Questions

Stay updated wit
latest technologi
trends
Join DataFlair o
Telegram!!

1. Apache Spark

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

Questions - Objective

Apache Spark is prevailing because of i capability to handle re time streaming and processing big data fas than **Hadoop MapReduce**. As the demand for Spark developers are expected to grow in a lightning fast manner, it is the golden time to polish your Apache Spark knowledge and build up your career as a **data analytics** professional, **data scientist** or **big data** developer. This guide on spark interview questions and answers will help you to improve the skills that will shape you for Spar developer job roles. This section contains top 50 Apache Spark Interview Questions and Answer. Hope these questions will help you to crack the Spark interview. Happy Job Hunting!

Questions.

50 Frequently Asked Apache Spark Interview Questions

2. Top 50 Apache Spark Interview Questions and Answers

Let's proceed further with Apache Spark Interview Questions and Answer-

1) What is Apache Spark? What is the reason behind the evolution of this framework?

Ans. Spark is an open source big data framework. It has an

efficiently execute streaming as well as the batch. Apache Spark provides faster and more general data processing platform engine. It basically designed for fast computation and developed at UC Berkeley in 2009. Spark is an Apache project which is also call as "lighting fast cluster computing". It distributes data in a file system across the cluster, and process that data in parallel. Spark covers a wide range of workloads like batch applications, iterative algorithms, interactive queries and streaming. It lets you write an application in Java, Python or Scala.

It was developed to overcome the limitations of **MapReduce** cluster computing paradigm. Spark keeps things in memory whereas map reduces keep shuffling things in and out of disk. It allows to cache data in

algorithm those used in machine learning.

Spark is easier to develop as it knows how to operate on data. It supports SQL queries, streaming data as well as graph data processing. Spark doesn't need Hadoop to run, it can run on its own using other storages like Cassandra, S3 from which spark can read and write. In terms of speed spark run programs up to 100x faster in memory or 10x faster on disk than Map Reduce.

2) Explain the features of Apache Spark because of which it is superior to Apache MapReduce?

Ans. Hadoop is designed for batch processing. Batch processing is very efficient in the processing of high volume data. Hadoop MapReduce is batch-oriented

processes it and produces a result.
Hadoop MapReduce adopted the batch-oriented model. Batch is essentially processing data at rest, taking a large amount of data at once and producing output. MapReduce process is slower than spark because due to produce a lot of intermediary data.

Spark also supports batch processing system as well as stream processing.

Spark streaming processes data streams in micro-batches, Micro batches are an essentially collect and then process kind of computational model. Spark processes faster than map reduces because it caches input data in memory by **RDD**.

3) Why is Apache Spark faster than

Ans. Apache Spark is faster than Apache Hadoop due to below reasons:

- Apache Spark provides **in-memory computing**. Spark is designed to transform data In-memory and hence reduces time for disk I/O. While MapReduce writes intermediate results back to Disk and reads it back.
- Spark utilizes **Direct Acyclic Graph** that helps to do all the optimization and computation in a single stage rather than multiple stages in the MapReduce model
- Apache Spark core is developed using **SCALA** programming language which is faster than JAVA. SCALA provides inbuilt concurrent execution by providing immutable

Thread to achieve parallel execution.

4) List down the languages supported by Apache Spark.

Ans. Apache Spark supports Scala, **Python**, **Java**, and R.
Apache Spark is written in Scala. Many people use Scala for the purpose of development. But it also has API in Java, Python, and R.

5) What are the components of Apache Spark Eco-system?

Ans. Spark Core

Spark Core is the base Spark for parallel and distributed processing of huge datasets. It is in charge of all the essential I/O functionalities, programming, and observance the roles on spark clusters. It is also

networking with different storage systems, **fault tolerance**, and economical memory management. Core uses special collection referred to as **RDD (Resilient Distributed Datasets)**.

- **Spark SQL**

SparkSQL is module/component in Apache Spark that is employed to access structured and semi-structured information. It is a distributed framework that is tightly integrated with varied spark programming like **Scala**, Python, and Java. Spark SQL supports relative process in Spark programs via RDD further as on external data source. It is used manipulating and taking information in varied formats. The means through that {we can|we will|we square measure able to} act with Spark SQL are

The main abstraction in SparkSQL is information sets that act on structured data. It translates ancient SQL and HiveQL queries into Spark jobs creating Spark accessible wide. It supports real-time data analytics, data streaming SQL.**SparkSQL defines 3 varieties of function:**

1. **Built-in perform or user-defined function:** Object comes with some functions for column manipulation. Using Scala we are able to outlined user outlined perform.
2. **Aggregate Function:** Operates on the cluster of rows and calculates one come back price per cluster.
3. **Window Aggregate:** Operates on the cluster of rows and calculates one come back price every

Different type of APIs for accessing SparkSQL:

SQL: Executing SQL queries or **Hive** queries, the result is going to become in the variety of DataFrame.

1. **DataFrame:** It is similar to the relative table in SparkSQL and distributed the assortment of tabular information having rows and named column. It will perform filter, intersect, join, sort mixture and much more. DataFrames powerfully trusts **options of RDD**. As it trusts RDD, it is **lazy evaluated** and immutable in nature. DataFrameAPI is offered in Scala, Java, and Python.
2. **Datasets API:** Dataset is new API to supply benefit

declarative in nature. Dataset is the assortment of object or records with the familiar schema. It should model in some data structure. DataSets API offers improvement of DataFrames and static kind safety of Scala. We can convert information set to Data Frame.

- **Spark Streaming**

Spark Streaming is a light-weight API that permits developers to perform execution and streaming of information application. Discretized Streams kind the bottom abstraction in Spark Streaming. It makes use of an endless stream o {input information|input file|computer file} to method data in the time period. It leverages the quick programming capability of Apache Spark core to perform streaming analytics by

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

Information in Spark Streaming accepts from varied information sources and live streams like Twitter, Apache Kafka, IoT Sensors, Amazon response, Apache Flume, etc. in an event-driven, fault-tolerant, and type-safe applications.

- **Spark element MLlib**
  MLlib in Spark stands for machine learning (ML) library. Its goal is to form sensible machine learning effective, ascendible and straightforward. It consists of some learning algorithms and utilities, as well as classification, regression, clustering collaborative filtering, spatial property reduction, further as lower-level improvement primitives and higher-level pipeline genus Apis.

graph process framework on prime of Apache Spark. because it predicate on RDDs, that square measure immutable, graphs square measure immutable and so GraphX is unsuitable for graphs that require being updated, in addition to in an exceedingly transactional manner sort of a graph info.

For more details, please read Apache Spark Eco-system.

6) Is it possible to run Apache Spark without Hadoop?

Ans. Yes, Apache Spark can run without Hadoop, standalone, or in the cloud. Spark doesn't need a Hadoop cluster to work. It can read and then process data from other file systems as well. **HDFS** is just one of

Spark does not have any storage layer, so it relies on one of the distributed storage systems for distributed computing like HDFS, Cassandra etc.

