Hadoop

Question 1.Mention Hadoop distribution? Difference between CDH and CDP

Answer: Hadoop distribution is a collection of software components, libraries and tools built on top of the apache Hadoop framework. These tools provide deployment, operation and integration.

Two popular distributions are:

**I Cloudera distribution for Hadoop (CDH):**

Earliers and Most widely adopted distribution.

It contains following components: HDFS, YARN, Map Reduce,Hive, Hbase, Pig, Impala Spark.

It also provides cloud era manager for cluster management and monitoring.

**CDP Cloudera DATA PLATFORM**

CDH is a traditional Hadoop distribution by Cloudera.

It addresses the challenges of modern data management and analytics.

CDP supports hybrid and multi cloud platform that supports various data processing engines and analytical tools.

It offers a comprehensive suite of Hadoop ecosystem components for on-premises deployments.

CDH focuses on providing enterprise-grade features for data management, processing, and analytics.

Cloudera Data Platform (CDP):

CDP is Cloudera's next-generation data platform.

It is designed for hybrid and multi-cloud deployments, offering seamless data management across on-premises and public cloud environments.

CDP integrates various Cloudera and Hortonworks technologies into a unified platform, including features like Shared Data Experience (SDX) for consistent security and governance.

2.Explain Hadoop Architecture

Answer: Hadoop is an open source framework design to store and process large volumes of data in a distributed env. It architecture consist of various components that works together to enable scalable and fault tolerant distributed data processing.

1. **Hadoop Distributed File System (HDFS):**

HDFS is the primary storage system used by Hadoop. It is designed to store large amounts of data across distributed commodity hardware.

**HDFS follows a master-slave architecture with two main components: NameNode and DataNode**.

The NameNode manages the metadata and namespace of files/directories including their location, while DataNodes store the actual data blocks.

1. **Yet another Resource Negotiator (YARN)**

IT IS Resposible for managing resource in the cluster to be allocated to applications.

Two Major components of YARN are:

Resource Manager: Central authority allocating cluster resources based on applications requiremtns

Node Manager: Manages resources on individual nodes.

**MapReduce Engine:**

MapReduce is the processing engine used in Hadoop for distributed data processing.

It consists of two main phases: the Map phase, Data is divided into smaller chunks processed in parallel across multiple nodes in cluster and **transformed in KEY-VALUE PAIR.** where data is processed in parallel across multiple nodes, and the

**Reduce phase,** INTERMEDIATE KEY VALUE PAIRS ARE GROUPED TOGETHER TO GENERATE OUTPUT. where the intermediate results from the Map phase are aggregated to produce the final output.

MapReduce tasks are executed on nodes where data is stored, enabling data locality and efficient processing.

YARN (Yet Another Resource Negotiator):

YARN is the resource management layer of Hadoop, introduced in Hadoop 2.x.

It decouples the resource management and job scheduling functions from the MapReduce engine, allowing Hadoop to support multiple processing frameworks beyond MapReduce, such as Spark, Hive, and HBase.

YARN consists of two main components: ResourceManager and NodeManager. ResourceManager manages cluster resources, while NodeManager runs on individual nodes to manage resources locally.

**OTHER ECOSYSTEM COMPONENTS:**

i. Apache Hive, Apache Spark, Apachepig and Apache HBASE.

3. Configuration files used during hadoop installation

Answer: hadoop-env.sh:

This file sets environment variables for Hadoop components, such as JAVA\_HOME, HADOOP\_HOME, and HADOOP\_LOG\_DIR.

core-site.xml:

Contains settings for Hadoop core services, such as the default file system (fs.defaultFS), Hadoop temporary directory (hadoop.tmp.dir), and I/O settings.

hdfs-site.xml:

Includes configuration parameters related to HDFS (Hadoop Distributed File System), such as replication factor (dfs.replication), block size (dfs.block.size), and data node directories (dfs.datanode.data.dir).

mapred-site.xml:

Contains configuration settings specific to the MapReduce framework, such as framework name (mapreduce.framework.name), task tracker settings, and job history server settings.

yarn-site.xml:

Includes configuration parameters for YARN (Yet Another Resource Negotiator), such as resource manager address (yarn.resourcemanager.hostname), node manager settings, and container allocation settings.

hadoop-metrics.properties:

Configures metrics collection and reporting for Hadoop components.

log4j.properties:

Configures logging for Hadoop components, including log levels, log file locations, and log formats.

