**Assignment-5**

**Q.1) Explain concept of Kafka with it's Architecture?**

|  |
| --- |
| Kafka Architecture - Apache Kafka APIs |

* **Concept:** Kafka acts as a distributed streaming platform that allows the continuous flow of data between systems or applications in a publish-subscribe manner. Producers publish data to Kafka topics, and consumers subscribe to these topics to receive and process the data. Kafka persists the data for a configurable period, allowing both real-time and batch processing of the data.
* **Architecture:** Kafka's architecture consists of several components that work together to provide its streaming capabilities. These components include producers, topics, partitions, brokers, and consumers:
* **Producers:** Producers are the data publishers that push messages to Kafka topics. They are responsible for choosing which topic to publish the message to and write data to Kafka in a fault-tolerant manner. Producers can be part of any application or system that generates data.
* **Topics:** Topics are named feeds or categories to which producers publish messages. A topic represents a specific stream of data in Kafka. Topics are divided into partitions to allow for scalability and parallel processing. Each message within a topic is assigned a unique offset, representing its position within the partition.
* **Partitions:** Topics are divided into multiple partitions to enable distributed storage and parallel processing. Each partition is an ordered and immutable sequence of messages. Partitions allow Kafka to scale horizontally by distributing the data across multiple brokers in a cluster.
* **Brokers:** Brokers are the servers that form the Kafka cluster. They store and replicate the partitions of Kafka topics. Each broker is responsible for handling read and write requests from producers and consumers. Kafka clusters typically consist of multiple brokers for fault tolerance and scalability.
* **Consumers:** Consumers subscribe to one or more topics and read data from Kafka. They consume messages in the order they are stored within each partition. Consumers can be part of any application or system that needs to process or analyze the data published by producers. Consumers can be part of consumer groups, where each group processes a subset of the partitions of a topic, enabling parallel processing of data.
* **ZooKeeper:** Kafka relies on Apache ZooKeeper for cluster coordination, maintaining metadata, and detecting broker failures. ZooKeeper tracks the status of brokers, topics, and partitions and helps in leader election and partition reassignment in case of failures.
* **Replication and Fault Tolerance:** Kafka provides fault tolerance by replicating the data across multiple brokers. Each partition has a leader and multiple replicas. The leader handles read and write requests, while the replicas serve as backups. In case of a broker failure, one of the replicas is automatically elected as the new leader to ensure uninterrupted data availability.
* **Scalability and Performance:** Kafka's architecture allows for horizontal scalability by adding more brokers to the cluster. With increasing topics and partitions, Kafka can handle high message throughput and support real-time data streaming at massive scales.

**Q.2) What are the features and components of Kafka?**

* **Features:**
* **Publish-Subscribe Messaging:** Kafka follows a publish-subscribe model where producers publish messages to topics, and consumers subscribe to those topics to receive and process the messages. This decoupled architecture enables multiple consumers to read from the same topic concurrently.
* **Fault Tolerance:** Kafka provides fault tolerance through data replication. Each topic partition has multiple replicas, with one replica serving as the leader and others as followers. If a leader fails, one of the followers is automatically promoted as the new leader, ensuring data availability and continuity.
* **Scalability:** Kafka is designed to scale horizontally by distributing data across multiple brokers. Additional brokers can be added to a Kafka cluster to handle increasing data loads and throughput. Scaling can be achieved without downtime or interruption to the streaming process.
* **High Throughput:** Kafka is built for high-performance data streaming. It efficiently handles large volumes of data and can sustain high message throughput by leveraging features such as sequential disk I/O and in-memory caching.
* **Durability and Persistence:** Kafka persists data for a configurable period, allowing both real-time and batch processing. The data is stored in a distributed, fault-tolerant manner, making it durable and reliable.
* **Stream Processing:** Kafka supports stream processing by integrating with frameworks like Apache Flink, Apache Samza, and Apache Spark. These frameworks enable real-time data processing and analytics on the streaming data.
* **Exactly-Once Semantics:** Kafka supports end-to-end exactly-once message processing semantics, ensuring that data is processed and delivered exactly once, even in the presence of failures.
* **Components:**
* **Producers:** Producers are responsible for publishing messages to Kafka topics. They write data to specific topics and partitions, which are then made available to consumers for processing.
* **Topics:** Topics are named feeds or categories to which producers publish messages. They represent a specific stream of data in Kafka. Topics can be partitioned to achieve scalability and parallel processing.
* **Partitions:** Topics are divided into partitions, allowing for horizontal scalability and parallelism. Each partition is an ordered and immutable sequence of messages. Partitions enable efficient distribution and processing of data across multiple brokers.
* **Brokers:** Brokers are the servers or nodes in a Kafka cluster. They store and replicate the partitions of Kafka topics. Each broker is responsible for handling read and write requests from producers and consumers. A Kafka cluster consists of multiple brokers for fault tolerance and scalability.
* **Consumers:** Consumers subscribe to one or more topics and read data from Kafka. They process the messages in the order they are stored within each partition. Consumers can be part of consumer groups, where each group processes a subset of the partitions of a topic, allowing for parallel processing of data.
* **ZooKeeper:** Kafka relies on Apache ZooKeeper for cluster coordination, maintaining metadata, and detecting broker failures. ZooKeeper tracks the status of brokers, topics, and partitions and assists in leader election and partition reassignment.

