Hadoop and Big Data

**1. Explain “Big Data” and what are five V’s of Big Data?**

Big Data is a term used to describe a collection of data that is huge in volume and yet growing exponentially with time. In short such data is so large and complex that none of the traditional data management tools are able to store it or process it efficiently.

Five V’s of Big Data are : Volume, velocity, variety, veracity and value

**2. What is Hadoop and its components?**

Apache Hadoop is an open source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data

Main componet of hadoop include: HDFS, MapReduce, YARN, and Hadoop Common.

**3. List the difference between Hadoop 1 and Hadoop 2?**

Hadoop 1 only supports MapReduce processing model in its architecture and it does not support non MapReduce tools. On other hand Hadoop 2 allows to work in MapReducer model as well as other distributed computing models like Spark, Hama, Giraph, Message Passing Interface) MPI & HBase coprocessors.

**4. What is a checkpoint?**

Checkpoint node in hadoop is a new implementation of the Secondary NameNode to solve the drawbacks of Secondary NameNode. Main function of the Checkpoint Node in hadoop is to create periodic checkpoints of file system metadata by merging edits file with fsimage file.

**5. What is the difference between an “HDFS Block” and an “Input Split”?**

Input splits are a logical division of your records whereas HDFS blocks are a physical division of the input data.

**6. What is Big Data?**

Huge and complex data that none of the traditional data management tools are able to store it or process it efficiently.

**7. What is the difference between Hadoop and Traditional RDBMS?**

Hadoop has the ability to process and store all variety of data whether it is structured, semi-structured or unstructured whereasTraditional RDBMS is used only to manage structured and semi-structured data

**8. Differentiate between Structured and Unstructured data?**

Structured data is highly-organized and formatted in a way so it's easily searchable in relational databases. Unstructured data has no pre-defined format or organization, making it much more difficult to collect, process, and analyze.

**9. On what concept the Hadoop framework works?**

Hadoop Framework works on the following two core components-

1)HDFS – Data in HDFS is stored in the form of blocks and it operates on the Master Slave Architecture.

2)Hadoop MapReduce-MapReduce distributes the workload into various tasks that can run in parallel. Hadoop jobs perform 2 separate tasks- job. The map job breaks down the data sets into key-value pairs or tuples. The reduce job then takes the output of the map job and combines the data tuples to into smaller set of tuples. The reduce job is always performed after the map job is executed.

**10. What are the main components of a Hadoop Application?**

Answered in Q2

**11. What is Hadoop streaming?**

Hadoop Streaming is an utility that comes with hadoop distribution and allows users to write mapreduce programs in any programming/scripting language that can read standard input (stdin) and write to standard output (stdout)

**12. What are the most commonly defined input formats in Hadoop?**

The most common Input Formats defined in Hadoop are:

**Text Input Forma**t- This is the default input format defined in Hadoop.

**Key Value Input Format**- This input format is used for plain text files wherein the files are broken down into lines.

**Sequence File Input Format-** This input format is used for reading files in sequence.

**13. What are the steps involved in a big data pipeline?**

Data source→ Ingestion→ Processing→ Storage→ Analysis→ Visualization

**14. What are CSV Files, when would you use csv file?**

CSV files are simple and widely spread file format used to store tabular data, such as a spreadsheet or database.

**15. What are JSON Files, when would you use JSON file?**

JSON (JavaScript Object Notation) file is a lightweight file format that is used for data interchanging.An example of where this is used is web services responses.

**16. What are Avro Files, when would you use Avro file?**

Avro is a row-based storage file format for Hadoop which is widely used as a serialization platform. Avro stores the data definition (schema) in JSON format making it easy to read and interpret by any program. The data itself is stored in binary format making it compact and efficient.

**17. What are Parquet Files, when would you use Parquet file?**

Parquet, an open source file format for Hadoop. Parquet stores nested data structures in a flat columnar format. Compared to a traditional approach where data is stored in row-oriented approach, parquet is more efficient in terms of storage and performance.

Parquet is an open source file format available to any project in the Hadoop ecosystem.

**18. Hadoop architecture?**

The main parts that make up hadoop architecture are following:

HDFS – MAPREDUCE- YARN – HADOOP COMMON

https://www.dezyre.com/article/hadoop-ecosystem-components-and-its-architecture/114

**19. Minimum system requirements for running a Hadoop Cluster with High Availability**

Hadoop needs one Namenode ,one Standby Node,Network shared Storage space(shared between two name nodes), and at least 2 datanodes for running cluster with high availability.

**20. What is active and passive NameNode in Hadoop?**

Active Namenode is the primary Namenode which works and runs in the cluster. Passive Namenode is a standby Namenode, which has similar metadata as active Namenode. When the active Namenode goes down, the passive Namenode replaces the active Namenode in the cluster.

**Hadoop MapReduce**

**1. Explain the usage of Context Object?**

Context object: allows the Mapper/Reducer to interact with the rest of the Hadoop system. It includes configuration data for the job as well as interfaces which allow it to emit output. Applications can use the Context: to report progress.

**2. What are the core methods of a Reducer?**

The three core methods of a Reducer are:

1. setup(): this method is used for configuring various parameters like input data size, distributed cache.

2. reduce(): heart of the reducer always called once per key with the associated reduced task

3. cleanup(): this method is called to clean temporary files, only once at the end of the task

**3. Explain about the partitioning, shuffle and sort?**

Map -> Combiner -> Partitioner -> Sort -> Shuffle -> Sort -> Reduce

Partitioning is the phase between Map phase and Reduce phase in Hadoop workflow. Since partitioner gives output to Reducer, the number of partitions is same as the number of Reducers. Partitioner will partition the output from Map phase into distinct partitions by using a user-defined condition.