However, there are a lot of advantages to running Spark on top of Hadoop (HDFS (for storage) + **YARN** (resource manager)), but it's not the mandatory requirement. It meant for distributed computing. In this case, the data distribute across the computers and Hadoop's distributed file system HDFS is used to store data that does not fit in memory.

One more reason for using Hadoop with Spark is they both are open source and both can integrate with each other rather easily as compared to other data storage system.

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

[Spark Compatibility with Hadoop]

7) What is RDD in Apache Spark? How are they computed in Spark? what are the various ways in which it can create?

Ans. RDD in Apache Spark is the representation of a set of records, it is the immutable collection of objects with distributed computing. RDD is the large collection of data or an array of reference of partitioned objects. Each and every dataset in RDD is logically partitioned across many servers so that they can compute on different nodes of the cluster. [RDDs are fault tolerant] i.e. self-recovered / recomputed in the case of failure. The dataset could data load externally by the users which can be in the form of JSON file, CSV file, text file or database via

RDD is Lazily Evaluated i.e. it is memorized or called when required or needed, which saves lots of time. RDD is a read-only, partitioned collection of data. RDDs can creates through deterministic operations or on stable storage or from other RDDs. It can also generates by parallelizing an existing collection in your application or referring a dataset in an external storage system. It is cacheable. As it operates on data over multiple jobs in computations such as logistic regression, k-means clustering, PageRank algorithms, which makes it reuse or share data among multiple jobs.

To learn more about the RDD follow: RDD

3. Apache Spark Interview Questions For Beginners

8) What are the features of RDD, that makes RDD an important abstraction of Spark?

Ans. RDD (Resilient Distributed Dataset) is a basic abstraction in **Apache Spark**. Spark RDD is an immutable, partitioned collection of elements on the cluster which can operates in parallel.

Each RDD is characterized by five main properties :

Below operations are lineage operations.

- List or Set of partitions.

RDD

- A function to compute each partition

Below operations are used for optimization during execution.

- Optional preferred location **[i.e. block location of an HDFS file] [it's about data locality]**
- Optional partitioned info **[i.e. Hash-Partition for Key/Value pair –> When data shuffled how data will travel]**

Examples :

#

HadoopRDD: HadoopRDD provides core functionality for reading data stored in Hadoop (**HDFS**, **HBase**, Amazon S3..) using the older **MapReduce** API (org.apache.hadoop.mapred)

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

1. *List or Set of partitions: One per HDFS block.*
2. *List of dependencies on parent RDD: None.*
3. *A function to compute each partition: read respective HDFS block*
4. *Optional Preferred location: HDFS block location*
5. *Optional partitioned info: None*

#FilteredRDD **:** Properties of FilteredRDD:

1. *List or Set of partitions: No. of partitions same as parent RDD*
2. *List of dependenci on parent RDD: 'one-to-one' as the parent (same as parent)*
3. *A function to compute each partition: compute parent and then filter it*

Parent)

5. *Optional partitioned info: None*

Find features of RDD in RDD Features in Spark

9) List out the ways of creating RDD in Apache Spark.

Ans. There are three ways to create **RDD**

(1) By Parallelizing collections in the driver program

(2) By loading an external dataset

(3) Creating RDD from already existing RDDs.

Create RDD By Parallelizing collections :

Parallelized collections are created by calling **parallelize**() method on an existing collection in driver program.

```
1.  val rdd1 =
    Array(1,2,3,4,5
2.  val rdd2 =
    sc.parallelize
```

OR

```
1.  val myList =
    sc.parallelize
    to 1000), 5) wh
    is the number o
    partitions
2.    [If we do not
    specify then de
    partition is 1
```

Create by loading an external Dataset

In Spark, the distributed dataset can form from any data source supported by Hadoop, including the local file system, HDFS, Cassandra, **HBase** etc. In this, the data is loaded from the external dataset. To create text file RDD, we can use SparkContext's textFile method. It takes URL of the file and read it as a collection of line. URL can be a local path on the machine or a hdfs://, s3n://, etc. Use SparkSession.read to

DataFrameReader supports many file formats-

i) csv (String path)

```
1.  import org.apa
2.  def main(args:
3.  object DataFor
4.  val spark =
    SparkSession.bu
5.  val dataRDD =
```

ii) json (String path)

```
1.  val dataRDD =
    spark.read.jso
```

iii) textFile (String path)

```
1.  val dataRDD =
    spark.read.text
```

Creating RDD from existing RDD:

Transformation mutates one RDD into another RDD, thus transformation is the way to create an RDD from already existing RDD.

```
1.  val
    words=spark.spa
    "quick", "brown
    "the", "lazy",
2.    "dog"))
3.  val wordPair =
    w))
4.  wordPair.forea
```

description on creating RDD read [How to create RDD in Apache Spark]

10) Explain Transformation in RDD. How is lazy evaluation helpful in reducing the complexity of the System?

Ans. Transformations are lazy evaluated operations on RDD that create one or many new RDDs, e.g. map, filter, reduceByKey, join, cogroup, randomSplit. Transformations are functions which take an RDD as the input and produces one or many RDDs as output. They don't change the input RDD as RDDs are immutable and hence cannot change or modify but always produces new RDD by applying the computations operations on them. By applying transformations you incrementally build an

final RDD(s).

Transformations are lazy, i.e. are not executed immediately. Transformations can execute only when actions are called. After executing a transformation, the result RDD(s) will always be different from their ancestors RDD and can be smaller (e.g. filter, distinct, sample), bigger (e.g. flatMap, union, cartesian) or the same size (e.g. map) or it can vary in size.

RDD allows you to create dependencies b/w RDDs. Dependencies are the steps for producing results in a program. Each RDD in lineage chain, string of dependencies has a function for operating its data and has a pointer dependency to its ancestor RDD. Spark will divide RDD dependencies into stages and tasks and

[Follow this link to read more]

11) What are the types of Transformation in Spark RDD Operations?

Ans. There are two kinds of transformations:

Narrow transformations:

Narrow transformations are the result of map, filter and in which data to transform id from a single partition only, i.e. it is self-sustained.
An output RDD has partitions with records that originate from a single partition in the parent **RDD**.

Wide Transformations

Wide transformations are the result of groupByKey and reduceByKey. The

partition may reside in many partitions of the parent RDD.

Wide transformations are also called shuffle transformations as they may or may not depend on a shuffle. All of the tuples with the same key must end up in the same partition, processed by the same task. To satisfy these operations, **Spark** must execute RDD shuffle, which transfers data across the cluster and results in a new stage with a new set of partitions.

12) What is the reason behind Transformation being a lazy operation in Apache Spark RDD. How is it useful?

Ans. Whenever a **transformation operation** is performed in **Apache Spark**, it is lazily evaluated. It won't execute until an action is

transformation operation to the **DAG (Directed Acyclic Graph)** of computation, which is a directed finite graph with no cycles. In this DAG, all the operations are classified into different stages, with no shuffling of data in a single stage.

By this way, Spark can optimize the execution by looking at the DAG at its entirety, and return the appropriate result to the driver program.

