**Hadoop 2.0 Cluster Architecture Federation**

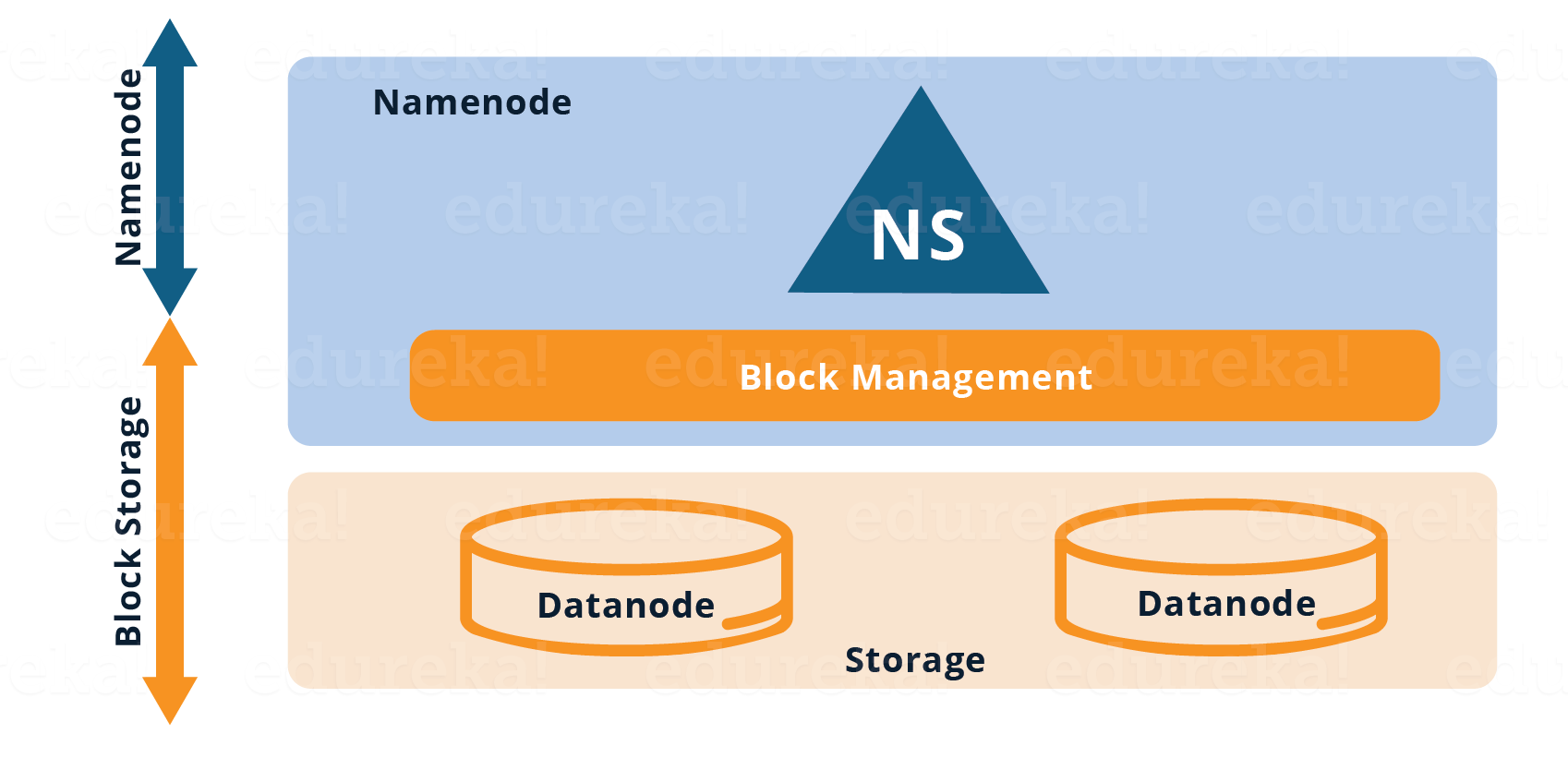
**Introduction:**

In this blog, I will deep dive into Hadoop 2.0 Cluster Architecture Federation. Apache Hadoop has evolved a lot since the release of Apache Hadoop 1.x. As you know from my previous blog that the [***HDFS Architecture***](http://www.edureka.co/blog/apache-hadoop-hdfs-architecture?utm_source=blog&utm_medium=related-posts&utm_campaign=overview-of-hadoop-cluster-2-0-federation) follows Master/Slave Topology where NameNode acts as a master daemon and is responsible for managing other slave nodes called DataNodes. In this ecosystem, this single Master Daemon or NameNode becomes a bottleneck and on the contrary, companies need to have NameNode which is highly available. This very reason became the foundation of HDFS Federation Architecture and [***HA (High Availability) Architecture***](http://www.edureka.co/blog/how-to-set-up-hadoop-cluster-with-hdfs-high-availability?utm_source=blog&utm_medium=related-posts&utm_campaign=overview-of-hadoop-2-0-cluster-architecture-federation).

The topics that I have covered in this blog are as follows:

* The current HDFS Architecture
* Limitations of current HDFS Architecture
* HDFS Federation Architecture

**Overview of Current HDFS Architecture:**



As you can see in the figure above, the current HDFS has two layers:

* **HDFS Namespace (NS):** This layer is responsible for managing the directories, files and blocks. It provides all the File System operation related to Namespace like creating, deleting or modifying the files or the file directories.
* **Storage Layer:** It comprises two basic components.
  1. **Block Management**: It performs the following operations:
     + Checks heartbeats of DataNodes periodically and it manages DataNode membership to the cluster.
     + Manages the block reports and maintains block location.
     + Supports block operations like creation, modification, deletion and allocation of block location.
