JAIN DEEMED-TO-BE UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND IT

Master of Computer Applications /
Master of Science in Information Technology

Subject code – **Fundamentals of Big Data**

MODULE 2

INTRODUCTION TO APACHE HADOOP

Exploring the World of Hadoop

SEARCH ENGINE INNOVATORS LIKE YAHOO! AND GOOGLE NEEDED TO FIND A WAY TO MAKE SENSE OF THE MASSIVE AMOUNTS OF DATA THAT THEIR ENGINES WERE COLLECTING. THESE COMPANIES NEEDED TO BOTH UNDERSTAND WHAT INFORMATION THEY WERE GATHERING AND HOW THEY COULD MONETIZE THAT DATA TO SUPPORT THEIR BUSINESS MODEL. HADOOP WAS DEVELOPED BECAUSE IT REPRESENTED THE MOST PRAGMATIC WAY TO ALLOW COMPANIES TO MANAGE HUGE VOLUMES OF DATA EASILY. HADOOP ALLOWED BIG PROBLEMS TO BE BROKEN DOWN INTO SMALLER ELEMENTS SO THAT ANALYSIS COULD BE DONE QUICKLY AND COST-EFFECTIVELY.

Hadoop Distributed File System: A reliable, high-bandwidth, low-cost, data storage cluster that facilitates the management of related files across machines.

✓ **MapReduce engine:** A high-performance parallel/distributed data processing implementation of the MapReduce algorithm.

The Design of HDFS

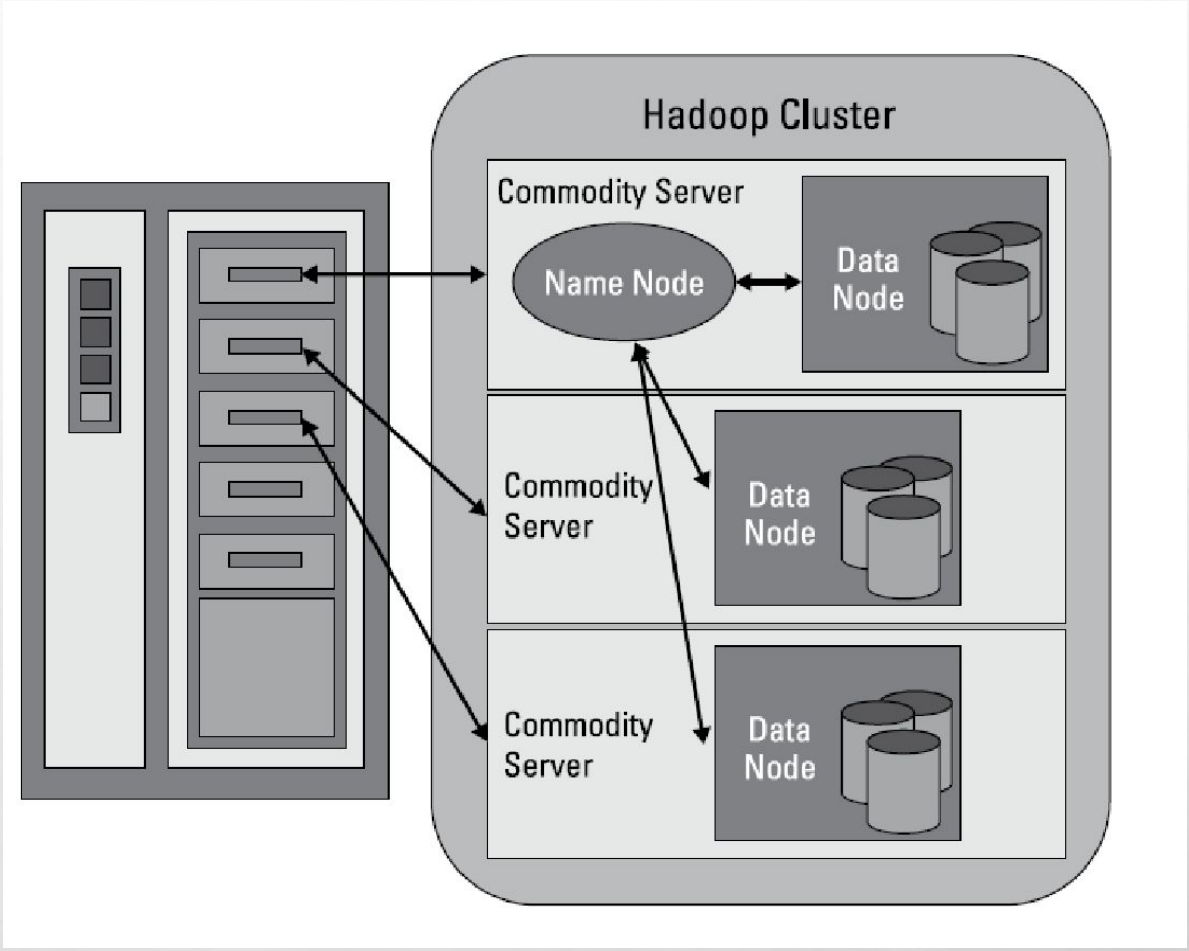
Very large files
Streaming data access
Commodity hardware
Low-latency data access
Lots of small files
Multiple writers, arbitrary file modifications

HDFS Concepts

Blocks

A disk has a block size, which is the minimum amount of data that it can read or write. Filesystems for a single disk build on this by dealing with data in blocks, which are an integral multiple of the disk block size. Filesystem blocks are typically a few kilobytes in size, while disk blocks are normally 512 bytes. This is generally transparent to the filesystem user who is simply reading or writing a file—of whatever length. However, there are tools to perform filesystem maintenance, such as *df* and *fsck*, that operate on the filesystem block level.

