Big Data - Interview Questions

What is MapReduce?

- MapReduce is a programming model for processing large datasets in parallel using multiple machines.
- MapReduce enables large scale data analysis and data mining using clusters of machines, taking much less time compared to other data processing methods.

Describe the programming model of MapReduce?

- MapReduce processes datasets in two phases - the Map phase and the Reduce phase. Each phase has key-value pairs as input and output.
    - **Map phase** - The map function takes an input key-value pair and produces a set of intermediate key-value pairs.
    - **Reduce phase** - Reduce phase takes the output key-value pairs from the Map phase as input and does further processing on that data to compute and generate the final required output.

What are input data splits in MapReduce?

- The MapReduce framework splits input files into multiple pieces of fixed sizes, say 16 MB to 64 MB each.
- The splits are then sends as input to multiple map tasks, which can be run parallelly on a cluster of machines.

What is partitioning in MapReduce?

- The intermediate key-value pairs generated by the map task are buffered in memory and are periodically written to the local disc on which the map task runs.
- This data is partitioned into multiple regions on the disk, based on the hash of the key and number of reducers, by the partitioning function.
- The locations of the buffered data on the local disc are send to reduce tasks, which use remote procedure calls to pull the data and process them.

What is shuffling and sorting in MapReduce?

- The map tasks generate intermediate key-value pairs and store them on partitions on the local discs.
- Shuffling is the process of transferring this intermediate data generated on multiple machines to reducers.
- This data fetched by the reducers are sorted and grouped by the intermediate key, before the data is sends as input to the reduce task.

How does MapReduce tolerate node failures?

- MapReduce is resilient to machine failures. The master pings the workers periodically, and marks a worker as failed if it does not receive a response within a specific time period.

- A map task or reduce task that was being processed by the failed worker is reset to idle state, so that the task is processed again by another worker.

What are counters in MapReduce?

- Counter is a facility provided in the MapReduce framework that can be used to count various events aggregated across the cluster machines.
- The counter is created in the MapReduce user program and can be incremented in the map or reduce task.

What are combiner functions?

- In a multi-node clustered environment, the efficiency of MapReduce jobs is limited by the bandwidth availability on the cluster.
- In such environments, combiner functions minimize the data transferred between the map and reduce tasks.
- The combiner function acts on the output from the map task, and minimizes the data before it is sent to the Reduce task.

What is Apache Flume?

- Apache Flume is an open-source platform to efficiently and reliably collect, aggregate and transfer massive amounts of data from one or more sources to a centralized data source.

What are the key components used in Flume data flow?

- Flume flow has three main components - Source, Channel and Sink
    - **Source:** Flume sources are customized to handle data from specific sources.
    - **Channel:** Flume sources inject data and store them into one or more channels
    - **Sink:** Flume sinks removes the data stored in channels and puts it into a central repository such as HDFS or Hive.

What is flume Agent?

- A Flume agent is a JVM process that hosts the components through which events flow from an external source to either the central repository or to the next destination.

How is reliability of data delivery ensured in Flume?

- Flume uses a transactional approach to guarantee the delivery of data.
- Events or data is removed from channels only after they have been successfully stored in the terminal repository for single-hop flows, or successfully stored in the channel of next agent in the case of multi-hop flows.

How do you handle agent failures?

- Flume agent goes down then all flows hosted on that agent are aborted.

- Once the agent is restarted then flow will resume. If the channel is set up as in-memory channel then all events that are stored in the channels when the agent went down are lost.
- But channels setup as file or other stable channels will continue to process events where it left off.

What is Apache Kafka?

- Apache Kafka is a distributed streaming platform that provides the following key capabilities.
    - **Messaging** - Kafka provides the concept of Topics, which are streams of data records.
    - **Storage** - Kafka by default stores streams of data in a distributed, replicated and fault-tolerant cluster.
    - **Processing** - Apache Kafka provides the Kafka Streams API, which enables processing and transforming live data streams.

What are some use cases where Kafka can be used?

- **1. Messaging** - Kafka can be used in distributed messaging systems, instead of message brokers such as IBM MQ, ActiveMQ, or RabbitMQ.
- **2. Log Aggregation** - Kafka can be used to aggregate logs from different servers and put in a central location.
- **3. Stream processing** - Kafka streams API facilitates this data streaming process.
- **Event based architectures** - Kafka can be used in event-based architectures, where events are tracked and acted upon as time-ordered sequence of events.
- **Commit log** - Kafka can be used as an external commit-log for nodes of a distributed system.

What are the key components of Kafka?

- **1. Kafka Cluster** - Kafka cluster contains one or more Kafka brokers (servers) and balances the load across these brokers.
- **2. Kafka Broker** - Kafka broker contains one or more Kafka topics.
- **3. Kafka Topics** - Kafka topics are categories or feeds to which streams of messages are published to.
- **4. Kafka Partitions** - A Kafka topic can be split into multiple partitions.
- **5. Kafka Offsets** - Messages in Kafka partitions are assigned sequential id number called the offset.
- **6. Kafka Producers** - Kafka producers are client applications or programs that post messages to a Kafka topic.
- **7. Kafka Consumers** - Kafka consumers are client applications or programs that read messages from a Kafka topic.

What is replication factor?