For example, consider a 1TB of a log file in HDFS containing errors, warnings, and other information. Below are the operations being performed in the driver program:

1. **Create an RDD** of this log file
2. It perform a flatmap() operation on this **RDD** to split the data in the log file based on tab delimiter.

data containing only error messages
4. Perform first() operation to fetch only the first error message.

If all the transformations in the above driver program are eagerly evaluated, then the whole log file will load into memory, all of the data within the file will split base on the tab, now either it needs to write the output of FlatMap somewhere or keep it in the memory. Spark needs to wait until the next operation is performed with the resource blocked for the upcoming operation. Apart from this for each and every operation spark need to scan all the records, like for FlatMap process all the records then again process them in filter operation.

On the other hand, if all the transformations are lazily evaluated, Spark

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

the execution plan for the application, now this plan will optimize the operation will combine/merge into stages then the execution will start. The optimized plan created by Spark improves job's efficiency and overall throughput.

By this lazy evaluation in Spark, the number of switches between driver program and cluster is also reduced thereby saving time and resources in memory, and also there is an increase in the speed of computation.

13) What is RDD lineage graph? How is it useful in achieving Fault Tolerance?

Ans. The RDD Lineag Graph or RDD operator graph could be a graph of the entire parent **RDDs** of an RDD. It's engineered as a result of materializing **transformations to the RDD** and then

The RDDs in Apache Spark rely on one or a lot of alternative RDDs. The illustration of dependencies in between RDDs is understood because of the lineage graph. Lineage graph info is employed to cypher every RDD on demand, so whenever a district of persistent RDD is lost, {the data | the info | the info} that's lost will recover using the lineage graph information.

For details on RDD DAG refer to Directed Acyclic Graph

14) Explain the various Transformation on Apache Spark RDD like distinct(), union(), intersection(), and subtract().

Ans. distnct() transformation – If one wants only unique elements in a **RDD**, in

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

RDD

Example

```
1.  val d1 =
    sc.parallelize
2.  val result = d
3.  result.foreach
```

OutPut:

p
t
m
c

- **union() transformation –** Its simplest set operation. rdd1.union(rdd2) which outputs RDD which contains the data from both sources. If the duplicates are present in the input RDD, an output of union() transformation wil[1] contain duplicate also which can fix using distinct().

Example

```
1.  val u1 =
    sc.parallelize
2.  val u2 =
    sc.parallelize
3.  val result = u
```

Output:

c
c
p
m
t
c
m
k

- **intersection() transformation –** It returns the elements which are present in both the RDDs and remove all the duplicate including duplicated in single RDD

```
1.  val is1 =
    sc.parallelize
2.  val is2 =
    sc.parallelize
3.  val result = i
4.  result.foreach
```

Output :

m
c

- **subtract() transformation –** Subtract(anotherrdd), returns an RDD that has an only value present in the first

Example

```
1.  val s1 =
    sc.parallelize
2.  val s2 =
    sc.parallelize
3.  val result = s
4.  result.foreach
```

Output:

t

p

For more transformation in Apache Spark refer to Transformation and Action

15) What is the FlatMap Transformation in Apache Spark RDD?

Ans. FlatMap is a **transformation operation in Apache Spark** to **create an RDD** from existing **RDD**. It takes one element from an RDD and can produce 0, 1 or many outputs based on business logic. It is similar to Map operation, but Map produces one to one output. If we perform

RDD will also be of length N. But for FlatMap operation output RDD can be of different length based on business logic

X——A x————a
Y——B y————b,c
Z——C z————d,e,f

Map Operation FlatMap Operation

We can also say as flatMap transforms an RDD of length N into a collection of N collection, then flattens into a single RDD of results.

If we observe the below example data1 RDD which is the output of Map operation has same no of element as of data RDD,
But data2 RDD does not have the same number of elements. We can also observe here as data2 RDD is a flattened output of data1 RDD

pranshu@pranshu-virtual-machine:~$ cat pk.txt

5 6 7 8 9

10 11 12

13 14 15 16 17

18 19 20

```
scala> val dat
sc.textFile("/h
```

17/05/17 07:08:20 WARN SizeEstimator: Failed to check whether UseCompressedOops is set; assuming yes

data: org.apache.spark.rdd.RDD[String] = /home/pranshu/pk.txt MapPartitionsRDD[1] at textFile at <console>:24

```
scala>
data.collect
```

res0: Array[String] = Array(1 2 3 4, 5 6 7 8 9, 10 11 12, 13 14 15 16 17, 18 19 20)

```
scala>
scala> val
data1 =
data.map(line
=>
line.split("
"))
```

= MapPartitionsRDD[2]
at map at <console>:26

```
scala>
scala> val dat
=
data.flatMap(l
=> line.split('
"))
```

data2:
org.apache.spark.rdd.RDD[String]
= MapPartitionsRDD[3]
at flatMap at
<console>:26

```
scala>
scala>
data1.collect
```

res1:
Array[Array[String]] =
Array(Array(1, 2, 3, 4),
Array(5, 6, 7, 8, 9),
Array(10, 11, 12),
Array(13, 14, 15, 16, 17),
Array(18, 19, 20))

```
scala>
scala>
data2.collec
```

res2: Array[String] =
Array(1, 2, 3, 4, 5, 6, 7, 8,
9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20)

For more details,
refer: Map Vs

16) Explain first() operation in Apache Spark RDD.

Ans. It is an action and returns the first element of the **RDD**.

Example :

```
1.  val rdd1 = sc.textFile("/h data.txt")
2.  rdd1.count
3.  rdd1.first
```

Output :

Long: 20

String: DataFlair is the leading technology training provider

4. Apache Spark Interview Questions For Intermediate

Spark Tutorials +

Spark Interview Questio... ×

- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

outer join supported?

Ans. join() is transformation and is in package org.apache.spark.rdd.pairRDDFunction

def join[W](other: RDD[(K, W)]): RDD[(K, (V, W))]Permalink

Return an RDD containing all pairs of elements with matching keys in this and other.

Each pair of elements will returns as a (k, (v1, v2)) tuple, where (k, v1) is in this and (k, v2) is in other. Performs a hash join across the cluster.

It is joining two datasets. When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for e... key. Outer joins are supported through leftOuterJoin, rightOuterJoin, and fullOuterJoin.

Example 1:

```
val rdd1 = sc.
```

```
3.  val joinrdd =
4.  joinrdd.collec
```

Output:

Array[(String, (Int, Int))] = Array((m,(55,60)), (m,(55,65)), (m,(56,60)), (m,(56,65)), (s,(59,61)), (s,(59,62)), (s,(54,61)), (s,(54,62)))

Example 2:

```
1.  val myrdd1 = sc.parallelize
2.  val myrdd2 = sc.parallelize
3.  val myjoinedrc
4.  myjoinedrdd.cc
```

Output:

Array[(Int, (Int, Int))] = Array((3,(4,9)), (3,(6,9)))

18) Describe coalesce() operation. When can you coalesce to a larger number of partitio... Explain.