**Q.3) Explain concept and features of Apache spark?**

* **Concept:** At its core, Apache Spark introduces the concept of Resilient Distributed Datasets (RDDs), which are immutable distributed collections of objects. RDDs represent the fundamental data structures in Spark and provide a high-level abstraction for data processing. Spark allows users to perform distributed computing tasks by distributing the workload across multiple machines in a cluster.
* **Features:**
* **Speed:** Spark is known for its speed and performance. It achieves this through in-memory computing, caching intermediate data in memory, and minimizing disk I/O operations. By processing data in-memory, Spark can deliver much faster processing times compared to traditional disk-based systems.
* **Ease of Use:** Spark provides high-level APIs in multiple programming languages such as Scala, Java, Python, and R, making it accessible to a wide range of developers. It also offers an interactive shell for exploring and prototyping data processing tasks.
* **Fault Tolerance:** Spark provides built-in fault tolerance mechanisms. If a node in the cluster fails, Spark can recover lost data and continue processing without interruption. It achieves this through RDDs, which track the lineage of transformations applied to the data and can reconstruct lost partitions.
* **Scalability:** Spark is highly scalable and can efficiently process large datasets by distributing the computation across a cluster of machines. It supports vertical scalability (adding more resources to a single machine) as well as horizontal scalability (adding more machines to the cluster).
* **Versatility:** Spark offers a wide range of libraries and APIs that extend its functionality for various data processing tasks. These include Spark SQL for working with structured data, Spark Streaming for real-time data processing, MLlib for machine learning, and GraphX for graph processing.
* **Data Processing Capabilities:** Spark supports batch processing, where it can process large volumes of data in parallel. It also supports streaming data processing, enabling real-time analytics on data streams. Spark's unified processing engine allows developers to seamlessly switch between batch and streaming processing modes.
* **Integration:** Spark integrates with other popular big data tools and frameworks such as Hadoop Distributed File System (HDFS), Apache Hive, Apache HBase, and more. It can read data from various data sources and write results to different storage systems.
* **Advanced Analytics**: Spark provides advanced analytics capabilities, including machine learning (MLlib) and graph processing (GraphX). MLlib offers a range of machine learning algorithms and tools for tasks such as classification, regression, clustering, and recommendation. GraphX enables graph processing and analysis, making it useful for social network analysis, fraud detection, and more.

**Q.4) Describe features and components of Apache spark?**

* **Components of Spark:**

|  |
| --- |
| https://media.geeksforgeeks.org/wp-content/uploads/20200616181455/spark2.png |

* **Spark Core:** All the functionalities being provided by Apache Spark are built on the highest of the Spark Core. It delivers speed by providing in-memory computation capability. Spark Core is the foundation of [parallel and distributed processing](https://www.geeksforgeeks.org/difference-between-parallel-computing-and-distributed-computing/) of giant dataset. It is the main backbone of the essential I/O functionalities and significant in programming and observing the role of the spark cluster. It holds all the components related to scheduling, distributing and monitoring jobs on a cluster, Task dispatching, Fault recovery. The functionalities of this component are:
  + It contains the basic functionality of spark. (Task scheduling, memory management, fault recovery, interacting with storage systems).
  + Home to API that defines RDDs.
* **Spark SQL Structured data:** The Spark SQL component is built above the spark core and used to provide the structured processing on the data. It provides standard access to a range of data sources. It includes Hive, JSON, and JDBC. It supports querying data either via SQL or via the hive language. This also works to access structured and semi-structured information. It also provides powerful, interactive, analytical application across both streaming and historical data. Spark SQL could be a new module in the spark that integrates the relative process with the spark with programming API. The main functionality of this module is:
  + It is a Spark package for working with structured data.
  + It Supports many sources of data including hive tablets, parquet, json.
  + It allows the developers to intermix SQK with programmatic data manipulation supported by RDDs in python, scala and java.
* **Spark Streaming:** Spark streaming permits ascendible, high-throughput, fault-tolerant stream process of live knowledge streams. Spark can access data from a source like a flume, TCP socket. It will operate different algorithms in which it receives the data in a file system, database and live dashboard. Spark uses Micro-batching for real-time streaming. Micro-batching is a technique that permits a method or a task to treat a stream as a sequence of little batches of information. Hence spark streaming groups the live data into small batches. It delivers it to the batch system for processing. The functionality of this module is:
  + Enables processing of live streams of data like log files generated by production web services.
  + The API’s defined in this module are quite similar to spark core RDD API’s.
* **Mllib** **Machine Learning:** MLlib in spark is a scalable Machine learning library that contains various machine learning algorithms. The motive behind MLlib creation is to make the implementation of machine learning simple. It contains machine learning libraries and the implementation of various algorithms. For example, [clustering](https://www.geeksforgeeks.org/clustering-in-machine-learning/), [regression](https://www.geeksforgeeks.org/types-of-regression-techniques/), [classification](https://www.geeksforgeeks.org/ml-classification-vs-regression/) and [collaborative filtering](https://www.geeksforgeeks.org/collaborative-filtering-ml/).
* **GraphX graph processing:** It is an API for graphs and graph parallel execution. There is network analytics in which we store the data. Clustering, classification, traversal, searching, and pathfinding is also possible in the graph. It generally optimizes how we can represent vertex and edges in a graph. GraphX also optimizes how we can represent vertex and edges when they are primitive data types. To support graph computation, it supports fundamental operations like subgraph, joins vertices, and aggregate messages as well as an optimized variant of the Pregel API.