4.Difference between Hadoop fs and hdfs dfs

Answer: *The commands hadoop fs and hdfs dfs are both command-line interfaces for interacting with the Hadoop Distributed File System (HDFS). They provide similar functionality but have some differences in terms of usage and compatibility. Here's a crisp comparison:*

*hadoop fs:*

*This is the legacy command-line interface for HDFS operations.*

*It is compatible with older versions of Hadoop.*

*Commands are executed in the form: hadoop fs -<command> <args>.*

*Example: hadoop fs -ls /user.*

*hdfs dfs:*

*This is the newer, more user-friendly command-line interface introduced in later versions of Hadoop.*

*It's designed to be more intuitive and easier to remember.*

*It's backward compatible with hadoop fs, meaning it can execute commands supported by hadoop fs.*

*Commands are executed in the form: hdfs dfs -<command> <args>.*

*Example: hdfs dfs -ls /user.*

5. Difference between Hadoop 2 and Hadoop 3

Answer: Hadoop 2 and Hadoop 3 are different versions of the Apache Hadoop framework, each introducing various improvements, features, and changes. Here's a *crisp comparison highlighting some of the key differences between the two:*

*YARN Improvements:*

*Hadoop 2 introduced YARN (Yet Another Resource Negotiator) as a resource management layer, separating the processing engine (MapReduce) from resource management.*

*Hadoop 3 builds upon YARN with several improvements, including support for resource types beyond memory and CPU****, better support for Docker containers, and enhanced resource scheduling capabilities.***

*Erasure Coding:*

*Hadoop 3 introduces support for erasure coding, a more efficient alternative to replication for data durability. Erasure coding reduces storage overhead compared to replication, while still providing fault tolerance.*

*Improved NameNode Scalability:*

*Hadoop 3 includes enhancements to the HDFS (Hadoop Distributed File System) NameNode, such as the introduction of a standby NameNode shared with the active NameNode, improving scalability and availability.*

*Enhanced APIs and Libraries:*

*Hadoop 3 provides updates and improvements to various APIs and libraries, including support for newer versions of Java, enhancements to Hadoop Common, HBase, Hive, and other components.*

*Compatibility:*

*Hadoop 3 maintains backward compatibility with Hadoop 2, allowing existing applications and workflows to migrate with minimal changes.*

*Performance Improvements:*

*Hadoop 3 includes various performance optimizations and enhancements, such as improvements to data storage formats, optimizations in data processing algorithms, and enhancements to resource management.*

*Security Enhancements:*

*Both versions include security features, but Hadoop 3 may introduce additional security enhancements and improvements, such as enhancements to Kerberos authentication, encryption, and authorization mechanisms.*

6.What is replication factor ? why its important

Answer: The replication factor refers to the number of copies (replicas) of each data block stored in a distributed file system like Hadoop's HDFS (Hadoop Distributed File System). It is a critical parameter that determines the fault tolerance, reliability, and availability of data in the distributed system.

Here's why replication factor is important:

**Fault Tolerance**: By storing multiple replicas of each data block across different nodes in the cluster, the system can tolerate node failures or data corruption without losing data. If one copy becomes unavailable due to a node failure or other issues, the system can still access the data from the remaining replicas.

**Data Reliability**: Replication helps ensure the reliability of data by providing redundancy. Even if one or more replicas become unavailable or corrupted, the system can still access the data from the remaining replicas, maintaining data integrity.

**Data Availability**: Having multiple replicas of data blocks improves data availability. Users can access the data from any available replica, even if some nodes or replicas are temporarily offline or inaccessible.

**Performance**: Replication can also improve read performance by allowing the system to read data from the replica located on the node closest to the requesting client, reducing network latency.

**Parallel Processing**: Replication facilitates parallel processing by allowing multiple nodes to access the same data simultaneously, improving overall system throughput and performance.

7.What if Datanode fails?

Answer:

If a DataNode fails in a Hadoop cluster, it can impact the availability and fault tolerance of the data stored in that node. However, Hadoop's architecture is designed to handle such failures gracefully through replication and other mechanisms. Here's what typically happens when a DataNode fails:

**Replication**: One of the primary mechanisms for fault tolerance in Hadoop is data replication. Each block of data stored in HDFS is replicated across multiple DataNodes (typically three by default). If a DataNode fails, the blocks stored on that node are still available from the replicas stored on other DataNodes.

**Block Replication**: When a DataNode fails, the NameNode, which manages the metadata of the HDFS, detects the failure through periodic heartbeat checks. The NameNode is aware of which blocks were stored on the failed DataNode and initiates the replication process for any blocks that no longer meet the desired replication factor. It schedules the replication of those blocks to other DataNodes in the cluster to maintain the desired replication factor.

