**Big Data : The story of Hadoop**

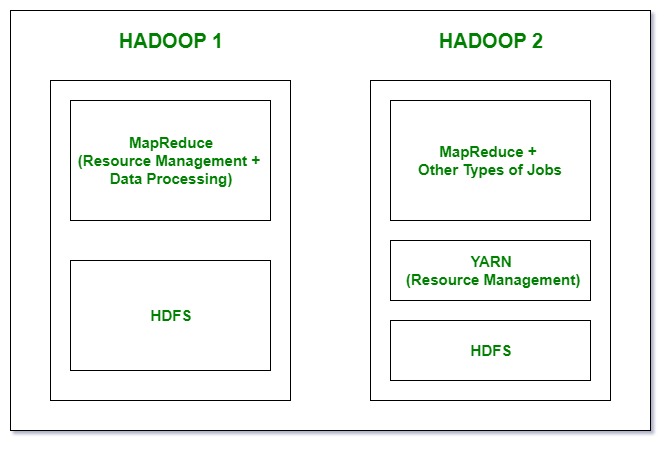
**By - Sapoonjyoti DuttaDuwarah**

The term Big-data is thrown around a lot nowadays isn’t it? People have started using big data as a term so frequently that it has become as popular as soda used to be back in the day. However, as the secrets to soda and how it’s made seems to be closely guarded at Coca-cola factory(Sorry not sorry Pepsi) in Atlanta, what’s under the hood in terms of big data is out for the public to see as open source softwares have allowed not just access but an opportunity for everyone to join the movement and share the developments to grow the world together. In essence, Big data is what it sounds like, just a huge amount of data. While it wasn’t long as that your limewire download of 50 MB files would look big, when people refer to big data it’s often in way larger magnitudes of petabyte. Now while, data became bigger instead of just building one big computer with all the power in the world like people said supercomputers would become, instead the process seems to be distributed over smaller computers allowing for many benefits and problems of it’s own. A distributed file system is a file system with data stored on a server. The data is accessed and processed as if it was stored on the local client machine. The DFS makes it convenient to share information and files among users on a network in a controlled and authorized way. A way for people to use DFS is Hadoop which became a game changer once people started deploying it with 1000s of clusterers. This article dives into Hadoop and how far it’s come from Hadoop 1.0 to it’s current version Hadoop3.0, the technological challenges that it overcame (and still is overcoming) while also building a usable system that has grown at a pace of a highly addictive drug.

Apache Hadoop is an open source software framework for storage and large scale processing of data-sets on clusters of commodity hardware. Hadoop is an Apache top-level project being built and used by a global community of contributors and users. There have been 3 major releases so far, with the first stable release being deemed hadoop 1.0 and incrementally moving forward with every release. While this gives a nice bird’s eye overview of what hadoop really is, there are things that we need to familiarize ourselves with first before we dive deep into the world of hadoop.

Hadoop 1 was the first big release which encountered its own problems as it became popular. From not being able to use it on windows servers, having only 2 major components in Map reduce and Hadoop distributed file system as well as not allowing an user to have multiple masternodes became issues as the popularity of the software grew. As these problems were being tackled, it finally let to the next big update that came about as Hadoop version 2.