- Replication factor specifies the number of copies that each partition of a topic will have.
- A replication factor of 2 will create 2 copies (replicas) of each partition. A replication factor of 3 will create 3 copies (replicas) of each partition.
- One of the replicas will be the leader, the remaining will be the followers.

What is an in-sync replica (ISR)?

- An in-sync replica is a server that has the latest messages for a given partition.
- A leader is always an in-sync replica since it always has the latest messages.
- A follower replica is an in-sync replica only if it is up-to-date with the latest messages for that partition.

What is the significance of acks setting on a Kafka producer?

- There are three values for acks - 0, 1, and all
- **acks=0** - Producer considers the write to be successful immediately after sending the message - it does not wait for response from the broker
- **acks=1** - Producer considers the write to be successful when the message is committed to the leader
- **acks=all** - Producer considers the write to be successful only after the message is committed to all the in-sync replicas.

How is Kafka used as a stream processing?

- Kafka can be used to consume continuous streams of live data from input Kafka topics, perform processing on this live data, and then output the continuous stream of processed data to output Kafka topics.
- For performing complex transformations on the live data, Kafka provides a fully integrated Streams API.

What is the role of Zookeeper in Apache Kafka framework?

- Zookeeper performs four important functions in Kafka

1. Manages cluster memberships - Zookeeper maintains, and keep track of the status, of all the brokers that are a part of the cluster.

2. Manages configuration of topics - Zookeeper maintains and manages the configuration of topics.

3. Elects the leader broker - Zookeeper elects which Kafka broker will act as the leader for a partition.

4. Manages the service discovery of brokers - Each broker knows about all the other brokers in the cluster.

How do you create a topic using the Kafka command line client?

```
/** create topic */
> bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 3 --partitions 1 --topic test_topic
```

How do you send messages to a Kafka topic using Kafka command line client?

- Kafka comes with a command line client and a producer script *kafka-console-producer.sh* that can be used to take messages from standard input on console and post them as messages to a Kafka queue.

```
> bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test-topic
>Test Message 1
>Test Message 2
```

How do you consume messages from a Kafka topic and output the messages to console using Kafka command line client?

- Kafka comes with a command line client and a consumer script *kafka-console-producer.sh* that can be used to consume messages from a Kafka topic and output them to the standard console.

```
> bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test-topic --from-beginning
Test Message 1
Test Message 2
```

What are the core APIs provided in Kafka platform?

- Kafka provides the following core APIs
    - **Producer API** - An application uses the Kafka producer API to publish a stream of records to one or more Kafka topics.
    - **Consumer API** - An application uses the Kafka consumer API to subscribe to one or more Kafka topics and consume streams of records.
    - **Streams API** - An application uses the Kafka Streams API to consume input streams from one or more Kafka topics, process and transform the input data, and produce output streams to one or more Kafka topics.
    - **Connect API** - An application uses the Kafka connect API to create producers and consumers that connect Kafka topics to existing applications or data systems.

What is Kafka Connect?

- Kafka Connect is a tool to stream data between Kafka and other systems scalable and reliably.
- Kafka Connect defines a common framework for Kafka connectors, can be deployed in distributed or standalone modes, and is ideal for bridging streaming and batch data systems.

What is a Kafka Streams?

- Kafka streams is a client library for processing input data from input Kafka clusters, transforming the data, and sending the transformed data to output Kafka clusters.

How do you setup a multi-broker Kafka cluster?

- To setup a multi-broker Kafka cluster you have to do the following 3 tasks.

1. Create a server.properties file for each of the servers you want in the cluster. At a minimum, you have to change the server id, port and log location for each of the property files.

2. Start the zookeeper server.

3. Start the servers in the cluster one by one using the property file of each server.

What is Hive?

- Hive is a data warehousing framework based on Apache Hadoop
- Hive provides a SQL-like (HiveQL) interface to query data from Hadoop based databases and file systems.
- Hive is suitable for batch and data warehousing tasks.
- HiveQL queries are implicitly converted to MapReduce, Apache Tez and Spark jobs.

What are the key features of Hive?

- Hive provides a SQL-like interface to query data from Hadoop based databases and file systems.
- Hence Hive enables SQL based portability on Hadoop.
- Hive supports different storage types such as HBase, text files etc.

What are the key architectural components of Hive?

- Following are the major components of a Hive architecture.
    - **meta store -** Stores the metadata for each of the Hive tables.
    - **Driver -** Controller that receives the HiveQL statements.
    - **Compiler -** Compiles the HiveQL query to an execution plan.
    - **Optimizer -** Performs transformations on the execution plan.
    - **Executor -** Executes the tasks after compilation and execution.
    - **UI -** Command line interface to interact with Hive.

How do you create Hive tables using HiveQL?

- You can create a Hive table using the DDL 'CREATE TABLE' statement.

```
CREATE TABLE employees (fname STRING, age INT)
```

How do you create Hive tables that can be partitioned using HiveQL?

- You can create a Hive table that can be partitioned using the DDL 'CREATE TABLE... PARTITIONED BY... ' statement?

```
- hive> CREATE TABLE employees (fname STRING, lname STRING, age INT) PARTITIONE
  D BY (ds STRING);
```

How do you list tables in Hive?

- You can list Hive tables using the DDL statement 'SHOW TABLES' statement?

```
SHOW TABLES
```

How do you list columns of a table in Hive?

- You can list columns of a tables using the DDL statement 'DESCRIBE'?

```
DESCRIBE employees
```

How do you add new columns to a table in Hive?

