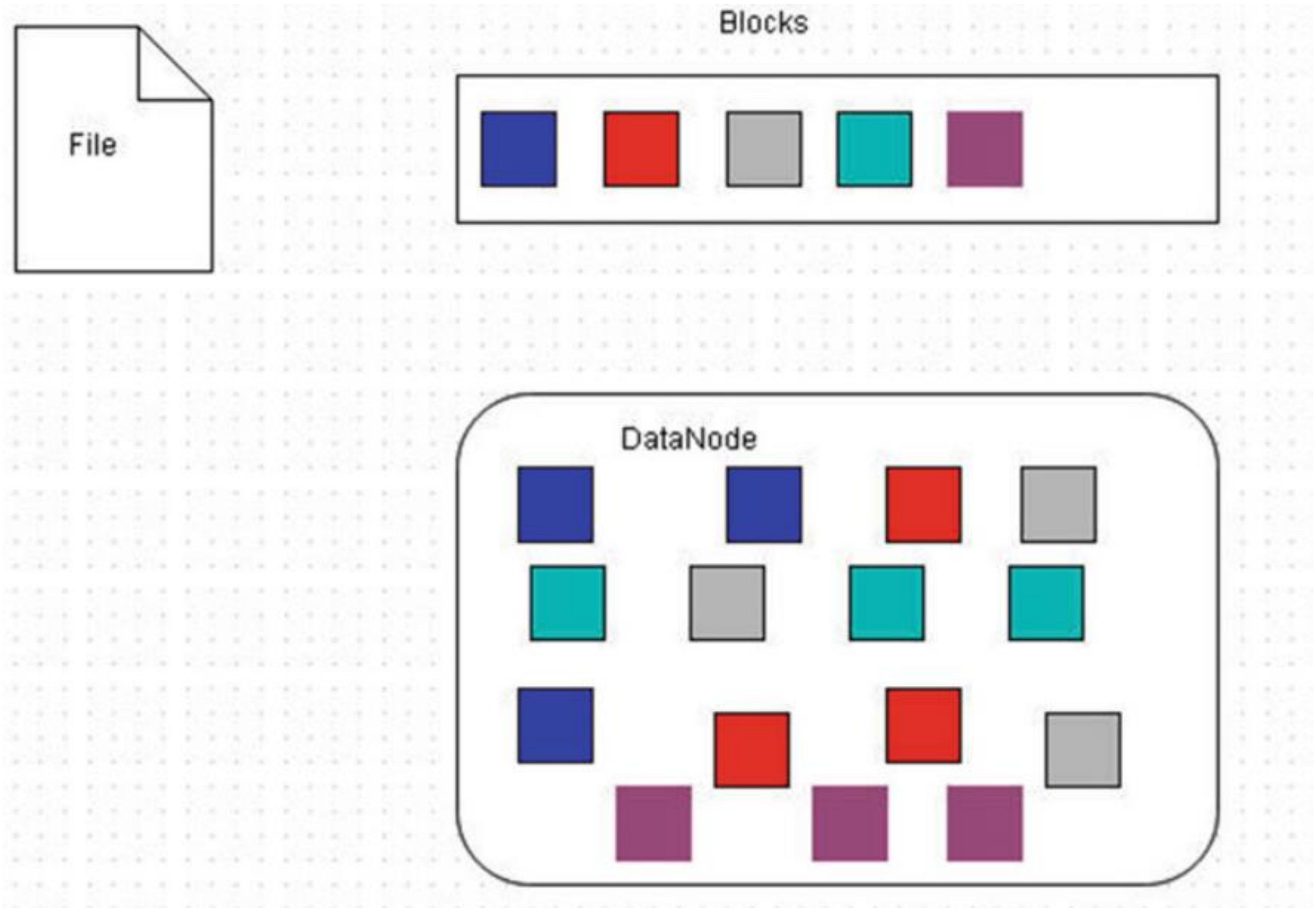
Configuring Replication

The default block replication is configured using the configuration property dfs.replication in hdfs-site.

