# Interview Question Dictionary

# Common Career Question

## Tell me about yourself

* 1. **First of all thank you so much for sparing time for the technical discussion. Thanks**
  2. **Starting with Academics:** 
     + **I have done MTech from the BITS Pilani in software System.**
  3. **Experience**
     + **I have more than 3.5 years of experience in Big Data Technology using Hadoop and Spark Ecosystem.**
     + **However I have total 15 years of experience. left over all the experience exist in the Microsoft Technology using Microsoft SQL Server, C#, VB Dot Net, ASP Dot Net and Windows technology and SQL Server Management Studio.**
     + **My Major Involvement is in the SQL Server since start of my career and activity I performed as listed below.**
       - **Design**
       - **Development**
       - **Performance Tuning**
       - **Troubleshooting PROD environment**

## Tell me about yourself along with your current roles and responsibility in current project/organization (10 times)

1. **Current Roles and Responsibility: Currently I am the part of the Team Who is BROADLY engaged two type of activity**
   * + **Batch Process Migration of the Existing System: Conversion of the Batch Process to the Big Data and Hadoop. We have around 100+ process.**
     + **Data Processing for the prediction and making system robust.**
       - **Price Optimization**
       - **Most Visited product**
       - **Dynamic Customer Dashboard Creation**
       - **Most Frequent Error during the Site Processing**
     + **My Major Engagement is in Data Transformation**
       - **Hive**
       - **Spark SQL**
       - **SCALA**
2. **Big Data Life Cycle follows in our project.**
   * + **Data Ingestion**
       - **SQOOP**
       - **FLUME**
       - **SPARK STEAMING (POC)**
     + **Data Storage**
       - **HDFS**
       - **HBASE**
     + **Data Transformation**
       - **Pig (Earlier we worked)**
       - **Hive**
       - **Spark SQL**
       - **SCALA**
       - **Spark MLib (POC)**
         * **Clustering**
         * **Correlation and Regression**
     + **Data visualization**
       - **SQOOP**
       - **BO**
3. **Working Methodology**
   * + **We work in the Agile Methodology**
     + **We have daily scrum calls with Senior Technical Architect at USA**
       - **Share the status of work**
       - **Discuss the functional query.**
       - **Discuss and resolve any technical hurdle**

## What is the Reason of the change?

1. **A better opportunity**
2. **More challenges**
3. Career **growth**

## What is the current Architecture of your Batch Application in which you are working?

1. **Data Ingestion**
   * + **SQOOP**
     + **FLUME**
2. **Data Transformation**
   * + **Pig**
     + **Hive**
     + **Spark SQL**
3. **Data Visualization**
   * + **SQOOP**
     + **BO(Business Object)**

## Do you prefer good data or good models? Why?

1. **Many companies want to follow a strict process of evaluating data, means they have already selected data models. In this case, having good data can be game-changing.**
2. **The other way around also works as a model is chosen based on good data.**
3. **Don’t say that having both good data and good models is important as it is hard to have both in real life projects**

## Will you optimize algorithms or code to make them run faster?

1. **Yes, the interviewer might also be interested to know if you have had any previous experience in code or algorithm optimization.**
2. **However, be honest about your work, and it is fine if you haven’t optimized code in the past. Just let the interviewer know your real experience and you will be able to crack the big data interview.**

## How do you approach data preparation?

1. **Data preparation is required to get necessary data which can then further be used for modeling purposes. You should convey this message to the interviewer.**
2. **You should also emphasize the type of model you are going to use and reasons behind choosing that particular model.**
3. **You should also discuss important data preparation terms such as transforming variables, outlier values, unstructured data, identifying gaps, and others.**

## How would you transform unstructured data into structured data?

1. **The unstructured data should be transformed into structured data to ensure proper data analysis.**
2. **You can now discuss the methods you use to transform one form to another.**
3. **You might also share the real-world situation where you did it.**
4. **Follow Below Steps**
   * 1. **Define: What do you mean by structure? Identify attributes and required relationship between those attributes.**
     2. **Pattern Match: Now, its time to validate if all rows (or documents) has those parameters. You can safely remove those which does not confirm to your structure.**
     3. **Now Refine: Now, its time to do some refactoring on the data which you removed in step 2, Try to find out which attributes were missing, and do changes in your plan according.**
     4. **Repeat Step 3 and 2 as many times as you required.**
     5. **In the end, remaining rows can be discarded as data noise, or you can still try to find some pattern. Regular expression and knowledge of “What you are looking for” will be useful.**

# Spark

## Which version of spark you are using.

**Answer: 2.1.0**

## \*\*\* Out of Memory Error \*\*\*

1. **Memory Mangement**
   * 1. **Identify Error Source**
        1. **Worker**
        2. **Executor**
        3. **Task (Inside the Executor)**
     2. **Spark Memory = (System Memory – 300 MB)\*60%**
     3. **Before Spark 1.6 Executor Memory and Storage Memory are configurable**
     4. **After 1.6 Memory (Execution and Storage is dynamic) by Default each is half of Spark Memory**
     5. **If executor memory require more space** 
        1. **If Storage have available then it gives to executor memory**
        2. **If Storage Memory have not available then push the storage to disk and taken by Executor Memory**
     6. **If Storage Memory require more space** 
        1. **If Execution Memory have available then it gives to Storage memory**
        2. **If Execution Memory not available then does not do any thing.**
     7. **Error is any exception or Out of Memory**
2. **Choose Proper Partition**
   * 1. **Partition Count = No of Machines \* No of Cores in a Machine**
3. **Check for the Unbalance Partition: One file have too much nested data causing the issue**

