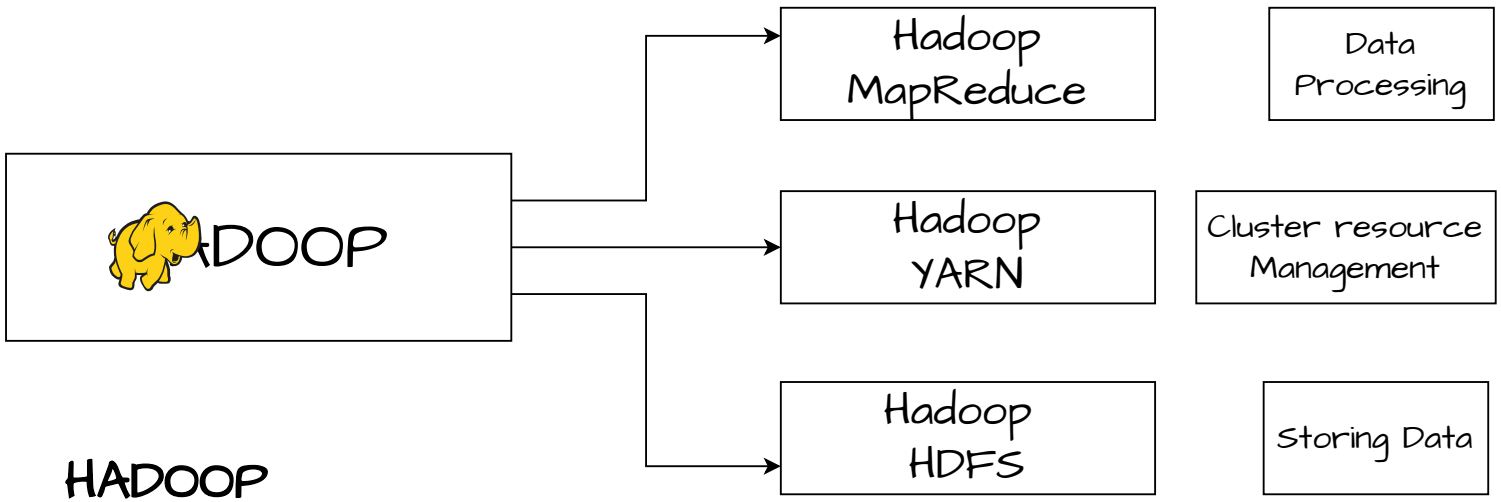
HADOOP



HADOOP

Hadoop is a open-source framewor designed for storing and processing big data efficiently. It allows you to distribute large datasets across clusters of computers, enabling parallel processing for faster analysis. Instead of relying on a single powerful machine, Hadoop leverages the combined resources of multiple machines within the cluster

Links

Hadoop Architecture
Hadoop Architecture 2.0
Hadoop Simplilearn
Hadoop Quiz
Hadoop PLaylist