- You can add new columns to a table in Hive by using the DDL statement 'ALTER TABLE... ADD COLUMNS'?

```
hive> ALTER TABLE employees ADD COLUMNS (lname STRING);
```

How do you load data from flat files into Hive?

You can load data from flat files into Hive by using the command 'LOAD DATA... INTO TABLE' command.

```
//Load from local files
hive> LOAD DATA LOCAL INPATH './files/employees.txt' OVERWRITE INTO TABLE employees;

//Load from Hadoop files
hive> LOAD DATA INPATH './files/employees.txt' OVERWRITE INTO TABLE employees;
```

Hadoop

What is Hadoop?

- Hadoop is an implementation of MapReduce programming model, that enables distributed and parallel processing of large data sets across clusters of computers using simple programming models.
- Hadoop is designed to scale up from single server to clusters of thousands of machines, utilizing the local computation and storage of each server.
- This enables Hadoop to process massive amounts of data in times that cannot be done with other distributed computing processes.
- Hadoop is designed to detect node failures in the clusters and handle the failure at the application layer.

What are the core modules of Apache Hadoop framework?

- Apache Hadoop contains the following core modules.

1. Hadoop Common: Hadoop Common contains the common utilities and libraries that support the other Hadoop modules.

2. Hadoop Distributed File System (HDFS): Hadoop HDFS is a distributed file system that provides high-throughput and fault-tolerant access to application data.

3. Hadoop YARN: Hadoop YARN is a framework for job scheduling and cluster resource management.

4. Hadoop MapReduce: Hadoop MapReduce is the implementation of the MapReduce programming model that enables parallel processing of large data sets.

What do you understand by Hadoop distributions? What are some of the commonly used Hadoop distributions? Which Hadoop distribution di you use in your pervious projects.

- Hadoop is an open-source project developed by the Apache community.
- In addition to core Hadoop, Apache has many other projects that are related to Hadoop and the Big Data ecosystem.
- Third party vendors package together - the core Hadoop Apache framework. related projects, and add propriety utilities and capabilities on top and package into a unified project.

What class does Hadoop framework provide to configure, execute and manage MapReduce jobs?

- Hadoop provides the 'org.apache.hadoop.mapreduce.Job' class that is used to configure a MapReduce job, submit the job, control the execution of the job and query the state of the job.

What are the key steps in a MapReduce Job program?

1. Create an instance of Job class.

2. Set job specific parameters.

3. Set mapper, reducer and optionally a combiner class for the job.

4. Set input and output paths.

5. Submit job and poll for completion.

What are the key methods provided in the Mapper class?

- Mapper class provides the following methods.
    - **setup ()** – setup () is called once at the beginning of the map task and is used to setup one-time resources for the task.
    - **Map ()** – map () is called once for each key/value pair of the input split and perform the map functionality

- **Cleanup ()** – cleanup () is called once at the end of the map task and is used to clean up resources.
- **Run ()** – run () method can be overridden to get more complete control over the execution of the Mapper

What are the three primary phases performed in a Reducer?

- Following are the three primary phases of a Reducer.
  - **Shuffle -** In this phase the Reducer copies the sorted output from each Mapper using HTTP across the network.
  - **Sort -** In this phase the framework merge sorts Reducer inputs by keys (since different Mappers may have output the same key).
  - **Reduce -** The output of the reduce task is typically written to a RecordWriter via TaskInputOutputContext.write(Object, Object).

What is YARN?

- Apache YARN, which stands for 'Yet Another Resource Negotiator', is Hadoop's cluster resource management system.
- YARN provides APIs for requesting and working with Hadoop's cluster resources.
- These APIs are usually used by components of Hadoop's distributed frameworks such as MapReduce, Spark, Tez etc. which are built on top of YARN.
- User applications typically do not use the YARN APIs directly.

What are the key components of YARN?

- YARN consists of the following different components
  - **Resource Manager** - The Resource Manager is a global component or daemon, one per cluster, which manages the requests to and resources across the nodes of the cluster.
  - **Node Manager** – Node Manager runs on each node of the cluster and is responsible for launching and monitoring containers and reporting the status back to the Resource Manager
  - **Container**- Container is YARN framework is a Unix process running on the node that executes an application-specific process with a constrained set of resources (Memory, CPU, etc.)

What is Resource-Manager in YARN?

- The Resource-Manager has two main components - Scheduler and Applications-Manager
  - **Scheduler** - The scheduler is responsible for allocating resources to and starting applications based on the abstract notion of resource containers having a constrained set of resources.

- **Application-Manager** - The Applications Manager is responsible for accepting job-submissions, negotiating the first container for executing the application specific Application Master and provides the service for restarting the Application Master container on failure.

What are the scheduling policies available in YARN?

- YARN scheduler is responsible for scheduling resources to user applications based on a defined scheduling policy.
- YARN provides three scheduling options - FIFO scheduler, Capacity scheduler and Fair scheduler.
    - **FIFO Scheduler** - FIFO scheduler puts application requests in queue and runs them in the order of submission.
    - **Capacity Scheduler** - Capacity scheduler has a separate dedicated queue for smaller jobs and starts them as soon as they are submitted.
    - **Fair Scheduler** - Fair scheduler dynamically balances and allocates resources between all the running jobs.

What is HDFS?