Understanding the Hadoop Distributed File System (HDFS)



The Hadoop Distributed File System is a versatile, resilient, clustered approach to managing files in a big data environment. HDFS is not the final destination for files. Rather, it is a data service that offers a unique set of capabilities needed when data volumes and velocity are high. Because the data is written once and then read many times thereafter, rather than the constant read-writes of other file systems, HDFS is an excellent choice for supporting big data analysis. The service includes a "NameNode" and multiple "data nodes" running on a commodity hardware cluster and provides the highest levels of performance when the entire cluster is in the same physical rack in the data center. In essence, the NameNode keeps track of where data is physically stored. Figure 9-1 depicts the basic architecture of HDFS.

NameNodes

HDFS works by breaking large files into smaller pieces called *blocks*. The blocks are stored on data nodes, and it is the responsibility of the NameNode to know what blocks on which data nodes make up the complete file. The NameNode also acts as a "traffic cop," managing all access to the files, including reads, writes, creates, deletes, and replication of data blocks on the data nodes. The complete collection of all the files in the cluster is sometimes referred to as the file system *namespace*. It is the NameNode's job to manage this namespace.

Data nodes

Data nodes are not smart, but they are resilient. Within the HDFS cluster, data blocks are replicated across multiple data nodes and access is managed by the NameNode. The replication mechanism is designed for optimal efficiency when all the nodes of the cluster are collected into a rack. In fact, the NameNode uses a "rack ID" to keep track of the data nodes in the cluster. HDFS clusters are sometimes referred to as being "rack-aware." Data nodes also provide "heartbeat" messages to detect and ensure connectivity between the NameNode and the data nodes. When a heartbeat is no longer present, the NameNode unmaps the data node from the cluster and keeps on operating as though nothing happened. When the heartbeat returns (or a new heartbeat appears), it is added to the cluster transparently with respect to the user or application.

Under the covers of HDFS

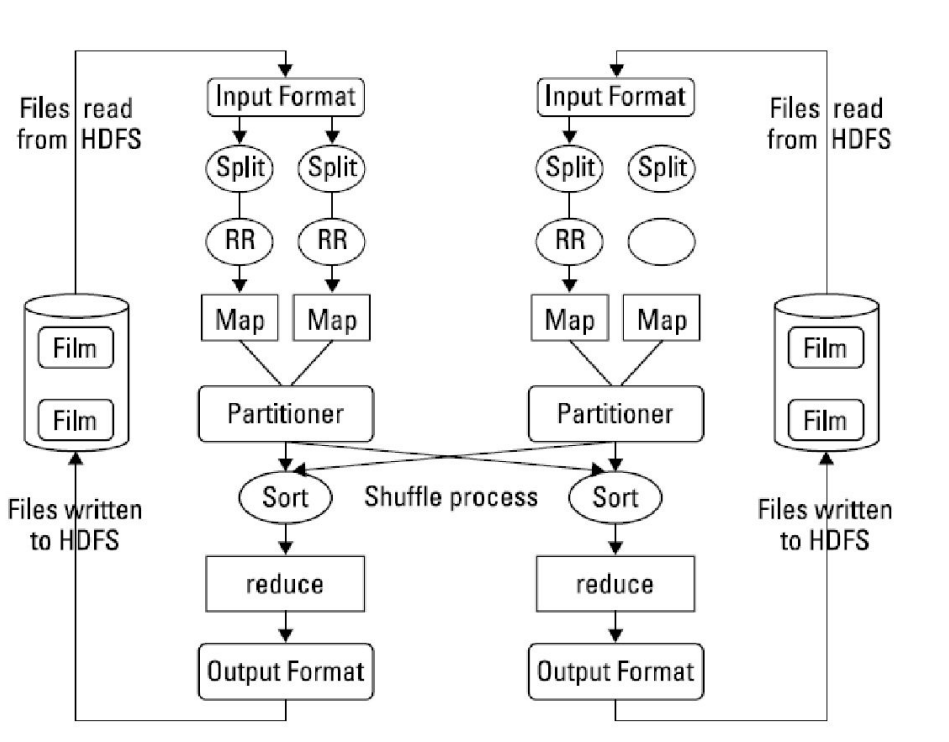
Metadata is defined as "data about data." Software designers have been using metadata for decades under several names like data dictionary, metadata directory, and more recently, tags. Think of HDFS metadata as a template for providing a detailed description of the following:

- When the file was created, accessed, modified, deleted, and so on
- Where the blocks of the file are stored in the cluster
- Who has the rights to view or modify the file
- How many files are stored on the cluster
- How many data nodes exist in the cluster
- The location of the transaction log for the cluster

- Stores (and retrieves) the data blocks in the local file system of the server. HDFS is available on many different operating systems and behaves the same whether on Windows, Mac OS, or Linux.
- Stores the metadata of a block in the local file system based on the meta-data template in the NameNode.
- Performs periodic validations of file checksums.

- Sends regular reports to the NameNode about what blocks are available for file operations.
- Provides metadata and data to clients on demand. HDFS supports direct access to the data nodes from client application programs.
- Forwards data to other data nodes based on a "pipelining" model.

Hadoop MapReduce



Reduce and combine

For each output pair, reduce is called to perform its task. In similar fashion to map, reduce gathers its output while all the tasks are processing. Reduce can't begin until all the mapping is done, and it isn't finished until all instances are complete. The output of reduce is also a key and a value. While this is necessary for reduce to do its work, it may not be the most effective output format for your application. Hadoop provides an OutputFormat feature, and it works very much like InputFormat. OutputFormat takes the key-value pair and organizes the output for writing to HDFS. The last task is to actually write the data to HDFS. This is performed by RecordWriter,