Hadoop 1.0 Architecture

Distributed Computing and Hadoop

- Hadoop follows a distributed computation model—it distributes computations involving humongous chunks of data sets to a set of nodes, each of which works on a portion of the data set. Before I get into the nitty-gritty of Hadoop architecture itself, let me take a moment to explain the challenges posed by distributed computation and how Hadoop meets them. At its core, distributed computing seeks to meet the following requirements:
- Scalability: Increasing the number of machines should result in a linear increase in processing capacity and storage.
- Fault tolerance: If one of the nodes in a distributed cluster fails, the main computational process itself shouldn't fail or be adversely affected.
- Recoverability: If a job or a part of it fails, you shouldn't lose any data.

Hadoop has been explicitly and carefully designed to meet these fundamental requirements of distributed computing. It meets the challenges of distributed computing through the following strategies and principles that underlie its architecture:

- Data is stored on all or most of the cluster's nodes. Bringing code to the data and not the other way around, Hadoop efficiently processes large amounts of data.
- Developers focus on the data and their algorithms, with Hadoop taking care of the low-level details of distributed programming.
- Jobs are highly tolerant of failures. If one or more nodes of a cluster fail or a component of a job (called a task) fails, the job itself will continue to completion.

Hadoop Cluster.

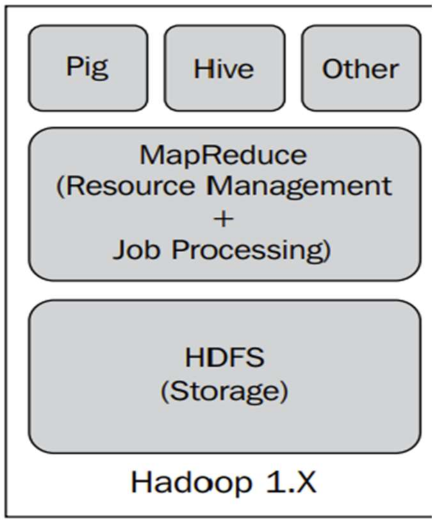
- To summarize, a Hadoop cluster is a collection of machines that use the Hadoop software on the foundations of a distributed file system (HDFS) and a cluster resource manager (YARN). Anything more than a single machine will technically constitute a cluster. You can run small clusters with just a few nodes and there are very large clusters maintained by organizations such as Yahoo, whose largest Hadoop clusters range over 10,000 nodes. Regardless of its size, everything works the same in every Hadoop cluster.
- ✓ **In practical terms, Hadoop implements storage and processing through a set of daemon processes that run in the background. Users aren't concerned with these processes, as they perform input/output over the network without any intervention from users. On a Linux system, each of these daemons (processes) runs within a separate Java Virtual Machine (JVM).**

Master and Worker Nodes: -

The nodes in a Hadoop cluster are classified into two basic types:

- Master nodes: These nodes run the services that coordinate the cluster's work. Clients contact the master nodes to perform computations. In each cluster, there are a handful of master nodes, ranging from three to six, depending on the size of the cluster.

- Worker nodes: These nodes perform under the direction of processes running on the master nodes. Most of a cluster's nodes are worker nodes. The worker nodes are where the data is stored, and computations are performed



HDFS Unique Features

HDFS has several unique features that make it ideal for large-scale distributed processing. In the following sections, I briefly review how HDFS supports the efficient processing of large data sets.

Handling Large Data Sets

Typically, non-Hadoop databases are small, with at most a few terabytes of data and a few data files. Hadoop deals with petabytes of data and thousands of data files.

Fault Tolerance

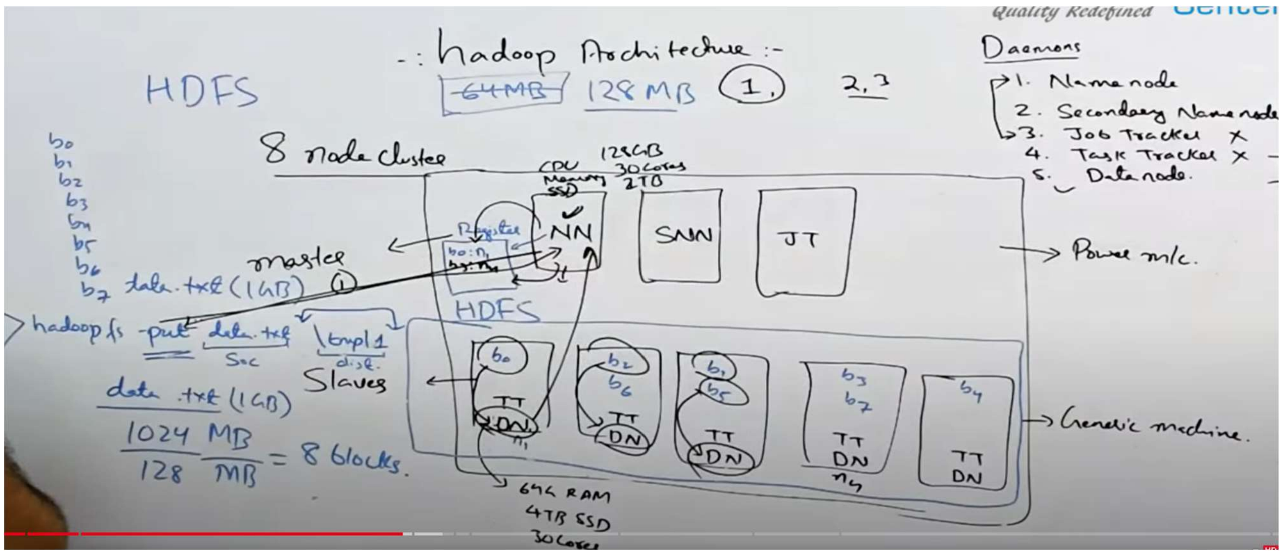
Hadoop depends on large numbers of servers so it can parallelize work across them. Server and storage failures are to be expected, and the system isn't affected by the non-functioning storage units—or even failed servers. Data is, by default, replicated thrice in Hadoop, meaning that each data block in HDFS is stored on three different nodes. You can decrease or increase the default "replication factor." You can also employ different replication levels for different sets of data, as replication is applied at the file level.′