## \*\*\* Spark Performance Increase \*\*\*

1. **User Proper format: ORC preferred with compression mode Snappy or Paraquet**
2. **Use the Operator Wisely**
   * 1. **Reduce by Key (Map Join) Vs Group By Key (Reduce Join)**
     2. **Repartition (costly) Vs Coalsec (less cost but for reducing the partition)**
3. **asda**

## \*\* Spark Job Monitor \*\*\* Application Logs are getting created how you monitor your spark job for the error identification and analysis like performance and speed. Which tool used to monitor the spark jobs?

## \*\* Tool Used to Analyze Data \*\*\* Once the processing is done and then which tool or utility used to analyze the output data.

## Where are you running the spark? Running on your local machine or cluster or what. (2 Times)

**Answer: On local machine during the development. By using the sbt project.**

## \*\*\* MapReduce Vs Spark \*\*\* What is the main difference between the MapReduce and Spark

|  |  |  |
| --- | --- | --- |
|  | SPARK | MAPREDUCE |
| **Speed** | Apache Spark runs applications up to 100x faster in memory and 10x faster on disk than Hadoop. | MapReduce reads and writes from disk |
| **Difficulty** | Easy to program as it has tons of high-level operators with [RDD – Resilient Distributed Dataset](http://data-flair.training/blogs/rdd-in-apache-spark/) | Developers need to hand code each and every operation |
| **Easy to Manage** | Spark is capable of performing   1. Batch 2. Interactive 3. Streaming   Makes it a complete [data analytics](http://data-flair.training/blogs/data-analytics-comprehensive-guide/) engine | MapReduce only provides the batch engine. we are dependent on different engines |
| **Real-time analysis** | It can process real time data i.e. data coming from the real-time event streams at the rate of millions of events per second | N/A |
| **Latency** | Low-latency computing. | High latency computing |
| **Interactive mode** | can process data interactively | N/A |
| **Streaming** | process real time data through [Spark Streaming](http://data-flair.training/blogs/apache-spark-streaming-comprehensive-guide/) | You can only process data in batch mode. |
| **Ease of use** | Spark is easier to use. | MapReduce is complex. |
| **Recovery** | RDDs allows recovery of partitions on failed nodes by re-computation of the [DAG](http://data-flair.training/blogs/directed-acyclic-graph-dag-in-apache-spark/) | MapReduce is naturally resilient to system faults or failures. |
| **Scheduler** | Due to [in-memory computation](http://data-flair.training/blogs/apache-spark-in-memory-computing/) spark acts its own flow scheduler. | MapReduce needs an external job schedule |
| **Security** | Spark is little less secure in comparison to MapReduce because it supports the only authentication through shared secret password authentication. | Apache Hadoop MapReduce is more secure because of Kerberos and it also supports Access Control Lists (ACLs) which are a traditional file permission model |
| **Cost** | As spark requires a lot of RAM to run in-memory. Thus, increases the cluster, and also its cost. | MapReduce is a cheaper option available while comparing it in terms of cost. |
| **Language Developed** | Spark is developed in [Scala](http://data-flair.training/blogs/why-you-should-learn-scala-introductory-tutorial/). | Hadoop MapReduce is developed in Java. |
| **Category** | It is [data analytics](http://data-flair.training/blogs/data-analytics-comprehensive-guide/) engine. Hence, it is a choice for [Data Scientist.](http://data-flair.training/blogs/valuable-skills-to-become-successful-data-scientist/) | It is basic data processing engine. |
| **The line of code** | Apache Spark is developed in merely 20000 line of codes. | Hadoop 2.0 has 1,20,000 line of codes |
| **Programming Language support** | Scala, Java, Python, R, SQL | Primarily Java, other languages like C, C++, Ruby, Groovy, Perl, Python are also supported using Hadoop streaming. |
| **SQL support** | SQL queries using [Spark SQL](http://data-flair.training/blogs/spark-sql-tutorial/). | It enables users to run SQL queries using Apache[Hive](http://data-flair.training/blogs/hive-tutorial-an-introductory-guide-for-beginners/). |
| **Machine Learning** | M;achine learning ie MLlib. | Hadoop requires machine learning tool for example Apache Mahout. |
| **Caching** | Spark can cache data in memory for further iterations. As a result it enhances the system performance. | MapReduce cannot cache the data in memory for future requirements.  So, the processing speed is not that high as that of Spark. |
| **Hardware Requirements** | [Spark](http://data-flair.training/blogs/apache-spark-introduction-tutorial/) needs mid to high-level hardware. | MapReduce runs very well on commodity hardware |

