1. **What is Spark?**

   Spark is a parallel data processing framework. It allows to develop fast, unified big data application combine batch, streaming and interactive analytics.

2. **Why Spark?**

   Spark is third generation distributed data processing platform. It's unified bigdata solution for all bigdata processing problems such as batch , interacting, streaming processing.So it can ease many bigdata problems.

3. **What is RDD?**

   Spark's primary core abstraction is called Resilient Distributed Datasets. RDD is a collection of partitioned data that satisfies these properties. Immutable, distributed, lazily evaluated, catchable are common RDD properties.

4. **What is Immutable?**

   Once created and assign a value, it's not possible to change, this property is called Immutability. Spark is by default immutable, it's not allows updates and modifications. Please note data collection is not immutable, but data value is

   immutable.

5. **What is Distributed?**

   RDD can automatically the data is distributed across different parallel computing nodes.

6. **What is Lazy evaluated?**

   If you execute a bunch of program, it's not mandatory to evaluate immediately. Especially in Transformations, this Laziness is trigger.

7. **What is Catchable?**

   keep all the data inmemory for computation, rather than going to the disk. So Spark can catch the data 100 times faster than Hadoop.

8. **What is Spark engine responsibility?**

   Spark Engine is responsible for scheduling, distributing, and monitoring the data application across the cluster.

9. **What are common Spark Ecosystems?**

   Spark SQL(Shark) for SQL developers,

   Spark Streaming for streaming data,

   MLLib for machine learning algorithms,

   GraphX for Graph computation,

SparkR to run R on Spark engine,

BlinkDB enabling interactive queries over massive data are common Spark ecosystems. GraphX, SparkR and BlinkDB are in incubation stage.

10. What is Partitions?

partition is a logical division of the data, this idea derived from Mapreduce

(split). Logical data specifically derived to process the data. Small chunks of data also it can support scalability and speed up the process. Input data,

intermediate data and output data everything is Partitioned RDD.

11. How spark partition the data?

Spark use mapreduce API to do the partition the data. In Input format we can create number of partitions. By default HDFS block size is partition size (for best performance), but its' possible to change partition size like Split.

12. How Spark store the data?

Spark is a processing engine, there is no storage engine. It can retrieve data from any storage engine like HDFS, S3 and other data resources.

Is it mandatory to start Hadoop to run spark application?

No not mandatory, but there is no separate storage in Spark, so it use local file system to store the data. You can load data from local system and process it, Hadoop or HDFS is not mandatory to run spark application.

13. What is SparkContext?

When a programmer creates a RDDs, SparkContext connect to the Spark cluster to create a new SparkContext object.

SparkContext tell spark how to access the cluster. SparkConf is key factor to create programmer application.

14. What is SparkCore functionalities?

SparkCore is a base engine of apache spark framework. Memory management, fault tolarance, scheduling and monitoring jobs, interacting with store systems are primary functionalities of Spark.

15. How SparkSQL is different from HQL and SQL?

SparkSQL is a special component on the sparkCore engine that support SQL and HiveQueryLanguage without changing any syntax. It's possible to join SQL table and HQL table.

16. When did we use Spark Streaming?

Spark Streaming is a real time processing of streaming data API. Spark streaming gather streaming data from different resources like web server log

files, social media data, stock market data or Hadoop ecosystems like Flume, and Kafka.

17. How Spark Streaming API works?

Programmer set a specific time in the configuration, with in this time how much data gets into the Spark, that data separates as a batch. The input stream (DStream) goes into spark streaming. Framework breaks up into small chunks called batches, then feeds into the spark engine for processing. Spark Streaming API passes that batches to the core engine. Core engine can generate the final results in the form of streaming batches. The output is also in the form of batches. It can allows streaming data and batch data for processing.

18. What is Spark MLlib?

Mahout is a machine learning library for Hadoop, similarly MLlib is a Spark library. MetLib provides different algorithms, that algorithms scale out on the cluster for data processing. Most of the data scientists use this MLlib library.

19. What is GraphX?

GraphX is a Spark API for manipulating Graphs and collections. It unifies ETL, other analysis, and iterative graph computation. It's fastest graph system, provides fault tolerance and ease of use without special skills.

20. What is File System API?

FS API can read data from different storage devices like HDFS, S3 or local FileSystem. Spark uses FS API to read data from different storage engines.

21. Why Partitions are immutable?

Every transformation generate new partition. Partitions uses HDFS API so that partition is immutable, distributed and fault tolerance. Partition also aware of data locality.

22. What is Transformation in spark?

Spark provides two special operations on RDDs called transformations and Actions. Transformation follow lazy operation and temporary hold the data until unless called the Action. Each transformation generate/return new RDD.