Streaming Access to Data

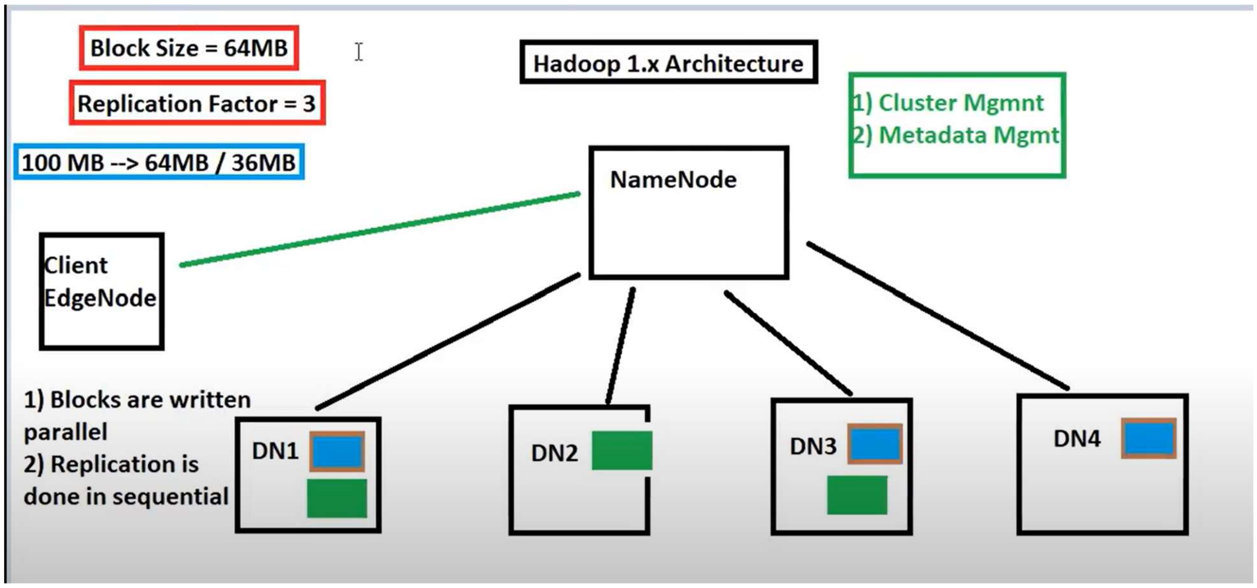
Traditional databases are geared mostly toward fast access to data and not for batch processing. Hadoop was originally designed for batch processing (although newer developments have enabled other processing paradigms such as interactive SQL, iterative processing, search processing, and stream processing) and provides streaming access to data sets.

Simple Data Consistency Model

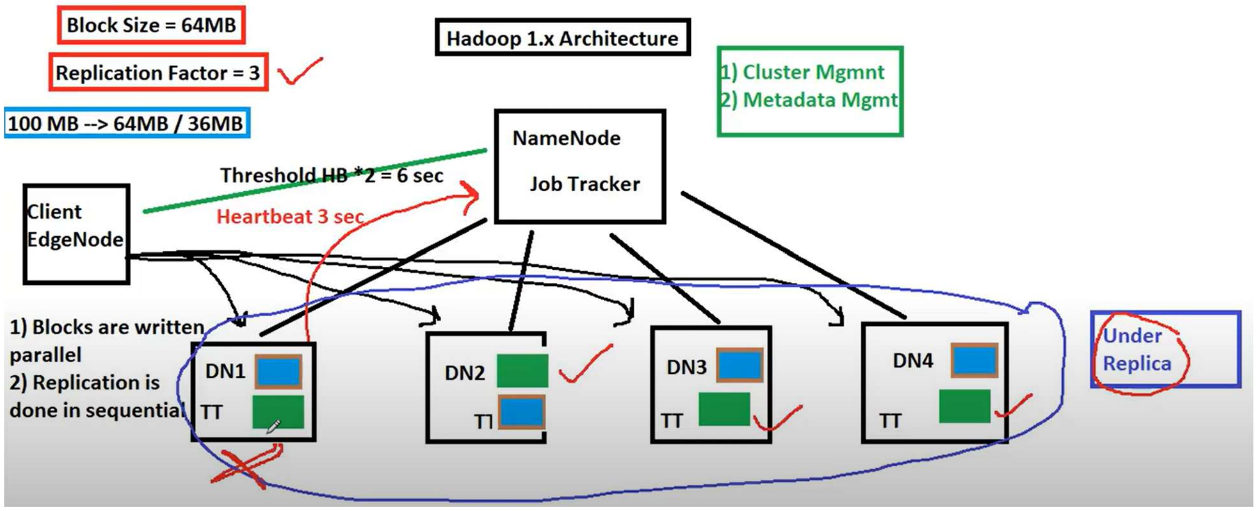
Unlike traditional databases, Hadoop data files employ a write-once-read-many access model. Data consistency issues that may arise in an updateable database aren't an issue with Hadoop file systems because only a single writer can write to a file at any time.