Ans. It is a **transformation** and it's in a package **org.apache.spark.rdd.ShuffledRDD**

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

Int, shuffle: Boolean = false, partitionCoalescer: Option[PartitionCoalescer] = Option.empty) (implicit ord: Ordering[(K, C)] = null): RDD[(K, C)]

Return a new **RDD** that is reduced into numPartitions partitions.

This results in a narrow dependency, e.g. if you go from 1000 partitions to 100 partitions, there will not be a shuffle, instead, each of the 100 new partitions will claim 10 of the current partitions.

However, if you're doing a drastic coalesce, e.g. to numPartitions = 1, this may result in your computation taking pl on fewer nodes than you like (e.g. one node in the case of numPartitions = 1). To avoid this, you can pass shuffle = true. This will add a shuffle step but means the current upstream partitions will

partitioning is).

Note: With shuffle = true, you can actually coalesce to a larger number of partitions. This is useful if you have a small number of partitions, say 100, potentially with a few partitions being abnormally large. Calling coalesce(1000, shuffle = true) will result in 1000 partitions with the data distributed using a hash partitioner.

Coalesce() operation changes a number of the partition where data is stored. It combines original partitions to a new number of partitions, so it reduces the number of partitions. Coalesce() operation is an optimized version of repartition that allows data movement, but only if you are decreasing the number of RDD partitions. It runs operations more efficiently after filtering large datasets.

```
1. val myrdd1 =
   sc.parallelize
   1000, 15)
2. myrdd1.partiti
   h
3. val myrdd2 =
   myrdd1.coalesce
4. myrdd2.partiti
   h
5. Int = 5
```

Output :

Int = 15

Int = 5

19) Explain pipe() operation. How it writes the result to the standard output?

Ans. It is a transformation.

def pipe(command: String): RDD[String]

Return an RDD created by piping elements to a forked external process.

- In general, Spark is using Scala, Java, and Python to write the program. However, if that is not enough, and one want to pipe (inject) the data which written in other languages like

mechanism in the form of pipe() method.

- Spark provides the pipe() method on RDDs.
- With Spark's pipe() method, one can write a transformation of an RDD that can read each element in the RDD from standard input as String.
- It can write the results as String to the standard output.

For more transformation on RDDs see: Apache Spark Operations

20) What is the key difference between textFile and wholeTextFile method?

Ans. Both are the method of org.apache.spark.SparkContext.

textFile() :

Int = defaultMinPartitions): RDD[String]

- Read a text file from HDFS, a local file system (available on all nodes), or any Hadoop-supported file system URI, and return it as an RDD of Strings
- For example sc.textFile("/home/hdadmin/wc-data.txt") so it will create RDD in which each individual line an element.
- Everyone knows the use of textFile.

wholeTextFiles() :

- def wholeTextFiles(path: String, minPartitions: Int = defaultMinPartitio RDD[(String, String)]
- Read a directory of text files from HDFS, a local file system (available on all nodes), or any Hadoop-supported file system URI.

wholeTextFile() returns pairRDD.

- For example, you have few files in a directory so by using wholeTextFile() method, it creates pair RDD with a filename with a path as key, and value is the whole file as a string

```
1.  val myfilerdd
    sc.wholeTextFi
2.  val keyrdd = m
3.  keyrdd.collect
4.  val filerdd =
5.  filerdd.collec
```

Output :

Array[String] =
Array(
file:/home/hdadmin/MyFiles/JavaSparkPi.java,
file:/home/hdadmin/MyFiles/sumnumber.txt,
file:/home/hdadmin/MyFiles/JavaHdfsLR.java,
file:/home/hdadmin/MyFiles/JavaPageRank.java,
file:/home/hdadmi...
file:/home/hdadmin/MyFiles/wc-data.txt,
file:/home/hdadmin/MyFiles/nosum.txt)

Array[String] =

Array(“/*

Spark Tutorials +
Spark Interview Questio... ×
✦ Spark Interview Questions – 1
✦ Spark Interview Questions – 2
✦ **Spark Interview Question...**
✦ Spark Interview Questions – 4
Spark Quiz +

Foundation (ASF) under one or more

- Contributor license agreements. See the NOTICE file distributed with
- This work for additional information regarding copyright ownership.
- The ASF licenses this file to You under the Apache License, Version 2.0
- The “License”; you may not use this file except in compliance with
- The License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

- Unless required by applicable law or agreed to in writing, software
- Distributed under the License is distributed on an “AS IS” BASIS,
- WITHOUT WARRANTIES OR

express or implied.
- See the License for the specific language governing permissions and

21) What is Action, how it process data in Apache Spark?

Ans. Actions return final result of **RDD** computations/operat... It triggers execution using **lineage graph** to load the data into original RDD, and carries out all intermediate transformations and returns final result to Driver program or write it out to file system.

For example: First, take, reduce, collect, count, aggregate are some of the actions in spark.

Action produces a value back to the **Apache Spark** driver program. It may trigger a previously constructed, **lazy RDD** to evaluate. It is an

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

Action function materializes a value in a Spark program. So basically an action is RDD operation that returns a value of any type but RDD[T] is an action. Actions are one of two ways to send data from executors to the driver (the other being accumulators).

For detail study of Action refer **Transformation and Action in Apache Spark**.

22) How is Transformation on RDD different from Action?

Ans. Transformations create new RDD from existing RDD
Transformations are executed on demand. (**Lazy computation**)
Ex: filter(), union()

An **Action** will return a non-RDD type (your stored value types usually)

load the data into original RDD
Ex: count(), first()

23) What are the ways in which one can know that the given operation is Transformation or Action?

Ans. In order to identify the operation, one needs to look at the return type of an operation.

- **If the operation returns a new RDD, in that case, an operation is 'Transformation'**
- **If the operation returns any other type than RDD, in that case, an operation is 'Action'**

Hence, Transformation constructs a new RDD from an existing one (previous one) while Action computes the result based on applied transformation and

save it to the external storage.

Also, refer to operations of RDD in Apache Spark and its Operations

24) Describe Partition and Partitioner in Apache Spark.

Ans. Partition in Spark is similar to split in HDFS. A partition in Spark is a logical division of data stored on a node in the cluster. They are the basic units of parallelism in Apache Spark. RDDs are a collection of partitions. When some actions are executed, a task is launched per partition.

By default, partitions are automatically created by the framework. However, the number of partitions in Spark are configurable to suit the needs. For the number of partitions, if spark.default.parallelism is set, then we should use

defaultParallelism, othewrwise we should use the max number of upstream partitions. Unless spark.default.parallelism is set, the number of partitions will be the same as that of the largest upstream RDD, as this would least likely cause out-of-memory errors.

A partitioner is an object that defines how the elements in a key-value pair RDD are partitioned by key, maps each key to a partition ID from 0 to numPartitions – 1. It captures the data distribution at the output. With the help of partitioner, the scheduler can optimize the future operations. The contract of partitioner ensures that records for a given key have to reside on a single partition.

