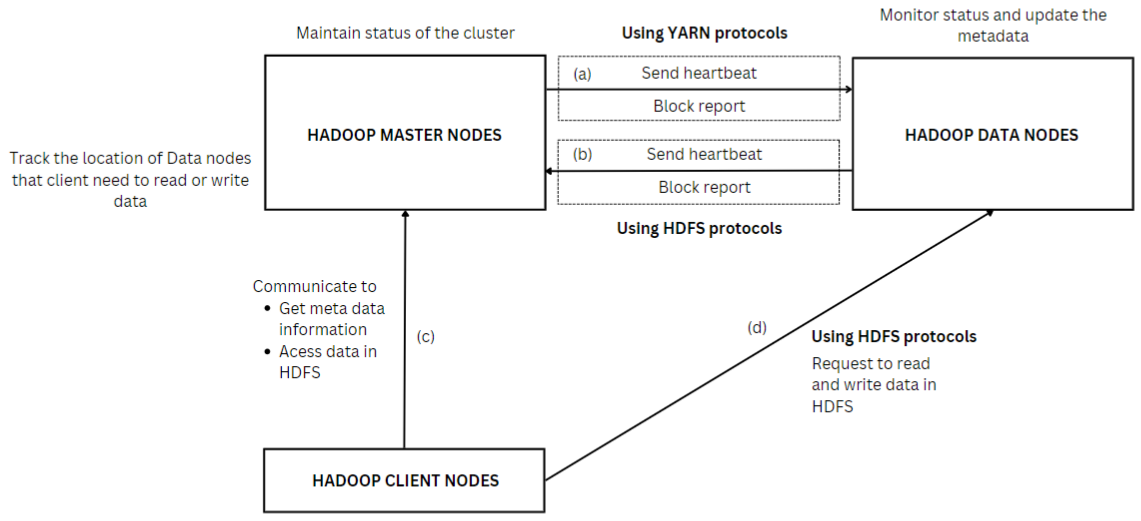
1. **Explain Hadoop Master, Data, and Client nodes. What are their purposes? How do they communicate?**

According to Databricks: "Apache Hadoop is an open source, Java-based software platform that manages data processing and storage for big data applications. The platform works by distributing Hadoop big data and analytics jobs across nodes in a computing cluster, breaking them down into smaller workloads that can be run in parallel.Some key benefits of Hadoop are scalability, resilience and flexibility. The Hadoop Distributed File System (HDFS) provides reliability and resiliency by replicating any node of the cluster to the other nodes of the cluster to protect against hardware or software failures."

Hadoop has 3 components, including Master, Data and Client nodes which are explained in detail:

- **Master node** controls and manages distribution of work of Hadoop cluster with two main components: NameNode and ResourceManagers. NameNode responsible for keeping track of metadata & maintaining file system 's namespace. ResourceManagers allocate resources.
- **Data node** is a worker node, performing functional tasks in the cluster such as reading & writing requests or replicating & deleting blocks by the NameNode.
- **Client node** is a machine where users submit their MapReduce jobs. It allows users to access data in HDFS and interact with the Hadoop cluster.

The communication between Hadoop Master, Data and Client nodes is explained in this diagram:



(a) Master nodes (ResourceManager) and Data nodes: ResourceManager sends heartbeat and block reports to Data nodes which Data nodes use to monitor status and update the metadata in case Client nodes send request to get access data in HDFS and need to write or read data.

(b) Master nodes (NameNodes) and Data nodes: Data nodes send heartbeat and block reports to NameNodes, which NameNodes use to maintain the status of the cluster and make execution decisions such as task assignment.
(c) Master nodes and Client Nodes: Client nodes send the request to get metadata information and access data in HDFS. Master nodes receive the request and send the request to the Data nodes to track the location of data nodes that clients need to read or write data.
(d) Client nodes and Data nodes: Client sends requests directly to Data nodes to read or write data.

2. **Conduct research regarding Spark. How does Spark run on top of Hadoop? Explain Spark Scala, PySpark, and SparkR?**

According to IBM: “Apache Spark (Spark) is an open-source data-processing engine for large data sets. It is designed to deliver the computational speed, scalability, and programmability required for Big Data—specifically for streaming data, graph data, machine learning, and artificial intelligence (AI) applications.”

To run on top of Hadoop, Spark uses Hadoop YARN as its resource manager (requesting CPU, memory, & network, and monitoring the application) to leverage the distributed storage and maintain high-level APIs libraries and in-memory processing features.

Because Spark supports developers in using various programming languages, it has APIs for these languages, including Spark Scala, PySpark and SparkR.

- **Spark Scala:** Spark Scala is a distributed computing framework and programming language that allows writing Spark applications using Java, and Python that shortens syntax and provides structural functional constructs for building complex data processing pipelines. It allows developers to process large-scale data across multiple machines in a cluster.
- **PySpark:** PySpark is a distributed computing framework, built on top of Spark’s Jave API and Py4J that allows writing Spark applications using Python. It provides built-in libraries such as NumPy, Pandas, Matplotlib, etc. to use in real-time data processing and machine learning.
- **SparkR:** SparkR is a distributed computing framework, built on top of Spark’s Java API and uses the RJava package that allows writing Spark applications using R. It provides built-in libraries such as dplyr, ggplot2, caret, etc. to use in real-time data processing and machine learning.

