Unit 2

1) What is apache hadoop? Explain hadoop Eco-system

> Apache Hadoop is an open source software platform for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. Hadoop services provide for data storage, data processing, data access, data governance, security, and operations.

Hadoop is a framework that allows for the distributed processing of large datasets across clusters of computers using simple programming models. It is inspired by a technical document published by Google.
The word Hadoop does not have any meaning. Doug Cutting, who discovered Hadoop, named it after his son yellow-colored toy elephant.
Let us discuss how Hadoop resolves the three challenges of the distributed system, such as high chances of system failure, the limit on bandwidth, and programming complexity.
The four key characteristics of Hadoop are:

- Economical: Its systems are highly economical as ordinary computers can be used for data processing.
- Reliable: It is reliable as it stores copies of the data on different machines and is resistant to hardware failure.
- Scalable: It is easily scalable both, horizontally and vertically. A few extra nodes help in scaling up the framework.
- Flexible: It is flexible and you can store as much structured and unstructured data as you need to and decide to use them later.

Hadoop Ecosystem Hadoop has an ecosystem that has evolved from its three core components processing, resource management, and storage. In this topic, you will learn the components of the Hadoop ecosystem and how they perform their roles during Big Data processing. The
Hadoop ecosystem is continuously growing to meet the needs of Big Data. It comprises the following twelve components:

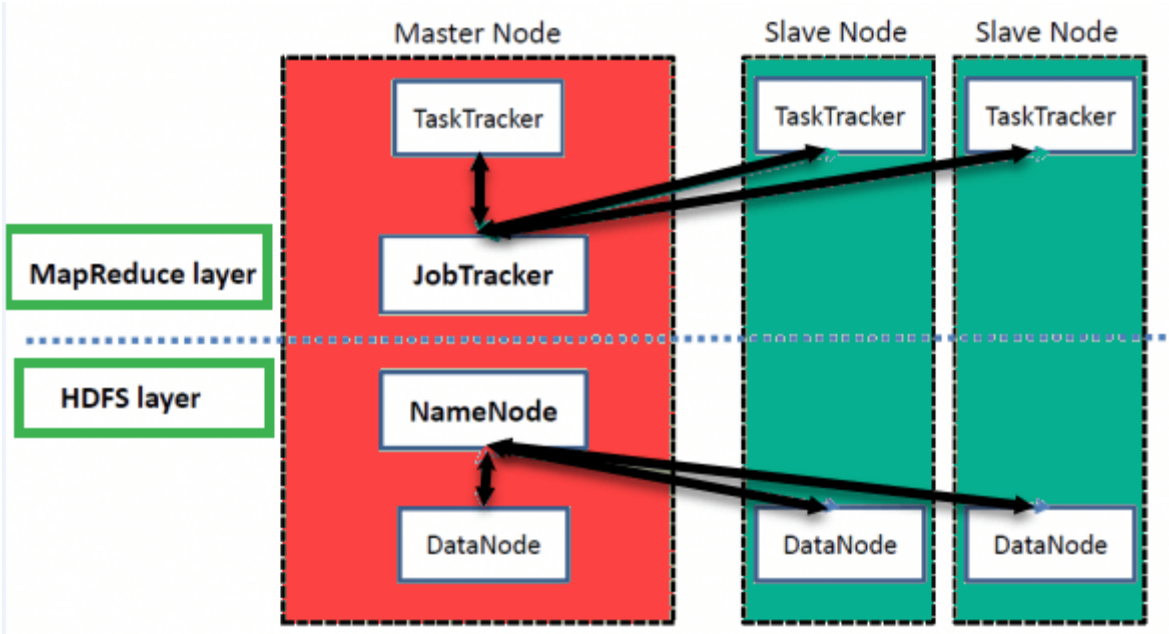
(refer prajval also)

- HDFS(Hadoop Distributed file system)
- HBase
- Sqoop
- Flume
- Spark
- Hadoop MapReduce

- Pig
- Impala
- Hive
- Cloudera Search
- Oozie
- Hue.

2) Explain hadoop framework.

Hadoop Architecture



High Level Hadoop Architecture

Hadoop has a Master-Slave Architecture for data storage and distributed data processing using MapReduce and HDFS methods.

NameNode:

Namecode represented every files and directory which is used in the namespace

DataNode:

DataNode helps you to manage the state of an HDFS node and allows you to interacts with the blocks

MasterNode:

The master node allows you to conduct parallel processing of data using Hadoop MapReduce.

Slave node:

The slave nodes are the additional machines in the Hadoop cluster which allows you to store data to conduct complex calculations. Moreover, all the slave node comes with Task Tracker and a DataNode. This allows you to synchronize the processes with the NameNode and Job Tracker respectively.