     + Maintains replication factor consistent throughout the cluster.

2. **Physical Storage**: It is managed by DataNodes which are responsible for storing data and thereby provides Read/Write access to the data stored in HDFS.

So, the current HDFS Architecture allows you to have a single namespace for a cluster. In this architecture, a single NameNode is responsible for managing the namespace. This architecture is very convenient and easy to implement. Also, it provides sufficient capability to cater the needs of the small production cluster.

**Limitations of Current HDFS:**

As discussed earlier, the current HDFS did suffice to the needs and use cases of a small production cluster. But, big organizations like Yahoo, Facebook found some limitations as the HDFS cluster grew exponentially. Let us have a quick look at some of the limitations:

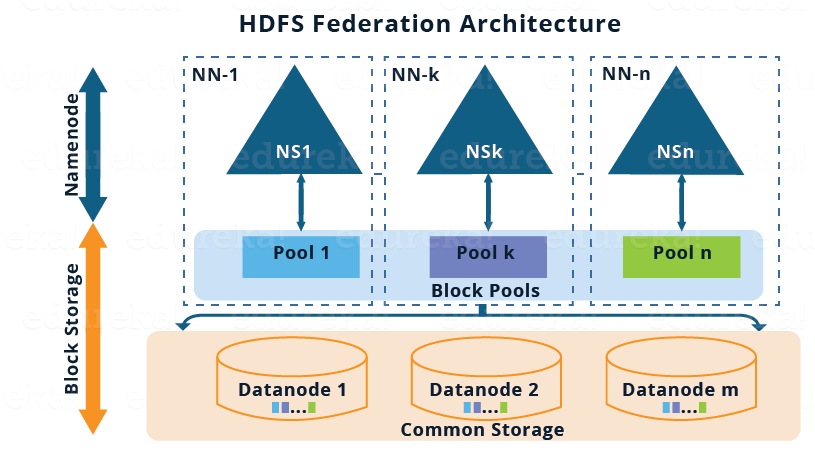
1. The namespace is **not scalable** like DataNodes. Hence, we can have only that number of DataNodes in the cluster that a single NameNode can handle.
2. The two layers, i.e. Namespace layer and storage layer are **tightly coupled** which makes the alternate implementation of NameNode very difficult.
3. The performance of the entire Hadoop System depends on the **throughput** of the NameNode. Therefore, entire performance of all the HDFS operations depends on how many tasks the NameNode can handle at a particular time.
4. The NameNode stores the entire namespace in RAM for fast access. This leads to limitations in terms of **memory size** i.e. The number of namespace objects (files and blocks) that a single namespace server can cope up with.
5. Many of the organizations (vendor) having HDFS deployment, allows multiple organizations (tenant) to use their cluster namespace. So, there is no separation of namespace and therefore, there is **no isolation** among tenant organization that are using the cluster.