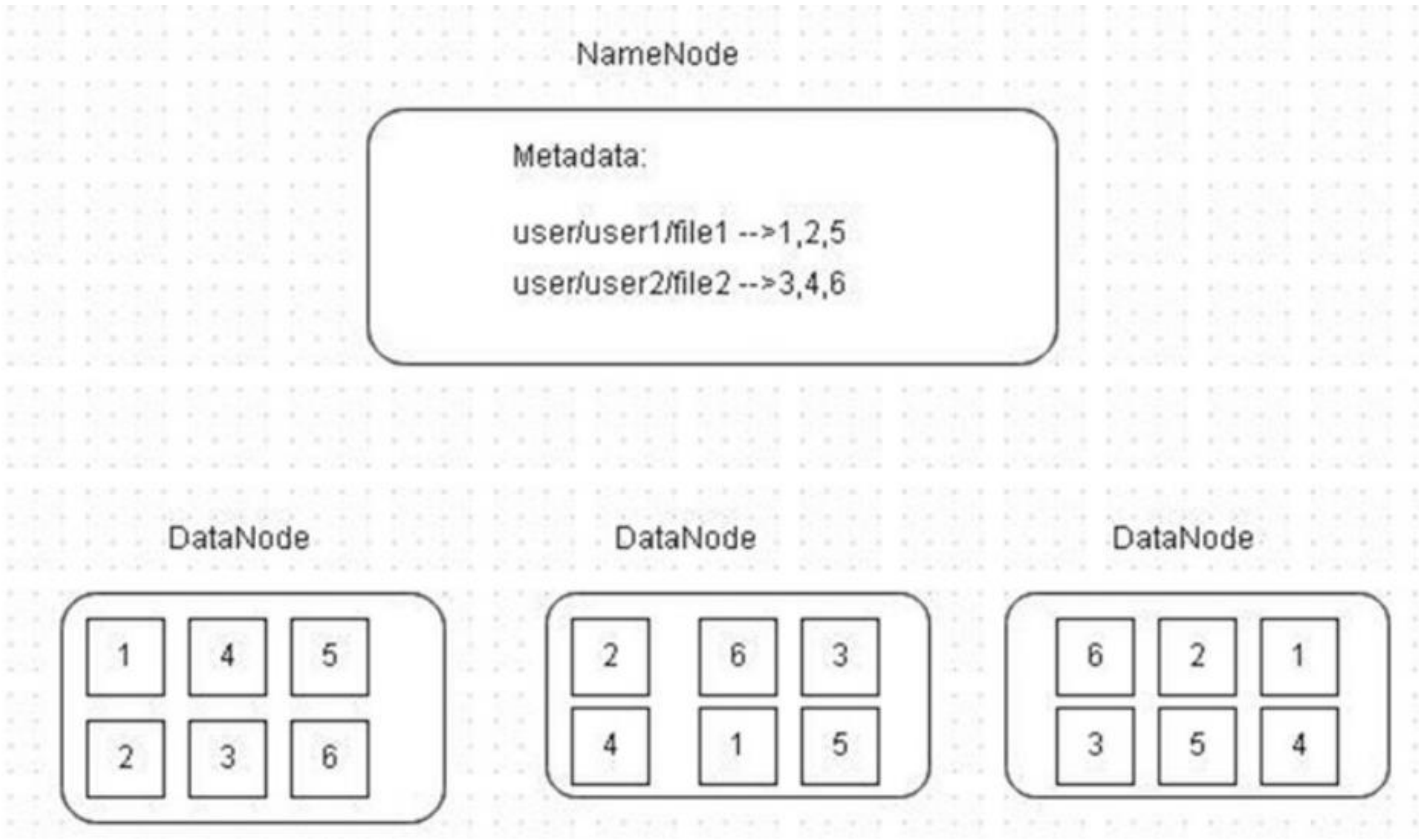
xml or hdfs-default.xml .

```
<configuration>
<property>
<name>dfs.replication</name>
<value>3</value>
</property>
</configuration>
```

Block replication example



NameNode keeps the block locations



The sequence followed in replicating file blocks

Step 1 : The client sends a request to create a file using the FileSystem API. First, a temporary internal buffer is created for the file. The client sends file data to the internal buffer. A client-side buffer is used to improve network throughput. Without client-side buffering network speed and congestion reduces the throughput.