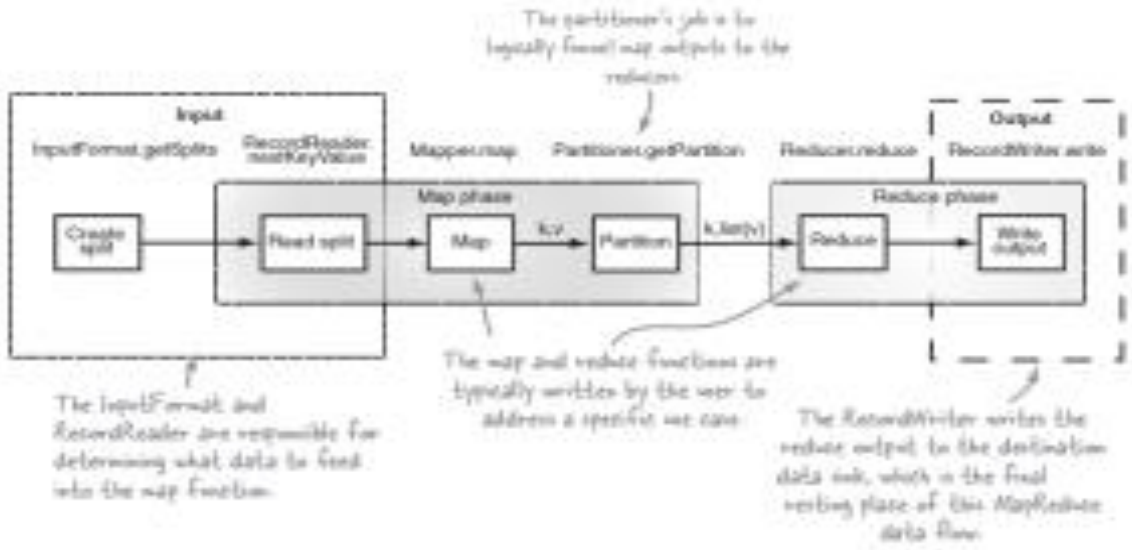
In Hadoop, master or slave system can be set up in the cloud or on-premise

(refere also prajval)

3) Explain inputs and outputs of MapReduce.

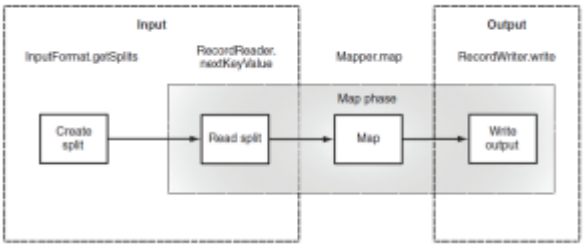
UNDERSTANDING INPUTS AND OUTPUTS IN MAPREDUCE

Your data might be XML files sitting behind a number of FTP servers, text log files sitting on a central web server, or Lucene indexes1 in HDFS. How does MapReduce support reading and writing to these different serialization structures across the various storage mechanisms? You'll need to know the answer in order to support a specific serialization format.



Data input :-

The two classes that support data input in MapReduce are InputFormat and Record-Reader. The InputFormat class is consulted to determine how the input data should be partitioned for the map tasks, and the RecordReader performs the reading of data from the inputs.



INPUTFORMAT :-

Every job in MapReduce must define its inputs according to contracts specified in the InputFormat abstract class. InputFormat implementers must fulfill three contracts: first, they describe type information for map input keys and values; next, they specify how the input data should be partitioned; and finally, they indicate the RecordReader

instance that should read the data from source.

RECORDREADER :-

The RecordReader class is used by MapReduce in the map tasks to read data from an input split and provide each record in the form of a key/value pair for use by mappers. A task is commonly created for each input split, and each task has a single RecordReader that's responsible for reading the data for that input split.

Data output :-

MapReduce uses a similar process for supporting output data as it does for input data.Two classes must exist, an OutputFormat and a RecordWriter. The OutputFormat performs some basic validation of the data sink properties, and the RecordWriter writes each reducer output to the data sink.

OUTPUTFORMAT:-

Much like the InputFormat class, the OutputFormat class, as shown in figure 3.5, defines the contracts that implementers must fulfill, including checking the information related to the job output, providing a RecordWriter, and specifying an output committer, which allows writes to be staged and then made "permanent" upon task and/or job success.

RECORDWRITER:-

You'll use the RecordWriter to write the reducer outputs to the destination data sink.

It's a simple class.

(refere also prajval)

4) Explain Data Serialization.

https://www.tutorialspoint.com/avro/avro_serialization.htm