We should choose a partitioner to use for a cogroup-like operations.

we should choose that one. Otherwise, we use a default HashPartitioner.

There are three types of partitioners in Spark :

- Hash Partitioner
- Range Partitioner
- Custom Partitioner

Hash – Partitioner:
Hash- partitioning attempts to spread the data evenly across various partitions based on the key.

Range – Partitioner:
In Range- Partitioning method, tuples having keys with same range will appear on the same machine.

RDDs can create with specific partitioning in two ways :

i) Providing explicit partitioner by calling partitionBy method on an RDD

ii) Applying transformations that

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

5. Apache Spark Interview Questions For Experience

25) How can you manually partition the RDD?

Ans. When we create the **RDD** from a file stored in **HDFS**.

```
1.  data = context.textFi
    name")
```

By default one partition is created for one block. ie. if we have a file of size 1280 MB (with 128 MI block size) there will be 10 HDFS blocks, hence the similar number of partitions (10) will create.

If you want to create more partitions than the number of blocks, you

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

```
data = context.textFi
name", 20)
```

It will create 20 partitions for the file. ie for each block 2 partitions will create.

NOTE: It is often recommended to have more no of partitions than no of the block, it improves the performance

26) Name the two types of shared variable available in Apache Spark.

Ans. There are two types of shared variables available in **Apache Spark**:

- **Accumulators**: used to Aggregate ... Information.
- **Broadcast variable**: to efficiently distribute large values.

When we pass the function to Spark, say

defined outside of the function but within the Driver program but when we submit the task to Cluster, each worker node gets a new copy of variables and update from these variables not propagated back to Driver program.

Accumulators and Broadcast variable are used to remove above drawback (i.e. we can get the updated values back to our Driver program)

27) What are accumulators in Apache Spark?

Ans. This discussion is in continuation with a question, Name the two types of shared variable available in Apache Spark.

Introduction of Accumulator :

- An accumulator is a shared variable in **Apache Spark**, used to aggregating

- In other words, aggregating information/values from worker nodes back to the driver program. (How we will see in below session)

Why Accumulator :

- When we use a function inside the operation like map(), filter() etc these functions can use the variables which defined outside these function scope in the driver program.
- When we submit the task to cluster, each task running on the cluster gets a new copy of these variables and updates from these variabl do not propagate back to the driver program.
- *Accumulator* lowers this restriction.

Use Cases :

accumulator counts the events that occur during job execution for debugging purpose.
- Meaning count the no. of blank lines from the input file, no. of bad packets from a network during a session, during Olympic data analysis we have to find age where we said (age != 'NA') in SQL query in short finding bad/corrupted records.

Examples :

```
1.  scala> val rec
    spark.read.text
    data-blankline
```

record:
org.apache.spark.sql.D
= [value: string]

```
1.  scala> val
    emptylines =
    sc.accumulator
    warning: there
    were two
    deprecation
```

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

mptylines: org.apache.spark.Accumulator[Int] = 0

```
scala> val processdata = record.flatMap => {
  if(x == "")
    emptylines += 1
  x.split(" ")
})
```

processdata: org.apache.spark.sql.Dataset[String] = [value: string]

```
scala> processdata.col
```

16/12/02 20:55:15 WARN SizeEstimator: Failed to check whether UseCompressedOops is set; assuming yes

Output :

res0: Array[String] = Array(DataFlair, provides, training, on, cutting, edge, technologies., “”, DataFlair, is, the, leading, training, provider,, we, have, trained, 1000s, of,

aspects, which, industy, needs, rather, than, theoretical, knowledge., “”, DataFlair, helps, the, organizations, to, solve, BigData, Problems., “”, Javadoc, is, a, tool, for, generating, API, documentation, in, HTML, format, from, doc, comments, in, source, code., It, can, be, downloaded, only, as, part, of, the, Java, 2, SDK., To, see, documentation, generated, by, the, Javadoc, tool,, go, to, J2SE, 1.5.0, API, Documentation., “”, Javadoc, FAQ, -, This, FAQ, covers, where, to, download, the, Javadoc, tool,, how, to, find, a, list, of, known, bugs, and, feature, reque...

scala> println(“No. of Empty Lines : ” + emptylines.value)

No. of Empty Lines: 10

28) Explain SparkContext in

Ans. A **SparkContext** is a client of Spark's execution environment and it acts as the master of the **Spark** application. SparkContext sets up internal services and establishes a connection to a Spark execution environment. You can **create RDDs**, accumulators and broadcast variables, access Spark services and run jobs (until SparkContext stops) after the creation of SparkContext. Only one SparkContext may be active per JVM. You must stop() the active SparkContext before creating a new one.

In Spark shell, a special interpreter-aware SparkContext is already created for the user, in the variable called sc.

The first step of any Spark driver application is to create a SparkContext. The SparkContext allows the

through a resource manager. The resource manager can be **YARN**, or **Spark's Cluster Manager**.

Few functionalities which SparkContext offers are:

1. We can get the current status of a Spark application like configuration, app name.
2. We can set Configuration like master URL, default logging level.
3. One can create Distributed Entities like **RDDs.**

29) Discuss the role of Spark driver in Spark application.

Ans. The spark driver is that the program that defines the **transformations and actions on RDDs** of knowledge and submits a request to the master. Spark driver is a

machine which declares transformations and actions on knowledge RDDs.

In easy terms, the driver in Spark creates **SparkContext**, connected to a given Spark Master. It conjointly delivers the RDD graphs to Master, wherever the standalone cluster manager runs.

Also, see How Spark works

30) What role does worker node play in Apache Spark Cluster? And what is the need to register a worker node with the driver program?

Ans. Apache Spark follows a master/slave architecture, with one master or driver process and more than one slave or worker processes

1. The master is the driver that runs the

context is created. It then interacts with the cluster manager to schedule the job execution and perform the tasks.

2. The worker consists of processes that can run in parallel to perform the tasks scheduled by the driver program. These processes are called executors.

Whenever a client runs the application code, the driver programs instantiates Spark Context, converts the **transformations and actions** into logical **DAG** of execution. This logical DAG is then converted into a physical execution plan, which is then broken down into smaller physical execution units. The driver then interacts with the cluster manager to negotiate the resources required to perform the tasks of the application

with each of the worker nodes to understand the number of executors running in each of them.

The role of worker nodes/executors:

1. Perform the data processing for the application code
2. Read from and write the data to the external sources
3. Store the computation results in memory, or disk.

The executors run throughout the lifetime of the Spark application. This is a static allocation of executors. The user can also decide how many numbers of executors are required to run the tasks, dependi on the workload. This is a dynamic allocation of executors.

Before the execution of tasks, the executors are registered with the driver program through the

many numbers of executors are running to perform the scheduled tasks. The executors then start executing the tasks scheduled by the worker nodes through the cluster manager.

Whenever any of the worker nodes fail, the tasks that are required to perform will automatically allocates to any other worker nodes

For information on how Spark works Spark-How it works

31) Discuss the various running mode of Apache Spark.