Example of transformations: Map, flatMap, groupByKey, reduceByKey, filter, cogroup, join, sortByKey, Union, distinct, sample are common spark transformations.

23. What is Action in Spark?

Actions is RDD's operation, that value return back to the spar driver programs, which kick off a job to execute on a cluster. Transformation's output is input of Actions. reduce, collect, takeSample, take, first, saveAsTextfile,

saveAsSequenceFile, countByKey, foreach are common actions in Apache spark.

24. What is RDD Lineage?

Lineage is a RDD process to reconstruct lost partitions. Spark not replicate the data in memory, if data lost, Rdd use linege to rebuild lost data.Each RDD remembers how the RDD build from other datasets.

25. What is Map and flatMap in Spark?

Map is a specific line or row to process that data. In FlatMap each input item can be mapped to multiple output items (so function should return a Seq rather than a single item). So most frequently used to return Array elements.

26. What are broadcast variables?

Broadcast variables let programmer keep a read only variable cached on each machine, rather than shipping a copy of it with tasks. Spark supports 2 types of shared variables called broadcast variables (like Hadoop distributed cache) and accumulators (like Hadoop counters). Broadcast variables stored as Array Buffers, which sends read only values to work nodes.

27. What are Accumulators in Spark?

Spark offline debuggers are called accumulators. Spark accumulators are similar to Hadoop counters, to count the number of events and what's happening during job you can use accumulators. Only the driver program can read an accumulator value, not the tasks.

28. How RDD persist the data?

There are two methods to persist the data, such as persist() to persist permanently and cache() to persist temporarily in the memory. Different storage level options there such as MEMORY_ONLY, MEMORY_AND_DISK, DISK_ONLY and many more. Both persist() and cache() uses different options depends on the task.

29. When do you use apache spark? OR What are the benefits of Spark over Mapreduce?

a. Spark is really fast. As per their claims, it runs programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk. It aptly utilizes RAM to produce the faster results.

b. In map reduce paradigm, you write many Map-reduce tasks and then tie these tasks together using Oozie/shell script. This mechanism is very time consuming and the map-reduce task have heavy latency.

c. And quite often, translating the output out of one MR job into the input of another MR job might require writing another code because Oozie may not suffice.

d. In Spark, you can basically do everything using single application / console (pyspark or scala console) and get the results immediately. Switching between 'Running something on cluster' and 'doing something locally' is fairly easy and straightforward. This also leads to less context switch of the developer and more productivity.

e. Spark kind of equals to MapReduce and Oozie put together.

30. Is there are point of learning Mapreduce, then?

A: Yes. For the following reason:

1. Mapreduce is a paradigm used by many big data tools including Spark. So, understanding the MapReduce paradigm and how to convert a problem into series of MR tasks is very important.
2. When the data grows beyond what can fit into the memory on your cluster, the Hadoop Map-Reduce paradigm is still very relevant.
3. Almost, every other tool such as Hive or Pig converts its query into MapReduce phases. If you understand the Mapreduce then you will be able to optimize your queries better.

31. When running Spark on Yarn, do I need to install Spark on all nodes of Yarn Cluster?

Since spark runs on top of Yarn, it utilizes yarn for the execution of its commands over the cluster's nodes.

So, you just have to install Spark on one node.

32. What are the downsides of Spark?

Spark utilizes the memory. The developer has to be careful. A casual developer might make following mistakes:

1. She may end up running everything on the local node instead of distributing work over to the cluster.
2. She might hit some webservice too many times by the way of using multiple clusters.

The first problem is well tackled by Hadoop Map reduce paradigm as it ensures that the data your code is churning is fairly small a point of time thus you can make a mistake of trying to handle whole data on a single node.

The second mistake is possible in Map-Reduce too. While writing Map-Reduce, user may hit a service from inside of map() or reduce() too many times. This overloading of service is also possible while using Spark.

33. What is a RDD?

The full form of RDD is resilience distributed dataset. It is a representation of data located on a network which is

1. Immutable - You can operate on the rdd to produce another rdd but you can't alter it.
2. Partitioned / Parallel - The data located on RDD is operated in parallel. Any operation on RDD is done using multiple nodes.
3. Resilience - If one of the node hosting the partition fails, another nodes takes its data.

RDD provides two kinds of operations: Transformations and Actions.

34. What is Transformations?

The transformations are the functions that are applied on an RDD (resilient distributed data set). The transformation results in another RDD. A transformation is not executed until an action follows.

The example of transformations are:

1. map() - applies the function passed to it on each element of RDD resulting in a new RDD.
2. filter() - creates a new RDD by picking the elements from the current RDD which pass the function argument.