## \*\*\* Data Fit Vs Data Frame \*\*\*: What is the main difference between Data Fit and Data Frame?

[**https://data-flair.training/blogs/apache-spark-rdd-vs-dataframe-vs-dataset/**](https://data-flair.training/blogs/apache-spark-rdd-vs-dataframe-vs-dataset/)

|  |  |  |  |
| --- | --- | --- | --- |
|  | RDD APIs | DataFrame APIs | DataSets APIs |
| **Intro** | It is Read-only partition collection of records. | Data organized into named columns. For example a table in a relational database | Extension of DataFrame API which provides type-safe, object-oriented programming interface. |
| **Spark Release** | 1.0 | 1.3 | 1.6 |
| **Data Representation** | Distributed Collection of data elements spread across many machines in the cluster. | Distributed collection of data organized into named columns. It is conceptually equal to a table in a relational database. | Extension of DataFrame API that provides the functionality of – type-safe, object-oriented programming interface |
| **Data Formats** | 1. Easily and efficiently process data which is structured as well as unstructured. 2. Like Dataframe and DataSets, RDD does not infer the schema of the ingested data and requires the user to specify it. | 1. Process structured and unstructured data. 2. Organizes the data in the named column. 3. Spark to manage schema. | 1. Processes structured and unstructured data. 2. Represents data in the form of JVM objects of row or a collection of row object. |
| **Data Sources API** | RDD could come from any data source e.g. text file, a database via JDBC etc. and easily handle data with no predefined structure. | Data processing in different formats (AVRO, CSV, JSON, and storage system**[HDFS](http://data-flair.training/blogs/apache-hadoop-hdfs-introduction-tutorial/)**, [**HIVE**](http://data-flair.training/blogs/apache-hive-tutorial-introductory-guide/)tables, MySQL). | Dataset API of spark also support data from different sources. |
| **Immutability and Interoperability** | Each partition is one logical division of data which is immutable and created through some transformation on existing partitions. Immutability helps to achieve consistency in computations. We can move from RDD to DataFrame (If RDD is in tabular format) by **toDF()**method or we can do the reverse by the **.rdd** method. | After transforming into DataFrame one cannot regenerate a domain object. For example, if you generate testDF from testRDD, then you won’t be able to recover the original RDD of the test class. | It overcomes the limitation of DataFrame to regenerate the RDD from Dataframe. Datasets allow you to convert your existing RDD and DataFrames into Datasets. |
| **Compile-time type safety** | RDD provides a familiar object-oriented programming style with compile-time type safety | If you are trying to access the column which does not exist in the table in such case Dataframe APIs does not support compile-time error. It detects attribute error only at runtime. | It provides compile-time type safety. |
| **Optimization** | When working with structured data, RDDs cannot take advantages of sparks advance optimizers. For example, catalyst optimizer and Tungsten execution engine. Developers optimise each RDD on the basis of its attributes. | Optimization takes place using catalyst optimizer. Dataframes use catalyst tree transformation framework in four phases: a) Analyzing a logical plan to resolve references.  b) Logical plan optimization.  c) Physical planning.  d) Code generation to compile parts of the query to Java bytecode. | It includes the concept of Dataframe Catalyst optimizer for optimizing query plan. |
| **Serialization** | Whenever [**Spark**](http://data-flair.training/blogs/important-apache-spark-terminologies-and-concepts-you-must-know/)needs to distribute the data within the cluster or write the data to disk, it does so use Java serialization. | Spark DataFrame Can serialize the data into off-heap storage ([**in memory**](http://data-flair.training/blogs/apache-spark-in-memory-computing/)) in binary format and then perform many transformations directly on this off heap memory because spark understands the schema | When it comes to serializing data, the Dataset API in Spark has the concept of an encoder which handles conversion between JVM objects to tabular representation. |
| **Garbage Collection** | There is overhead for garbage collection that results from creating and destroying individual objects. | Avoids the garbage collection costs in constructing individual objects for each row in the dataset. | There is also no need for the garbage collector to destroy object because serialization takes place through Tungsten. That uses off heap data serialization. |
| **Efficiency/Memory use** | Efficiency is decreased when serialization is performed individually on a java and scala object which takes lots of time. | Use of off heap memory for serialization reduces the overhead. | It allows performing an operation on serialized data and improving memory use. |
| **Lazy Evolution** | Spark evaluates RDDs lazily. They do not compute their result right away. Instead, they just remember the transformation applied to some base data set. | Spark evaluates DataFrame lazily, that means computation happens only when action appears (like display result, save output). | It also evaluates lazily as RDD and Dataset. |
| **Programming Language Support** | RDD APIs are available in **Java, Scala, Python,** and[**R**](http://data-flair.training/blogs/r-programming-tutorial/) languages. | It also has APIs in the different languages like Java, Python, Scala, and R. | Dataset APIs is currently only available in Scala and Java. |
| **Schema Projection** | In RDD APIs use schema projection is used explicitly. Hence, we need to define the schema (manually). | Auto-discovering the schema from the files and exposing them as tables through the [**Hive Meta store**](http://data-flair.training/blogs/apache-hive-metastore/). | Auto discover the schema of the files because of using [**Spark SQL**](http://data-flair.training/blogs/spark-sql-tutorial/)engine. |
| **Aggregation** | RDD API is slower to perform simple grouping and aggregation operations | DataFrame API is very easy to use. It is faster for exploratory analysis, creating aggregated statistics on large data sets. | In Dataset it is faster to perform aggregation operation on plenty of data sets. |