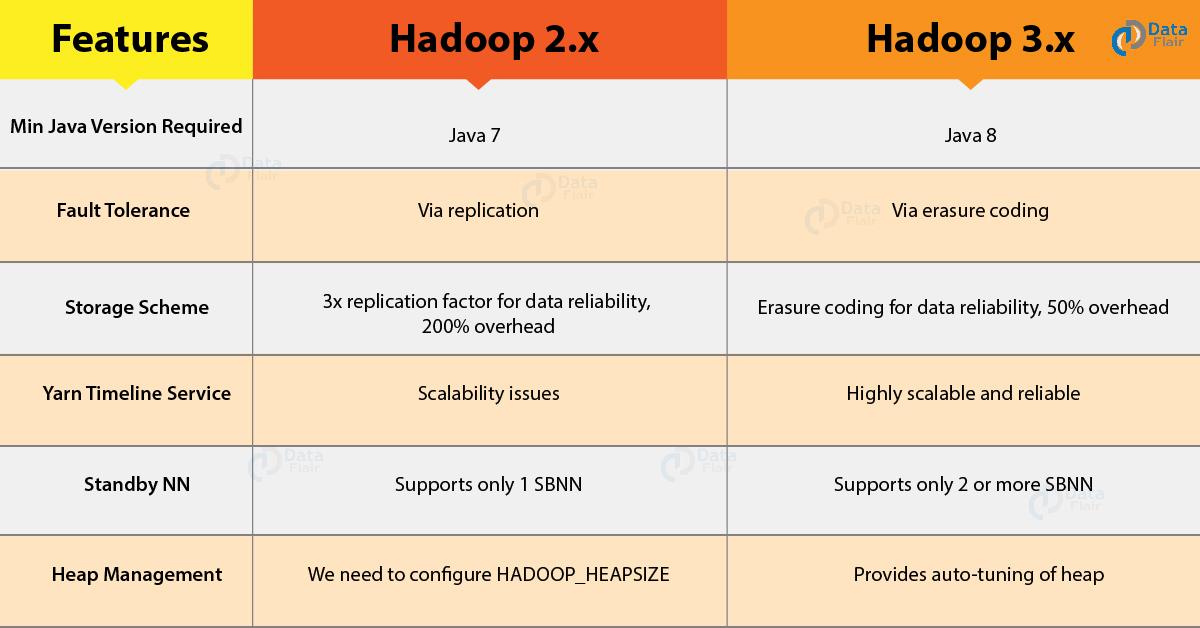
As Hadoop 2 was released and everyone started using it more, it became very clear that there were multiple advantages that came with hadoop 2. In Hadoop 2, there is again HDFS which is again used for storage and on the top of HDFS, there is YARN which works as Resource Management. It basically allocates the resources and keeps all the things going on.



The primary difference and an absolute game changer when going from hadoop 1 to hadoop 2 was the introduction of YARN as the resource manager. As mentioned above, there was a problem with the fact that hadoop 1 didn’t allow users to create multiple masternodes which built in an automatic single point of failure, which essentially meant that if your one masternode went down, your cluster was about to come crashing down regardless of how well your slave nodes were doing. Hadoop 2 took care of this by allowing users to have multiple masternodes which would ensure that in case a masternode was disconnected/crashed, it wouldn’t result in the whole system coming down. Looking at Hadoop 1 and Hadoop 2 combined, Hadoop has grown by 102% overall when we compare 2017 to 2016 total usage, even with Hadoop 1’s deprecation and the community moving to support Hadoop 2 going forward. As workloads have moved to Hadoop 2, usage has increased by 364% since December 2016, while Hadoop 1 (Hive on MR) has declined by 308% in usage.This emphasizes the fact that as Hadoop 2 updates made a big difference for the users and dividing up the environment in version 2 really helped Hadoop become even more popular.

HADOOP 3.

Technology develops at a stunningly rapid pace which means that software that is being written to help in that development needs to develop at an even quicker pace. That allowed for the development of Hadoop 3 which was released recently. While there isn’t a stable version that is going to be deployed by facebook tomorrow, the developments have called for optimism as hadoop 3 offers even more than what hadoop 2 does currently.



As seen in the features above Hadoop has made it a priority to build a fault tolerant system by ensuring to use erasure coding as well as allowing for more scalability with yarn. Replication technique provides for fault tolerance in Hadoop 2. We can configure the replication factor as per the requirement. Its default value is three. In the event of loss of any file block, Hadoop recovers it from the existing replicated blocks. Under erasure coding the blocks are not replicated in fact HDFS calculates the parity blocks for all file blocks. Now whenever the file blocks get corrupted, the Hadoop framework recreates using the remaining blocks along with the parity blocks.

The storage overhead in Hadoop 2 is 200% with the default replication factor of 3. Suppose a file “A” divides into 6 blocks in HDFS. With a replication factor of 3, we would be having 18 blocks for the file “A” stored in the system. As Hadoop 3 adopts Erasure Coding for fault tolerance, it minimizes the storage overhead of the data. Again take the example of a file with 6 blocks. Erasure Coding creates 3 more parity blocks.

Hadoop 2 works on the principle of guaranteed containers. In this, the container will start running immediately as there is a guarantee that the resources will be available. But it has two drawbacks :

a) FeedBack Delays – Once the container finishes execution it notifies the resource manager about the released resources. When the Resource Manager schedules a new container at that node, the application master gets notified. Then AM starts the new container. Hence there is a delay introduced because of these notifications given to RM and AM.

b) Allocated v/s utilized resources – The resources which RM allocates to the container can be under-utilized. For example, RM may allocate a container 4 GB of memory out of which it uses only 2GB. This lowers the effective resource utilization.