[Become a Hadoop Expert Now!](http://www.edureka.co/big-data-and-hadoop?utm_source=blog&utm_medium=blog-cta&utm_campaign=hadoop-2-cluster-architecture-federation)

**HDFS Federation Architecture:**

* In HDFS Federation Architecture, we have horizontal scalability of name service. Therefore, we have multiple NameNodes which are federated, i.e. Independent from each other.
* The DataNodes are present at the bottom i.e. Underlying storage layer.
* Each DataNode registers with all the NameNodes in the cluster.
* The DataNodes transmit periodic heartbeats, block reports and handles commands from the NameNodes.

The pictorial representation of the HDFS Federation Architecture is given below:



Before moving ahead, let me briefly talk about the above architectural image:

* There are multiple namespaces (NS1, NS2,…, NSn) and each of them is managed by its respective NameNode.
* Each namespace has its own block pool ( NS1 has Pool 1, NSk has Pool k and so on ).
* As shown in the image, the blocks from pool 1 (sky blue) are stored on DataNode 1, DataNode 2 and so on. Similarly, all the blocks from each block pool will reside on all the DataNodes.

Now, let’s understand the components of the HDFS Federation Architecture in detail:

**Block Pool:**

Block pool is nothing but set of blocks belonging to a specific Namespace. So, we have a collection of block pool where each block pool is managed independently from the other. This independence where each block pool is managed independently allows the namespace to create Block IDs for new blocks without the coordination with other namespaces. The data blocks present in all the block pool are stored in all the DataNodes. Basically, block pool provides an abstraction such that the data blocks residing in the DataNodes (as in the Single Namespace Architecture) can be grouped corresponding to a particular namespace.

**Namespace Volume:**

Namespace volume is nothing but namespace along with its block pool. Therefore, in HDFS Federation we have multiple namespace volumes. It is a self-contained unit of management, i.e. Each namespace volume can function independently. If a NameNode or namespace is deleted, the corresponding block pool which is residing on the DataNodes will also be deleted.

**Demo On Hadoop 2.0 Cluster Architecture Federation | Edureka**

Now, I guess you have a pretty good idea about HDFS Federation Architecture. It is more of a theoretical concept and people do not use it in a practical production system generally. There are some implementation issues with HDFS Federation that makes it difficult to deploy. Therefore, the *HA (High Availability) Architecture* is preferred to solve the Single Point of Failure problem. I have covered the [***HDFS HA Architecture***](http://www.edureka.co/blog/how-to-set-up-hadoop-cluster-with-hdfs-high-availability?utm_source=blog&utm_medium=related-posts&utm_campaign=overview-of-hadoop-2-0-cluster-architecture-federation) in my next blog.

[<< Previous Blog](http://www.edureka.co/blog/how-to-set-up-hadoop-cluster-with-hdfs-high-availability/?utm_source=blog&utm_medium=blog-cta&utm_campaign=hadoop-2-cluster-architecture-federation-p)   [Next Blog >>](http://www.edureka.co/blog/install-hadoop-single-node-hadoop-cluster?utm_source=blog&utm_medium=blog-cta&utm_campaign=hadoop-2-cluster-architecture-federation-n)