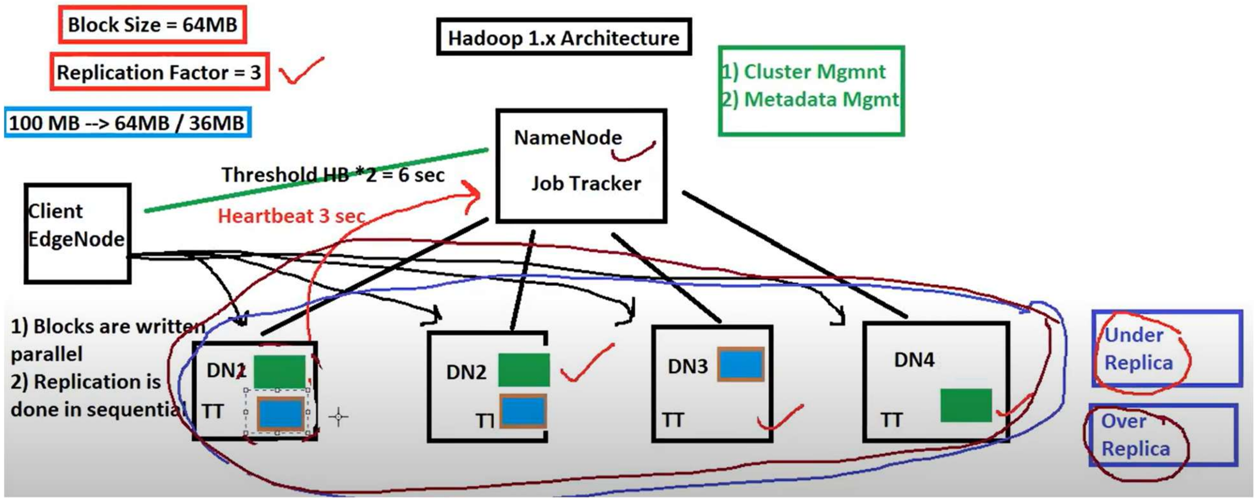
Replication in Hadoop: -



After segregation(name node and data node =HDFS) for computational purposes(JT and TT) and if the data node is down the name node tries to maintain the replication factor.