35. What are Actions?

An action brings back the data from the RDD to the local machine. Execution of an action results in all the previously created transformation. The example of actions are:

1. reduce() - executes the function passed again and again until only one value is left. The function should take two argument and return one value.

take() - take all the values back to the local node form RDD.

36. Say I have a huge list of numbers in RDD(say myrdd). And I wrote the following code to compute average:

```
def myAvg(x, y):
  return (x+y)/2.0;
avg = myrdd.reduce(myAvg);
```

What is wrong with it? And How would you correct it?

The average function is not commutative and associative;

I would simply sum it and then divide by count.

```
def sum(x, y):
   return x+y;
total = myrdd.reduce(sum);
avg = total / myrdd.count();
```

The only problem with the above code is that the total might become very big thus over flow. So, I would rather divide each number by count and then sum in the following way.

```
cnt = myrdd.count();
def devideByCnd(x):
   return x/cnt;
myrdd1 = myrdd.map(devideByCnd);
avg = myrdd.reduce(sum);
```

37. **Say I have a huge list of numbers in a file in HDFS. Each line has one number.And I want to compute the square root of sum of squares of these numbers. How would you do it?**

We would first load the file as RDD from HDFS on spark

numsAsText = sc.textFile("hdfs://hadoop1.knowbigdata.com/user/student/sgiri/mynumbersfile.txt");

```
# Define the function to compute the squares
def toSqInt(str):
   v = int(str);
   return v*v;
#Run the function on spark rdd as transformation
nums = numsAsText.map(toSqInt);

#Run the summation as reduce action
total = nums.reduce(sum)
```

```
#finally compute the square root. For which we need to import math.
import math;
print math.sqrt(total);
```

38. Is the following approach correct? Is the *sqrtOfSumOfSq* a valid reducer?

```
numsAsText =sc.textFile("hdfs://hadoop1.knowbigdata.com/user/student/sgiri/mynumbersfile.txt");
def toInt(str):
   return int(str);
nums = numsAsText.map(toInt);
def sqrtOfSumOfSq(x, y):
   return math.sqrt(x*x+y*y);
total = nums.reduce(sum)
import math;
print math.sqrt(total);
```

Yes. The approach is correct and ***sqrtOfSumOfSq*** is a valid reducer.

39. Could you compare the pros and cons of the your approach (in Question 2 above) and my approach (in Question 3 above)?

You are doing the square and square root as part of reduce action while I am squaring in map() and summing in reduce in my approach.

My approach will be faster because in your case the reducer code is heavy as it is calling math.sqrt() and reducer code is generally executed approximately **n-1** times the spark RDD.
The only downside of my approach is that there is a huge chance of integer overflow because I am computing the sum of squares as part of map.

40. If you have to compute the total counts of each of the unique words on spark, how would you go about it?

```
#This will load the bigtextfile.txt as RDD in the spark
```

```
lines =
sc.textFile("hdfs://hadoop1.knowbigdata.com/user/student/sgiri/bigtextfile.txt");



#define a function that can break each line into words


def toWords(line):


     return line.split();



# Run the toWords function on each element of RDD on spark as flatMap
transformation.


# We are going to flatMap instead of map because our function is returning
multiple values.


words = lines.flatMap(toWords);



# Convert each word into (key, value) pair. Her key will be the word itself and
value will be 1.


def toTuple(word):


    return (word, 1);



wordsTuple = words.map(toTuple);



# Now we can easily do the reduceByKey() action.
```

Tutortek
#1569, Opp Navami Hotel,
East End Main Road,
Jayanagar 9th Block, Bangalore,560069.
Website: www.tutortek.com
Call Us:+91 9886635020 / 9686511166
Email: enquiry@tutortek.com

```
def sum(x, y):

    return x+y;


counts = wordsTuple.reduceByKey(sum)


# Now, print

counts.collect()
```

41. **In a very huge text file, you want to just check if a particular keyword exists. How would you do this using Spark?**

```
lines = sc.textFile("hdfs://hadoop1.knowbigdata.com/user/student/sgiri/bigtextfile.txt");
def isFound(line):
   if line.find(“mykeyword”) > -1:
                return 1;
   return 0;
foundBits = lines.map(isFound);
sum = foundBits.reduce(sum);
if sum > 0:
   print “FOUND”;
else:
   print “NOT FOUND”;
```

42. **Can you improve the performance of this code in previous answer?**

    Yes. The search is not stopping even after the word we are looking for has been found. Our map code would keep executing on all the nodes which is very inefficient.