Step 2 : When the internal buffer has data equivalent to fill one block, the client request is forwarded to the NameNode

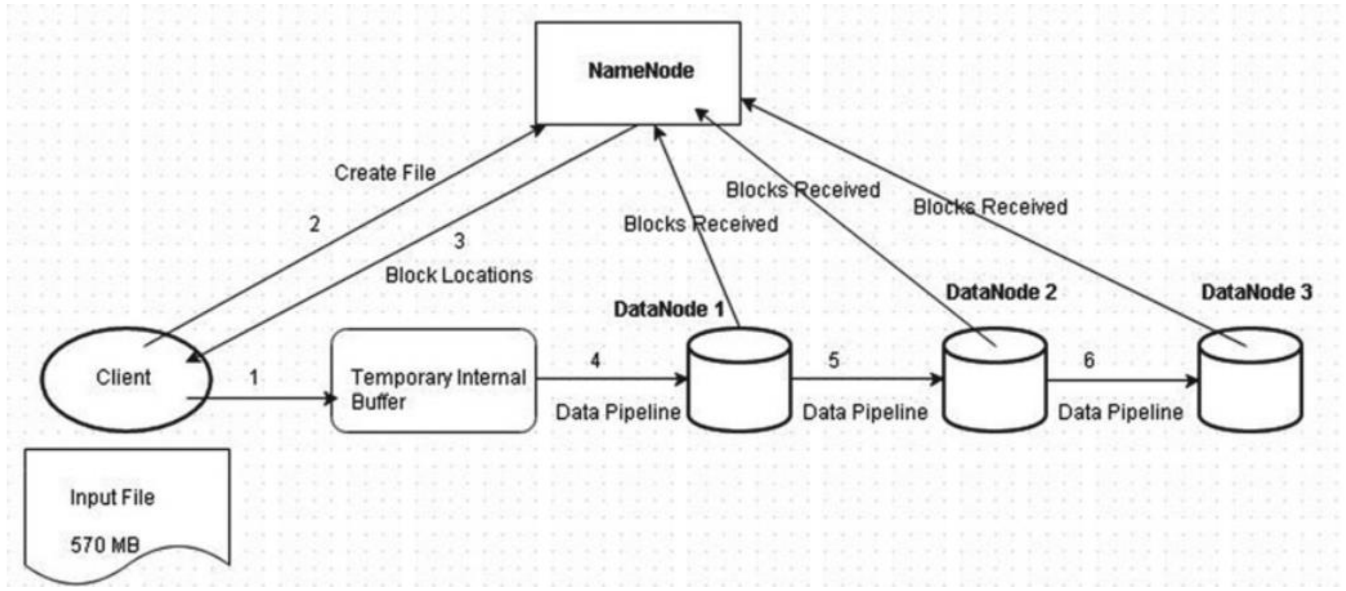
Step 3 :The NameNode assigns blocks and DataNodes for the configured number of replicas for the block and sends the block locations and DataNodes hosting the blocks to the client. The DataNodes and block locations are ordered according to their proximity from the client.

Step 4: The block of data in the internal buffer is sent to the first DataNode in the pipeline in a sequence of packets

The sequence followed in replicating file blocks

Step 5 : When the first DataNode has received a packet of data and written the packet to its repository it forwards the packet of data to the second DataNode in the pipeline.

Step 6: The second DataNode receives block data a packet at a time and fills a block replica. The second DataNode forwards block data, a packet at a time, to the third DataNode in the pipeline. The DataNodes send acknowledgement message to the NameNode to indicate that they have received blocks.