- HDFS, which stands for 'Hadoop Distributed File System', is a distributed file system that comes with the Hadoop platform.
- HDFS is designed for storing large files on clusters of commodity hardware, with high throughput streaming access to the file system data.

Describe the architecture of HDFS?

- Following are the main components of a HDFS installation.
    - **Name-Node** - An HDFS cluster consists of a single Name-Node, which is a master server that manages the file system namespace and regulates access to files by clients.
    - **Data-node** - An HDFS cluster consists of many Data-Nodes, usually one per node in the cluster, which manages the storage attached to that particular node.
    - **Data Blocks** - Files in HDFS are split into one or more blocks and these blocks are stored and replicated across a set of Data Nodes.

How do you copy files from HDFS filesystem to local filesystem using HDFS command-line interface?

- You can copy files from HDFS file system to local filesystem via command-line interface by using the sub-command **-copyToLocal**

```
% hadoop fs -copyToLocal /temp/test1.txt /docs/test1.txt
```

How do you read files from HDFS filesystem using Java programming language?

- The Hadoop API provides the class *org.apache.hadoop.fs.FileSystem* to access the HDFS filesystem. You call the *open()* method on the FileSystem class and pass the HDFS filesystem path as a parameter.

```
String uri = 'hdfs://localhost/interviewgrid/hadoop_questions.txt'
FileSystem fs = FileSystem.get(URI.create(uri), conf);
fs.open(new Path(uir));
```

How do you access HDFS data?

- The Hadoop API provides the class *org.apache.hadoop.fs.FileSystem* to access the HDFS filesystem. You call the *create()* method on the FileSystem class and pass the HDFS filesystem path as a parameter. The *create()* method returns an output stream to which you can write the data to.

```
String uri = 'hdfs://localhost/interviewgrid/hadoop_questions.txt'
FileSystem fs = FileSystem.get(URI.create(uri), conf);
fs.create(new Path(uri));
```

Spark

What is Apache Spark?

- Apache Spark is a fast and general-purpose cluster computing system.
- Apache Spark has an advanced DAG execution engine that performs in-memory computing and supports acyclic data flows.
- This makes Spark computations super-fast. Spark programs run up to 100X faster than Hadoop MapReduce in memory and 10X faster on disk.
- Spark supports multiple programming languages. Spark provides built-in APIs in Java, Scala, Python and R programming languages.
- Apache Spark provides multiple components on top of Spark core. These are Spark SQL, Spark Streaming, MLIB, GraphX

How is Apache Spark different from Hadoop?

- Apache Spark supports in-memory cluster computing instead of storing data on disk. Hadoop uses MapReduce that uses data on disks.
- Spark programs run up to 100X faster than Hadoop MapReduce in memory and 10X faster on disk.
- Apache Spark requires less components to manage. Hadoop MapReduce only provides support for batch processing.
- Hadoop depends on other components such as Storm, Giraph etc. for other capabilities. Hence Hadoop required more components to manage.

What are the core library components that make up the Spark ecosystem?

- Apache Spark ecosystem comes with four component libraries - Spark SQL, Spark Streaming, ML lib and GraphX.
    - **Spark SQL** - Spark SQL makes it possible to seamlessly use SQL queries in Spark applications.
    - **Spark Streaming** - Spark Streaming makes it easy to build scalable fault-tolerant streaming applications.
    - **ML lib –** ML lib is Apache Spark's scalable machine learning library. MLlib contains many algorithms, utilities and workflows that support machine learning applications.
    - **GraphX -** GraphX is Apache Spark's API for graphs and graph-parallel computations.

What is a Resilient Distributed Dataset (RDD)?

- Apache Spark provides Resilient Distributed Datasets (RDD) which are fault-tolerant collection of data elements, partitioned across the nodes of the cluster, and can be operated on parallelly by Spark.
- Resilient Distributed Datasets are created by transforming files in the Hadoop file system or Hadoop-supported file system.
- Spark can persist RDDs to memory and reuse them across parallel operations. RDDs automatically recover from node failures.

How do you create Resilient Distributed Datasets (RDDs)?

- RDDs can be created in two ways.

1. **Parallelizing an existing collection** - RDD can be created by parallelizing an existing data collection in the the Spark driver program. A collection can be parallelized by calling the method *parallelize()* on the spark context object and passing the data collection.

2. **Referencing an external file** - RDD can be created by referencing an external Hadoop or Hadoop-supported file system such as HDFS, HBase, Amazon S3, Cassandra etc.

What kinds of operations can be performed on Resilient Distributed Datasets (RDDs)?

- Two kinds of operations
    - **Transformations** - Transformations create a new dataset from an existing dataset. Transformations in Spark are lazy
    - **Actions** - Actions perform computations on a dataset and return a value.

What are some of the common actions supported by Spark?

- **Reduce (func)** – reduce () action aggregates the elements of the dataset using a function func which takes two arguments and returns one.
- **Collect ()** – collect () action returns all the elements of the dataset as an array to the driver program.

- **Count ()** – count () action returns the number of elements in the dataset.
- **First ()** – first () action returns the first element of the dataset.
- **countByKey () –** countByKey () action returns a hashmap of (K, Int) pairs with the count of each key.

What is shared variable in Apache Spark?