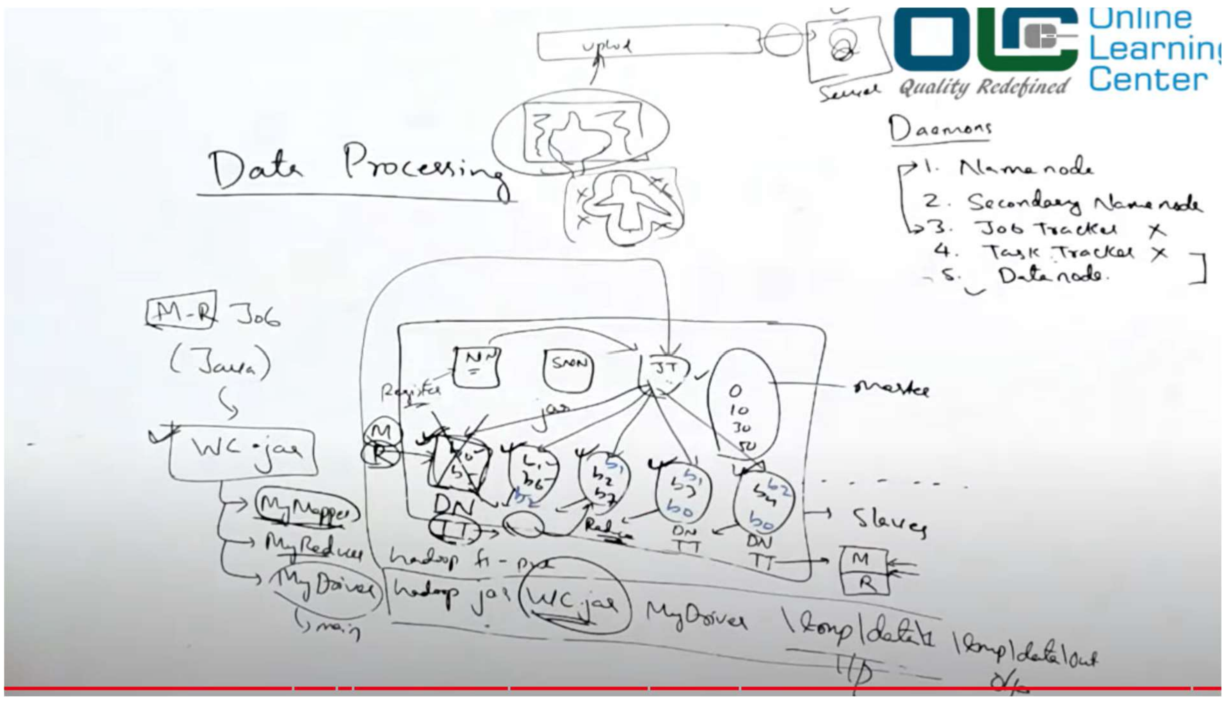


Data node after 10 minutes if it's up then the name node tells to erase the data it has and asks to pick up some work from other data nodes and assist them in completing instead of sitting idle.

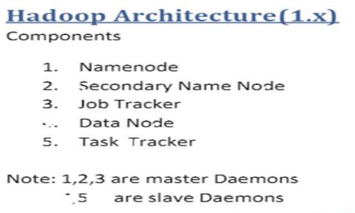


This is all bout storage of how the HDFS file system works.

Data Processing by Job Tracker and Task tracker



Important points to remember: -

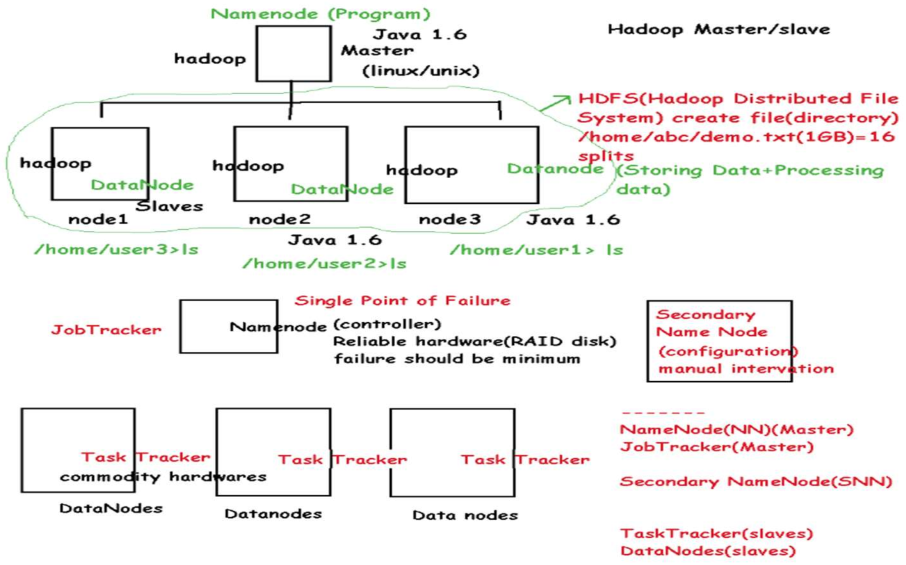


Hadoop Architecture(1.x)

Components

1. Namenode
2. Secondary Name Node
3. Job Tracker
4. Data Node
5. Task Tracker

Note: 1,2,3 are master Daemons
4,5 are slave Daemons



N ..ne Node.
https://hadoo ..apache.org/doc /r3.3.0/hadoop- rcje t-dist/hadoo -hd /. disDesig .ht nl

The Name Node is the master piece of an HDFS file system. It keeps the directory tree of all files in the file system, and tracks where across the cl. ster the file data is kept. It does not store the data of these files itself.

Client applications talk to the Name Node ..henever they ..ish to locate a file, or ..hen they want to add/copy/move/delete a file. The Name Node responds the successful requests by returning a list of relevant Data Node servers where the data lives

The Name Node is a Single Point of Failure for the HDFS Cluster. HDFS is not currently a High Availability system in Hadoop1, but they are highly available in Hadoop 2 and 3. When the Name Node goes down, the file system goes offline. here is an optional Secondary Name Node that can be hosted on a separate machine It only creates checkpoints of the namespace by merging the edits file into the fsimage file and does not provide any real backt p.

Best Practices for Namenode ns allation

It is essential to look after the Name Node. Here are some recommendations from production use

1. Use a good server with lots of RAM. The more RAM you have, the bigger the file system, or the smaller the block size.
2. Use ECC(Error correcting code) RAM.
3. On Java6u15 or later, run the server VM with compressed pointers -XX:+UseCompressedOops to cut the JVM heap size down.
4. List more than one name node directory in the configuration, so that multiple copies of the file system meta-data will be stored. As long as the directories are on separate disks, a single disk failure will not corrupt the meta-data.
5. Configure the Name Node to store one set of transaction logs on a separate disk from the image.
6. Configure the Name Node to store another set of transaction logs to a network mounted disk.
7. Monitor the disk space available to the Name Node. If free space is getting low, add more storage.
8. Do not host Data Node, Job Tracker or Task Tracker services on the same system.