**Rebalancing**: Hadoop's HDFS is designed to automatically rebalance data across the cluster. If a DataNode fails and is later replaced or brought back online, the system will redistribute data blocks to ensure a balanced distribution across the remaining nodes in the cluster.

**Read and Write Operations**: During a DataNode failure, Hadoop continues to serve read and write operations for the data stored in the cluster. Read operations are served from the available replicas of the data blocks stored on other DataNodes. Write operations continue to be directed to the remaining DataNodes, ensuring that new data is stored with the desired replication factor.

**Data Integrity**: Hadoop employs checksums to ensure data integrity. When a DataNode retrieves data from another DataNode to fulfill a read request, it verifies the checksum to ensure the data's integrity. If the data is found to be corrupted, Hadoop retrieves an alternate replica of the data.

8.What if Namenode fails?

Answer:

If the NameNode in a Hadoop cluster fails, it can result in a significant disruption to the cluster's operations because the NameNode is responsible for maintaining metadata and coordinating access to data stored in HDFS (Hadoop Distributed File System). Here's what happens when a NameNode fails:

Loss of Metadata Availability: The primary consequence of a NameNode failure is the loss of access to metadata. This metadata includes information about the file system namespace, file permissions, file-to-block mappings, and other essential details required for file operations.

Cluster Downtime: Without a functioning NameNode, the Hadoop cluster may experience downtime or reduced functionality. Clients attempting to read or write data will be unable to do so because they won't be able to locate the necessary metadata.

Data Accessibility: Despite the NameNode failure, the actual data stored in HDFS remains intact on the DataNodes. However, without metadata accessibility, clients won't be able to access this data until the NameNode is restored or a failover procedure is initiated.

Recovery Procedures:

Manual Intervention: In the absence of automated failover mechanisms like High Availability (HA), recovery from a NameNode failure typically involves manual intervention by the Hadoop administrator.

Backup and Restore: Administrators may restore the NameNode from a backup copy, which includes the file system metadata. This process can take time, especially for large clusters, and may result in extended downtime.

Edit Log Replay: In addition to restoring from a backup, administrators may need to replay the edit logs to recover any changes made to the file system since the last backup. This ensures that no data is lost during the recovery process.

High Availability Configuration: To minimize downtime and improve resilience, Hadoop clusters often employ High Availability configurations for the NameNode. In this setup, multiple NameNodes are configured in an active-standby configuration. If the active NameNode fails, the standby NameNode automatically takes over, ensuring continuity of operations with minimal disruption.

9. Why is block size 128 MB? What if I increase or decrease the block size

Answer:

The default block size in Hadoop Distributed File System (HDFS) is indeed 128 MB, but it's not a fixed rule. The block size was chosen based on several factors, including considerations of efficiency, performance, and scalability. However, the block size can be adjusted based on the specific needs and characteristics of your data and workload. Here's why the block size is typically set to 128 MB and what might happen if you increase or decrease it:

Reasons for a 128 MB Block Size:

Efficient Disk I/O: Larger block sizes reduce the overhead associated with managing a large number of small files. With a 128 MB block size, HDFS can read or write larger amounts of data sequentially, which is typically more efficient than handling numerous small blocks.

Reduced NameNode Load: With larger block sizes, the NameNode, which manages metadata, has to track fewer blocks overall, reducing its memory and processing requirements.

Better Network Throughput: Larger block sizes can lead to better network throughput because each block is transferred as a unit, and larger transfers are generally more efficient.

Data Locality: Hadoop's data processing model benefits from data locality, where computation is performed close to the data. Larger block sizes increase the likelihood that computation tasks will operate on data stored locally on the same DataNode, improving performance.

Effects of Increasing or Decreasing Block Size:

Increasing Block Size:

Considerations: If your workload involves large files and sequential processing, increasing the block size could be beneficial. However, consider the potential impact on storage efficiency and the NameNode's memory requirements.

Decreasing Block Size:

Considerations: If your workload involves many small files or random access patterns, decreasing the block size could be beneficial. However, be mindful of the potential impact on NameNode scalability and overall system performance.

10. Small file problem

Answer:

The small file problem arises when a large number of small files are stored in a distributed file system like HDFS. It leads to increased metadata overhead, inefficient space utilization, reduced data locality, and higher NameNode load. To mitigate this problem, consider consolidating small files, using optimized file formats, compression, or partitioning data.

11.What is Rack awareness?