## \*\*\* RDD/ Data Set / Data Frame Usage \*\*\* When to use the RDD and When Dataset/ Data Frame

|  |  |
| --- | --- |
| * You can use RDDs When you want low-level transformation and actions on your data set. * Use RDDs When you need high-level abstractions. | * One can use both DataFrame and dataset API when we need a high level of abstraction. * For unstructured data, such as media streams or streams of text. * You can use both Data Frames or Dataset when you need domain specific APIs. * When you want to manipulate your data with functional programming constructs than domain specific expression. * We can use either datasets or DataFrame in the high-level expression. For example, filter, maps, aggregation, sum, [**SQL**](https://data-flair.training/blogs/spark-sql-tutorial/) queries, and columnar access. * When you do not care about imposing a schema, such as columnar format while processing or accessing data attributes by name or column. * in addition, If we want a higher degree of type safety at compile time. |

## In RDD vs Dataframe, Is data frame is advantages to the end user or only advantages to the developer?

## \*\*\* No. of Executor and Memory \*\*\*

1. **Resources Information**
   * 1. **6 Machines**
     2. **16 Cores (Total cores in Cluster = 16\*6 = 96)**
     3. **64 GB RAM**
2. **Options**
   * 1. **Smallest Size Executor**
        1. **16 Core have 64 GB Ram 🡺 1 Core = 4GB RAM per executor**
        2. **16 Executor on each machine**
     2. **Biggest Size Executor**
        1. **6 Executor**
        2. **64 GB RAM per executor**
        3. **16 Cores per executor**
        4. **Problem**
           1. **IO Connection**
           2. **No Resources for OS**
           3. **No Memory overhead for YARN**
     3. **Right Approach**
        1. **No of Cores = Total Cores – No of Machines = 96 – 6 = 90**
        2. **No of Cores Per Machine = Tot Cores / Tot Machine = 90/6 = 15**
        3. **No of Cores per executor = 5**
        4. **No of Executors / Machine = Total Cores per machine / Total Cores per executor = 15/5 = 3**
        5. **Memory for Yarn overhead = 1 GB**
        6. **Memory for Per Executor = 63/3 = 21 GB**

# Scala

## What is advantage of writing a code in Scala

1. **Easy to write**
2. **Spark is also written in the Scala**

## How are you running the Scala code

1. **On the YARN not on the stand alone mode.**

# Hive

## How to optimized the hive query (2)

1. **De-normalizing data**
2. **Vectorization**
3. **Input Format Selection**
4. **Use ORCFile for Data Storage**
5. **Partitioning Tables**
6. **Bucketing**
7. **Compress map/reduce output**
8. **Map join**
9. **Parallel execution**
10. **Unit Testing**
11. **Sampling**
12. **De-normalizing data**
    * 1. **Normalization is a standard process used to model your data tables with certain rules to deal with redundancy of data and anomalies.**
      2. **If you normalize your data sets, you end up creating multiple relational tables which can be joined at the run time to produce the results.**
      3. **Joins are expensive and difficult operations to perform and are one of the common reasons for performance issues.**
      4. **It’s a good idea to avoid highly normalized table structures because they require join queries to derive the desired metrics.**
13. **Vectorization** 
    * 1. **Vectorization improves the performance by fetching 1,024 rows in a single operation instead of fetching single row each time.**
      2. **Operations are performed on the entire column vector, which improves the instruction pipelines and cache usage.**
      3. **To enable vectorization, set this configuration parameter SET hive.vectorized.execution.enabled=true.**
         1. **set hive.vectorized.execution.enabled=true;**
         2. **set hive.vectorized.execution.reduce.enabled=true;**
14. **Input Format Selection**
    * 1. **JSON, the text type of input formats. These type of readable formats actually take a lot of space and have some overhead of parsing (e.g JSON parsing).**
      2. **Use columnar input formats like RCFile, ORC etc. Columnar formats allow you to reduce the read operations in analytics queries by allowing each column to be accessed individually.**
      3. **There are some other binary formats like Avro, sequence files, Thrift and ProtoBuf, which can be helpful in various use cases too.**
15. **Use ORCFile for Data Storage**
    * 1. **Optimized Row Columnar format provides highly efficient ways of storing the hive data by reducing the data storage format by 75% of the original. The ORCFile format is better than the Hive files format when it comes to reading, writing, and processing the data. It uses techniques like predicate push-down, compression, and more to improve the performance of the query.**
      2. **Consider two tables: employee and employee\_details, tables that are stored in a text file. Let's say we will use join to fetch details from both tables.**
      3. **Select a.EmployeeID, a.EmployeeName, b.Address,b.Designation from Employee a Join Employee\_Details b On a.EmployeeID=b.EmployeeID;**
      4. **Above query will take a long time, as the table is stored as text. Converting this table into ORCFile format will significantly reduce the query execution time.**
      5. **Create Table Employee\_ORC (EmployeeID int, EmployeeName varchar(100),Age int)**