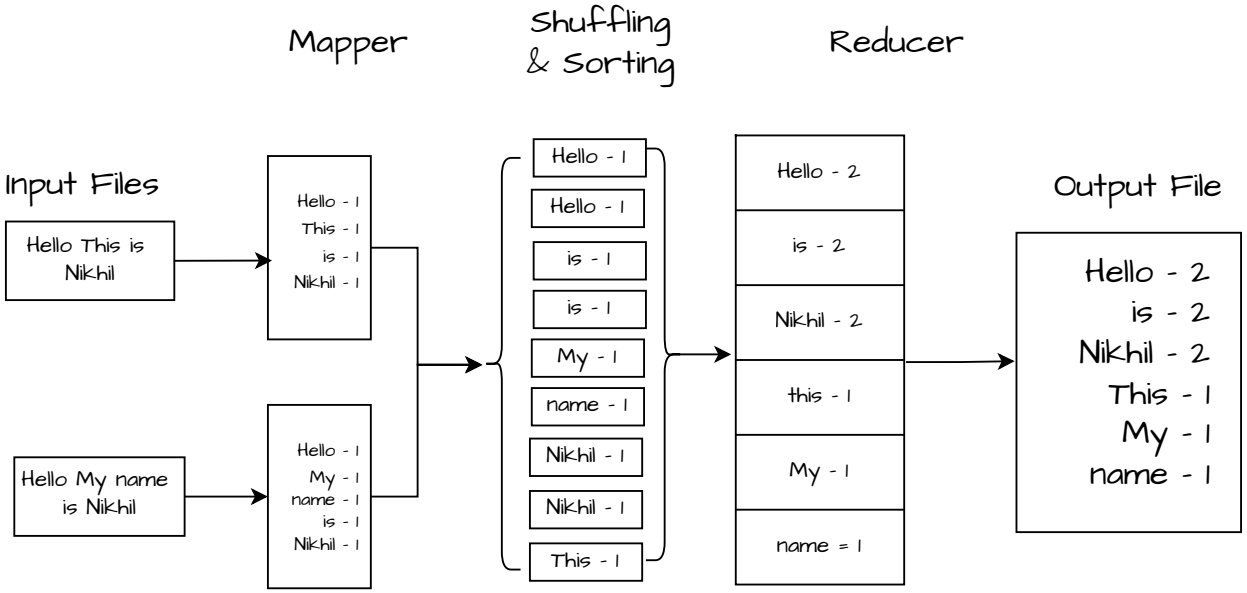
HDFS

Hadoop Distributed File system(HDFS) is the storage layer of Hadoop that stores data in multiple data server..

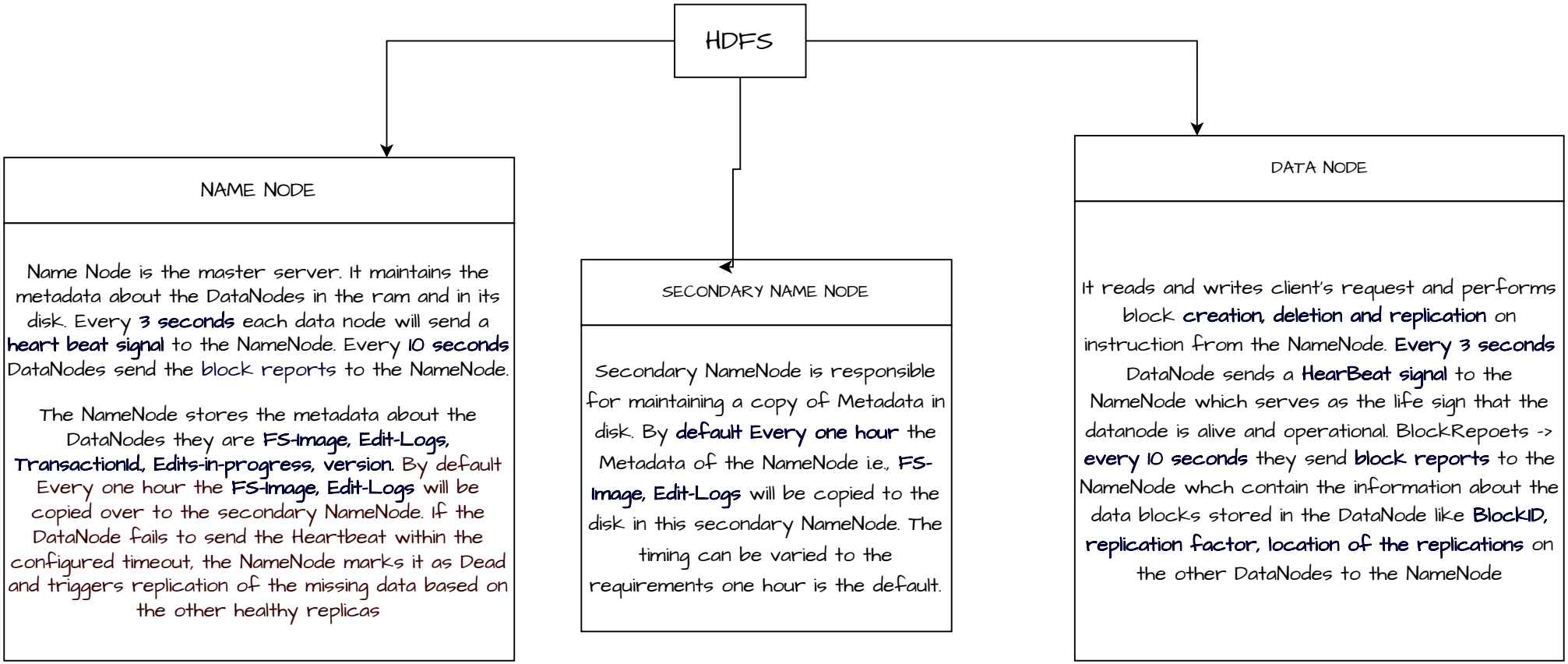
Data is divided into multiple blocks, the default size of the block in Hadoop 2.0 is 128Mb, which was increased from 64Mb in the previous version. Hadoop is based on a leader/follower architecture. Each cluster is typically composed of a single NameNode, an optimal SecondaryNameNode, and an arbitary