Answer:

Rack awareness is a feature in Hadoop's HDFS (Hadoop Distributed File System) and other distributed storage systems that enhances data reliability and network efficiency by considering the physical network topology when placing data replicas. In Hadoop, a rack is a collection of DataNodes (servers) that are physically located in close proximity within the same network switch or network segment.

The concept of rack awareness ensures that when HDFS replicates data blocks for fault tolerance, it places replicas on different racks to minimize the risk of data loss due to rack-level failures. By distributing replicas across multiple racks, Hadoop can withstand rack failures while still ensuring data availability.

12.What is SPOF ? how its resolved ?

Answer:

SPOF stands for Single Point of Failure, which refers to a component or system that, if it fails, will cause the entire system to fail. It's a vulnerability in the system where the failure of a single part can result in a complete outage or loss of service.

Resolving a SPOF involves implementing redundancy or fault-tolerant measures to eliminate or mitigate the risk posed by the single point of failure. Some common strategies for resolving SPOFs include:

**Redundancy**: Introducing redundant components or systems to provide backup in case of failure. This can involve having multiple servers, network paths, or storage devices that can take over if the primary component fails.

**Load Balancing**: Distributing workload across multiple components to avoid overloading any single component and to ensure that if one component fails, the workload can be handled by others.

**Failover Mechanisms**: Implementing failover mechanisms that automatically switch to backup components or systems when a failure is detected. This can include hot standby systems that are ready to take over immediately or warm standby systems that require some time to activate.

**High Availability Architectures**: Designing systems with high availability in mind, using redundant components, load balancing, and failover mechanisms to minimize downtime and ensure continuous operation even in the face of failures.

**Monitoring and Alerting**: Implementing monitoring tools to continuously monitor the health and performance of components and systems. Alerting mechanisms can notify administrators of potential issues or failures so they can take proactive action to resolve them before they cause downtime.

**Regular Maintenance and Testing**: Performing regular maintenance and testing of systems to identify and address potential single points of failure. This can involve hardware upgrades, software patches, and testing failover procedures to ensure they work as expected.

13. Explain zookeeper?

Answer:

ZooKeeper is a centralized service for maintaining configuration information, providing distributed synchronization, and offering group services. It's used in distributed systems to ensure reliability, fault tolerance, and coordination among multiple processes. ZooKeeper stores data in a hierarchical namespace and provides primitives for synchronization, such as locks and barriers, making it essential for building robust distributed applications.

14. Difference between -put and -CopyFromLocal?

Answer the –put is a generic command being used to copy data from local files system to HDFS.

For ex: hadoop fs –put /path/to/local/ file/path/in/hdfs

15. What is erasure coding?

Answer: Erasure coding is a technique used to achieve fault tolerance and data redundancy by encoding data into a set of redundant fragments, allowing reconstruction of original data in case of data lost.

1. Redundance: Encoding redundancy using data blocks
2. Fault tolerance: Redundant fragments are distributed across multiple storage nodes.
3. Efficiency: Erasure coding offers higher storage effiicieny comparative to traditional methods.
4. Encoding and decoding algorithms such as reed Solomon are used to decode and recove data.
5. Use Cases: Erasure coding is particularly beneficial large scale distributed systems where storage efficiency, fault tolerance and and data durability are critical factors.

16. What is speculative execution?

Answer Speculative execution is a technique used in distributed computing environments such as hadoop to improve job performance and fault tolerance. It involves running duplicates instances of tasks concurrently across different nodes in cluster with the expectation of of the instances will complete faster than other .

17. Explain Yarn Architecture ?

Answer: Apache Yet Another resource negotiator is a resource management and job scheduling framework that forms the core of Hadoop version 2.x

Here are key components:

1. Resource Manager: The Resource manager is the central authority for managing resources and allocating them to the application in cluster.
2. Node Manager: Node managers are responsible for managing individual nodes in cluster (CPU, MEMORY).
3. Application Master: The Application master is the per application component responsible for coordinating the execution of a specific application.
4. Container: Each container runs single instance of an application task managed by application master.
5. Client: Client submits the application to the cluster for execution. It interacts with resource manager for submission, monitoring progress and retrieval of results.

18. How does ApplicationManager and Application Master differ ?

Answer: In the context of Yet another resource negotiator , the application manager and the application master are both key components of job execution and resource management.

They both serve at different level of abstraction in YARN.

1. Application Manager is the System level component responsible for managing the lifecycle of an entire application of YARN Cluster.
2. Application Master is an application specific component responsible for coordination the execution of a single application on a YARN cluster.

