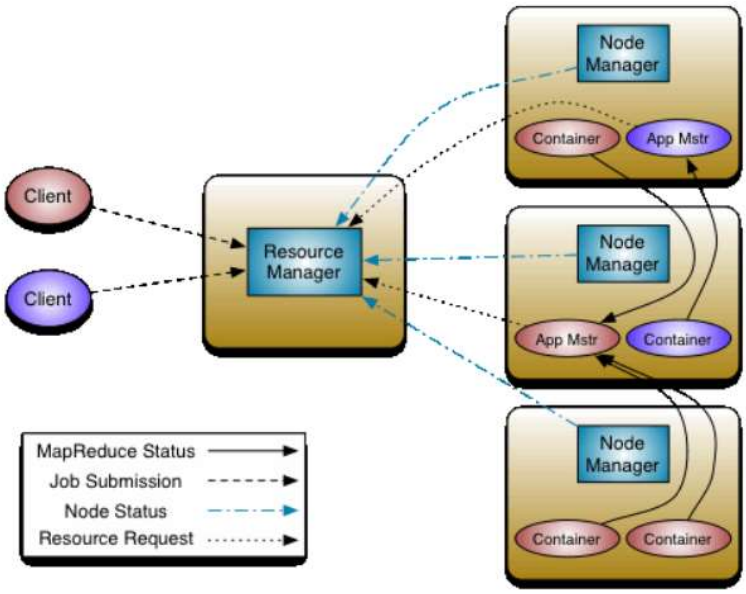
Big Data Concepts

26 December 2023 09:33

- Hadoop distributed File System --> distributed storage designed to store vast amount of data across multiple machines in a Hadoop cluster.
  - Acts as central role in the Apache Hadoop echo systems. This is the primary storage layer for big data processing.
  - Where we required to address the 3V's we can make use of HDFS, Anaalytics by providing a scalable and Fault tolerant stolution for storaing and managing large datasets.
    - Write-once Read-Many (WORM) --> once a file sis written to HDFS, it becomes immutable (it can't be modified). Modifications or updates involves creating a new version of the file.
    - Read-optimized: this is well suited for large-scale data analytics where data is primarily appended and read.
    - FT / Data integrity / Parallel process - High Throughput / interoperability
      - Interoperability ---> Integrates with other Hadoop projects and tools such as Apache Hive, Apache Pig and Apache Spark etc..,
  - Storage for very large data sets hence called as bigdata
  - We can also use or implement data governance, access control and encryption to ensure secure and compliant data storage
- Core Components of HDFS :-
  - Name Node --> master server that manage the metadata of the file system
  - Data node --> slave server responsible for storing the actual data blocks
  - Blocks ---> files divided into size or 64M or 128MB (preferred is 128MB or we can also go for 256MB)
  - Namespace and Metadata
    - Namespace --> refers to the file and directory structure maintained by the namenode. Manages the organization of files and directories in a tree-like structure
    - Metadata --> includes information about files, directories, permissions and block locations. Tracks metadata for each file , including its size, permission of the locations of the data blocks.
  - Secondary NameNode
    - It helps the primary namenode in checkpoint and creating periodic snapshots of the file system metadata to prevent is corruptions
    - Note: it does not act as a backup for the primary namenode.
    - Merges the edit log and fsimage to create a new, updateed fsimage
    - Helps reduce the recovery time in the event of a namenode failure.
  - Block Report
    - Datanodes periodically send block reports to the namenode, provide the information about the blocks they are storing --> health status
      - Informs the namenode about the existence and health of data blocks
      - Assists the namenpde in maintaining an up-to-date view of the block locations

- HDFS Deamons:
  - Mandatory - Namenode, Datanode, secondary namenode,
  - Optional - Backupnode --> additional layer for FT for the namenode by maintaining a copy of the file system namespace.
    - Acts as a standby for namenode --> overcome the failure of namenode and reduces the downtime
    - Receive the namespace updates from the active namenode

- YARN Deamons: (Yet Another Resource Negotiator)
  - Provides an scalable and efficient framework for managing resources and scheduling applications on a hadoop cluster.
  - Enables resource sharing among various applications and support diverse processing models
  - The daemons:
    - Resource Manager --> main control which managers and authorizes the resources among all the applicators in the hadoop cluster.
      - Receives the resource requests from the application master
      - Allocates resources based on availability
      - Monitors and tracs resource usage across the cluster
    - NodeManager (runs on each worker nodes) (ie, on all the slave nodes in an hadoop cluster)
      - Responsible for managing resources locally on each node.
      - Monitors resource usage (CPU, Memory (RAM)) on the node
      - Managers containers, which are execution env for app processes
      - Reports resource usage and availability to the resource manager
    - ApplicationMasters:
      - Depends from application to application
      - Responsible for negotiating resources with the resource manager and manage execution of specific application
      - Requesting the resource from resource manager, negotiates container allocations for application tasks and monitors , reports the progress to the resource manager
    - HitoryServer:
      - Knowns as jobhistroyserver -- responsible for managing and serving historical information about completed applications
      - Very useful for analysis and debugging.