Map Reduce

MapReduce is a framework tailor-made for processing large datasets in a distributed fashion across multiple machines.
The core of MapReduce job can be, err, reduced to three operations: map an input data set into collection of <key, value> pairs, shuffle the resulting data(transfer data to the reducers), then reduce over all pairs with the same key.
The top level unit of work in MapReduce is a job. Each job is composed of one or more map or reduce tasks
In earlier versions of Hadoop(pre 2.0), MapReduce took care of its own resource allocation and job scheduling as well as the actual computation.
Newer versions of Hadoop(2.0+) decouple the scheduling from the computation with YARN, which handles the allocation of computational resources for MapReduce Jobs. This allows other processing frameworks to share the cluster without resource constraint.



MapReduce



QJM is the preferred approach for High availability. It used Zookeeper, a distributed coordination service, to manage the namespace edits and facilitates failover.

YARN

Yet Another Resource Negotiator is the framework for \assigning computational resources for application execution.
Yarn consists of Three core components:
- ResourceManger (one per cluster) - ApplicationMaster(One per application) - NodeManger (One per node)

Resource Manger :

It is responsibel for talking to inventory of available resouces and runs serveral critical services, the most important of which is scheduler. It is a pure scheduler in that it does not monitor or track application status or progress. As it performs no monitoring it cannot gurantee that tasks will restart should they fail the execution of an application over its full lifespan. from requesting additional containers from the ResourceManger, to submitting release requests to the NodeManager

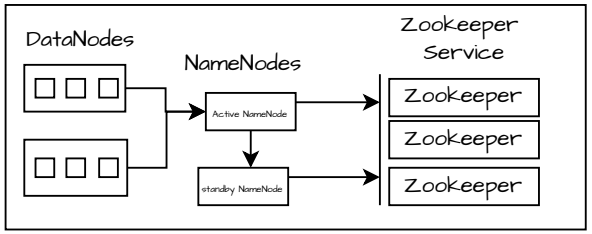
NodeMangers :

The NodeManger is a per-node agent tasked with overseeing containers throughout their lifecycles, monitoring container resource usage, and periodically communicating with the resource Manger

Execution flow in Yarn:

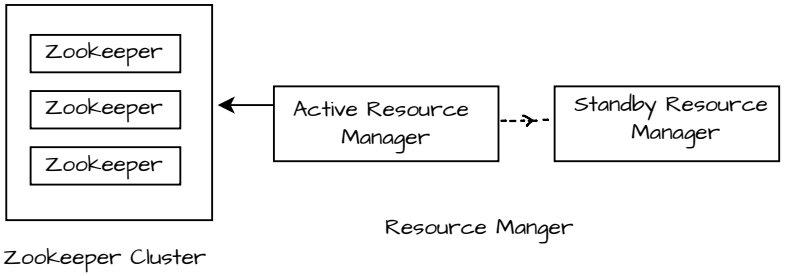
* client program submits the MapReduce application to the ResourceManger, along with the info to launch the application specific ApplicationMaster.
* ResourceManger negotiates a container for the ApplicationMaster and launches the ApplicationMaster.
* Application Master boots and registers with the ResourceManger, the original calling client to interface directly with theApplicationMaster
*ApplicationMaster negotiates resources for client application
* ApplicationMaster gives the container launch specification
to the NodeManager, which launches a container for
application.
* During execution, client polls ApplicationMaster for application status and progress.
* Upon completion, ApplicationMaster deregisters with the ResourceManager and shutsdown, returning its container to the resource Pool.