When mappers finish their tasks, their output is a series of key-

value pairs. Shuffling is simply the act of transferring the mapper output to the reducers.

Sorting is the process or sorting the mappers' output by key. For example, unsorted output from the mappers might look like this:

sam,1

joe,1

rajesh,1

joe,1

bill,1

andy,1

joe,1

When it's sorted, it would look like this:

andy,1

bill,1

joe,1

joe,1

joe,1

rahesh,1

sam,1

Sorting helps the system determine when a new reducer should be started. The example above has very little data, so doesn't make much sense - but if you have millions of rows, you will likely have many duplicate keys. Sorting makes it easy to get all of the "bill" values to one reducer, the "joe" values to another reducer, and so on.

**4. How to write a custom partitioner for a Hadoop MapReduce job?**

A Custom Partitioner can be written by overriding the getPartition method. The getPartition method takes two parameters which is the key and value.

5. What are side data distribution techniques in Hadoop?

The extra read only data required by a hadoop job to process the main dataset is referred to as side data. Hadoop has two side data distribution techniques -

i) Using the job configuration — This technique should not be used for transferring more than few kilobytes of data as it can pressurize the memory usage of hadoop daemons,particularly if your system is running several hadoop jobs.

ii) Distributed Cache — Rather than serializing side data using the job configuration, it is suggested to distribute data using hadoop’s distributed cache mechanism.

**6. Explain 3 Stages of MapReduce?**

MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

Map stage − The map or mapper’s job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

Reduce stage − This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

**Flume**

**1. Explain about the core components of Flume?**

Apache Flume is an agent for data collection. It is generally used for log data. Flume takes data from several sources like Avro, Syslog’s, and files and delivers to various destinations like Hadoop HDFS or HBase.

Apache Flume is a composed of 6 important components-

• **Events-** The data units that are transferred over a channel from source to sink. The size of events is usually 4 KB.

• **Sources**- They accept data from a server or an application. Sources listen for events and write events to a channel.

• **Sinks** receive data and store it in HDFS repository or transmit the data to another source. The sink basically write event data to a target and remove the event from the queue.

• **Channels** connect between sources and sink by queuing event data for transactions.

• **Interceptors** drop data or transfer data as it flows into the system.

• **Agents** are used to run the sinks and sources in flume.

**2. Explain about the different channel types in Flume. Which channel type is faster?**

There are three different built in channels in Flume:

MEMORY CHANNEL: Events are study from the source into memory and passed to the sink

JDBC CHANNEL:It stores the events in an embedded Derby Database

FILE CHANNEL: It writes the contents to a file on the file system after reading the event from a source. The file is erased only after the contents are successfully delivered to the sink.

MEMORY CHANNEL: Is the fastest channel among the three however has the risk of data loss. The channel that you choose totally depends on the nature of the big data application and the value of each event.

**3. Which is the reliable channel in Flume to ensure that there is no data loss and why?**

FILE Channel is the most reliable channel.

**4. Flume architecture?**

Following are the components of Flume Architecture: Flume Event, Flume Agent, and Flume Client.

**5. Explain about the replication and multiplexing selectors in Flume?**

Channel Selectors are used to handle multiple channels. Based on the Flume header value, an event can be written just to a single channel or to multiple channels. If a channel selector is not specified to the source then by default it is the Replicating selector. Using the replicating selector, the same event is written to all the channels in the source’s channels list. Multiplexing channel selector is used when the application has to send different events to different channels.

**6. How does multi-hop agent can be set up in Flume?**

Avro RPC Bridge mechanism is used to set up Multi-hop agents in Apache Flume.

**7. Differentiate between FileSink and FileRollSink?**

The major difference between HDFS FileSink and FileRollSink is that HDFS File Sink writes the events into the Hadoop Distributed File System (HDFS) whereas File Roll Sink stores the events into the local file system.

**8. Sqoop vs flume?**

Both Sqoop and Flume pull the data from the source and push it to the sink. The main difference is Flume is event driven, while Sqoop is not. Sqoop is a connectivity tool for moving data from non-Hadoop data stores – such as relational databases and data warehouses.

**9. Kafka vs Flume**?

Kafka can support data streams for multiple applications, whereas Flume is specific for Hadoop and big data analysis. Kafka can process and monitor data in distributed systems whereas Flume gathers data from distributed systems to land data on a centralized data store.

**23. What is the role of zookeeper in HDFS?**

Zookeeper in Hadoop can be viewed as centralized repository where distributed applications can put data and get data out of it. It is used to keep the distributed system functioning together as a single unit, using its synchronization, serialization and coordination goals.

## Spark

**1. Your experience with spark streaming?**

In my 4 years working in the Big Data Industry, I have mostly used spark as a tool for processing, because of its robustness, highly reliable fast in memory computation and fault tolerance capabilities.

**2. How does spark execute?**

Spark uses master/slave architecture i.e. one central coordinator and many distributed workers. Here, the central coordinator is called the driver. The driver runs in its own Java process. These drivers communicate with a potentially large number of distributed workers called executors. Each executor is a separate java process. A Spark Application is a combination of driver and its own executors. With the help of cluster manager, a Spark Application is launched on a set of machines.

