HADOOP

Start -all.sh

start -dfs.sh

start -yarn.sh

jps

hdfs dfs -mkdir /input

hdfs dfs -put /home/hadoop/Desktop/data.txt /input

hdfs dfs -ls /input/

yarn jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-<version>.jar wordcount /input /output

hdfs dfs -cat /output/* or hdfs dfs -cat /output/part-r-00000

stop -yarn.sh

stop -dfs.sh

Hadoop is an open-source framework that allows distributed processing of large data sets across clusters of computers using simple programming models. It provides a scalable and fault-tolerant solution for processing and storing big data.

The core components of the Hadoop framework are:

1. Hadoop Distributed File System (HDFS): HDFS is a distributed file system that provides high-throughput access to data across a cluster of machines. It is designed to store and manage large datasets by dividing them into blocks and distributing those blocks across multiple machines in the cluster. HDFS provides fault tolerance by replicating data across multiple machines, ensuring data availability even in the presence of node failures.

Data storage Nodes in HDFS.

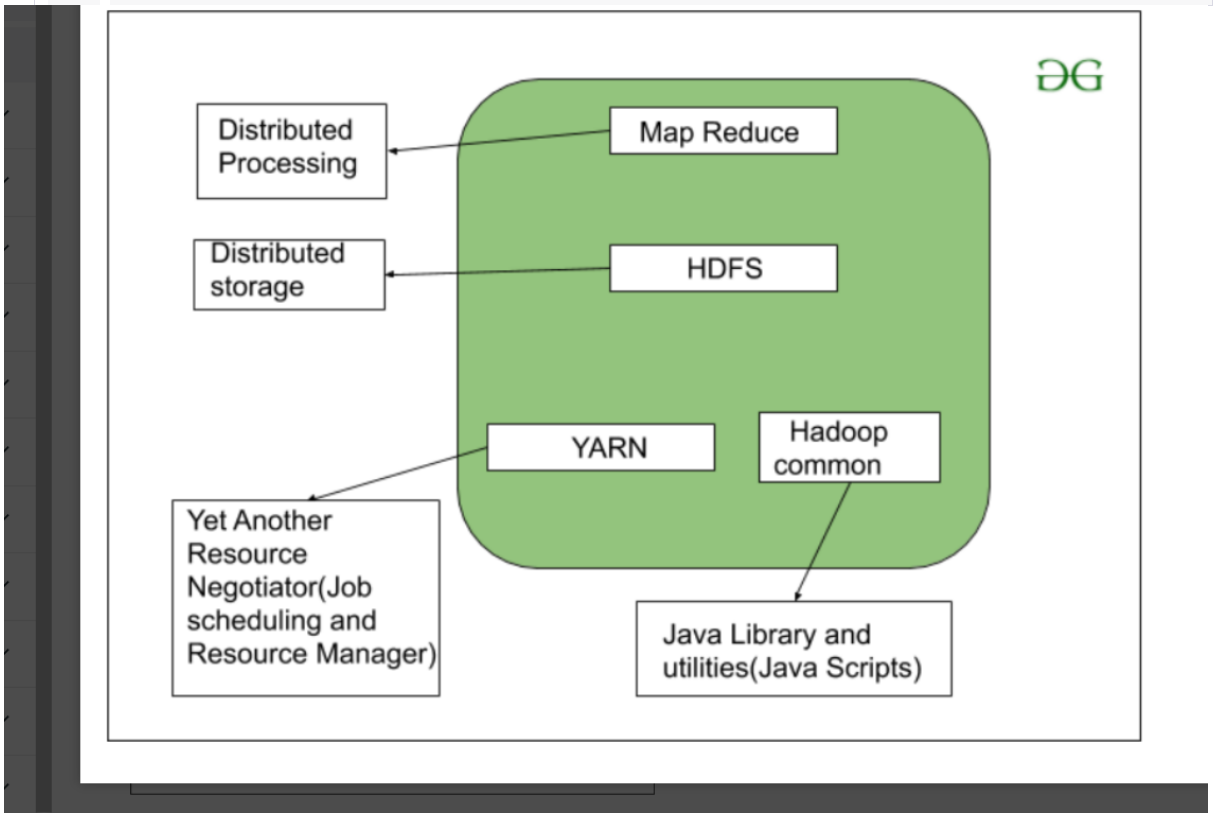
- NameNode(Master)
- DataNode(Slave)

NameNode: NameNode works as a Master in a Hadoop cluster that guides the Datanode(Slaves). Namenode is mainly used for storing the Metadata i.e. the data about the data. Meta Data can be the transaction logs that keep track of the user's activity in a Hadoop cluster.
Meta Data can also be the name of the file, size, and the information about the location(Block number, Block ids) of Datanode that Namenode stores to find the closest DataNode for Faster Communication. Namenode instructs the DataNodes with the operation like delete, create, Replicate, etc.

DataNode: DataNodes works as a Slave DataNodes are mainly utilized for storing the data in a Hadoop cluster, the number of DataNodes can be from 1 to 500 or even more than that. The more number of DataNode, the Hadoop cluster will be able to store more data. So it is advised that the DataNode should have High storing capacity to store a large number of file blocks.