19. Explain Map reduce working?

Answer: MapReduce is a programming model and framework designed to process and analyze large datasets in a distributed computing environment efficiently. It consists of two main phases: the Map phase and the Reduce phase. Here's a detailed explanation of how MapReduce works:

Map Phase:

Input Data: MapReduce takes input data from a distributed storage system like Hadoop Distributed File System (HDFS). This input data is divided into smaller chunks called input splits.

Mapper Function: Each input split is processed by a mapper function, which is designed by the user. The mapper function reads the input data, processes it, and generates a set of intermediate key-value pairs.

Key-Value Pair Generation: The mapper function emits key-value pairs based on the processing logic. Typically, the key represents a unique identifier or a category, and the value contains relevant data associated with that key.

Shuffle and Sort Phase:

Data Shuffle: After the map phase, the MapReduce framework performs a shuffle and sort phase. It redistributes the intermediate key-value pairs generated by the mappers across the cluster based on the keys.

Grouping by Key: Key-value pairs with the same key are grouped together. This ensures that all values associated with a particular key are sent to the same reducer during the reduce phase.

Reduce Phase:

Reducer Function: Each reducer receives a subset of key-value pairs from the shuffle and sort phase. The reducer function, also defined by the user, processes these key-value pairs to produce the final output.

Aggregation and Analysis: The reducer aggregates and analyzes the data associated with each key. It may perform various operations such as summation, counting, averaging, or more complex computations.

Output Generation: The reducer generates the final output, typically in the form of key-value pairs or a structured format, which is written to the output storage system.

Fault Tolerance and Task Execution:

MapReduce frameworks like Hadoop handle task execution and fault tolerance automatically. If a mapper or reducer task fails, it is automatically restarted on another node in the cluster.

Intermediate results are stored persistently to handle failures. If a node fails during processing, the framework can recompute the lost results from the intermediate data stored on other nodes.

Completion and Cleanup:

Once all mappers and reducers have completed their tasks and generated the final output, the MapReduce job is marked as complete.

Any temporary resources or intermediate data generated during the job execution are cleaned up to free up cluster resources.

20. How many mappers are created for 1 GB file?

Answer: The number of mappers created for a 1GB file in a Map reduce job is influenced by various factors including the input format, block size and size of the file itself.

FOR 1 GB FILE APPROXIMATELY 8 MAPPERS OF 128 MB EACH WOULD BE CREATED.

21. How many reducers are created for 1 GB file?

Answer: The number of reducers that are created is not solely determined by the size of 1 G.B File. It depends upon multiple factors: Cluster configuration, level of parallelism and processing of the job

Cluster configuration: The number of available reducers slots in a cluster can influence the number of reducer assigned to a job.

Configuration Parameters: In Hadoop, you can explicitly specify the number of reducers for a MapReduce job using configuration parameters like **mapreduce.job.reduces.**

22. What is combiner?

Answer: A Combiner also known as as ‘mini reducer’ . it is an optional component in map reduce framework. It is primarily employed to optimise the data transfer between the map and reduce phases by performing local aggregation. The Combiner aggregates the key value pairs outputted by the mapper within the same node.

Local Aggregation: After the mapping phase, before the shuffle and sort phase, the combiner aggregates the intermediate key-value pairs outputted by the mapper within the same node. This aggregation reduces the amount of data that needs to be transferred over the network to the reducers.

Efficiency Improvement: By aggregating data locally on each mapper node, the combiner can significantly reduce the volume of data sent across the network during the shuffle and sort phase. This helps to alleviate network congestion and can improve overall job performance by minimizing data transfer overhead.

Same Functionality as Reducer: The combiner performs the same function as the reducer, but it operates locally on each mapper node. It aggregates key-value pairs with the same key to produce partial results, which are then sent to the reducers.

Optional Component: While the combiner can enhance the efficiency of MapReduce jobs, its usage depends on the specific requirements of the job. Not all MapReduce programs require a combiner, and its inclusion depends on factors such as the size of intermediate data, network bandwidth, and the complexity of the aggregation function.

Q23.What is partitioner?

Answer: In Hadoop, a partitioner is a crucial component of the MapReduce framework responsible for distributing the intermediate key-value pairs generated by the mappers to the appropriate reducers. It ensures that all key-value pairs with the same key end up in the same reducer, enabling the reducers to process related data together.

During the Map Phase, each mapper processes the portion of input data. The mapper emits a Key value pair.

ii. After the map phase the partitioner comes into play, it receives the key value pair emitted by the mapper and determines which reducer should receive each pair.

iii. Finally during the reduce phase , each reducer receives the key value pairs assigned to it by the partitioner and generates the Result set .