Ans. We can launch spark application in four modes:

1) Local Mode (local[*],local,local[2]... etc)

-> When you launch spark-shell without

Spark Tutorials +

Spark Interview Questio... ×

- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

in local mode

spark-shell –master local[1]

-> spark-submit –class com.df.SparkWordCount SparkWC.jar local[1]

2) Spark Standalone cluster manger:

-> spark-shell –master spark://hduser:7077

-> spark-submit –class com.df.SparkWordCount SparkWC.jar spark://hduser:7077

3) Yarn mode (Client/Cluster mode):

-> spark-shell –master yarn or

(or)

->spark-shell –master yarn –deploy-mode client

Above both commands are same.

To launch spark application in cluster mode, we have to use a spark-submit command.

shell because when we run spark application, driver program will be running as part application master container/process. So it is not possible to run cluster mode via spark-shell.

-> spark-submit –class com.df.SparkWordCount SparkWC.jar yarn-client

-> spark-submit –class com.df.SparkWordCount SparkWC.jar yarn-cluster

4) Mesos mode:

-> spark-shell –master mesos://HOST:5050

32) Describe the run-time architecture of Spark.

Ans. There are 3 important components of Runtime architecture of Apache Spark as described below.

- Client process
- Driver

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

Responsibilities of the client process component

The client process starts the driver program.

For example, the client process can be a spark-submit script for running applications, a spark-shell script, or a custom application using Spark API. The client process prepares the classpath and all configuration options for the Spark application.

It also passes application arguments, if any, to the application running on the driver.

Responsibilities of the driver component

The driver orchestrate- and monitors the execution of a Spark application. There's always one driver per Spark application.

The driver is like a wrapper around the

(the Spark context and scheduler) are responsible for:

- requesting memory and CPU resources from cluster managers
- breaking application logic into stages and tasks
- sending tasks to executors
- collecting the results

Responsibilities of the executors

The executors, which is a JVM processes, accept tasks from the driver, execute those tasks, and return the results to the driver. Each executor has several task slots (or CPU cores) for running tasks in parallel.

6. Best Apache Spark Interview

and Answers

33) What is the command to start and stop the Spark in an interactive shell?

Ans. Command to start the interactive shell in Scala:

```
1.  >>>bin/spark
-shell
```

First, go the spark directory i.e.

```
1.  hdadmin@ubuntu
1.6.1-bin-hadoo
2.  hdadmin@ubuntu
1.6.1-bin-hadoo
bin/spark-shell
```

Command to stop the interactive shell in Scala:

```
1.  scala>Press
(Ctrl+D)
```

One can see the following message

```
1.  scala>
Stopping
spark
context.
```

Ans. Spark SQL is a Spark interface to work with Structured and Semi-Structured data (data that as defined fields i.e. tables). It provides abstraction layer called **DataFrame** and **DataSet** through with we can work with data easily. One can say that DataFrame is like a table in a relational database. Spark SQL can read and write data in a variety of Structured and Semi-Structured formats like Parquets, JSON, Hive. Using SparkSQL inside Spark application is the best way to use it. This empowers us to load data and query it with SQL. we can also combine it with "regular" program code in Python Java or **Scala**.

For a detailed study on SparkSQL, Refer link: **Spark SQL**

35) What is SparkSession in

Ans. Starting from **Apache Spark** 2.0, Spark Session is the new entry point for Spark applications.

Prior to 2.0, **SparkContext** was the entry point for spark jobs. **RDD** was one of the main APIs then, and it was created and manipulated using Spark Context. Every other APIs, different contexts were required – For SQL, SQL Context was required;
For **Streaming**, Streaming Context was required; For **Hive**, Hive Context was required.

But from 2.0, RDD along with DataSet and its subset **DataFrame** A are becoming the standard APIs and are a basic unit of data abstraction in Spark. All of the user-defined code will be written and evaluated against the

So, there is a need for a new entry point build for handling these new APIs, which is why Spark Session has been introduced. Spark Session also includes all the APIs available in different contexts – Spark Context, SQL Context, Streaming Context, Hive Context.

36) Explain API create Or Replace TempView().

Ans. It's basic Dataset function and under org.apache.spark.sql

- **def createOrReplaceTempView(viewName: String): Unit**
- **Creates a temporary view using the given name.**
- **The lifetime of this temporary view is tied to the SparkSession that was used to create this Dataset.**

df:
org.apache.spark.sql.DataFrame
= [_c0: string, _c1: string
... 9 more fields]

```
1.  scala>
    df.printSchema
```

root

|– _c0: string (nullable =
true)

|– _c1: string (nullable =
true)

|– _c2: string (nullable =
true)

|– _c3: string (nullable =
true)

|– _c4: string (nullable =
true)

|– _c5: string (nullable =
true)

|– _c6: string (nullabl
true)

|– _c7: string (nullable =
true)

|– _c8: string (nullable =
true)

Spark Tutorials +

Spark Interview Questio... ×

- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

|– _c10: string (nullable
= true)

```
scala>
df.show
```

+–+–+–+–––––––––
+––-+–––––+
–––––––––+––-+
–––––––+––+––+

|_c0|_c1|_c2| _c3| _c4|
_c5| _c6| _c7| _c8| _c9|
_c10|

+–+–+–+–––––––––
+––-+–––––+
–––––––––+––-+
–––––––+––+––+

| 1|1st| 1|Allen, Miss
Elisa...|29.0000|Southampton|
St Louis, MO| B-5| 24160
L221| 2|female|

| 2|1st| 0|Allison, Miss
Hel...|
2.0000|Southampton|Montreal,
PQ / Ch...| C26| null|
null|female|

| 3|1st| 0|Allison, Mr
Hudso...|30.0000|Southampton|Montreal,
PQ / Ch...| C26| null|
(135)| male|

Spark Tutorials +

Spark Interview Questio... ×

- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

PQ / Ch...| C26| null|
null|female|

| 5|1st| 1|Allison, Master
H...|
0.9167|Southampton|Montreal,
PQ / Ch...| C22| null| 11|
male|

| 6|1st| 1| Anderson, Mr
Harry|47.0000|Southampton|
New York, NY| E-12|
null| 3| male|

| 7|1st| 1|Andrews, Miss
Kor...|63.0000|Southampton|
Hudson, NY| D-7| 13502
L77| 10|female|

| 8|1st| 0|Andrews, Mr
Thoma...|39.0000|Southampton|
Belfast, NI| A-36| null|
null| male|

| 9|1st| 1|Appleton, Mrs
Edw...|58.0000|Southampton|
Bayside, Queens, NY| C-
101| null| 2|female|

| 10|1st| 0|Artagaveytia,
Mr ...|71.0000|
Cherbourg| Montevideo,
Uruguay| null| null| (22)|
male|