To eradicate the above drawbacks Hadoop 3.x implements opportunistic containers. In this case, containers wait in a queue if the resources are unavailable. The opportunistic containers have lower priority than guaranteed containers. Hence the scheduler preempts opportunistic containers to make room for guaranteed containers.

Hadoop 2 supports a single active NameNode and a single standby NameNode. This architecture is capable of tolerating the failure of one NameNode. Hadoop 3 has improved so that we can configure multiple standby Namenode. In a system having three NameNodes configured can tolerate the failure of two NameNodes. Every firm big or small has understood the benefit of implementing Hadoop and harnessing the power of data. The new version provides better optimization and usability. It also provides certain architectural improvements. These improvements bring even more capabilities into the hands of the users as soon as possible.

As we move forward, we will have more and more systems being designed on top of hadoop and hence it’s important to understand how far we have come in terms of being able to compute such copious amounts of data efficiently. While there is a long way still to go from achieving the ultimate computing goals, we can at least be excited right now about how far Hadoop has gone and helped us achieve goals that even just at the beginning of this century seemed improbable.

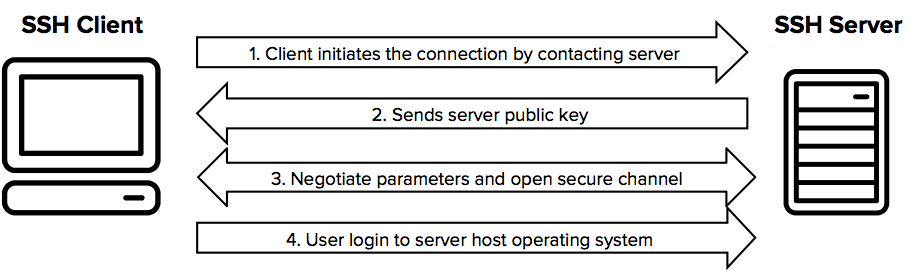
**The Age Old Story of SSH**

**By - Sapoonjyoti DuttaDuwarah**

As this world got more and more connected through the explosion of the internet, there became a need to be able to securely communicate amongst people who needed to be able to securely communicate within their own network but also over huge networks like the internet. This is where SSH has played a very key role (you’ll get the joke later later, I promise) in allowing people to do so.

The SSH protocol (also referred to as Secure Shell) is a method for secure remote login from one computer to another. It provides several alternative options for strong authentication, and it protects the communications security and integrity with strong encryption. It is a secure alternative to the non-protected login protocols (such as telnet, rlogin) and insecure file transfer methods (such as FTP).

The protocol works in the client-server model, which means that the connection is established by the SSH client connecting to the SSH server. The SSH client drives the connection setup process and uses public key cryptography (now you understand the joke) to verify the identity of the SSH server. After the setup phase the SSH protocol uses strong symmetric encryption and hashing algorithms to ensure the privacy and integrity of the data that is exchanged between the client and server.



There are several options that can be used for user authentication. The most common ones are passwords and public key authentication.

The public key authentication method is primarily used for automation and sometimes by system administrators for single sign-on. It has turned out to be much more widely used than we ever anticipated. The idea is to have a cryptographic key pair - public key and private key - and configure the public key on a server to authorize access and grant anyone who has a copy of the private key access to the server. The keys used for authentication are called SSH keys. Public key authentication is also used with smartcards, such as the CAC and PIV cards used by US government.

The main use of key-based authentication is to enable secure automation. Automated secure shell file transfers are used to seamlessly integrate applications and also for automated systems & configuration management.

Once a connection has been established between the SSH client and server, the data that is transmitted is encrypted according to the parameters negotiated in the setup. During the negotiation the client and server agree on the symmetric encryption algorithm to be used and generate the encryption key that will be used. The traffic between the communicating parties is protected with industry standard strong encryption algorithms (such as AES (Advanced Encryption Standard)), and the SSH protocol also includes a mechanism that ensures the integrity of the transmitted data by using standard hash algoritms (such as SHA-2 (Standard Hashing Algorithm)).