- Spark supports two types of shared variables. Broadcast variables and Accumulators
    - **Broadcast Variables** - Broadcast variables are used to cache a value in memory on all nodes.
    - **Accumulators** - Accumulators are variables that are incremented, such as counters and sums.

What is Apache Spark GraphX?

- Apache Spark GraphX is a component library provided in the Apache Spark ecosystem that seamlessly works with both graphs as well as with collections.
- GraphX implements a variety of graph algorithms and provides a flexible API to utilize the algorithms.

What are the different types of operators provided in the Apache GraphX library?

- Apache Spark GraphX provides the following types of operators - Property operators, Structural operators and Join operators.
    - **Property Operators** - Property operators modify the vertex or edge properties using a user defined map function and produces a new graph.
    - **Structural Operators** - Structural operators operate on the structure of an input graph and produces a new graph.
    - **Join Operators** - Join operators add data to graphs and produces a new graph.

What are the property operators provided in the GraphX library?

- Property operators modify the vertex or edge properties using a user defined map function and produces a new graph.
- Property operators do not impact the graph structure, but the resulting graph reuses the structural indices of the original graph.
- Apache Spark GraphX provides the following property operators - mapVertices(), mapEdges(), mapTriplets()

What are the structural operators provided in the Grapx library?

- Structural operators modify the structure of input graph and produces a new graph.
- Apache Spark Graphx provides the following structural operators. **reverse()**, **subgraph()**, **mask()**, **groupEdges()**

What are the join operators provided in the Grapx library?

- Join operators join data from external collections (RDDs) with graphs. Apache Spark Graphx provides the following join property operators.
    - **joinVertices()** - The joinVertices() operator joins the input RDD data with vertices and returns a new graph. The vertex properties are obtained by applying the user defined map() function to the result of the joined vertices. Vertices without a matching value in the RDD retain their original value.
    - **outerJoinVertices()** - The outerJoinVertices() operator joins the input RDD data with vertices and returns a new graph. The vertex properties are obtained by applying the user defined map() function to the all vertices, and includes ones that are not present in the input RDD.

How do you build graphs from a collection of vertices and edges in an RDD using GraphX library?

- Apache Spark Graphx provides various operation to build graphs from an RDD of vertices and edges.
    - **GraphLoader.edgeListFile()**
    - **Graph.apply()**
    - **Graph.fromEdges()**
    - Graph.fromEdgeTuples()

What are the analytic algorithms provided in Apache Spark GraphX?

- Apache Spark GraphX provides a set of algorithms to simplify analytics tasks.
    - **Page Rank** - PageRank measures the importance of each vertex in a graph.
    - **Connected Components** - The connected components algorithm labels each connected component of the graph with the ID of its lowest-numbered vertex.
    - **Triangle Counting** - A vertex is part of a triangle when it has two adjacent vertices with an edge between them.

What is Apache Spark MLlib?

- MLlib is a library provided in Apache Spark for machine learning. It provides tools for common machine learning algorithms, featurizations, Pipelines, Persistence and utilities for statistics, data handling etc.

What are the correlation methods provided in Spark MLlib library?

- Apache Spark MLlib provides support for two correlation methods - Pearson's correlation and Spearman's correlation.

What methods are provided in Spark MLlib library for Hypothesis testing?

- Hypothesis testing is a statistical tool that determines if a result occurred by chance or not, and whether this result is statistically significant.
- Apache Spark MLlib supports Pearson's Chi-squared tests for independence.

What are ML Pipelines?

- Apache Spark MLlib provides ML Pipelines which is a chain of algorithms combined into a single workflow. ML Pipelines consists of the following key components.
    - **Data Frame** - The Apache Spark ML API uses Data Frames provided in the Spark SQL library to hold a variety of data types such as text, feature vectors, labels and predictions.
    - **Transformer** - A transformer is an algorithm that transforms one data frame into another data frame.
    - **Estimators** - An estimator is an algorithm that can be applied on a data frame to produce a Transformer.

What are some of the Transformation algorithms provided in Spark MLlib?

- **Tokenizer** - Tokenizer breaks text into smaller terms usually words.
- **StopWordsRemover** - Stop words remover takes a sequence of strings as input and removes all stop words for the input. Stop words are words that occur frequently in a document but carries little importance.
- **n-gram** - An n-gram contains a sequence of n tokens, usually words, where n is an integer. NGram takes as input a sequence of strings and outputs a sequence of n-grams.
- **Binarizer** - Binarizer is a transformation algorithm that transforms numerical features to binary features based on a threshold value. Features greater than the threshold value are set to 1 and features equal to or less than 1 are set to 0.
- **PCA**
- **PolynomialExpansion** - PolynomialExpansion class provided in the Spark MLlib library implements the polynomial expansion algorithm. Polynomial expansion is the process of expanding features into a polynomial space, based on n-degree combination of original dimensions.
- **Discrete Cosine Transform** - The discrete cosine transformation transforms a sequence in the time domain to another sequence in the frequency domain.
- **StringIndexer** - StringIndexer assigns a column of string labels to a column of indices.
- **IndexToString** - IndexToString maps a column of label indices back to a column of original label strings.
- **OneHotEncoder** - One-hot encoder maps a column of label indices to a column of binary vectors.
- **VectorIndexer** - VectorIndexer helps index categorical features in dataset of vectors.
- **Interaction** - Interaction is a transformer which takes a vector or double-valued columns and generates a single column that contains the product of all combinations of one value from each input column.