**STORED AS ORC tblproperties("compress.mode"="SNAPPY");**

**Select \* from Employee Insert into Employee\_ORC;**

**Create Table Employee\_Details\_ORC (EmployeeID int, Address varchar(100),Designation Varchar(100),Salary int)**

**STORED AS ORC tblproperties("compress.mode"="SNAPPY");**

* + 1. **Select \* from Employee\_Details Insert into Employee\_Details\_ORC;**
    2. **Select a.EmployeeID, a.EmployeeName, b.Address,b.Designation from Employee\_ORC a Join Employee\_Details\_ORC b On a.EmployeeID=b.EmployeeID;**
    3. **ORC supports compressed (ZLIB and Snappy), as well as uncompressed storage.**

1. **Partitioning Tables**
   * 1. **Partitioning allows you to store data in separate sub-directories under table location. It greatly helps the queries which are queried upon the partition key(s).**
     2. **Partition key is always a sensitive decision, e.g. if your data is associated with time dimension, then date could be a good partition key. Similarly, if data has association with location, like a country or state, then it’s a good idea to have hierarchical partitions like country/state.**
2. **Bucketing**
   * 1. **Bucketing improves the join performance if the bucket key and join keys are common.**
     2. **SET hive.enforce.bucketing=true (every time before writing data to the bucketed table.)**
     3. **SET hive.optimize.bucketmapjoin=true (hints to Hive to do bucket level join during the map stage join. )**
3. **Compress map/reduce output**
   * 1. **Compression techniques significantly reduce the intermediate data volume, which internally reduces the amount of data transfers between mappers and reducers.**
     2. **Compression can be applied on the mapper and reducer output individually.**
     3. **Keep in mind that gzip compressed files are not splittable.**
     4. **A compressed file size should not be larger than a few hundred megabytes. Otherwise it can potentially lead to an imbalanced job.**
     5. **Other options of compression codec could be snappy, lzo, bzip, etc.**

**•For map output compression set mapred.compress.map.output to true**

**•For job output compression set**

**mapred.output.compress to true**

1. **Map join**
   * 1. **Map joins are really efficient if a table on the other side of a join is small enough to fit in the memory.**
     2. **hive.auto.convert.join, which when it’s set to “true” suggests that Hive try to map join automatically**
2. **Parallel execution**
   * 1. **Complex Hive queries commonly are translated to a number of MapReduce jobs that are executed by default sequencing.**
     2. **Some of a query’s MapReduce stages are not interdependent and could be executed in parallel.**
     3. **The configuration in Hive to change this behavior is merely switching a single flag SET hive.exec.parallel=true**
3. **Unit Testing**
   * 1. **Unit testing esure your code works exactly as you expect.**
     2. **Unit testing Benefits i.e. detecting problems early, making it easier to change and refactor code**
     3. **In Hive, you can unit test UDFs, SerDes, streaming scripts, Hive queries and more.**
     4. **Some of them that you might want to look at HiveRunner, Hive\_test and Beetest.**
4. **Sampling**
   * 1. **Sampling allows users to take a subset of dataset and analyze it, without having to analyze the entire data set.**
     2. **Hive offers a built-in TABLESAMPLE clause that allows you to sample your tables.**
     3. **TABLESAMPLE can sample at various granularity levels** 
        1. **it can return only subsets of buckets (bucket sampling),**
        2. **or HDFS blocks (block sampling),**
        3. **or only first N records from each input split.**

## How to read the data in the hive from the csv file

1. **LOAD DATA IN PATH ‘hdfs folder name of the csv file’ INSERT OVERWRITE TABLE table\_name**
2. **For the CSV file reading we only provide the folder name not the file name**

## How to store the data in the ORC file

1. **Create Table Command**

**Create Table Employee\_Details\_ORC**

**(EmployeeID int, Address varchar(100),**

**Designation Varchar(100),**

**Salary int)**

**STORED AS ORC tblproperties("compress.mode"="SNAPPY");**

## How to optimize the query using joins

1. **Map Side Join:** 
   * 1. **Set Below Parameters**

**set hive.auto.convert.join=true;  
set hive.auto.convert.join.noconditionaltask=true;  
set hive.auto.convert.join.noconditionaltask.size=10000000;**

* + 1. **To force to use the map join we can use “MAPJOIN”**

**select /\*+ MAPJOIN(a) \*/ a.\* from passwords a, passwords2 b where a.col0=b.col0 ;**

1. **Sort Merge Bucket Join: it is a requirement that all the tables need to be created bucketed and sorted on the same join columns**
   * 1. **Data Insert**
        1. **create table c1(col0 string,col1 string,col2 string,col3 string,col4 string,col5 string,col6 string)  
           clustered by (col0) sorted by (col0) into 32 buckets;**
        2. **create table c2(col0 string,col1 string,col2 string,col3 string,col4 string,col5 string,col6 string)  
           clustered by (col0) sorted by (col0) into 32 buckets;**
        3. **set hive.enforce.bucketing = true;**
        4. **From passwords insert OVERWRITE  table c1 select \* order by col0;**
        5. **From passwords insert OVERWRITE  table c2 select \* order by col0;**
     2. **Property Set**
        1. **set hive.auto.convert.sortmerge.join=true;**
        2. **set hive.optimize.bucketmapjoin = true;**
        3. **set hive.optimize.bucketmapjoin.sortedmerge = true;**
        4. **set hive.auto.convert.sortmerge.join.noconditionaltask=true;**