**3. What is windowing in spark streaming?**

Window (also, windowing or windowed) functions perform a calculation over a set of rows. It is an important tool to do statistics. Most Databases support Window functions.

**4. How to join 2 input streams?**

The RDD generated by stream1 will be joined with the RDD generated by stream2. You can also do Left Outer Join, Right Outer Join, Full Outer Join.

**5. What is a pair RDD?**

Paired RDDs are nothing but RDDs containing a key-value pair. Basically, key-value pair (KVP) consists of a two-linked data item in it. Here, the key is the identifier, whereas value is the data corresponding to the key value.

**6. What are broadcast variables?**

Hence, Broadcast variables are read-only variables that are distributed across worker nodes in-memory instead of shipping a copy of data with tasks. Broadcast variables are mostly used when the tasks across multiple stages require the same data or when caching the data in the deserialized form is required.

**7. What are examples of actions in Spark?**

Reduce, collect, takeSample, take, first, saveAsTextfile, saveAsSequenceFile, countByKey, foreach are common actions in Apache spark

**8. What does lazy mean in spark?**

As the name itself indicates its definition, lazy evaluation in Spark means that the execution will not start until an action is triggered.

**9. What are types of transformations?**

Two most basic types of transformations are map(), and filter().

**10. What are some challenges you’ve had with spark?**

Some challenges that I have faced with Spark - and it is likely that others would have faced these too:

* Installation and configuration - Spark is not straightforward to install and configure.
* Resource management - Spark also isn't quite as straight-forward, when it comes to setting up Spark contexts and configuration.

But I realized that it became easier to navigate over time and with practice.

**11. What is RDD Lineage?**

RDD lineage is nothing but the graph of all the parent RDDs of an RDD. We also call it an RDD operator graph or RDD dependency graph. To be very specific, it is an output of applying transformations to the spark.

**12. Do you have experience with Spark streaming?**

Yes, I do. I have used Spark as a processing tool in a number of projects that I have worked on.

**13. What is an RDD?**

Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects.

**14. What’s the difference between a transformation and an action?**

Transformations [create new RDD](http://data-flair.training/blogs/how-to-create-rdds-in-apache-spark/) from existing [RDD](http://data-flair.training/blogs/rdd-in-apache-spark/). Transformations are executed on demand.([Lazy computation](http://data-flair.training/blogs/lazy-evaluation-in-apache-spark-guide/)) Ex: filter(), union()

An Action will return a non-RDD type (your stored value types usually). Actions triggers execution using [lineage graph](http://data-flair.training/blogs/directed-acyclic-graph-dag-in-apache-spark/) to load the data into original RDD Ex: count(), first()

**15. What is shuffling in Spark?**

Shuffle operation is used in Spark to re-distribute data across multiple partitions. In general a single task in Spark operates on elements in one partition. To execute shuffle, we have to run an operation on all elements of all partitions.It is also called all-to- all operation.

**16. What RDD function or method performs shuffling?**

The ‘.flatMap’, ’.map’, ’.reduceByKey’, and ‘.collect()’ functions perform shuffling in spark.

**17. What are window functions is Spark? What are some examples?**

Window (also, windowing or windowed) functions perform a calculation over a set of rows. It is an important tool to do statistics. Most Databases support Window functions.

Examples are row\_number(): Column (Returns a sequential number starting from 1 within a window partition) and rank(): Column (Returns the rank of rows within a window partition, with gaps).

**18. Window aggregation function with scenario**

Window aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning.

Example:

Id Name Salary

-----------------------

1 A 80

2 B 40

3 C 60

4 D 70

5 E 60

6 F Null

Count():

*Count(\*):* Returns total number of records .i.e 6.

*Count(salary):* Return number of Non Null values over the column salary. i.e 5.

*Count(Distinct Salary):*  Return number of distinct Non Null values over the column salary .i.e 4

Sum():

*sum(salary):*  Sum all Non Null values of Column salary i.e., 310

*sum(Distinct salary):* Sum of all distinct Non-Null values i.e., 250.

**19. Difference between sort by and order by**

The difference between "order by" and "sort by" is that "order by" guarantees total order in the output while "sort by" only guarantees ordering of the rows within a reducer. If there are more than one reducer, "sort by" may give partially ordered final results.

**20. How to create data frame in spark?**

**toDF()-** toDF() provides a concise syntax for creating DataFrames and can be accessed after importing Spark implicits.

**createDataFrame()-** The createDataFrame() method addresses the limitations of the toDF() method and allows for full schema customization unlike the toDF() method.

**createDF()-** createDF() is defined in spark-daria.

**21. How to work with JSON file in Spark**

Spark SQL can automatically infer the schema of a JSON dataset and load it as a DataFrame. using the read. json() function, which loads data from a directory of JSON files where each line of the files is a JSON object.

**22. Within spark streaming, how do you have the stream check once on?**

**23. Why use dstream instead of structured streaming?**

Streaming with DStreams is really good for more complicated topologies because of its flexibility and ease of use. Otherwise, structured streaming is more efficient.

**24. Data skewness in spark**

Skewness is the statistical term, which refers to the value distribution in a given dataset. When we say data is skewed, it means it is not properly or evenly distributed.

**25. Coalesces vs partition?**

Coalesce uses existing partitions to minimize the amount of data that's shuffled, while repartition creates new partitions and does a full shuffle.

**26. Spark expertise?**

I would rate my expertise working with Spark, on an intermediate level.