Zookeeper



Apache Zookeeper is a popular tool used for coordination and sychronization of distributed systems. Since Hadoop 2.0, Zookeeper has become an essential service for Hadoop clusters, providing a mechanism for enabling high-availability of fomer single points of failure, specifically the HDFSNameNode and YARN ResourceManager
PROBLEM
In Previous versions of Hadoop, the NameNode represented a single point of failuer-should the NameNode fail, the Entire HDFS cluster would beoome unavailable as the metadata containing the file-to-block mappings would be lost
SOL:
Hadoop .20 brought many imporovements, among them a high-availability NameNode Service. when Zookeeper is used in conjunction with QJM or NFS, it enables automatic failover
Automatic NameNode failover requires two components a zookeeper quorum, and ZKFailoverController (ZKFC) process running on each NameNode. The NameNode and standby NameNodes maintain persistent sessions in Zookeeper, with the NameNode holding a special, ephemeral "lock" ZNode; if the NameNode doesnot maintain contact with Zookeeper ensemble, its session is expired, triggering a failover. ZKFailoverController is a process that runs alongside the NameNode and StandBy NameNodes, periodically checking the health of the node it is running On Healthy nodes, ZKFC will try to acquire the lock ZNode, succeding if no other node holds the lock(which means the Primary NameNode has failed). Once the lock is acquired, the new NameNode transitions to the active NameNode the lock file (Znode) is the key part there will be only one lock file it will check for the lock file among all the NameNodes which ever NameNode has the lock file will be the Primary NameNode (leader), Zookeeper will check for the lock file always in the case of the primary NameNode fails the lock file will not be available in contact with the Zookeeper Ensemble so it will trigger the ZKFC controller the ZKFC will be true when no node has the lock file.

YARN AND ZooKeeper



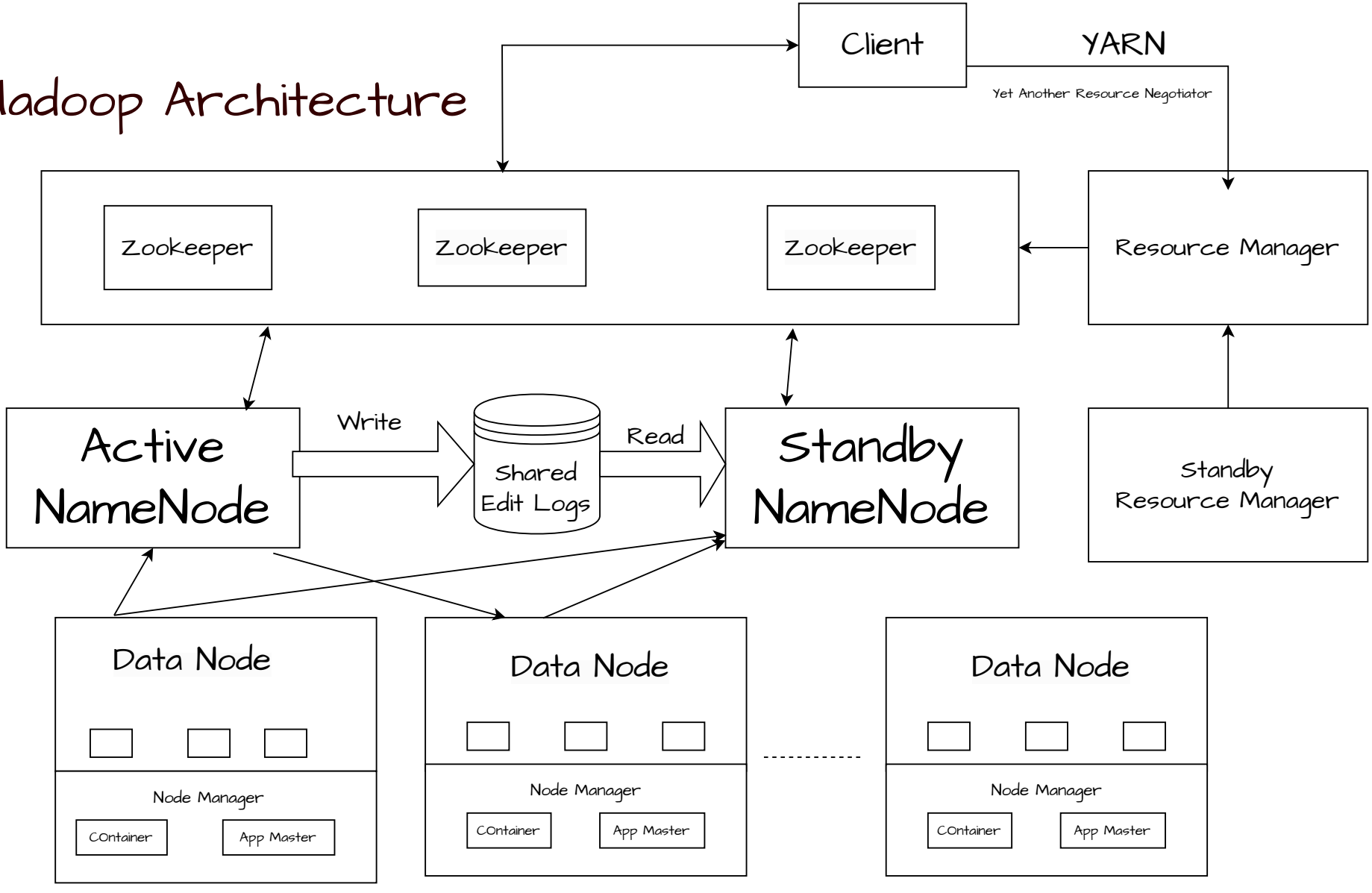
When YARN was initially created, its Resource Manager represented a single point of failure if NodeManager lost Contact with ResourceManager, all jobs in progress would be halted, and no new jobs could be assigned Hadoop 2 improved YARN'S resiliance with the release of the Resource Manger high-availability feature. The new feature incorporates Zookeeper to allow for automatic failover to standby ResourceManager in the event of primary's failuer Like HDFS, YARN uses a smiliar, Zookeeper-manager lock to ensure only one ResourceManager is active at once.Unlke HDFS, YARN's automatic failover mechanism does not run as a seperate process, its ActiveStandbyElector service is a part of the ResourceManager process itself. ZKFailoverController, the ActiveStandbyElector service on ResourceManager continuously seeks for control of the ZNode's ActiveStandbyElectorLock. if the currelntly active ResourceManager allows the to expire ie out of usage (dead) the ResourceManager that successfully acquires the lock on the ActiveStandbyElectorLock will automatically be promoted to active State Scheduler allocates resources to various running applications