3. **Explain the differences between parallel and distributed computing?**

Parallel and distributed computing are used to speed up computations. They have some differences in their fundamentals:

Criteria	**Parallel computing**	**Distributed computing**
Architecture	Processors or cores within a single computer system work together to solve a problem.	Processors or cores connected through a network together to solve a problem.
	Processors work under a shared-memory model.	Processors work under a distributed-memory model.
Scalability	Limited by the available processors or cores in a single computer system	Better scalability because it is not limited by the available processors or cores in a single computer system, instead, it can add more nodes to satisfy the demands of specific tasks.
Communication	Through shared memory, therefore the speed is faster and direct	Through distributed memory, therefore the speed is slower and indirect
Fault tolerance	Failure of a single processor can affect the entire computation.	More fault-tolerance because nodes operate independently
Problem types	Computational intensity and data locality	Independent tasks
Application	Scientific computing, simulation, high-performance computing, etc.	Big data processing, cloud computing, distributed databases, etc.

4. How do Tez and Spark work as compared to MR?

Tez, Spark and MapReduce (MR) are data processing frameworks, helping process data on a massive scale. The difference between Tez, Spark and MR can be explained as the following:

Criteria	**Tez**	**Spark**	**MapReduce**
Programming model	Flexible Directed Acyclic Graph (FDAG)	DAG-based execution	Two-stage Map and Reduce operations
Performance	Faster than MR but lower than Spark	Fastest among three	Lowest among three
Data processing	Be useful for	Be useful for execution	Be useful for batch

types	in-memory processing and DAG-based execution	plans, batch, and streaming	processing but not for in-memory or real-time processing
Scalability	Be highly scalable for high-performance data processing	Be highly scalable for distributing processing	Be highly scalable for large dataset processing
Fault tolerance	Allows for pluggable fault tolerance	Has built-in fault tolerance through Resilient Distributed Dataset partitioning	Be dependent on data replication and speculative execution.
Convenience	More flexible but need understanding DAG-based architecture	Most convenient tool which supports multiple programming languages and built-in library to use	Need more manual optimizations

By comparing Tez, Spark and MapReduce, we can conclude that MapReduce is less flexible and easy to use than Tez and Spark. Spark is easiest to use and friendly for developers to access because it supports multiple programming languages and has a built-in library. Sparks are also available to handle multiple tasks that Tez or MR can only handle some of them, such as batch, interactive, streaming, real-time processing.

5. Compare and contrast Hive and Hbase

According to Amazon Web Services:

- "Apache Hive is a distributed, fault-tolerant data warehouse system that enables analytics at a massive scale. A data warehouse provides a central store of information that can easily be analyzed to make informed, data driven decisions. Hive allows users to read, write, and manage petabytes of data using SQL."
- "Apache HBase is an open-source, NoSQL, distributed big data store. It enables random, strictly consistent, real-time access to petabytes of data. HBase is very effective for handling large, sparse datasets."

Both Apache Hive and Apache HBase are data processing, storage, and analysis at a massive scale. They have the difference in their functions:

Criteria	Apache Hive	Apache HBase
Run on top of	Hadoop	HDFS
Query language	Uses a built-in SQL-like query	Not use a built-in SQL-like query

	language: HiveQL	language, instead, provides APIs for languages to interact with data
Latency	Batch processing	Low-latency operations
	Analytical query on HDFS that gets translated to MapReduce jobs	Real-time frequently or randomly access data to analyze on its database
	Support bucketing in data organization and retrieval	Handle data partitioning to tables to split into column families
Functions	Data ingestion, data organization, data transformation, data analysis and data export	Data model design, data ingestion, data organization, data access, and data maintenance
Applications	Data warehousing, analytics and reporting	Real-time data access and low-latency operations

6. Explain the role of the Hadoop ecosystem in Big Data

The role of the Hadoop ecosystem is to solve Big Data problems by providing suites, tools, frameworks and model design. Each component of the Hadoop ecosystem is used to help leverage Big Data management and data processing.

- Hadoop Distributed File System (HDFS): This distributed system helps leverage the storage capacity and fault tolerance ability when managing high-volume data. Therefore, HDFS plays an essential role in keeping the velocity & veracity of Big Data.
- MapReduce: This programming model helps leverage data processing and simplify writing distributed applications. It plays a key role in maintaining the velocity of Big Data.
- Yet Another Resource Negotiator (YARN): As a resource management, YARN helps leverage the cluster utilization, and support various data processing frameworks, allowing developers to use various programming languages and improve data processing steps.
- Other components of Hadoop can be mentioned such as Apache HBase, Spark, Oozie, Drill, Zookeeper, Flume & Kafka. They are useful for data processing in complex workflows and massive data scales. For example, Apache Spark is widely used in batch processing, real-time streaming, and ML.
- In conclusion, the Hadoop ecosystem has an essential role in Big Data because it helps organizations scale infrastructure and tolerate faults while storing and processing data.

The variety of tools, platforms with built-in libraries and support of programming languages help address multiple Big Data challenges nowadays.