**27. What is RDD vs DF vs DS?**

**An RDD** stands for Resilient Distributed Datasets. It is Read-only partition collection of records. RDD is the fundamental data structure of Spark. It allows a programmer to perform in-memory computations on large clusters in a [fault-tolerant](http://data-flair.training/blogs/apache-spark-streaming-fault-tolerance/) manner. **A DataFrame**, unlike an RDD, is data organized into named columns. For example a table in a relational database. It is an immutable distributed collection of data. **Datasets** in Apache Spark are an extension of DataFrame which provides type-safe, object-oriented programming interface. Dataset takes advantage of Spark’s Catalyst optimizer by exposing expressions and data fields to a query planner.

**28. Specific use case of DS?**

**29. Have you worked with Unstructured data?**

Yes I have. I once worked on a project where I was part of a team tasked with getting APIs from Instagram based on the #StayHomeStaySafe hashtag, for analysis against the number of infected cases and deaths from the Covid-19 virus. The intention was to see if there was a correlation between those posts and infections and deaths i.e If there were less deaths in geographical regions where the hashtag was used, as a result of the impact of its sensitization, or whether it made no impact at all. Of course, data from instagram came in a mixture of media (pictures and videos) and texts from captions, comments and tags.

**30. Spark Session vs Spark Context?0**

**SparkContext** is used as a channel to access all **spark** functionality. **SparkSession** provides a single point of entry to interact with underlying Spark functionality and allows programming Spark with Dataframe and Dataset APIs.

**31. Have used Databricks?**

Yes

**32. How to install dependencies in Spark?**

You can download the jar file and save it to the path, call it in spark-submit

**33. What are spark dynamically allocated sources.**

Spark provides a mechanism to dynamically adjust the resources your application occupies based on the workload. This means that your application may give resources back to the cluster if they are no longer used and request them again later when there is demand. This feature is particularly useful if multiple applications share resources in your Spark cluster.

**34. Explain the Distribution of Executors, Cores and Memory for a Spark Application running in Yarn.**

Hadoop/Yarn/OS Deamons: When we run spark application using a cluster manager like Yarn, there’ll be several daemons that’ll run in the background like NameNode, Secondary NameNode, DataNode, JobTracker and TaskTracker. So, while specifying num-executors, we need to make sure that we leave aside enough cores (~1 core per node) for these daemons to run smoothly.

Yarn ApplicationMaster (AM): ApplicationMaster is responsible for negotiating resources from the ResourceManager and working with the NodeManagers to execute and monitor the containers and their resource consumption. If we are running spark on yarn, then we need to budget in the resources that AM would need (~1024MB and 1 Exe cutor).

HDFS Throughput: HDFS client has trouble with tons of concurrent threads. It was observed that HDFS achieves full write throughput with ~5 tasks per executor . So it’s good to keep the number of cores per executor below that number.

MemoryOverhead: Following picture depicts spark-yarn-memory-usage.

**35. Cluster spark job executor memory usage throughput.**

Number of available executors = (total cores/num-cores-per-executor) = 150/5 = 30. Leaving 1 executor for ApplicationManager => --num-executors = 29. Number of executors per node = 30/10 = 3. Memory per executor = 64GB/3 = 21GB.

**36. How do you code extensions in spark?**

Spark has extension points that help third parties to add customizations and optimizations without needing these optimizations to be merged into Apache Spark

**37. How can you optimize memory issues in spark?**

* How to leverage Tungsten,
* Execution plan analysis,
* Data management (caching, broadcasting),
* Cloud-related optimizations (including S3).

**38. What is the maximum size for a broadcast object in Spark?**

Default is 10mb but we have used till 300 mb which is controlled by spark

**39. What are broadcast variables and accumulators in Apache Spark?**

Spark supports two types of shared variables: **broadcast** variables, which can be used to cache a value in memory on all nodes, and **accumulators,** which are variables that are only “added” to, such as counters and sums.

**40. What is shuffling?**

The process of moving the data from partition to partition in order to aggregate, join, match up, or spread out in some other way, is known as shuffling. The aggregation/reduction that takes place before data is moved across partitions is known as a map-side shuffle.

**41. How can you decrease shuffling in spark?**

Using the broadcast variables

By using the broadcast variable, you can eliminate the shuffle of a big table, however you must broadcast the small data across all the executors

This may not be feasible all the cases, if both tables are big.

**42. Explain a spark groupBy.**

Spark groupBy function is defined in RDD class of spark. It is a transformation operation which means it will follow lazy evaluation. We need to pass one function (which defines a group for an element) which will be applied to the source RDD and will create a new RDD as with the individual groups and the list of items in that group.

**43. Why is spark faster than hadoop**

Because it can process in memory

**44. Spark what is an rdd**

Refer to above

**45. What is the difference between map and flatMap and a good use case for each**

map transforms an RDD of length N into another RDD of length N.

But flatMap (loosely speaking) transforms an RDD of length N into a collection of N collections, then flattens these into a single RDD of results.

**46. Difference of semantics between reduce and reduceByKey**

The main difference between reduce and reduceByKey is, reduce operates on rdd of object, whereas reduceByKey operates on RDD of key value pairs. The function reduce() is a member of RDD[T] class, while reduceByKey() is member of PairRDDFunctions[K,V] class.

reduce() function returns a collection which will be included in DAG for the next level, so it is implemented as action. While reduceByKey returns another RDD which might a part of sequence of operations in a DAG, so it it implemented as transformation.

**47. When are accumulators truly reliable?**

Accumulators re truel reliable when they are present in an Action operation.