Secondary Name Node

The name node stores the HDFS filesystem information in a file named fsimage. Updates to the file system (add/remove blocks) are not updating the fsimage file, but instead are logged into a file, so the I/O is fast append only streaming as opposed to random file writes. When restarting, the name node reads the fsimage and then applies all the changes from the log file to bring the filesystem state up to date in memory. This process takes time.

The secondary namenode job is not to be a secondary to the name node, but only to periodically read the filesystem changes log and apply them into the fsimage file, thus bringing it up to date. This allows the name node to start up faster next time

Unfortunately, the secondary namenode service is not a standby secondary namenode, despite its name Specifically, it does not offer High Availability for the namenode.

<u>Data Node</u>

The data node is where the actual data resides.

1. All data nodes send a heartbeat message to the namenode every 3 seconds to say that they are alive. If the namenode does not receive a heartbeat from a particular data node for 10 minutes, then it considers that data node to be dead/out of service and initiates replication of blocks which were hosted on that data node to be hosted on some other data node.

   You can set the heartbeat interval in the **hdfs-site.xml** file by configuring the parameter **dfs.heartbeat.interval**

   More info here: https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml

2. The data nodes can talk to each other to rebalance data, move and copy data around and keep the replication high.
3. When the data node stores a block of information, it maintains a checksum for it as well. The data nodes update the namenode with the block information periodically and before updating verify the checksums. If the checksum is incorrect for a particular block i.e. there is a disk level corruption for that block, it skips that block while reporting the block information to the namenode. In this way, namenode is aware of the disk level corruption on that data node and takes steps accordingly.

<u>Job Tracker</u>

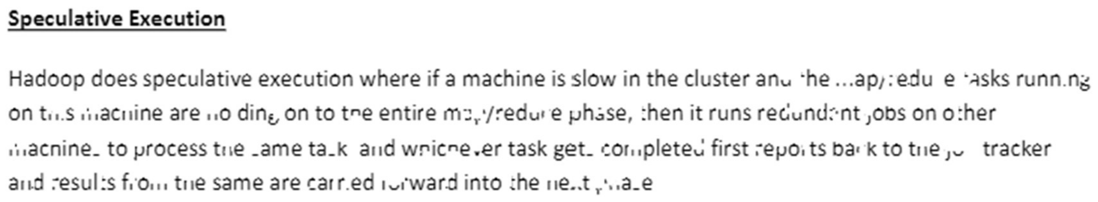
The primary function of the job tracker is resource management (managing the task trackers), tracking resource availability and task life cycle management (tracking its progress, fault tolerance etc.)

<u>TaskTracker</u>

The task tracker has a simple function of following the orders of the job tracker and updating the job tracker with its progress status periodically.

The task tracker is pre-configured with a number of slots indicating the number of tasks it can accept. When the job tracker tries to schedule a task, it looks for an empty slot in the task tracker running on the same server which hosts the data node where the data for that task resides. If not found, it looks for the machine in the same rack. There is no consideration of system load during this allocation.

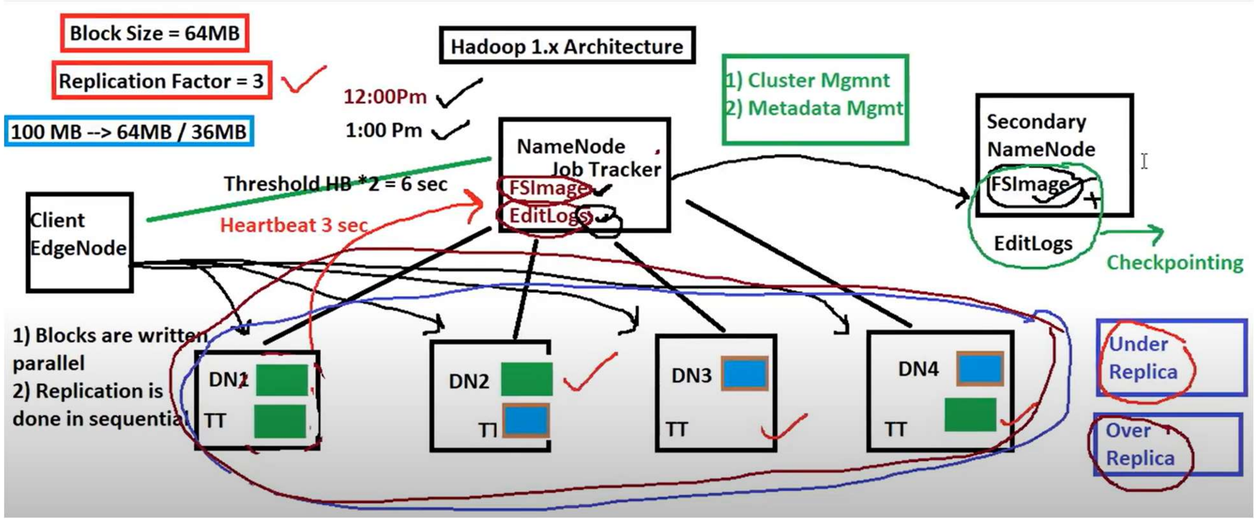
HDFS is rack aware in the sense that the namenode and the job tracker obtain a list of rack ids corresponding to each of the slave nodes (data nodes) and creates a mapping between the IP address and the rack id. HDFS uses this knowledge to replicate data across different racks so that data is not lost in the event of a complete rack power outage or switch failure.