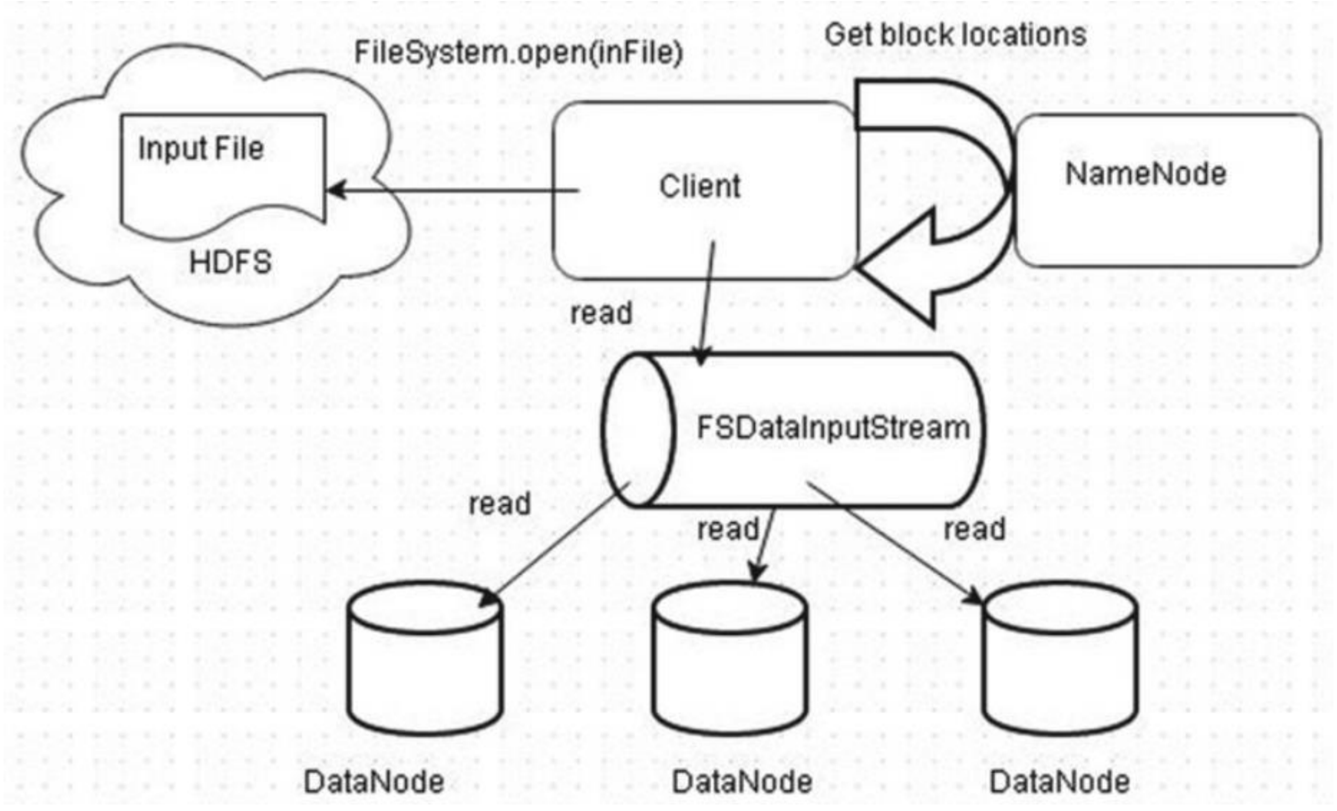
How Does HDFS Store, Read, and Write Files?

Reading a File :

✓ To read a file from the HDFS, first obtain a FileSystem object.

✓Create a Configuration object using the default configuration parameters.