2. MapReduce: MapReduce is a programming model for processing and analyzing large datasets in parallel across a distributed cluster. It simplifies the development of distributed applications by allowing developers to write their logic in two functions: `map()` and `reduce()`. The `map()` function processes input data and produces a set of intermediate key-value pairs, and the `reduce()` function combines the values associated with the same key to produce the final output.
3. YARN (Yet Another Resource Negotiator): YARN is the job scheduling resource management framework in Hadoop. It manages resources in a Hadoop cluster, allocating resources to different applications and managing their execution. YARN allows different data processing engines, such as MapReduce, Apache Spark, and Apache Flink, to run on the same cluster, making Hadoop more versatile for various types of workloads.
4. Hadoop Common: Hadoop Common provides the common utilities and libraries required by other Hadoop components. It includes modules for file I/O, networking, security, and other common functionalities.
5. Hadoop common or Common utilities are nothing but our java library and java files or we can say the java scripts that we need for all the other components present in a Hadoop cluster. these utilities are used by HDFS, YARN, and MapReduce for running the cluster. Hadoop Common verify that Hardware failure in a Hadoop cluster is common so it needs to be solved automatically in software by Hadoop

Framework.



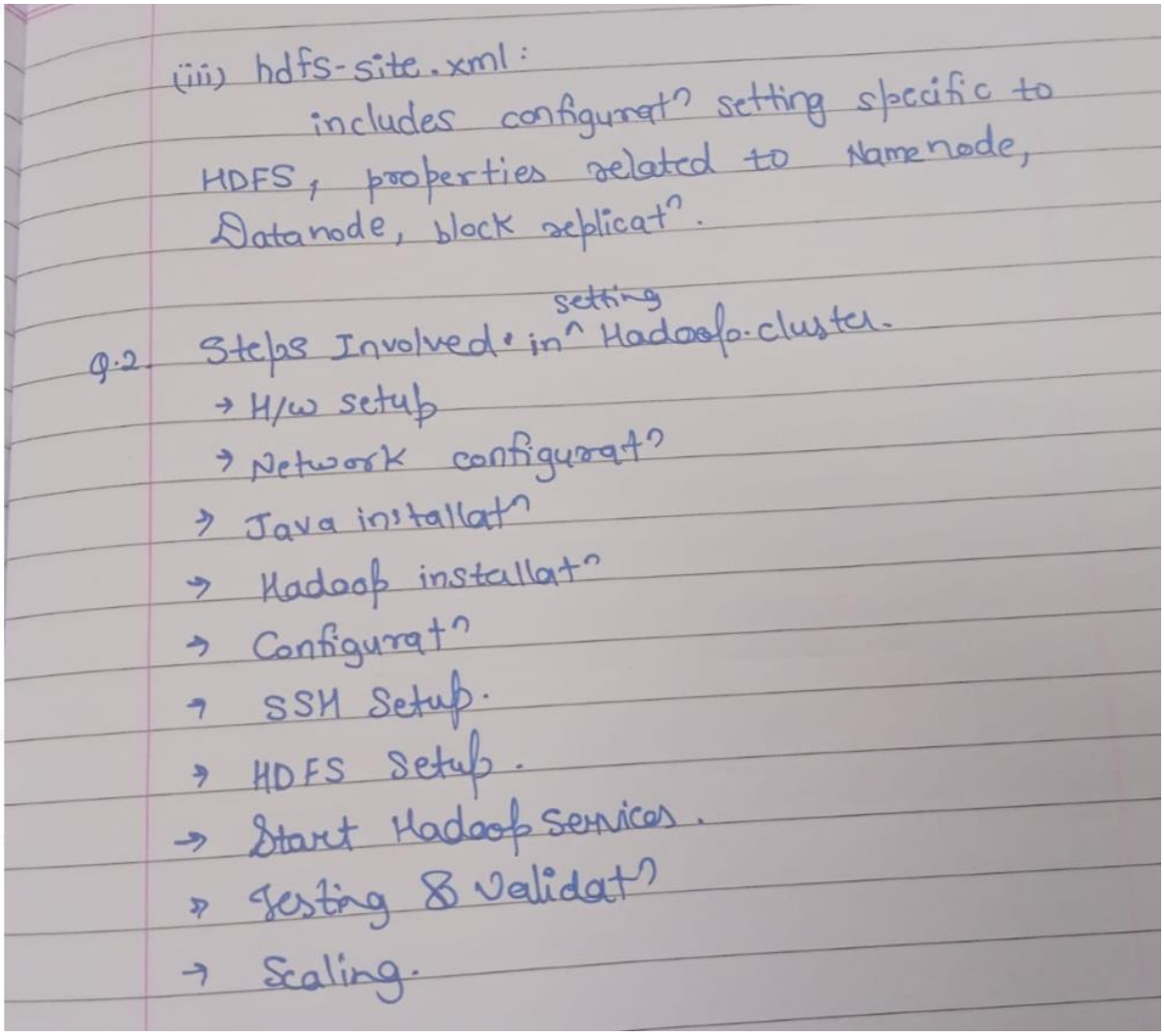
Which configuration files are required for setting the hadoop environment (mapred-site.xml, core-site.xml, hdfs-site.xml

Q.1

→ To setup Hadoop environment need to configure three important XML files :

i) mapred-site.xml - Contains configuration settings specific to MapReduce framework. Includes properties related to job execution, task distribution, task tracking.

ii) core-site.xml - includes core Hadoop configuration settings shared across various components., Properties related to HDFS such as default filesystem URI, cluster's name, locat^n of Namenode.



SPARK

Localhost:4040 for spark
Sc spark context entry point for spark compiler to start processing code

SparK:
1.bin/spark-shell
2. var text=sc.textFile(“/home/test/input.txt”)
3. var counts=text.flatMap(line=>line.split(“ “))
4. val mapf=counts.map(word=>(word,1))
5. val reducef=mapf.reduceByKey(_+_) //group by step
6. reducef.collect

2-5 are transformations
Only after action (6) output is available