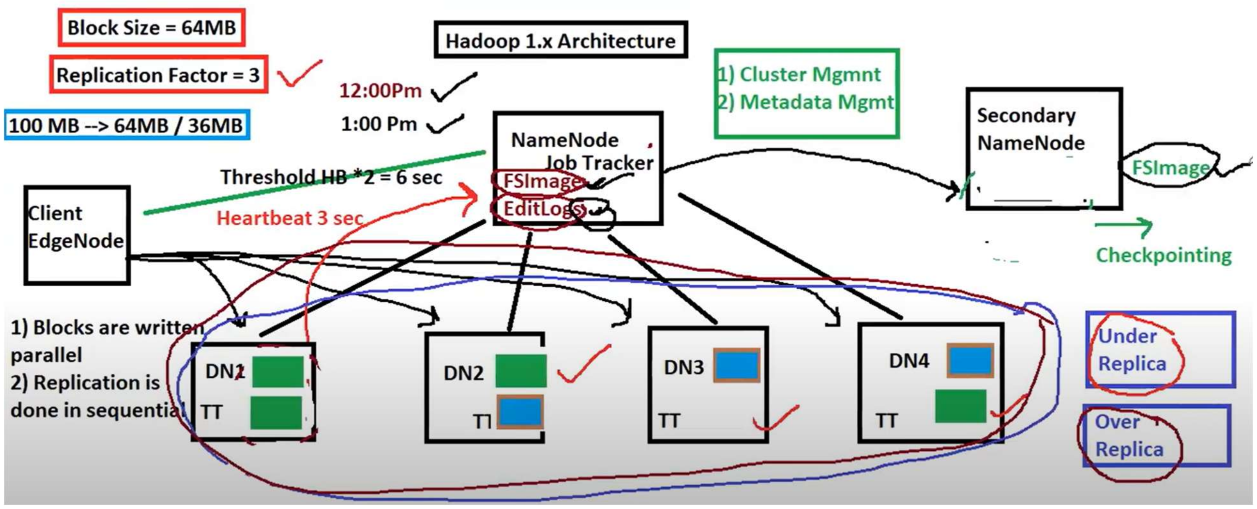


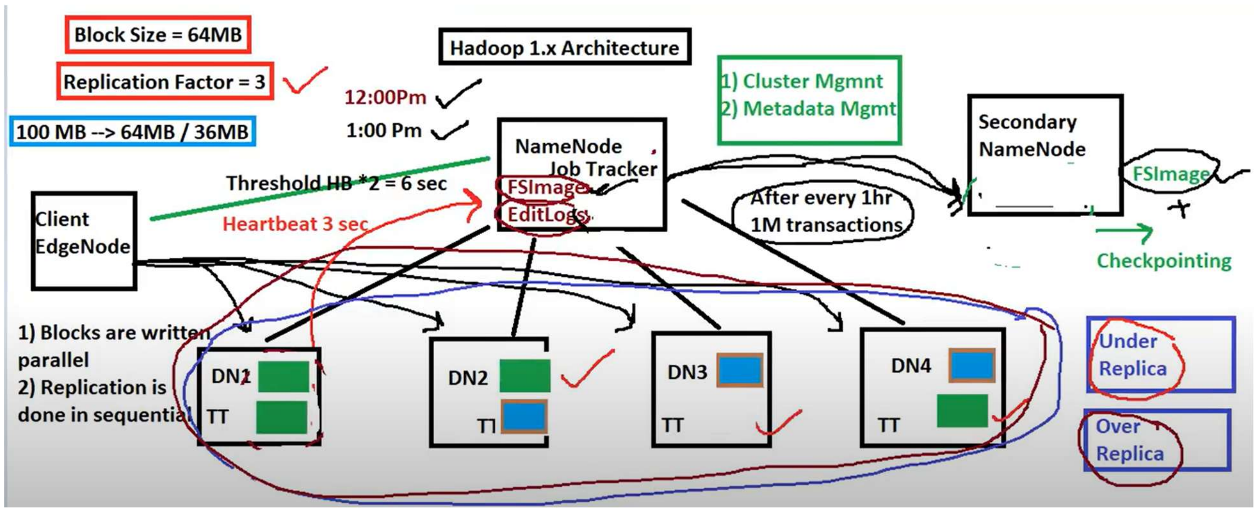
Speculative Execution

Hadoop does speculative execution where if a machine is slow in the cluster and the map/reduce tasks running on this machine are holding on to the entire map/reduce phase, then it runs redundant jobs on other machines to process the same task and whichever task gets completed first reports back to the job tracker and results from the same are carried forward into the next phase

At the start of the name node, the task of anything asked by the client will be stored in fsimage and edit logs but there will secondary name node with an empty fsimage. After 1 hr there will be checkpointing and edit logs will get erased in the secondary name node and have a new fsimage which is in sync with the name node.