**48. Spark worker nodes and executors?**

Spark has a master-slave architecture, where it has got one central coordinator,i.e. Driver, that communicates with many distributed workers nodes(executors).

DRIVER

The driver is basically a process where the main method runs. It converts the user program into tasks and just after that it schedules the tasks on the executors.

WORKERS

Workers (slaves) are running Spark instances where executors live to execute tasks. They are the compute nodes in Spark.

A worker receives serialized tasks that it runs in a thread pool.

It hosts a local Block Manager that serves blocks to other workers in a Spark cluster. Workers communicate among themselves using their Block Manager instances.

When you create SparkContext, each worker starts an executor. This is a separate process (JVM), and it loads your jars, too. The executors connects back to your driver program and now the driver can send them commands, like flatMap, map and reduceByKey. When the driver quits, the executors shuts down.

EXECUTORS

Executor is a distributed agent that is responsible for executing tasks. It typically runs for the entire lifetime of a Spark application which is called static allocation of executors.

Executors reports to HeartbeatReceiver RPC Endpoint on the driver by sending heartbeat and partial metrics for active tasks. It runs multiple tasks over its lifetime(parallel and sequentially). They track running tasks (by their task ids in runningTasks internal registry). They are also used to provide in-memory storage for RDDs that are cached by user programs through Block Manager.

**49. Spark vs MapReduce?**

The key difference between Hadoop MapReduce and Spark lies in the approach to processing: Spark can do it in-memory, while Hadoop MapReduce has to read from and write to a disk. As a result, the speed of processing differs significantly – Spark may be up to 100 times faster. However, the volume of data processed also differs: Hadoop MapReduce is able to work with far larger data sets than Spark.

**50. Spark Checkpoint**

Checkpointing is a process to truncate RDD lineage graph. It saves the application state timely to reliable storage (HDFS). As the driver restarts the recovery takes place.

**51. Spark architecture**

Driver -

Mode-

Client -

driver running from local(eg. Python code) / spark-submit-automation

Cluster -

Driver on cluster/spark-shell -easy to run, flexibility

Master node - reads and writes

-components: driver

Slave node - runs the job/ transformation

-components

-executors

-cores

Resource negotiator

Yarn

Mesos

Spark

For this job i need 3 nodes with 3 executors and 3 cores

Job → task →

Core takes Rdd in and executors come and pick up the data to process it.

1 core => 1 executor

Num of executors cannot be more than num of cores

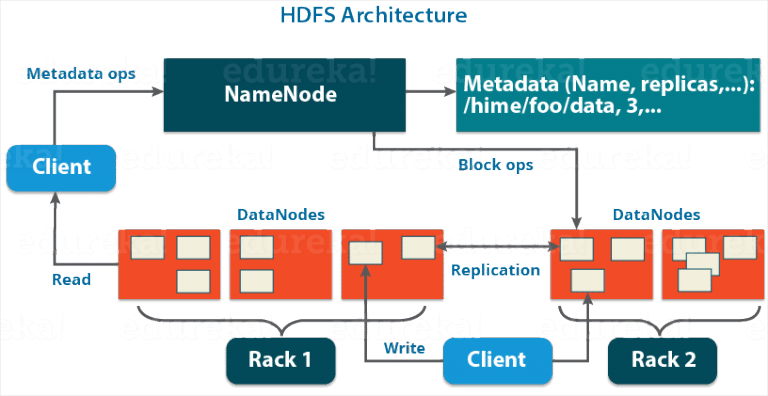
## Hadoop HDFS

**1. What is a block and block scanner in HDFS?**

**Block** is the physical representation of data. It contains a minimum amount of data that can be read or write.

Block scanner is used to identify corrupt datanode **Block**. During a write operation, when a datanode writes into the **HDFS**, it verifies a checksum for that data. This checksum helps in verifying the data corruptions during the data transmission.

**2. HDFS architecture?**



**Namenode:** Contains metadata and maintains the blocks on the datanodes of the hadoop filing system.

**Datanode:** Is where the data is stored as well as replication is done.

**3. Explain the difference between NameNode, Backup Node and Checkpoint NameNode.**

**Namenode:** The namenode contains the metadata of the files in HDFS storing them in

Fsimage and the editlogs. FsImage contains the complete state of the file system namespace since the start of the NameNode. Whereas, EditLogs contains all the recent modifications made to the file system with respect to the most recent FsImage.

**Backup Node:** Backup Node in hadoop is an extended checkpoint node that performs checkpointing and also supports online streaming of file system edits. So unlike the namenode, the edits are not stored in an editlog to update the fsimage file. The changes are registered immediately as a stream.

**Checkpoint Namenode:** Creates periodic checkpoints when the FsImage file and the EditLogs are merged.

**4. What is the port number for NameNode, Task Tracker and Job Tracker?**

Namenode : 50070

Task Tracker: 50060

Job Tracker: 50030

**5. How can you overwrite the replication factors in HDFS?**

Go to hdfs-site. xml and change the dfs.replication property, the default value is set to 3, you can alter it to any desired number, keep in mind it should not be more than the number of datanodes in the cluster.

**6. Explain what happens if during the PUT operation, HDFS block is assigned a replication**

factor 1 instead of the default value 3?

If the replication factor during the PUT operation is set to 1 instead of the default value 3, then it will have a single copy of data. Under these circumstances when the replication factor is set to 1 ,if the DataNode crashes under any circumstances, then the only copy of the data would be lost.