**Q.5) Explain concept and Architecture of Kerbors authentication?**

* It has proved to be one of the essential components of client or server applications. It is also used in various fields for network security and providing mutual authentication.
* The following diagram shows the workflow of the Kerberos protocol:

|  |
| --- |
| Kerberos Architecture |

* **Step 1:** Initially, there is an authentication request from the client. The user requests TGS from the authentication server.
* **Step 2:** After the client’s request, the client data is validated by the KDC. The authentication server verifies the client and the TGS from the database. The authentication server then generates a cryptographic key (SK1) after checking both values and implementing the hash of the password. The authentication server also computes a session key. This session key uses the secret key of the client (SK2) for encryption.
* **Step 3:** The authentication server then creates a ticket that consists of the ID, network address, secret key, and lifetime of the client.
* **Step 4:** The decryption of the message is then performed by the client by using the client’s secret key.
* **Step 5:** Now, the client demands entrance into the server by using TGS. The TGS creates a ticket that acts as an authenticator here.
* **Step 6:** Another ticket is generated by KDC for the file server. Then, the TGS decrypts the ticket for obtaining the secret key initiated by the client. It checks the network address and ID by decrypting the authenticator. If the client ID and the network address match successfully, then KDC shares a service key with the client and the server.
* **Step 7:** The client utilizes the file ticket for authentication. The message is decrypted by using SK1 to obtain SK2. Again, the TGS generates a new ticket to send to the target server.
* **Step 8:** Here, the target server decrypts the file ticket by using the secret key. After that, the server performs checks on the client details by decrypting SK2. The target server also checks the validity of the ticket. Finally, when all of the client’s encrypted data is decrypted and verified, the server authenticates the client to use the services.
* This is how we use and implement the Kerberos protocol for securing a system and client-server interactions.

**Q.6) Describe Features and components of Kerbors authentication?**

|  |
| --- |
| Kerberos Components |

* **Features:**
* **Enhanced security**
* **Access control**
* **Transparency and auditability**
* **Shared authentication**
* **Limited-lifetime ticket**
* **Scalability**
* **Reusable authentications**
* **Components of kerbors authentication:** Kerberos mainly provides two services. They are:
* Authentication service
* Ticket-granting service

For providing these services, Kerberos uses its various components. The following principal components that are used for authentication:

1. **Client:** The client helps to initiate a service request for communicating with the user.
2. **Server:** All the services that are required by the user are hosted by the server.Enhance your cybersecurity skills with our comprehensive [Cyber Security Training](https://intellipaat.com/cyber-security-course-certification/) program!
3. **Authentication Server (AS)**: As the name suggests, AS is used for the authentication of the client and the server. AS assigns a ticket through Ticket Granting Ticket (TGT) to the client. The assigned ticket ensures the authentication of the client to other servers.
4. **Key Distribution Center (KDC)**: There are three parts to the Kerberos authentication service:

* Database
* Ticket Granting Server (TGS)
* Authentication Server (AS) These parts reside in a single unit known as the Key Distribution Center.

1. **Ticket Granting Server (TGS):** This server provides a service to assign tickets to the user as a unique key for authentication. There are unique keys that are used by the authentication server and the TGS for both clients and servers.

* **Client or User Secret Key:** It is the hash of the password set by the user that acts as the client or user secret key.
* **TGS Secret Key:** It is the secret key that helps in deciding TGS.
* **Server Secret Key:** It helps to determine the server that provides the services.

**(PPT Give you more understanding than PDF)** The material for the PDF has been compiled from various sources such as books, tutorials (offline and online), lecture notes, several resources available on Internet. The information contained in this PDF is for general information and education purpose only. While we endeavor to keep the information up to date and correct, **we make no representation of any kind about the completeness and accuracy of the material.** The information shared through this PDF material should be used for educational purpose only.

**Thank you…**