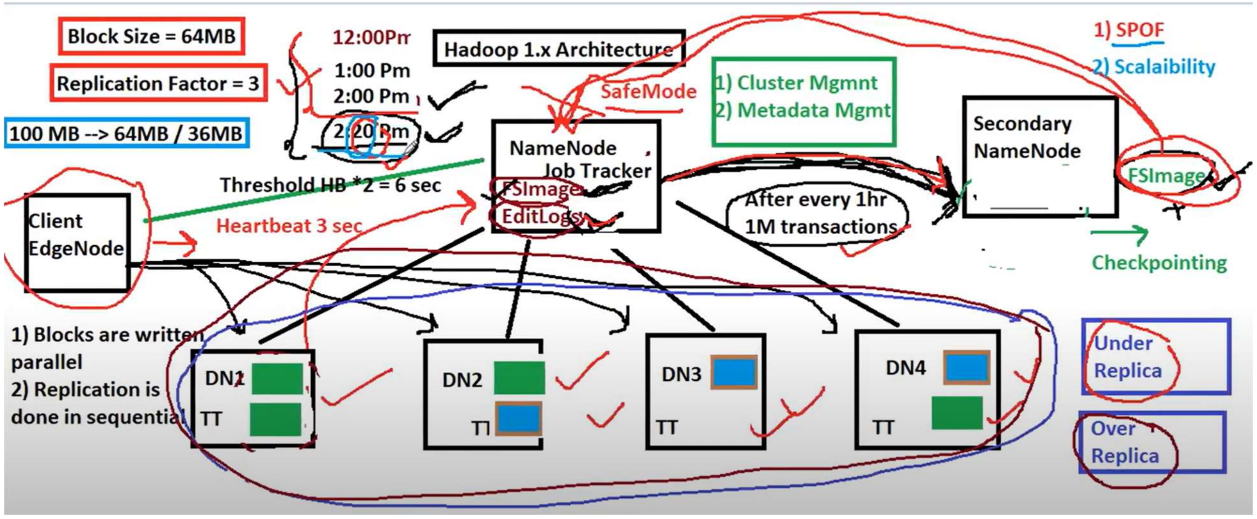




After getting in sync the edit logs of the name node will get erased. Important to remember that for every hour the edit logs of the name node will send for checkpointing.

Limitations:-

The name node goes down it will somehow use fsimage of SNN but will lose some data during that process leading to SPOF and NAME NODE scalability.



Namenode:

- The Namenode is the centre piece of an HDFS file system.
- It keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept.
- It does not store the data of these files itself.
- It is a single Point of Failure for the HDFS Cluster.
- HDFS is not currently a High Availability system. When the Name Node goes down, the file system goes offline. There is an optional Secondary Name Node that can be hosted on a separate machine. It only creates checkpoints of the namespace by merging the edits file into the fsimage file and does not provide any real redundancy.

How Data is read or written in HDFS

Client applications (like get, put) talk to the Name Node whenever they wish to locate a file, or when they want to add/copy/move/delete a file. The Name Node responds the successful requests by returning a list of relevant Data Node servers where the data lives.

Recommended Namenode for production use.

- Use a good server with lots of RAM. The more RAM you have, the bigger the file system or the smaller the block size.
- Use ECC RAM.
- On Java6u15 or later, run the server VM with compressed pointers -XX:+UseCompressedOops to cut the JVM heap size down.
- List more than one name node directory in the configuration, so that multiple copies of the file system meta-data will be stored. As long as the directories are on separate disks, a single disk failure will not corrupt the meta-data.
- Configure the NameNode to store one set of transaction logs on a separate disk from the image.
- Configure the NameNode to store another set of transaction logs to a network mounted disk.
- Monitor the disk space available to the NameNode. If free space is getting low, add more storage.
- Do not host DataNode, JobTracker or TaskTracker services on the same system.

1. **FsImage :** It is a file stored in OS Filesystem that contains the complete directory structure of HDFS with details like file location, permissions, replication details etc. This file is used by the Name Node when it is started.
2. **Edit Log :** It is a transaction log that captures all the changes in the HDFS file system since the last FSImage file was successfully created.it stores all information like addition/deletion of new files, change of any file permission, changing replication factors etc.

The above 2 files are stored in local file system of Name node.

When namenode starts, it reads all the content of FsImage which contains the complete information about each and every file as well as applies all the changes in the edit logs and refreshes the edit log. Once the namenode is up and running, all further IO operations are written back in edit log file and never on the FsImage file.

This is done because too many read and write on FsImage will make the Image file slow and you will also have to take care of lock so that wrong information is never read.