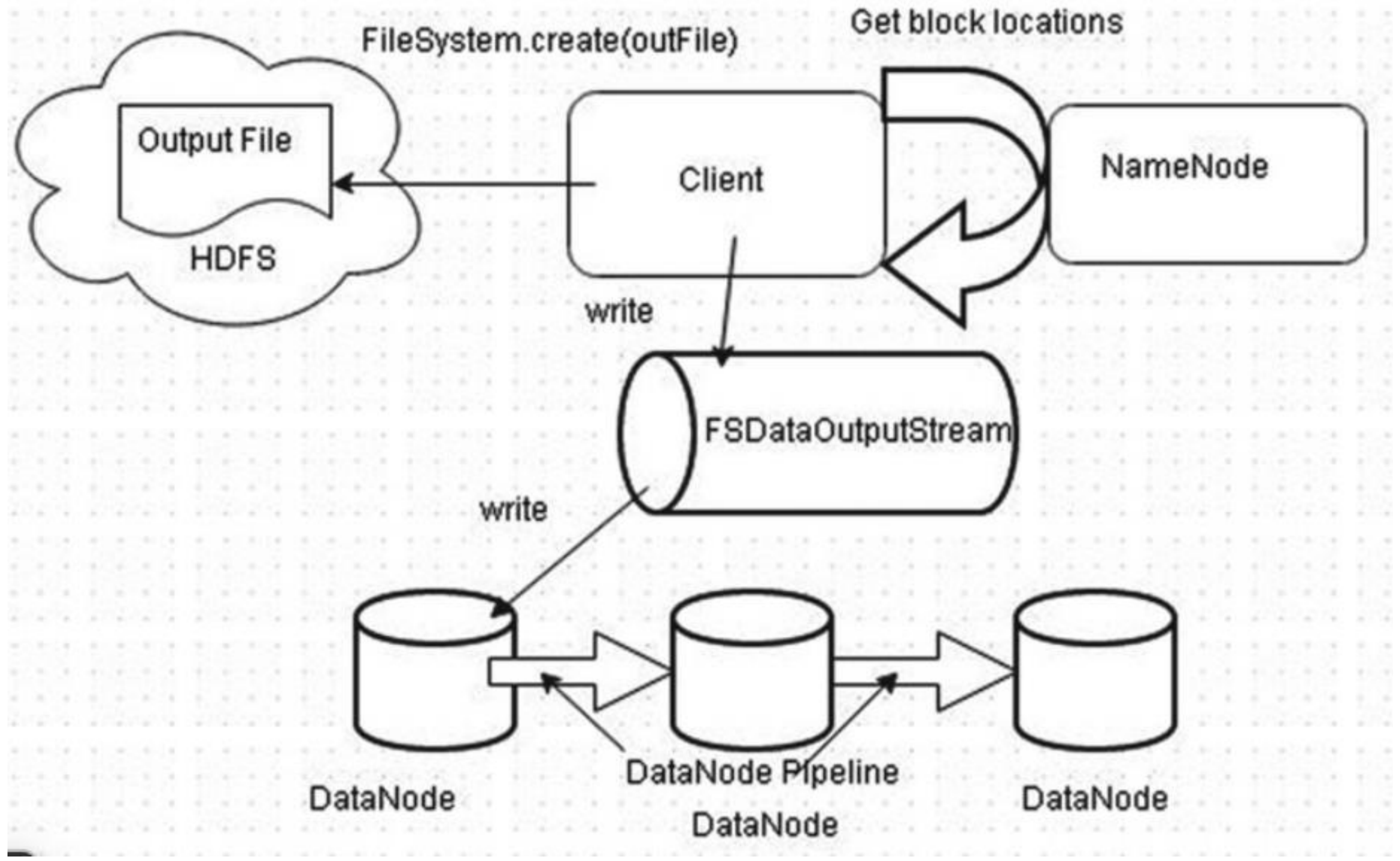
✓If hdfs-site.xml and hdfs-default.xml are not in the classpath, add them to the classpath.



How Does HDFS Store, Read, and Write Files?