Cherbourg| New York, NY| null|17754 L224 10s 6d|(124)| male|

| 12|1st| 1|Astor, Mrs John J...|19.0000| Cherbourg| New York, NY| null|17754 L224 10s 6d| 4|female|

| 13|1st| 1|Aubert, Mrs Leont...| NA| Cherbourg| Paris, France| B-35| 17477 L69 6s| 9|female|

| 14|1st| 1|Barkworth, Mr Alg...| NA|Southampton| Hessle, Yorks| A-23| null| B| male|

| 15|1st| 0| Baumann, Mr John D.| NA|Southampton| New York, NY| null| null| null| male|

| 16|1st| 1|Baxter, Mrs James...|50.0000| Cherbourg| Montreal, PQ|B-58/60| null| 6|female|

| 17|1st| 0|Baxter, Mr Quigg ...|24.0000| Cherbourg| Montreal,

| 18|1st| 0| Beattie, Mr Thomson|36.0000| Cherbourg| Winnipeg, MN| C-6| null| null| male|

| 19|1st| 1|Beckwith, Mr Rich...|37.0000|Southampton| New York, NY| D-35| null| 5| male|

| 20|1st| 1|Beckwith, Mrs Ric...|47.0000|Southampton| New York, NY| D-35| null| 5|female|

+—+—+—+————————+——-+————+————-+————+————+

only showing top 20 rows

```
1.  scala>
    df.createOrRepl
```

37) What are the various advantages of DataFrame over RDD in Apache Spark?

Ans. DataFrames are the distributed collection of data. In DataFrame, data is organized into named columns. It is

database.
we can construct DataFrames from a wide array of sources. Such as structured data files, tables in Hive, external databases, or existing RDDs.

As same as **RDDs**, DataFrames are evaluated lazily(**Lazy Evaluation**). In other words, computation only happens when an action (e.g. display result, save output) is required.

Out of the box, DataFrame supports reading data from the most popular formats, including JSON files, Parquet files, Hive tables. Also, can read from distributed file systems (**HDFS**), local file systems, cloud storage (S3), and external relational database systems through JDBC. In addition, through **Spark SQL's** external data sources API, DataFrames

or sources. Existing third-party extensions already include Avro, CSV, ElasticSearch, and Cassandra.

There is much more to know about DataFrames. Refer link: **Spark SQL DataFrame**

38) What is a DataSet? What are its advantages over DataFrame and RDD?

Ans. In **Apache Spark**, Datasets are an extension of DataFrame API. It offers object-oriented programming interface. Through Spark SQL, it takes advantage of **Spark's Catalyst optimizer** by exposing e data fields to a query planner.

In **SparkSQL**, Dataset is a data structure which is strongly typed and is a map to a relational schema. Also, represents structured queries with encoders. DataSet has

Spark Tutorials +

Spark Interview Questio... ×

- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

In serialization and deserialization (SerDe) framework, encoder turns out as a primary concept in Spark SQL. Encoders handle all translation process between JVM objects and Spark's internal binary format. In Spark, we have built-in encoders those are very advanced. Even they generate bytecode to interact with off-heap data.

On-demand access to individual attributes without having to de-serialize an entire object is provided by an encoder. Spark SQL uses a SerDe framework, to make input-output time and space efficient. Due to encoder knows the schema of record, it became possible to achieve serialization as well as deserialization.

Spark Dataset is structured and lazy query expression(**lazy**

dataset represents a logical plan. The logical plan tells the computational query that we need to produce the data. the logical plan is a base catalyst query plan for the logical operator to form a logical query plan. When we analyze this and resolve we can form a physical query plan.

As Dataset introduced after **RDD** and **DataFrame**, it clubs the features of both. It offers the following similar features:

1. The convenience of RDD.
2. Performance optimization of DataFrame.
3. Static type-safety of Scala.

Hence, we have observed that Datasets provides a more functional programming interface to work with structured data.

DataSets, refer link: **Spark Dataset**

39) On what all basis can you differentiate RDD, DataFrame, and DataSet?

Ans. DataFrame: A Data Frame is used for storing data into tables. It is equivalent to a table in a relational database but with richer optimization. Spark DataFrame is a data abstraction and domain-specific language (DSL) applicable on a structure and semi-structured data. It is distributed the collection of data in the form of named column and row. It has a matrix-like structure whose column may be different types (numeric, logical, factor, or character). We can say data frame has the two-dimensional array like structure where each column contains the value of one variable and row contains one set of values for each column

For more details about DataFrame, please refer: **DataFrame in Spark**

RDD is the representation of a set of records, immutable collection of objects with distributed computing. RDD is a large collection of data or RDD is an array of reference of partitioned objects. Each and every dataset in RDD is logically partitioned across many servers so that they can compute on different nodes of the cluster. RDDs are fault tolerant i.e. self-recovered/recomputed in the case of failure. The dataset can load externally by the users which can be in the fo... of JSON file, CSV file, text file or database via JDBC with no specific data structure.

DataSet in Apache Spark, Datasets are an extension of DataFrame

interface. Through Spark SQL, it takes advantage of Spark's Catalyst optimizer by exposing e data fields to a query planner.

For more details about RDD, please refer: **RDD in Spark**

For the detailed comparison between RDD vs DataFrame, follow: RDD vs DataFrame vs DataSet

40) What is Apache Spark Streaming? How is the processing of streaming data achieved in Apache Spark? Explain.

Ans. Data arriving continuously, in an unbounded sequence is a data stream. Continuously flowing input data is divided into discrete units with the help of streaming for further processing. Through Stream processing analyzing of

latency processing.

In the year 2103 Spark Streaming was introduced to Apache Spark. It is an extension of the core Spark API. Streaming offers scalable, high-throughput and **fault-tolerant** stream processing of live data streams. It is possible to do Data ingestion from many sources. For Example Apache Flume, Kafka, Amazon Kinesis or TCP sockets. And, By using complex algorithms that are expressed with high-level functions processing can do. For example reduce, map, join and window. Afterwards, processed data can push out to live dashboards, filesystems and databases.

Streaming's Key abstraction is Discretized Stream. It is also known as **Spark DStream**. A stream of data divided into small batches is

Spark's core data abstraction"RDDs". Streaming allows integration with any other Apache Spark components like **Spark SQL** and **Spark MLlib**.

To know more about Spark Streaming, follow the link: **Spark Streaming Tutorial for Beginners**

7. Latest Apache Spark Interview Questions

41) What is the abstraction of Spark Streaming?

Ans. A **Discretized Stream (DStream)**, the basic abstraction in Spark Streaming, is a continuous sequence of **RDDs**representing a continuous stream of data. DStreams can either

from **HDFS**, Kafka or Flume) or it can generate by **transformation** existing DStreams using operations such as map, window and reduceByKeyAndWindow.

Internally, there are few basic properties by which DStreams is characterized:

1. DStream depends on the list of other DStreams.
2. A time interval at which the DStream generates an RDD
3. A function that is used to generate an RDD after each time interval

for complete introduction, refer link: **Apache Spark DStream (Discretiz Streams)**

42) Explain what are the various types of Transformation on DStream?