We could utilize accumulators to report whether the word has been found or not and then stop the job. Something on these line:

```
import thread, threading
from time import sleep
result = "Not Set"
lock = threading.Lock()
accum = sc.accumulator(0)
def map_func(line):
  #introduce delay to emulate the slowness
  sleep(1);
  if line.find("Adventures") > -1:
            accum.add(1);
            return 1;
  return 0;
def start_job():
  global result
  try:
            sc.setJobGroup("job_to_cancel", "some description")
            lines =
sc.textFile("hdfs://hadoop1.knowbigdata.com/user/student/sgiri/wordcount/inp
ut/big.txt");
            result = lines.map(map_func);
            result.take(1);
  except Exception as e:
            result = "Cancelled"
  lock.release()
def stop_job():
  while accum.value < 3 :
            sleep(1);
  sc.cancelJobGroup("job_to_cancel")
supress = lock.acquire()
supress = thread.start_new_thread(start_job, tuple())
```

```
supress = thread.start_new_thread(stop_job, tuple())
supress = lock.acquire()
```

43. What is PageRank?

A unique feature and algorithm in graph, PageRank is the measure of each vertex in the graph. For instance, an edge from u to v represents endorsement of v's importance by u. In simple terms, if a user at Instagram is followed massively, it will rank high on that platform.

44. What do you understand by worker node?

Worker node refers to any node that can run the application code in a cluster.

45. Name types of Cluster Managers in Spark.

The Spark framework supports three major types of Cluster Managers:

• Standalone: a basic manager to set up a cluster

• Apache Mesos: generalized/commonly-used cluster manager, also runs Hadoop MapReduce and other applications

• Yarn: responsible for resource management in Hadoop

46. What is Spark Executor?

When SparkContext connect to a cluster manager, it acquires an Executor on nodes in the cluster. Executors are Spark processes that run computations and store the data on the worker node. The final tasks by SparkContext are transferred to executors for their execution.

47. How to create RDD?

Spark provides two methods to create RDD:

• By parallelizing a collection in your Driver program. This makes use of SparkContext's 'parallelize' method

val IntellipaatData = Array(2,4,6,8,10)

val distIntellipaatData = sc.parallelize(IntellipaatData)

• By loading an external dataset from external storage like HDFS, HBase, shared file system

48. Illustrate some demerits of using Spark.

Since Spark utilizes more storage space compared to Hadoop and MapReduce, there may arise certain problems. Developers need to be careful while running their applications in Spark. Instead of running everything on a single node, the work must be distributed over multiple clusters.

49. List the functions of Spark SQL.

Spark SQL is capable of:

• Loading data from a variety of structured sources

• Querying data using SQL statements, both inside a Spark program and from external tools that connect to Spark SQL through standard database connectors (JDBC/ODBC). For instance, using business intelligence tools like Tableau

• Providing rich integration between SQL and regular Python/Java/Scala code, including the ability to join RDDs and SQL tables, expose custom functions in SQL, and more

50. What is Yarn?

Similar to Hadoop, Yarn is one of the key features in Spark, providing a central and resource management platform to deliver scalable operations across the cluster . Running Spark on Yarn necessitates a binary distribution of Spar as built on Yarn support.

51. What file systems Spark support?

• Hadoop Distributed File System (HDFS)

• Local File system

• S3

52. What is a Parquet file?

Parquet is a columnar format file supported by many other data processing systems. Spark SQL performs both read and write operations with Parquet file and consider it be one of the best big data analytics format so far.

53. What is Spark SQL?

SQL Spark, better known as Shark is a novel module introduced in Spark to work with structured data and perform structured data processing. Through this module, Spark executes relational SQL queries on the data. The core of the component supports an altogether different RDD called SchemaRDD, composed of rows objects and schema objects defining data type of each column in the row. It is similar to a table in relational database.

54. What does MLlib do?

MLlib is scalable machine learning library provided by Spark. It aims at making machine learning easy and scalable with common learning algorithms and use cases like clustering, regression filtering, dimensional reduction, and alike.

55. Define Spark Streaming.

Spark supports stream processing – an extension to the Spark API , allowing stream processing of live data streams. The data from different sources like

Flume, HDFS is streamed and finally processed to file systems, live dashboards and databases. It is similar to batch processing as the input data is divided into streams like batches.

56. What is Hive on Spark?

Hive contains significant support for Apache Spark, wherein Hive execution is configured to Spark:

hive> set spark.home=/location/to/sparkHome;

hive> set hive.execution.engine=spark;

Hive on Spark supports Spark on yarn mode by default.

57. What is Spark Driver?

Spark Driver is the program that runs on the master node of the machine and declares transformations and actions on data RDDs. In simple terms, driver in Spark creates SparkContext, connected to a given Spark Master.

The driver also delivers the RDD graphs to Master, where the standalone cluster manager runs.