## Incremental load in hive.

## Hive partition types and performance improvement using partitions.

## Spark is 10-20 times faster than Hive. Check the correctness

## What is the underlying Engine the Hive is using?

## What is the other engine the Hive is using?

# Microservices

## What is Micorservices

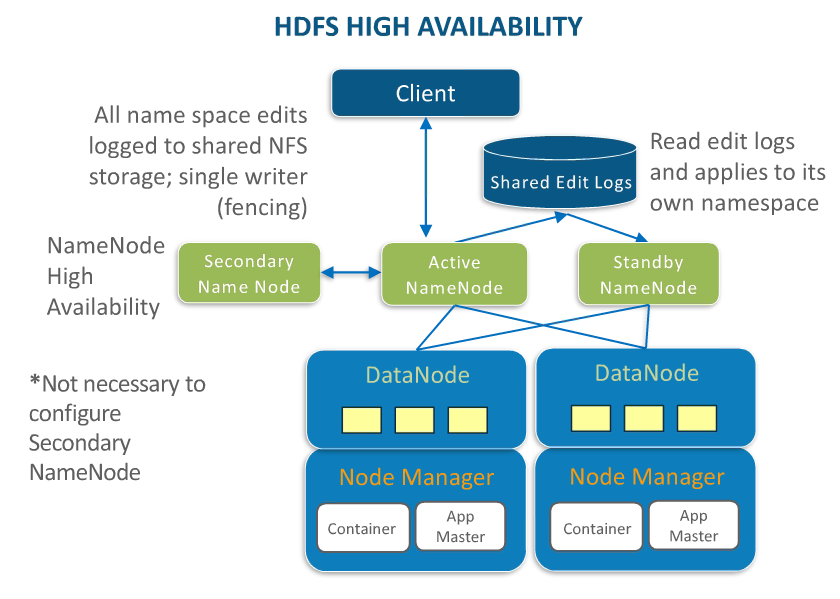
## How you used microservices in Big Data

# Hadoop Architecture

## What is the difference between Hadoop 1 and Hadoop 2. Or Difference between MRV1 and MRV2.

|  |  |  |  |
| --- | --- | --- | --- |
| **Serial** | **Property** | Hadoop 1.0 | Hadoop 2. |
| **1** | **Processing Model** | Supports MapReduce (MR) processing model only. Does not support non-MR tools | Allows to work in MR as well as other distributed computing models like Spark, Hama, Giraph, Message Passing Interface) MPI & HBase coprocessors. |
| **2** | **What does** | MR does both processing and cluster-resource management. | YARN (Yet Another Resource Negotiator) does cluster resource management and processing is done using different processing models. |
| **3** | **Nodes Scaling** | Has limited scaling of nodes. Limited to 4000 nodes per cluster | Has better scalability. Scalable up to 10000 nodes per cluster |
| **4** | **Work on Concept** | Works on concepts of slots – slots can run either a Map task or a Reduce task only. | Works on concepts of containers. Using containers can run generic tasks. |
| **5** | **Name node Details** | A single Namenode to manage the entire namespace. | Multiple Namenode servers manage multiple namespaces. |
| **6** | **Name node Failure** | Has Single-Point-of-Failure (SPOF) – because of single Namenode- and in the case of Namenode failure, needs manual intervention to overcome. | Has to feature to overcome SPOF with a standby Namenode and in the case of Namenode failure, it is configured for automatic recovery. |
| **7** | **MR Program Execute** | MR API is compatible with Hadoop1x. A program written in Hadoop1 executes in Hadoop1x without any additional files. | MR API requires additional files for a program written in Hadoop1x to execute in Hadoop2x. |
| **8** | **How Platform Serve** | Has a limitation to serve as a platform for event processing, streaming and real-time operations. | Can serve as a platform for a wide variety of data analytics-possible to run event processing, streaming and real-time operations. |
| **9** | **Name node failure affect** | A Namenode failure affects the stack. | The Hadoop stack – Hive, Pig, HBase etc. are all equipped to handle Namenode failure. |
| **10** | **Support to Microsoft WIndows** | Does not support Microsoft Windows | Added support for Microsoft windows |

## What is HA(High Availability)



## Hadoop cluster Initial Block size is 128 and data reside in the cluster. Now I want to increase or decrease the block size of the cluster. So how I have to manage my existing data? Does data impart or do we have to do any operations.

1. **If you change the block size in hdfs-site.xml configuration file, it won't affect the existing data which is already stored in HDFS.**

## Can I increase the Heart Beat

1. **We can increase it and configurable.**

## Tell me the current cluster size

1. **For Development we use single node cluster. But Production System has approx 40 machine cluster with 8 GB ram and 2TB hard disk**

## How much each node data capacity

1. **8/16 GB ram and 2TB hard disk**

## Why did you choose big data when we have many oracle application are there. When already existing database are there Teradata and oracle.