- **Normalizer** - Normalizer is a Transformer which transforms a dataset of Vector rows, normalizing each Vector to have unit norm. This normalization can help standardize your input data and improve the behavior of learning algorithms.
- **StandardScaler** - StandardScaler transforms a dataset of Vector rows, normalizing each feature to have unit standard deviation and/or zero mean.
- **MinMaxScaler** - MinMaxScaler transforms a dataset of Vector rows, rescaling each feature to a specific range (often [0, 1]).
- **MaxAbsScaler** - MaxAbsScaler transforms a dataset of Vector rows, rescaling each feature to range [-1, 1] by dividing through the maximum absolute value in each feature. It does not shift/center the data, and thus does not destroy any sparsity.
- **Bucketizer** - Bucketizer transforms a column of continuous features to a column of feature buckets, where the buckets are specified by users.
- **ElementwiseProduct** - ElementwiseProduct multiplies each input vector by a provided “weight” vector, using element-wise multiplication. In other words, it scales each column of the dataset by a scalar multiplier. This represents the Hadamard product between the input vector, v and transforming vector, w, to yield a result vector.
- **SQLTransformer** - SQLTransformer implements the transformations which are defined by SQL statement.
- **VectorAssembler** - VectorAssembler is a transformer that combines a given list of columns into a single vector column.
- **QuantileDiscretizer** - QuantileDiscretizer takes a column with continuous features and outputs a column with binned categorical features.
- **Imputer** - The Imputer transformer completes missing values in a dataset, either using the mean or the median of the columns in which the missing values are located.

What is Apache Spark SQL?

- Spark SQL is a library provided in Apache Spark for processing structured data. Spark SQL provides various APIs that provides information about the structure of the data and the computation being performed on that data. You can use SQL as well as Dataset APIs to interact with Spark SQL.

What are Datasets in Apache Spark SQL?

- A Dataset is a distributed collection of data that was introduced in Spark 1.6 that provides the benefits of RDDs plus the benefits of Spark SQL's optimized execution engine.

How do you create Datasets in Apache Spark SQL?

- A Dataset can be constructed from Java objects and then manipulated using functional transformations such as map(), flatMap(), filter() etc.

How do you convert existing RDDs to Spark Datasets?

- Three steps.
    - Create an RDD of rows using the original RDD.
    - Create the schema represented by a StrutType matching the structure of rows in the RDD.
    - Apply the schema to the RDD of rows via *createDataFrame()* method provided by *SparkSession*.

What is SparkSession?

- SparkSession is the entry point to a Spark application which contains information about the application and the application configuration parameters and values.
- A SparkSession object can be created by using the command *builder()* on a *SparkSession* object.

```
SparkSession spark = SparkSession.builder()
.appName('Spark SQL Example')
.config('config.option','value')
.getOrCreate()
```

What are DataFrames?

- A DataFrame is a Dataset organized into named columns. A DataFrame is equivalent to a Relational Database Table.
- DataFrames can be created from a variety of sources such as structured data files, external databases, Hive tables and Resilient Distributed Datasets.

What is the difference between Temp View and Global Temp View?

- Temporary views in Spark SQL are tied to the Spark session that created the view, and will not be available once the Spark session is terminated.
- Global Temporary views are not tied to a Spark session, but can be shared across multiple Spark sessions. Global Temporary views are available until the Spark application is terminated.

```
df.createGlobalTempView('global_temp.test');
spark.sql('SELECT * FROM global_temp.test').show();
spark.newSession().sql('select * from global_temp.test').show();
```

What is Apache Spark Streaming?

- Spark Streaming is a library provided in Apache Spark for processing live data streams that is scalable, has high-throughput and is fault-tolerant.
- Processed data can be pushed to live dashboards, file systems and databases.

Describe how Spark Streaming processes data?

- Apache Spark Streaming component receives live data streams from input sources such as Kafka, Flume, Kinesis etc. and divides them into batches. The Spark engine processes these input batches and produces the final stream of results in batches.

What is a StreamingContext object?

- StreamingContext object represents the entry point to a Spark streaming application and contains the batch interval for the Spark streaming program and the SparkConf object. The SparkConf object and the batch interval are set when creating a new instance of the StreamingContext object.

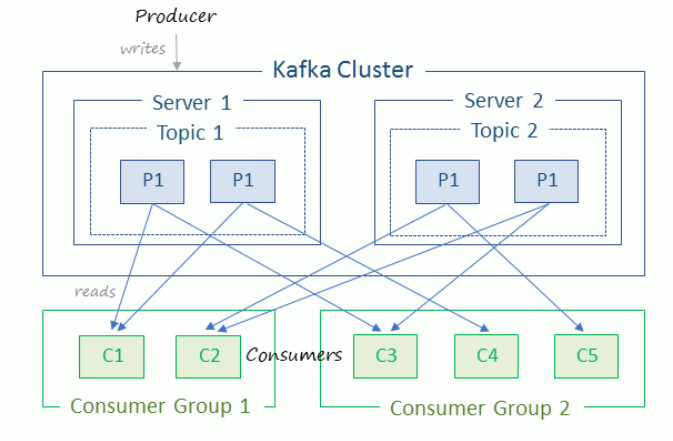
---------- **Kafka** ----------

How does Apache Kafka improve traditional architectures?