**7. What is the process to change the files at arbitrary locations in HDFS?**

HDFS does not support modifications at arbitrary offsets in the file or multiple writers but files are written by a single writer in append only format i.e. writes to a file in HDFS are always made at the end of the file.

**8. Explain about the indexing process in HDFS**

Indexing process in HDFS depends on the block size. HDFS stores the last part of the data that further points to the address where the next part of data chunk is stored.

**9. What is a rack awareness and on what basis is data stored in a rack?**

All the data nodes put together form a storage area i.e. the physical location of the data nodes is referred to as Rack in HDFS. The rack information i.e. the rack id of each data node is acquired by the NameNode. The process of selecting closer data nodes depending on the rack information is known as Rack Awareness.

**10. What happens to a NameNode that has no data?**

There does not exist any NameNode without data. If it is a NameNode then it should have some sort of data in it

**11. What happens when a user submits a Hadoop job when the NameNode is down?**

The Hadoop job fails when the NameNode is down.

**12. What happens when a user submits a Hadoop job when the Job Tracker is down?**

The Hadoop job fails when the Job Tracker is down.

**13. Whenever a client submits a hadoop job, who receives it?**

NameNode receives the Hadoop job which then looks for the data requested by the client and provides the block information. JobTracker takes care of resource allocation of the hadoop job to ensure timely completion

**14. What is an edge nodes in Hadoop?**

**Edge nodes** are the interface between the **Hadoop** cluster and the outside network. For this reason, they're sometimes referred to as gateway **nodes**. Most commonly, **edge nodes** are used to run client applications and cluster administration tools.

**15. Modifying the block placement strategy of HDFS**

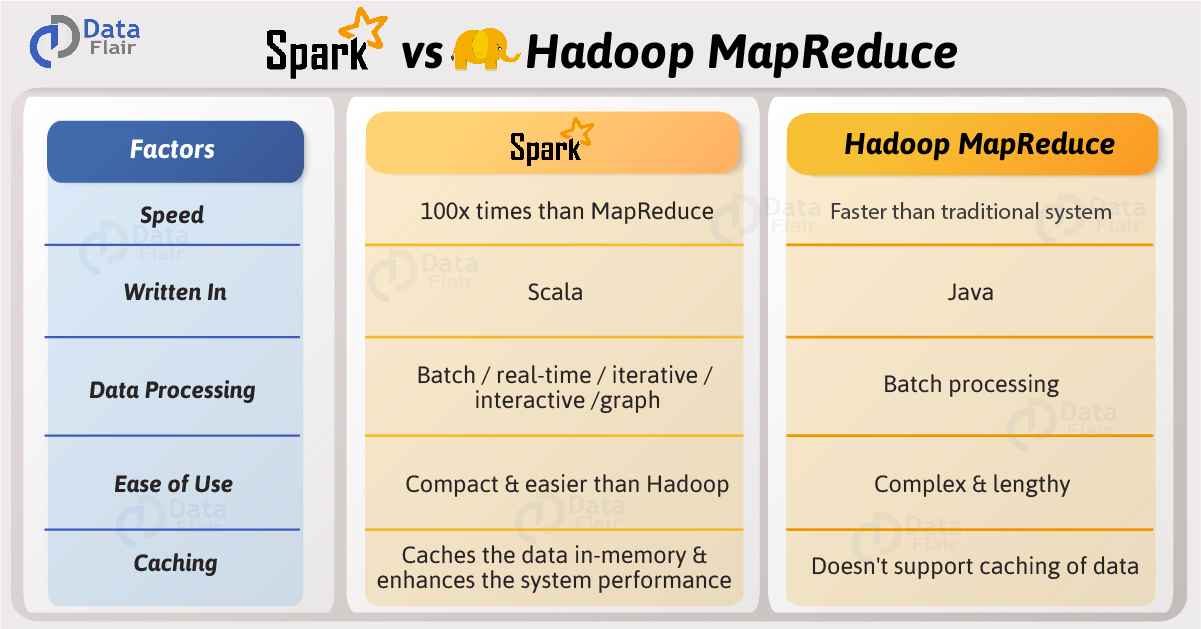
The default behaviour of the block placement policy can be modified by extending the [BlockPlacementPolic](http://hadoop.apache.org/docs/hdfs/r0.21.0/api/org/apache/hadoop/hdfs/server/namenode/BlockPlacementPolicy.html)y interface and pointing the class to the **dfs.block.replicator.classname** property in the Hadoop configuration files.

**16. How to Configure Replication Factor and Block Size for HDFS?**

Edit the following in hdfs-site.xml file: dfs.block.size, dfs.replication.

**17. hdfs vs spark?**

Spark is used for processing data whereas hdfs is used to store data. Therefore, doesn't have its own distributed filesystem, but can use HDFS.



**18. What is Checkpoint HDFS?**

A merge between FsImage and the EditLog

**19. how do you balance disk space on HDFS clusters?**

Hadoop has its balanced policy, apart from that for the unbalanced situation like new nodes adding, deletion caused unbalancing etc, there is HDFS balancer to rebalance the space usage among the cluster data nodes.

**20. Alter file partition on hdfs?**

I dont believe this is possible. I think file partitions can only be altered on hive or whatever relational database they are on. Not on HDFS.

**21. Move data from ORACLE to HDFS?**

Apache Sqoop import.