At any point of time, If you read FsImage and apply all the edits of the edit log, then you get the complete information of the file system.

Namenode restart does not take place frequently, which means if we restart the namenode after 1 month, then the size of edit file might have grown huge and it might take a lot of time for namenode to start up, since it has to apply all the edit logs.

What if someone can do this job periodically and silently. That's what is done by Secondary name node

Responsibility of Secondary Namenode:

1. Secondary namenode periodically reads the latest fsimage file and edit log file from Namenode.
2. Applies each transaction in the edit log into latest fsimage file and creates a new merged fsimage file.
3. Merged fsimage file is send back to Namenode. This entire process is called as check point process.

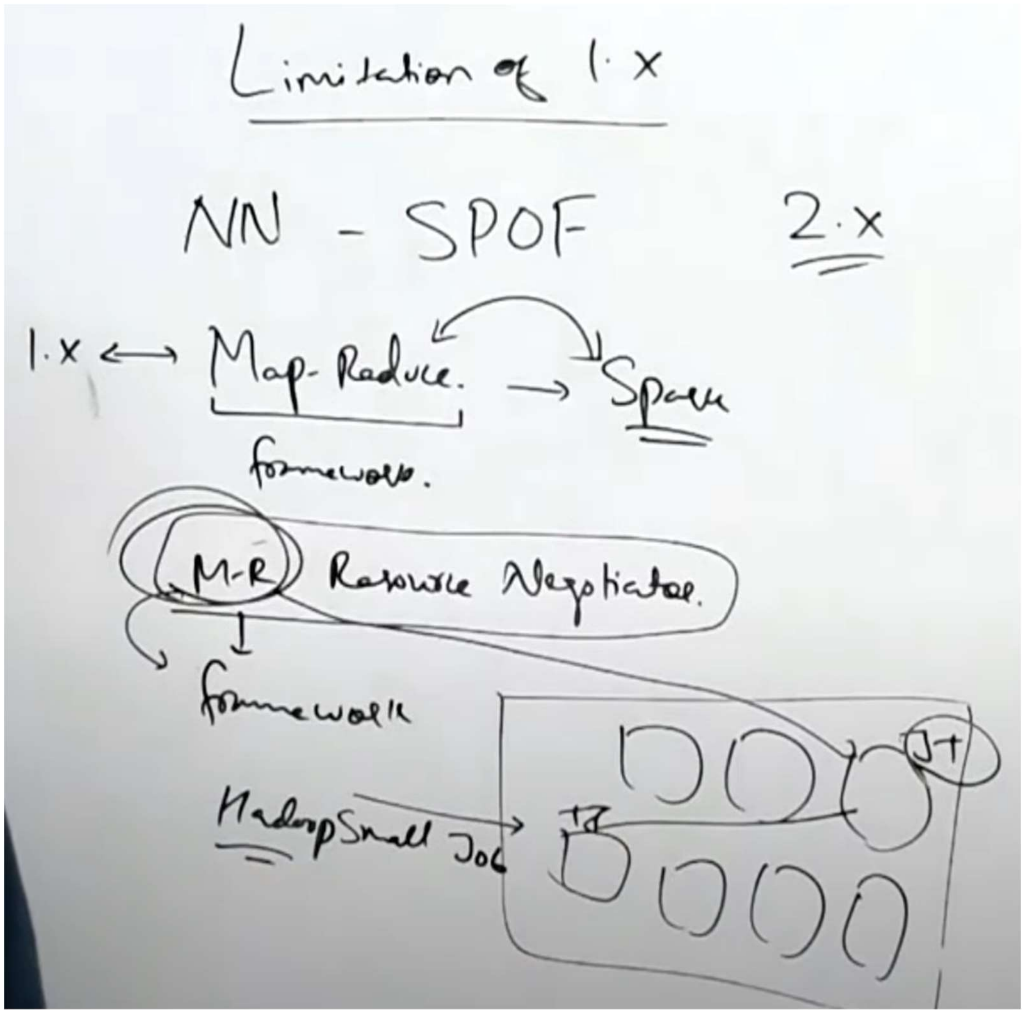
How Checkpoint is decided

We have hdfs-site.xml which can have below 2 parameters. When its value is reached checkpoint process is initiated.

1. **dfs.namenode.checkpoint.period** This defines the maximum delay between 2 checkpoints. It defaults to 1 hour

2. **dfs.namenode.checkpoint.txns** This defines number of unchecked transactions after which checkpoint should take place. Defaults to 1 million.

Which means, by default every 1 hour or whenever total unchecked transaction reaches 1 million, which ever occurs first, we will have checkpoint process initiated.



References:-

1. https://www.youtube.com/watch?v=Ra6o8pi5oFw&list=PL2jvR2CemrGopSjBChw8_Hfn10dIGIssX&index=2
2. https://www.youtube.com/watch?v=eBQyWMqw1J8&list=PL2jvR2CemrGopSjBChw8_Hfn10dIGIssX&index=3
3. https://www.youtube.com/watch?v=7r_u7BPNzHE&list=PLmDjnFKgzl2jwB6dmSimTBubJ5Mv_ZT0Z