* Scheduler resources based on the requierments of the applications.
* Does not monitor or track the status of the applications
* Application Manager accepts job submissions
* Monitors and restarts application masters in case of failure

There are three different types of schedulers available in YARN they are

* **FIFO scheduler**: The FIFO scheduler places applications in a queue and runs them in the order of submissions(first in first out)
* **Capacity Scheduler**: A seperate dedicated queue allows the small job to start as soon as it is submitted. The large job finishes later than when using the FIFO scheduler
* **Fair Scheduler**: There is no need to reserve a set amount of capacity, since it will dynamically balance resources between all the running jobs.

Hadoop Architecture



IMPORTANT QUSTIONS ABOUT HADOOP

1. What is Hadoop?

-> Hadoop is a distributed framework which has distriibuted storage and distributed processing mechanism. It uses HDFS for storage and MapReduce for distributed processing. Hadoop is Highly durable and easily scalable.

2. What are the component of Hadoop?

-> Hadoop architecture consist of NameNode, StandbyNameNode, DataNodes, JournalNodes, Zookeeper and YARN

3. Block size in HDFS?

-> The default block size in HDFS 2.0 is **128MB** which has been inreased from 64MB in the previous versions 1 of Hadoop.

4.What is fault Tolerence in Hadoop?

-> Fault tolerence is the ability of the HDFS system to withstand hardware failures and continue operating without data loss. Hadoop has replication factor of 3 in which each block of data gets replicated 3 times, in case of any node failure (faults) it can be handled.

5. Safe Mode in Hadoop?

-> When NameNode gets restarted. To create FS Image (To server client) name node goes to safe mode and does checkpointing with Edit log and FS Image in the standbyNode only after the checkpointing is over it leaves the safe mode. All of this the NameNode is not in service and client cannot access the cluster.

6. What is Heartbeat and its Interval?

-> DataNodes in the cluster send a signal to the NameNode or YARN which acts like a heatbeat like pulse, indicating that the DataNode is alive and functioning. It indicates the Liveness of the DataNode. The interaval of time is 3 seconds every three seconds the DataNode sends this hearbeat signal to the NameNode. if the NameNode doesnt recieve any signal from the DataNode it will wait for 10 min by default after that that particular DataNode is deemed as a failed Node.

7. What is FS Image and Edit Log in Hadoop?

-> FS Image in NameNode it stores the Meta information about every transaction and location of files in the DataNode, Edit log is a temporary file which helps the checkpointing for every 1 hour by default. Secondary NameNode helps for checkpointing.