**22. What are HDFS snapshots?**

**HDFS Snapshots** are read-only point-in-time copies of the file system.

## Hive

**1. How to connect HBase with Hive?**

create a hbase table in Hive and populate it through Hive

CREATE EXTERNAL TABLE hbase\_hive\_table (key int, value string)

STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'

WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,cf1:val")

TBLPROPERTIES ("hbase.table.name" = "hbase\_hive\_table", "hbase.mapred.output.outputtable" = "hbase\_hive\_table");

INSERT OVERWRITE TABLE HBASE\_HIVE\_TABLE SELECT \* FROM pokes WHERE foo=98;

**2. How to optimize hive queries and pig?**

Hive: ORC file, Tez execution engine, partition and bucketing,vectorization(scans, aggregations, filters, and joins. It happens by performing them in batches of 1024 rows at once instead of single row each time.), indexing table for reference, cost-based optimization

Pig: PartitionFilterOptimizer, PredicatePushdownOptimizer, ConstantCalculator, SplitFilter, PushUpFilter, MergeFilter, PushDownForEachFlatten, LimitOptimizer, ColumnMapKeyPrune, AddForEach, MergeForEach, GroupByConstParallelSetter

**3. How do you do Join in Hive?**

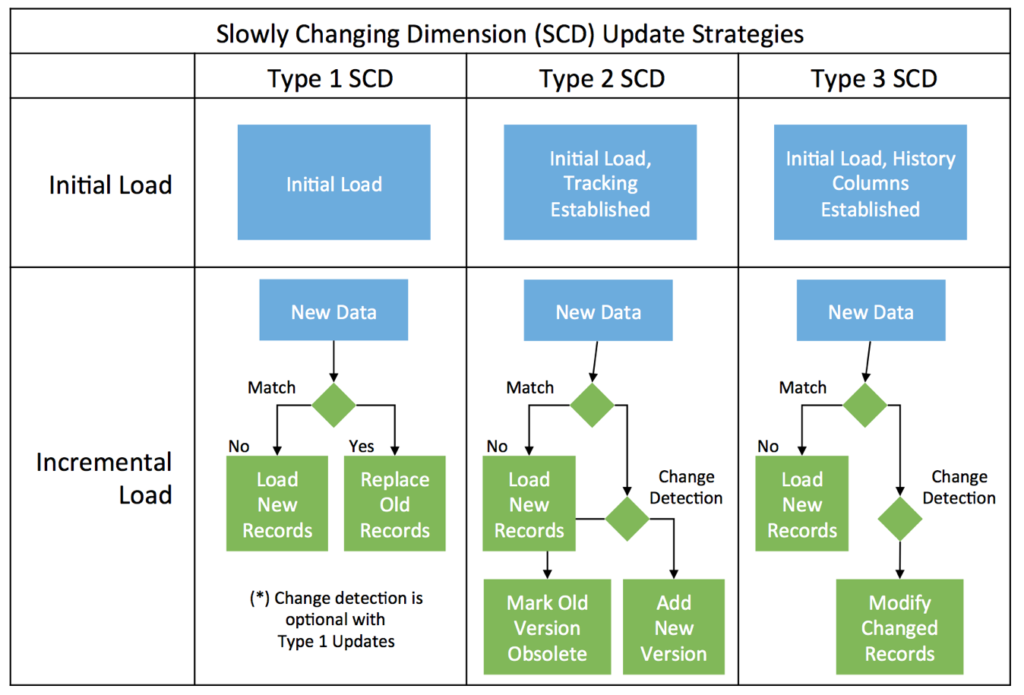
hive> select c.id, c.name, o.order\_date, o.amount from customers c inner join orders o ON (c.id = o.customer\_id);

Inner, full, left, right, left semi join

**4. hive - partitioning and bucketing?**

Partition is to split the table by columns, bucketing is further dividing the table

**5. How do you manage slow changing dimensions in Hive?**



**6. what are the commands to connect and pull the 10 most recent records for hive?**

select a,date,time,b from (select \* ,row\_number() over ( partition by b order by date desc ,time desc ) as rn from mytable ) t where t.rn = 1

**7. What it the default delimiter in hive?**

CTRL-A (ie Octal \001)

**8. Static vs dynamic partitioning in hive?**

in static partitioning we need to specify the partition column value in each and every LOAD statement.

dynamic partition allow us not to specify partition column value each time.

**9. What is the difference between external and internal table in Hive?**

External table you can delete the schema and keep the data, internal you delete the whole thing.

**10. What would happen if we add the data manually to Hive table?**

Using load to add data to Hive table, the partition can be specified in the command.

**11. how to change the meta store in hive?**

We can change the setting of metastore in hive-site.xml

**12. Write a hive query to remove duplicate from a table**

Distinct function

**13. difference between SQL in Hive vs Spark?**

Sql in hive is only for oltp.