Ans. There are two types of transformation on

- **Stateless transformation:** In stateless transformation, the processing of each batch does not depend on the data of its previous batches.
- **Stateful transformation:** Stateful transformation use data or intermediate results from previous batches to compute the result of the current batch.

Also, refer to Transformation operation in Spark Streaming.

43) Explain the level of parallelism in Spark Streaming. Also, describe its need.

Ans. In order to reduce the processing time, one need to increase the parallelism. In Spark Streaming, there are

1. **Increase the number of receivers :** If there are too many records for single receiver (single machine) to read in and distribute so that is bottleneck. So we can increase the no. of receiver depends on scenario.
2. **Re-partition the receive data :** If one is not in a position to increase the no. of receivers in that case redistribute the data by re-partitioning.
3. **Increase parallelism in aggregation :**

for complete guide on Spark Streaming you may refer to Apache Spark-Streaming guide

44) Discuss writeahead logging in Apache Spark Streaming.

Ans. There are two types of failures in any **Apache**

the **worker failure**.

When any worker node fails, the executor processes running in that worker node will kill, and the tasks which were scheduled on that worker node will be automatically moved to any of the other running worker nodes, and the tasks will accomplish.

When the driver or master node fails, all of the associated worker nodes running the executors will kill, along with the data in each of the executors' memory. In the case of files being read from reliable and fault tolerant file systems like HDFS, zero data loss is always guaranteed, as the data is ready to be read anytime from the file system. Checkpointing also ensures **fault tolerance in Spark** by periodically saving the application data in specific intervals.

zero data loss is not always guaranteed, as the data will buffer in the executors' memory until they get processed. If the driver fails, all of the executors will kill, with the data in their memory, and the data cannot recover.

To overcome this data loss scenario, **Write Ahead Logging (WAL) has been introduced in Apache Spark 1.2.** With WAL enabled, the intention of the operation is first noted down in a log file, such that if the driver fails and is restarted, the noted operations in that log file can apply to the data. For sources that read streaming data, like Kafka or Flume, receivers will be receiving the data, and those will store in the executor's memory. With WAL enabled, these received data will also store in the log files.

1. Setting the checkpoint directory, by using streamingContext.checkpoint(path)

2. Enabling the WAL logging, by setting spark.stream.receiver.WriteAheadLog.enable to True.

45) What are the roles of the file system in any framework?

Ans. In order to manage data on computer, one has to interact with the File System directly or indirectly. When we install Hadoop on our computer, actually there are two file system exists on machine

(1) Local File System ,
(2) HDFS (Hadoop Distributed File System)

HDFS is sits top on of Local File System.

Following are the genera functions of File System (be it Local or HDFS)

- Control the data access mechanism (i.e

Spark Tutorials +
Spark Interview Questio... ×
✦ Spark Interview Questions – 1
✦ Spark Interview Questions – 2
✦ Spark Interview Question...
✦ Spark Interview Questions – 4
Spark Quiz +

- Manages the metadata about the Files / Folders (i.e. created date, size etc)
- Grants the access permission and manage the securities
- Efficiently manage the storage space

For more details, please follow: HDFS

46) What do you mean by Speculative execution in Apache Spark?

Ans. The **Speculative task** in **Apache Spark** is task that runs slower than the rest of the task in the job.It is health check process that verifies the task is speculated, meaning the task that runs slower than the median of successfully completed task in the task sheet. Such tasks are submitted to another worker. It runs the new copy in parallel rather than shutting down the slow task.

as *TaskSchedulerImp1with spark.speculation enabled. It executes periodically every spark.speculation.interval after the initial spark.speculation.interval* passes.

8. Apache Spark Interview Questions for Practice

47) How do you parse data in XML? Which kind of class do you use with java to pass data?

Ans. One way to parse the XML data in Java is to use the JDOM library. One can download it and import the JDOM library in your project. You can get help from Google. If still, required help post your problem in the forum. I will try to give you the solution. For Scala, Scala has the

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- Spark Interview Question...
- Spark Interview Questions – 4

Spark Quiz +

1.0.2 jar (please check them for new version if available).

48) Explain Machine Learning library in Spark.

Ans. It is a scalable machine learning library. It delivers both blazing speed (up to 100x faster than **MapReduce**) and high-quality algorithms (e.g., multiple iterations to increase accuracy). We can use this library in Java, **Scala**, and Python as part of Spark applications so that you can include it incomplete workflows. There are many tools, which are provided by MLlib. Such as-

- **ML Algorithms:** Common learning algorithms such as classification, regression, clustering, and collaborative filtering.
- **Featurization:** Feature extraction,

reduction, and selection.
- **Pipelines:** Tools for constructing, evaluating, and tuning ML Pipelines.
- **Persistence:** Saving and load algorithms, models, and Pipelines.
- **Utilities:** Linear algebra, statistics, data handling, etc.

For detailed insights, follow link: **Apache Spark MLlib (Machine Learning Library)**

49) List various commonly used Machine Learning Algorithm.

Ans. Basically, there are three types of Machine Learning Algorithms :

(1) Supervised Learning Algorithm

(2) Unsupervised Learning Algorithm

Spark Tutorials +
Spark Interview Questio... ×
✦ Spark Interview Questions – 1
✦ Spark Interview Questions – 2
✦ **Spark Interview Question...**
✦ Spark Interview Questions – 4
Spark Quiz +

Most commonly used Machine Learning Algorithm is as follows :

1. Linear Regression
2. Logistic Regression
3. Decision Tree
4. K-Means
5. KNN
6. SVM
7. Random Forest
8. Naïve Bayes
9. Dimensionality Reduction Algorithm
10. Gradient Boost and Adaboost

For what is MLlib see Apache Spark Ecosystem

50) Explain the Parquet File format in Apache Spark. When is it the best to choose this?

Ans. Parquet is the columnar information illustration that is that the best choice for storing long run massive information for analytics functions. It will perform each scan and write

Spark Tutorials +
Spark Interview Questio... ×
- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

columnar information storage format.

Parquet, create to urge the benefits of compressed, economical columnar information illustration accessible to any project, despite the selection of knowledge process framework, data model, or programming language.

Parquet could a format which will process by a variety of various systems: **Spark-SQL**, Impala, **Hive**, Pig, niggard etc. It doesn't lock into a particular programming language since the format is outlined exploitation, Thrift that supports numbers of programming languages. as an example, Aepyceros melampus is written in C++ whereas Hive is written in Java however they will simply interoperate on an equivalent Parquet information.

Answers. Hope you like the Apache spark Interview Questions and Answers.

3. Conclusion – Advance Apache Spark Interview Questions

Here we have covered all the top Apache spark interview questions which you can encounter in you spark interview. You can share the spark interview questions that you have faced in your interview and what was you experience in it. Also, for any feedback on Apache spark interview questions, feel free to comment.

Spark Tutorials +

Spark Interview Questio... ×

- Spark Interview Questions – 1
- Spark Interview Questions – 2
- **Spark Interview Question...**
- Spark Interview Questions – 4

Spark Quiz +

Answers

LEAVE A REPLY

Comment

Name * **Email** *

Post Comment