YARN Architecture

Hadoop Server Roles: Name Node, Secondary Name Node, and Data Node :-

These three server roles together form the core components of Hadoop's distributed file system (HDFS).

- The NameNode is the central authority for metadata management,.
- The Secondary NameNode assists in metadata checkpointing
- DataNodes are responsible for storing and managing the actual data blocks.

This distributed architecture enables Hadoop to handle large-scale data storage and processing across a cluster of commodity hardware.
It's essential for ensuring fault tolerance, scalability, and efficient storage and retrieval of data in a distributed computing environment.

Scaling and Rebalancing the HDFS

Scaling the HDFS -->

1. **Adding nodes to the cluster --> upon increase in the demand for processing we can add mode nodes to grow the data volumes. Adding more data nodes is called as horizontal scaling. If we add more nodes we get more storage capacity and processing power.**
2. **Increase block size by adjusting the block size in HDFS. Use a larger block size for more efficient storage for large files**
3. **We can also control data redundancy and storage capacity by adjusting the replication factor (determines how many copies of each data block are maintained across the clusters)**
4. **While scaling down a nodes in the clusters - proper care need to be taken i.e, while removing a node for the clusters we need to take proper care**

Rebalancing in HDFS --> process of redistributing data blocks across the datanodes to ensure a uniform distribution of data and maintain a balanced utilization of cluster resources

1. **Dynamic Rebalancing ---> new data blocks are placed on nodes with available storage space during the normal course of operations**
2. **Triggered Rebalancing --> can be imitated manually or automatically. The Admin can decide to relance the cluster based up the response (uneven data distribution or during the cluster operation)**
3. **HDFS balancer tool --> Balancer analyzes the distribution of data blocks and moves blocks between datanodes to achieve a more even distribution.**

Note: Upon datanode decommission or datanode removal the HDFS balancer ensures that the data blocks on the decommissioned node are replicated to other nodes to maintain the desired replication factor.
Hadoop admin can adjust confirmation parameters related to HDFS balancer (i.e, threshold for initiating relancing and the maximum bandwidth used for data movement)

Considerations while scaling and rebalancing -->
1. Impact on performance
2. Network bandwidth
3. Storage considerations
4. Automation and monitoring

For eg: CPU utilization :
form 1% to 60% --> Green 61% to 80% --> Amber 81% to 95% --> Red 95% and above (resource under very high utilization and congestion)

Note : CPU utilization can reach 100% but should not remain there for a long period of time

For eg: the current setup in our lab we have one namenode and two datanodes i.e, there are two nodes in the clusters. If one node fails then the entire load should be handed by the nodes which is working and hence the load on that node will increase and hence can lead to a SPOF (single point of failures)

HDFS Replication:-

Replication is a fundamental feature of the Hadoop Distributed File System (HDFS) that contributes to its fault tolerance and reliability.
In HDFS, data is replicated across multiple nodes in the cluster, ensuring that even if some nodes fail, the data remains accessible.
Replication factor --> number of copies (replica) of each datablock that are maintained across the cluster. The default replication factor in HDFS is three, meaning each data block is replicated to two additional nodes.
Fault Tolerance: --> If a DataNode or a block becomes unavailable due to hardware failure or other issues, the system can retrieve the data from one of the replica nodes. Replication enhances fault tolerance by providing multiple copies of data.
Based up the storage efficiency and fault tolerance we can balance or adjust the Hadoop cluster.

How does the replication occurs -->

1. Initial replication --> When a file is initially written to HDFS, the NameNode determines the data nodes where replicas will be stored. The replication process begins with the creation of these replicas
2. Replica Placement --> HDFS aims to distribute replicas across different racks to minimize the risk of data loss in case of rack-level failures. The placement strategy is configurable.
3. Block Reports and Heartbeats --> DataNodes periodically send block reports and heartbeats to the NameNode. Block reports include information about the blocks stored on each DataNode.
4. Replica Maintenance --> The NameNode uses block reports and heartbeats to track the health of DataNodes and the status of replicas. If a replica is deemed unavailable or under-replicated, the NameNode initiates the creation of additional replicas.

We need to consider the network latency and in case of heterogeneous cluster admin need to configure the replications policies to balance storage based upon performance considerations.
Strategies for replications --> block placement polices, balancing load or adaptive replications

HDFS High Availability NameNode

As per our current setup we do not have high availability for namenodes but for datanodes we have high availability
(2node clusters against the required 3 nodes cluster setup)

Exiting Configurations :-

Per node 8GB

Data nodes - 2Nos --> 16GB
Namenode --> 1 Nos --> 8 GB

Processing power for applications
2*2.6 GHz per nodes --> 5.2GHz

overall 5.2 *2nodes --> 10.4GHz

Storage allocations
Pernode --> 40GB
approximatey we get 2 * 40 --> 80GB
Network Bandwidth -> 1Gbps per NIC - we have 2 NIC hence we can combine and bond the network interface together to get 2Gbps speed per node.