- Traditional architectures usually start simple with a few systems, with point-to-point data exchange integrations between them, and a data lake or data warehouse to which data is sourced through ETL processes.
- Kafka is a distributed messaging system, where multiple systems can post messages to and consume messages from.

What are the key components of Kafka?

- Kafka consists of the following key components.
    - **Kafka Cluster** - Kafka cluster contains one or more Kafka brokers (servers) and balances the load across these brokers.
    - **Kafka Broker** - Kafka broker contains one or more Kafka topics.
    - **Kafka Topics** - Kafka topics are categories or feeds to which streams of messages are published to.
    - **Kafka Partitions** - A Kafka topic can be split into multiple partitions. Kafka partitions enable the scaling of topics to multiple servers.
    - **Kafka Offsets** - Messages in Kafka partitions are assigned sequential id number called the offset. The offset identifies each record location within the partition.
    - **Kafka Producers** - Kafka producers are client applications or programs that post messages to a Kafka topic.
    - **Kafka Consumers** - Kafka consumers are client applications or programs that read messages from a Kafka topic.



What is the retention policy for Kafka records in a Kafka cluster?

- Kafka cluster retains all data records using a configurable retention period.
- The data records are retained even if they have been consumed by the consumers.
- For example, if the retention period is set as one week, then the data records are stored for one week after their creation before they are deleted.

What is replication factor?

- Replication factor specifies the number of copies that each partition of a topic will have.
- A replication factor of 2 will create 2 copies (replicas) of each partition.
- A replication factor of 3 will create 3 copies (replicas) of each partition.
- One of the replicas will be the leader, the remaining will be the followers.

MongoDB

What is MongoDB? How is it different from other relational or non-relational databases?

- MongoDB is a non-relational, document-based database.
- Relational databases such as My SQL and Oracle store data in tables, rows and columns. Relational databases are structured, and tables can be linked with each other via foreign keys.
- Non-Relational databases, also called NoSQL databases, contain unstructured data and are commonly used in big data solutions to store and process massive amounts of disparate data. There are four different kinds of NoSQL databases.
    - **Graph databases** – Graph databases are based on graph theory. These databases are designed for data which needs to be represented as graphs.
    - **Key-Value stores** – These databases store data as an indexed key and value pairs. These databases store data in a schema-less way.

    - **Column store** – These databases are designed to store data as columns of data, rather than as rows as data.
    - **Document databases** – Document databases are designed to store documents, with each document having a unique key. What is the key features MongoDB?

- Document store, High Availability, Horizontal scalability, Query Language

How is data stored in MongoDB? How does it compare to a relational database?

- MongoDB database contains Collections. Collections contains Documents. Documents contains fields and values in BSON format.
- Collections are analogous to tables in relational database. Documents are analogous to rows in relational database.

How do you create a new MongoDB database from Mongo shell?

```
> use interview_grid_db
switched to db interview_grid_db
```

How do you create a new collection in MongoDB via the Mongo shell?

- **Explicit creation** – You can explicitly create a new collection by using the command db.createCollection(). This enables us to set properties on the collection such as the setting the maximum file size, validation rules etc.

```
//Explicit Creation
>db.createCollection("employees")
{ "ok" : 1 }
```

- **Implicit creation** – MongoDB creates a new collection automatically, if you insert a document into a collection and that collection does not exist.

```
//Implicit Creation
>db.employees.insert({fname:"John", lname:"Doe", age:"25",
 title:"Manager", dept:"IT"})
WriteResult({ "nInserted" : 1 })
```

What are capped collections in MongoDB?

- Capped collections are collections that store a fixed number of documents and maintains the insertion order of the documents.
- If the number of documents in a capped collection reached the maximum, then the earliest inserted document will be deleted to make space for the new document.

What is Mongo shell?

- Mongo shell is a command line user interface to MongoDB. You can use Mongo shell to query and update data from MongoDB. MongoDB is written in Java script.

What is the difference between the operations db.collection.insertOne() and db.collection.insertMany()?

- **db.collection.insertOne()** – Inserts a single document into a MongoDB collection. It returns a document containing the inserted document's _id field.
- **db.collection.insertMany()** – Inserts a single document or multiple documents into a MongoDB collection. It returns a document containing each inserted document's _id.

```
- > db.employees.insertOne({fname:"John", lname:"Doe", age:"25",
  title:"Manager",dept:"IT"})
  {
  "acknowledged" : true,
  "insertedId" : ObjectId("58479913fa42b4972b1efe40")
  }
```

```
> db.employees.insertMany([{fname:"John", lname:"Doe", age:"25",
title:"Manager", dept:"IT"},{fname:"Mike", lname:"Adams", age:"32",
title:"Director", dept:"IT"}])
{
"acknowledged" : true,
"insertedIds" : [
ObjectId("58479c2dfa42b4972b1efe46"),
ObjectId("58479c2dfa42b4972b1efe47")
]
}
```

What is the difference between the operations db.collection.insertMany() and db.collection.insert()?

- **db.collection.insertMany()** – Inserts a single document or multiple documents into a MongoDB collection. It returns a document containing each inserted document's _id.
- **db.collection.insert()** – Inserts one or multiple documents into a MongoDB collection. It returns a BulkWriteResult object with status of the operation including details.

What is WriteResult object?

- WriteResult object is an object returned by the db.collection.insertOne() and db.collection.insertMany() operations, which contains the object ids of the documents inserted by the operation.