Writing a File

The procedure to write a file to the HDFS is similar to reading a file. First, obtain a FileSystem object using a Configuration object created with the default configuration parameters



How Does HDFS Store, Read, and Write Files?

Storing a File

- ✓ HDFS stores files on DataNodes local filesystem.
- ✓A file is broken into blocks and the blocks are replicated, three times by default and configurable with the dfs.replication configuration property.
- ✓The HDFS blocks are an abstraction over the underlying filesystem.
- ✓Each block is stored as a data file and a corresponding checksum file.
- ✓The data files and the checksum files are stored in the DataNode’s data directories, which are configured in hdfs-default.xml .

Data Serialization Options

Data serialization is the process of converting data in memory to bytes that are transmitted over the network or written to disk.

Data serialization is performed at three phases of a MapReduce application .

- ✓ Serialization to the wire (network) for interprocess communication (IPC) using RPC
- ✓ Serialization of intermediate map outputs locally to disk
- ✓ Serialization at the reducers, which is also the MapReduce application output, to HDFS

Data Serialization Options

Writables - The default serialization interface in Hadoop is *org.apache.hadoop.io.Writable*.

- ✓A Writable is a serializable object that implements simple, efficient, serialization protocol based on DataInput and DataOutput.
- ✓The DataInput interface is used to read bytes from a binary stream and convert to any of the Java primitive types such as short, int, double.
- ✓DataInput may also be used to read a string from data in a slightly modified UTF8 format.
- ✓The DataOutput interface is used to convert from any Java primitive types and outputting bytes to a binary stream

Data Serialization Options

Avro - Avro has replaced Hadoop Record I/O, which was used, but is presently deprecated, for simplifying serialization and deserialization of records in a language-neutral method.

Avro is a schema-based serialization framework, which stores data using Avro data files.

The schema is serialized with the data file, which has the advantage of not having to store the type information in the data file

Data Serialization Options

Thrift

Thrift is a software library and a code-generation toolset developed with the goal to enable reliable and efficient communication across programming languages by providing a common library for the data types and service interfaces.

All the data types and interfaces may be defined in a single language-neutral Thrift file to create RPC clients and servers efficiently.

Thrift’s architecture consists of two layers in the runtime library:

protocol layer and transport layer.

- ✓The protocol layer provides the serialization and deserialization
- ✓The transport layer provides the reading and writing to the network.

Data Serialization Options

Protocol Buffers

Protocol buffers are a language-neutral and platform-neutral mechanism for serializing structured data and for interprocess communication (IPC).

Language-neutral implies implementations for protocol buffers are available in all the commonly used languages such as Java, C++, PHP, and JavaScript.

Choosing a Serialization Mechanism

Serialization to the wire.

Protocol buffers or Thrift, protocol buffers being the default.

- **Serialization of intermediate map outputs locally to disk.**

Avro or SequenceFiles

- **Serialization of output from reducers**

Writables or Avro are suitable based on other factors..

Filesystem Shell Commands for HDFS

Hadoop filesystem commands are run using the bin/hdfs script as follows.

hdfs dfs [COMMAND [COMMAND_OPTIONS]]

Making a Directory

The mkdir command is used to create one or more directories in the HDFS. The mkdir command is used as follows.

hdfs dfs - mkdir [-p] <paths>

Listing Files and Directories

The ls command lists the stats for a given file or directory. For a given directory, the command lists the files and subdirectories. Its usage is as follows.

hdfs dfs -ls <args>

Filesystem Shell Commands for HDFS

Putting Files in the HDFS

The put command is used to put files from the local filesystem, or stdin in the HDFS. The put command is as follows.

hdfs dfs -put <localsrc> ... <dst>

Creating a File

The touchz command is used to create a file of zero length and has the following syntax.

hdfs dfs -touchz URI [URI ...

Changing Permissions of Files, Changing Owner of Files and Directories , Copying Files to the Local Filesystem