* 1. **Big Data has SQOOP to port data on HDFS and use spark/MR/HIve for processing**
  2. **Pointed query in Hbase is much faster than any RDBMS**
  3. **For unstructured data which is growing at good pace teradata /oracle cannot solve the purpose**

## Did you face any problem in using the Big Data application? Tell me some of the problem.

1. **When you have to do analysis over data, since most of data sets are un-structured.**
2. **When you have to clean data, since there are lakhs of columns and millions of rows in your dataset.**
3. **When you are installing software and there is a version mis-match.**
4. **Finding the correct Signal in the Noise**
5. **Dealing with data growth**
6. **Generating insights in a timely manner**
7. **Recruiting and retaining big data talent**
8. **Integrating disparate data sources**
9. **Validating data**
10. **Securing big data**
11. **Organizational resistance**

## What is fsck?

* 1. **fsck stands for File System Check, It is a command used by HDFS.**
  2. **This command is used to check inconsistencies and if there is any problem in the file.**
  3. **For example, if there are any missing blocks for a file, HDFS gets notified through this command.**

## What is the Command to format the NameNode?

1. **$ hdfs namenode –format**

## What happens when two users try to access the same file in the HDFS?

1. **HDFS NameNode supports exclusive write only. Hence, only the first user will receive the grant for file access and the second user will be rejected.**

## How to recover a NameNode when it is down?

1. **Use the FsImage which is file system metadata replica to start a new NameNode.**
2. **Configure the DataNodes and also the clients to make them acknowledge the newly started NameNode.**
3. **Once the new NameNode completes loading the last checkpoint FsImage which has received enough block reports from the DataNodes, it will start to serve the client.**

# Map Reduce

## What are the configuration parameters in a “MapReduce” program?

* 1. **Input locations of Jobs in the distributed file system**
  2. **Output location of Jobs in the distributed file system**
  3. **The input format of data**
  4. **The output format of data**
  5. **The class which contains the map function**
  6. **The class which contains the reduce function**
  7. **JAR file which contains the mapper, reducer and the driver classes**

## What is Distributed Cache in a MapReduce Framework

1. **Distributed Cache is a feature of Hadoop MapReduce framework to cache files for applications.**
2. **Hadoop framework makes cached files available for every map/reduce tasks running on the data nodes.**
3. **Hence, the data files can access the cache file as a local file in the designated job.**

# Oozie

## How much you are comfortable in Oozie. Did you configured and Oozie jobs.

## What type of scheduler are present in the big data

* 1. **FIFO scheduler:**

1. **An original Hadoop Job Scheduling Algorithm which was integrated within the JobTracker is the FIFO.**
2. **Basically, as a process, a JobTracker pulled jobs from a work queue, that says oldest job first, this is a Hadoop FIFO scheduling.**
3. **Moreover, this is simpler as well as efficient approach and it had no concept of the priority or size of the job.**
   1. **Fair Scheduler**
4. **Further, to give every user a fair share of the cluster capacity over time, we use the Fair Scheduler in Hadoop.**
5. **It gets all of the**[**Hadoop Clusters**](https://data-flair.training/blogs/hadoop-cluster/)**if a single job is running.**
6. **Further, free task slots are given to the jobs in such a way as to give each user a fair share of the cluster, as more jobs are submitted.**
7. **Furthermore, just set the mapred.jobtracker.taskScheduler property to: org.apache.hadoop.mapred.FairScheduler**
   1. **Capacity Scheduler**
8. **Except for one fact that within each queue, jobs are scheduled using FIFO scheduling in Hadoop (with priorities), this is like the Fair Scheduler.**
9. **It takes a slightly different approach for multiuser scheduling.**
10. **Moreover, for each user or an organization, it permits to simulate a separate MapReduce Cluster along with FIFO scheduling.**

## How you decide which scheduler is best and why.

1. **We concluded that the capacity scheduler is the right choice when we want to ensure guaranteed access with the potential in order to reuse unused capacity as well as prioritize jobs within queues, while we are running a large Hadoop cluster, along with the multiple clients.**
2. **When we use both small and large clusters for the same organization with a limited number of workloads, the fair scheduler works well. Also, in a simpler and less configurable way, it offers the means to non-uniformly distribute capacity to pools (of jobs). Furthermore, it can offer fast response times for small jobs mixed with larger jobs (supporting more interactive use models). Hence, it is useful in the presence of diverse jobs.**

# Machine Learning

## What is supervise and un- supervise learning in Machine Learning.

1. **LET’S LEARN SUPERVISED AND UNSUPERVISED LEARNING WITH A REAL LIFE EXAMPLE**

[Learn supervised and unsupervised learning with a real life example:**CLICK TO TWEET**](https://twitter.com/intent/tweet?url=https://dataconomy.com/?p=11299&text=Learn%20supervised%20and%20unsupervised%20learning%20with%20a%20real%20life%20example%3A&via=DataconomyMedia&related=DataconomyMedia)