What is BulkWriteResult object?

- BulkWriteResult object is an object returned by the db.collection.insert() operation in which multiple documents are inserted.
- BulkWriteResult object contains status of the operation including details such as error, number of documents inserted, number of documents upserted etc.

How do you find documents from a collection?

- MongoDB provides db.collections.find() operation to find documents in a collection. The syntax of find() operation is *db.collections.find({query filter},{projection})*.

How do you find all documents from a collection?

- You can find all the documents from a collection by using the find () operation without the query filter section.
- You can use *db.collections.find()* or *db.collections.find({})* to find all the documents contained in a collection.
- For example, db.employees.find() returns all the documents contained in employees collection

How do you search for exact field matches in MongoDB. i.e how do you find all documents that contains a field with a specific value? For example, how do you find all employees in the employee collection, whose 'title' is 'Manager'?

```
>db.employees.find({"title":"manager"})

{ "_id" : ObjectId("588e54d4363650c07be0817b"),
 "fname" : "John", "lname" : "Doe",
 "age" : "25", "title" : "manager",
 "dept" : "IT" }, {...}, ... , {...}
```

How do you search for documents in which a specific field have one or more values? For example, how do you find all employees in the employee collection, whose 'title' is either 'Manager' or 'supervisor?

```
>db.employees.find({ title: { $in: ["manager" , "supervisor"] } } )

{ "_id" : ObjectId("588e54d4363650c07be0817b"),
 "fname" : "John", "lname" : "Doe",
 "age" : "25", "title" : "manager",
 "dept" : "IT" }, {...}, ... , {...}
```

How do specify AND conditions when searching for MongoDB documents? For example, how do you find all employees in the employee collection, whose 'title' is 'Manager' AND 'age' is less than '30'?

```
>db.employees.find({ title: "manager", age: { $lt: 30 } } )

{ "_id" : ObjectId("588e54d4363650c07be0817b"),
 "fname" : "John", "lname" : "Doe",
 "age" : "25", "title" : "manager",
 "dept" : "IT" }, {...}, ... , {...}
```

How do specify OR conditions when searching for MongoDB documents? For example, how do you find all employees in the employee collection, whose 'title' is 'Manager' OR 'age' is less than '30'?

```
>db.employees.find( { $or: [ { title: "manager" } , { age: { $lt: 30 } } ] } )

{ "_id" : ObjectId("588e54d4363650c07be0817b"),
 "fname" : "John", "lname" : "Doe",
 "age" : "25", "title" : "manager",
 "dept" : "IT" }, {...}, ... , {...}
```

What are text indexes in MongoDB?

- MongoDB provides text indexes to support and optimize text search queries on text content.
- Text indexes can include one or more fields whose value is a string or an array of strings.
- A collection can have only one text index, but that single text index can include multiple fields

```
>db.employees.createIndex({fname:"Text", lname:"Text"})
```

What are aggregation operations in MongoDB?

- MongoDB aggregation operations act on groups of values from multiple documents, perform operations on the grouped values and return a single computed result.

What are the different ways to perform aggregations in MongoDB?

- MongoDB provides three ways to perform aggregations.
    - **Aggregation pipeline** – MongoDB provides aggregation framework that follows the concept of data processing pipeline. The pipeline includes multiple stages that transform the document into an aggregated result.
    - **Map-reduce operation** – MondoDB provides map-reduce operations to perform aggregation.
    - **Single purpose aggregation methods** – *db.collections.distinct()* and *db.collections.count()* that aggregate documents from a collection.

How are relationships maintained in MongoDB?

- Two ways relationship between documents can be maintained in MongoDB.
    - **References** - References store the links or references from one document to the other. Data in this form is normalized data.
    - **Embedded documents** – MongoDB documents can embed documents within fields or within array elements. Data in this form is renormalized data.

How do you model One-to-One relationship in MongoDB?

- You can model One-to-One relationships between documents in MongoDB by either referencing documents or by embedding documents.

- In general, for One-to-One relationships, if you query the documents frequently then embedding documents is more efficient then referencing documents.

How do you model One-to-Many relationship in MongoDB?

- You can model One-to-Many relationships between documents in MongoDB by either referencing documents or by embedding documents.
- If the data on 'many' side of the relationship is not repetitive and it has to be queried frequently then embedding the data is more efficient.
- But if data 'many' side of the relationship is repetitive then referencing data may be more efficient.

How is replication performed in MongoDb?

- MongoDB performs replication by means of replica sets. Replica sets are group of Mongod processes that maintain the same data across data sets.

What is sharding. How does MongoDB perform sharding?

- Sharding is a method of distributing data across multiple machines. MongoDB supports horizontal scaling by Sharding. MongoDB supports deployments with large data sets and high throughput operations via Sharding.

What are the components of a MongoDB sharded cluster?

- Three components.
    - 1. Shard: Shard contains a subset of the sharded data. Each shard can be deployed as a replica set
    - 2. Mongos: Mongos provide an interface between the client applications and the mongo cluster. Mongos act as a query router to the sharded cluster.
    - 3. Config servers: Config servers store metadata and configuration settings for the MongoDB sharded cluster

What is a shard key?

- Shard key is used by MongoDB to distribute the documents of a collection across shards.
- Shard key consists of a field or fields that exist in every document of the MongoDb collection