8. Under and Over replication in Hadoop?

-> As default replication factor is 3. If any of the Node goes down, The blocks in that Node has only two replication which is less tha 3 we call this as under replication. In some cases the Node that has failed will reconnect to the cluster but with the same data after some time but by then HDFS would have already filled up the missing replication in such cases the HDFS would have four copies of the Data which is called as Over replication in such cases the NameNode will instruct to the Node that has reconnected to delete all the data within it.

9. What is Split Brain Scenario in Hadoop?

-> If dead NameNode becomes active and starts writing to the edit logs in the journalNode again and also when the recent active NameNode does the same then this scenario is called as Split Brain scenario

10. What is Speculative Execution in Hadoop?

-> In Hadoop during Speculative Execution, a certain number of duplicate tasks are launched. On a different DataNode multiple copies of the same mapReduce job can be executed which ever Node finishes the job first would be sent as the result. This is particularly helpful when in some cases one DataNode is slower to respond than the other DataNodes.

11. What is RackAwareness in Hadoop?

-> Rack awareness in Hadoop is the concept to choose a nearby DataNode there by reducing the network traffic. Hadoop supports the configuration of rack awareness to ensure the placement of one replica of the Data Block on a differenet rack.

12. Small file issues in Hadoop?

-> Storing lot of small files which are smaller than the block size cannot be effeciently handled by HDFS. Reading through small files involve lotsof seeks and lots of hopping between the DataNodes , and also creates a lot of metaData in the NameNode, which is inturn inefficient data processing.

8. If you have an inputfile of 350MB, how many input splits will be created by HDFS and what is the size of each input split?

-> * Each block by default is divided into 128MB

* The size of all blocks except the last block will be 128MB * So, There are 3 input splits in total * The size of each split is 128MB, 128MB, and 94MB

9. How does Rack Awareness work in HDFS? ##

-> HDFS Rack Awareness is about having knowledge of different data nodes and how it is distributed across the racks of a HadoopCluster. The rule of replication of HDFS racks is you will not have all the replicas on the same Node.

10. What is distributed CACHE in MapReduce?

-> It is a meachanism supported by the Hadoop MapReduce framwork. The Data Coming from the disk can be cached and made available for all the worker nodes where the map/reduce tasks are running for a given job.

11. What are the benefits of YARN and how does it solve the issues of MapReduce

-> Benefits:

* Scalability * Compatibility * Resource Utilization * Multitenancy(many operations other than MapReduce)

MapReduce v1 Issues:

* Scalability * Availability * Resource Utilization * Can't run non-MapReduce Jobs

1. What are the different vendor specific distributions of Hadoopp?

-> Cloudera, HortonWorks, AWS EMR, IBM infoshpere, Mapr, Microsoft Azure

2. What are the different Hadoop config files?

-> hadoop-env.sh, mapred-set.xml, core-site.xml, yarn-site.xml, hdfs-site.xml, Master and slaves

3. what are the Three modes in Hadoop can run ?

-> Standard Mode

Psuedo-distributed mode

Fully distributed mode

4. What are the differences between regular file system and HDFS?

-> Regular File system	HDFS
Data is maintained in a single system	* Data is distributed and maintaned on multiple systems
*If the machine crashes, data recovery is very difficult due to low fault tolerence	* If a data node crashes, data can still be recovered from the other nodes in the cluster
*Seek time is more and hence it takes more time to process the data.	* Time taken to read data is comparatively more as there is local data read to disc and coordination of data from multiple systems.

5. Why is HDFS fault Tolerent?

-> HDFS is fault tolerent as it replicates data on different dataNodes. By default, a block of data gets replicated on 3 DataNodes

6. what are the two types of metadata a NameNode server holds?

-> MetaData in disk = EditLog, FSImage

MetaData in RAM = Name, replicas, file paths, replication factor

HDFS Federation	HDFS High Availability
* There is no limitation to the number of namenodes and the namenodes are not related to each other.	* There are 2 NameNodes which are related to each other. Both active and standby NameNodes work all the time
* All the namenodes share a pool of metadata will have its dedicated pool	* At a time, active NameNode will be up and running while standby NameNode will be idle and updating it's metaData once in a while
* Provides fault tolerence i.e., if one NameNode goes down, that will not affect the data of the other NameNode	* Requires two seperate machines, On First, the active NameNode will be coinfigured while the secondary NameNode will be configured on the other system.