[](https://dataaspirant.files.wordpress.com/2014/09/basket.jpg)

* suppose you had a basket and it is fulled with some different kinds of fruits, your task is to arrange them as groups.
* For understanding let me clear the names of the fruits in our basket.
* We have four types of fruits. They are:**apple, banana, grape and cherry.**

### SUPERVISED LEARNING:

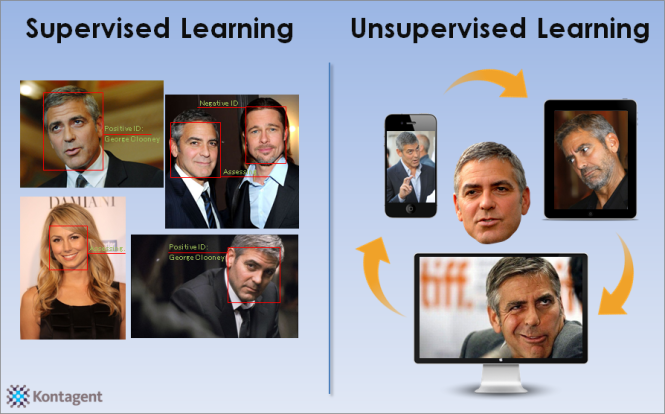
* You already learn from your previous work about the physical characters of fruits.
* So arranging the same type of fruits at one place is easy now.
* Your previous work is called as **training data** in data mining.
* So you already learn the things from your train data, this is because of **response variable.**
* Response variable mean just a **decision variable.**
* You can observe response variable below (**FRUIT NAME**) .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NO.** | **SIZE** | **COLOR** | **SHAPE** | **FRUIT NAME** |
| 1 | Big | Red | Rounded shape with a depression at the top | Apple |
| 2 | Small | Red | Heart-shaped to nearly globular | Cherry |
| 3 | Big | Green | Long curving cylinder | Banana |
| 4 | Small | Green | Round to oval,Bunch shape Cylindrical | Grape |

* Suppose you have  taken an new fruit from the basket then you will see the size , color and shape of that particular fruit.
* If  size  is Big , color is Red , shape is rounded shape with a depression at the top, you will conform the fruit name as apple and you will put in apple group.
* Likewise for other fruits also.
* Job of groping fruits was done and happy ending.
* You can observe in the table that  a column was labeled as “**FRUIT NAME**” this is called as response variable.
* If you learn the thing before from training data and then applying that knowledge to the test data(for new fruit), This type of learning is called as**Supervised Learning**.
* **Classification** comes under supervised learning.

### UNSUPERVISED LEARNING

* Suppose you had a basket and it is fulled with some different types fruits, your task is to arrange them as groups.
* This time you don’t know any thing about that fruits, honestly saying this is the first time you have seen them.
* so how will you arrange them.
* What will you do first???
* You will take a fruit and you will arrange them by considering physical character of that particular fruit. suppose you have considered color.
* Then you will arrange them on considering base condition as **color.**
* Then the groups will be some thing like this.
* RED COLOR GROUP: apples & cherry fruits.
* GREEN COLOR GROUP: bananas & grapes.
* so now you will take another physical character such as **size** .
* RED COLOR AND BIG SIZE: apple.
* RED COLOR AND SMALL SIZE: cherry fruits.
* GREEN COLOR AND BIG SIZE: bananas.
* GREEN COLOR AND SMALL SIZE: grapes.
* job done happy ending.
* Here you didn’t know learn any thing before ,means no train data and no response variable.
* This type of learning is know unsupervised learning.
* clustering comes under unsupervised learning.

[](https://dataaspirant.files.wordpress.com/2014/09/george-clooney5.png)

# Microservices

## How you used Microservices in your project.

# Interview Status

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Serial #** | **Company Name** | **Position** | **Interview Date** | **Round** | **I&Q Logged** |
| **1** | **Tech Mahindra** | **Technical Architect** | **08-Sep-18** | **1** | **Yes** |
| **2** | **Tech Mahindra** | **Solution Architect** | **11-Oct-18** | **1** | **Yes** |
| **3** | **Team Computers** | **Technical Architect** | **12-Oct-18** | **1** | **Yes** |
| **4** | **HCL Technology** | **Technical Architect** | **13-Oct-18** | **1** | **Yes** |
| **5** | **Tech Mahindra** | **Scala Developer** | **03-Jan-19** | **2** | **Yes** |
| **6** | **NIIT Technology** | **Technical Architect** | **08-Jan-19** | **1** | **Yes** |
| **7** | **Sapient** | **Senior position in Big Data** | **17-Jan-19** | **1** | **Yes** |
| **8** | **Luxoft** | **Senior Big Data Development Engineer** | **07-May-19** | **1** |  |
| **9** | **prevalent.AI** | **Big Data Architect** | **14-May-19** | **1** |  |
| **10** | **Met Life** | **Big Data Architect** | **28-Jun-19** | **1** |  |
| **11** | **IBM** | **Big Data Senior Technical Architect** | **23-Aug-19** | **1** | **Yes** |
| **12** | **Knoldus Software** | **Tech Lead Big Data** | **23-Sep-19** | **1** |  |