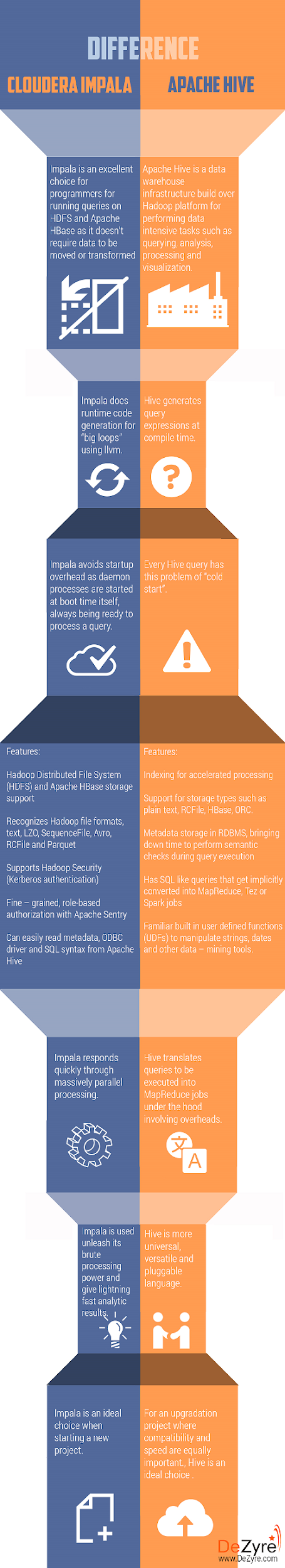
**14. what are the different joins in hive?**

Inner, left, right, outer, semi-left join

**15. difference between hive and Hbase?**

HBase is non relation database

**16. impala vs hive?**



**17. explain vectorization in hive and how it is compiled?**

Vectorization optimizes the query execution by processing a block of 1024 rows at a time, inside which each row is saved as a vector. Simple operations like arithmetic and comparisons are done by quickly iterating through the vectors in a tight loop, with no or very few function calls or conditional branches inside the loop. These loops compile in a streamlined way that uses relatively few instructions and finishes each instruction in fewer clock cycles, on average, by effectively using the processor pipeline and cache memory.

**18. Differences between Apache hive and Apache Spark?**

Hive is a sql like query language on base of HDFS

Spark is distributed processing framework which can process data in batches or as stream

**19. Hadoop hive core component?**

The major components of Apache Hive are the Hive clients, Hive services, Processing framework and Resource Management, and the Distributed Storage.Number of available executors = (total cores/num-cores-per-executor) = 150/5 = 30. Leaving 1 executor for ApplicationManager => --num-executors = 29. Number of executors per node = 30/10 = 3. Memory per executor = 64GB/3 = 21GB.Number of available executors = (total cores/num-cores-per-executor) = 150/5 = 30. Leaving 1 executor for ApplicationManager => --num-executors = 29. Number of executors per node = 30/10 = 3. Memory per executor = 64GB/3 = 21GB.Number of available executors = (total cores/num-cores-per-executor) = 150/5 = 30. Leaving 1 executor for ApplicationManager => --num-executors = 29. Number of executors per node = 30/10 = 3. Memory per executor = 64GB/3 = 21GB.

**20. hive scripting?**

Hive scripts can be defined as a group of Hive commands bundled together to reduce the execution time.

**21. hive default storage location?**

HDFS in /usr/hive/warehouse directory or the directory set in hive-site.xml

**22. what is a Dynamic-Partition in hive?**

With dynamic partitioning in hive, partitions get created automatically at load times. New partitions can be created dynamically from existing data. Partitions are automatically created based on the value of the last column.

**23. What is the advantage of partitioning and bucketing Hive Table?**

Bucketed tables: Allow much more efficient sampling than the non-bucketed tables. With sampling, we can try out queries on a section of data for testing and debugging purpose when the original data sets are very huge.

Partitioned tables: Help to organize the data in logical fashion and when we query the partitioned table using partition column, it allows hive to skip all but relevant sub-directories and files.

**24. how to put data into hive?**

Using **Insert** Query.

Using Sqoop import.

Using Load Query.

Using HDFS put command

**25. how to find the number of rows in hive?**

show tblproperties yourTableName("rawDataSize")

**26. How to optimize hive queries?**

1. Execution Engine.
2. Usage of suitable file format.
3. By partitioning.
4. Use of bucketing.
5. Use of vectorization.
6. Cost based **optimization**.
7. Use of indexing.

**27. how many mappers are required for a query in hive?**

10 Mappers per Node

**28. how many reducers are required for a query in hive?**

10 Reducers per Node

**29. What is the hive bucketing formula to determine volume?**

(File Size)/(Number of Buckets) >= HDFS Block Size

**30. How does dynamic partition help in hive?**

When,

Loading from an existing table that is not partitioned

Unknown values for partition columns

**31. Why do we need dynamic partition to help in hive?**

Dynamic partition is useful when you load a huge file into a Table X and then you run an insert query into a Table Y. In case of dynamic partition, every row of the data is read and data is partitioned through an MapReduce job.

**32. Main difference between dynamic partition and static partition in hive.**

You can perform Static partition on Hive Manage table or external table. With dynamic partitioning in hive, partitions get created automatically at load times. New partitions can be created dynamically from existing data. Partitions are automatically created based on the value of the last column.

**33. How can we create dynamic partition column?**

INSERT OVERWRITE TABLE T PARTITION (ds, hr)

SELECT key, value, ds, hr FROM srcpart WHERE ds is not null and hr>10;

**34. Difference between orderby and sortby in hive?**

The difference between "order by" and "sort by" is that the former guarantees total order in the output while the latter only guarantees ordering of the rows within a reducer.

**35. What is a hive map side join?**

Map-side Joins allows a table to get loaded into memory ensuring a very fast join operation, performed entirely within a mapper and that too without having to use both map and reduce phases.

**36. Advantages of Hive?**

The main advantage of Apache Hive is for data querying, summarization, and analysis. Hive is designed for better productivity of the developer and also comes with the cost of increasing latency and decreasing efficiency. It is also built on top of HDFS

**37. How to put json into hive?**

Use the following:

ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'