From performance tuning --> CPU , Memory (RAM), Network and Storage

Components of HDFS High Availability:

Active NameNode:
The Active NameNode is the primary NameNode that manages the metadata and namespace of the HDFS.
It is the authoritative server for handling client requests and maintaining the file system's consistency.

Standby NameNode:
The Standby NameNode is a redundant, passive counterpart to the Active NameNode.
It maintains a copy of the file system metadata and stays synchronized with the Active NameNode to quickly take over in case of a failure.

Quorum Journal Manager (QJM):
QJM is a set of nodes that form a highly available and fault-tolerant journal service.
It stores the edit logs, which record changes to the file system namespace, ensuring that both the Active and Standby NameNodes have a consistent view of the metadata.

Failover and recovery of HDFS :-

1. Automatic failover
2. Apache zookeeper (decision about active and standby namenodes)
3. Shared storage (required to store logs and checkpoints) (we can use SAN or NAS storage)
4. Graceful handover.

HDFS High Availability enhances the reliability of Hadoop clusters by introducing redundancy and failover capabilities for the NameNode. The Active and Standby NameNodes, along with the Quorum Journal Manager and ZooKeeper, work together to provide a continuous and resilient HDFS service. This is particularly crucial for mission-critical applications that require uninterrupted access to large-scale distributed data.

HDFS Rack awareness and Data pipelining

Rack awareness --->

HDFS should be capable enough to be aware bout the network topology, specifically about the physical racks in which datanodes are located.
Network topology for Hadoop clusters --> typically organized into racks which are groups of machines located close to each other with a data center.
Replica placements --> data is spread across different racks to ensure fault tolerance in the event of a rack-level failure
Rack Awareness Configuration: Administrators can configure rack awareness by specifying the network topology of the cluster, including the mapping of nodes to racks. This information is used by HDFS to make informed decisions about replica placement.
Improved data locality

Data pipelining -->

Process by which data is transmitted in a sequence of stages or pipelines from one node to another during the writing or reading of data in HDFS
Stages in Pipelining:

Writing of data:- the process involves multiple stages where data is transferred from the client to the first DataNode, then to subsequent DataNodes in the pipeline.
Reading of data or Data retrieval: the process involves transferring data from DataNodes back to the client through a similar pipeline.
Helps in Parallel data transfer and data streaming

Rack awareness and data pipelining work together to optimize the placement and transmission of data in HDFS.
When replicas are placed during the write process, rack awareness helps in spreading them across different racks for fault tolerance.
During data retrieval, the pipelining process considers the network topology to construct an efficient pipeline for transferring data back to the client.

Node failure management

HDFS node failure management involves a combination of data replication, proactive monitoring, recovery mechanisms, and fault-tolerant designs. By ensuring redundancy and implementing failover strategies, HDFS can maintain data availability and reliability, even in the face of individual node failures or planned maintenance activities. Regular testing and monitoring are crucial components of a robust node failure management strategy in HDFS.

Hadoop in the Cloud
Hadoop, the open-source distributed computing framework, has found significant adoption in cloud computing environments. Running Hadoop in the cloud offers several advantages, including scalability, flexibility, cost-effectiveness, and the ability to leverage managed services.

1. No on-premises overheads
2. Scalability
3. Cost Effective (pay-as-you-usemodel)
4. Managed services (platform as a service -PaaS)
5. Hadoop Data storage --- Amazon S3, Azure Blob Storage or Google cloud storage
6. Integration --> we can integrate with cloud-native database, data ware house and other analytic services
7. Hybrid and Multi-Cloud Deployments
8. Security and compliance [(identity and access management - IAM) , Encryption]
9. We can used the cloud based CDN's - content delivery networks
10. Manage vendor-specific Hadoop distribution
11. Big-data analytics Bulit-in integrated analytics services
12. Global reach, data governance and metadata management

Running Hadoop in the cloud offers a flexible and scalable solution for organizations looking to harness the power of big data processing without the need to invest heavily in on-premises infrastructure. The cloud environment provides a variety of services and features that complement Hadoop deployments, enabling organizations to focus on their data processing and analytics goals

Additional HDFS commands
$ hdfs dfsadmin -report > myreport.txt
$ hdfs balancer

```
hadoop@mainserver1:~$ hdfs balancer
23/12/26 10:07:39 INFO balancer.Balancer: namenodes = [hdfs://mainserver1:9000]
23/12/26 10:07:39 INFO balancer.Balancer: p         = Balancer.Parameters[BalancingPolicy.Node, threshold
=10.0]
Time Stamp               Iteration#  Bytes Already Moved  Bytes Left To Move  Bytes Being Moved
23/12/26 10:07:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... u
sing builtin-java classes where applicable
23/12/26 10:07:43 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.56.102:50010
23/12/26 10:07:43 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.56.101:50010
23/12/26 10:07:43 INFO balancer.Balancer: 0 over-utilized: []
23/12/26 10:07:43 INFO balancer.Balancer: 0 underutilized: []
The cluster is balanced. Exiting...
Balancing took 6.343 seconds
hadoop@mainserver1:~$
```

$ hdfs fsck /