Nodes and Data Distribution

- A collection of connected nodes is called a cluster
- All nodes know about all the other nodes in the cluster and can forward client requests to the appropriate node. Besides that, each node serves one or more purpose

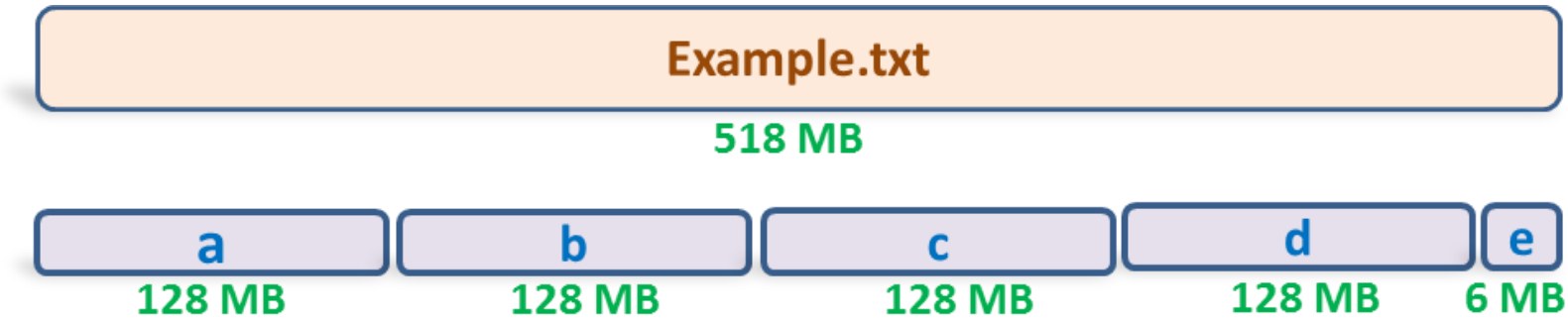
Master-eligible node

- A node that has node.master set to true (default), which makes it eligible to be elected as the *master node*, which controls the cluster.
- The master node is responsible for lightweight cluster-wide actions such as creating or deleting an index, tracking which nodes are part of the cluster, and deciding which shards to allocate to which nodes. It is important for cluster health to have a stable master node.
- Master nodes must have access to the data/ directory (just like data nodes) as this is where the cluster state is persisted between node restarts.
- Indexing and searching your data is CPU-, memory-, and I/O-intensive work which can put pressure on a node's resources. To ensure that your master node is stable and not under pressure, it is a good idea in a bigger cluster to split the roles between dedicated master-eligible nodes and dedicated data nodes.
- While master nodes can also behave as coordinating nodes and route search and indexing requests from clients to data nodes, it is better *not* to use dedicated master nodes for this purpose. It is important for the stability of the cluster that master-eligible nodes do as little work as possible.
- Type of metadata in NameNode/ Master-eligible is List of files, Block locations of files, c) and File access control information

Data node

- In Hadoop, HDFS splits huge files into small chunks known as blocks. These are the smallest unit of data in a filesystem. We (client and admin) do not have any control over the block like block location. Namenode decides all such things.
- The main benefit of having dedicated data nodes is the separation of the master and data roles.
- Data Nodes add storage and processing capacity.
- Data Nodes are plug-n-play and can be added to a deployment at any time.
- It is the slave node that stores actual data.

Data Node Example



- HDFS stores each file as blocks. However, the block size in HDFS is very large. The default size of the HDFS block is 128MB which you can configure as per your requirement. All blocks of the file are the same size except the last block, which can be either the same size or smaller. The files are split into 128 MB blocks and then stored into the Hadoop file system. The Hadoop application is responsible for distributing the data block across multiple nodes.

Why is HDFS Block size 128 MB in Hadoop?

- Many of us have this question in mind that “why the block size in HDFS is so large?” Let us understand this.
- HDFS have huge data sets, i.e. terabytes and petabytes of data. So like **Linux** file system which have 4 KB block size, if we had block size 4KB for HDFS, then we would be having too many data blocks in Hadoop HDFS and therefore too much of metadata. So, managing this huge number of blocks and metadata will create huge overhead and traffic which is something which we don’t want.
- On the other hand, block size can’t be so large that the system is waiting a very long time for one last unit of data processing to finish its work.

Advantages of Hadoop HDFS blocks

Simplicity of storage management

As the size of HDFS blocks is fixed, so it is very easy to calculate the number of blocks that can be stored on the disk.

b. Ability to store very large files

HDSF can store very large files which can be even larger than the size of a single disk as the file is broken into blocks and distributed across various nodes.

c. Fault tolerance and High Availability of HDFS

Blocks are easy to replicate between the datanodes and thus provide **fault tolerance** and **high availability of HDFS**.

d. Simple Storage mechanism for datanodes

HDFS blocks simplify the storage of the datanodes. Metadata of all the blocks is maintained by namenode. The datanode doesn't need to concern about the block metadata like file permissions etc.

Ingest node

- Use an ingest node to pre-process documents before the actual document indexing happens. The ingest node intercepts bulk and index requests, it applies transformations, and it then passes the documents back to the index or bulk APIs.
- All nodes enable ingest by default, so any node can handle ingest tasks. You can also create dedicated ingest nodes. To disable ingest for a node

Tribe node

- The tribes feature allows a tribe node to act as a federated client across multiple clusters.
- The tribe node works by retrieving the cluster state from all connected clusters and merging them into a global cluster state. With this information at hand, it is able to perform read and write operations against the nodes in all clusters as if they were local. Note that a tribe node needs to be able to connect to each single node in every configured cluster.

Coordinating node

- If you take away the ability to be able to handle master duties, to hold data, and pre-process documents, then you are left with a *coordinating* node that can only route requests, handle the search reduce phase, and distribute bulk indexing. Essentially, coordinating only nodes behave as smart load balancers.
- Coordinating only nodes can benefit large clusters by offloading the coordinating node role from data and master-eligible nodes. They join the cluster and receive the full cluster state, like every other node, and they use the cluster state to route requests directly to the appropriate place(s).
- Adding too many coordinating only nodes to a cluster can increase the burden on the entire cluster because the elected master node must await acknowledgement of cluster state updates from every node! The benefit of coordinating only nodes should not be overstated — data nodes can happily serve the same purpose.

Virtual Cluster and Virtual Node

- Virtual clusters are built with virtual machines (VMs) installed at distributed servers from one or more physical clusters. The VMs in a virtual cluster are interconnected logically by a virtual network across several physical networks.
- Virtual clusters are formed with physical machines or a VM hosted by multiple physical clusters. Provisioning VMs to a virtual cluster is done dynamically to have the following properties:

1. The virtual cluster nodes can be either physical machines or VMs. You can deploy multiple VMs running different OSes on the same physical node.
2. A VM runs with a guest OS (often different from the host OS) that manages resources in the physical machine where the VM is implemented.
3. The purpose of using VMs is to consolidate multiple functionalities on the same server. This will greatly enhance server utilization and application flexibility.
4. You can colonize or replicate VMs in multiple servers for the purpose of promoting distributed parallelism, fault tolerance and disaster recovery.
5. The size (number of nodes) of a virtual cluster can grow or shrink dynamically, similar to how an overlay network varies in size within a peer-to-peer network.
6. If any physical node fails, it might disable some of the VMs installed on the failing nodes. However, any VM failure will not pull down the host system.

Couti….

- You have to effectively manage VMs running on a mass of physical computing nodes (also called virtual clusters) in a high-performance virtualized computing environment. This involves virtual cluster deployment, monitoring and management over large-scale clusters. You'll also have to apply resource scheduling, load balancing, server consolidation, fault tolerance and other techniques. In a virtual cluster system, it's important to store the large number of VM images efficiently.

Namenode* *SPoF in Gen 1*

Jobtracker

File system
Meta-data

Secondary namenode
Moves meta-data from
RAM to Disk (Checkpoint)
Does NOT provide redundancy!

File system
Job tracking
i.e. copyToLocal

MASTER

Slave

Datanodes

File system
'chunked'
data

Tasktracker

Node level task
tracking

An edge node is a node that 'externalizes' the HDFS cluster for client access

Reference

- https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html